



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



DÉPARTEMENT 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: BARKA Ahlem

SUJET

**Méthodes bio-inspirées pour la fragmentation d'entrepôt
de données : Etude et évaluation**

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

Dr. H. Belouadeh
Dr. C. Lamiche
A. Guerna

Université de M'sila
Université de M'sila
Université de M'sila

Président
Rapporteur
Examineur

Promotion : 2015/2016

Remerciements

Mes remerciements sont d'abord destinés à mon promoteur **Dr. C. Lamiche**, enseignant au département d'informatique de l'Université de M'sila, pour avoir accepté d'encadrer ce travail.

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

Je remercie tous les enseignant qui Nous ont orientés pendant tout Notre parcours.

A tous les gens qui ont aidés du pré et du loin pour réaliser ce modeste travail.

Merci à tous et à toutes

Table des matières

-Liste des figures.....	i
-Liste des Tables.....	iii
-Liste des Algorithmes	iv
-Introduction générale	1
CHAPITRE I : Entrepôts de données : Concepts Fondamentaux.	
1. Introduction.....	2
2. L'entrepôt de donnée (ED).....	2
2.1.Définition.....	2
2.2.Caractéristiques.....	2
2.3.Les types de données.....	3
2.4.L'architecture d'entrepôt de données.....	4
2.4.1. Sources de données.....	5
2.4.2. Processus Extract, Transform, Load (ETL).....	5
2.4.3. Les magasins.....	5
2.4.4. Les applications clients.....	6
3. Les modèles des entrepôts de données.....	7
3.1.La modélisation conceptuelle.....	7
3.1.1. Concept de fait, de dimension et de hiérarchie.....	7
3.1.2. Modèles en étoile, en flocon et en constellation.....	7
3.1.2.1. Modèle en étoile.....	7
3.1.2.2. Modèle en flocons.....	8
3.1.2.3. Modèle en constellation.....	9
3.2.Modélisation logique.....	9
3.2.1. les modèles multidimensionnels.....	9
3.2.1.1. Modèle R-OLAP.....	10
3.2.1.2. Modèle M-OLAP.....	11
3.2.1.3. Modèle H-OLAP.....	11
3.3.La conception physique des entrepôts de données.....	11
4. Choix des techniques d'optimisation.....	11
4.1.Les techniques redondantes.....	12
4.1.1. Les index.....	12

4.1.1.1. Index Bitmap.....	12
4.1.1.2. Index B-Arbre.....	13
4.1.1.3. Les index de jointure binaires.....	14
4.1.2. La Fragmentation verticale.....	14
4.1.3. Les vues matérialisés.....	15
4.2. Les techniques non redondantes.....	16
4.2.1. La fragmentation horizontale.....	16
4.2.2. La fragmentation mixte.....	17
5. Conclusion.....	
CHAPITRE II : La fragmentation horizontale dans les entrepôts de données	
1. Introduction.....	18
2. La fragmentation horizontale FH.....	18
2.1. Fragmentation primaire.....	18
2.2. Fragmentation dérivée.....	19
2.3. Les avantages de la fragmentation horizontale.....	20
3. Méthodologie de fragmentation horizontale dans les entrepôts de données.....	20
4. Le Procédure de FH.....	21
4.1. Module de gestion des requêtes	22
4.2. Règles de correction de la fragmentation horizontale.....	22
5. Le problème de sélection d'un schéma de fragmentation.....	22
5.1. Travaux de recherche sur le problème de fragmentation.....	23
5.1.1. Travaux de DATTA et AL. (1999).....	23
5.1.2. Travaux de Wu et Buchmaan (1997).....	23
5.1.3. Travaux de NOAMAN & BARKER (1999).....	23
5.1.4. Travaux de Bellatreche (2000).....	23
6. Algorithmes de sélection d'un schéma de FH.....	23
6.1. Algorithmes basées sur les prédicats.....	24
6.2. Algorithmes basés sur l'affinité.....	24
6.3. Algorithmes à base de modèle de coût.....	27
7. Conclusion.....	28
CHAPITRE III : Métaheuristiques et fragmentation d'entrepôt de données.	
1. Introduction.....	29
2. Description du problème.....	29

3. Les Algorithmes génétiques.....	30
3.1. Définition des algorithmes génétique.....	30
3.2. Principe de base.....	30
3.2.1. Le codage des individus.....	32
3.2.2. L'évaluation d'un individu.....	32
3.2.3. Les opérateurs génétiques.....	33
3.2.3.1. L'opérateur de sélection.....	33
3.2.3.2. L'opérateur de croisement ou crossover.....	34
3.2.3.3. L'opérateur de mutation.....	35
4. Les recuits simulés.....	35
5. Application de GA au Problème de sélection de schéma de FH.....	36
5.1. Processus génétique de la fragmentation horizontale.....	36
5.1.1. Codage adopté.....	36
5.1.2. Génération de la population initiale.....	37
5.1.3. Sélection des individus.....	37
5.1.4. Croisements de individus.....	37
5.1.5. Mutation.....	38
5.1.6. Fonction d'évaluation : <i>fitness</i>	38
5.1.7. Critère d'arrêt.....	38
6. L'application de recuit simulé RS sur la solution de GA.....	40
7. Conclusion.....	41
CHAPITRE IV : Réalisation et expérimentation	
1. Introduction.....	42
2. Les outils de développement.....	42
2.1 Microsoft Visual Studio.....	42
2.1.1 Microsoft Visual Studio 2012.....	42
2.1.2 Langage de programmation C#	43
2.2 Oracle.....	43
2.2.1 Oracle Développer.....	44
3. Implémentation	44
3.1 L'entrepôt de données (Entrepôt de banc d'essai Benchmark).....	44
3.1.1. Types de requêtes prises en compte	45
3.1.2. Vecteur de sélectivité	46
3.2. Implémentation de solution	46

3.2.1. Procédure de solution	46
3.2.2. L'architecture de l'application.....	47
3.2.3. les cas d'utilisation de l'interface	47
4. Présentation graphique	48
5. Résultats obtenus.....	49
5.1 Analyse des résultats.....	51
6. Conclusion.....	51
-Conclusion général.....	52
- Bibliographique	53
-Annexes.....	58

Liste des figures

Figure I.1 : Architecture d'un entrepôt de données.

Figure I.2 : Magasins de données (datamarts).

Figure I.3 : Schémas en étoile.

Figure I.4 : Schémas en flocons.

Figure I.5 : exemple d'un cube de données.

Figure I.6 : Table multidimensionnelle (Visualisation d'une 'tranche' du cube).

Figure I.7 : Classification des techniques d'optimisation.

Figure I.8: Index Bitmap.

Figure I.9: Index B-Arbre.

Figure I.10 : Fragmentation verticale.

Figure I.11 : Le processus de sélection des vues matérialisées.

Figure I.12: Fragmentation Horizontale.

Figure I.13 : fragmentation mixte.

Figure II.1 : Exemple d'une fragmentation primaire et dérivée.

Figure II.2 : Approche basée sur les prédicats.

Figure II.3 : Approche basée sur les affinités.

Figure II.4 : Approche basée sur un modèle de coût.

Figure III.1 : Les cinq niveaux d'organisation d'un algorithme génétique.

Figure III.2 : Exemple de croisement à un point.

Figure III.3 : Exemple de croisement à deux points.

Figure III.4 : Exemple une mutation.

Figure III.5 : Exemple de codage.

Figure III.6 : Exemple de croisement

Figure III.7: Exemple de Mutation.

Figure IV.1 : Visual Studio 2012.

Figure IV.2 : Oracle.

Figure IV.3 : Oracle Développer.

Figure IV.4 : Schéma de l'entrepôt utilisé APB benchmark

Figure IV.5 : Procédure de travail.

Figure IV.6 : Architecture de l'application

Figure IV.7 : Diagramme de cas d'utilisation.

Figure IV.8 : Visualisation d'entrepôt de données.

Figure IV.9 : Analyse des requêtes.

Figure IV.10 : Fenêtre d'application de GA et GA+RS.

Figure IV.11 : Exemple d'application de l'AG et l'AG+RS.

Figure IV.12 : Évaluation de coûts d'exécutions en fonction de nombre d'itérations.

Figure IV.13 : Évaluation de taux d'amélioration en fonction de nombre d'itérations.

Liste des Tables

Table I.1 : Base de données vs Entrepôt de données.

Table I.2 : Comparaison entre OLTP et OLAP.

Table II.1 : Matrice d'usage des sous-domaines de l'attribut Ville.

Table II.2 : Matrice d'affinité des sous-domaines de l'attribut Ville.

Table IV.1 : Taille des tables.

Table IV.2 : Résultats d'exécutions d'AG et AG + RS avec le taux d'amélioration.

Liste des Algorithmes

Algorithme III.1 : Algorithme génétique générique.

Algorithme III.2 : Algorithme génétique Recuit Simulé.

Algorithme III.3 : Algorithme génétique pour la sélection d'un schéma de FH.

Algorithme III.4 : Algorithme Recuit Simulé.

Introduction générale

Introduction générale

De nos jours, les entrepôts décisionnels d'entreprise constituent des ressources critiques. Avec l'évolution des exigences métiers, ils doivent gérer des volumes et des vitesses de données sans cesse grandissants. Les entrepôts sont construits autour d'une table de faits, souvent très volumineuse, qui peut atteindre plusieurs téraoctets, entourée de plusieurs tables de dimensions qui contiennent les axes d'analyse relativement aux indicateurs ou mesures contenus dans la table des faits. Ces derniers sont accédés par des requêtes décisionnelles complexes caractérisées par de multiples opérations de sélection, de jointure et d'agrégation. Pour optimiser ces requêtes, l'administrateur est amené à effectuer une tâche importante dans la conception physique, durant cette phase l'administrateur choisit un ensemble de techniques d'optimisation à sélectionner.

La fragmentation horizontale est une structure non redondante d'optimisation de requêtes OLAP. Dans le cadre de ce mémoire, nous formalisons d'abord le problème de sélection d'un schéma de fragmentation pour un entrepôt de données relationnel comme un problème d'optimisation. Nous proposons ensuite une méthode hybride combinant un algorithme génétique **AG** et un algorithme de recuit simulé **RS**.

Ce mémoire est organisé de la manière suivante : Chapitre 01 est consacré aux définitions des concepts fondamentaux d'entrepôt de données, aussi aux techniques d'optimisations des requêtes. Le deuxième chapitre présente d'une façon détaillée la technique d'optimisation *Fragmentation Horizontale* **FH** dans les entrepôts de données. Et concerne le troisième chapitre nous avons spécifié le problème générale qui on vas le traité, et après l'application des algorithmes génétique et des recuit simulé sur ce problème, Nous présentons aussi dans ce chapitre le modèle de coût utilisé. Dans le quatrième chapitre, nous avons présenté nos résultats de validation sur un banc d'essai APB-1 et un ensemble de 8 requêtes. Enfin, une conclusion rappelle la problématique étudiée ainsi que les résultats obtenus. Cette conclusion nous a permis de lister quelques perspectives de prolongement de nos travaux.

CHAPITRE I

Entrepôts de données : Concepts Fondamentaux

1 Introduction

L'entrepôt est le lieu de stockage centralisé d'un extrait des sources. Il historise les données utiles pour la décision. Son organisation doit faciliter la gestion efficace des données et la conservation des évolutions. Ils manipulent généralement des données d'une taille de plusieurs *Téraoctets*.

Dans ce chapitre on va présenter d'une part, les concepts fondamentaux de l'entrepôt de données, ses caractéristiques, son architecture et ses principes de modélisation. D'autre part, nous devons donner un aperçu sur les techniques d'optimisation des requêtes dans les entrepôts de données.

2 L'entrepôt de donnée (ED)

2.1 Définition

Un entrepôt de données est une structure informatique dans laquelle est centralisé un volume important de données consolidées à partir des différentes sources de renseignements d'une entreprise et qui est conçue de manière que les personnes intéressées aient accès rapidement à l'information stratégique dont elles ont besoin [1]. Il faut noter que l'entrepôt de données est généralement destiné aux décideurs de l'entreprise souvent non spécialistes de l'informatique. L'entrepôt doit être muni d'interfaces graphiques adaptées à des utilisateurs non informaticiens [7]. Les points clefs garantissant le succès d'un entrepôt de données sont les suivants :

- Les informations d'un entrepôt de données doivent être accessibles et fiables (de qualité).
- La conception d'un entrepôt de données doit répondre à un besoin de ROI¹ élevé (Return On Investment, retour sur investissement).
- La réponse aux demandes très diverses des utilisateurs.
- L'entrepôt de données doit évoluer avec les besoins des utilisateurs et du système d'information [14].

2.2 Caractéristiques

Les caractéristiques d'un entrepôt de données (datawarehouse) peuvent donc se résumer comme suit :

1. Il reprend des données collectées pour diverses finalités.
2. Il les compile et les prépare pour une réutilisation dans un but de data mining.

¹ La notion de ROI est très présente pour mesurer la rentabilité des actions de marketing, notamment dans les domaines du marketing direct et du marketing digital.

3. Il ne modifie pas les données mais les analyse pour créer de l'information nouvelle.
4. Ses finalités sont généralement d'ordre statistique mais peuvent être très variées et entraîner notamment des décisions relatives à des personnes.
5. Son utilisation peut évoluer suivant les besoins [26].

2.3 Les types de données

L'organisation des données doit faciliter l'analyse décisionnelle, cette organisation peut se structurer en quatre classes importantes :

- *Les données détaillées* : représentent les données fraîchement extraites à partir des différentes sources opérationnelles. Elles reflètent donc les événements les plus récents [37].
- *Les données agrégées* : Les données agrégées correspondent à des éléments d'analyse représentant les besoins des utilisateurs. Elles constituent déjà un résultat d'analyse et une synthèse de l'information contenue dans le système décisionnel, et doivent être facilement accessibles et compréhensibles [14].
- *Les données historisées* : Chaque nouvelle insertion de données ne détruit pas les anciennes valeurs, mais crée une nouvelle occurrence munie d'une étiquette temporelle. [37][14].
- *Les métadonnées* : ce sont des données définies sur les données. Ils représentent un dictionnaire unique qui résout l'hétérogénéité des données sources et permet de maintenir leur cohérence [37].

	Systèmes opérationnels	Entrepôt de données
But	Exécution d'un processus métier	Évaluation d'un processus métier
Interaction avec l'utilisateur	Insertion, Modification, Interrogation, Suppression	Interrogation
Données	Courantes	Courantes et Historiques
Usage	Support de l'opération de l'entreprise	Support de l'analyse de l'entreprise
Requêtes	Simple, Prédéterminées	Complexes
Type d'utilisateur	Employé	Décideur
Type d'utilisateur Employé Décideur	Principe de conception Troisième forme normale	Conception multidimensionnelle

Table I.1 : Base de données vs Entrepôt de données [14].

2.4 L'architecture d'entrepôt de données

Généralement, un entrepôt de données est composé de trois phases : la phase de l'intégration, la phase de restructuration des données et la phase de l'exploitation (accès aux données) [13].

➤ Intégration

Cette première étape est assez délicate, car elle consiste à extraire et regrouper les données, provenant de sources multiples, et hétérogènes. Un certain nombre de problèmes est à résoudre à ce niveau : les données doivent être filtrées, triées, homogénéisées et nettoyées [9].

➤ Structuration

Cette étape consiste à réorganiser les données, dans des magasins afin de supporter efficacement les processus d'analyse et d'interrogation, et d'offrir aux différents utilisateurs, des vues appropriées à leurs besoins [14].

➤ Interrogation et Analyse

L'exploitation de l'entrepôt, pour l'aide à la décision peut se faire de différentes façons, dont [9]:

- L'interrogation à travers un langage de requêtes.
- La connexion à des composants de report, pour des représentations graphiques et tabulaires.
- L'utilisation des techniques OLAP (OnLine Analytical Process).
- L'utilisation des techniques de fouille de données (Data Mining).

Ces trois phases sont représentées dans la figure 1 ci-dessous :

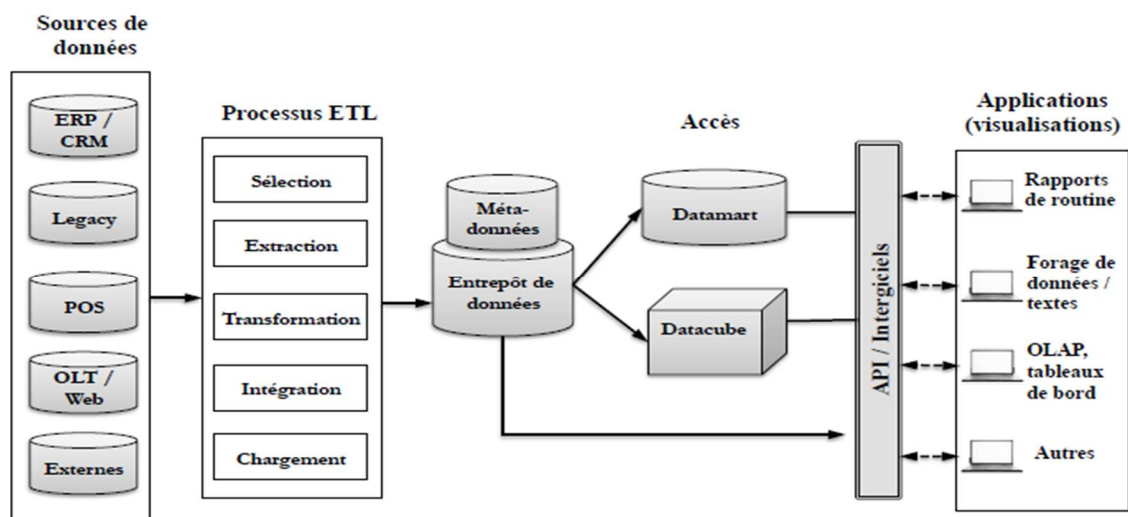


Figure I.1 : Architecture d'un entrepôt de données [40].

2.4.1 Sources de données

- *Enterprise resource planning (ERP)* : Gèrent les processus opérationnels d'une entreprise (ex: ressources humaines, finances, distribution, approvisionnement, etc.).
- *Customer Relationship Management (CRM)* : Gèrent les interactions d'une entreprise avec ses clients (ex: marketing, ventes, après-vente, assistance technique, etc.).
- *Systèmes legacy* : Matériels et logiciels obsolètes mais difficilement remplaçables.
- *Point of sale (POS)* : Matériels et logiciels utilisés dans les caisses de sorties d'un magasin.
- *Externes* : données concurrentielles achetées, données démographiques, ... [40].

