REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA
FACULTE DES MATHEMATIQUES ET DE L'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

MEMOIRE de fin d'étude

Présenté pour l'obtention du diplôme de MASTER

Domaine : Mathématiques et Informatique

Filière : Informatique

Spécialité : Systèmes d'Informations Avancés

Par : MOUSSAOUI Rime

SUJET

Métaheuristiques appliquées au problème de fragmentation d'entrepôts de données : Etude comparative

Soutenu publiquement le : 01 /06 /2016 devant le jury composé de :

Dr. LAMICHE Chaabane	**Université de M'sila**	**Rapporteur**
N. Mouhoub	**Université de M'sila**	**Président**
A. Khettaf	**Université de M'sila**	**Examinateur**
……	**Université de M'sila**	**Examinateur**

Promotion : 2015 /20 16


بسم الله الرحمن الرحيم

Remerciements

Nous tenons à exprimer nos vifs remerciements à ALLAH qui nous

A donné la volonté et la patience afin de finir notre étude malgré les difficultés rencontrées.

Nous tenons à remercier par ce travail tous ceux qui nous ont aidés

De loin ou de près à réaliser ce mémoire de fin d'étude

Nous remercions ***Dr. LAMICHE chaabane*** *qui a bien voulu*

Accepter l'encadrement de ce mémoire et nous a apporté des conseils

Précieux durant toutes les étapes de ce travail.

Nous souhaitons également remercier les membres du Jury pour nous

Avoir fait l'honneur d'accepter de juger et d'évaluer notre travail.

Nous remercions tous nos enseignants pour toutes les Connaissances.

Tables des Matières

Chapitre 02 : Optimisation dans les entrepôts de données

Chapitre 03 : Résolution du problème de sélection de schéma de fragmentation optimale

Chapitre 04 : Réalisation et expérimentations

Liste Des Figures

Pmut=0.01).

Figure 4.14 : Coût du schéma optimal donné par AG et AG + Méthode descente Taille population=30, Pmut=0.01).

Figure 4.15 : Coût du schéma optimal donné par AG et AG + Méthode descente (Taille population=100, Pmut=0.01).

Liste Des Tableaux

Liste Des Abréviations

ED : Entrepôts de Données

OLTP: On-Line Transaction Processing

OLAP: On-Line Analytical Processing

SGBD : Système de Gestion de Base de Données

FASMI: Fast Analysis of Shared Multidimensional Information

ROLAP: Relational On-Line Analytical Processing

MOLAP: Multidimensional On-line Analytical Processing

HOLAP: Hybrid On-Line Analytical Processing

PSI : Problème de Sélection des Indexes

FH : Fragmentation Horizontale

AED : Administrateur d'Entrepôts de Données

AG : Algorithme Génétique

ESP : Ensemble des Simple Prédicats

EPCM : Ensemble des Prédicats Complet et Minimal

MU : Matrice Usage

POO : Programmation Orienté Object

Liste Des Algorithmes

Introduction générale

Introduction générale

Les entrepôts de données (data warehouse en anglais) c'est un outil décisionnel qui permet l'exploitation des données d'une organisation dans le but de faciliter la prise de décision. Pour cela les dirigeants des entreprises intégrant des données dites de production dans une base de données centralisée (entrepôts) ou elles sont agrégée, historisées et structurée de manière à en permettre le regroupement, nettoyage et intégration des données et aussi établissement des requêtes, rapports et des analyses et offre la possibilité d'extraire des connaissances (fouille de données).

Les entrepôts de données sont souvent modélisés par un schéma en étoile. Cette dernierre est caractérisé par une table de fait de très grand taille et un ensemble de tables de dimensions de plus petite taille.

Le table de fait contient qu'un ensemble de mesures collectées durant l'activité de l'organisation. Les tables de dimension contiennent des données qualitatives qui représentent des axes sur lesquels les mesures ont été collectées. Les requêtes de type OLAP définies sur un schéma en étoile (connues par requêtes de jointure en étoile) sont caractérisées par des opérations de sélection sur les tables de dimension, suivies de jointures avec la table des faits. Aucune jointure n'existe entre les tables de dimension. Toute jointure doit passer par la table des faits, ce qui rend le coût d'exécution de ces requêtes très important Sans technique d'optimisation, leur exécution peut prendre des heures, voire des jours. [1]

Dans le domaine optimisation dans entrepôts de données il existe beaucoup des techniques. Les techniques redondante(les vues matérialisé, les indexes, fragmentation verticale) et techniques non redondante (fragmentation horizontale). Fragmentation horizontale consiste à partitionner la table de fait en un ensemble des fragments en utilisant des fragments des tables de dimensions.

L'objectif de notre étude est l'adaptation du metaheuristique à population (un algorithme génétique) et méthode descente pour obtention un schéma optimal des tables de dimensions permettant de nous donner un schéma de fragmentation optimal d'entrepôts.

Nous avons organisé notre mémoire de la façon suivant :

Dans le chapitre 1, nous avons présenté les éléments fondamentaux d'entrepôts de données. Le chapitre 2 est réservé à la fragmentation des entrepôts de données. La méthodologie proposée

pour la résolution du problème de sélection de schéma de fragmentation optimale est largement discutée dans le chapitre 3. Enfin la réalisation de notre application et nos et expérimentations. sont présentées dans le chapitre 4. Le mémoire se termine par une conclusion où nous avons adressé quelques perspectives.

Chapitre 01 : Eléments fondamentaux des entrepôts de données

1. Introduction

Pour faire face aux nouveaux enjeux, l'entreprise doit collecter, traiter, analyser les informations de son environnement pour anticiper. Mais cette information produite par l'entreprise est surabondante, non organisée et éparpillée dans de multiples systèmes opérationnels hétérogènes et peut provenir de toutes les places de marchés (mondialisation des échanges).Il devient fondamental de rassembler et d'homogénéiser les données afin de permettre l'analyse des indicateurs pertinents pour faciliter la prise de décisions. L'objet de l'entrepôt de données est de définir et d'intégrer une architecture qui serve de fondation aux applications décisionnelle [2].

Le concept d'entrepôt de données a été formalisé pour la première fois en 1990 par Bill Inmon. Il s'agissait de constituer une base de données orientée sujet, intégrée et contenant des informations historisées, non volatiles et exclusivement destinées aux processus d'aide à la décision. En effet, la simple logique de production (produire pour répondre à une demande) ne suffit plus pour pérenniser l'activité d'une entreprise. Elle est un système ouvert sur son environnement au cœur des systèmes d'informations confrontée à des phénomènes économiques et sociaux lourd de conséquences.

Dans ce chapitre, nous donnerons une vue global sur :

- les concepts fondamentaux relatifs aux entrepôts de données.
- les techniques d'optimisations des requêtes dans ces dernières.

2. Les entrepôts de données

2.1 Définition d'un entrepôt de données

Un entrepôt de données (ED) est une base de données consolidée, provenant toutes ses informations à partir des bases de données de production.

Généralement, un ED est utilisé par les applications décisionnelles afin d'analyser la situation de l'entreprise à court terme ou à long terme et de faire un suivi de toutes les activités stratégiques de 1' entreprise. Celles-ci sont visualisées graphiquement dans un tableau de bord permettant aux gestionnaires de prendre les décisions en temps opportuns [3].

La définition donnée par BILL Inmon est comme suit : « L'entrepôt de données est une collection de données orientés sujet, intégrées, non volatiles et historiées, organisées pour support d'un processus d'aide à la décision ».

Les caractéristiques d'un entrepôt de données :

- **Orientées sujet :** un ED rassemble et organise des données associées aux différentes structures fonctionnelles de l'entreprise, pertinentes pour un sujet ou thème et nécessaire aux besoins d'analyse
- **Intégrées :** les données résultent de l'intégration de données provenant de différentes sources pouvant être hétérogènes
- **Historisées** : les données d'un ED représentent l'activité d'une entreprise durant une certaine période (plusieurs années) permettant de d'analyser les variations d'une donnée dans le temps
- **Non-volatiles :** les données de l'ED sont essentiellement utilisées en interrogation (consultation) et ne peuvent pas être modifiées (sauf certain cas de rafraîchissement) [4].

2.2 Architecture d'un entrepôt de données

L'entrepôt de données joue un rôle stratégique dans la vie d'une entreprise. Il stocke des données pertinentes aux besoins de prise de décision en provenance des systèmes opérationnels de l'entreprise et d'autres sources externes. A la différence d'une base de données classique supportant des requêtes transactionnelles de type OLTP (On-Line Transaction Processing), un entrepôt de données est conçu pour supporter des requêtes de type OLAP (On-Line Analytical Processing). L'interrogation est l'opération la plus utilisée dans le contexte d'entrepôt de données où la mise à jour consiste seulement à alimenter l'entrepôt.

Le processus de construction d'un entrepôt de données est composé de trois principales phases : (1) extraction des données à partir des différentes sources, (2) organisation et intégration des données dans l'entrepôt et (3) accès aux données intégrées et analyse de ces dernières dans une forme efficace et flexible (voir figure 1.1). 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.

Les métadonnées contiennent des informations utiles sur la création, l'utilisation et la gestion de l'entrepôt. Durant la troisième phase, un serveur OLAP se charge de présenter les informations demandées par les utilisateurs sous plusieurs formes : tableaux, rapports, statistiques, etc [5].

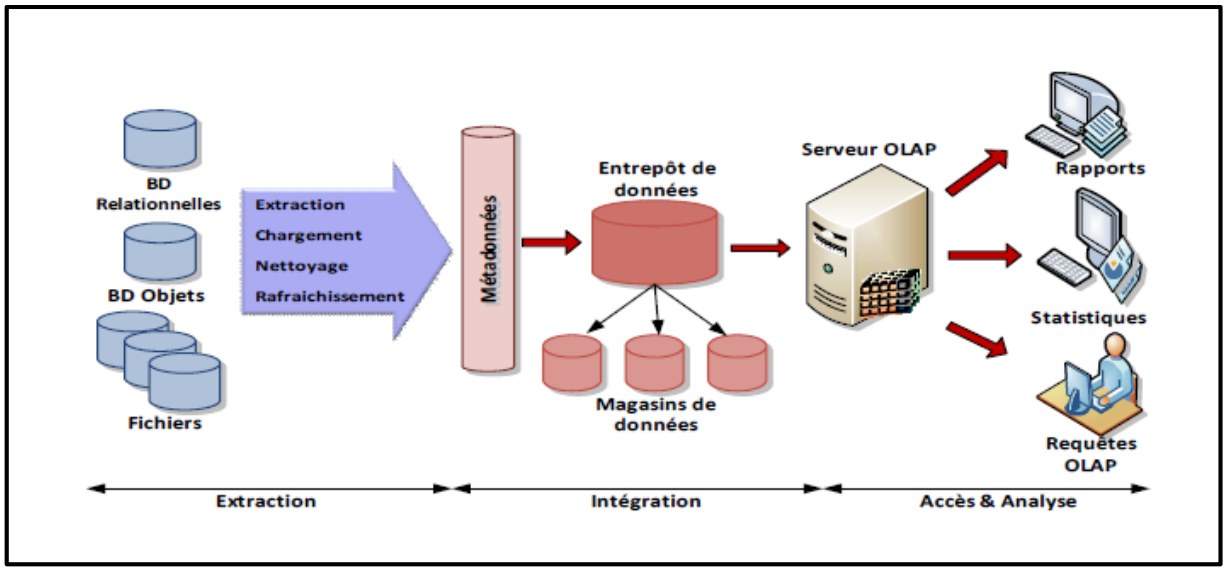


Figure 1.1 : Architecture des entrepôts de données [5].

2.3 Les sources et les types de données dans ED

❖ **Les sources :** Les données de l'entrepôt sont extraites de diverses sources souvent réparties et hétérogène, et qui doivent être transformées avant leur stockage dans l'entrepôt. Nous avons deux types de sources des données : interne et externes

- **Interne :** la plupart des données sont saisies à partir des différents systèmes de production qui rassemblent les divers SGBD opérationnels, ainsi que les anciens systèmes de production qui contiennent des données encore exploités par l'entreprise.
- **Externes :** ils représentent des données externes à l'entreprise et qui sont souvent achetée. Par exemple, les sources de données démographiques.

❖ **Les types**

Il existe plusieurs types de données dans un entrepôt, qui correspondent à diverses utilisations, comme :

- **Données de détail courantes :** Ce sont l'ensemble des données quotidiennes et plus couramment utilisée. Ces données sont généralement stockées sur le disque pour avoir un accès rapide.
- **Données de détail anciennes :** Ce sont des données quotidiennes concernant des évènements passés. Ces données utilisé pour arriver à l'analyse des tendances ou des requêtes prévisionnelles et elles sont plus rarement utilisées que les précédentes, souvent stockées sur des mémoires d'archives.
- **Données résumés ou agrégées :** Ce sont des données moins détaillées que les deux premières et elles permettent de réduire le volume des données à stocker. Le type de

données, en fonction de leur niveau de détail permet de les classifier comme des données légèrement ou fortement résumées.

- **Les métadonnées :** ce sont des données essentielles pour parvenir à une exploitation efficace du contenu d'un entrepôt. Elles représentent des informations nécessaires à l'accès et l'exploitation des données dans l'entrepôt [6].

2.4 Systèmes OLTP versus systèmes OLAP

Les bases de données sont utilisées dans les entreprises pour gérer les importants volumes d'informations contenus dans leurs systèmes opérationnels. Ces données sont gérées selon des processus transactionnels en ligne (OLTP : "On-Line Transactionnel Processing"). Qui se caractérisent de la manière suivante :

- Ils sont nombreux au sein d'une entreprise,
- Ils concernent essentiellement la mise à jour des données,
- Ils traitent un nombre d'enregistrements réduit,
- Ils sont définis et exécutés par de nombreux utilisateurs.

L'exploitation de l'information contenue dans ces systèmes opérationnels est devenue une préoccupation essentielle pour les dirigeants des entreprises qui désirent améliorer leur prise de décision par une meilleure connaissance de leur propre activité, de celle de la concurrence, des employés, des clients et des fournisseurs. Les entreprises sont donc à la recherche de systèmes supportant efficacement les applications d'aide à la décision. Ces applications décisionnelles utilisent des processus d'analyse en ligne de données (OLAP : "On-Line Analytical Processing»). OLAP caractérise par :

- ils sont peu nombreux, mais leurs données et traitements sont complexes.
- il s'agit uniquement de traitements semi-automatiques visant à interroger, visualisé et synthétiser les données.
- ils concernent un nombre d'enregistrements importants aux structures hétérogènes.
- ils sont définis et mis en œuvre par un nombre réduit d'utilisateurs qui sont les décideurs [7].

2.5 Système transactionnel et système décisionnel

Les SGBD ont été créés pour gérer de grands volumes d'information contenus dans les différents systèmes opérationnels qui appartiennent à l'entreprise .Ces données sont manipulées en utilisant des processus transactionnels en ligne [8].

Parallèlement à l'exploitation de l'information contenue dans ces systèmes opérationnels, les dirigeants des entreprises ont besoin d'avoir une vision globale concernant toute cette

information pour faire des calculs prévisionnels, des statistiques ou pour établir des stratégies de développement et d'analyses des tendances.

Le tableau (1.1) compare les caractéristiques des systèmes transactionnels et décisionnels par rapport aux données et aux utilisateurs [6].

	Système transactionnel	**Système décisionnel**
Données	Exhaustives Courantes Dynamiques Orienté applications	Résumées Historiques Statiques Orienté sujets
Utilisateur	Nombreux Variés Concurrents Mise à jour et interrogations Requêtes prédéfinies Réponses immédiates Accès à peu d'information	Peu nombreux Uniquement les décideurs Non concurrents Interrogations requêtes complexes Réponses moins rapide Accès à de nombreuses informations

Tableau 1.1 : Comparaison des systèmes.

2.6 Modélisation multidimensionnelles

2.6.1 Concept de Faits

Le fait modélise le sujet de l'analyse. Un fait est formé de mesures correspondant aux informations de l'activité analysée.

Les mesures d'un fait sont numériques et généralement valorisées de manière continue. Les mesures sont numériques pour permettre de résumer un grand nombre d'enregistrements en quelques enregistrements (on peut les additionner, les dénombrer ou bien calculer le minimum, le maximum ou la moyenne). Les mesures sont valorisées de façon continue car il est important de ne pas valoriser le fait avec des valeurs nulles. Elles sont aussi souvent additives ou semi-additives afin de pouvoir les combiner au moyen d'opérateurs arithmétiques [9].

2.6.2 Concept de dimension

Une dimension modélise une perspective de l'analyse. Une dimension se compose de paramètres correspondant aux informations faisant varier les mesures de l'activité.

Les dimensions servent à enregistrer les valeurs pour lesquelles sont analysées les mesures de l'activité. Une dimension est généralement formée de paramètres (ou attributs) textuels et discrets. Les paramètres textuels sont utilisés pour restreindre la portée des requêtes afin de limiter la taille des réponses. Les paramètres sont discrets, c'est à dire que les valeurs possibles sont bien déterminées et sont des descripteurs constants [9].

2.6.3 Schémas relationnels

Les schémas relationnels adaptés aux besoins du modèle multidimensionnel possèdent certains avantages par rapport aux schémas en troisième forme normale. L'étoile, le flocon et la constellation autorisent l'expression des mesures, des dimensions et des hiérarchies, d'une manière simple qui permet de les distinguer clairement [5].

2.6.3.1 Schémas en étoile

Dans ce type de schéma, les mesures sont représentées par une table de faits et chaque dimension par une table de dimensions. La table des faits référence 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 des faits figurent les tables de dimensions qui regroupent les caractéristiques des dimensions. La table des faits est normalisées et peut atteindre une taille importante par rapport au nombre de n-uplets. Les tables de dimension sont généralement dé normalisées afin de minimiser le nombre de jointures n´nécessaires pour ´évaluer une requête. Ce schéma est largement utilisé dans les applications industrielles. Les requêtes typiques de ce schéma sont appelées les requêtes de jointure en étoile (star-join queries) qui ont les caractéristiques suivantes :

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

La syntaxe générale de ces requêtes est la suivante :

SELECT <Liste de projection> <Liste d'agrégation>

FROM <Nom de la table des faits> <Liste de noms de tables de dimension>

WHERE <Liste de prédicats de sélection & jointure>

GROUP BY <Liste des attributs de tables de dimension> [10].

2.6.3.2 Schémas en flocon de neige

Le schéma en étoile ne reflète pas les hiérarchies associées à une dimension [11] .Il exige que les informations complètes associées à une hiérarchie de dimension soient représentées

dans une seule table, même lorsque les différents niveaux de la hiérarchie ont des propriétés différentes. Pour résoudre ce problème, le schéma en flocon de neige a été proposé. Ce dernier est une extension du schéma en étoile. Il consiste à garder la même table des faits et `a ´éclater les tables de dimensions afin de permettre une représentation plus explicite de la hiérarchie. Cet éclatement peut être vu comme une normalisation des tables de dimensions [11]. Contrairement au schéma en étoile, le schéma en flocon de neige capture les hiérarchies entre les attributs [10].

2.6.3.3 Schémas en constellation

Le schéma en constellation représente le regroupement de plusieurs schémas en étoiles qui possèdent des dimensions communes. Il est employé lorsque les faits analysés ne sont pas tous déterminés par les mêmes dimensions [12].

2.6.4 Schémas multidimensionnel (cubes)

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 d´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 [10].

Le cube représente le concept central du modèle multidimensionnel, lequel est constitué des éléments appelés cellules qui peuvent contenir une ou plusieurs mesures.

La localisation de la cellule est faite à travers les axes, qui correspondent chacun à une dimension. La dimension est composée de membres qui représentent les différentes valeurs. En reprenant une partie du schéma en étoile de la figure 1.1, nous pouvons construire le schéma multidimensionnel suivant [7].

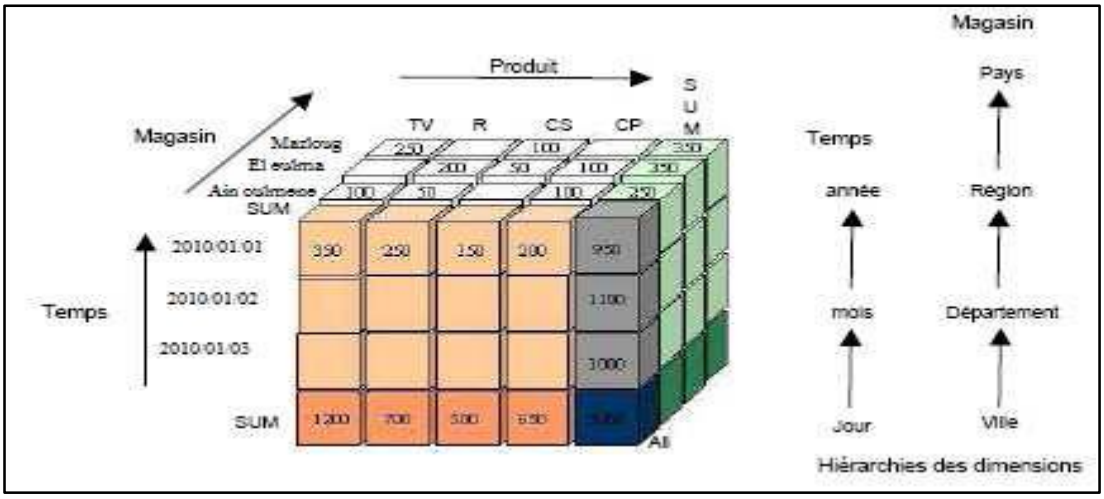


Figure 1.2 : Exemple de schéma multidimensionnel.

La figure 1.2 présente un schéma multidimensionnel pour les ventes qui ont été réalisées dans les magasins pour les différents produits au cours d'un temps donné (jour). Par exemple, nous avons la quantité de 100 Téléviseurs vendus dans le magasin pendant le 1er janvier 2010[7].

