Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Ibn Khaldoun -Tiaret-

Faculté des sciences et sciences de l'ingénieur
Département d'Informatique

École Doctorale

Sciences et Technologies de l'Information et de Communication
STIC

Mémoire

Pour l'obtention du diplôme de Magistère

Option : Système d'Information et de Connaissance (SIC)

Présenté par

Abdelkader Ouared

La Contribution des Agents pour la Conception Physique des Entrepôts de Données Relationnels

Membres d'évaluation du mémoire :

Mlle. Latifa MAHDAOUI	**M.C-A, USTHB, Alger**	**Président**
Mr. Samir KECHID	**M.C-A, USTHB, Alger**	**Examinateur**
Mr. Youcef DAHMANI	**M.C-A UIK, Tiaret**	**Examinateur**
Mr. Ladjel BELLATRECHE	**Pr. ENSMA Poitiers**	**Directeur de mémoire**
Mr. Kamel BOUKHALFA	**M.C-A, USTHB, Alger**	**Co-Directeur de Mémoire**

ANNEE UNIVERSITAIRE 2011/2012

Remerciements

Au terme de ce travail de recherche, je tiens à présenter mes vifs remerciements à toutes les personnes qui ont contribué, de près ou de loin, à accomplir ce présent travail.

Je tiens à remercier le Professeur Ladjel BELLATRECHE et le Docteur Kamel BOUKHALFA pour avoir encadré mon travail de magistère et pour leur entière disponibilité et tous leurs conseils avisés.

Je tiens à remercier très sincèrement l'ensemble des membres du jury qui me font le grand honneur d'accepter de juger mon travail.

- Mlle. Latifa MAHDAOUI M.C-A, USTHB, Alger Président
- Mr. Samir KECHID M.C-A, USTHB, Alger Examinateur
- Mr. Youcef DAHMANI M.C-A UIK-Tiaret Examinateur

.

Résumé

La fragmentation horizontale et les index de jointure binaire sont deux techniques d'optimisation importantes utilisées dans le contexte des entrepôts de données relationnels. Une forte similarité a été identifiée entre ces deux techniques. Le processus de sélection des schémas d'indexation et de fragmentation est souvent basé sur un ensemble d'attributs de sélection utilisés par une charge de requêtes. Le nombre de schémas de fragmentation et d'indexation est proportionnel au nombre d'attributs candidats. Il peut être très grand dans un entrepôt de données réel rendant le processus de sélection de ces techniques très complexe. Nous proposons dans cet article, une méthodologie de sélection de schémas de fragmentation horizontale et d'index de jointure binaires basée sur les Systèmes Multi-agent. Nous réalisons des expérimentations sur un banc d'essai pour évaluer notre approche.

Mots clés - entrepôt de données; index de jointure binaire; la fragmentation horizontale; requêtes décisionnelles; Système Multi-agent

Abstract

The horizontal partitioning and binary join index are two important optimization techniques used in the context of relational data warehouses. A strong similarity was found between the horizontal partitioning (HP) and the binary join index (JBI). The process of selecting indexing schemes and partitioning is often based on a set of attributes used by a selection request load. The number of fragmentation patterns and indexing is proportional to the number of attributes. This latter may be important in a real data warehouse the process of selecting leading to be more complex. In this paper, we propose a methodology for selecting schemes of HP and JBI based on Multi-Agent Systems. We realize experiments upon benchmark to evaluate our approach.

Keywords: data warehouse; Binary Join Index, Horizontal partitioning, decision-support queries, Multi-Agent System

Sommaire

Table des figures

Liste des tableaux

Introduction Générale

Un entrepôt de données est une base de données très grande qui intègre l'information extraite à partir de points d'émission multiples, indépendants, hétérogènes pour soutenir des activités d'analyse commerciale et des tâches de prise de décision. Les entrepôts de données sont particulièrement conçus pour permettre à des cadres, des directeurs, et des analystes de prendre rapidement les meilleures décisions [2].

Les applications des entrepôts de données incluent le marketing, les affaires, la gestion de Reporting, le budget, les prévisions, les soins de santé, etc. Les utilisateurs sont principalement intéressés par une mesure de l'information en fonction de quelques aspects d'affaires (dimensions) ; Par exemple, considérer un entrepôt qui garde l'information relative aux ventes dans des compagnies. Pour chaque événement de ventes, l'information dimensionnelle intéressante peut être le produit vendu, la période de la vente, et le client.