2.4.2 Processus Extract, Transform, Load (ETL)

Ce processus constitue la phase de migration des données de production dans le système décisionnel après qu'elles ont subi des opérations de sélection, de nettoyage et de reformatage dans le but de les homogénéiser. Cette phase constitue une étape importante et très chronophage dans la mesure où on l'estime à environ 80% du temps de mise en place de la solution décisionnelle [8]. Le processus ETL à 5 étapes [40].

- Sélection des données sources.
- Extraction des données.
- Nettoyage et Transformation.
- Intégration.
- Chargement.

2.4.3 Les magasins (Datamart)

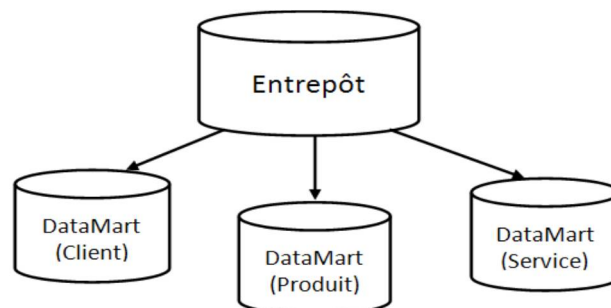


Figure I.2 : Magasins de données (datamarts) [37].

- Un Magasin de données (Datamart) est un sous-ensemble de l'entrepôt.

- Il Correspond à une classe de décideurs intéressés par le même thème.
- Son Volume réduit permet un accès plus rapide aux données.
- Généralement le magasin est modélisé sous forme multidimensionnelle.
- Les outils ETL peuvent être utilisés à Ce niveau [32].

2.4.4 Les applications clients

Applications d'acquisition de données, également connu sous le nom d'extraction, de transformation et de chargement (ETL) ou outils de back-end, qui permettent les données d'être extraites, transformées et chargées dans l'entrepôt de données [27].

L'OLAP est l'une des applications importantes pour les entrepôts de données. Des requêtes OLAP typiques scrutent de nombreuses relations, réalisent de nombreuses jointures et agrégations. Au contraire, les systèmes de gestion de bases de données, qui sont utilisés pour un travail opérationnel quotidien, appelés systèmes de traitement transactionnels en ligne (OLTP), sont optimisés pour assurer un petit nombre de transactions et les requêtes utilisent des clés primaires et des indexes spécialisés [45].

• OLTP vs OLAP

Critère	Online Transaction Processing (OLTP)	Online Analytical Processing (OLAP)
But	Contrôler et exécuter les tâches quotidiennes et fondamentales de l'entreprise	Assister dans la planification, la résolution de problème et la prise de décision
Types de données	Données opérationnelles (transactions)	Données historiques consolidées
Sources de données	BD transactionnelles	Entrepôts de données ou magasins de données
Ce que montrent les données	Portrait instantané des processus d'affaires de l'entreprise	Vue multidimensionnelle de plusieurs activités d'affaires de l'entreprise
Insertions et mises-à-jour	Courtes requêtes d'insertion et de mise-à-jour lancées par les usagers finaux	Longs traitements en lot servant à rafraichir les données
Requêtes	Simple requêtes retournant quelques enregistrements (lignes) de la BD	Requêtes complexes impliquant souvent plusieurs tables et faisant l'agrégation de valeurs
Temps de réponses	Instantané	Quelques secondes à 1 m max

Table I.2 : Comparaison entre OLTP et OLAP [40].

3 Les modèles des entrepôts de données

3.1 La modélisation conceptuelle

3.1.1 Concept de fait, de dimension et de hiérarchie

Le modèle multidimensionnel est une alternative mieux adéquate aux besoins de l'analyse des données d'un entrepôt. La modélisation multidimensionnelle part du principe que l'objectif majeur est la vision multidimensionnelle des données. Le constructeur fondamental de ces modèles est le cube de données, qu'offre une abstraction très proche de la façon dont l'analyste voit et interroge les données. Il organise les données en une ou plusieurs dimensions, Qui déterminent une mesure d'intérêt ou bien le fait. 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 [11].

- **Faits**

Le fait modélise le sujet de l'analyse. Un fait est formé de mesures correspondant aux informations de l'activité analysée [48]. Le sujet analysé est représenté par le concept de fait. Les mesures (attributs) d'un fait sont numériques ; on peut les additionner, les dénombrer ou bien calculer le minimum, le maximum ou la moyenne [34].

- **Dimension**

Les dimensions servent à enregistrer les valeurs pour lesquelles sont analysées les mesures de l'activité. Une dimension est généralement formée de paramètres (attributs) textuels (pour restreindre la portée des requêtes) et discrets (les valeurs possibles sont bien déterminées et constantes) [34]. 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 [11].

3.1.2. Modèles en étoile, en flocon et en constellation

3.1.2.1 Modèle en étoile

A partir du fait et des dimensions, il est possible d'établir une structure de données simple qui correspond au besoin de la modélisation multidimensionnelle. Cette structure est constituée du fait central et des dimensions. Ce modèle représente visuellement une étoile, on parle de *modèle en étoile* (star schema).

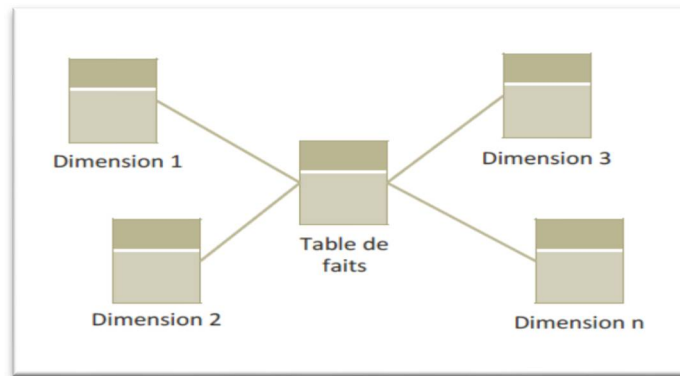


Figure I.3 : Schémas en étoile [32].

- Une table « faits » par indicateur.
- Une table dénormalisée pour chaque dimension.
- associations en étoile entre les tables « faits » et « dimensions » [29].

- **Avantage**

- Facilite de navigation.
- Performances: Nombre de jointures limite ; gestion des données creuses. Gestion des agrégats.
- Fiabilité des résultats.

- **Inconvénients**

- Toutes Les dimensions ne concernent pas les mesures Redondances Dans les dimensions.
- Alimentation complexe [32].

3.1.2.2 Modèle en flocons

Le modèle doit être simple à comprendre, on peut augmenter sa lisibilité en regroupant certaines dimensions [32]. Le schéma en flocon de neige est un raffinement du schéma en étoile où certaines tables de dimensions sont normalisées (donc décomposées) [35].

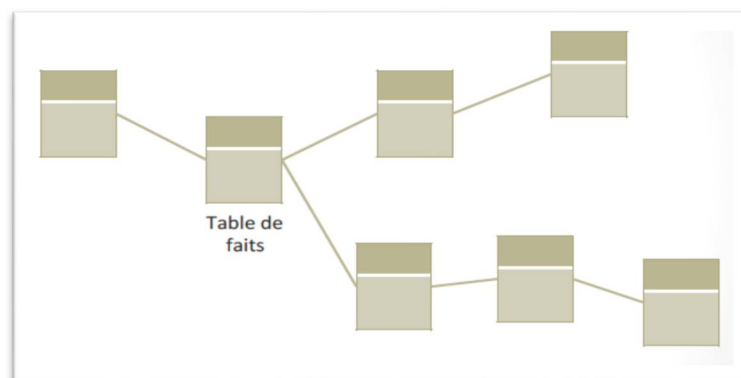


Figure I.4 : Schémas en flocons [32].

- **Avantages**

- Réduction du volume.
- Première des analyses par pallier (drill down) sur la dimension hiérarchisée.

- **Inconvénients**

- Navigation difficile.
- Nombreuses jointures [32].

3.1.2.3 Modèle en constellation

La constellation de faits permet de représenter plusieurs tables de faits partageant quelques tables de dimension [35]. Ces différentes relations de faits composent une famille qui partage les dimensions mais où chaque relation de faits a ses propres dimensions [10]. Un modèle en constellation comprend donc plusieurs faits et des dimensions communes ou non [23].

3.2. Modélisation logique

La conception logique d'un entrepôt de données vise à organiser et classer les informations des sources de données par sujet fonctionnel. Elle est préliminaire à la modélisation dimensionnelle où chaque sujet correspond à une table gérée au sein de l'entrepôt. Ce dernier permet d'isoler les données stratégiques et de conserver les métadonnées. L'interrogation des entrepôts de données se fait par des requêtes OLAP. Ces requêtes correspondent à une structuration des données selon plusieurs axes d'analyse pouvant représenter des notions variées telles que le temps de la localisation géographique ou le code identifiant des produits. Les modèles de conception des systèmes transactionnels ne sont pas adaptés à ce type de requêtes complexes qui utilisent beaucoup de jointures, demandent beaucoup de temps de calcul et sont de nature ad hoc. Pour ce type d'environnement, une nouvelle approche de modélisation a été suggérée: *les modèles multidimensionnels* [43].

3.2.1 les modèles multidimensionnels

Le modèle multidimensionnel permet d'organiser les données selon des axes représentant des éléments essentiels de l'activité d'une entreprise. Trois niveaux de représentation des données sont définis dans le processus décisionnel : l'entrepôt qui regroupe des données transversales à l'ensemble des métiers de l'entreprise, le magasin de données qui est une représentation verticale des données portant sur un métier particulier et enfin le cube de données (ou hypercube). Le cube correspond à une vue métier où l'analyste choisit les mesures à observer selon certaines dimensions. Un cube est une collection de données agrégées et consolidées pour résumer l'information et expliquer la pertinence d'une

observation. Le cube de données est exploré à l'aide de nombreuses opérations qui permettent sa manipulation [8].

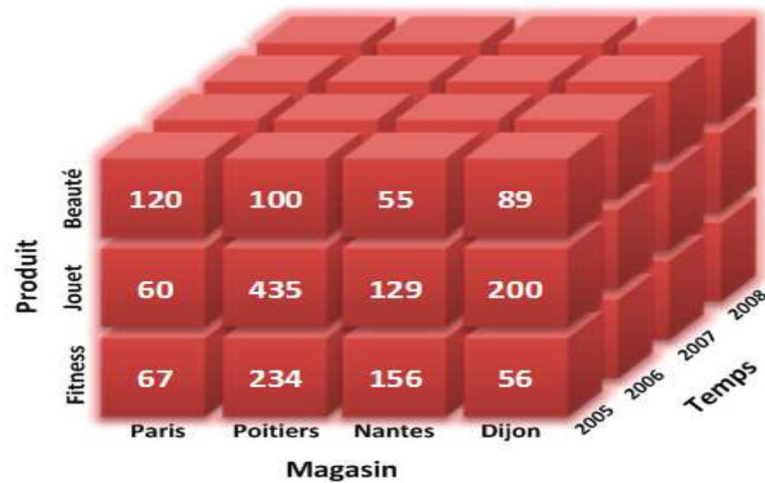


Figure I.5 : exemple d'un cube de données [15].

- **Table multidimensionnelle (TM)**

Une table multidimensionnelle « multidimensional table » est une structure de visualisation d'une tranche de données à deux dimensions. Plus précisément, elle présente des instances de mesures d'un fait, en fonction des instances des paramètres des dimensions représentées en lignes et colonnes.

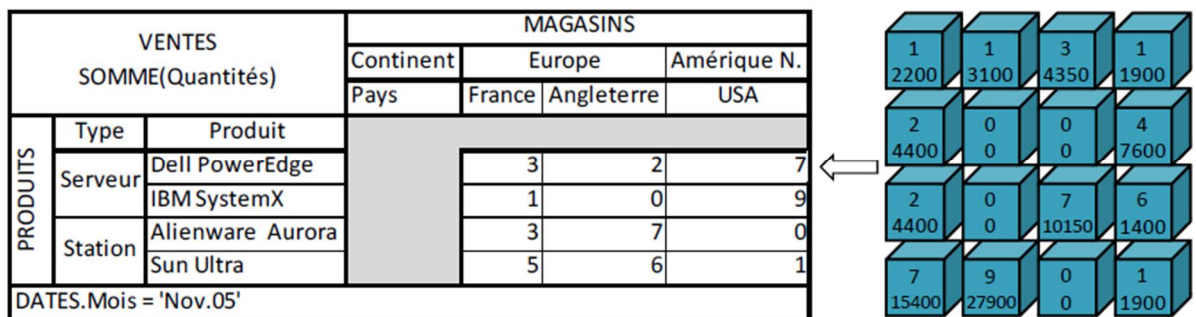


Figure I.6 : Table multidimensionnelle (Visualisation d'une 'tranche' du cube)[3].

Au niveau logique plusieurs possibilités sont envisageables pour la modélisation multidimensionnelle. Il est possible d'utiliser :

3.2.1.1 Modèle R-OLAP (Relational - On Line Analytical Processing)

Les systèmes de type ROLAP ("Relational On-Line Analytical Processing") utilisent un SGBD relationnel pour stocker les données de l'entrepôt. Le moteur OLAP est un élément supplémentaire qui fournit une vision multidimensionnelle de l'entrepôt, des calculs de données dérivées et des agrégations à différents niveaux. Il est aussi responsable de la génération des requêtes SQL mieux adaptées au schéma relationnel, qui bénéficient des structures d'optimisation existantes pour exécuter efficacement ces requêtes. Ces systèmes

peuvent stocker de grands volumes de données, mais ils peuvent présenter un temps de réponse élevé. Leurs principaux avantages sont la facilité d'intégration dans les SGBDs relationnels existants et une bonne efficacité pour stocker les données multidimensionnelles [21].

3.2.1.2 Modèle M-OLAP (Multidimensional-On Line Analytical Processing)

Une alternative à ces deux approches consiste à utiliser un système multidimensionnel "pur" qui gère des structures multidimensionnelles natives ; on parle de l'approche **MOLAP**. Les structures multidimensionnelles natives utilisées sont des tableaux à n dimensions. Dans la littérature, les termes de cube, hypercube et table multidimensionnelle sont utilisés de manière interchangeable. Nous utiliserons le terme d'**hypercube** pour désigner des structures à deux, trois ou à plus de trois dimensions. Cette approche permet de stocker les données de manière multidimensionnelle. L'intérêt est que les temps d'accès sont optimisés, mais cette approche nécessite de redéfinir des opérations pour manipuler ces structures multidimensionnelles [34].

3.2.1.3 Model H-OLAP (Hybrid - On Line Analytical Processing)

Ce modèle réunit les avantages des deux modèles M-OLAP et R-OLAP en diminuant leurs inconvénients. Il est utilisé surtout dans les outils commerciaux (Oracle Application Server, Microsoft Analyses Services). Il consiste à stocker les données détaillées dans des tables relationnelles (comme le modèle R-OLAP), tandis qu'il stocke les données agrégées sous une forme multidimensionnelle (comme le modèle MOLAP) [33].

3.3 La conception physique des entrepôts de données

La conception physique d'une base ou entrepôt de données consiste à établir une configuration physique sur le support de stockage. Cela comprend la spécification détaillée des éléments de données, les types de données et la sélection des techniques d'optimisation. Cette dernière est au cœur de la conception physique. Dans la première génération de moteurs d'exécution de requêtes dédiés aux bases de données traditionnelles, la conception physique n'avait pas autant d'importance. Aujourd'hui face à la complexité des requêtes à traiter et le volume important des données, la conception physique a reçu une importance phénoménale [15].

4 Choix des techniques d'optimisation

Si nous explorons la littérature et les éditeurs de gestion de bases de données commerciaux, nous trouvons une large panoplie de techniques d'optimisation qui peuvent être redondantes ou non. Les techniques redondantes nécessitent un espace de stockage et un coût

de maintenance (les vues matérialisées, les index avancées, la fragmentation verticale, etc.) par contre les techniques non redondantes ne nécessitant ni espace de stockage ni coût de maintenance (la fragmentation horizontale, le traitement parallèle, etc.). Pour optimiser les requêtes définies sur l'entrepôt, l'AED peut choisir une ou plusieurs techniques parmi ces deux catégories. Ce choix est souvent difficile dû au fait que certaines techniques sont bénéfiques pour certaines requêtes et non pas pour d'autres [8].

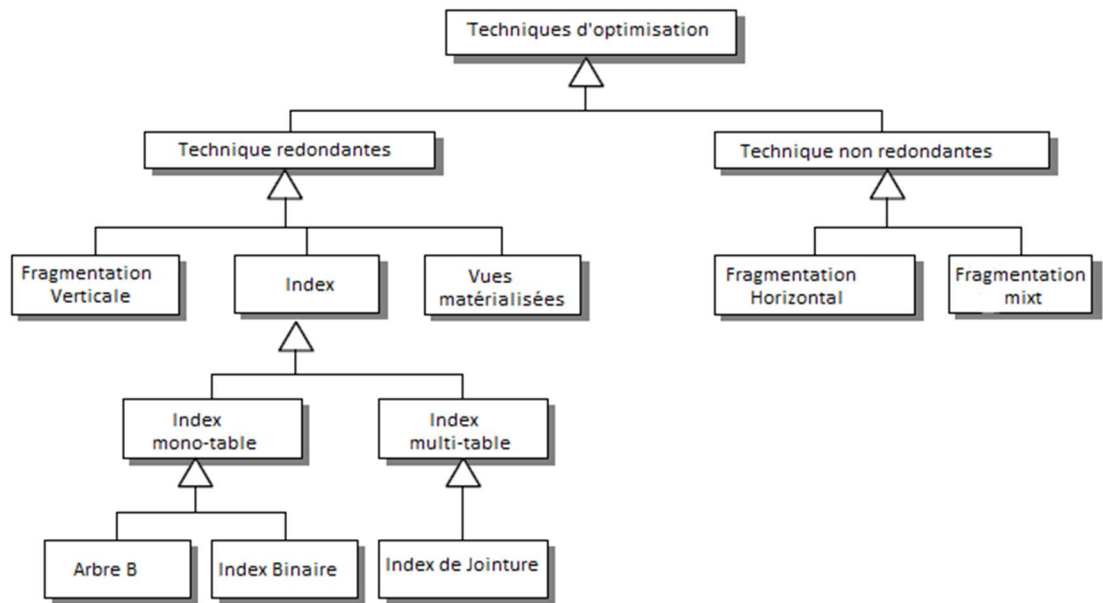


Figure I.7 : Classification des techniques d'optimisation

4.1 Les techniques redondantes

4.1.1 Les indexe

Pour améliorer les temps d'accès, l'administrateur emploie en général des indexes pour rechercher rapidement les informations nécessaires à une requête sans parcourir toutes les données. La sélection d'index est difficile car leur nombre est exponentiel en nombre total d'attributs dans la base. Donc l'index joue un rôle important dans la performance des bases de données [20].