2.7 Manipulation des données

2.7.1 Opérations classique

Ces opérations correspondent aux opérations relationnelles de manipulation des données :

- **La sélection :** Résulte en un sous-ensemble de données qui respecte certaines conditions d'appartenance. Nous pouvons avoir une sélection avec des conditions soit sur les mesures, soit sur les membres
- **La projection :** Résulte en un sous-ensemble des attributs d'une relation, qui sont soit des dimensions, soit des niveaux de granularité. Dans les systèmes décisionnels, les opérations de sélection et de projection sont appelées souvent "slice-and-dice".
- **La jointure :** Permet d'associer les données de relations différentes [7].

2.7.2 Opération sur la structure

Les opérations agissant sur cette structure multidimensionnelle de l'information sont motivées par l'aspect interactif de l'analyse en ligne de données, et le souci d'offrir des possibilités d'animation de la représentation. De plus, elles illustrent l'importance des liens entre la manipulation des données et la représentation du cube à l'écran.

Ces opérations sont regroupées sous le nom de restructuration. Tout cube obtenu par une opération de restructuration d'un cube initial contient tout ce qu'il faut pour régénérer le cube initial par restructuration réciproque. Ces opérations sont : pivot, swich, split, nest, push, et pull.

- **Pivot :** Cette opération consiste à faire effectuer à un cube une rotation autour d'un des trois axes passant par le centre de deux faces opposées, de manière à présenter un ensemble de faces différent.
- **Switch** : Cette opération consiste à inter changé la position des membres d'une dimension.
- **Split :** Elle consiste à présenter chaque tranche du cube, et à passer d'une représentation tridimensionnelle d'un cube à sa représentation sous la forme d'un ensemble de tables.

D'une manière générale, cette opération permet de réduire le nombre de dimensions d'une représentation. On notera que le nombre de tables résultant d'une opération split dépend des informations contenues dans le cube de d´épart et n'est pas connu à l' avance.

- **Nest :** Cette opération permet d'imbriquer des membres. L'un de ses intérêts est qu'elle permet de grouper sur une même représentation bidimensionnelle toutes les informations (mesures et membres) d'un cube, quel que soit le nombre de ses dimensions. L'opération réciproque, "unnest", reconstitue une dimension séparée à partir des membres imbriqués.
- **Push :** Cette opération consiste à combiner les membres d'une dimension aux mesures du cube, et donc de faire passer des membres comme contenus de cellules. L'opération réciproque appelée pull, permet de changer le statut de certaines mesures d'un cube en membres, et de constituer une nouvelle dimension pour la représentation du cube, à partir de ces nouveaux membres [10].

2.7.3 Opération sur la granularité

Les opérations agissant sur la granularité des données analysées, permettent de hiérarchiser la navigation entre les différents niveaux de détail d'une dimension. Dans la suite nous traitons les deux opérations de ce type :

- **Le forage vers le haut (drill-up ou roll-up)** : Permet de représenter les données du cube à un niveau plus haut de granularité en respectant la hiérarchie de la dimension. Nous utilisons une fonction d'agrégation (somme, moyenne,...), qui est paramétrée, pour indiquer la façon de calculer les données du niveau supérieur à partir de celles du niveau inférieur.
- **Le forage vers le bas (drill-down ou roll-down ou scale-down**) : Consiste à représenter les données du cube à un niveau de granularité inférieur, donc sous une forme plus détaillée.

Ces types d'opérations ont besoin d'informations non représentées dans un cube, pour augmenter ou affiner des données, à partir d'une représentation initiale vers une représentation de granularité différente. Le forage vers le haut a besoin de connaître la fonction d'agrégation utilisée tandis que le forage vers le bas nécessite de connaître les données au niveau inférieur [7].

2.8 Serveur OLAP (on-line Analytical Processing)

Les données opérationnelles constituent la source principale d'un système d'information décisionnel. Les systèmes décisionnels complets reposent sur la technologie OLAP, conçue pour répondre aux besoins d'analyse des applications de gestion.

L'acronyme FASMI (Fast Analysis of Shared Multidimensional Information) permet de résumer la définition des produits OLAP. Cette définition fut utilisée pour la première fois en 1995 et depuis aucune autre définition n'est plus proche pour résumer le terme OLAP [13].

- Fast : Le temps de réponse aux demandes des utilisateurs oscille entre 1 et 20 secondes. Les constructeurs utilisent des pré-calculs pour réduire les durées des requêtes.
- Analysis : Le système doit pouvoir faire face à toutes les logiques d'affaires et de statistiques, ainsi que fournir la possibilité aux utilisateurs de construire leurs calculs et leurs analyses sans avoir à programmer. Pour cela, il y a des outils qui seront fournis par le constructeur.
- Shared : Le système doit créer un contexte où la confidentialité est préservée et doit gérer les cas où plusieurs utilisateurs ont des droits en écritures. Ce point constitue la plus grosse faiblesse des produits actuels.
- Multidimensional : C'est la caractéristique clé. Le système doit fournir des vues conceptuelles multidimensionnelles des données. Il doit supporter aussi les hiérarchies.
- Informations : L'ensemble des données et les informations nécessaires pour un produit OLAP [14].

2.8.1 ROLAP

Dans les systèmes relationnels OLAP, l'entrepôt de données utilise une base de données relationnelle. Le stockage et la gestion de données sont relationnels. Toutefois, le modèle relationnel requiert des extensions pour supporter les requêtes d'analyses multidimensionnelles du niveau d'application. Le moteur ROLAP traduit dynamiquement le modèle logique de données multidimensionnel M en modèle de stockage relationnel R (la plupart des outils requièrent que la donnée soit structurée en utilisant un schéma en étoile ou un schéma en flocon de neige). Techniquement, le moteur ROLAP exécute une transformation à partir d'une requête multidimensionnelle m contre M vers une requête relationnelle r contre R. L'efficacité du résultat de la requête est le facteur dominant pour la performance et le passage à l'échelle global du système. Ainsi, les stratégies d'optimisation représentent le point principal qui distingue les systèmes ROLAP existants [15].

La technologie ROLAP a deux avantages principaux : (1) elle permet la définition de données complexes et multidimensionnelles en utilisant un modèle relativement simple, et (2)

elle réduit le nombre de jointures à réaliser dans l'exécution d'une requête. Le désavantage est que le langage de requêtes tel qu'il existe, n'est pas assez puissant ou n'est pas assez flexible pour supporter de vraies capacités d'OLAP [16].

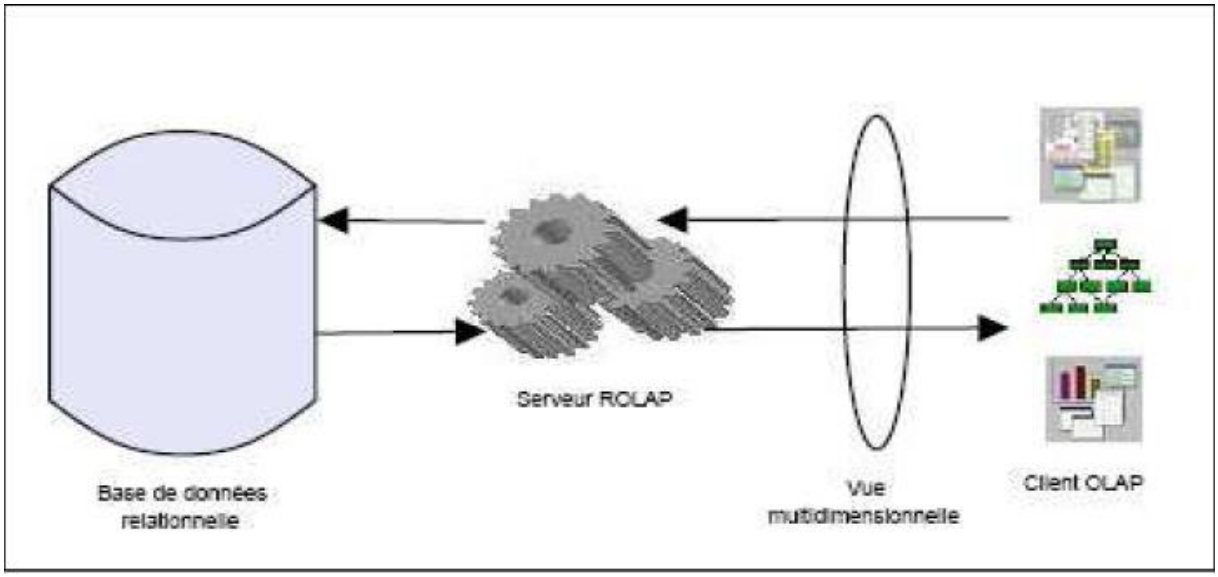


Figure 1.3 Architecture ROLAP [5].

2.8.2 MOLAP

Les systèmes multidimensionnels OLAP utilisent une base de données multidimensionnelle pour stocker les données de l'entrepôt et les applications analytiques sont construites directement sur elle. Dans cette architecture, le système de base de données multidimensionnel sert tant au niveau de stockage qu'au niveau de gestion des données. Les données des sources sont conformes au modèle multidimensionnel, et dans toutes les dimensions, les différentes agrégations sont pré calculées pour des raisons de performance [14].

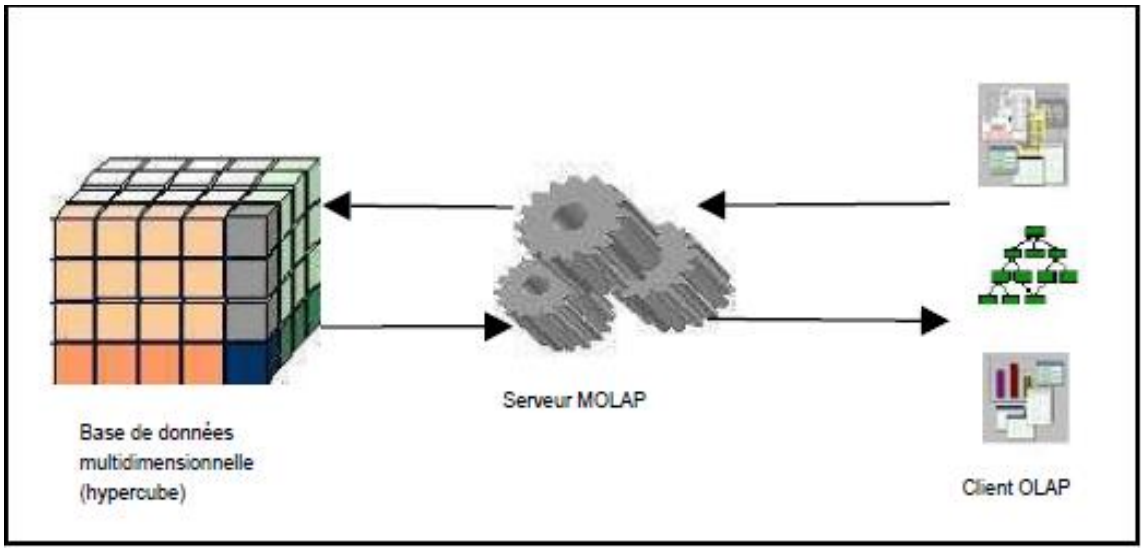


Figure 1.4 : Architecture MOLAP.

Les avantages des systèmes MOLAP sont basés sur les désavantages des systèmes ROLAP et elles représentent la raison de leur création. D'un côté, les requêtes MOLAP sont très puissantes et flexibles en termes du processus OLAP, tandis que, d'un autre côté, le modèle physique correspond plus étroitement au modèle multidimensionnel. Néanmoins, il existe des désavantages au modèle physique MOLAP [14].

2.8.3 HOLAP

Un système HOLAP est un système qui supporte et intègre un stockage des données multidimensionnel et relationnel d'une manière équivalente pour profiter des caractéristiques de correspondance et des techniques d'optimisation.

Ci-dessous, nous traitons une liste des caractéristiques principales qu'un système HOLAP doit fournir [17] :

- **La transparence du système** : Pour la localisation et l'accès aux données, sans connaître si elles sont stockées dans un SGBD relationnel ou dimensionnel. Pour la transparence de la fragmentation,... 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,...
- **Une réallocation automatique :** Toutes les caractéristiques traitées ci-dessus changent dans le temps. Ces changements peuvent provoquer la réorganisation de la distribution des données dans le système de stockage multidimensionnel et relationnel, pour assurer des performances optimales. Actuellement, la plupart des systèmes commerciaux utilisent une approche hybride. Cette approche permet de manipuler des informations de l'entrepôt de données avec un moteur ROLAP, tandis que pour la gestion des datamarts, ils utilisent l'approche multidimensionnelle [7].

3. Les Techniques d'optimisation

Le processus d'analyse de données s'appuie souvent sur des requêtes qui regroupent et filtrent des données de différentes formes. Compte tenu de la nature interactive des entrepôts de données, la nécessité d'avoir un temps de réponse rapide est un objectif critique pour l'utilisateur et/ou le décideur. Des délais trop longs sont inacceptables dans la plupart des systèmes décisionnels, puisqu'ils peuvent diminuer la productivité. L'exigence courante est un

temps de réponse ne dépassant pas quelques secondes ou quelques minutes. Sans technique d'optimisation de requêtes, l'interrogation d'un entrepôt de données serait complexe, et le traitement de ces requêtes pourrait prendre des heures. Les travaux concernant l'optimisation des performances des requêtes décisionnelles sont principalement basés sur des techniques héritées des bases de données relationnelles/objets. Nous pouvons les classées suivant deux catégories [1] :

- Techniques redondantes
- Techniques non redondantes

3.1 Les techniques redondantes

Les techniques redondantes sont des structures d'optimisation qui nécessitent un espace de stockage supplémentaire pour leur implémentation. Elles causent ainsi une redondance des données qui sont présentes à la fois dans les tables et dans les structures d'optimisation. [12]

3.1.1 Les indexes

Les techniques d'indexation utilisées dans les bases de données de type OLTP ne sont pas parfaitement adaptées aux environnements des entrepôts de données. En effet, la plupart des transactions OLTP accèdent à un nombre faible de n-uplets et les techniques utilisées sont adaptées à ce type de situation. Les requêtes décisionnelles dans le cas des entrepôts de données accèdent au contraire à un très grand nombre de n-uplets. Réutiliser les techniques des systèmes OLTP conduirait à des index avec un grand nombre de Un index peut être défini sur une seule ou sur plusieurs colonnes d'une relation, Ce type d'index est appelé mono-index. Il existe également des index définis sur deux relations comme les index de jointure qui sont appelés multi-index. Dans les entrepôts de données, les deux types d'index sont utilisés : index sur liste de valeur, index de projection (mono-index) et index de jointure en étoile (star join index) pour les index multi-index.

Table client

Nom	*Age*	….	*Sexe*
Driss	20	….	M
Layla	42	….	F
Jamal	21	….	M
Mounir	52	….	M
Alia	18	….	F
Karima	17	….	F
Hassan	36	….	M

Index de projection

Age
20
42
21
52
18
17
36

Figure 1.5 : Index de projection sur l'attribut Age.

Les requêtes complexes définies sur une base de données relationnelles demandent fréquemment des opérations de jointures entre plusieurs tables. L'opération de jointure est fondamentale dans les bases de données et est très coûteuse en termes de temps de calcul lorsque les tables concernées sont de grande taille, comme le cas des entrepôts de données. Plusieurs méthodes ont été proposées afin d'accélérer ces opérations. Ces méthodes incluent, les boucles imbriquées, le hachage et la fusion.

Un index de jointure matérialise les liens entre deux relations par le biais d'une table à deux colonnes contenant les Row ID (identifiant des n-uplets) des n-uplets joints deux à deux [18]. Cet index peut être vu comme une jointure pré-calculée. Créé à l'avance, il est implémenté par une relation d'arité 2 (figure 1.6). L'efficacité dépend du coefficient de sélectivité de jointure. Si la jointure a une forte sélectivité, l'index de jointure sera de taille faible et aura une grande efficacité. Ce genre d'index est parfaitement adapté pour les requêtes des systèmes OLTP parce qu'elles possèdent fréquemment des jointures entre deux tables.

Par contre, pour les entrepôts de données modélisés par un schéma en étoile, ces index sont limités. En effet, les requêtes décisionnelles définies sur un schéma en étoile possèdent plusieurs jointures (entre la table des faits et les tables de dimensions). Il faut dans ce cas subdiviser la requête en fonction des jointures. Or le nombre de jointures possibles est de l'ordre de N ! (N étant le nombre de tables à joindre).

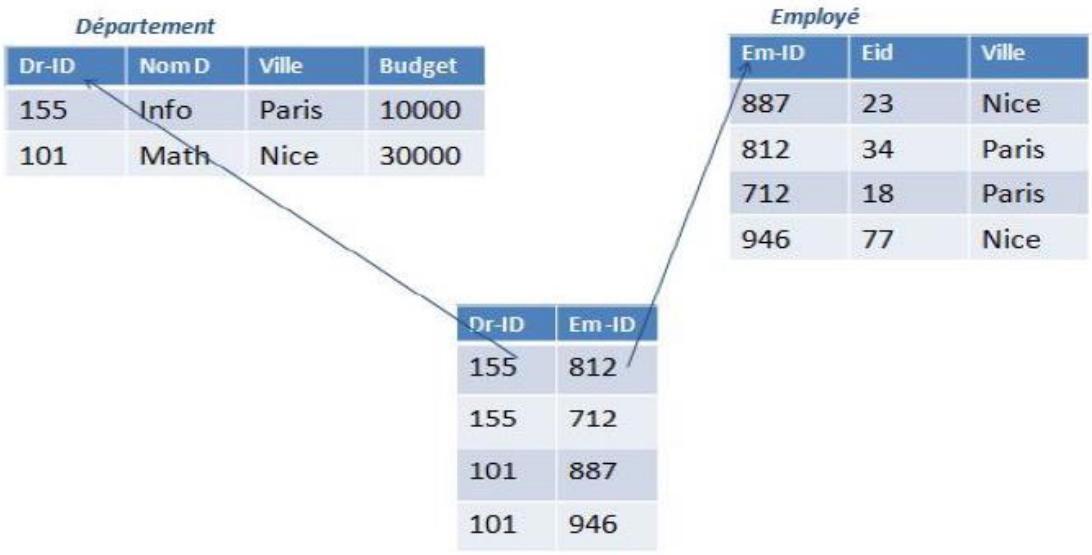


Figure 1.6 : Index de jointure pour l'équijointure Département.ville = Employé.ville

Afin de résoudre ce problème, un nouvel index appelé index de jointure en étoile (star join index) a été introduit [19], et adapté aux requêtes définies sur un schéma en étoile.

Un index de jointure en étoile peut contenir toute combinaison de clés étrangères de la table des faits. Il peut être n'importe quelle combinaison contenant la clé de la table des faits et une ou plusieurs clés primaires des tables de dimensions. Ce type d'index est dit complet s'il est construit en joignant toutes les tables de dimensions avec la table des faits. Un index de jointure partiel est construit en joignant certaines tables de dimensions avec la table des faits. En conséquence, l'index complet est bénéfique pour n'importe quelle requête posée sur le schéma en étoile. Il exige cependant beaucoup d'espace de stockage.

Deux remarques concernant l'index de jointure :

• Ce type d'index est exclusivement adapté aux schémas en étoile et par conséquent, il ne l'est pas pour les schémas en flocon.

• Il n'existe pas d'algorithme de sélection d'index de jointure en étoile pour un ensemble de requête. [1]

Au niveau de la conception physique, l'administration de l'entrepôt de données doit sélectionner des index afin d'accélérer l'exécution des requêtes. Ce problème est connu sous le nom de problème de sélection d'index (PSI), plusieurs solutions ont été proposées [20]

3.1.2 La fragmentation verticale

La fragmentation verticale permet de diviser une relation verticalement en plusieurs sous relations, chacune va comporter un ensemble d'attributs de la relation initiale. Pour chaque sous relation ou fragment vertical, l'attribut clé est ajouté afin de pouvoir reconstituer la relation initiale par opération de jointure. Cette technique est redondante car elle duplique la clé primaire de la table fragmentée, plus la duplication d'autres attributs selon le besoin de fragmentation. Elle nécessite donc un espace de stockage supplémentaire.

La fragmentation verticale accélère les opérations de projection. En effet, le traitement de la requête de projection nécessite uniquement le chargement des sous relation dont les attributs figurent dans la requête, il n'est pas nécessaire de charger toute la table. Par contre, son principal inconvénient est qu'elle nécessite des opérations de jointures très coûteuses si une requête accède à plusieurs fragments verticaux (Bellatreche, 2000).

Exemple 9 : Soit une table « Client ». Le résultat de fragmentation verticale de cette dimension est présenté dans la figure 1.7 [12]

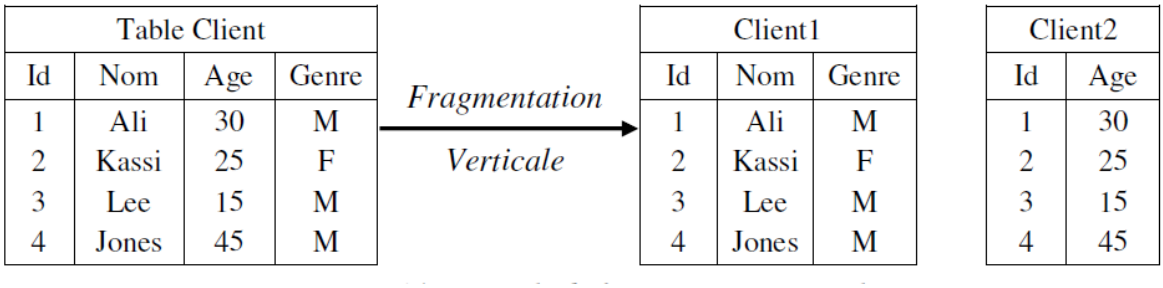


Figure 1.7 : Exemple de fragmentation verticale.

3.1.3 Les vues matérialisé

Une vue matérialisée est une table contenant les résultats d'une requête. Les vues améliorent l'exécution des requêtes en pré calculant les opérations les plus coûteuses comme la jointure et l'agrégation, et en stockant leurs résultats dans la base.

En conséquence, certaines requêtes nécessitent seulement l'accès aux vues matérialisées et sont ainsi exécutées plus rapidement. Les vues dans le contexte OLTP ont été largement utilisées pour répondre à plusieurs rôles : la sécurité, la confidentialité, l'intégrité référentielle, etc.