Dans la pratique, le nombre de dimensions pour un entrepôt de données peut être relativement grand, et chaque dimension peut avoir un certain nombre d'attributs distincts qui sont stockés dans une table séparée de dimension (par exemple, les attributs du « produit » pourraient être sa « couleur », sa « taille », son « poids », etc.). Les requêtes d'utilisateur spécifient typiquement ces attributs, et la préparation d'une réponse à une requête peut impliquer une recherche étendue par un certain nombre de tables de dimension pour des valeurs d'attribut appropriées. En conséquence, elle peut être tout à fait longue pour répondre directement à des requêtes agrégat à partir de données stockées dans la base de données.

Afin de répondre au besoin de l'utilisateur, plusieurs structures d'optimisation ont été proposées pour améliorer la performance de ces requêtes complexes d'analyse principalement : les index, les vues matérialisées et la fragmentation horizontale et verticale. Sélectionner un ensemble optimal de chacune de ces structures est un problème NP-Complet [1]. Cette sélection n'est pas toujours suffisante pour optimiser toute la charge de requêtes. Pour satisfaire le maximum de requêtes ; il est indispensable d'opter pour une sélection multiple de structures comme la FH et les IJBs (voir [1] [32] [37]) ce qui engendre une complexité énorme.

Dans le contexte d'entrepôts de données, les applications réelles manipulent un nombre de 100, 200 attributs ou plus. Par conséquent, la sélection d'un schéma de fragmentation horizontale et une configuration d'index sur un tel ensemble d'attribut devient une tâche difficile [1]. Ce problème de sélection constitue un aspect très important dans la phase

de conception physique des entrepôts de données. Durant cette phases, l'administrateur est amené à effectuer plusieurs choix concernant plusieurs niveaux : les techniques d'optimisation utilisées, la manière de leur sélection (isolée ou combinée), les attributs les plus adaptés à chaque technique et les algorithmes de sélection utilisés. Ainsi, les changements effectués durant le cycle de vie de l'entrepôt de données doivent être considérés dans trois aspects pour éviter sa dégradation [1] :(1) le schéma et le contenu des tables, (2) la taille des techniques d'optimisation déjà sélectionnées et (3) la charge des requêtes sur laquelle la sélection des techniques d'optimisation a été établie.
Effectuer ces choix manuellement nécessite un temps d'administration non négligeable. Nous constatons un besoin crucial concernant le développement des outils d'administration des entrepôts de données.

Nous proposons une nouvelle démarche de sélection combinée d'Index de Jointure Binaire (IJB) et un schéma de Fragmentation Horizontale (FH) dirigée par les Systèmes Multi-agents (SMA) pour aider l'administrateur dans ses tâches de conception physique. Ce systeme nous montre l'aspect distribué, les interactions et les relations qui peuvent avoir lieu entre ces techniques d'optmisation .

L'idée d'utiliser le SMA est motivée par le fait que les deux techniques FH et IJB ayant une forte similarité [1], sont concurrentes sur la même ressource représentant l'ensemble des attributs de sélection définis sur les tables de dimension et pour prendre l'impact de l'indexation sur la fragmentation.

Notre démarche permet d'effectuer l'élagage de l'ensemble d'attribut et de le partager entre l'ensemble des agents qui simule la sélection de deux technique IJB et FH. Les agents sélectionnent les index et les fragments les plus efficaces, et prennent en charge l'interaction inter et intra techniques pour juger les attributs les mieux adaptés pour la fragmentation, l'indexation ou les deux et intégrer la dynamique dans notre solution pour pouvoir considérer plusieurs scénarios durant le processus de la sélection.

Notre mémoire est organisé en quatre chapitres :

Dans le premier chapitre, nous présentons un état de l'art sur les techniques d'optimisation de l'entrepôt de données, nous commencerons par une introduction au concept général d'un entrepôt, nous aborderons ses implémentations physiques puis nous parlerons de la problématique liée aux entrepôts. Par la suite, nous abordons les techniques d'optimisation: définition, concept général et classification des techniques

Le second chapitre traite le problème de sélection isolée d'index et la fragmentation horizontale, ainsi que de la sélection simultanée de ces structures. Nous présenterons un état de l'art sur les travaux effectués dans ce sens, et une comparaison des travaux pour

chaque technique citée. Nous y détaillerons aussi les principaux travaux proposés pour résoudre ces problèmes

Dans le troisième chapitre nous présenterons notre contribution. Nous modéliserons notre système ainsi que les approches utilisées par un système multi-agent afin de résoudre les problèmes de sélection des structures d'optimisation de l'entrepôt de données. Par la suite nous présenterons la projection de ce système sur le problème de sélection d'IJB et FH. Cette partie est consacrée à la description détaillée de l'approche proposée. Nous décrivons la démarche de sélection des d'IJB et FH et les concepts importants pour réaliser notre approche comme le modèle de coût ainsi que les algorithmes de sélection implémentés.