Il y'a deux types d'indexes, les indexes mono-table (Exemple : index Bitmap, index B-Arbre), et les indexes multi-tables (Exemple : Index de jointure).

4.1.1.1 Index Bitmap

Un index bitmap comporte autant de colonnes qu'il y a de valeurs possibles pour l'index, et autant de lignes que de lignes dans la table. Pour chaque ligne l de la table et chaque valeur c de l'index, on stocke un bit positionné à 0 ou à 1 qui indique si la ligne l a la valeur c pour la colonne indexée.

C'est un type d'index qui permet de trouver rapidement les lignes répondant à un critère particulier. Il est aussi utile pour combiner facilement plusieurs critères. Par exemple, si on a des index bitmap sur la ville et sur le sexe, on peut combiner les bitmaps avec un $\wedge$ (et) logique pour obtenir directement la liste des étudiantes habitant Dijon [10].

	célibataire	marié	veuf
ligne1	1	0	0
ligne2	0	1	0
ligne3	0	0	1
ligne4	0	1	0

Figure I.8: Index Bitmap.

4.1.1.2 Index B-Arbre

L'index B-arbre stocke les pointeurs d'index et les valeurs à d'autres nœuds d'index en utilisant une structure d'arbre récursive. Les données sont facilement relevées par les traces des pointeurs. Le plus haut niveau de l'index est appelé racine pendant que le niveau le plus bas on l'appelle nœud de feuille ou « leaf node ». Tous les autres niveaux entre eux sont appelés des branches ou nœuds internes. Toutes les racines et les branches contiennent des entrées qui pointent vers le niveau suivant de l'index.

Nœuds feuilles se composent de la clé d'index et des pointeurs pointant vers l'emplacement physique des enregistrements. Nous présentons plus de détails sur la structure d'index arbre-B [30]. La structure B-Arbre est utilisée par le serveur de base de données pour configurer l'index (Figure1).

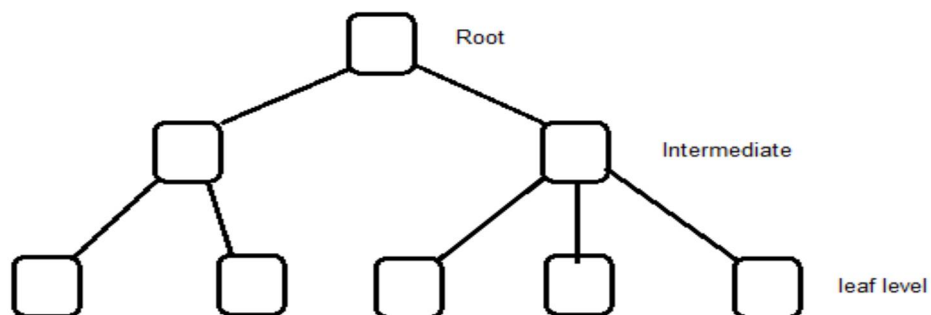


Figure I.9 : Index B-Arbre [30].

- o **Root ou racine:** plus haut niveau de l'index pointe vers les niveaux suivants des nodes branches.
- o **Nœuds intermédiaires ou branches :** contiennent des pointeurs vers le niveau suivant des branches ou vers les nœuds de feuilles.
- o **Nœud de feuilles ou leaf nodes :** le plus bas niveau de l'index pointe vers d'autres nœuds de feuilles [20].

4.1.1.3 Les index de jointure binaires

L'IJB sert à précalculer les jointures entre une ou plusieurs tables de dimension et la table des faits dans les entrepôts de données modélisés par un schéma en étoile. Au contraire des index binaires standards où les attributs indexés appartiennent à la même table, l'IJB est défini sur un ou plusieurs attributs appartenant à plusieurs tables. Plus précisément, il est défini sur la table des faits en utilisant des attributs appartenant à une ou plusieurs tables de dimension.

Pour montrer comment un IJB est construit, supposons un attribut A ayant n valeurs distinctes $V_1, V_2, \dots, V_n$ appartenant à une table de dimension D , supposons que la table des faits F est composée de m instances. La construction de l'index de jointure binaire défini sur l'attribut A se fait de la manière suivante:

1. Créer n vecteurs composés chacun de m entrées.
2. Le i ème bit du vecteur correspondant à une valeur V_k est mis à **1** si le n -uplet de rang i de la table des faits est joint avec un n -uplet de la table de dimension D tel que la valeur de A de ce n -uplet est égale à V_k . Il est mis à 0 dans le cas contraire [14].

4.1.2 La Fragmentation verticale

La fragmentation est une technique de conception logique utilisée dans les modèles relationnels et objet. Elle consiste à partitionner le schéma d'une base de données en plusieurs sous-schémas dans le but de réduire le temps d'exécution des requêtes [20]. L'unification respecte la règle de reconstruction d'une table fragmentée à partir de ses fragments verticaux et vise à réduire la redondance des vues. Les auteurs supposent que leur approche peut être bénéfique pour la distribution de l'entrepôt sur une architecture parallèle et proposent de combiner leur algorithme de fragmentation avec un algorithme d'allocation des fragments sur les nœuds distants [44].

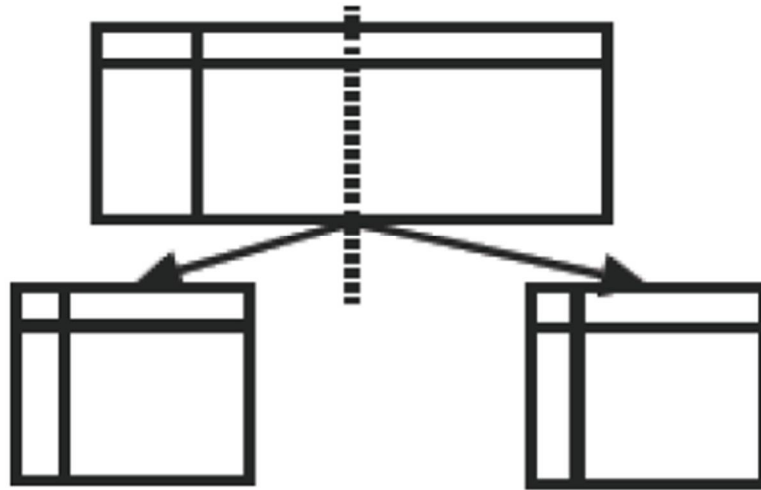


Figure I.10 : Fragmentation verticale.

4.1.3 Les vues matérialisés

Une vue est une requête nommée. 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. Il existe trois possibilités pour sélectionner un ensemble de vues [10].

1. *matérialiser toutes les vues* : Dans le cube de données cette approche consiste à matérialiser la totalité du cube, tandis que dans le cas du ROLAP, elle consiste à matérialiser tous les nœuds intermédiaires des arbres algébriques représentant les requêtes. Cette approche donne le meilleur temps de réponse pour toutes les requêtes. Mais stocker et maintenir toutes les cellules/nœuds intermédiaires est impraticable pour un entrepôt important.

2. *ne matérialiser aucune vue* : Dans ce cas nous sommes obligés d'accéder aux données des relations de base. Cette solution ne fournit aucun avantage pour les performances des requêtes.

3. *matérialiser seulement une partie du cube/des nœuds* : Dans un cube, il existe une certaine dépendance entre les cellules, c'est à dire que la valeur de certaines cellules peut être

calculée à partir des valeurs d'autres cellules. Dans le cas d'un système ROLAP, on trouve également cette dépendance dans les arbres algébriques. Il est alors souhaitable de matérialiser les parties partagées (cellules ou nœuds) par plusieurs requêtes. Cette approche a pour but de sélectionner les cellules ou les nœuds partagés. Cette solution semble la plus intéressante par rapport aux deux approches précédentes [27].

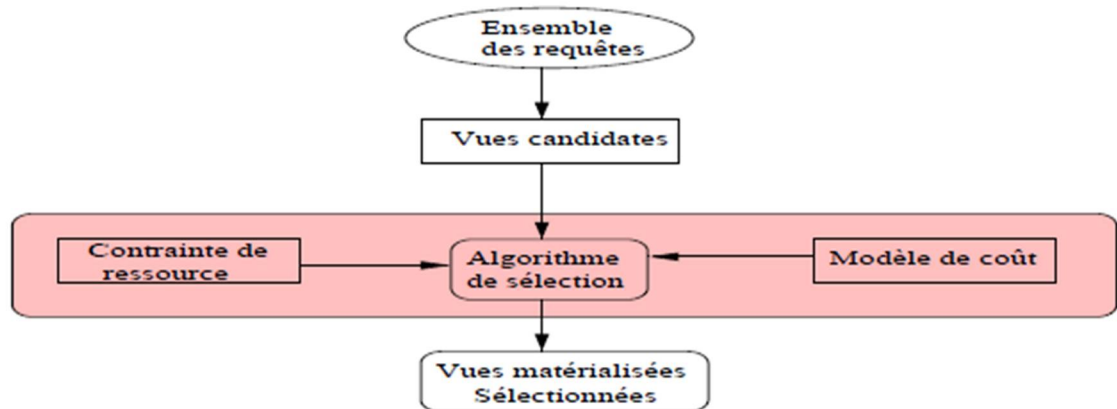


Figure I.11 : Le processus de sélection des vues matérialisées [8].

4.2 Les techniques non redondantes

4.2.1 La fragmentation horizontale

La fragmentation horizontale constitue le cœur de la conception d'un entrepôt de données parallèle. Dans le contexte des entrepôts de données relationnels, elle consiste à décomposer réellement ou virtuellement les tables de dimension en utilisant la fragmentation primaire (en exploitant les prédicats de sélection définis dans les requêtes) ensuite utiliser leurs schémas de fragmentation pour décomposer la table des faits. Cette méthodologie de fragmentation peut générer un nombre important de fragments de la table des faits [33].

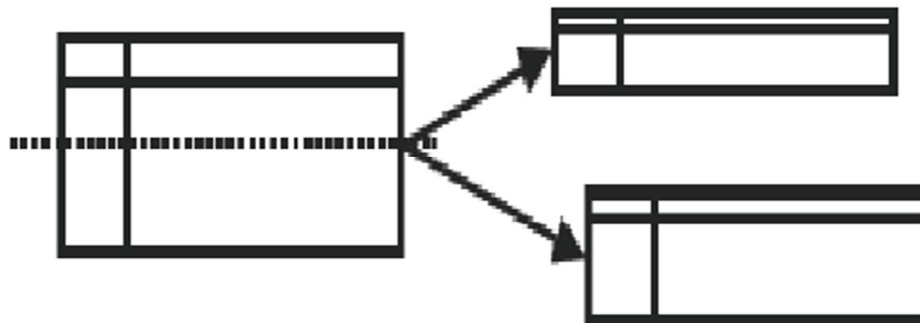


Figure I.12: Fragmentation Horizontale.

4.2.2 La fragmentation mixte

La fragmentation mixte résulte de la combinaison des deux sortes de fragmentation citées ci-dessus; pour atteindre les avantages de chaque type de fragmentation. Elle consiste à partitionner une relation en sous ensemble d'ensemble, cette dernière étant définie par la fragmentation verticale et les sous-ensembles par la fragmentation horizontale [34].

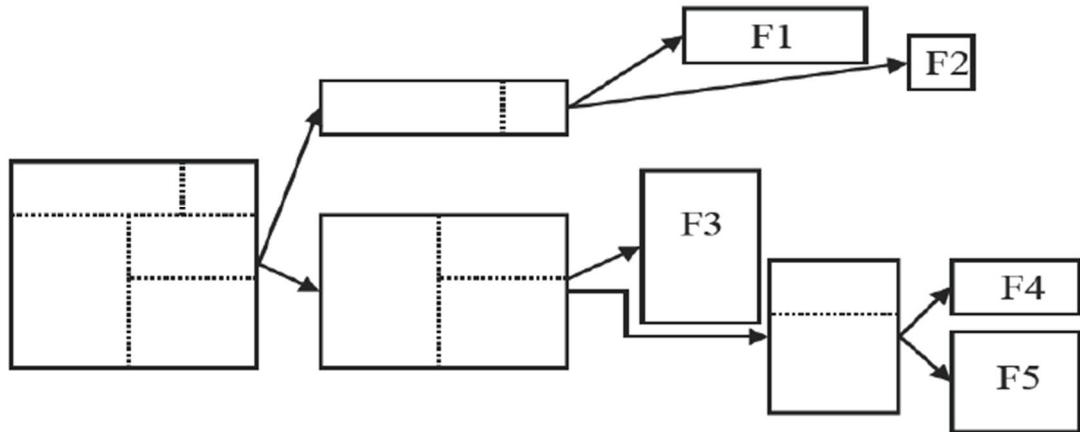


Figure I.13 : fragmentation mixte.

5 Conclusion

Nous avons présenté dans ce chapitre un état de l'art portant sur l'architecture, et les différents modèles de conception des entrepôts de données. Enfin, nous décrivons les principales techniques d'optimisation utilisées qui ont été suggérées dans la littérature pour améliorer les performances des entrepôts de données.

CHAPITRE II

La fragmentation horizontale dans les entrepôts de
données

1 Introduction

La fragmentation horizontale FH vise à restructurer les données afin de faciliter leur exploitation et interrogation par les requêtes décisionnelles. Elle permet de répartir les instances d'une table, d'un index ou d'une vue matérialisée en un ensemble de partitions disjointes appelés fragments horizontaux [6]. Le grand avantage tiré de cette structure revient au non duplication des données par rapport aux autres structures, une caractéristique qui participe directement à une réduction de coût de mises à jour. La fragmentation horizontale peut être combinée avec les autres structures, et elle est de deux types primaire et dérivée [23]. Dans ce chapitre en va présenter généralement la fragmentation horizontale FH, et déterminer le problème de sélection d'un schéma de FH.

2 La fragmentation horizontale FH

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 [25]. La fragmentation horizontale se base sur la répartition des tuples d'une table en plusieurs sous-ensembles suivant des critères de fragmentation. Ces critères sont des prédicats de sélections définis sur un ensemble d'attributs de fragmentation. Ainsi, la définition d'un schéma de fragmentation revient à répartir ces prédicats en sous-ensembles où chaque sous ensemble permet de définir un fragment [21].

La fragmentation horizontale dans les entrepôts de données relationnels est plus difficile que celle dans les bases de données relationnelles et objets. Pour fragmenter une table de faits, l'administrateur de l'entrepôt de données (DWA) doit exécuter les étapes suivantes :

1. Sélectionner la (les) table(s) de dimension participant dans le processus de fragmentation de la table de faits.
2. Décomposer ces tables de dimension.
3. Utiliser leurs schémas pour fragmenter la table de faits [18].

Il existe deux types de fragmentation horizontale.

2.1 Fragmentation primaire

La fragmentation horizontale primaire permet de répartir les tuples d'une table suivant la conjonction de prédicats de sélection définis sur les attributs de la même table. Elle favorise les opérations de sélection portées sur les attributs de fragmentation, car pour exécuter une requête qui contient un prédicat de sélection sur un attribut de fragmentation, l'optimiseur ne charge que les fragments concernés par le prédicat de sélection. Elle est réalisée grâce à l'application de l'opération de restriction sur les tuples de la table à fragmenter [36].

2.2. Fragmentation dérivée

La fragmentation horizontale dérivée permet de fragmenter une relation suivant la fragmentation horizontale primaire d'une seconde relation, à condition de l'existence de relation mère-fille entre les deux tables. Elle est réalisée par l'exécution de semi-jointure entre les fragments propriétaires et la table membre à fragmenter [31]. Et pour bien comprendre, (Bellatreche) en [20] donner une explication pour la Fragmentation dérivée, une relation **S** peut contenir une clé étrangère vers une relation **R**. La fragmentation horizontale dérivée est définie sur la relation membre S en fonction des fragments horizontaux de la relation propriétaire R. Plus formellement, soit R une relation propriétaire horizontalement fragmentée en **m** fragments horizontaux $\{R_1, R_2, \dots, R_{m1}\}$.

Soit **S** une relation membre. Les fragments horizontaux dérivés $\{S_1, S_2, \dots, S_m\}$ de S sont définis par l'opération de semi-jointure suivante : $S_i = S \bowtie R_i$.

Trois informations sont nécessaires pour effectuer une fragmentation horizontale dérivée :

1. Le nom de la relation membre et de la relation propriétaire.
2. Le nombre de fragments horizontaux de la relation membre.
3. La qualification de la jointure entre ces deux relations.

La fragmentation horizontale primaire favorise le traitement des requêtes de restriction portant sur les attributs utilisés dans le processus de la fragmentation. La fragmentation horizontale dérivée est utile pour le traitement des requêtes de jointure¹.

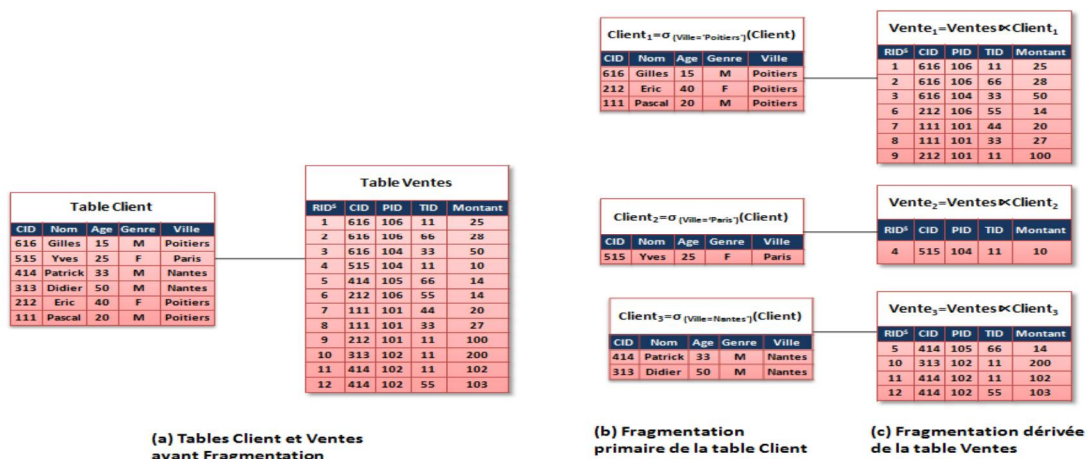


Figure II.1 : Exemple d'une fragmentation primaire et dérivée [15].

¹ Une requête qui consulte plusieurs lignes de la même ou de différentes tables en même temps est appelée requête de jointure.