Les vues matérialisées peuvent être utilisées pour satisfaire plusieurs objectifs, comme l'amélioration de la performance des requêtes ou la fourniture des données dupliquées. [7]

3.2 Les techniques non redondantes

3.2.1 La fragmentation horizontale

La fragmentation horizontale (FH) se base sur le principe de réorganisation des données par la répartition des tuples d'une table en plusieurs sous-ensembles disjoints, appelés fragments horizontaux et pouvant être accédés séparément. Cette répartition est effectuée par une opération de restriction sur la table initiale et une simple opération d'union permet de reconstituer la relation initiale. Deux types de fragmentation horizontale existent : la FH primaire et la FH dérivée.

1. **La fragmentation horizontale primaire :** cette fragmentation horizontale est appelée primaire car elle permet de répartir les tuples d'une table suivent la conjonction de prédicats de sélection définis sur les attributs de la même table. Cette fragmentation est réalisée grâce à l'application de l'opération de restriction sur les tuples de la table à fragmenter. La FH primaire favorise les opérations de sélection portée sur les attributs de fragmentation.
2. **La fragmentation horizontale dérivée :** c'est une FH dérivée car elle permet de fragmenter une relation suivant la fragmentation horizontale primaire d'une seconde relation, à condition de l'existence d'une clé étrangère (relation père-fils) qui pointe de la première

table (relation membre) à la seconde table (relation propriétaire). Le calcul des fragments dérivés se fait par la semi-jointure des fragments propriétaire avec la relation membre.

3.2.2 La fragmentation mixte (hybride)

La fragmentation mixte (figure 1.8) combine les deux types de fragmentation : horizontale et verticale. Elle consiste à partitionner une relation en sous-ensembles de sous relations, ces dernières étant définies par la fragmentation verticale et les sous-ensembles par la fragmentation horizontale[12].

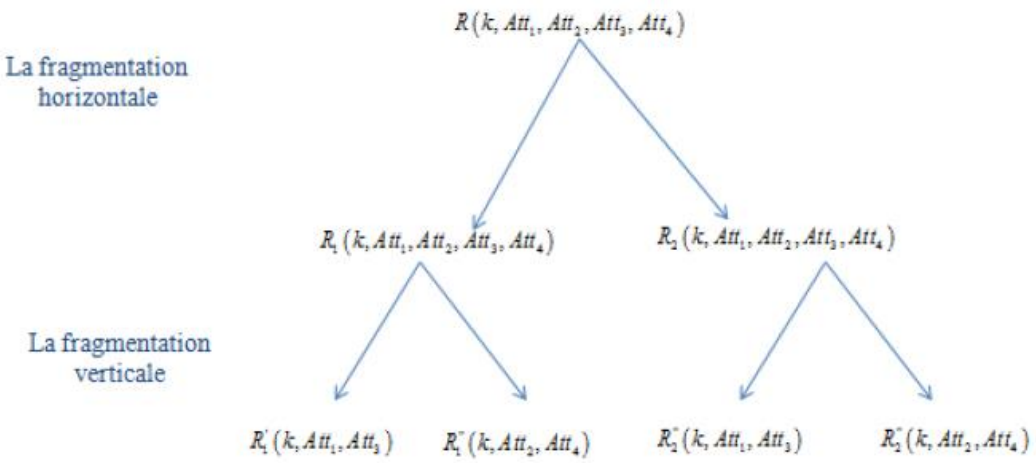


Figure 1.8 : La fragmentation mixte suivant une relation R (k, Att1, Att2, Att3).

3. Conclusion

Dans ce chapitre, nous avons donné un aperçu sur les concepts de base des entrepôts de données, de plus nous avons présenté l'architecture et la modélisation multidimensionnelle des entrepôts. Enfin, nous avons passé en revue sur les différentes techniques d'optimisation couramment utilisées pour l'exploitation de ces entrepôts.

Chapitre 02 : Optimisation dans les entrepôts de données

1. Introduction

Dans ce chapitre, nous allons définir la fragmentation dans les entrepôts de données. Puis nous rappellerons la méthodologie de fragmentation. Nous décrirons après le processus de fragmentation horizontale, ainsi que le problème de sélection d'un schéma de fragmentation. On conclure avec la citation des avantages de fragmentation horizontale.

2. Fragmentation dans les entrepôts de données

2.1 Définition

« La fragmentation consiste à diviser un ensemble de données en plusieurs partitions, appelées fragments, de manière à ce que la combinaison des fragments recouvre l'intégralité des données sources sans ajout ni perte d'information » [21]. Et pour bien expliquer cette définition, nous allons donner l'exemple suivant :

Soient les trois (03) tables de dimensions suivantes :

Twilaya {Code_Wilaya, Libellé Wilaya}, et TRégion {Code Région, Libellé Région}, et TNCitoyen {Niveau_Scolaire_Citoyen, Libellé_NS_C}, telles que les valeurs des codes Wilaya appartiennent à {01,02,…,48} et les codes Régions appartiennent à {Est, Ouest, Sud, Nord}, et Niveaux Scolaire Citoyens appartiennent à {Sans, Primaire, Secondaire, Universitaire, Post_graduant}.

La table des Faits est TCitoyen {Matricule_Citoyen, Code_Région, Code_Wilaya, Niveau_Scolaire_Citoyen, Date, Salaire_Citoyen}.

Les combinaisons mentionnées dans la définition précédente sont énumérées de la façon suivante, et ceci dans un schéma de fragmentation qui tient en compte les deux axes (Code_Région, Niveau_Scolaire_Citoyen) :

Sans Λ Est, Sans Λ Ouest, Sans Λ Sud, Sans Λ Nord.

Primaire Λ Est, Primaire Λ Ouest, Primaire Λ Sud, Primaire Λ Nord.

Secondaire Λ Est, Secondaire Λ Ouest, Secondaire Λ Sud, Secondaire Λ Nord.

Universitaire Λ Est, Universitaire Λ Ouest, Universitaire Λ Sud, Universitaire Λ Nord.

Post_Graduant Λ Est, Post_Graduant Λ Ouest, Post_Graduant Λ Sud, Post_Graduant Λ Nord.

Nous remarquons que le nombre de combinaisons est égal à Quatre (04) ou nombre de valeurs prise par Code_Région multiplié par Cinq (05) ou nombre de valeurs prises par l'attribut Niveau_Scolaire_Citoyen, soit Vingt (20) combinaisons. La génération de ces combinaisons assure la complétude de la table des Faits fragmentée par rapport à toutes ces combinaisons [22].

2.2 Méthodologie de fragmentation horizontale

L'analyse multidimensionnelle qui justifie la raison d'être des entrepôts de données augmente l'enjeu de la structure de fragmentation horizontale dans la conception physique des entrepôts. En effet dans un entrepôt de données relationnel deux types de tables rentrent en jeux à savoir les tables de dimensions et les tables de faits. Ces tables sont toutes concernées par une telle fragmentation horizontale, et dans ce sens plusieurs scénarios peuvent être envisagés :

- ❖ Fragmenter uniquement les tables de dimensions en fonction de la fragmentation horizontale primaire. Ce scénario n'est pas du tout utile pour les raisons suivantes : tout d'abord les valeurs ou mesures cherchées se trouvent au niveau de la table de faits qui sont par conséquence l'objet des requêtes décisionnelles en passant par des opérations de jointure généralement coûteuses, la volumétrie des données figure dans les tables de faits et pas dans les tables de dimensions. D'où ce cas est à éliminer.
- ❖ Fragmenter uniquement la table des faits en se basant sur une fragmentation primaire.

Ce scénario aboutit au même titre que le précédent à la mise à l'écart vu les raisons suivantes : la restitution des informations figurant dans la table de faits fait référence généralement aux clés étrangères des tables de dimensions, et les prédicats de sélections sont aussi généralement définis sur les attributs des tables de dimensions, comme il y a absence totale des jointures entre les tables de dimensions.

- ❖ Fragmenter totalement ou partiellement les tables de dimensions à l'aide de fragmentation primaire et se baser sur le résultat de leurs schémas de fragmentation pour passer à la fragmentation de la table de faits. Ce scénario retenu pour notre étude, procède comme suit : fragmenter certaines/toutes les tables de dimensions en utilisant les prédicats de sélection simples, cette méthodologie considère la conception relationnelle entre la table des faits et les tables de dimensions [23].

2.3 Processus de Fragmentation horizontale [5]

2.3.1 Préparation de la fragmentation

Cette étape permet de collecter toutes les informations nécessaires pour le processus de fragmentation à partir de l'ensemble des requêtes les plus fréquentes $Q = \{Q1, Q2, \cdots, Qm\}$. Chaque requête est caractérisée par sa fréquence d'accès ainsi que les prédicats de sélection qu'elle utilise. L'entrepôt de données est modélisé par un schéma en étoile composé d'une table des faits F et un ensemble de tables de dimensions $D = \{D1, D2, \cdots, Dn\}$.

La fragmentation de l'entrepôt de données nécessite la collecte de deux types d'informations

: (1) des informations quantitatives, comme la fréquence d'accès des requêtes et les facteurs de sélectivité des prédicats de sélection et de jointure, et (2) des informations qualitatives représentant l'ensemble des prédicats utilisés par les requêtes les plus fréquentes [24].

L'étape de préparation de la fragmentation est composée de quatre sous-étapes : (1) l'extraction des prédicats de sélection, (2) l'identification des tables de dimensions concernés par le processus de fragmentation, (3) la généralisation d'un ensemble des prédicats complet et minimal pour chaque table candidate et (4) le découpage du domaine de chaque attribution en sous-domaines [25].

2.3.1.1 Extraction des prédicats de sélection

Chaque requête de jointure en étoile utilise un ensemble de prédicats de sélection définis sur des attributs de dimension. Un prédicat de sélection Pk défini sur un attribut Aj d'une table de dimension Di possède la forme suivante :

Di.Aj θ Valeur Où θ représente un opérateur de comparaison parmi l'ensemble {=, <,>, <=,>=} et valeur ∈ Domaine(Aj). Rappelons que le domaine d'un attribut est l'ensemble de toutes ses valeurs que peut prendre. L'ensemble des prédicats de sélection extraits à partir de l'ensemble des requêtes est noté par P (soit l sa cardinalité). A partir de cet ensemble, nous regroupons les prédicats par attributs. Pour chaque attribut de dimension Aj, nous construisons l'ensemble de ses prédicats, cet ensemble est noté EPSAj. Par conséquent, le résultat de cette étape est un ensemble de prédicats de sélection pour chaque attribut.

2.3.1.2 Identification des tables de dimensions candidates

Dans le scénario de fragmentation que nous avons adopté, un sous-ensemble de tables de dimension sera fragmenté. Le but de cette étape est de définir l'ensemble des tables de dimension candidates au processus de fragmentation. Une table de dimension Di est candidate pour la fragmentation s'il existe des prédicats de sélection définis sur ses attributs. Nous construisons pour chaque table de dimension Di, l'ensemble de ses prédicats de sélection EPSDi. Cet ensemble est défini comme suit : EPSDi = {pk ∈ P/pk ∈ EPSAj ∧ Aj

EAi, $1 \leq k \leq l$, $1 \leq j \leq ni$} où EAi représente l'ensemble des attributs de tables Di et ni sa cardinalité. Deux scénarios peuvent être considérés pour chaque ensemble EPSDi :

- EPSDi = -, ce cas signifie qu'aucun prédicat n'est défini sur les attributs de la table Di.

Par conséquent, cette table ne sera pas candidate à la fragmentation et donc non utilisée pour fragmenter la table des faits.

– EPSDi 6= -, dans ce cas la table de dimension Di peut être fragmentée en utilisant l'ensemble EPSDi. Elle est donc candidate au processus de fragmentation.

A partir des ensembles EPSDi ($1 \leq i \leq d$) non vides nous définissons l'ensemble des tables de dimension candidates comme suit : $D_{candidates} = \{Di/Di \in D \wedge EPSDi \ 6= -, 1 \leq i \leq d\}$. Soit h la cardinalité de l'ensemble $D_{candidates}$.

2.3.1.3 Génération d'un ensemble de prédicats complet et minimal

Le but de cette étape est de générer pour chaque table de dimension un ensemble de prédicats complet et minimal. La complétude garantit que chaque fragment soit accédé par toutes les applications avec la même probabilité. La minimalité garantit que tous les fragments d'une table soient accédés différemment par au moins une application.

Définition : un ensemble de prédicats simples P est complet si et si seulement il y a une probabilité équivalente d'accès pour chaque application à n'importe quel n-uplets appartenant à un fragment défini par un minterm généré par P[1].

La minimalité d'un ensemble de prédicats stipule que si un prédicat dans cet ensemble permet de partitionner un fragment f en deux fragments fi et fj, alors au moins une application accède à fi et fj individuellement. Cela veut dire que le prédicat est pertinent dans la détermination de fi et fj [22].

Exemple *:* Supposons l'ensemble P composé des prédicats P1, P2 et P3, définis sur la table Client tel que : P1 : Ville =′ Poitiers′, P2 : Ville =′ Nantes′ et P3 : Ville =′ Paris′.

Supposons que la table Client est fragmentée en trois fragments : Client1, Clients2, Clients3 définis respectivement par les minterms m1, m2, et m3, où chaque minterm mi correspondant à un prédicat Pi ($1 \leq i \leq 3$).

Supposons qu'il existe une seule application accédant à la table Client en utilisant l'attribut Ville (nous supposons qu'il y a seulement trois villes). Tous les n-uplets de Client1, Client2 et Client3 ont la même probabilité d'être accédés par cette application, donc l'ensemble P = {P1, P2, P3} est complet.

Supposons maintenant une autre application accédant à ces trois fragments de la table Client mais pour les clients âgés de moins de 18 ans. Dans ce cas, certains n-uplets des trois fragments ont une forte probabilité d'être accédés, donc P devient non complet. Pour rendre cet ensemble complet, nous ajoutons le prédicat P4 : Age < 18 et sa négation P5 : Age ≥ 18, ce qui rend l'ensemble P′ = {P1, P2, P3, P4, P5} complet.

L'ensemble P′ = {P1, P2, P3, P4, P5} est aussi minimal car tous les prédicats le constituant sont pertinents pour lui. Si nous ajoutons un prédicat P6 : Saison =′ hiver′ à l'ensemble P′, ce

[1] Un minterm est une conjonction de prédicats définissant un fragment horizontal

dernier devient non minimal. Cela est dû au fait que le prédicat P6 n'est pas pertinent à P′ car toutes les applications accèdent à tous les fragments générés par P6 en même temps.

La génération d'un ensemble de prédicats simple minimal et complet est un enjeu important avant de fragmenter une table. Ozsu et al ont proposé un algorithme appelé COM−MIN permettant de générer un ensemble de prédicats complet et minimal. L'algorithme COM−MIN repose sur la règle de complétude et de minimalité suivante :

Règle1 : une relation ou un fragment est partitionné en au moins deux fragments qui sont accédés différemment par au moins une application.

L'algorithme COM −MIN est représenté dans l'algorithme 1. Il commence par initialiser l'ensemble des prédicats résultats par un prédicat et sa négation qui partitionne la table en deux fragments en respectant la règle1. Ensuite itérativement, ajoute tout nouveau prédicat qui partitionne chaque fragment existant en deux fragments en respectant la règle1. Après chaque itération de l'algorithme, les prédicats non pertinents par rapport à l'ensemble des prédicats résultats seront éliminés.

Algorithme COM-MIN
1 : Entrée : R : relation, P : ensemble de prédicats simples
2 : Sortie : P' : ensemble de prédicats complet et minimal
3 : Variables : F : l'ensemble des fragments, fi : un fragment défini par minter mi
4 : Début
5 : $P' = \Theta$
6 : Choisir un prédicat pi 2 P tel que pi partitionne R en respectant la règle 1
7 : $P' = \{p_i\}$; $P = P - \{p_i\}$; $F = \{f_i\}$;
8 : Répéter
9 : Choisir un prédicat pi 2 P tel que pi partitionne des fragments de R en respectant la règle 1
10 : $P' = P' [\{p_i\}$; $P = P - \{p_i\}$; $F = F [\{f_i\}$;
11 : S'il existe un prédicat p_j 2 P′ qui n'est pas pertinent alors $P' = P' - \{p_j\}$; $F = F - \{f_j\}$;
12 : Jusqu'à ce que (P' est complet)
13 :Fin.

Algorithme 2.1 : Algorithme COM-MIN.

2.3.1.4 Découpage du domaine de chaque attribut en sous-domaines

A l'issue de l'étape précédente, à chaque table de dimension Di est attribué un ensemble de prédicats simple complet et minimal EPSSMDi. Cet ensemble contient un ensemble de prédicats définis sur des attributs de la table Di.

Nous créons pour chaque attribut A_j ($1 \leq j \leq na$, où na est le nombre d'attributs de sélection[2]) l'ensemble de ses prédicats à partir de l'ensemble $EPSSMD_i$ de la table de dimension. Cet ensemble est noté EPCMAj.

Chaque prédicat pk dans EPCMAj permet de sélectionner un sous-ensemble de valeurs du domaine de définition de l'attribut Aj . Ce prédicat permet alors de découper le domaine de cet attribut en deux sous-domaines sd1 et sd2. Le sous-domaine sd1 correspond aux valeurs sélectionnées par Pk et sd2 correspond aux valeurs restantes. Par exemple, le prédicat P : Age <18 permet de découper le domaine de Age en deux sous-domaines, [0, 18] et] 18, 120]. Si nous considérons tous les prédicats dans EPCMAj , un découpage du domaine de l'attribut Aj en plusieurs sous-domaines est généré.

La FH d'une table de dimension Di est basée sur son ensemble de prédicats EPSCMDi or cet ensemble permet de découper les domaines de valeurs des attributs de fragmentation en sous-domaines. Par conséquent, la fragmentation horizontale primaire d'une table correspond à un certain découpage des domaines des attributs de fragmentation en sous-domaines.

Un découpage initial des domaines des attributs de sélection est effectué par l'AED. Dans ce découpage, chaque domaine est décomposé en un ensemble de sous-domaines stables9[3]. Ce découpage peut se faire selon deux manières :

– A priori : le découpage est réalisé lors de la conception de l'application accédant à l'entrepôt de données. Par exemple le système de gestion des réservations de la SNCF gère des clients en fonction de leur âge, carte enfants, carte 12-25 ans et carte senior.
– A posteriori : l'AED découpe le domaine de chaque attribut selon les prédicats définis sur cet attribut dans les requêtes accédant à l'entrepôt.

2.3.1.5 La sélection d'un schéma de fragmentation

Nous avons vu dans la section précédente que la FH d'une table correspond à un découpage du domaine de ses attributs de fragmentation. La sélection d'un schéma de fragmentation permet, à partir des tables de dimension candidates ainsi que le découpage initial des domaines

[2] Un attribut de sélection est un attribut sur lequel un prédicat de sélection est défini

[3] Un sous-domaine stable est un sous-domaine appartenant au découpage initial du domaine d'un attribut

des attributs de sélection, de générer tous les schémas de fragmentation possibles et de choisir le meilleur schéma. Un schéma de fragmentation d'une table de dimension Di est défini par :
– La liste des attributs de fragmentation de cette table.
– Pour chaque attribut de fragmentation, un découpage de son domaine de valeurs. Ce découpage est déterminé à partir du découpage initial en combinant certains sous-domaines stables.

Le schéma de fragmentation sélectionné doit réduire le temps d'exécution des requêtes. Donc, chaque schéma de fragmentation doit être évalué et sa contribution dans la réduction du coût d'exécution des requêtes quantifiée. La sélection d'un schéma de fragmentation est composée de trois étapes, la génération des schémas de fragmentation, l'évaluation de chaque schéma et la sélection du meilleur schéma.

2.3.2 Problème de sélection d'un schéma de fragmentation [26]

Le problème de sélection d'un schéma de fragmentation est formalisé comme suit :
Étant donné :

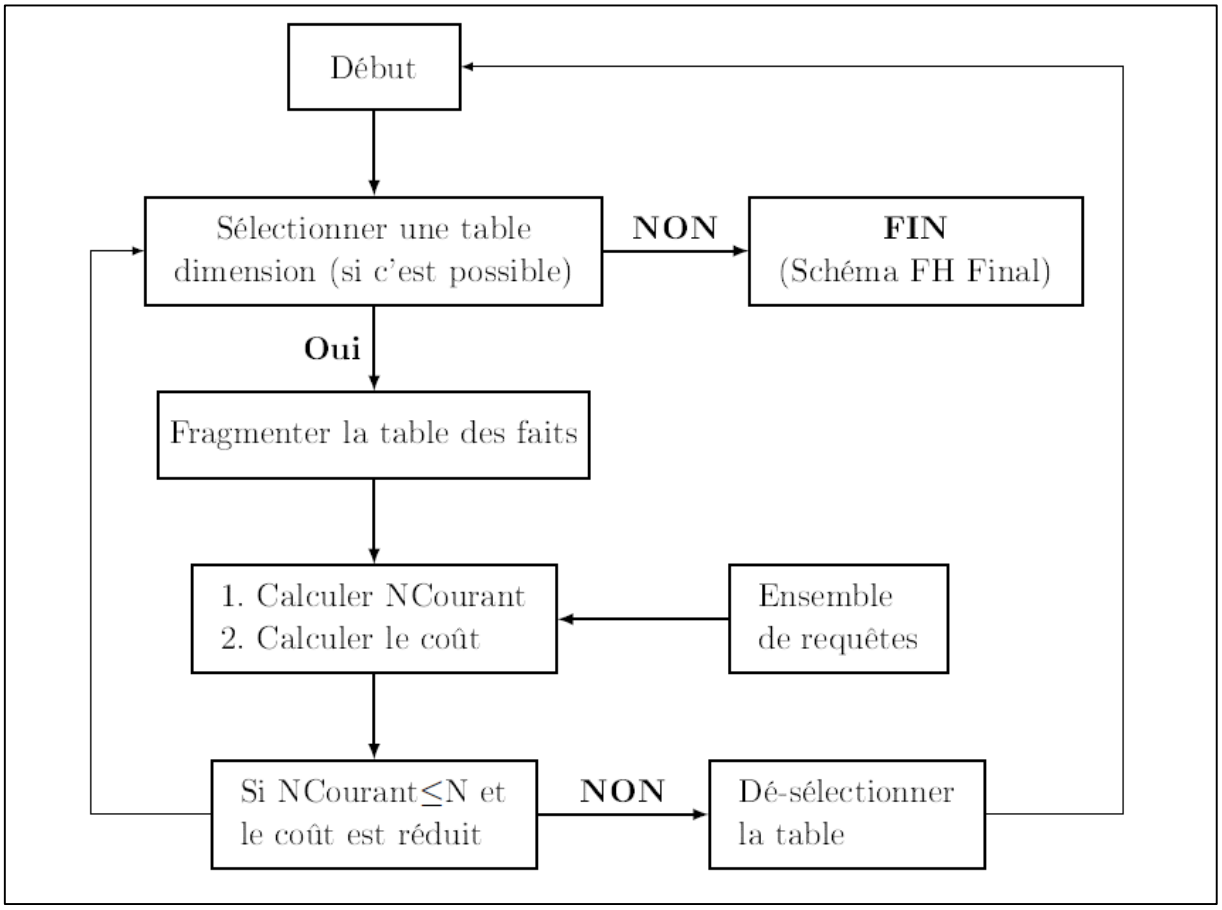
- une table T à fragmenter
- Une Charge de requêtes Q = {Q1, Q2, · · ·, Qm}

Le problème de sélection d'un schéma de fragmentation consiste à fragmenter la table T en n fragments horizontaux tel que le coût d'exécution de la charge de requête soit optimisé. Nous allons présenter les travaux qui proposent une démarche de fragmentation horizontale dans les entrepôts de données relationnels, centralisés et modélisés en étoile.