Le dernier chapitre sera consacré à la validation de notre approche de sélection et du système multi-agent. Nous présenterons la description de l'outil que nous avons implémenté et nommé « AdminSMA ». C'est un outil assistant les administrateurs dans leur tâche de conception physique. Cet outil doit fournir à l'administrateur un ensemble de recommandations concernant les techniques d'optimisation et les ressources utilisées par ces dernières. L'étude expérimentale et les tests effectués sur un entrepôt de données généré par le banc d'essais APB-1 sous *Oracle* 10g et sa charge de requêtes décisionnelle.

Chapitre I : Les Structures d'optimisation dans un Entrepôt de Données Relationnel

Introduction

Le système d'information décisionnel offre un support d'aide à la décision en rassemblant et en homogénéisant les données dans des entrepôts de données qui intègrent une architecture servant de fondation pour l'exploitation et l'analyse des indicateurs pertinents à la mesure de l'activité de l'entreprise. Ces entrepôts peuvent être conçus d'une manière multidimensionnelle ou relationnelle. Par rapport au modèle relationnel, les entrepôts de données sont de deux types ; en flocon de neiges ou en étoile. En suivant l'architecture en étoile, l'entrepôt est constitué d'une table de faits volumineuse, entourée de plusieurs tables de dimensions qui contiennent les axes d'analyse.

Afin de permettre aux décideurs de procéder à l'analyse des données stockées dans l'entrepôt, cette dernière est souvent réalisée par l'intermédiaire de requêtes complexes caractérisées par des opérations de sélections, de jointures et d'agrégations. Par conséquent, l'exécution de ces requêtes devient de plus en plus complexe et nécessite un temps de réponse très élevé qui n'est pas acceptable par les décideurs.

Pour réduire ce dernier et satisfaire les besoins des décideurs, l'administrateur de l'entrepôt est amené à faire une bonne conception physique qui consiste à sélectionner et combiner un ensemble de techniques d'optimisation qu'il considère pertinentes pour l'ensemble des besoins des décideurs.

Ces techniques ou structures de données peuvent être des index, des vues matérialisées, ou la technique de la fragmentation (horizontale ou verticale).

Nous présentons dans cette section la problématique liées à notre contexte de travail. Nous présentons la conception physique des entrepôts de données, nous commençons par présenter l'architecture et la modélisation des entrepôts de données, nous abordons ensuite les principales techniques d'optimisation, leurs définitions et leurs différents concepts.

I.1 L'entrepôt de données(ED)

I.1.1 Définitions

Dans [2], Immon le père du concept « datawarehouse », le définit comme suit : "*Le data warhouse est une collection de données orientées sujet, intégrées, non volatiles et historisées, organisées pour le support d'un processus d'aide à la décision* ».

Dans [3], une deuxième définition est donnée :

« Un entrepôt de données sert à consolider les données stratégiques de l'organisation et les mettre à la disposition des gestionnaires à des fins d'aide à la décision.
Les nombreuses données accumulées dans les fichiers et les bases de données des systèmes transactionnels sont souvent dispersées, disparates, incohérentes, mal connues, et leur mise à jour n'est pas toujours synchronisée.
Il s'agit alors de réorganiser ces grandes masses de données opérationnelles en fonction de quatre caractéristiques: intégration, orientation sujet, sensibilité au temps, non volatilité, et d'un objectif: l'aide à la décision ».
Ces définitions mettent l'accent sur les caractéristiques suivantes des informations incluses dans l'entrepôt, à savoir :

1- L'information doit être thématique, c'est-à-dire liée à un métier par exemple finances, ressources humaines, production,…etc.
2- Elle doit être organisée pour aider à la décision.
3- Elle doit être historiée pour permettre au décideur d'aller dans le temps et faire des comparaisons et des projections dans le temps pour pouvoir donner des explications et des interprétations relative à l'évolution de cette information.

I.1.2 Architecture d'un entrepôt de données

L'entrepôt de données stocke des données pertinentes aux besoins de prise de décision issues de plusieurs sources hétérogènes (systèmes opérationnels de l'entreprise, fichiers textes, documents web…), puis de les organiser et les intégrer dans l'entrepôt d'une manière multidimensionnelle afin de les mettre en disposition des décideurs (Voir Figure I-1). Une fois conçu, l'entrepôt est exploité par plusieurs outils d'aide à la décision comme les outils de datamining, d'analyse des tendances ou de prédiction.