2.3 Les avantages de la fragmentation horizontale

La FH permet d'éclater les objets de la base de données (tables, vues et index) en plusieurs partitions de plus petites tailles. L'administrateur de la ED et l'utilisateur final considèrent différemment la ED fragmentée. Du point de vue de l'administrateur, une ED fragmentée est composée de plusieurs partitions qui peuvent être gérées et manipulées collectivement ou individuellement. Cela pourra donner une meilleure flexibilité pour gérer ces partitions.

Du point de vue utilisateur, une base de données fragmentée est identique à celle non fragmentée. Aucune modification n'est nécessaire dans les requêtes interrogeant la ED. La fragmentation horizontale peut être bénéfique pour une multitude d'applications en améliorant trois aspects : la facilité de gestion, la performance de requêtes et la maintenance [15].

3 Méthodologie de fragmentation horizontale dans les entrepôts de données

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 jeu 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 [23] :

- *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 conséquent 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 [23].

- *Partitionner seulement la table des faits en utilisant la fragmentation primaire* : Notons que la table des faits est composée des clés étrangères des tables de dimensions et des mesures numériques, comme le montant des ventes, le nombre d'articles soldés, etc. Généralement, dans une requête décisionnelle, nous trouvons rarement des prédicats de sélection définis sur des attributs d'une table des faits [5].

- *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.

4 Le Procédure de FH

Pour illustrer cette procédure de fragmentation, nous considérons un schéma en étoile avec trois tables de dimension : *Client*, *Temps* et *Produit* et une table de fait *Ventes*. Supposons que la table de dimension *Client* est fragmentée en deux fragments *Client1* et *Client2* définis par les clauses suivantes : $Client1 = \frac{3}{4} Sexe='M' (Client)$ et $Client2 = \frac{1}{4} Sexe='F' (Client)$. Ce type de fragmentation est supporté par les SGBDs commerciaux. Cette fragmentation est matérialisée par la clause SQL suivante :

- CREATE TABLE Client (NC number(4), Cnom varchar2(14), Sexe varchar2(1))
- PARTITION BY LIST (Sexe) (PARTITION Client1 values ('M'), PARTITION Client2 values ('F')) ;

La table de faits *Ventes* peut être fragmentée en deux fragments *Ventes1* et *Ventes2* définis par : $Ventes1 = Ventes \bowtie Client1$ et $Ventes2 = Ventes \bowtie Client2$, où $\bowtie$ représente l'opération de semi-jointure.

Ce processus de fragmentation génère deux sous schémas en étoile *S1* et *S2* tels que : *S1* : (*Ventes1*, *Client1*, *produit*, *temps*) (activités de ventes réalisées par les clients masculins seulement) et *S2* : (*Ventes2*, *Client2*, *produit*, *temps*) (activités de ventes réalisées par les clientes féminine seulement).

Le nombre de fragments horizontaux de la table de faits (noté *N*) généré par cette procédure de fragmentation est donné par : $N = \prod_{i=1}^g m_i$, où m_i et g représentent respectivement le nombre de fragments de la table de dimension *Di*, et le nombre de tables de dimension participant dans le processus de fragmentation. Ce nombre peut être très grand [22]. En conséquence, au lieu de gérer un seul schéma en étoile, doit gérer *N* sous schémas en étoile. Il lui sera très difficile de maintenir tous ces sous schémas.

La fragmentation peut améliorer l'exécution de la jointure entre la table des faits et les tables de dimension. Si par exemple, l'administrateur veut exécuter une requête OLAP avec des opérations d'agrégation ne concernant que les clientes, la jointure entre table des faits *Ventes* et la table de dimension *Client* se traduit par une jointure entre *Vente2* et *Client2*. Parallèlement, la fragmentation pourrait être concurrente dans l'utilisation aux index de jointure binaires (bit map join indexes). L'index de jointure est construit en traduisant les restrictions à la valeur de la colonne de la table de dimension [16]. Nous pouvons construire un index de jointure binaire afin d'optimiser la requête précédente [16] :

```
CREATE BITMAP INDEX IJB
ON Client (Sexe)
```

FROM Client, Ventes

WHERE Client.NC = Ventes.NC.

4.1 Module de gestion des requêtes

Ce module permet d'aider l'DWA à définir la charge de requêtes les plus fréquentes (Q) sur laquelle l'optimisation est basée. Le module permet une édition manuelle des requêtes ou une importation à partir de fichiers externes. Il permet aussi de gérer la charge en donnant la possibilité d'ajouter, de supprimer ou de modifier des requêtes. Le module intègre un parseur qui permet d'identifier les erreurs syntaxiques ainsi que les tables et attributs utilisés par chaque requête [18].

4.2 Règles de correction de la fragmentation horizontale

Pour qu'une fragmentation horizontale d'une table T en p fragments $T_1, T_2, \dots, T_p$ soit valide, elle doit vérifier trois règles de correction : complétude, disjonction et reconstruction.

1. *Complétude* : toute instance de T doit être présente dans au moins un fragment ($i \in T \Rightarrow \exists T_m \in \{T_1, T_2, \dots, T_p\} \wedge i \in T_m$). La complétude garantit qu'aucune instance ne sera perdue après la fragmentation de la table.
2. *Disjonction* : toute instance appartenant à un fragment T_k ne doit pas appartenir à un autre fragment T_j ($i \in T_j \Rightarrow \nexists T_k \neq T_j \wedge i \in T_k$).
3. *Reconstruction* : une fois la fragmentation effectuée, il doit être toujours possible de reconstruire la table d'origine à partir de ses fragments. Pour cela, il faut disposer d'un opérateur OP qui permet de reconstruire la table T ($T = OP(T_1, T_2, \dots, T_p)$). La règle de correction garantit que tout processus de fragmentation est toujours réversible. OP représente l'opération d'union. En respectant ces trois règles de correction, l'opération de fragmentation horizontale réalise une partition des n-uplets de la table T, au sens mathématique du terme. [15]

5 Le problème de sélection d'un schéma de fragmentation

Le problème de sélection d'un schéma de fragmentation est un problème d'optimisation mono-objectif qui vise à optimiser un objectif, qui est le coût d'exécution de la charge de requêtes, avec une contrainte W sur le nombre de fragments faits pouvant être générés. [25] Le problème de fragmentation horizontale consiste à (1) déterminer un ensemble de tables de dimension à fragmenter, et (2) utiliser leurs schémas de fragmentation pour fragmenter la table de faits F en un ensemble de N fragments horizontaux (appelés les fragments de faits) tels que le coût total d'exécution des requêtes sur le schéma en étoile fragmenté soit réduit [13].

5.1 Travaux de recherche sur le problème de fragmentation

Les travaux qui prennent en charge la fragmentation dans les entrepôts de données relationnels se permettent des travaux effectués sur les bases de données relationnels. La fragmentation peut être horizontale, verticale, ou mixte.

5.1.1 Travaux de DATTA et AL. (1999)

Les auteurs ont exploité la fragmentation verticale pour créer un index appelé « cuio » basé sur la matérialisation des fragments au lieu des attributs indexés afin d'accélérer l'accès aux données et minimiser l'espace de stockage [25].

5.1.2 Travaux de Wu et Buchmaan (1997)

Les auteurs recommandent de combiner la fragmentation horizontale et la fragmentation verticale (Wu & Buchmann, 1997). Selon eux, la table de faits peut être partitionnée horizontalement à partir de fragments définis sur les dimensions. Elle peut aussi être partitionnée verticalement selon les clés étrangères faisant référence aux dimensions. Cependant, aucun algorithme de fragmentation n'est proposé [25].

5.1.3 Travaux de NOAMAN & BARKER (1999)

Les travaux de ces auteurs sont développés dans un environnement distribué d'un entrepôt de données, ils partent d'un schéma conceptuel global de l'entrepôt, puis ils procèdent par la répartition pour construire les schémas conceptuels locaux. Cette répartition se fait en deux étapes, à savoir la fragmentation et l'allocation, suivies éventuellement d'une optimisation locale [27].

5.1.4 Travaux de Bellatreche (2000)

Pour étudier la complexité du problème de sélection d'un schéma de fragmentation horizontale nous considérons un problème de décision simplifié qui prend en considération un seul domaine d'un attribut d'une table de dimension pour partitionner la table des faits. Nous appelons ce problème, *Problème de Fragmentation Horizontale à un Seul Domaine (PFHSD)*. Le problème d'optimisation correspondant consiste à partitionner la table des faits telle que le nombre de partitions de cette table soit borné par une constante et le nombre d'opérations d'entrées/sorties (E/S) nécessaires soit minimisé [25].

6 Algorithmes de sélection d'un schéma de FH

Plusieurs travaux ont étudié la fragmentation horizontale et montré son intérêt dans le contexte des bases de données classiques, le contexte des bases de données objets et le contexte d'entrepôts de données. Les algorithmes proposés peuvent être classés en trois principales catégories : algorithmes basés sur les prédicats, algorithmes basés sur l'affinité et algorithmes basés sur un modèle de coûts. Tous ces algorithmes commencent par une table à

fragmenter et un ensemble de prédicats ou de requêtes les plus fréquentes ainsi que leurs fréquences d'accès et donnent en sortie un schéma de fragmentation horizontale de la table. [15]

6.1 Algorithmes basées sur les prédicats

Le principe de ces Algorithmes revient à identifier les sous-ensembles des prédicats contenus dans les tables de dimensions à condition que ces prédicats garantissent la minimalité (disjonction des fragments obtenus deux à deux) et la complétude (possibilité de la reconstruction d'une relation à l'aide de l'union de l'ensemble des fragments la constituant). Ces sous-ensembles sont exploités ensuite pour la fragmentation dérivée [28].

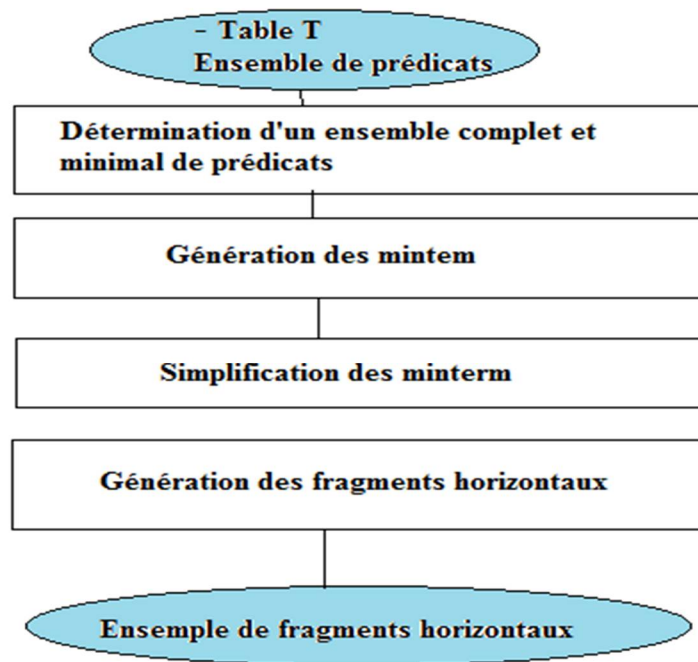


Figure II.2 : Approche basée sur les prédicats [15].

6.2 Algorithmes basés sur l'affinité

Cette méthode est une adaptation de la fragmentation verticale. Elle est basée sur le principe d'affinité qui désigne la somme des fréquences d'accès des requêtes introduisant simultanément ces prédicats. Le résultat est l'aboutissement à des regroupements de prédicats de sélection selon les fréquences des requêtes qui les utilisent [50]. Cette approche est composée de cinq étapes [15]:

1. L'énumération des prédicats simples.
2. La construction de la matrice d'usage des prédicats.
3. La génération de la matrice d'affinité des prédicats.
4. Le regroupement des prédicats.

5. La génération des fragments horizontaux.

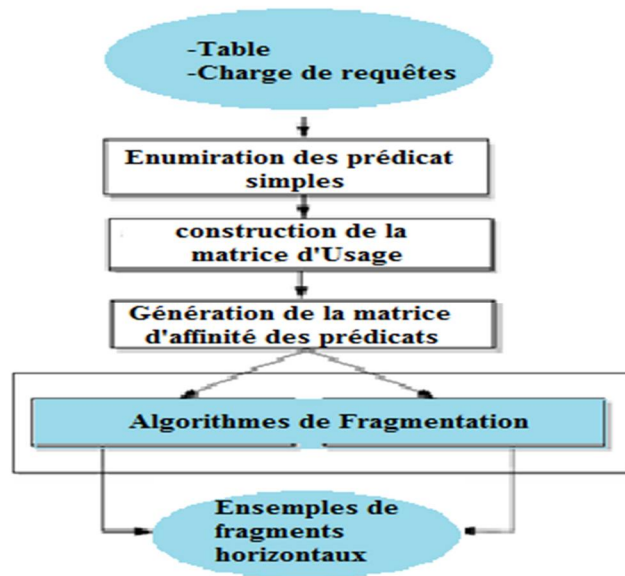


Figure II.3 : Approche basée sur les affinités [15].

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 de n attributs de fragmentation [36].

2. Construire la matrice d'usage MUS : La matrice d'usage des prédicats (MUP) est une matrice $n \times m$ où n représente le nombre de requêtes et m le nombre de prédicats. Un élément reçoit 1 si la requête Q_i utilise le prédicat P_j , sinon il reçoit 0 [15].

Soit l'attribut Ville de la table Clients avec le découpage en 5 sous-domaines $SD_1=\{\text{Alger}\}$, $SD_2=\{\text{Oran}\}$, $SD_3=\{\text{Blida}\}$, $SD_4=\{\text{Kala}\}$ et $SD_5=\{\text{Tizi}\}$. 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 [36].

	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

Table II.1 : Matrice d'usage des sous-domaines de l'attribut Ville [36].

3. *Construire la matrice d'affinité des sous-domaines* : c'est une matrice carrée ($n_k \times n_k$) où les lignes et les colonnes représentent les sous-domaines. Chaque élément j de cette matrice représente la somme des fréquences des requêtes à céder simultanément aux deux sous-domaines SD_k, i et SD_k, j . La table 1.2 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

Table II.2 : Matrice d'affinité des sous-domaines de l'attribut Ville [36].

4. *Regroupement des prédicats* : cette étape permet de décomposer l'ensemble des prédicats en sous-groupes de prédicats. [15]

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 aucune garantie sur la qualité de la solution obtenue. [36]

6.3 Algorithmes à base de modèle de coût

Un modèle de coût est employé afin d'estimer le taux de réduction de la complexité des requêtes décisionnelles par le schéma de fragmentation horizontale de l'entrepôt de données [38]. Les chercheurs dans [21] et [24] proposent une stratégie de sélection qui emploie un modèle de coût afin d'évaluer chaque schéma de fragmentation généré. Elle est réalisée en trois étapes :

- *Phase de génération des schémas de fragmentation* : à partir des la charge de requêtes, les prédicats de sélection sont extraits. Ils permettent d'identifier les différents fragments à travers les minterms. Un générateur produit tous les schémas de fragmentation possibles.
- *Phase d'évaluation* : se base sur un modèle de coût, elle permet d'évaluer le coût d'exécution des requêtes sur chaque schéma de fragmentation généré.

- *Phase de sélection du schéma optimal* : permet de déterminer le schéma de fragmentation le plus bénéfique pour l'exécution de la charge de requête.

Le premier algorithme proposé est exhaustif et consiste à construire tous les schémas de fragmentation possibles par fragmentation horizontale. Il énumère ensuite ces schémas et calcule pour chacun d'entre eux le coût d'exécution des requêtes de la charge. Il sélectionne finalement le schéma qui correspond au coût minimum [37].

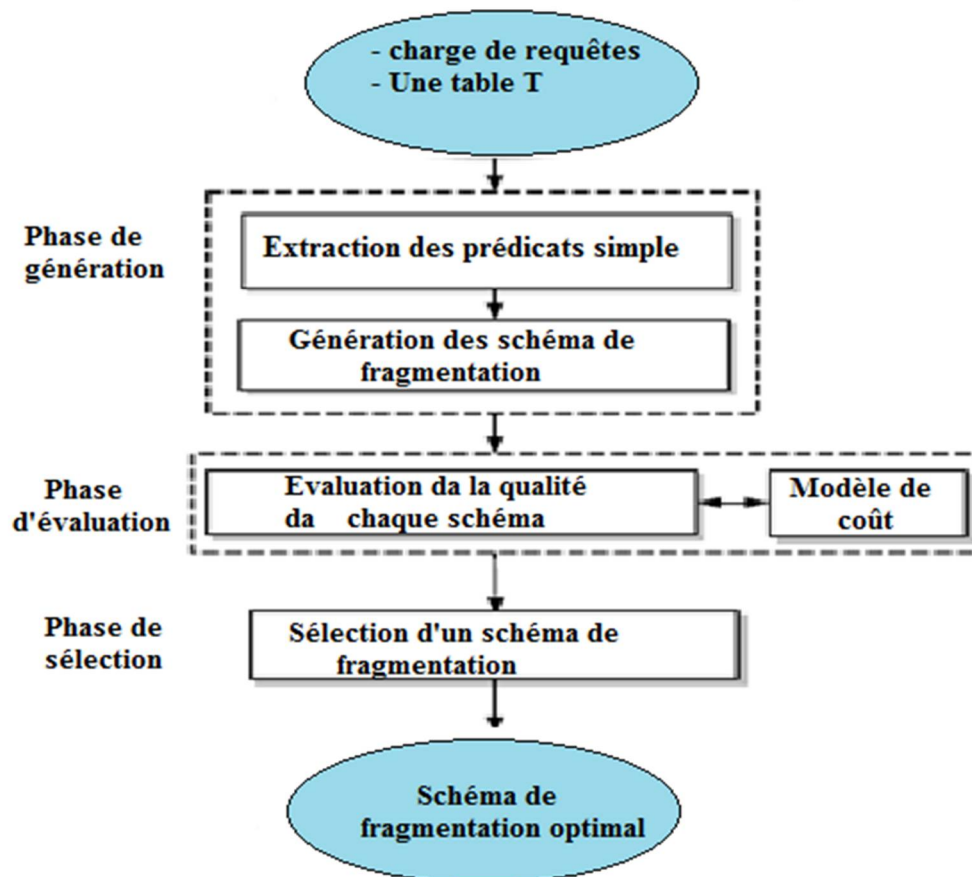


Figure II.4 : Approche basée sur un modèle de coût [15]

7 Conclusion

Pour assurer une bonne conception physique des entrepôts de données, il faut utiliser des structures redondantes comme les vues matérialisées et l'indexation et des structures non redondantes comme la fragmentation des données. Dans ce chapitre, nous sommes intéressés à la fragmentation horizontale des entrepôts de données relationnels, nous avons présenté une méthodologie complète pour la fragmentation horizontale dans les entrepôts de données modélisés par des schémas en étoile, on appelle ce formalisme Problème d'optimisation.