2.3.2.1 Travaux de Boukhalfa et al.

Afin de répondre au problème d'explosion du nombre de fragments de la table de faits après fragmentation, ils formalisent le problème comme suit :
Etant donné :

– une charge de requêtes Q = {Q1, Q2, · · ·, Qm},
– F une table de fait et D = {D1, D2, · · ·, Dd} d tables de dimension,
– W le nombre de fragments maximum de la table de fait imposé par l'administrateur



Algorithme 2.2 : Algorithme glouton pour la fragmentation : Bellatreche et al.

Le problème consiste à sélectionner un schéma de fragmentation primaire SF des tables dimensions tel que :

- le coût d'exécution des requêtes soit minimal,
- après fragments dérivée de F suivant SF, le nombre de fragments de F ne dépasse pas W.

Le problème formalisé ainsi est un problème NP-Complet. Ils proposent ainsi une démarche de sélection d'un schéma de fragmentation optimal des tables dimensions qui se base sur des métaheuristiques (Hill Climbing, Recuit Simulé, Algorithmes Génétiques). Cette démarche englobe les étapes suivantes :

Préparer la fragmentation à partir de la charge de requêtes, un ensemble des prédicats de sélection est extrait. Les prédicats de sélection permettent de définir les tables dimensions candidates à la fragmentation. Ce sont les tables contenant un attribut figurant dans au moins prédicat. Ensuite, l'algorithme COM-MIN est exécuté afin de généré l'ensemble de prédicats complet et minimal.

Découper les domaines des attributs A l'issue de l'étape précédente, à chaque table de dimension Di est attribué un ensemble EPi de prédicats complet et minimal qui permet de définir les

attributs de fragmentation de la table Di. Pour chaque attribut Aj , il est possible de définir un découpage de son domaine en sous-domaines à partir de l'ensemble EPi. Par exemple, le prédicat P :

Age < 18 permet de découper le domaine de l'attribut Age en deux sous-domaines, [0,18[et [18,80].

Par conséquent, le découpage des domaines en sous-domaines des attributs de fragmentation de Di, à partir d'EPi, permet de définir une fragmentation horizontale primaire de Di. Le tableau 2.1 montre un exemple de dé coupage des domaines des attributs Age, PNom et Ville en sous-domaine.

Age	[0,18[	[18,80]		
PNom	P1	P2	P3	
Ville	Alger	Oran	Blida	Kala

Tableau 2.1 : Découpage des domaines d'attributs en sous-domaines

Coder le schéma de fragmentation Le découpage en sous-domaines des domaines d'attributs de fragmentation permet de définir un schéma de fragmentation primaire des tables dimensions. Il est possible de définir d'autres schémas de fragmentation en regroupant des sous-domaines d'un même attribut dans un seul ensemble. Afin de coder tous les schémas possibles, on représente sous forme d'un tableau de vecteurs le schéma de fragmentation, où chaque vecteur caractérise un attribut et où les cellules du vecteur caractérisent un sous-domaine. Chaque cellule possède un numéro tel que les sous-domaines qui possèdent le même numéro sont regroupés dans le même sous-ensemble. Le tableau 2.2 illustre un exemple de codage. Ce codage signifie que pour l'attribut Ville, les valeurs Alger et Oran sont regroupées dans un sous-ensemble et Blida et Kala dans un autre sous-ensemble. Les valeurs P1, P2 et P3 de l'attribut PNom sont toutes regroupées dans un seul sous-ensemble.

Age	1	2		
PNom	1	1	1	
Ville	1	1	2	2

Tableau 2.2 : Codage d'un schéma de fragmentation.

Fragmenter les tables à partir d'un codage Le codage du schéma de fragmentation permet de définir des prédicats de sélection sur les attributs de fragmentation. Pour chaque table dimension, la conjonction des prédicats de sélection définie sur ses attributs permet de donner une fragmentation sur cette table. Un fragment nommé ELSE, défini par la conjonction des

négations de tous les prédicats, est ajouté à chaque table dimension, afin d'assurer la complétude. Le codage, présenté dans le tableau 2.1 montre que toutes les valeurs de l'attribut PNom sont regroupées dans un même ensemble. Par conséquent, au une fragmentation n'est définie sur l'attribut PNom et don sur la table Produits.

Pour la table Clients, sa fragmentation est la suivante :

Client1 = σ (0<Age<18) ∧ (Ville=Alger ∨ Ville=Oran) (Clients)

Client2 = σ (0<Age<18) ∧ (Ville=Blida ∨ Ville=Kala) (Clients)

Client3 = σ (18≤Age<80) ∧ (Ville=Alger ∨ Ville=Oran) (Clients)

Client4 = σ (18≤Age<80) ∧ (Ville=Blida ∨ Ville=Kala) (Clients)

Suivant la fragmentation primaire obtenue des tables dimensions, la fragmentation dérivée de la table de faits est effectuée.

Sélectionner un schéma de fragmentation final Les auteurs proposent dans un premier lieu d'employer l'algorithme d'affinité afin de définir un schéma de fragmentation horizontale primaire des tables dimensions. Cet algorithme a été défini pour regrouper les attributs afin de réaliser la fragmentation verticale. L'idée part du principe que les tuples d'une table accédés simultanément doivent appartenir au même fragment horizontal. Vu qu'un fragment horizontal est défini à partir du découpage des domaines des attributs, il faut regrouper les sous-domaines fréquemment accédés ensemble dans le même ensemble. Le regroupement des sous-domaines s'effectuent à la base de leur affinité. L'affinité entre deux sous-domaines est donnée par la somme des fréquentes des requêtes utilisant simultanément ces deux sous-domaines. La sélection d'un schéma de fragmentation basée sur l'algorithme d'affinité est constituée des étapes suivantes :

1. Extraire les sous-domaines utilisés par les requêtes et figurant dans les prédicats de sélection (clause WHERE). Cela donne un ensemble de sous-domaines $\{SD_{1,1}, \cdots, SD_{1,n1}, \cdots SD_{n,1}, \cdots, SD_{n,nn}\}$ de n attributs de fragmentation,
   où $\{SDk_{,1}, \cdots, SD_{k,nk}\}$ est l'ensemble des sous-domaines de A_k.
2. Construire la matrice d'usage MUS : elle décrit l'usage des sous-domaines par les requêtes. Pour chaque attribut A_k, on construit une matrice MUS^k (m×nk) où m est le nombre de requêtes et nk le nombre de sous-domaines de l'attribut Ak. Un élément $MUS^k{}_{ij}$ ($1 \leq i \leq m$; $1 \leq j \leq nk$) est mis à 1 si Qi référence le sous-domaine $SD_{k,j}$, il est mis à zéro sinon. Soit l'attribut Ville de la table Clients avec le découpage en 5 sous-domaines SD_1= {Alger}, SD_2= {Oran}, SD_3= {Blida}, SD_4={Kala} et SD_5={Tizi}. Soit 8 requêtes utilisant des prédicats

de sélection définis sur l'attribut Ville. L'utilisation des sous-domaines par les 8 requêtes permet de définir la matrice d'usage décrite dans le tableau 2.3.

	SD_1	SD_2	SD_3	SD_4	SD_5	Fréq
Q_1	1	0	0	0	1	10
Q_2	0	1	1	0	0	50
Q_3	0	0	0	1	0	25
Q_4	0	0	0	0	1	5
Q_5	0	1	1	0	1	45
Q_6	1	0	0	0	1	15
Q_7	1	0	0	0	0	25
Q_8	0	0	0	1	0	15

Tableau 2.3 : Matrice d'usage des sous-domaines de l'attribut Ville

3. Construire la matrice d'affinité des sous-domaines : c'est une matrice carrée $MAS^k(n_k \times n_k)$ où les lignes et les colonnes représentent les sous-domaines. Chaque élément MAS^k_{ij} représente la somme des fréquentes des requêtes accédant simultanément aux deux sous-domaines $SD_{k,i}$ et $SD_{k,j}$. Le tableau 2.4 représente la MAS des sous-domaines de l'attribut Ville.

	SD_1	SD_2	SD_3	SD_4	SD_5
SD_1	50	0	0	0	25
SD_2	0	95	95	0	45
SD_3	0	95	95	0	45
SD_4	0	0	0	40	0
SD_5	25	45	45	0	75

Tableau 2.4 : Matrice d'affinité des sous-domaines de l'attribut Ville

4. Grouper graphiquement les sous-domaines en partitions : l'algorithme de regroupement graphique proposé est exécuté sur chaque matrice d'affinité de chaque attribut. L'algorithme génère un graphe complet et étiqueté, appelé graphe d'affinité des sous-domaines, où les noeuds représentent les sous-domaines et une arête la valeur d'affinité. L'algorithme forme

des cycles où chacun permet de regrouper les sous-domaines en un ensemble. La figure 2.1 illustre l'exécution du regroupement graphique pour l'attribut Ville. Il permet de définir la répartition

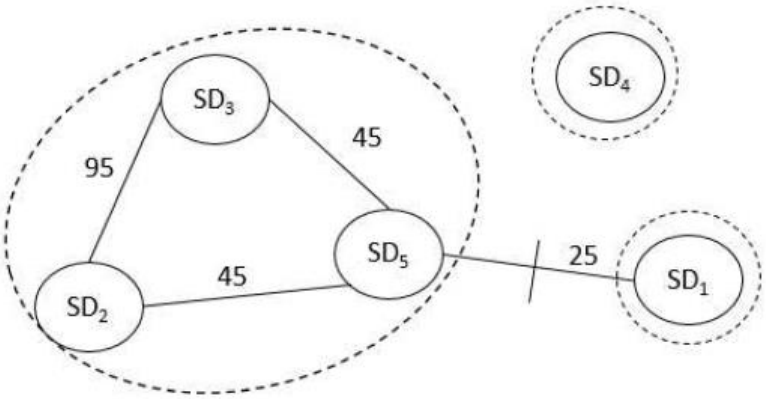


Figure 2.1 : Regroupement graphique des sous-domaines de l'attribut Ville

des sous-domaines en trois ensembles comme suit : Ens_1= {Alger}, Ens_2}= {Oran, Blida, Tizi} et Ens_3={Kala}.

5. Générer le schéma de fragmentation : chaque ensemble de sous-domaine se voit attribuer le même numéro pour le codage du schéma de fragmentation. Pour l'attribut Ville, on obtient Le codage {1, 2, 2, 3,2}. A l'issu de cette étape, on obtient un tableau numérique représentant le codage du schéma de fragmentation des tables dimensions. L'avantage de l'algorithme d'affinité est sa simplicité et sa complexité réduite pour la sélection d'un schéma de fragmentation. Cependant, ses inconvénients majeurs et qu'il utilise uniquement la fréquence d'accès comme critère de groupement, ne permet pas de contrôler le nombre de fragments généré et ne donnent au une garantie sur la qualité de la solution obtenue. Pour cela, la solution obtenue par l'algorithme d'affinité comme solution initiale pour trois algorithmes : l'algorithme de Hill Climbing, le Recuit Simulé et les Algorithmes génétiques. Le processus de sélection proposé est illustré sur la figure 2.2.

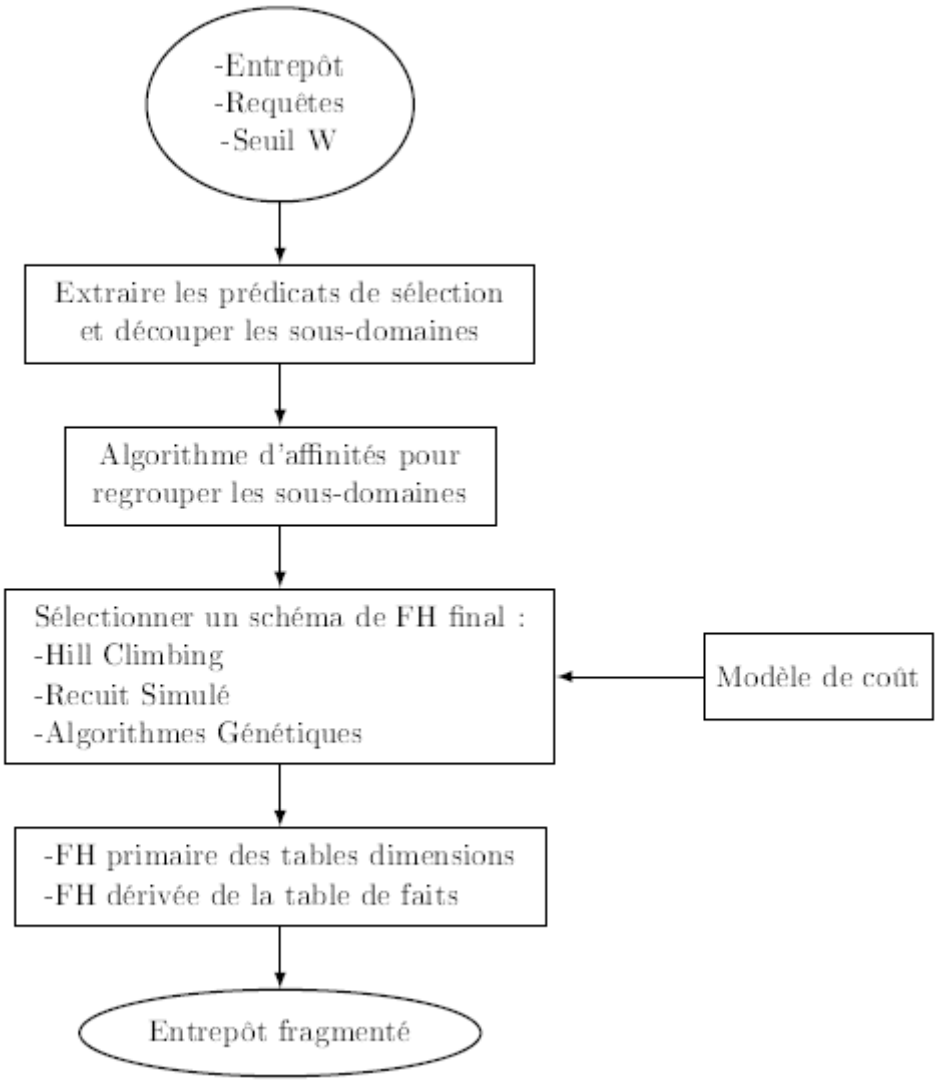


Figure 2.2 : Processus de fragmentation : Boukhalfa et al.

3. Les avantages de fragmentation horizontale

3.1 Fragmenter pour améliorer la performance

La FH permet d'éliminer les données non nécessaires pour répondre à une requête ce qui réduit considérablement le temps de réponse et donc augmente la performance. Une partition non pertinente pour une requête est une partition qui ne contient aucun n-uplet référencé par cette requête.

3.2 Fragmenter pour améliorer la facilité de gestion

La FH permet de décomposer les tables, vues et index en plusieurs partitions de taille inférieure donc plus faciles à manipuler et à gérer. Elle préserve aussi le schéma logique, ce qui implique que toutes les opérations effectuées sur les objets d'origine sont aussi applicables sur leurs partitions. Par conséquent, le système manipule une partition à la fois au lieu de la totalité de l'objet. Plusieurs opérations ont été définies pour manipuler des partitions. Nous pouvons

citer l'opération de fusion de deux partitions (MERGE PARTITION), l'opération d'éclatement d'une partition en deux partitions (SPLIT PARTITION), l'opération de conversion d'une partition en une table (EXCHANGE PARTITION).

3.3 Fragmenter pour améliorer la disponibilité

Lors des opérations de maintenance, la base de données est généralement non accessible, ce qui pénalise les utilisateurs du système. La FH permet de cibler la tâche de maintenance où seules certaines partitions seront concernées par cette tâche, ce qui réduit le temps d'arrêt du système. Les partitions non concernées par la maintenance resteront toujours disponibles. Lors de la fragmentation d'une table, l'administrateur peut spécifier un emplacement physique pour stocker chaque partition (TABLESPACE). Si les emplacements physiques des différentes partitions se trouvent sur des espaces physiques différents, alors une panne au niveau d'un espace de stockage n'affecte que les partitions stockées sur ce dernier, les autres partitions resteront toujours disponibles.

4. conclusion

Dans ce chapitre, nous avons au premier lieu présenté la fragmentation dans les entrepôts de données comme une technique d'optimisation. Nous avons expliqué la méthodologie de fragmentation, et pour pouvoir fragmenter l'entrepôt nous avons détaillé le processus de fragmentation

Chapitre 03 : Résolution du problème de sélection de schéma de fragmentation optimale

1. Introduction

Les méthodes de résolution des problèmes d'optimisation sont nombreuses. Elles sont souvent classées en deux grandes classes : la classe des méthodes exactes et la classe des méthodes approchées. Les méthodes exactes garantissent l'optimalité de la solution, mais elles sont souvent exigeantes en termes de temps de calcul nécessaire. Notamment, pour la résolution des problèmes de grande taille. Pour pallier à cette lacune, les méthodes approchées sont considérées actuellement comme un bon choix de résolution.

L'investigation dans le domaine des méthodes approchées a donné naissance à une autre catégorie de méthodes appelées « Métaheuristiques ». Les métaheuristiques sont des méthodes générales est applicables sur une grande gamme de problèmes d'optimisation. Elles sont souvent inspirées des systèmes naturels dans différents domaines : physique, biologie, éthologie….etc. Dans ce chapitre on va présenter en premier temps deux Metaheuristique : les algorithmes génétiques (AG) et la méthode descente ces deux méthodes seront adaptées en deuxième temps au problème de sélection de schéma optimal de fragmentation de l'entrepôt de données.

2. Présentation des algorithmes génétiques (AGs)

2.1 Principe de fonctionnement

Les algorithmes génétiques (**AG**s) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Leur fonctionnement est extrêmement simple. On part avec une population de solutions potentielles (*chromosomes*) initiales arbitrairement choisies. On évalue leur performance (*fitness*) relative. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. On recommence ce cycle jusqu'à ce que l'on trouve une solution satisfaisante [27]. Le principe de fonctionnement d'un AG est illustré dans algorithme 3.1 ci-dessous :

Initialisation de la population
Evaluation de fonctions objectives de chaque individu
Calculer l'efficacité de chaque individu
Tant que le critère d'arrêt n'est pas atteint **faire**
  Sélection de parent pour la reproduction
  Mutation
  Croisement
  Evaluation de fonctions objectives de chaque enfant

Calculer l'efficacité de chaque enfant

Sélection pour remplacement

Fin Tant que

Algorithme 3.1 : Pseudo cade d'un algorithme génétique.

2.2 Eléments d'un algorithme génétique

La mise en œuvre de l'algorithme génétique est basée sur les éléments suivants :

2.2.1 Le codage

Premièrement, il faut représenter les différents états possibles de la variable dont on cherche la valeur optimale sous forme utilisable pour un AG : c'est le codage. Cela permet d'établir une connexion entre la valeur de la variable et les individus de la population, de manière à limiter la transcription génotype-phénotype qui existe dans le monde vivant. Il existe principalement trois types de codage : le codage binaire, le codage réel et le codage en base n.

2.2.1.1 Codage binaire

Les individus intervenant dans lui algorithme génétique étaient codés sous forme de chaines de bits. Le codage binaire contient toute l'information nécessaire à la description d'un point dans l'espace d'état. C'est le codage par l'alphabet binaire {0,1}, c'est-à-dire qu'un gène est un entier long de 32 bits, et par conséquent, le chromosome est représenté par un tableau des gènes, et les individus par un tableau des chromosomes. Un des avantages du codage binaire est que l'on peut facilement coder toutes sortes d'objet : des réels, des entiers, des valeurs booléennes, des chaînes de caractère... cela nécessite simplement l'usage des fonctions de codage et décodage pour passer d'une représentation à l'autre. Cependant, ce type de codage peut devenir si mauvais dans des espaces de grande dimension [28].

2.2.1.2 Codage réel

Il a le mérite d'être simple. Chaque chromosome est en fait un vecteur dont les composantes sont les paramètres du processus d'optimisation. Par exemple, si on recherche l'optimum d'une fonction de n variables f(x1, x2, . . ., xn−1, xn), on peut utiliser tout simplement un chromosome *ch* contenant les n variables : Avec ce type de codage, *ch*: x1 ,x2 ,· · · xn−1, xn.

La procédure d'évaluation des chromosomes est plus rapide vu l'absence de l'étape de transcodage (du binaire vers le réel). Les résultats donnés montrent que la représentation réelle aboutit souvent à une meilleure précision et un gain important en termes de temps d'exécution [7].

2.2.1.3 Codage en base n

Dans ce type de codage, les gènes constituant un chromosome sont des chiffres exprimés dans une base de numération n, ce qui permet de représenter n valeurs discrètes.