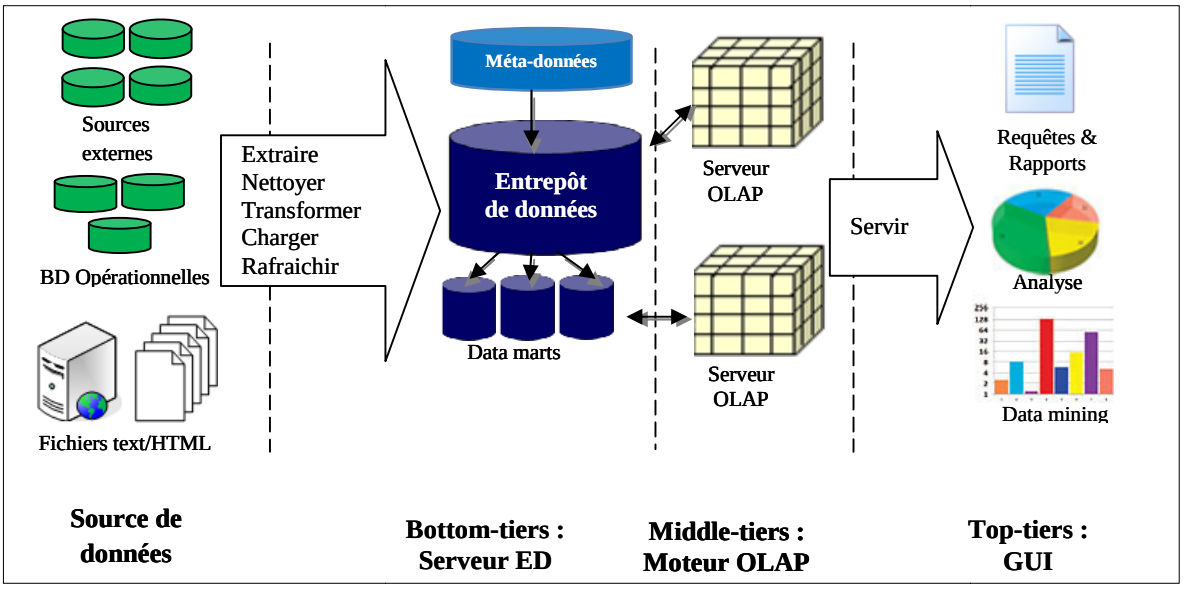


Figure I-1:Architecture conceptuelle d'un entrepôt de données

I.1.3 Les modèles de données

Un entrepôt de données est organisé de façon à s'adapter à des requêtes de type 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, et demandent beaucoup de temps de calcul. Pour ce type d'environnement, une nouvelle approche de modélisation a été suggérée : les modèles multidimensionnels.

I.1.3.1 Les modèles multidimensionnels

La modélisation multidimensionnelle a été introduite par Ralph Kimball [2]. Elle consiste en deux nouveaux concepts les faits et les dimensions. Chaque modèle multidimensionnel est composé d'une table contenant une clé, la table des faits qui permet de mesurer l'activité et d'un ensemble de tables dimensionnelles qui contiennent les informations contextuelles faisant varier les mesures de l'activité en question. La table de faits comporte des clés étrangères vers les tables de dimension qui les relie.

L'objectif majeur de cette modélisation est la vision multidimensionnelle des données, ce qui est assuré via le concept de cube de données; ce dernier organise les données en une ou plusieurs dimensions qui déterminent une mesure d'intérêt.

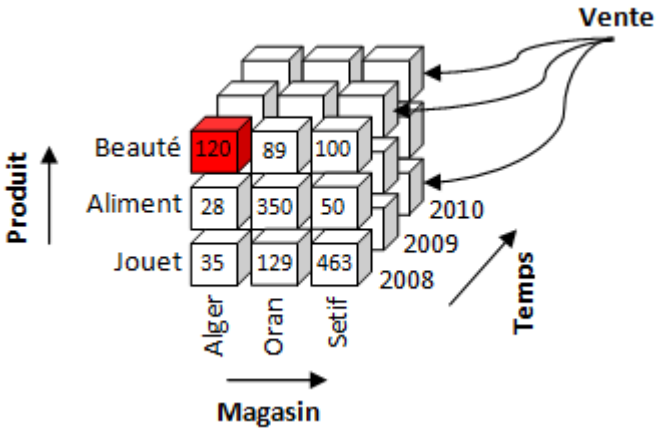


Figure I-2:Vue multidimensionnelle des données

I.1.3.2 Implémentation des modèles multidimensionnels

Techniquement, il existe deux modèles de stockage physique des données pour construire des systèmes basés sur un modèle multidimensionnel.