CHAPITRE III

Métaheuristiques et fragmentation d'entrepôt de
données

1 Introduction

Les métaheuristiques¹ sont généralement des méthodes bio-inspirées², elles forment aujourd'hui la majorité des algorithmes utilisés pour des problèmes d'optimisation difficile, dans tous les domaines de recherches, Elles sont généralement utilisées comme des méthodes génériques pouvant traiter une large gamme de problèmes différents. Dans ce chapitre nous devons donner premièrement le problème que nous avons traité, et après on va donner une vue générale sur les deux *Métaheuristiques* (les algorithmes génétiques **AG** et les recuits simulés **RS**) choisies pour la résolution du problème de sélection de schéma de fragmentation horizontale d'entrepôt de données pour lequel l'exécution d'une charge de requêtes prend un temps d'exécution très considérable sans sa fragmentation.

2 Description du problème

Un entrepôt de données stocke de grosses quantités de données consolidées et historisées. Il est particulièrement conçu pour répondre aux requêtes décisionnelles complexes [16], et la complexité des requêtes de jointure en étoile engendrent un temps d'exécution important. Pour réduire ce temps, l'administrateur de l'entrepôt (AED) doit sélectionner un ensemble de techniques d'optimisation [15]. La problématique traitée dans ce chapitre s'articule sur la *Sélection de Schéma de Fragmentation horizontale* : La plupart des approches proposées partitionne des tables en fonction des fréquences d'accès des prédicats de sélection et ne contrôle pas le nombre de fragments généré.

- Comment sélectionner un schéma de fragmentation horizontale optimisant une charge de requêtes et générant un nombre de fragments fixé par l'AED?
- **Pourquoi avons-nous besoin de tels algorithmes ?**

Certains problèmes sont trop complexes pour que l'on puisse proposer un algorithme, ou lorsqu'il n'y a aucune méthode exacte pour les résoudre, d'une modélisation délicate et/ou confuse où la solution au problème est inconnue. Ou alors un algorithme tolérant aux variations de données, susceptible de s'adapter à une nouvelle situation, pour qu'il évolue. Par exemple, comment effectuer le déplacement d'un robot ? Comment faire réagir ce robot suite à une bousculade, une perte d'équilibre ? Comment adapter le comportement du robot face au danger ? [52]

¹ Une **métaheuristique** est un algorithme d'optimisation visant à résoudre des problèmes d'optimisation difficile. Terminologie : On parle de **méta**, du grec *μετά* « au-delà » (comprendre ici « à un plus haut niveau »), **heuristique**, du grec *εὕρισκειν* / heuriskein, qui signifie « trouver ».

² Inspirées des principes de la biologie. Une informatique bio-inspirée est un système ou une architecture dont l'organisation est issue de la connaissance que l'on a du fonctionnement des systèmes vivants.

3 Les Algorithmes génétiques

3.1 Définition des algorithmes génétique

Les algorithmes génétiques sont des algorithmes d'exploration fondés sur les mécanismes de la sélection naturelle et de la génétique. Ils utilisent à la fois les principes de la survie des structures les mieux adaptées, et les échanges d'information pseudo-aléatoires, pour former un algorithme d'exploration qui possède de certaines des caractéristiques de l'exploration humaine [9].

3.2 Principe de base

Le fonctionnement des algorithmes génétiques est schématisé dans la figure 3.1 dont les étapes sont :

1. A l'initialisation de l'algorithme, une population de taille fixée est générée aléatoirement. Elle est en générale répartie uniformément sur l'espace de recherche. Chaque individu ensuite évalué et on lui associe son adaptation.
2. Pour la génération k , on reproduit une partie de la population en fonction de l'adaptation de chaque individu : plus la *fitness* (relative, i.e. par rapport à celle des autres individus) d'un individu est élevée, plus il sera reproduit dans la nouvelle population.
3. On choisit des couples d'individus parents au hasard dans cette nouvelle population, puis on leur applique avec la probabilité P_c l'opérateur de croisement, les enfants remplacent alors les parents dans la population de la génération $k + 1$. L'opérateur de croisement est traditionnellement l'heuristique d'un algorithme génétique.
4. On applique ensuite l'opérateur de mutation à chaque individu avec la probabilité P_m qui est en général choisit plus faible que P_c . La mutation effectue un déplacement «local» sur les individus. Les mutants remplacent alors les parents dans la nouvelle génération $k + 1$.
5. Les individus qui n'ont subi ni de croisement sont recopiés tels quels dans la nouvelle population.
6. On réitère ces opérations à partir de la 2ème étape jusqu'à ce qu'un critère d'arrêt soit satisfait. Différents critères d'arrêt de l'algorithme génétique peuvent être choisis : nombre de générations fixé (temps constant), convergence de la population, population n'évoluant plus suffisamment... [11].

L'algorithme démarre d'une population initiale, et effectue des opérations génétiques sur les individus. Les meilleurs individus sont conservés et croisés pour obtenir une nouvelle génération. Ainsi, à partir d'un nombre fini de générations, les individus acquièrent de

meilleures performances [4], et voici un algorithme plus détaillé d'un algorithme génétique [2] :

Algorithme III.1 : Algorithme génétique générique

Début

1. Initialiser les paramètres nécessaires ;
2. Initialiser une population de N individus ;
3. Évaluer les N individus ;
4. **Tant que** la condition d'arrêt n'est pas satisfaite **faire**
 1. Utiliser l'opérateur de sélection pour sélectionner K individus ;
 2. Appliquer l'opérateur de croisement sur les K individus avec la probabilité P_c ;
 3. Appliquer l'opérateur de mutation sur les individus avec la probabilité P_m ;
 4. Utiliser l'opérateur d'évaluation pour évaluer les enfants obtenus ;
 5. Utiliser l'opérateur de sélection pour remplacer quelques individus parents par des individus enfants ;

Fin Tant que

5. Retourner la ou les meilleures solutions ;

Fin

En répétant ce cycle plusieurs fois, on obtient une population composée de solutions meilleures. On utilise généralement les algorithmes génétiques pour trouver une solution, la meilleure solution après un certain nombre de générations [18].

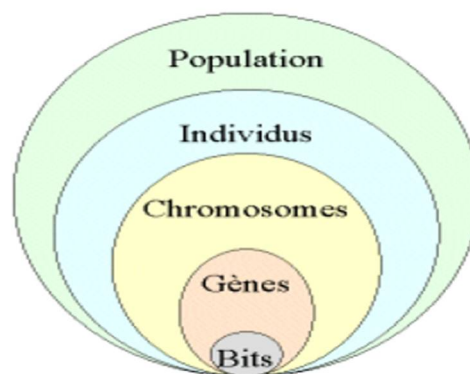


Figure III.1 : Les cinq niveaux d'organisation d'un algorithme génétique.

- **Un gène**: est un ensemble de symboles représentant la valeur d'une variable. Dans la plupart des cas, un gène est représenté par un seul symbole (un bit, un entier, un réel ou un caractère).

- **Un chromosome:** est un ensemble de gènes, présentés dans un ordre donné de manière qui prend en considération les contraintes du problème à traiter. Par exemple, dans le problème du voyageur de commerce, la taille du chromosome est égale au nombre de villes à parcourir. Son contenu représente l'ordre de parcours de différentes villes. En outre, on doit veiller à ce qu'une ville (représentée par un nombre ou un caractère par exemple) ne doit pas figurer dans le chromosome plus qu'une seule fois.
- **Un individu:** est composé d'un ou de plusieurs chromosomes. Il représente une solution possible au problème traité.
- **Une population:** est représentée par un ensemble d'individus (i.e. l'ensemble des solutions du problème).
- **Une génération:** est une succession d'itérations composées d'un ensemble d'opérations permettant le passage d'une population à une autre [2].

3.2.1 Le codage des individus

Le premier pas dans l'implantation des algorithmes génétique, Chaque paramètre d'une solution est assimilé à un gène, toutes les valeurs qu'il peut prendre sont les allèles de ce gène, on doit trouver une manière de coder chaque allèle différent de façon unique.

Un chromosome est une suite de gène, on peut par exemple choisir de regrouper les paramètres similaires dans un même chromosome (chromosome à un seul brin) et chaque gène sera repérable par sa position : son locus sur le chromosome en question.

Chaque individu est représenté par un ensemble de chromosomes, et une population est un ensemble d'individus [42].

Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'état une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité. La qualité du codage des données conditionne le succès des algorithmes génétiques. Les codages binaires ont été très utilisés à l'origine. Les codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs pour l'optimisation de problèmes à variables réelles [42].

3.2.2 L'évaluation d'un individu

La *fitness* d'un individu est calculée par la fonction d'évaluation ; elle mesure son adaptation à un environnement donné et le but de l'algorithme génétique est de maximiser cette fonction. Pour résoudre un problème d'optimisation, on prend naturellement la fonction de coût comme fonction d'évaluation [11].

3.2.3 Les opérateurs génétiques

Après avoir défini un codage, une population initiale de n_{ind} individus est générée aléatoirement. Chaque individu est caractérisé par un chromosome comptant L gènes et représente un point de l'espace de recherche. L'initialisation de la population peut donc être réalisée simplement en effectuant $n_{ind} \times L$ tirages au sort binaires, de manière à générer n_{ind} chromosomes. Le tirage aléatoire doit avoir une loi de probabilité uniforme, de façon à obtenir une occupation homogène de l'espace de recherche. À chaque génération, on évalue la performance de chaque individu de la population, ce qui nécessite n_{ind} évaluations de la fonction de coût. Pour un problème de minimisation, la valeur opposée ou inverse de la fonction de coût peut être choisie. Ensuite, des opérateurs font évoluer les caractéristiques génétiques de la population, de manière à explorer l'espace de recherche et favoriser les caractéristiques des meilleurs individus pour les générations futures [17].

3.2.2.1 L'opérateur de sélection

L'opérateur de sélection permet de sélectionner les meilleurs individus dans la population pour participer dans la génération de la prochaine population. Les différents types de sélection qui sont couramment utilisés :

- *La sélection par roulette* : Cette méthode associe chaque individu à sa qualité (valeur de la fonction d'évaluation). La roulette est partagée entre tous les individus, la surface allouée à chaque individu est en fonction de la qualité de l'individu. L'implémentation de cette méthode se fait de la manière suivante :
 - ✓ Calculer la valeur de la fonction d'évaluation de chaque individu et la somme totale de ces valeurs.
 - ✓ Chaque individu est associé à sa probabilité de sélection (segment de longueur égal à sa valeur de fitness divisée par la somme des fitness déjà calculée).
 - ✓ Tirer un nombre aléatoire et choisir l'individu dont le segment englobe ce nombre [15].
- *La sélection aléatoire* : cette sélection se fait aléatoirement, uniformément et sans intervention de la valeur d'adaptation. Chaque individu a donc une probabilité uniforme $\frac{1}{P_{size}}$ d'être sélectionné, où P_{size} est le nombre total d'individus dans la population.

- *Sélection par rang* : consiste à ranger les individus de la population dans un ordre croissant ou décroissant, selon l'objectif (fonction fitness).
- *Sélection par tournoi* : On tire deux individus aléatoirement dans la population et on reproduit le meilleur des deux dans la nouvelle population. On répète la procédure jusqu'à ce que la nouvelle population soit complète [1].

3.2.2.2 L'opérateur de croisement ou crossover

Le phénomène de croisement est une propriété naturelle de l'ADN³. Il a pour objectif d'enrichir la diversité de la population en manipulant les composantes des individus, c'est adire les chromosomes. C'est par analogie avec la notion d'hybridation de Darwin qu'a été conçu cet opérateur. Classiquement, les croisements sont envisagés avec un couple d'individus parents et génèrent deux enfants. Cet opérateur favorise l'exploration de l'espace de recherche. Considérons deux gènes A et B pouvant être améliorés par mutation. Il est peu probable que les deux gènes améliorés A' et B' apparaissent par mutation chez un même individu. Par contre l'opérateur de croisement permettra de combiner rapidement A' et B' dans la descendance de deux parents portant chacun un des gènes mutants. Il est alors possible que la présence simultanée des deux gènes produise un individu encore plus adapté.

L'opérateur de croisement assure donc le brassage du matériel génétique et l'accumulation des mutations favorables. En termes plus concrets, cet opérateur permet de créer de nouvelles combinaisons des paramètres des composants [46].

Ci-dessous, un schéma illustrant un croisement en un point, un autre pour un croisement en deux points.

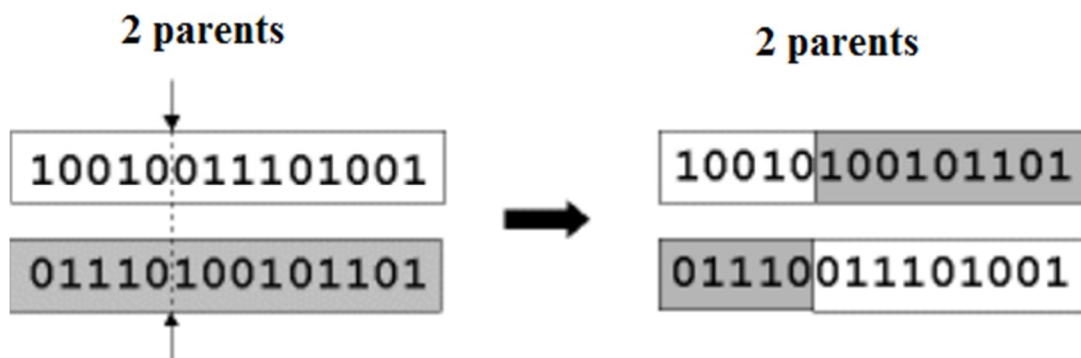


Figure III.2 : Exemple de croisement à un point [42].

³ Acide désoxyribonucléique : ADN est le support de l'hérédité car il constitue le génome des êtres vivants et se transmet en totalité ou en partie lors des processus de reproduction.

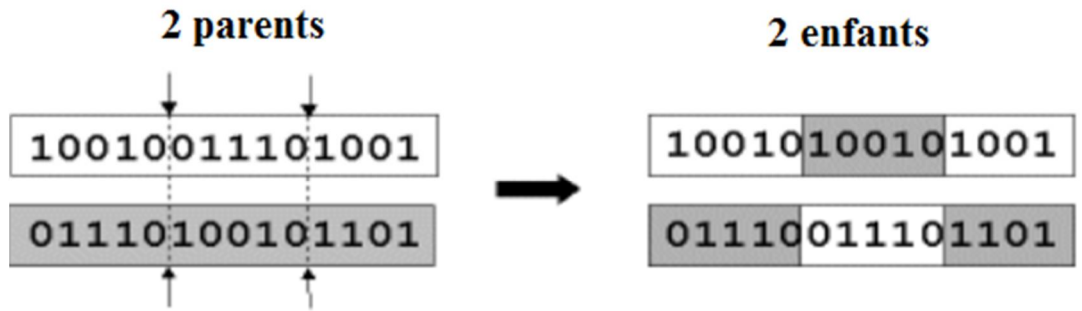


Figure III.3 : Exemple de croisement à deux point [42].

3.2.2.3 L'opérateur de mutation

Le rôle de la mutation est d'introduire de nouvelles caractéristiques génétiques, ou de les réintroduire, en modifiant quelques gènes des individus enfants. Pour chaque gène de chaque enfant, une permutation est réalisée avec une probabilité p_m . Cette valeur est généralement faible, de manière à ne pas transformer l'algorithme en une simple recherche aléatoire. La mutation permet d'explorer l'espace de recherche aléatoirement et de réintroduire certains gènes qui ont disparu au cours des générations [39].

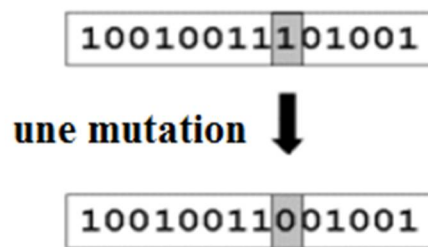


Figure III.4 : Exemple une mutation [42].

L'opérateur de mutation modifie donc de manière complètement aléatoire les caractéristiques d'une solution, ce qui permet d'introduire et de maintenir la diversité au sein de notre population de solutions. Cet opérateur joue le rôle d'un "élément perturbateur", il introduit du "bruit" au sein de la population [42].

4 Les recuits simulés (*Simulated Annealing*)

Le recuit simulé est une technique aléatoire pour trouver une solution proche de l'optimale des problèmes d'optimisation combinatoire. Il commence par une solution candidate aléatoire, ensuite il essaye à plusieurs reprises de trouver une solution meilleure en se déplaçant vers un voisin avec une fitness plus élevée, jusqu'à l'obtention de la meilleure solution. Pour éviter de se bloquer sur un optimum local, le RS, permet, occasionnellement, des déplacements vers des solutions avec une fitness

inférieure en utilisant un paramètre température pour contrôler l'acceptation des déplacements avec une certaine probabilité dépendant d'un certain nombre de paramètres. Ces deux algorithmes seront décrits en détails dans les sections suivantes [16].

Algorithme III.2 : Algorithme Recuit Simulé

1. Choisir une solution $s \in S$ ainsi qu'une température initiale T .
2. Tant qu'aucun critère d'arrêt n'est satisfait faire
3. Choisir aléatoirement $s' \in N(s)$;
4. Générer un nombre réel aléatoire r dans $[0,1]$;
5. Si $r < p(T, s, s')$ alors poser $s := s'$;
6. Mettre à jour T ;
7. Fin du tant que

La fonction $p(T, s, s')$ est généralement choisie comme étant égale à la distribution

Suivant $e^{\frac{f(s)-f(s')}{T}}$. Ainsi,

- Si $f(s) > f(s')$ alors $e^{\frac{f(s)-f(s')}{T}} > 1$, ce qui signifie que $r < p(T, s, s')$ et on accepte donc s'
- Si T a une très grande valeur alors $e^{\frac{f(s)-f(s')}{T}} \cong 1$, et on est donc presque sûr d'accepter s'
- Si T a une très petite valeur et si $f(s) < f(s')$ alors $e^{\frac{f(s)-f(s')}{T}} \cong 0$, et on va donc probablement refuser s'

5 Application de GA au Problème de sélection de schéma de FH

La fragmentation de la table des faits dans un environnement DW 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. Ce dernier a pour but de sélectionner les 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 [23] [22].