L'AG démarre avec une population composée de N individus dans le codage retenu. Le choix des individus conditionne fortement la rapidité de l'algorithme. Si la position de l'optimum dans l'espace de recherche est totalement inconnue, il est intéressant que la population soit répartie sur tout l'espace de recherche. Si par contre des informations à priori sur le problème sont disponibles, il paraît évident de générer les individus dans un espace particulier afin d'accélérer la convergence. Disposant d'une population initiale souvent non homogène, la diversité de la population doit être entretenue aux cours des générations afin d'explorer le plus largement possible l'espace de recherche. C'est le rôle des opérateurs de croisement et de mutation [7].

2.2.2 Génération de la population initiale

Le choix de la population initiale d'individus conditionne fortement la rapidité de convergence de l'algorithme. Si la position de l'optimum dans l'espace d'état est totalement inconnue, il est naturel de générer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace d'état en veillant à ce que les individus produits respectent les contraintes. Si des informations a priori sur le problème sont disponibles, il est naturel de générer les individus dans un sous domaine particulier afin d'accélérer la convergence [29].

2.2.3 Evaluation de la population

Evaluation de la population consiste à évaluer chaque solution contenue dans la population, la mesure de performance des solutions s'appuie sur la valeur de fonctions objectives [30]. Contrairement à bon nombre de méthodes qui requièrent beaucoup d'informations pour pouvoir fonctionner efficacement, les algorithmes génétiques nécessitent peu d'informations, ils fonctionnent essentiellement de manière aveugle. Pour effectuer une recherche de solutions meilleures, ils n'ont besoin que des valeurs des fonctions objectives associées aux chaînes individuelles. Ces valeurs ont pour but d'évaluer si un individu est mieux adapté qu'un autre à son environnement. Ce qui signifie qu'elle quantifie la réponse fournit au problème pour une solution potentielle donnée. Ainsi, les individus peuvent être comparés entre eux. Les individus déterminés par la fonction objective (fitness) vont servir au processus de sélection des candidats aptes à la reproduction et au processus de survie des espèces. Cette fonction, propre au problème, est souvent simple à formuler lorsqu'il y a peu de paramètres. Au contraire, lorsqu'il y a beaucoup de paramètres ou lorsqu'ils sont corrélés, elle est plus difficile à définir. Dans ce cas, la fonction devient une somme pondérée de plusieurs fonctions. Un ajustement des coefficients est alors nécessaire [28].

2.2.4 Opérateurs génétiques

2.2.4.1 Opérateur de sélection

Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. Il permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. A partir d'une population initiale d'individus, l'algorithme génétique sélectionne une population intermédiaire d'individus en faisant mie sélection sur la population initiale (mais même individu peut être sélectionné plusieurs fois ou peut ne pas être sélectionné du tout, en fonction de la valeur de sa fonction d'évaluation). On trouve dans la littérature un nombre important de principe de sélection plus ou moins adaptés aux problèmes qu'ils traitent [28]. Les plus couramment utilisés sont :

- **Sélection par roulette** : Dans ce type de sélection, les chromosomes parents sont choisis en fonction de leur performance. Meilleur est le résultat calculé à partir du code du chromosome, plus grandes sont ses chances d'être sélectionné. La sélection par roulette peut être vue comme une sorte de roulette de casino sur laquelle sont mis tous les chromosomes de la population. La place accordée à chaque chromosome étant en relation avec sa valeur d'adaptation. Un exemple de roulette de sélection est représenté par la figure 3.7.

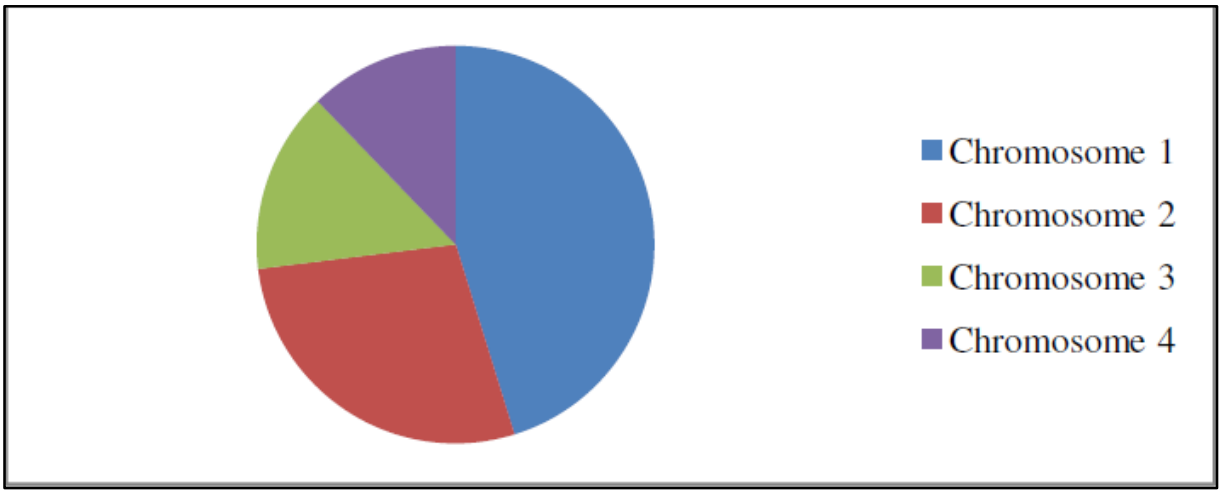


Figure 3.1 : schéma de roulette de sélection

Ensuite, la bille est lancée et s'arrête sur un chromosome. Ainsi, les meilleurs chromosomes peuvent avoir la chance d'être tirés plusieurs fois, et les plus mauvais ne jamais être sélectionnés. [31]

- **Sélection par rang** : La sélection par roulette est confrontée à des problèmes quand la valeur d'adaptation des chromosomes varie considérablement. Si la meilleure fonction d'évaluation d'un chromosome représente un grand pourcentage de la roulette alors les autres chromosomes auront très peu de chance d'être sélectionnés et on arriverait à un arrêt de l'évolution. La sélection par rang effectue d'abord un tri sur la population par rapport à la

fitness. Ensuite, à chaque chromosome est associé un rang en fonction de sa position. En conséquence, pour une population de N chromosomes, le plus mauvais aura le rang 1, le suivant 2, et ainsi de suite jusqu'au meilleur chromosome qui aura le rang N. La sélection par rang et la sélection par roulette sont presque identiques, sauf que, avec la sélection par rang les proportions sont en rapport avec le rang plutôt qu'avec la valeur de l'évaluation. Ainsi, les probabilités finales seront calculées avec la formule : Rang / ∑(des Rangs). Le tableau 3.1 fournit un exemple de sélection par rang Nous voyons qu'avec cette méthode de sélection, les chromosomes ont tous une chance d'être sélectionnés. Néanmoins, elle aboutit à une convergence plus lente vers la bonne solution, car les meilleurs chromosomes ne sont pas très différents des plus mauvais [31].

Chromosomes	**1**	**2**	**3**	**4**	**5**	**6**	**total**
Probabilités initiales	89%	5%	1%	4%	3%	2%	100%
Rang	6	5	1	4	3	2	21
Probabilités finales	29%	24%	5%	19%	14%	9%	100%

Tableau 3.1 : exemples de sélection par rang pour 6 chromosomes.

- **Sélection par tournoi** : Cette méthode ressemble le plus à ce qui se passe en réalité. Deux individus, ou plus, sont choisis au hasard et combattent (on compare leurs fonctions d'adaptation) pour que le meilleur gagne. Cette étape est répétée jusqu'à ce que la nouvelle population soit remplie. Il est tout à fait possible, lors d'un même tournoi, que certains individus seront choisis plusieurs fois s'ils gagnent, ou encore permettre au même individu d'être éligible plusieurs fois [28].

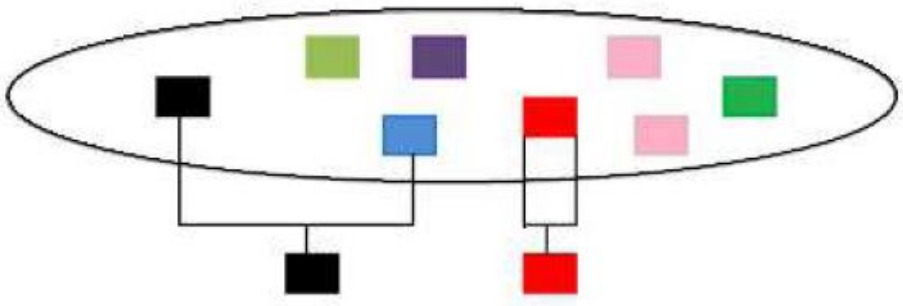


Figure 3.2 : le tournoi

Toutefois, aussi importante que soit la phase de sélection, elle ne crée pas de nouveaux individus dans la population. Ceci est le rôle des opérateurs de croisement et de mutation.

2.2.4.2 Opérateur de croisement

Le croisement est l'échange d'un certain nombre de bits entre deux chromosomes représentant deux individus de la population. En choisissant aléatoirement deux individus parents, l'opération de croisement donne naissance à deux nouveaux individus.

Il existe différents types de croisements pour un algorithme génétique les trois principaux types de croisement sont présentés (en un point, en deux points et uniforme). Cette opération est contrôlée par une probabilité.

Croisement en un point : La population courante est divisée en deux sous populations de même taille (p/ 2) et chaque couple formé par un membre provenant de chaque sous population participe à un croisement avec une probabilité. Si le croisement a eu lieu entre deux chromosomes parents (c*h*1 et c*h*2), constitués de l gènes, on tire aléatoirement une position de chacun des parents. On échange ensuite les deux sous chaînes terminales de chacun des chromosomes, ce qui produit deux enfants c*h*1' et c*h*2 ' comme indiqué sur la figure 3.3.a. On peut noter que le nombre de points de croisements ainsi que la probabilité de croisement permettent d'introduire plus ou moins de diversité. En effet, plus le nombre de points de croisements sera grand et plus la probabilité de croisement sera élevée plus il y aura d'échange de segments, donc d'échange de paramètres, d'information, et plus le nombre de points de croisements sera petit et plus la probabilité de croisement sera faible, moins le croisement apportera de diversité.

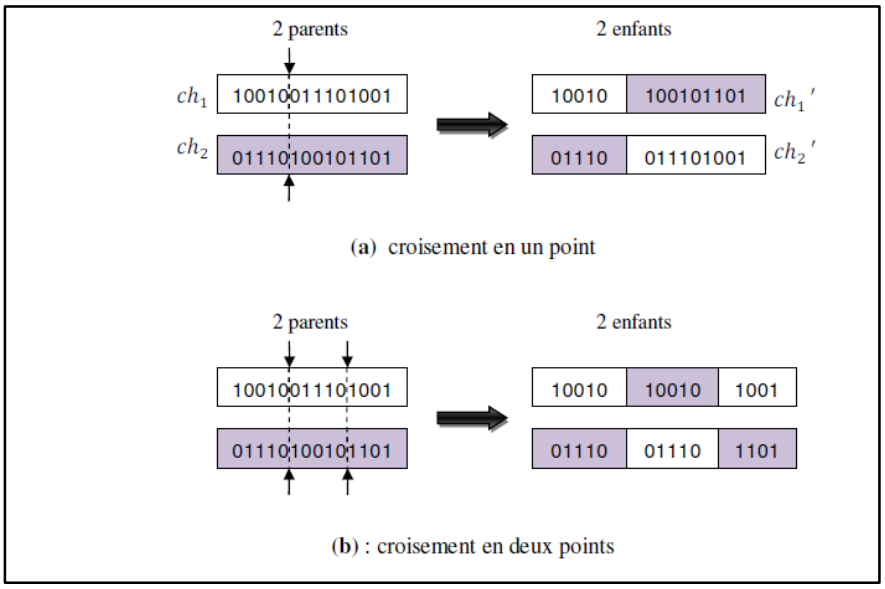


Figure 3.3 : Exemples d'opérations de croisement simple.

Croisement uniforme : Il consiste à définir de manière aléatoire un "masque", c'est-à-dire une chaîne de bits de même longueur que les chromosomes des parents sur lesquels il sera appliqué. Ce masque est destiné à savoir, pour chaque locus, de quel parent le premier fils devra hériter du gène s'y trouvant ; si face à un locus le masque présente un 0, le fils héritera le gène s'y trouvant du parent #1, s'il présente un 1 il en héritera du parent #2. La création du fils #2 se fait de manière symétrique : si pour un gène donné le masque indique que le fils #1 devra recevoir celui-ci du parent #1 alors le fils #2 le recevra du parent #2, et si le fils #1 le reçoit du parent #2 alors le fils #2 le recevra du parent #1[29].

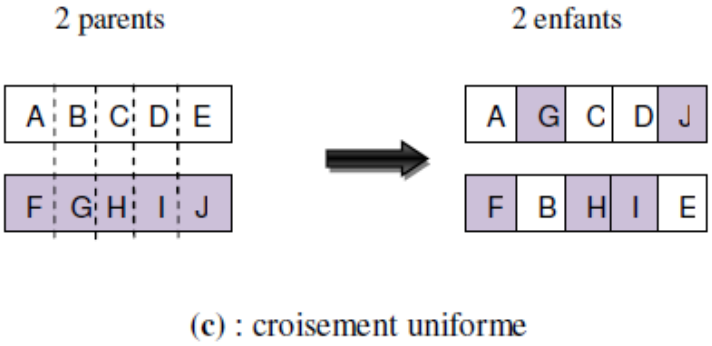


Figure 3.4 : croisement informe.

2.2.4.3 Opérateur de mutation

L'opérateur de mutation effectue en général un déplacement local sur la solution représentée par un individu. La mutation est considérée comme un opérateur marginal pour les AGs, bien qu'elle leur confère la propriété d'ergodicité, c'est-à-dire que tous les points de l'espace de recherche peuvent être atteints ; cet opérateur a une grande importance, de point de vue théorique. L'opérateur classique de mutation sur les chaînes de bits choisit aléatoirement une position dans le gène et le remplace par son complémentaire pour construire l'individu mutant comme l'illustre la figure 3.5.

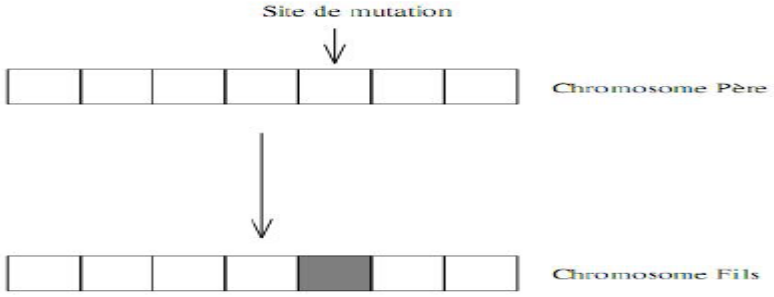


Figure 3.5 : Mutation classique .

On peut généraliser cette mutation sur des chaînes à domaines discrets de tailles quelconque en changeant la valeur du gène choisi par une autre de son domaine, proche de la valeur initiale.

Dans les problèmes continus (figure 3.6), on procède un peu de la même manière en tirant aléatoirement un gène dans le chromosome, auquel on ajoute un bruit aléatoire (par exemple un bruit gaussien) en veillant bien évidemment à ce que le gène résultant reste dans le domaine de définition qui lui est propre. L'écart type de ce bruit est délicat et difficile à régler : s'il est trop faible, l'exploration sera ralentie au début et on risque la convergence locale ; s'il est trop grand, les solutions seront modifiées trop brutalement et on ne pourra pas non plus converger localement vers l'optimum [1].

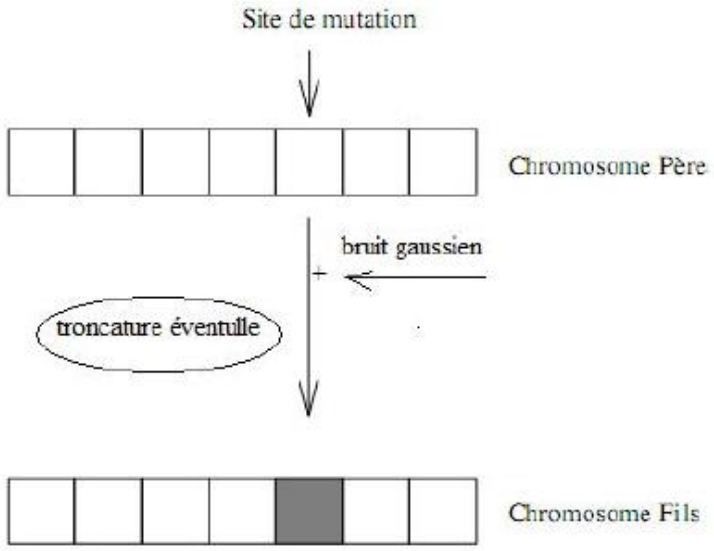


Figure 3.6 : Mutation continue.

Il existe également un principe de mutation adaptative figure 3.7, permettant d'optimiser le taux de mutation en codant ce dernier dans la structure du chromosome. Ce second chromosome est géré de la même manière que le premier chromosome codant l'espace d'état, c'est-à-dire lui-même soumit aux opérateurs génétiques (croisement et mutation). Au cours du déroulement de l'algorithme, les gènes et les individus ayant des probabilités de mutation élevées auront tendance à disparaître à mesure que la population converge vers l'optimum. De même, les gènes ayant des probabilités de mutation trop faibles ne peuvent évoluer favorablement et tendent à être supplantés.

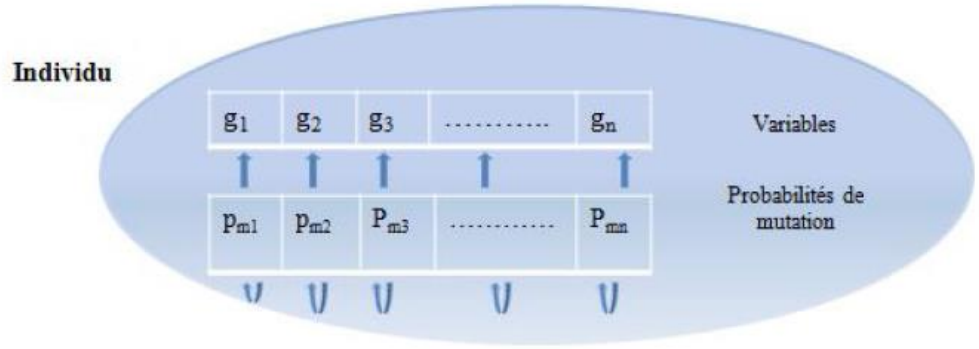


Figure 3.7 : Mutation adaptative.

2.2.5 Critère d'arrêt

Le cycle de génération et de sélection de population est répété jusqu'à ce qu'un critère d'arrêt soit satisfait ; ce critère peut être notamment un nombre maximum de générations, un temps maximal de calcul, une valeur de fitness minimale, ou/et une convergence vers une solution satisfaisante [29].

3. Présentation de la méthode descente

3.1 Principe de fonctionnement

La recherche locale simple ou la descente est un algorithme d'amélioration très ancien. Son principe consiste à explorer le voisinage de la solution courante afin d'améliorer sa qualité progressivement, comme le montre la figure 3.8 qui représente un schéma d'évolution d'une recherche locale simple. A chaque itération du processus d'amélioration, l'algorithme modifie un ensemble de composantes de la solution courante pour permettre le déplacement vers une solution voisine de meilleure qualité. Le processus est répété itérativement jusqu'à la satisfaction du critère d'arrêt. Il est à noter qu'il existe trois types de la descente : la descente déterministe, la descente stochastique et la descente vers le premier meilleur voisin.

L'algorithme 3.2 résume les étapes de l'algorithme général de la descente. L'algorithme entame la recherche par la construction d'une solution initiale s et l'évaluation de sa qualité f(s). Ensuite il commence l'ensemble des étapes du processus d'amélioration suivantes :

- Modifier s pour obtenir une nouvelle solution *s*' de meilleure qualité que s et qui appartient à son voisinage. Le choix de la fonction de voisinage est important. Il dépend de l'objectif à atteindre. En fait, si l'objectif est de minimiser le coût on doit suivre la direction de la vallée. Sinon (dans le cas de maximisation), on doit suivre la direction du sommet.
- Evaluer la qualité f (s') de la nouvelle solution *s*'.

- Remplacer s par *s*', si *s*' est de meilleure qualité.

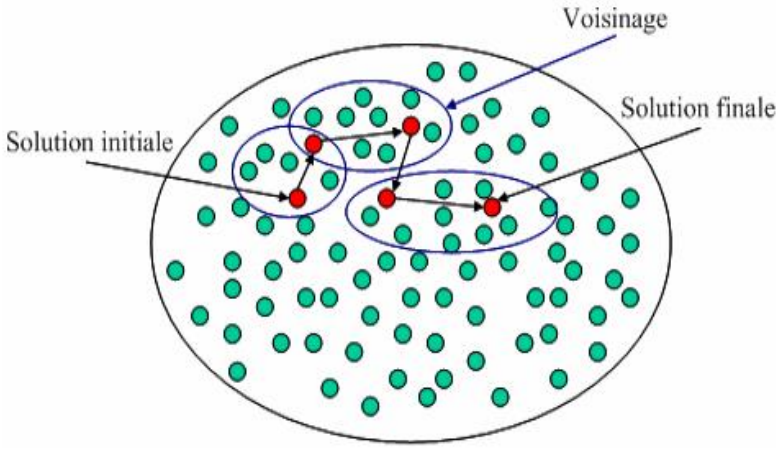


Figure 3.8 : Un schéma d'évolution de méthode descente.

Le processus d'amélioration sera répété jusqu'à arriver au cas où toutes les voisines candidates sont de piètre qualité que la solution courante. Autrement dit, la recherche s'arrête lorsqu'un optimum local est atteint. L'avantage de la recherche locale simple revient à sa simplicité et à sa rapidité[32].