5.1 Processus génétique de la fragmentation horizontale

5.1.1 Codage adopté

Le codage de solution est une étape très important dans l'algorithme génétique, en fait, il doit permettre la représentation des différentes solutions possibles. Le codage que nous avons semblable à le codage réel mais avec des nombre entiers, L'encodage d'un schéma de

fragmentation sous forme d'un chromosome est représenté dans le tableau 5.2. Ce codage permet d'attribuer à chaque sous-domaine un numéro, les sous-domaines qui possèdent le même numéro sont fusionnés en un seul ensemble.

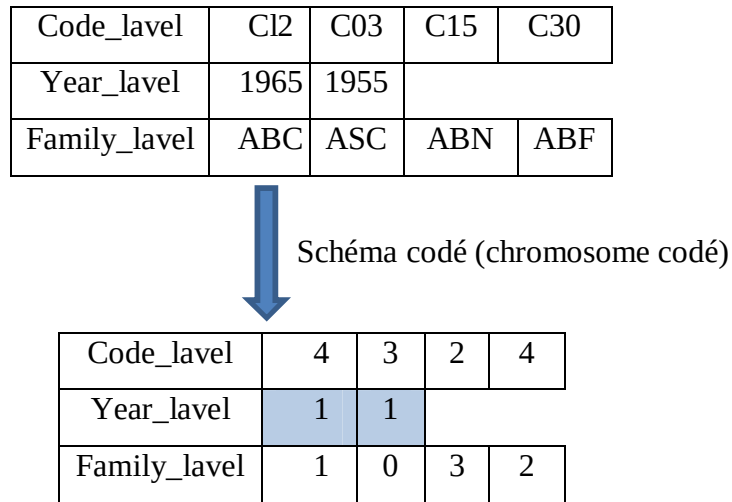


Figure III.5 : Exemple de codage.

5.1.2 Génération de la population initiale

Dans cette étape, une population initiale doit être générée, ou chaque chromosome représente une solution réalisable du problème (schéma de fragmentation). La procédure que nous avons utilisée pour générer la population initiale des individus, adaptée à notre problème est une génération aléatoire de P_{taille} individus.

5.1.3 Sélection des individus

Il permet l'échange des gènes entre parents (deux parent en général), pour créer 1 ou deux enfants en essayant de combiner les bonnes caractéristiques des parents. Le but de cet opérateur est la création de nouveaux individus en exploitant l'espace de recherche. Dans notre algorithme nous avons utilisé la sélection aléatoire. Cette sélection se fait aléatoirement, uniformément et sans intervention de la valeur de la fonction objectif.

5.1.4 Croisements des individus

L'opérateur de croisement est appliqué sur deux chromosomes parents pour obtenir deux chromosomes enfants. Nous avons utilisé le croisement multi-points. Le croisement est une généralisation du croisement à un point en ce sens qu'au lieu de choisir un seul point de coupure, on en sélectionne k . l'intérêt d'une telle approche apparaît rapidement, car il devient possible en un seul croisement d'échanger des *buildings*.

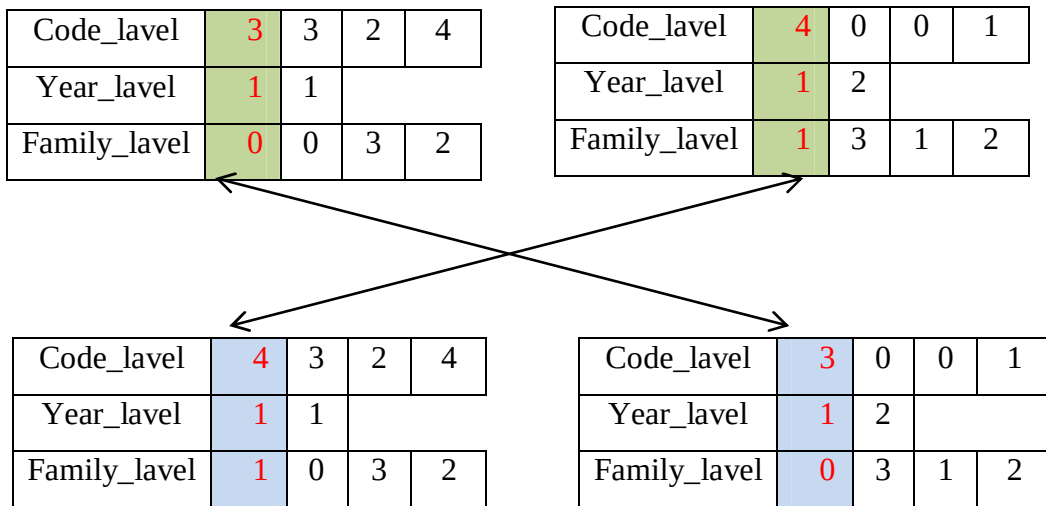


Figure III.6 : Exemple de croisement.

6.1.5 Mutation

Il faut choisir un juste milieu entre un trop faible et une trop forte mutation. 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.

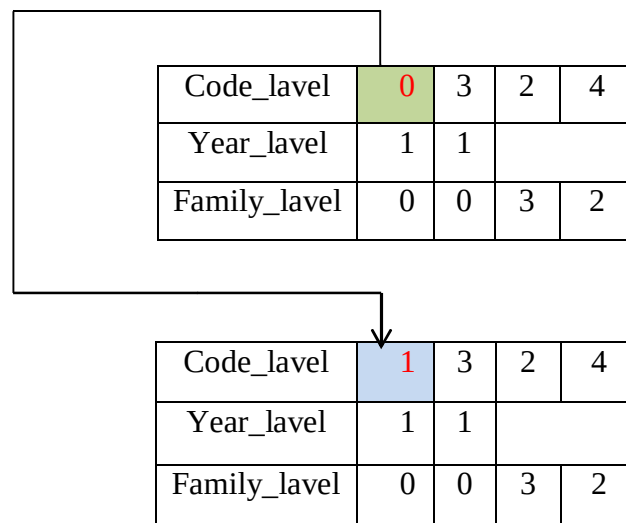


Figure III.7: Exemple de Mutation.

6.1.6 Fonction d'évaluation : *fitness*

La fonction d'évaluation consiste à minimiser le coût, elle génère tous les chromosomes de schéma de fragmentation. L'ensemble des tables de dimension $D = \{D_1, D_2, \dots, D_k\}$ ayant des attributs de sélection, où chaque prédicat de sélection P_i (défini sur une table de dimension D_i) possède un facteur de sélectivité noté par $(Sel_{D_i}^{P_i} \in [0, 1])$.

Finalement, ce coût de chaque chromosome peut être exprimé par l'équation suivante :

$$\text{Coût (CH)} = \sum_{i=1}^{N_i} \sum_{j=1}^{NP_j} \text{Sel}_{P_j}(\text{att}_i) \quad (1)$$

Avec, NP_j , N_i et att_i représentent :

- NP_j : Le nombre de prédicats de sélection utilisés pour définir les fragments de sous schéma en étoile.
- N_i : Le nombre des attributs participants dans le schéma de FH (chromosome).
- att_i : Les attributs.

5.1.7 Critère d'arrêt

Le critère d'arrêt est le nombre de génération égal à Nbr_{Max} . Après Nbr_{Max} génération, l'algorithme génétique s'arrête et donne le meilleur chromosome qui possède un Coût (CH) minimum.

$$\text{Fitness (schema): Max} \left(\frac{1}{\text{Coût(CH)}} \right) \quad (2)$$

Algorithme III.3 : Algorithme génétique pour la sélection d'un schéma de FH

1. Initialiser : P_{size} , P_{cross} , P_{mut} , Nbr_{Max} , $U = 0$.
2. Générer aléatoirement la population de la taille P_{size} .
3. Répéter
4. $I=0$
5. Tant que $U < \frac{P_{size}}{2}$ faire
6. Sélectionner aléatoirement deux parents de la population.
7. Faire le croisement des parents pour obtenir deux enfants par une probabilité P_{cross} .
8. Muter les deux enfants par une probabilité P_{mut} .
9. $U = U + 1$
10. Fin tant que
11. Évaluer tous les chromosomes ($2 P_{size}$) par la fonction d'évaluation
12. Ranger les parents et les enfants dans l'ordre croissant selon leur fonction d'évaluation fitness.
13. Supprimer les P_{size} chromosomes faibles et enregistrer les meilleurs chromosomes ($Best_{Pop}$) selon leur fonction d'évaluation fitness
14. Remplacer (P_{size} , $Best_{Pop}$)
15. $i = i + 1$
16. Jusqu'à $i = Nbr_{Max}$

6 L'application de recuit simulé RS sur la solution de GA

Le recuit simulé est appliqué sur la solution finale obtenue par notre AG (i.e., l'état initial du RS est le schéma de fragmentation généré par l'AG). La valeur de la fonction fitness de chaque solution est calculée en utilisant le modèle de coût développé dans la section 6.1.6.

Les différentes étapes de notre algorithme RS sont décrites dans Algorithme III.4 :

Algorithme III.4 : Algorithme Recuit Simulé

1. EtatInitial (représentant le schéma de fragmentation obtenu par l'AG) ;

TempératureInitiale ;

2. EtatMin := EtatInitial ; coût := **Evaluation(EtatInitial)** ; CoûtMin := Coût ;

3. Répéter

3.1. Répéter

NouveauEtat = Transformation(EtatMin) ;

NouveauCoût = F(Q, NouveauEtat) ;

Si (NouveauCoût < Coût) **alors**

EtatCourant = NouveauEtat ;

Coût = NouveauCoût ;

Sinon avec une probabilité $e^{\frac{\text{NouveauCoût} - \text{coût}}{\text{temp}}}$

EtatCourant := NouveauEtat ;

Coût := NouveauCoût ;

Fin si ;

Si (Coût < CoûtMin) **alors**

EtatMin = EtatCourant.

CoûtMin = Coût.

Fin si ;

Jusqu'à l'obtention d'un équilibre. // Nombre d'itérations

3.2 Réduire la température ;

Jusqu'à gel du système ; // Température = 0.

4. Retourner EtatMin

Fin ;

7 Conclusion

Dans ce chapitre nous avons déterminé le problème que nous abordons dans notre recherche, qui concerne à la sélection d'un schéma de fragmentation horizontal, et pour résoudre ce problème nous avons proposé une solution par les algorithmes génétiques et les requis simulés.

Nous avons décrit le fonctionnement et les différents opérateurs d'un algorithme génétique standard, et aussi nous avons vu les recuits simulé qui définit comme une technique aléatoire pour trouver une solution proche de l'optimale des problèmes d'optimisation combinatoire⁴, et après nous avons présenté l'application de GA au Problème de sélection de schéma de FH.

⁴ optimisation discrète, est une branche de l'optimisation en mathématiques appliquées et en informatique, également liée à la recherche opérationnelle

CHAPITRE IV

Réalisation et expérimentation

1 Introduction

Pour commencer l'implantation de la base de données pour s'interagir avec l'application réalisée il faut tout d'abord choisir un environnement matériel et logiciel adéquat pour le bon déroulement de travail. Dans ce chapitre nous allons présenter nos outils, nous exposons également choix concernant les technologies utilisées dans la réalisation et le développement de notre travail, telles que les langages de programmation, des bases de données et les serveurs etc. Ensuite nous allons présenter les pages et les fonctionnalités de notre travail.

2 Les outils de développement

2.1 Microsoft Visual Studio

Visual Studio uniformise le développement des applications en Visual Basic, Visual C# .NET, Visual C++ et Visual F#. ASP.NET, Cloud et JavaScript [47], utilisent tous le même environnement de développement intégré (IDE, *integrated development environment*), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du .NET Framework, qui fournit un accès à des technologies clés simplifiant le développement d'applications Web ASP et de services Web XML.

2.1.1. Microsoft Visual Studio 2012



Figure IV.1: Visual Studio 2012 [54].

Visual Studio 2012, introduit le développement d'applications sur l'environnement Windows RT. Le SDK Windows Phone 8.0 est également davantage mis en avant (s'il n'est pas inclus avec l'environnement, il est tout de même proposé au téléchargement). Il introduit la version 4.5 du .NET Framework et les versions compatibles de Windows sont Windows 7 et Windows 8 (et les déclinaisons serveur de ces éditions).

Le numéro de version interne de Visual Studio 2012 est 11.0 (le symbole `_MSC_VER` étant défini comme 1700) [55].

Enfin, nous notons que Visual Studio 2012 nous a également fourni des classes prédéfinies relatives à la gestion de base de données 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.

2.1.2 Langage de programmation C#

C# (à prononcer « C-sharp ») est un langage créé par Microsoft en 2001 et normalisé par l'ECMA (ECMA-334) et par l'ISO/CEI (ISO/CEI 23270). Il est très proche de Java et de C++, dont il reprend la syntaxe générale ainsi que les concepts orientés objet. Depuis sa création, et contrairement à d'autres langages de programmation, C# a beaucoup évolué à travers les différentes versions du .NET Framework, en particulier dans la version 3.5 où est introduit un langage de requête intégré appelé LINQ. Bien évidemment, il y a fort à parier que Microsoft ne s'arrêtera pas là et proposera certainement dans les versions ultérieures d'autres nouveautés !, C# est l'un des langages qui permet de manipuler la bibliothèque des classes du .NET Framework, plateforme de base permettant d'unifier la conception d'applications Windows ou Web [1].

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.

2.2. Oracle



Figure IV.2 : Oracle.

Oracle est un système de gestion de bases de données relationnelles (**SGBDR**) édité par Oracle Corporation. C'est actuellement le leader mondial des bases de données, devant des produits comme SQL Server de Microsoft. Oracle est un SGBD assurant des fonctionnalités telles que la définition et la manipulation des données, La cohérence des données, La confidentialité des données, L'intégrité des données, La sauvegarde et la restauration des données, La gestion des accès concurrents.

Oracle est fourni avec de nombreux outils d'administration et de développement parmi lesquels, Oracle Manager (**SQL*DBA**), Network Manager¹, Oracle Enterprise Manager, Import/Export (Échange de données entre deux bases Oracle), Oracle Designer, Oracle Développer, SQL*Plus : Interface permettant d'exécuter des requêtes SQL et PL/SQL sur la base de données [53].

2.2.1 Oracle Développer



Figure IV.3 : Oracle Développer.

Oracle Développer, est une suite de produits destinés au développement d'applications client-serveur. Il est composé de 4 applications : Oracle Forms (anciennement SQL*Forms), qui permet d'interroger la base de données de façon graphique sans connaissances du langage SQL [49].

- ✓ Alors afin de valider notre approche en utilisant le Benchmark APB1 installé sous Oracle 10g, nous avons programmé le problème dans un environnement de Programmation Orienté Objet qui est Visual Studio 2012 avec le langage de programmation C#, sous Windows 7 i3 et 2 G de RAM.

3 Implémentation

3.1 L'entrepôt de données (Entrepôt de banc d'essai Benchmark)

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'essais² Benchmark APB-1³ [11]. Le schéma en étoile que nous avons dégagé à partir de ce

¹ **Network-Manager** est l'outil de gestion des connexions réseau d'Ubuntu. Son utilité est la création et la configuration des accès à divers types de réseaux

² Le banc d'essai est souvent confondu avec le *benchmark* (« banc test ») qui ne fait que tester les performances d'un système.

banc d'essais est constitué d'une table de faits Actvars et de quatre tables de dimensions, *Prodlevel*, *Custlevel*, *Timelevel* et *Chanlevel* (figure 4.4).



Figure IV.4 : Schéma de l'entrepôt utilisé APB benchmark.

Les requêtes SQL de la création des tables sont représentées dans l'annexe A1.

Table	Nombre d'enregistrement	Taille d'enregistrement
Actvars	1 500 000	74
Prodlevel	9000	72
Custlevel	900	24
Chanlevel	9	24
Timelevel	24	36