Algorithme 2.1. La recherche locale simple (la descente)

1 : Début
2 : Construire une solution initiale s ;
3 : Calculer la fitness f (s) de s ;
4 : Tant que la condition d'arrêt n'est pas vérifiée **faire**
5 : Modifier s pour obtenir une nouvelle solution voisine *s*′;
6 : Calculer f (*s*′) ;
7 : Si f (*s*′) est meilleure que f (s) **alors**
8 : Remplacer s par s' ;
9 : Fin Si
10 : Fin Tant que
11 : Retourner s **;**
12 : Fin

Algorithme 3.2 : Algorithme de méthode descente.

4. Application des algorithmes génétiques au problème de sélection d'un schéma de fragmentation optimal

La fragmentation de la table des faits dans un environnement entrepôts de données pourrait engendrer un nombre important de fragments qui rendrait le processus de maintenance très coûteux. Afin de réduire ce nombre ou le rendre contrôlable par l'administrateur de l'entrepôt de données, plusieurs travaux de recherche ont proposé l'utilisation d'un Algorithme Génétique. Application les algorithmes génétique dans les entrepôts de données ayan comme

but la sélection des tables de dimension à fragmenter pour éviter l'expulsion du nombre de fragments de la table des faits et garantir une meilleure performance d'exécution des requêtes[1].

4.1 Codage

Tout algorithme de fragmentation horizontale nécessite les données d'un ensemble de requêtes les plus fréquentes. A partir de ces requêtes, nous extrairons deux types d'informations : qualitative et quantitative. Les informations qualitatives concernent les tables de dimension et sont représentées par les prédicats de sélection simples utilisés dans les requêtes. Les informations quantitatives concernent la sélectivité de ces prédicats et les fréquences d'accès des requêtes. Nous rappelons qu'un prédicat simple p est défini par p : A_i Θ Valeur, où A_i est un attribut d'une relation à fragmenter, $\Theta \in \{<,\leq,>,\geq,\neq,=\}$; Valeur $\in$ Dom (A_i) .

Avant de présenter notre mécanisme de codage des fragments des tables de dimension, nous devons identifier quelles tables de dimension participant dans la fragmentation de la table des faits. Pour se faire, nous procédons de la façon suivante :

1) Extraire tous les attributs simples employés par les m requêtes.
2) Assigner à chaque table de dimension Di ($1 \leq i \leq d$) son ensemble de prédicats simples, noté par SSPDi,
3) Les tables de dimension ayant un SSPD vide ne sont pas prises en considération dans le processus de fragmentation [1].

4.2 Représentation des fragments horizontaux

L'analyse des clauses représentant les fragments horizontaux permet d'effectuer une partition du domaine des attributs de fragmentation en sous-domaines appelés sous domaine. Les éléments de cette partition sont déterminés par les prédicats simples.

Exemple : Cet exemple montre comment les prédicats de fragmentation définissent des partitions de chaque domaine des attributs de fragmentation (un attribut de fragmentation est un attribut qui participe dans le processus de partitionnement). Considérons trois attributs de fragmentation : yearlevel, monthlevel et quarterlevel de la table de dimension TimeLevel. Supposons que les domaines des attributs de fragmentation sont :

Dom (yearlevel) = { '1995','1996'}.

Dom(monthlevel)={'199512','199511','199510','199509','199508','199507','199506','199505','199504','199503','199502','199501','199612','199611','199610','199609','199608','199607','199606','199605','199604','199603','199602','19901'}.

Dom (quarterlevel) = {'Q1','Q2','Q3','Q4'}.

Le domaine de l'attribut yearlevel est partitionné en deux sous domaine simple d_{11}= {'1995'} et d_{12} = {'1996'}.

D'une manière similaire, le domaine de l'attribut monthlevel est partitionné en 24 sous domaine simple d_{21}={'199512'},d_{22} ={'199511'},d_{23} ={'199510'},d_{24} ={'199509'},d_{25} ={'199508'},d_{26} ={'199507'},d_{27} ={199506},d_{28} ={'199505'},d_{29} ={'199504'},d_{210} ={'199503'},d_{211} ={'199502'}, d_{212} ={'199501'} , d_{213} ={'199612'},d_{214} ={'199611'},d_{215} ={'199610'},d_{216} ={'199609'}, d_{217} ={'199608'}, d_{218} ={'199607'} ,d_{219} ={'199606'} ,d_{220} ={'199605'},d_{221} ={'199604'}, d_{222} ={'199603'}, d_{223} ={'199602'}, d_{224}={'199601'}.

Finalement, l'attribut quarterlevel est partitionné en quatre sous domaine : d_{31} = {'Q1'}, d_{32} = {'Q2'}, d_{33} ={'Q3'} ,d_{34}={'Q4'}.

Les différents sous domaines de chaque attribut de fragmentation sont représentés par la figure 3.9 :

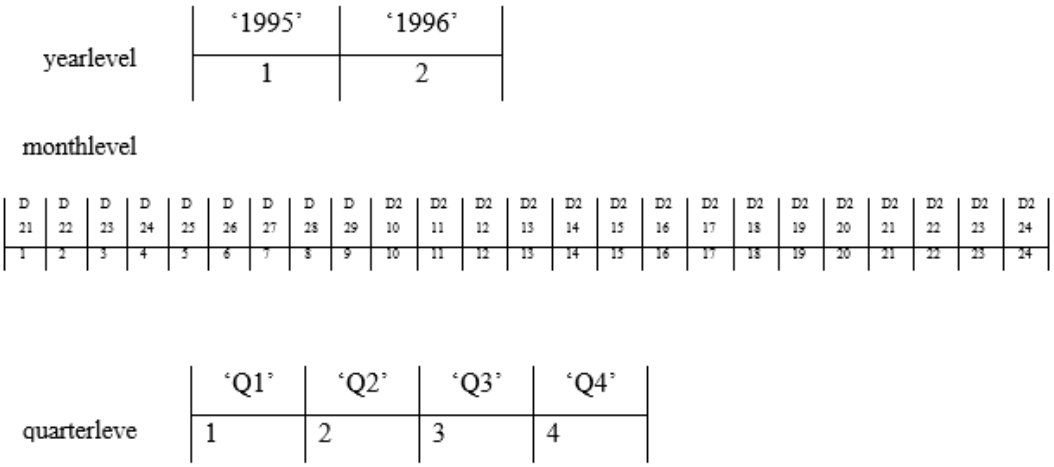


Figure 3.9 : les sous domaines des attributs de fragmentation.

Chaque attribut de fragmentation *i A* sera représenté par un tableau d'entiers de n_i cellules, où n_i correspond au nombre de sous domaines, les valeurs dans les cellules de ce tableau oscilleront entre 1 et n_i Si deux cellules ont la même valeur, on regroupe les sous domaines de manière à n'en former qu'un. Chaque individu (un schéma de fragmentation) est donc représenté par un tableau d'entiers (tableau 3.2).

yearlevel	1	2										
monthlevel	1	2	2	4	6	3	2	4	2	1	1	1
quarterlevel	1	2	1	3								

Tableau 3.2 : Exemple individus.

A partir du tableau 3.1, on peut déduire que la fragmentation de l'entrepôt fera sur l'attribut yearlevel, monthlevel et quarterlevel.

La représentation en tableau multidimensionnel peut être utilisée pour calculer le nombre de tous les schémas de fragmentation possibles (dénoté par H) :

$$H=2^{(\sum_{I=1}^{k} ni)}$$

Où *k* et n_i *représente* le nombre d'attributs de fragmentation et le nombre de sous domaines de l'attribut A_i, respectivement.

Si on considère les trois attributs de fragmentation : yearlevel, monthlevel et quarterlevel (de l'exemple précédent), le nombre de schémas possibles est : $2^{(2+12+4)}=262144$.

4.3 Sélection des individus

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et l'éliminer les mauvais, pendant le passage d'une génération à une autre, ce processus est basé sur la performance de l'individu. L'opérateur de sélection doit être conçu pour donner également une chance aux mauvais éléments, car ces éléments peuvent, par croisement ou mutation, engendrer une descendance pertinente par rapport au critère d'optimisation. Il existe différentes techniques de sélection par rapport la sélection (uniforme, par tournoi, roulette)[33].

Dans notre algorithme nous avons choisi la sélection aléatoire et uniforme où chaque individu *i* a la même probabilité $Prob(i)=1/T_{pop}$ comme tous les autres individus.

4.4 Types de croisements

Plusieurs types de croisement peuvent être envisagés : le croisement mono-point (1 seul point de croisement par individu), le croisement multipoints (2 ou plusieurs points de croisement par individu), le croisement uniforme (opération bit à bit entre les individus). Ont choisi d'utiliser un croisement multipoints pour la raison suivante :

Nous savons depuis la création d'un individu que les prédicats sont placés les uns à la suite des autres dans le vecteur d'entiers. Les individus sont croisés une fois sur chaque prédicat, pour ne pas désavantager le croisement d'un prédicat par rapport à un autre[1]. On sait en effet que chaque prédicat dispose d'un nombre d'entiers différents. Si on n'effectue qu'un croisement par individu le prédicat qui a un grand nombre d'entiers (par exemple, l'attribut Quarterlevel de la table de dimension TimeLevel aura autant d'entiers que d'âges) aura une probabilité de croisement supérieure à un attribut ayant un domaine de faible cardinalité, comme l'attribut yearlevel (1995 et 1996). Cette opération de sélection est effectuée tant que la population n'a pas atteint le nombre convenable d'individus.

4.5 Mutation

L'analogie avec le monde vivant nous amène à nous poser la question de la mutation. Dans la nature, les mutations sont fréquentes et entraînent des variations plus ou moins marquées de l'individu. Les mutations s'effectuent aux niveaux des attributs de notre individu. Suivant un nombre aléatoire, nous choisissons de muter un ou plusieurs attributs de l'individu. On peut voir sur le schéma suivant la mutation d'un attribut.

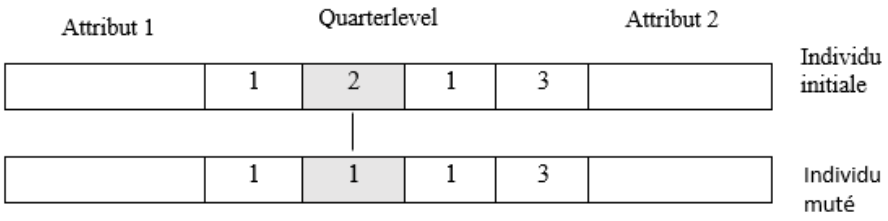


Figure 3.10 : Exemple Mutation.

5. Conclusion

Dans ce chapitre, nous avons au premier lieu présenté les deux metaheuristique tel que les algorithmes génétiques et la méthode descente. En deuxième temps nous avons adapté ces deux méthodes pour résoudre notre problème sujet d'étude. Le prochain chapitre sera consacré à la réalisation de notre application et la présentation des résultats obtenus.

Chapitre 04 : Réalisation et Expérimentations

1. Introduction

Ce chapitre est consacré à présentation du Bank essai benchmark et la description des requêtes utilisé. Par la suite nous présentons les organigrammes des solutions par algorithme génétique et par hybridation d'algorithme génétique avec méthode descente. On conclure par présentation d'application qui comporte la visualisation d'entrepôts de données et des requête et application d'un algorithme génétiques et méthode descente.

2. Banc d'essai Benchmark

2.1 Définition

L'évaluation des performances des SGBDs en général, et les tests d'efficacité des techniques d'optimisation des performances en particulier, sont généralement effectués à l'aide de bancs d'essais. Dans le contexte des entrepôts de données, nous avons utilisé le banc d'essais Benchmark APB-1 release II [26]. Le schéma en étoile que nous avons dégagé à partir de ce banc d'essais est constitué d'une table de faits Actvars et de quatre tables de dimensions, *Prodleve*l, *Custlevel*, *Timelevel* et *Chanlevel* (figure 4.1). Les caractéristiques de chaque table sont représentées dans le tableau 4.1.

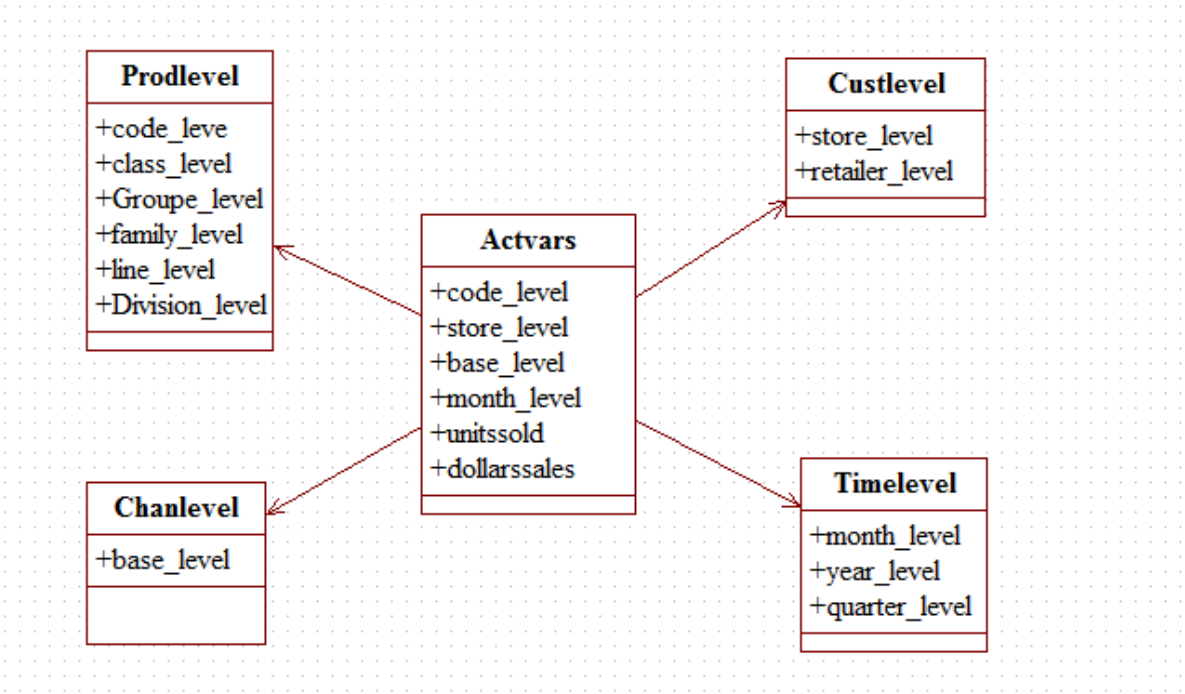


Figure 4.1 : Schéma de l'entrepôt utilisé APB benchmark.

Table	Nombre d'enregistrements	Taille d'enregistrement
Actvars	54537	56
Prodlevel	11073	72
Custlevel	1075	24
Chanlevel	10	12
Timelevel	23	36

Tableau 4.1 : Taille des tables.

2.2 Chargement d'entrepôts de données

Le banc d'essais ABP-1 est livré avec un fichier exécutable APB.exe [13], qui sert à générer les fichiers de données servant à chager l'entrepôt. Il est livré aussi avec un fichier contenant les scripts de création des tables constituant l'entrepôt. Nous avons utiliseé ces scripts pour la création des tables de dimension ProdLevel, TimeLevel, CustLevel et ChanLevel et la table des faits Actvars. Le chargement t de l'entrepôt se fait à l'aide de l'utilitaire fourni avec Oracle10g. La figure 4.2 montre le chargement de l'entrepôt en utilisant l'outil de génération APB.EXE fourni avec le banc d'essais APB-1 [13] et utilitaire fourni avec Oracle.

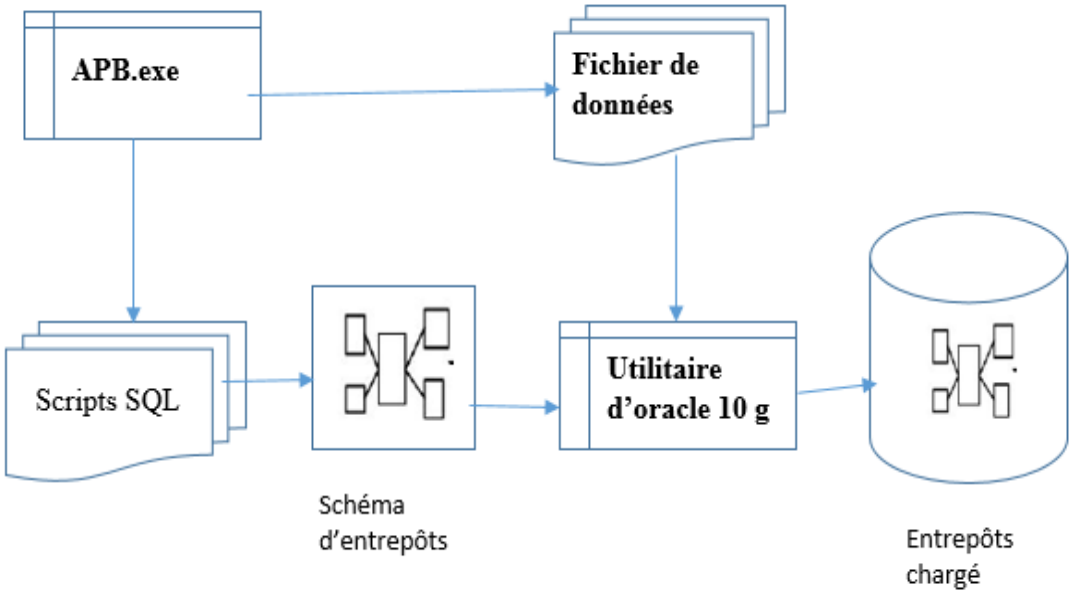


Figure 4.2 : Chargement de l'entrepôt de données.

3. Configuration de la solution

3.1 Type de requêtes prise en considération

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

SELECT [*SGA*], *FC1*(AA), *FC2*(AA),..., *FCn*(AA)

FROM F, D1, D2,... Dk

WHERE PJOIN AND PSEL

[GROUP BY GA]

Où :

- SGA : l'ensemble des attributs retournés dans la réponse à la requête (il peut être vide pour les requêtes retournant un calcul). Ces attributs peuvent être des attributs de dimension et/ou de faits.
- FC1, FC2,…, FCn : les fonctions de calculs (MIN, MAX, COUNT, SUM, AVG).
- AA : l'ensemble des attributs d'agrégation, ces attributs sont généralement des mesures définies dans la table des faits.
- k : le nombre de tables de dimension utilisées par la requête.
- PJOIN : un ensemble de prédicats de jointure entre la table des faits et les tables de dimension sous forme de conjonction. Un prédicat de jointure (équi jointure) a la forme suivante : F. fki = Di .ki . où fk_i , k_i représentent respectivement la clé étrangère et la clé primaire de la table de dimension D_i .
- PSEL : un ensemble de prédicats de sélection sous forme de conjonction, chaque conjonction est constituée de plusieurs prédicats définis sur la même table.
- GA : l'ensemble des attributs de groupement. Ces attributs peuvent être des attributs de faits et/ou de dimension [1].

3.2 Extraction des prédicats

L'analyse des 8 requêtes de la charge du benchmark APB1 utilisé a fait introduire 14 prédicats simples qui sont indiqués dans le tableau 4.2 :

Requête	Attributs
Q1	Y_l
Q1	R_l
Q2	C_l

Q2	F_l
Q2	B_l
Q3	M_l
Q3	F_l
Q4	M_l
Q4	F_l
Q5	F_l
Q6	G_l
Q7	M_l
Q8	Y_l
Q8	M_l

Tableau 4.2 : les prédicats

3.3 Organigramme de la solution basé sur l'algorithme génétique

Avant de passé à étape d'implémentation nous expliquons les étapes de la configuration de la solution dans la Figure 4.3.

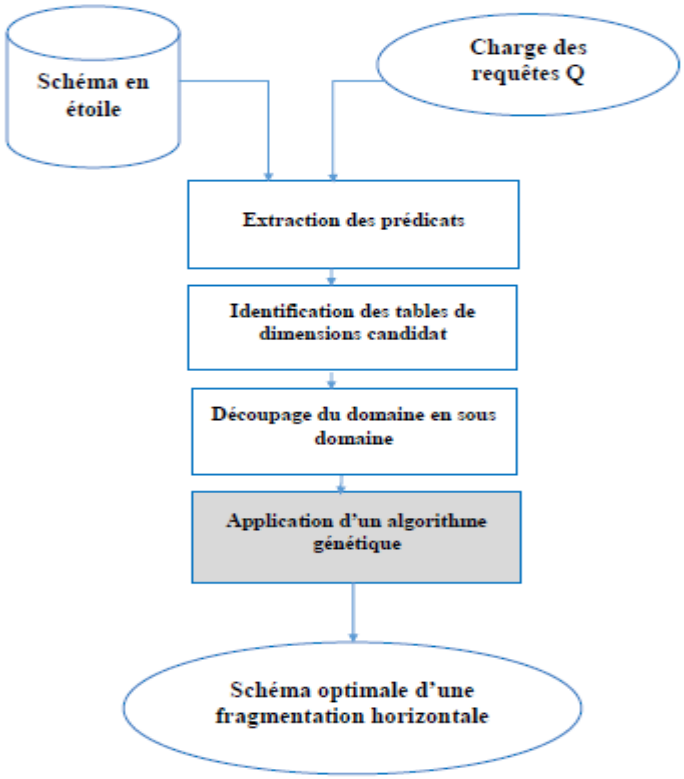


Figure 4.3 : Organigramme de la solution par l'algorithme génétique.

En basons sur l'ensemble d'éléments constitutifs de notre algorithme décrits dans la section 4 du chapitre 3, le pseudo-code de l'algorithme est illustré dans le tableau ci-dessous :

Algorithme 4.1 : Algorithme génétique proposé
1 : Initialiser : P_{taille}, P_{crois}, P_{mut}, $N_{gén}$, $U=0$
2 : Générer aléatoirement P_{taille} séquence des schémas
3 : répéter
4 : i=0
5 : Tant que ($U < (P_{taille}/2)$) faire
6 : sélectionner aléatoirement deux parents de la population
7 : Croisement des deux parents pour obtenir deux enfants par une probabilité P_{crois}
8 : Muter les deux enfants par une probabilité P_{mut}
9 : U=U+1
10 : Fin tant que
11 : Evaluer tous les chromosomes ($2*P_{taille}$, parents et enfants) par la fonction d'évaluation
12 : Ranger les parents et les enfants dans ordre croissant selon leur fonction d'évaluation
13 : Remplacer (P_{taille}, $Best_{sol}$)
14 : i=i+1
15 : Jusqu'à $i=N_{gén}$

Algorithme 4.1 : Algorithme génétique proposé.