Table IV.1 : Taille des tables

3.1.1 Types de requêtes prises en compte

Les requêtes prises en compte dans ces expériences 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]
```

- **SGA** : l'ensemble des attributs retournés dans la réponse à la requête (peuvent être des attributs de dimension et/ou de faits).
- **FC1, FC2,..., FCn** : {MIN, MAX, COUNT, SUM,AVG}.

³ L'objectif d'un benchmarking est d'identifier les processus les plus efficaces et professionnels pour aider l'organisation à atteindre ses objectifs, à déterminer un objectif "idéal" en termes de résultats, de qualité de service/produit et à améliorer les processus de l'organisation.

- **AA** : l'ensemble des attributs d'agrégation⁴
- **k** : le nombre des domaines ou tables de dimension utilisées par la requête.
- **PJOIN** : l'ensemble des prédicats de jointure entre la table des faits et les tables de dimension sous forme de conjonction.
- **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.

Les requêtes que nous avons prises en charges pour notre modèle de coût sont représentées dans l'annexe A1.

3.1.2 Vecteur de sélectivité

Chaque prédicat a une utilité ou un profit. Ce profit est égal au coût de la requête en nombre d'Entrées/Sorties de la requête introduisant ce prédicat. Comme nous avons vu dans le chapitre 3 (6.1.6) le coût est calculé en fonction de la sélectivité du prédicat.

$$\text{Coût (CH)} = \sum_{i=1}^{N_i} \sum_{j=1}^{NP_j} \text{Sel}_{pj}(\text{att}_i)$$

Nous avons prisent les vecteurs de sélectivité qui nous avons utilisé depuis le travail de Mr Mohamed BARR une étude antérieure [17] (voir l'annexe A2).

3.2 Implémentation de solution

3.2.1 Procédure de solution

Pour résoudre le problème de sélection d'un schéma de **FH**, Nous proposons une méthode hybride combinant un algorithme génétique **AG** et un algorithme de recuit simulé **RS** (Figure 4.5).

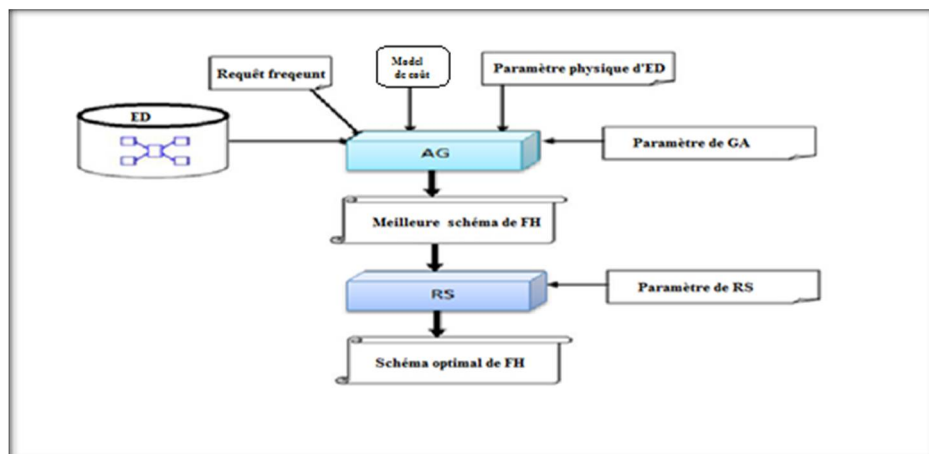


Figure IV.5 : Procédure de travail.

⁴ Ces attributs sont généralement des mesures définies dans la table des faits.

3.2.2 L'architecture de l'application

Nous avons utilisé la programmation orienté objet⁵ (POO) parce qu'il offre une lisibilité de manipulation du code source tel que maintenance, détection des erreurs et aider de corriger, débogage ...etc.

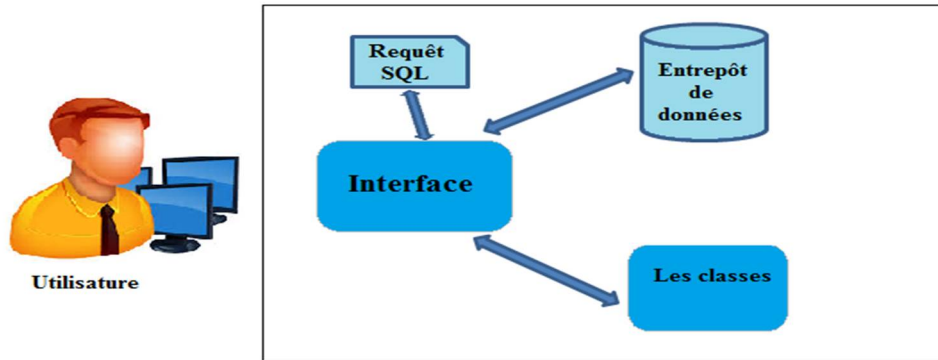


Figure IV.6 : Architecture de l'application.

- *Interface* : C'est une simple interface graphique permet aux utilisateurs une meilleur interaction avec l'application.
- *Les classes* : représente les classes principales utilisées dans la programmation ou chaque classe conserve les informations sur les types de données contenues par les objets nécessaires dans notre application, et aussi les comportements possibles de cet objet.

3.2.3 les cas d'utilisation de l'interface

Pour décrire la fonctionnalité de notre interface, on a opté le langage de modélisation unifié (UML) pour réalise le diagramme de cas d'utilisation UML (**figure 4.7**)

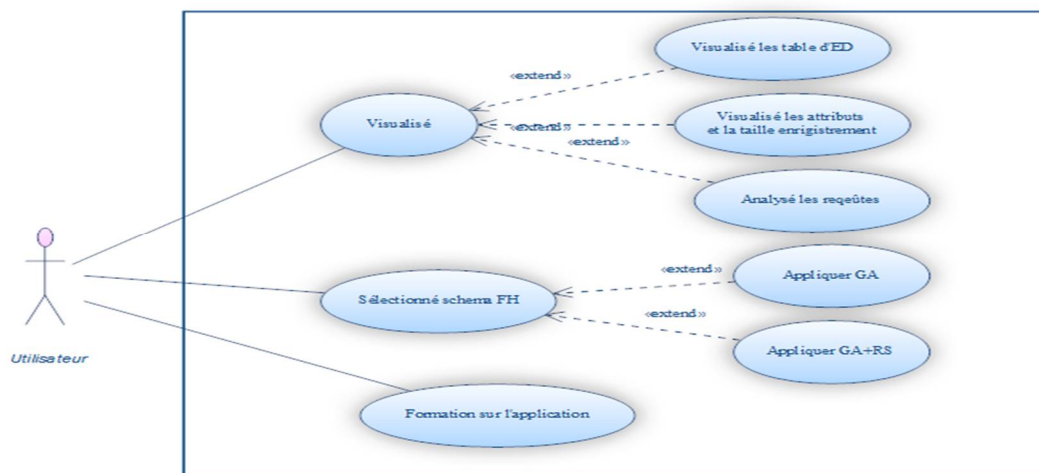


Figure IV.7 : Diagramme de cas d'utilisation.

⁵ Technique d'organisation du code d'un programme en le groupant en objets, les objets étant ici des éléments individuels comportant des informations (valeurs de données) et des fonctionnalités.

4 Présentation graphique

La page principale « Application sur l'ED » est un ensemble des pages (tabControl)

➤ *Visualisé l'ED*

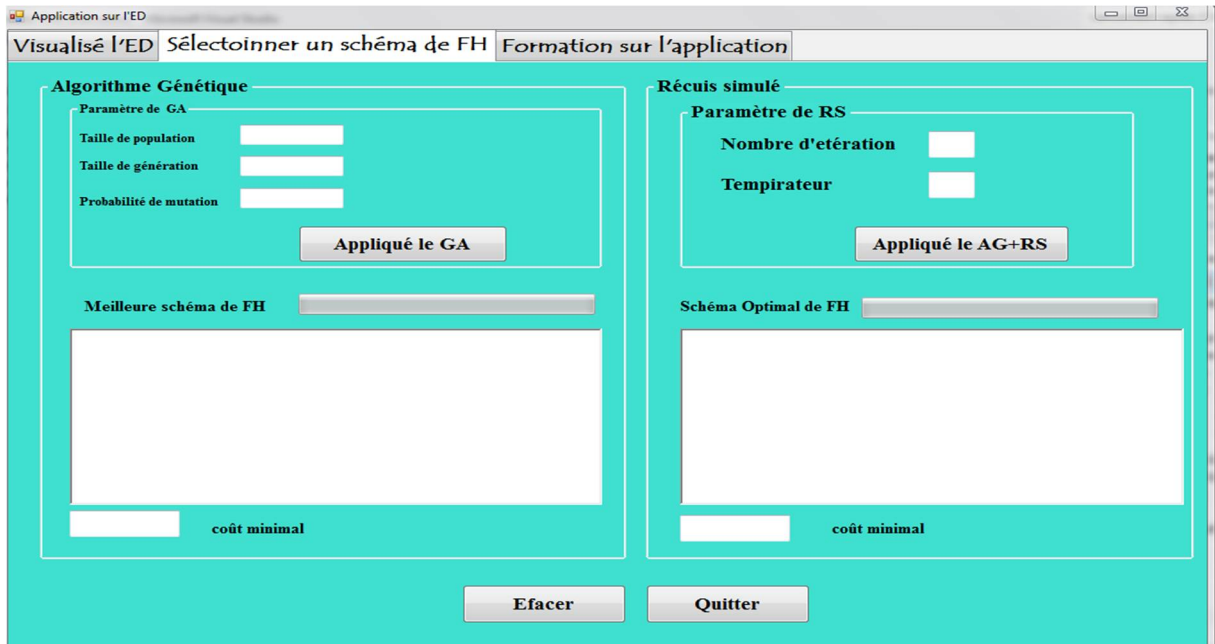


Figure IV.8 : Visualisation d'entrepôt de données.

Visualisation (Visualisé l'ED) : cette interface permet de consulter l'état actuel de l'entrepôt. Cet état comprend les informations concernant les tables, les attributs, les requêtes définies sur l'entrepôt.



Figure IV.9 : Analyse des requêtes.

➤ Application de GA et GA+RS

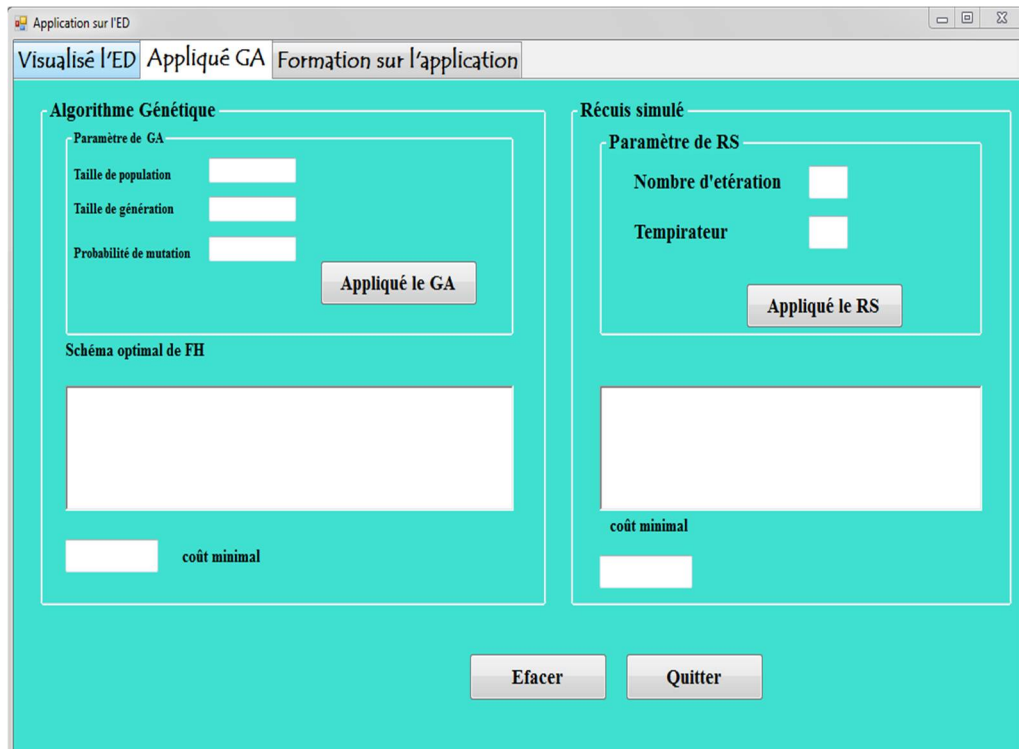


Figure IV.10 : Fenêtre d'application de GA et GA+RS.

5 Résultats obtenus

Pour évaluer les deux méthaheuristiques implémentées, des tests sont faites sur les données du Benchmark APB-1. Un exemple du schéma de fragmentation horizontale généré par chacune des méthodes est présenté dans la figure 4.11 alors que les résultats de comparaison sont dressés dans la table 4.2.

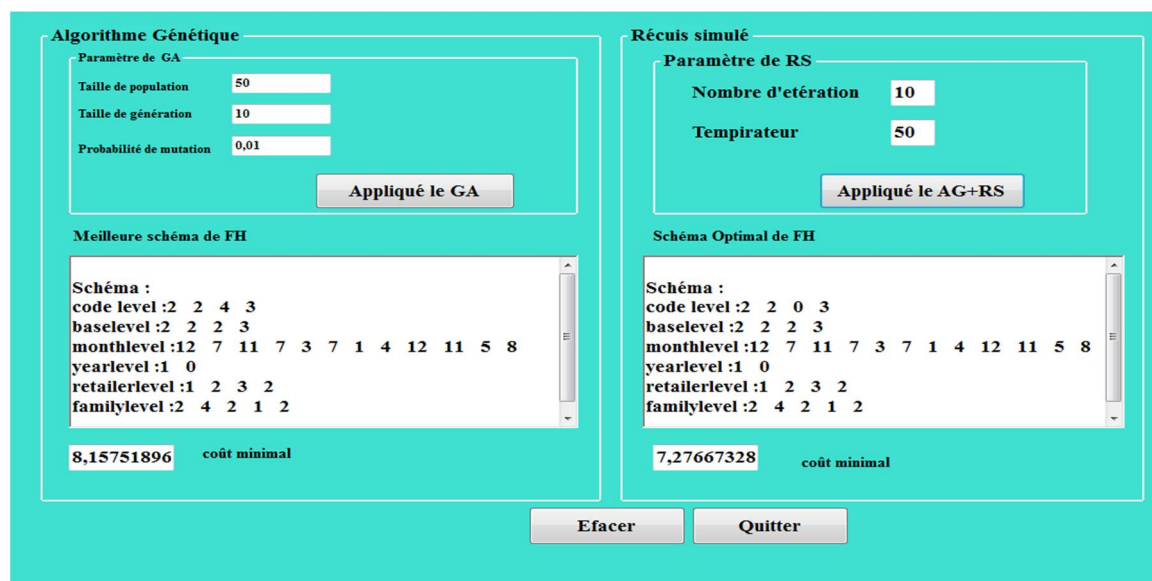


Figure IV.11 : Exemple d'application de l'AG et l'AG+RS.

Nombre d'itérations	Coût AG	Coût AG + RS	Taux d'amélioration
20	10,11164997	8,220117729	19%
30	6,333977811	5,147953442	19%
40	6,203387962	5,017363593	19%
50	3,725502178	2,53947781	33%
100	2,128087589	1,247616395	41%

Table IV.2 : Résultats d'exécutions d'AG et AG + RS avec le taux d'amélioration
(Taille population=50, Pmut=0.01, température = 50)

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

$$Taux = \frac{Coût(AG) - Coût(RS)}{Coût(AG)} \times 100 \quad (1)$$

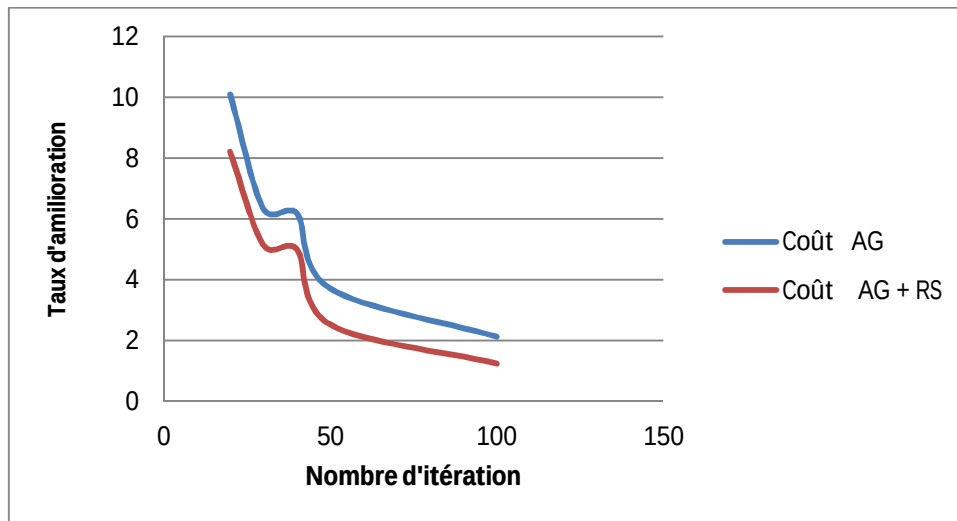


Figure IV.12 : Évaluation de coûts d'exécutions en fonction de nombre d'itérations.

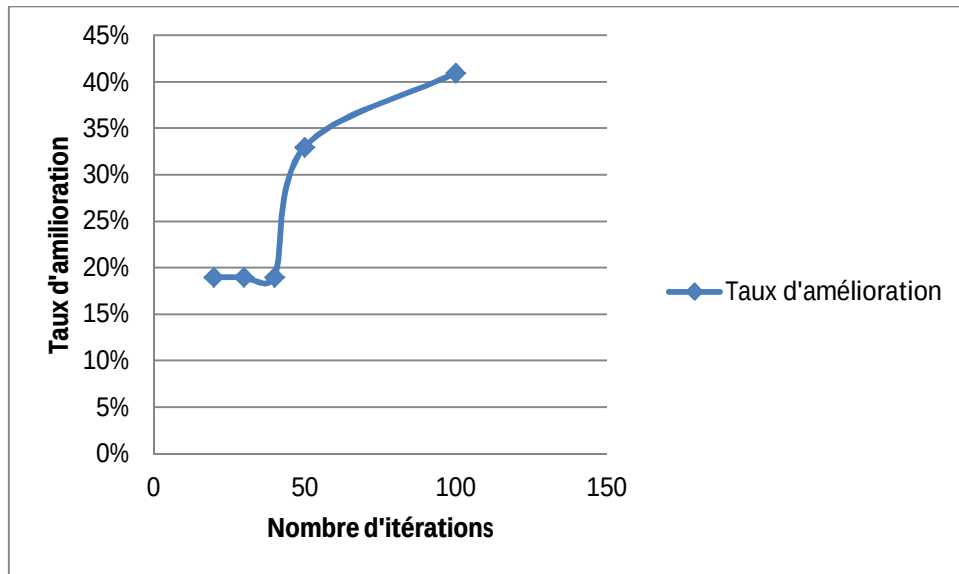


Figure IV.13 : Évaluation de taux d'amélioration en fonction de nombre d'itérations.

5.1 Analyse des résultats

- ✓ Le coût retourné par les deux méthodes diminue en fonction de nombre d'itérations, on peut remarquer aussi que le coût donné par **AG+SR** est meilleur que le coût donné par AG seul.
- ✓ Le Taux d'amélioration augmente en fonction du nombre d'itération ce qui indique qu'il y a une amélioration du coût par **AG+RS**.

6 Conclusion

Dans ce chapitre nous avons mis en évidence toutes les techniques nécessaires à la réalisation de notre travail. Nous avons tout d'abord commencé par indiquer la présentation de l'environnement de développement et les différents outils utiles et nécessaires d'implémentation, Par la suite, nous avons présenté l'architecture de notre application, la procédure de travail et quelques interfaces de l'application. Enfin, nous avons présenté nos résultats obtenus par les deux méthodes **AG** et **AG+RS**.

Conclusion générale

Conclusion générale

Dans le cadre de ce travail de master, nous avons vu et traité la complexité et la difficulté du problème de sélection de la technique de fragmentation horizontale dans un entrepôt de données. Au cœur de ce mémoire, nous avons présenté les différentes étapes de l'adaptation et la réalisation de notre application, nous avons commencé par la présentation générale des entrepôts de données, définition, caractéristique et l'architecture, ainsi les techniques d'optimisation des entrepôts de données. Nous avons présenté ensuite en détail la technique d'optimisation FH ou *fragmentation horizontale*, et après la détermination de problème de fragmentation horizontal « Sélection d'un schéma de fragmentation horizontale ». Et pour résoudre ce problème nous avons présenté en générale deux métaheuristiques: les algorithmes génétiques **GA** et recuis simulé **RS** et après nous avons adapté les GA pour résoudre le problème de sélection et l'algorithme RS pour améliorer les résultats d'algorithme génétique.

L'analyse des résultats présenté montre clairement que l'algorithme GA amélioré le cout de schéma de fragmentation (nombre Entré/Sortie) en fonction de nombre de générations, et aussi les résultats présentent l'amélioration de solution de GA avec l'hybridation '**GA+SR**'.

Comme perspectives à ce travail, Il est préférable d'exploiter la solution retournée par GA et GA+RS pour fragmenter l'entrepôt de données et ensuite l'exécution de la charge de requêtes sur les fragments trouvés pour réduire le temps d'exécution de ces requêtes.

Bibliographie

- [1] A. DAÂS, mémoire magister : Optimisation des requêtes dans le datawarehouse, ECOLE DOCTORALE D'INFORMATIQUE (STIC) Option : Ingénierie des Systèmes Informatiques
- [2] A. Gherboudj, THESE doctorat : Méthodes de résolution de problèmes difficiles académiques, Université de Constantine2, Faculté des Technologies de l'information et de la Communication, Département d'Informatique Fondamentale et ses Applications, 2013
- [3] A. HASSAN, Modélisation des Bases de Données Multidimensionnelles : Analyse par Fonctions d'Agrégation Multiples, thèse de doctorat, UNIVERSITÉ TOULOUSE 1 CAPITOLE, 1décembre 2014.
- [4] A. KERKAD, THESE doctorat : L'interaction au service de l'optimisation à grande échelle des entrepôts de données relationnels, Ecole Doctorale : Sciences et Ingénierie pour l'Information, Mathématiques Secteur de Recherche : INFORMATIQUE ET APPLICATIONS, 2015.
- [5] A. Noaman et k. Barker, « A Horizontal Fragmentation Algorithm for the Fact Relation in a Distributed Data Warehouse », in the 8th International Conference on Information and Knowledge Management, November, 1999.
- [6] A. Sanjay, V. R. Narasayya, and B. Yang, Integrating vertical and horizontal partitioning into automated physical database design. Sigmod (SIGMOD'04), June 2004.
- [7] B. Frédéric et T. Olivier, Construction graphique d'entrepôts et de magasins de données, IRI (Institut de Recherche en Informatique de Toulouse), équipe SIG, Université Toulouse III.
- [8] C. Favre, F. Bentayeb, O. Boussaid, J. Darmont, G. Gavin, N Harbi, N Kabachi et S Loudcher, Les entrepôts de données pour les nuls. . . ou pas !, Université de Lyon ERIC Lyon 2 et ERIC Lyon 1
- [9] Ceiec, Innovation dans la fourniture et la production de statistiques: importance des nouvelles technologies, Finland, 20 et 21 janvier 2000
- [10] E. Benitez, C. Collet, et M. Adiba. Entrepôts de données : caractéristiques et problématique. Revue TSI, 20(2), 2001.
- [11] E. ZIYATI, Optimisation de requêtes OLAP en Entrepôts de Données Approche basée sur la fragmentation génétique, thèse de doctorat, UNIVERSITÉ MOHAMMED VAGDAL, 08 Mai 2010.

- [12] E. Ziyati, L. Bellatreche et K. boukhalfa « La contribution des structures d'optimisation non redondantes dans la conception physique des entrepôts de données » (ISPS'2007 algiers, Alger).
- [13] Hadj MAHBOUBI, «Optimisation de la performance des entrepôts de données XML par fragmentation et répartition », thèse de doctorat, Université Lumière Lyon 2, 08/12/2008
- [14] J.-F. Desnos, Entrepôt de données – Introduction, 2004
- [15] K. Boukhalfa, De la conception physique aux outils d'administration et de tuning des entrepôts de données, Mémoire de doctorat, Ecole Doctorale : Sciences pour l'Ingénieur et Aéronautique, 2009.
- [16] K. Boukhalfa et L. Bellatreche, Combinaison des algorithmes génétique et de recuit simulé pour la conception physique des entrepôts de données, Université de Laghouat, Algérie et LISI/ENSMA France, 2006.
- [17] K. Boukhalfa, L. Bellatreche et B. Ziani, Index de Jointure Binaires: Stratégies de Sélection & Étude de Performances, USTHB - Bp 32 El-Alia Alger, Algérie boukhalk@ensma.fr , LISI/ENSMA - Université de Poitiers , Futuroscope 86960 France bellatreche@ensma.fr ,Université de Laghouat – Algérie bziani@mail.lagh-univ.dz.
- [18] K. Boukhalfa, L. Bellatreche, et Z. Alimazighi, OptAssist : outil d'assistance pour l'optimisation des entrepôts de données relationnels, USTHB Université - Algiers – Algeria, LISI/ENSMA - Poitiers Université – France ,2006.
- [19] K. Ghali, Méthodologie de conception système à base de plateformes reconfigurables et programmables, Mars 2005.
- [20] L. Bellatreche, Techniques d'optimisation des requêtes dans les data warehouses, LISI/ENSMA Téléport2, Avenue Clément Ader ,86960 Futuroscope France, bellatreche@ensma.fr
- [21] L. Bellatreche. Utilisation des vues matérialisées, des index et de la fragmentation dans la conception logique et physique des entrepôts de données". Phd. thesis, Université de Clermont-Ferrand II, Décembre 2000.
- [22] L. Bellatreche et K. Boukhalfa, La fragmentation dans les entrepôts de données : une approche basée sur les algorithmes génétiques, Revue des Nouvelles Technologies de l'Information (EDA'2005), Juin, 2005.
- [23] L.BELLATRECHE, K. BOUKHALFA, et S. Caffiau « ParAdmin: Un Outil d'Assistance à l'Administration et Tuning d'un Entrepôt de Données », rapport de recherche, LISI/ENSMA - Université de Poitiers – Futuroscope.

Bibliographie

- [24] L. BELLATRECHE, K. BOUKHALFA et P. RICHARD, Rapport de recherche N° 01 - 2008 : Fragmentation Primaire et Dérivée: Étude de Complexité, Algorithmes de Sélection et Validation sous ORACLE10g, Université de Poitiers, France 2008.
- [25] L. Bellatreche, K. Boukhalfa et P. Richard. Referential horizontal partitioning selection problem in datawarehouses: Hardness study and selection algorithms. International Journal of Data Warehousing and Mining, 2004
- [26] L. Nathalie, Y. Pouillet., 2008. Revue du Droit des Technologies de l'Information - n° 30/2008, Doctrine Entrepôts de données et vie privée.
- [27] M. BARR, Mémoire de magistère : Approche dirigée par les fourmis pour la fragmentation horizontale des entrepôts de données relationnels, Ecole nationale Supérieure d'Informatique, 2009
- [28] M. BARR et L. BELLATRECHE, Approche dirigée par les fourmis pour la fragmentation horizontale dans les entrepôts de données relationnels, Revue « Nature & Technologie ». n° 06/Janvier 2012.
- [29] M. Bouzeghoub, ENTREPÔTS DE DONNEES ARCHITECTURE ET FONCTIONNALITES, université de Versailles SAINT-QUETIN-EN-YVELINES, Laboratoire PRISM <http://www.prism.uvsq.fr> .
- [30] M. Merouani et E. Abdelouarit , Les index pour les entrepôts de données : comparaison entre index arbre-B et Bitmap, Faculté poly disciplinaire de Tétouan et Faculté des sciences de Tétouan, <https://hal.archives-ouvertes.fr/hal-00842579> , 9 Jul 2013
- [31] M. T. Özsu and P. Valduriez. Principles of Distributed Database Systems ,1999 .
- [32] M. Teisseire, H. Alatriza Salas et F. Bouillot, Déroulement Du cours: Entrepôt De données et l'Analyse en ligne, (EPSI l'école d'ingénieurs informatique)
- [33] M. VAN CANEGHEM, « Le voyageur de commerce Algorithme “branch and bound”, Algorithme Glouton, Méthode de recherche locale », support de cours, Département d'Informatique de la Faculté des Sciences de Luminy Université de la Méditerranée LIM : Laboratoire d'Informatique de Marseille, Décembre 2002.
- [34] O. Teste, Modélisation et manipulation d'entrepôts de données complexes et historisées, thèse doctorat, UNIVERSITE PAUL SABATIER DE TOULOUSE (SCIENCES), 18 décembre 2000.

Bibliographie

- [35] P. Frédérique Et T. Odile, A propos d'un entrepôt de données universitaire : modélisation des acteurs et méta données, LORIA, Campus Scientifique; Frederique.Peguiro@loria.fr , Odile.Thiery@loria.fr
- [36] R. BOUCHAKRI, thèse doctorat : Conception physique statique et dynamique des entrepôts de données, Écoles Doctorales, Sciences et Technologies de l'Information et de la Communication (STIC, ESI) Sciences et Ingénierie pour l'Information, Mathématiques (S2IM, ISAE-ENSMA), 17 Septembre 2015
- [37] R. BOUCHAKRI, Mémoire Pour l'obtention du diplôme de Magistère : Une approche dirigée par la classification des attributs pour fragmenter et indexer des entrepôts de données, (ESI) (Oued Semar, Alger), 2009
- [38] R. Bouchakri et L. Bellatreche, Sélection Incrémentale d'un Schéma de Fragmentation Horizontale d'un Entrepôt de Données Relationnel, Université de Poitiers France, Ecole nationale Supérieure d'Informatique (ESI) Alger.
- [39] R. Duvigneau, Introduction aux Méthodes d'Optimisation sans Gradient pour l'Optimisation et le Contrôle en Mécanique des Fluides, Ecole de printemps OCET Optimisation et Contrôle des Ecoulements et des Transferts, 12 - 17 Mars 2006.
- [40] R. Godin, C. Desrosiers, LOG660 - Bases de données de haute performance : Les entrepôts de données et l'analyse de données, Département de génie logiciel et des TI, Hiver 2011.
- [41] R. Moussa, Systèmes de Gestion de Bases de Données Réparties & Mécanismes de Répartition avec Oracle, Ecole Supérieure de Technologie et d'Informatique à Carthage, Année Universitaire 2005-2006.
- [42] S. Amédée et R. Francois-Gérard, ALGORITHMES GENETIQUES, TE de fin d'année Tutorat de Mr Philippe Audebaud, 21/06/2004.
- [43] S. BENKRID, THESE doctorat : Le déploiement, une phase à part entière dans le cycle de vie des entrepôts de données : application aux plateformes parallèles, École nationale Supérieure d'Informatique
- [44] S. SAICHI, Mémoire Pour l'obtention du diplôme de Magistère : Optimisation de requêtes dans les entrepôts de données, UNIVERSITE D'ORAN ES-SENIA, FACULTE DES SCIENCES DEPARTEMENT D'INFORMATIQUE, 27 juin 2009

Bibliographie

- [45] S Silhadi-Hacid et M Tarafi Du DataWarehouse au WebHouse : le croisement des réseaux et des bases de données
- [46] S. Ramkumar, Algorithmes génétiques, 26 avril 2007
- [47] T. Martin, Microsoft Visual basic 2012-2013: mon eFormation,(page 25).
- [48] T. Olivier, Modélisation et manipulation d'entrepôts de données complexes et historisées, Décembre 2000.
- [49] T. Vallée et M.Yildizoglu, Présentation des algorithmes génétiques et de leurs applications en économie, Université de Nantes et Université Montesquieu Bordeaux IV, 7 septembre 2001.
- [50] V G. Turreau, C#, Publie par Pearson Education France, 2010.
- [51] W. Inmon. H. Building the Data Warehouse. John Wiley & Sons, 1996.
- [52] http://igm.univ-mlv.fr/~drXPOSE2013tleroux_genetic_algorithmindex.html.
- [53] <http://www.egilia.com/formation-oracle-10g-11g>, consulté le : 10/05/2016
- [54] <http://computertrainingcenters.com/top-10-features-of-visual-studio-2012>, consulté le : 5/05/2016
- [55] Wikipidia, <http://www.Wikipidia.com/> Visual Studio, consulté le : 10/05/2016

Annexe

1. La création des tables de dimensions

Table CHANLEVEL

```
CREATE table "CHANLEVEL" (  
    "BASE_LEVEL" VARCHAR2(50),  
    "ALL_LEVEL" VARCHAR2(50),  
    constraint "CHANLEVEL_PK" primary key ("BASE_LEVEL")  
)
```

Table CUSTLEVEL

```
CREATE table "CUSTLEVEL" (  
    "STORE_LEVEL" VARCHAR2(50),  
    "RETAILER_LEVEL" VARCHAR2(50),  
    constraint "CUSTLEVEL_PK" primary key ("STORE_LEVEL")  
)
```

Table PRODLEVEL

```
CREATE table "PRODLEVEL" (  
    "CODE_LEVEL" VARCHAR2(50),  
    "CLASS_LEVEL" VARCHAR2(50),  
    "GROUP_LEVEL" VARCHAR2(50),  
    "FAMILY_LEVEL" VARCHAR2(50),  
    "LINE_LEVEL" VARCHAR2(50),  
    "DIVISION_LEVEL" VARCHAR2(50),  
    constraint "PRODLEVEL_PK" primary key ("CODE_LEVEL")  
)
```

Table TIMELEVEL

```
CREATE table "TIMELEVEL" (  
    "TID" VARCHAR2(50),  
    "MONTH_LEVEL" VARCHAR2(50),  
    "QUATER_LEVEL" VARCHAR2(50),  
    "YEAR_LEVEL" VARCHAR2(50),  
    constraint "TIMELEVEL_PK" primary key ("TID")  
)
```

2. Création de la table des faits : ACTVARS

```
CREATE table "ACTVARS" (  
    "CUSTOMER_LEVEL" VARCHAR2(50),  
    "PROUCT_LEVEL" VARCHAR2(50),  
    "CHANNEL_LEVEL" VARCHAR2(50),  
    "TIME_LEVEL" VARCHAR2(50),  
    "UNITSSOLD" BINARY_FLOAT,  
    "DOLLARSAES" BINARY_FLOAT,  
    "DOLLARCOST" BINARY_FLOAT  
)
```

-Create/ foreign key constraints

```
ALTER TABLE "ACTVARS" ADD CONSTRAINT "ACTVARS_FK"  
FOREIGN KEY ("CUSTOMER_LEVEL")  
REFERENCES "CUSTLEVEL" ("STORE_LEVEL")  
  
/  
  
ALTER TABLE "ACTVARS" ADD CONSTRAINT "ACTVARS_FK2"  
FOREIGN KEY ("PROUCT_LEVEL")  
REFERENCES "PRODLEVEL" ("CODE_LEVEL")  
  
/  
  
ALTER TABLE "ACTVARS" ADD CONSTRAINT "ACTVARS_FK3"  
FOREIGN KEY ("CHANNEL_LEVEL")  
REFERENCES "CHANLEVEL" ("BASE_LEVEL")  
  
/  
  
ALTER TABLE "ACTVARS" ADD CONSTRAINT "ACTVARS_FK4"  
FOREIGN KEY ("TIME_LEVEL")  
REFERENCES "TIMELEVEL" ("TID")
```

2.2. Liste des requêtes utilisée

Q1:

```
Select Time_level,count(*)  
From ACTVARS A, PRODLEVEL P  
Where  
A.PRODUCT_LEVEL=P.CODE_LEVEL and  
P.Class_LEVEL='A1DGFSPJTJ473'  
Group by Time_level
```

Q2:

```
Select Time_level,sum(dollarcost)  
From ACTVARS A, PRODLEVEL P
```

Where

A.PRODUCT_LEVEL=P.CODE_LEVEL and
P.Class_LEVEL='MVOSML0DZA5V'

Group by Time_level

Q3:

Select Time_level,count(*)

From ACTVARS A, PRODLEVEL P

Where

A.PRODUCT_LEVEL=P.CODE_LEVEL and
P.Class_LEVEL='ZYXHAYKT707N'

Group by Time_level

Q4:

Select Time_level,Avg(unitssold)

From ACTVARS A, PRODLEVEL P

Where

A.PRODUCT_LEVEL=P.CODE_LEVEL and
P.Class_LEVEL not in('A1DGFSPJTJ473', 'MVOSML0DZA5V', 'ZYXHAYKT707N')

Group by Time_level

Q5:

Select Time_level,count(*)

From ACTVARS A, PRODLEVEL P

Where

A.PRODUCT_LEVEL=P.CODE_LEVEL and
P.group_LEVEL='CSZD5QO5AW6J'

Group by Time_level

Q6:

Select Time_level,Max(unitssold)

From ACTVARS A, PRODLEVEL P

Where

A.PRODUCT_LEVEL=P.CODE_LEVEL and
P.group_LEVEL<>'CSZD5QO5AW6J'

Group by Time_level

Q7:

Select Time_level,count(*)

From ACTVARS A, PRODLEVEL P

Where

A.PRODUCT_LEVEL=P.CODE_LEVEL and
P.family_LEVEL='UHVQ8L35XZFF'

Group by Time_level

Q8:

Select Time_level,sum(dollarcost)

From ACTVARS A, PRODLEVEL P

Where

A.PRODUCT_LEVEL=P.CODE_LEVEL and
P.family_LEVEL='IIFSK6AKVBON'

Group by Time_level

الملخص :

اكتسبت مستودعات البيانات مكانة كبيرة في وقتنا الراهن لأهميتها الواضحة في اتخاذ القرارات حيث اعتبر الباحثون تطويرها وتسهيل الوصول إلى المعلومات أولوية كبيرة. وتعتبر تقنية التجزئة الأفقية من أبرز واحسن التقنيات في تحسين مستودع . في هذا العمل المخصص لمذكرة الماستر قمنا بتطبيق طريقة تهجين بين الخوارزميات الجينية و خوارزمية التخمير المحاكى لحل مشكل تحديد مسار التقسيم الأفقي لمستودع البيانات حيث استخدمنا مستودع البيانات بنشمارك باعتماد وكلغة برمجة اخترنا سي شارب .

الكلمات المفتاحية : مستودعات البيانات ، الخوارزميات الجينية، التخمير المحاكى، بنشمارك، الأوراكل و السي شارب

Résumé :

Les entrepôts de données occupent une place importante dans nos jours, grâce à son grand rôle dans la prise de décision, les chercheurs ont considéré que son développement est en priorité. Parmi les meilleures techniques d'optimisation des entrepôts de données c'est la fragmentation horizontale.

Dans notre travail, nous avons appliqué une méthode hybride combinant un algorithme génétique et un algorithme de recuit simulé pour résoudre le problème de Sélection d'schéma FH où en utilisant le banc d'essai Benchmark APB-1 en comptant sur Oracle, et comme langage de programmation on a choisi C #.

Mots clés : Entrepôt de données, Algorithmes génétiques, recuit simulé, Benchmark, Oracle et C#.

Abstract:

The data warehouse gained a big standing in our current time for its clear importance in making decision, where the researchers considered its development and facilitation of the arrival to the information big priority. The horizontal division Technique considers of the most prominent and best techniques in improving warehouse. In this specified work for master dissertation, we propose a hybrid method combining a genetic algorithm and a simulated annealing algorithm to solve the problem of determining horizontal partitioning of data warehouse where we have used data warehouse Benchmark adoption and chose C # as a programming language.

Keywords: Data warehouses, the genetic algorithms, simulated annealing, Benchmark, Oracle and C#.