Cet algorithme utilise une fonction d'évaluation (fitness) spécifique, Alors que notre problème essentiel réside dans la sélection d'un schéma de fragmentation horizontale dans laquelle le nombre d'entrées/sorties de l'entrepôt de données fragmenté est inférieur le maximum possible par rapport à celui non fragmenté. On évaluer notre schéma de fragmentation par un modèle de cout appui sur la sélectivité des prédicats. Les principaux éléments de notre modèle sont :

- Le cout d'entrepôts de données pour 8 requêtes :

$$N*(||F||*L)/PS$$

  - N : nombre des requêtes
  - F : cardinalité du table de fait,
  - L : la longueur de enregistrement de table de fait,
  - PS : la taille d'une page système

Alors le cout de entrepôts du données est : 8*(54537*56)/64*1024=373 entrées/sorties et pour une seule requête le cout sera égale à 47 entrées/sorties. Notre fonction évaluation est comme suit :

$$\textbf{Min } (\boldsymbol{Cout\_req} * \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} \boldsymbol{sél\_pré})$$

m : nombre des attributs représenter par le chromosome

n : taille de chaque attribut de fragmentation

Cout_att : cout pour une seule requête

Sél_pré : sélectivité de chaque prédicat

3.4 Organigramme de la solution basé sur hybridation d'un algorithme génétique et méthode descente

Après application de l'algorithme génétique on obtient un schéma de fragmentation on applique sur lequel la méthode descente (algorithme 3.2) afin d'améliorer cette schéma la figure 4.4 représente les étapes de la solution par l'algorithme génétique et méthode descente.

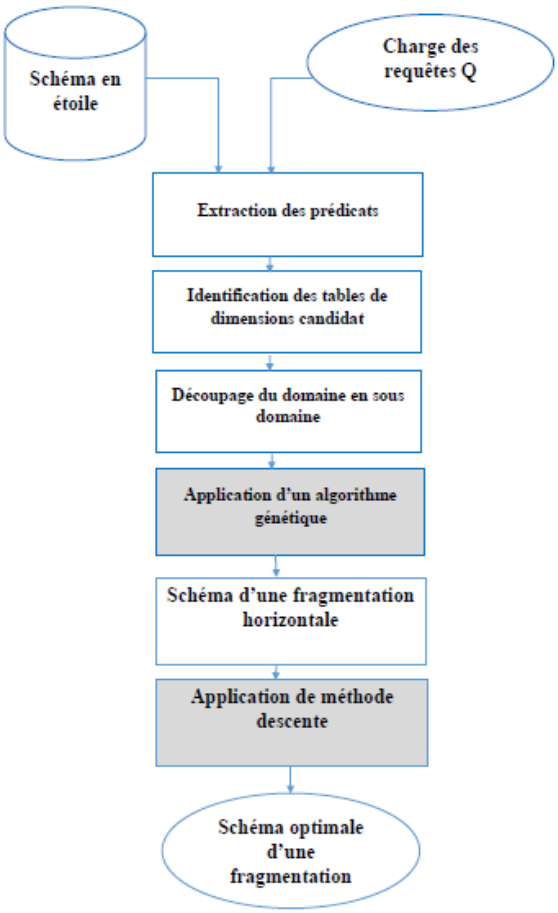
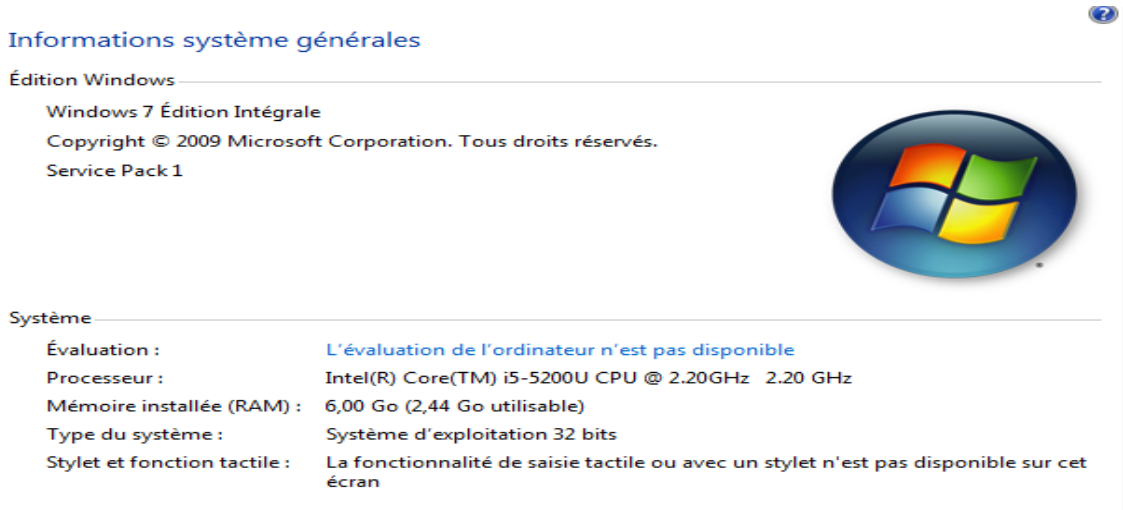


Figure 4.4 : Organigramme de la solution par l'algorithme génétiques et méthode descente.

4. Implémentation

4.1 Environnement de l'application

Notre application va être réalisée sur une machine qui comporte les caractéristiques suivant :



Environnement de développement

Pour mettre en œuvre notre application, nous avons choisis de travailler sous Visual studio 2012 et avec oracle 10g

VISUAL STUDIO 2012

- Cet environnement est une main complète d'outils de développement pour construire des applications Web ASP.NET ; Les services Web XML, des applications bureautiques et des applications mobiles, et comprend une vaste bibliothèque de plus de 4000 cours organisés dans les espaces de noms, qui offrent un large éventail de fonctionnalités qui couvrent de nombreux usages.
- D'autre part, cet environnement offre également de nombreux langages de programmation puissants (C #, Visual Basic, Visual C + +, ... etc. .). Nous avons adopté le langage C # pour les raisons suivantes :
  - La syntaxe de C # est simple et éloquent, utilise moins de 90 mots-clés, il est facile à assimiler.
  - C # est facile d'être reconnu par ses crochets si vous connaissez déjà C, C + + ou Java langues. Si vous avez déjà utilisé une de ces langues, vous pouvez rapidement devenir productifs en C #.
  - C # est un élégant langage orienté objet, et il a de type sécurisé qui permet aux développeurs de créer une large gamme d'applications sûres et fiables qui s'exécutent sur le. NET Framework.

- La syntaxe C # simplifie beaucoup des complexités de C + + en fournissant des fonctionnalités puissantes telles que les types des valeurs nulles initialement, les énumérations, les délégués, les méthodes anonymes, et de la mémoire accès directs qui ne figurent pas dans Java.
- C # prend en charge également les méthodes et les types génériques qui améliorent la sécurité et la performance de types.

Enfin, nous notons que Visual Studio 2012 nous a également fourni des classes prédéfinies et c'est un éditeur graphique permet de créer facilement l'interface graphique grâce à une barre d'outils permettant d'ajouter des composants à la fenêtre de l'application, et de modifier leurs propriétés.

Oracle 10g

Oracle a de nombreux avantages et caractéristiques qui le rend populaire et le rend aussi la plus grande société de logiciels d'entreprise du monde ainsi Un aspect important est bases de données Oracle ont tendance à être rétro-compatible. Oracle prend un rôle de premier plan en raison de certaines des raisons :

- Oracle est utilisé pour presque toutes les grandes applications et l'une des principales applications dans lequel Oracle prend sa présence importante est la banque. En fait dix des 10 banques mondiales utilisent les applications Oracle c'est parce que l'oracle offre une puissante combinaison de complets, des applications d'entreprise pré-intégrées, y compris des fonctionnalités spécifiquement conçues pour les banques et la technologie.
- Oracle est une base de données qui répond très bien avec d'excellentes performances dans des environnements exigeants [34].

4.2 L'architecture de l'application

Nous avons utilisé la programmation orienté objet (POO) parce qu'il offre une lisibilité de manipulation du code source tel que maintenance, détection des erreurs et aider de corriger, débogage …etc.

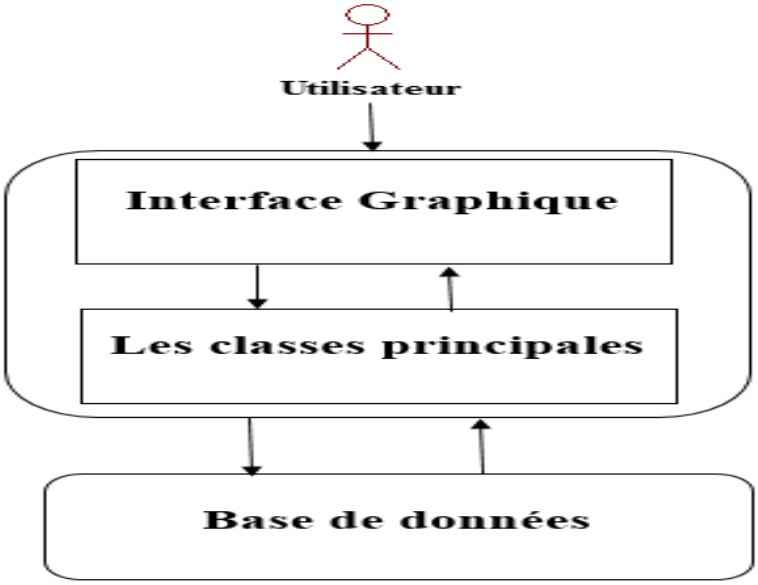


Figure 4.5 : Architecture d'application.

4.2.1 Interface Graphique

C'est une simple interface graphique permet aux utilisateurs une meilleur interaction avec application. Nous avons essayé de donner un aspect visuel et attrayant à travers une interface graphique ergonomique.

4.2.2 les classes principales

La figure 4.8 représente les classes principales utilisées dans la programmation ou chaque classe ayant ses propres attributs et méthodes.

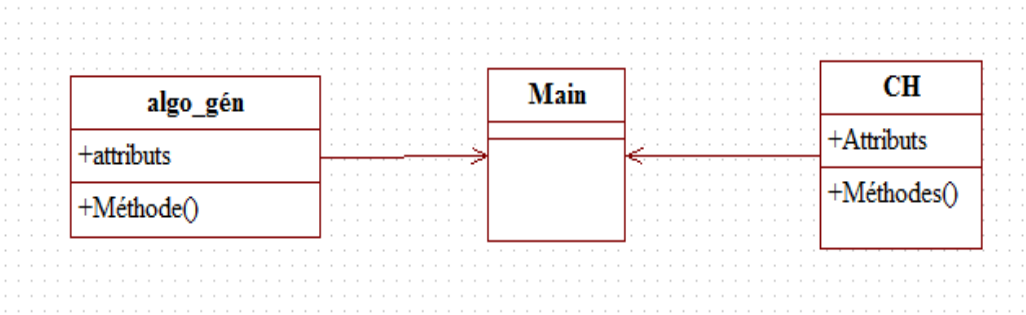


Figure 4.6 : C lasses principales de l'application.

4.3 Les interfaces de l'application

❖ A partir de cette interface on accède à l'interface principale de notre application

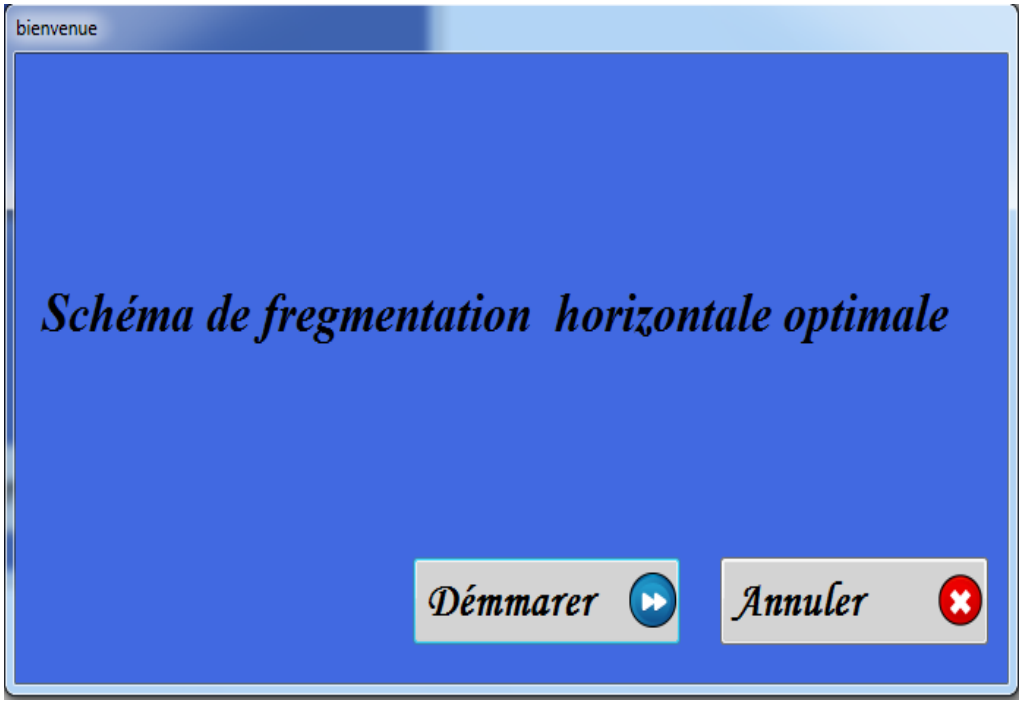


Figure 4.7 : Interface de démarrage.

❖ Interface principale de notre application contient un menu. Ce dernier offre deux possibilités soit la visualisation ou bien l'optimisation.

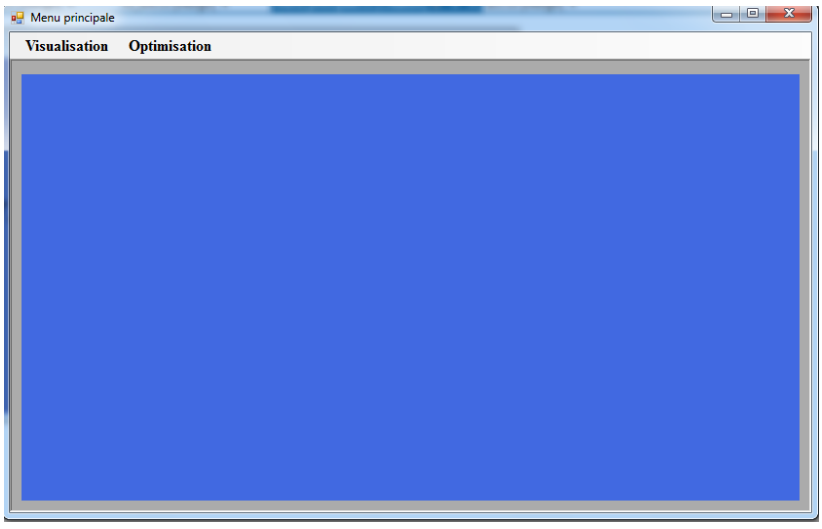


Figure 4.8 : Interface principale de l'application.

❖ La visualisation consiste à visualisation d'entrepôts de données et visualisation des requêtes.

1) La visualisation de entrepôts de données consiste à prendre une idée sur les tables de dimensions et la table de fais et leur nombre enregistrements.

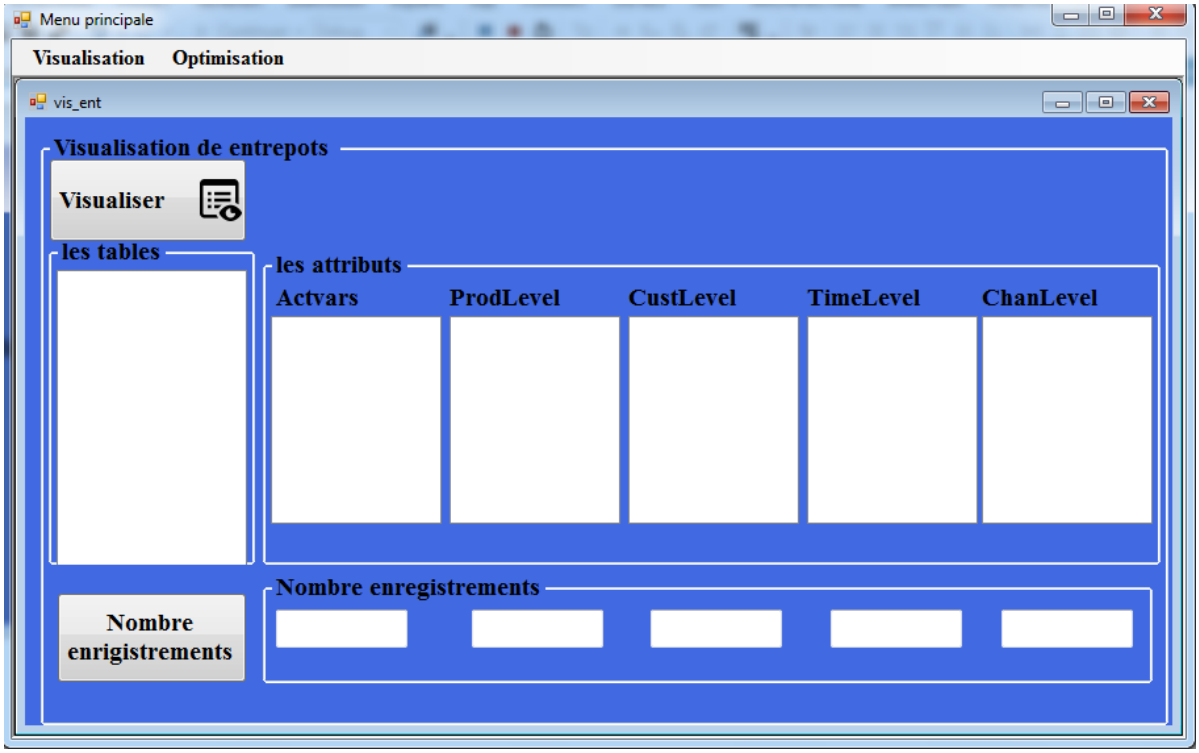


Figure 4.9 : Visualisation d'entrepôts de données

2) La visualisation des requêtes consiste à visualiser un ensemble des requêtes et leurs attributs de fragmentation et en fin affectation de chaque attribut a son table de faits.

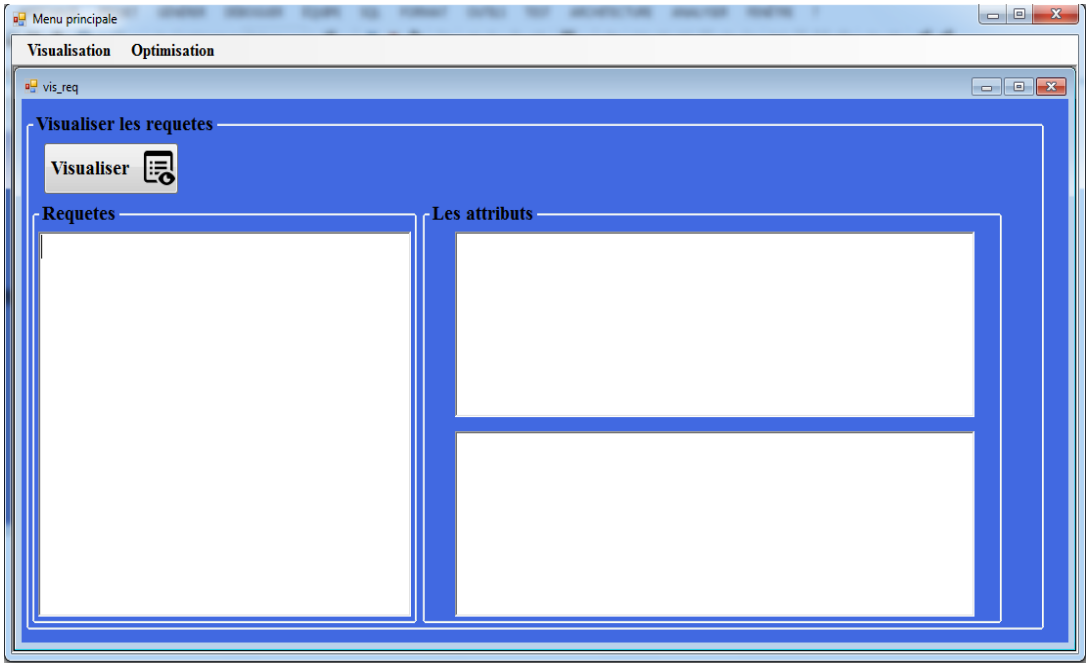


Figure 4.10 : Visualisation des requêtes.

❖ Optimisation consiste à application d'un algorithme génétique et méthode descente pour obtenir un schéma de fragmentation optimale.

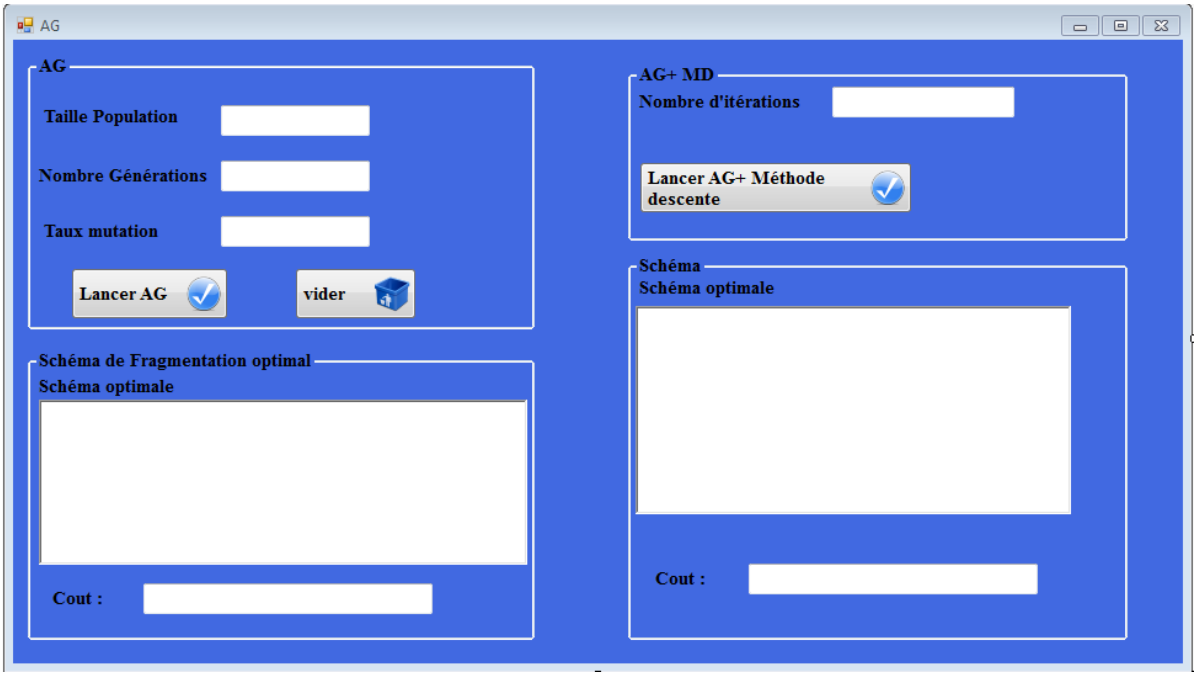


Figure 4.11 : Optimisation par AG et AG + Méthode descente.

4.4 Résultats obtenus

Pour montrer la performance et l'efficacité pour les algorithmes développés, nous avons réalisé une série d'expérimentations pour calculer le coût (nombre d'entrées/sorties) retourné par chacune d'elles et ainsi le taux d'amélioration. Les résultats obtenus sont dressés dans les tableaux 4.3 et 4.4 et les figures 4.12 et 4.13 ci-dessous :

- **Cas 1 :**

Nbr Gén/Itérations	AG	AG+Méthode descente	Taux d'amélioration
50	294	**94**	68%
100	339	**140**	58%
150	250	**94**	62%
200	430	**187**	56%

Tableau 4.3 : Résultats de comparaison entre AG et AG + Méthode descente

(*Taille population=30, Pmut=0.01*)

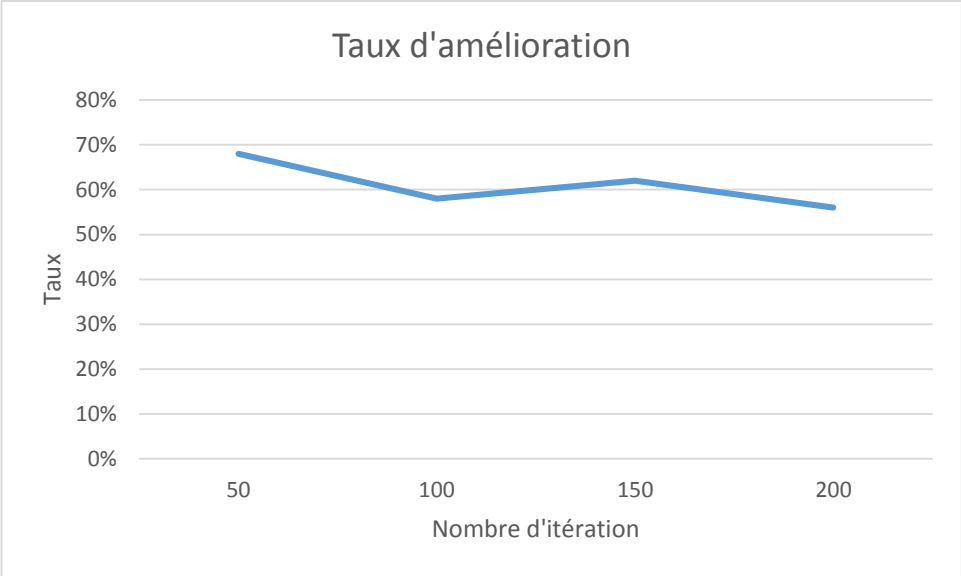


Figure 4.12 : Evolution de taux d'amélioration en fonction de nombre d'itérations (*Taille population=30, Pmut=0.01*).

Notons que le taux d'amélioration est donné par la formule suivante :

$$Taux\ d'amélioration = \frac{Coût(AG) - Coût(AG + Methode\ descente)}{\text{Coût(AG)}} \times 100$$

- **Cas 2 :**

Nbr Gén/Itérations	AG	AG+Méthode descente	Taux d'amélioration
10	288	**47**	83%
30	219	**48**	78%
50	266	**94**	64%
70	311	**140**	54%

Tableau 4.4 : Résultats de comparaison entre AG et AG + Méthode descente (*Taille population=100, Pmut=0.01*).

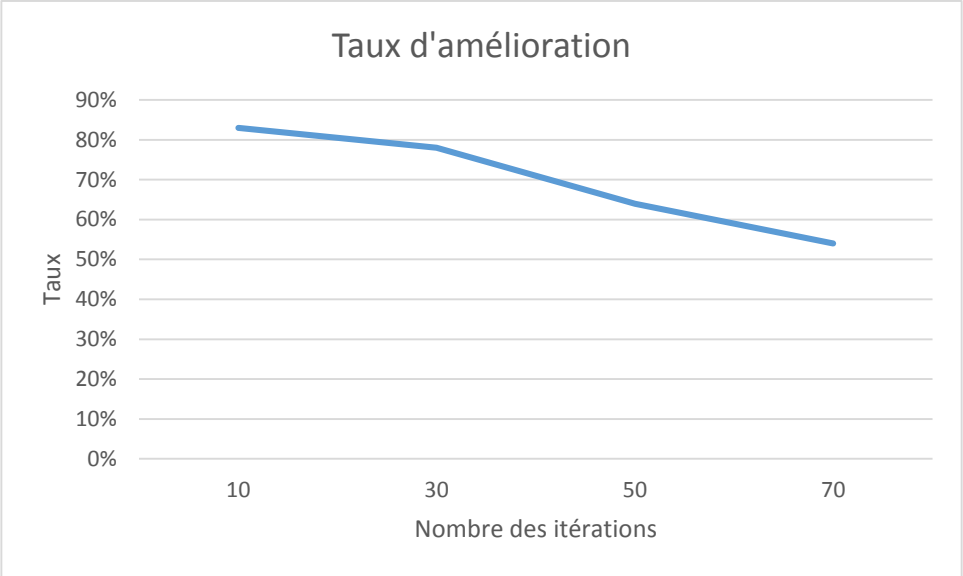


Figure 4.13 : Evolution de taux d'amélioration en fonction de nombre d'itérations (*Taille population=100, Pmut=0.01*).

Les résultats de l'étude Comparative entre AG et AG + Méthode descente sont présentés dans les figures 4.14 et 4.15 ci-dessous :

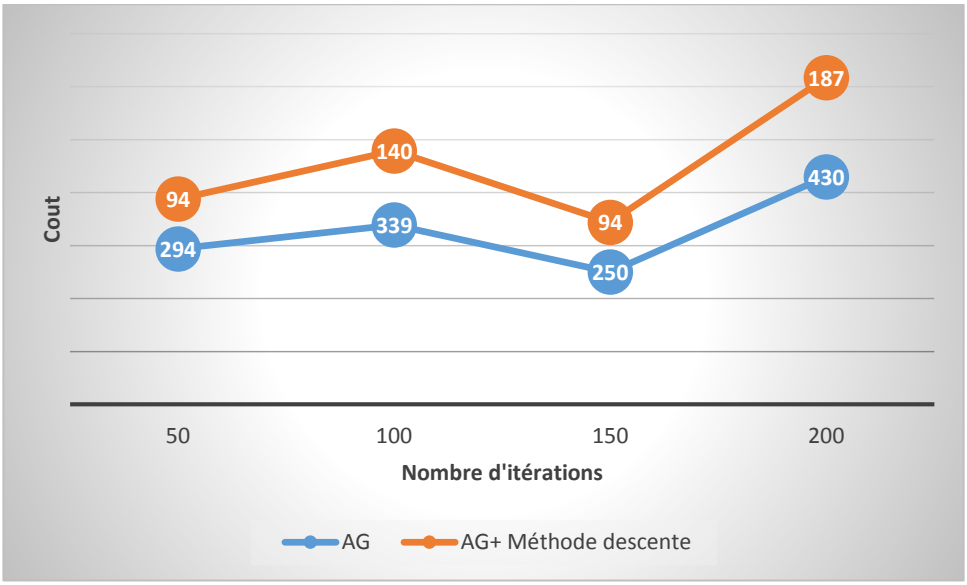


Figure 4.14 : Coût du schéma optimal donné par AG et AG + Méthode descente (*Taille population=30, Pmut=0.01*).

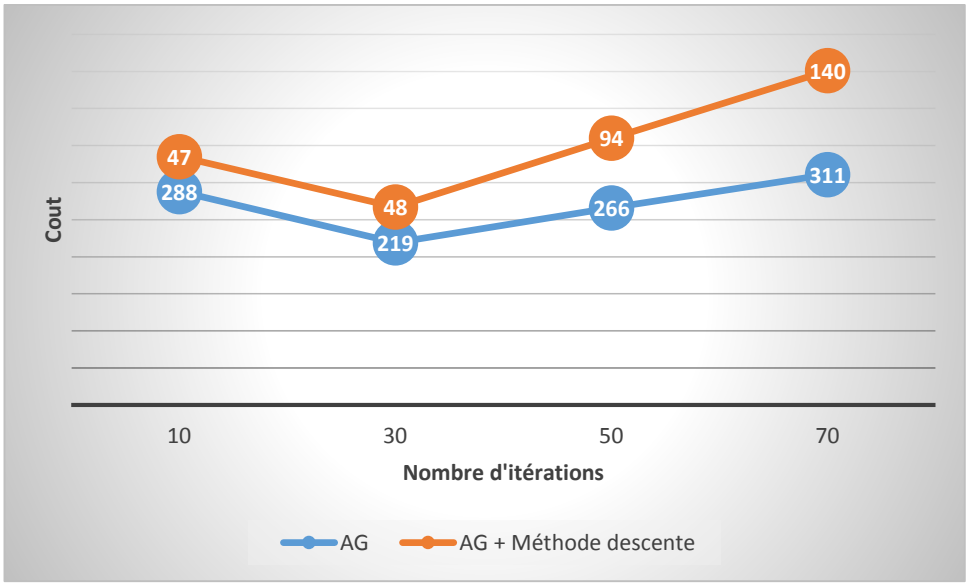


Figure 4.15 : Coût du schéma optimal donné par AG et AG + Méthode descente (*Taille population=100, Pmut=0.01*).

Analyse des résultats

L'analyse des résultats présentés dans le tableau 4.3, 4.4 et les figures 4.12, 4.13 montrent clairement que AG + méthode descente améliore grandement le coût de schéma de fragmentation beaucoup plus que AG seulement. Cette amélioration est largement lié en premier lieu, à la perturbation qu'on a fait au niveau de la configuration du schéma de fragmentation et en deuxième temps au nombre d'itérations alloué à la phase de la méthode de descente et ainsi à la qualité de la solution retournée par l'algorithme génétique utilisé qui a exploité et diversifié un espace de recherche de solutions donné par la population initiale.

5. Conclusion

Dans ce chapitre, en premier temps nous avons présenté le banc essai benchmark (schéma en étoile de l'entrepôt de données et leur chargement), par la suite nous expliquons les étapes de configuration de la solution. Enfin, nous avons présenté l'ensemble d'interfaces utilisés dans notre application.

Conclusion générale

Conclusion générale

Dans le cadre de ce travail de master, nous avons traité le problème de sélection d'un schéma de fragmentation optimale. Au cœur de ce mémoire nous avons présenté les différentes étapes de la modélisation de la solution, pour mieux cerner la problématique posée, nous avons commencé par la présentation générale des notions concerne les entrepôts de données et une citation des techniques d'optimisation des requêtes. En deuxième temps, nous avons passé à l'explication en détail de la technique d'optimisation basée sur la fragmentation horizontale. Pour guider notre choix de résolution, nous avons présenté d'une manière générale les algorithmes génétiques et ces opérateurs génétiques tels que codage, mutation et croisement et la méthode descente puis, nous avons adopté ces méthodes pour résoudre le problème de sélection d'un schéma de fragmentation optimale.

Nous avons choisi le langage c# pour écrire et développé notre application, on a utilisé oracle 10g pour la création de notre Schéma en étoile et on a utilisé les données fournis par banc essai benchmark.

Comme perspective à ce travail, l'utilisation d'autres métaheuristiques hybrides est désirée afin de trouver une solution approchée au problème de sélection d'un schéma de fragmentation optimale. On peut aussi augmenter la charge de requêtes utilisée pour montrer la performance de notre application. Enfin, et pour évaluer la technique de fragmentation développée, il est important d'utiliser le schéma générer par l'application pour fragmenter l'entrepôt de données ce qui réduit le temps d'exécution des requêtes.

En effet, ce travail étant une œuvre humaine n'est pas une solution unique et parfait, c'est pourquoi nous restons ouverts à toutes les critiques et nous somme prêts à recevoir toutes suggestions.

Bibliographie

[1] E. Ziyati, « optimisation de requêtes olap en entrepôts de données approche basée sur la fragmentation génétique », thèse de doctorat, université mohammed v –agdal rabat – maroc, mai 2010.

[2] http://www.upmf-grenoble.fr/ , consulté : le 05/04/20016

[3] D. Garar, « compression dans les entrepôts de données pour l'amélioration des performances », thèse de doctorat, université du québec à montréal, janvier 2013.

[4] B. Espinasse, « introduction aux entrepôts de données », université de marseille, septembre 2013.

[5] K. Boukhalfa, « de la conception physique aux outils d'administration et de tuning des entrepôts de données », thèse de doctorat, école doctorale : sciences pour l'ingénieur et aéronautique, juillet 2009.

[6] N. Zanoun, « la construction en ligne des tables individus variables par apprentissage automatique numérique (pmc) », mémoire de magister, université d'oran, 2010.

[7] A. Daâs ,« optimisation des requêtes dans le data warehouse », mémoire de magister, université ferhat abbas – sétif, 2012.

[8] E. F. codd , providing olap(on-line analytical processing)to user analystes : an it mandate, technical report,1993.

[9] O. teste, modélisation et manipulation d'entrepôts de données complexes et historisées, université paul sabatier - toulouse iii, 2000.

[10] L. Bellatreche, « la conception physique des data warehouses », ladjel bellatreche, lisi/ensma ,2006.

[11] H. jagadish, l. v. s. lakshmanan, and d. srivastava. «what can hierarchies do for your data warehouses», proceedings of the international conference on very large databases, pages 530–541, september 1999.

[12] R. Bouchakri « une approche dirigée par la classification des attributs pour

fragmenter et indexer des entrepôts de données », mémoire de magister, école nationale supérieure d'informatique (esi), 2009.

[13] The olap report - http://www.olapreport.com/fasmi.htm, 2004.

[14] M. trinidad et S. encinas, « entrepôts de données pour l'aide à la décision médicale : conception et expérimentation », maría trinidad serna encinas, université joseph fourier, le 27 juin 2005.

[16] P. vassiliadis and Timos k. sellis, « a survey of logical models for olap databases», sigmod record, 28(4): pp 64–69, 1999.

[17] H. gupta and I. s. mumick, « selection of views to materialize in a data warehouse », ieee trans. on knowledge and data eng, 17(1): pages 24–43, january 2005.

[18] P. valduriez. join indices, «acm transactions on database systems», 12(2) :218–246, june 1987.

[19] Red breck systems , «star schema processing for complex queries». white paper. Juillet 1997.

[20] S. chaudhuri et v. narasayya, «index merging», proceedings of the international conference on data engineering (icde) », pages 296-303, march 1999.

[21] H. mahboubi, « optimisation de la performance des entrepôts de données xml par fragmentation et répartition », thèse de doctorat, université lumière lyon 2, 08/12/2008.

[22] M. barr, « approche dirigée par les fourmis pour la fragmentation horizontale des entrepôts de données relationnels », thèse de magister, e.s.i, 2008.

[23] L. bellatreche et K. boukhalfa, « sélection de schéma de fragmentation horizontale dans les entrepôts de données formalisation et algorithmes », rapport de recherche, lisi/ensma poitiers, université de laghouat, algérie 2006.

[24] M. t. özsu et P. valduriez. « principles of distributed database systems» ,second edition,prentice hall, 1999.

[25] L. bellatreche, K. boukhalfa, et H. i. abdalla. saga « a combination of genetic and simulated annealing algorithms for physical data warehouse design»,in 23rd british national conference on databases, (212-219), july 2006.

[26] R. bouchakri, « conception physique statique et dynamique des entrepôts de données», thèse de doctorat, école nationale supérieure d'informatique (algérie) et école nationale supérieure de mécanique et d'aérotechnique (france),17 septembre 2015.

[27] T. Vallé et M. Yıldızo˘glu, « Présentation des algorithmes génétiques et de leurs applications en Economie», Université Montesquieu Bordeaux IV ,7 septembre 2001.

[28] S. Krour, « optimisation des paramètres d'une cellule photovoltaïque par les algorithmes génétiques », mémoire de magister, université de Farhat abbas 1,21/12/2014.

[29] N. Talbi, « Conception des Systèmes d'Inférence Floue par des Approches Hybrides : Application pour la Commande et la Modélisation des Systèmes Nonlinéaires», thèse de doctorat, Université de Constantine 1, le 25 /02 / 2014.

[30] A. Nafi, « la programmation pluriannuelle du renouvellement des réseaux d'eau potable », thèse de doctorat, université Louis Pasteur, Strasbourg I, 04/12/2006.

[31] F. Souam ait elhadj, « Approche de détection de communautés chevauchantes dans réseaux bipartis », thèse de doctorat, université mouloud Mammeri Tizi-Ouzou, 20/10/2013.

[32] A. Gherboudj, « Méthodes de résolution de problèmes difficiles académiques », thèse de doctorat, Université de Constantine2, 2013.

[33] http://www.learn.geekinterview.com/database/oracle/advantages-of-using -oracle.html, consultu le : 13/04/2016.

[34] H. E. Seribli, « développement et implémentaion d'un solveur Bio_inspiré pour l'alignement de séquences Biologiques », mémoire de master, université de m'sila ,14/06/2015.

Annexe

Ensemble les requêtes utilisé [13]:

Q1 : select Customer_Level, Product_level, Time_level, from ACTVARS A, CUSTLEVEL C,PRODLEVEL P,TIMELEVEL T where A.customer_level=C.store_level and A.prodcut_level=P.code_level and A.time_level=T.TID and T.year_level='1996' and C.retailer_level ='NXEYFSIQE3JM' and P.line_level='MJ1F1U1EG009' group by Customer_level, Product_level, Time_level	Q2 : Select prodlevel.code_level, prodlevel.family_level, chanlevel.all_level, sum(Actvars.dollarsales), Count(actvars.dollarsales), count(*) from actvars, chanlevel, prodlevel where chanlevel.base_level = actvars.channel_level and prodlevel.code_level = actvars.product_level and prodlevel.family_level = 'IKKIOVKDGTWO' group by prodlevel.code_level, prodlevel.family_level, chanlevel.all_level
Q3 : SELECT code_level, sum(dollarsales), sum(UnitsSold) FROM actvars, timelevel, prodlevel WHERE product_level = code_level and month_level between '01' and '03' and family_level = 'M8VWHZM5BS2N' group by code_level	Q4 : SELECT base_level, sum(dollarsales) , sum(UnitsSold) FROM actvars, timelevel, prodlevel, dw.chanlevel WHERE product_level = code_level and channel_level = base_level and month_level between '199510' and '199612' and family_level = 'M8VWHZM5BS2N' group by base_level
Q5 : SELECT code_level, sum(dollarsales) FROM actvars, prodlevel, chanlevel WHERE product_level = code_level and channel_level = base_level and family_level = 'M8VWHZM5BS2N' group by code_level	Q6: SELECT code_level, sum(dollarsales) FROM actvars, timelevel, prodlevel WHERE product_level = code_level and group_level = 'SV5L8W0J8UMZ' group by code_level
Q7: SELECT product_level,	Q8: SELECT product_level,

sum(dollarsales), sum(UnitsSold) FROM actvars, prodlevel, dw.timelevel WHERE product_level = code_level and month_level = '199503' and class_level = 'KPB8DCM3VDMK' and division_level = 'DVWQ0OHI8M00' Group by product_level	sum(dollarsales), sum(UnitsSold) FROM actvars, prodlevel, timelevel, dw.chanlevel WHERE product_level = code_level and channel_level = base_level and year_level = '1995' and month_level='199503' and class_level = 'G7DPNCB3JSFW' droup by product_level

Les sélectivités utilisées pour évaluation des schémas [1] :

codelevvel	**0**	0.0022222221
codelevvel	**1**	0.0022222221
codelevvel	**2**	0.9933333335
codelevvel	**3**	0.9933333335
baselevel	**0**	0.0122222221
baselevel	**1**	0.0122222221
baselevel	**2**	0.9133333335
baselevel	**3**	0.9133333335
monthlevel	**0**	0.117647059
monthlevel	**1**	0.117647059
monthlevel	**2**	0.117647059
monthlevel	**3**	0.117647059
monthlevel	**4**	0.117647059
monthlevel	**5**	0.117647059
monthlevel	**6**	0.058823529
monthlevel	**7**	0.058823529
monthlevel	**8**	0.058823529
monthlevel	**9**	0.058823529
monthlevel	**10**	0.058823529
monthlevel	**11**	0.058823529
yearlevel	**0**	0.705882353
yearlevel	**1**	0.294117647

retailerlevel	**0**	0.014622771
retailerlevel	**1**	0.01404321
retailerlevel	**2**	0.014266118
retailerlevel	**3**	0.957067901
familylevel	**0**	0.017756516
familylevel	**1**	0.018710562
familylevel	**2**	0.017797668
familylevel	**3**	0.018065844
familylevel	**4**	0.92766941