1) Les systèmes MOLAP

Ce modèle est caractérisé par la forme de tableaux Multi-dimensionnels, il est géré par un SGBD Multi-dimensionnel. Une dimension du cube est équivalente à une dimension au niveau du tableau. Tout le contenu dans cette représentation est utile. Les contenus agrégés sont pré-calculés et expliquent sa performance contre la difficulté de mise à jour et de gestion. Cette structure est valable pour une situation peu évolutive et ne dépassant pas les quelques giga-octets. Les MOLAP sont consommateurs d'espaces dans le cas de dispersion de données, une telle configuration nécessite une compression [4].

2) Les systèmes ROLAP

Dans ce modèle, la gestion des données est assurée par un SGBD Relationnel type. Néanmoins une représentation multi-dimensionnelle est présente et traduite par un moteur OLAP qui prend en charge aussi les pré-calculs dérivés et agrégés, ainsi que l'exécution des requêtes sur les vues matérialisées. Le schéma conceptuel du modèle ROLAP comporte une table appelée table de faits qui contient les mesures ou indicateurs, et des tables appelées des dimensions qui contiennent les dimensions ou axes d'analyses [4].
Nous présentons dans ce qui suit, trois modèles de données relationnels : schéma en étoile, flocon de neige et schéma en constellation.

I.1.3.2 Schéma en étoile

Ce type de schéma est largement utilisé par les industriels. Il contient une table des faits normalisée et des tables de dimension qui sont généralement dé-normalisées afin de minimiser le nombre de jointures nécessaires pour évaluer une requête.
Les requêtes typiques de ce schéma, nommées requêtes de jointure en étoile (star-join queries) ont les caractéristiques suivantes :

- Il y a des jointures multiples entre la table des faits et les tables de dimension.
- Il n'y a pas de jointure entre les tables de dimensions.
- L'opération de sélection est souvent effectuée sur les tables de dimensions

La syntaxe générale de ces requêtes est la suivante :

SELECT<Liste de projection><Liste d'agrégation>
FROM<Nom de la table des faits><Liste de noms de tables de dimension>
WHERE<Liste de prédicats de sélection & jointure>
GROUPBY<Liste des attributs de tables de dimension>

La Figure I-3 illustre un schéma en étoile où la table des faits VENTES stocke la quantité et le montant de vente d'un article pour un site donné. Les tables correspondant à PRODUIT, à TEMPS et à MAGASIN comportent les informations pertinentes sur ces dimensions.

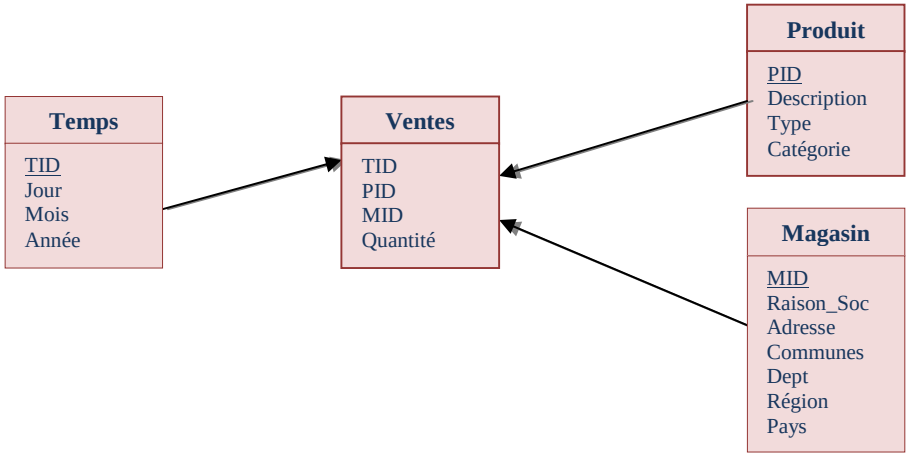


Figure I-3 : Un exemple d'un schéma en étoile

I.1.3.3 Schéma en flocon de neige

Ce schéma est un raffinement du schéma en étoile où certaines tables de dimensions sont normalisées, autrement dit, certaines tables de dimensions sont décomposées selon leur hiérarchie afin de les normaliser. Figure I-4 illustre ce modèle.

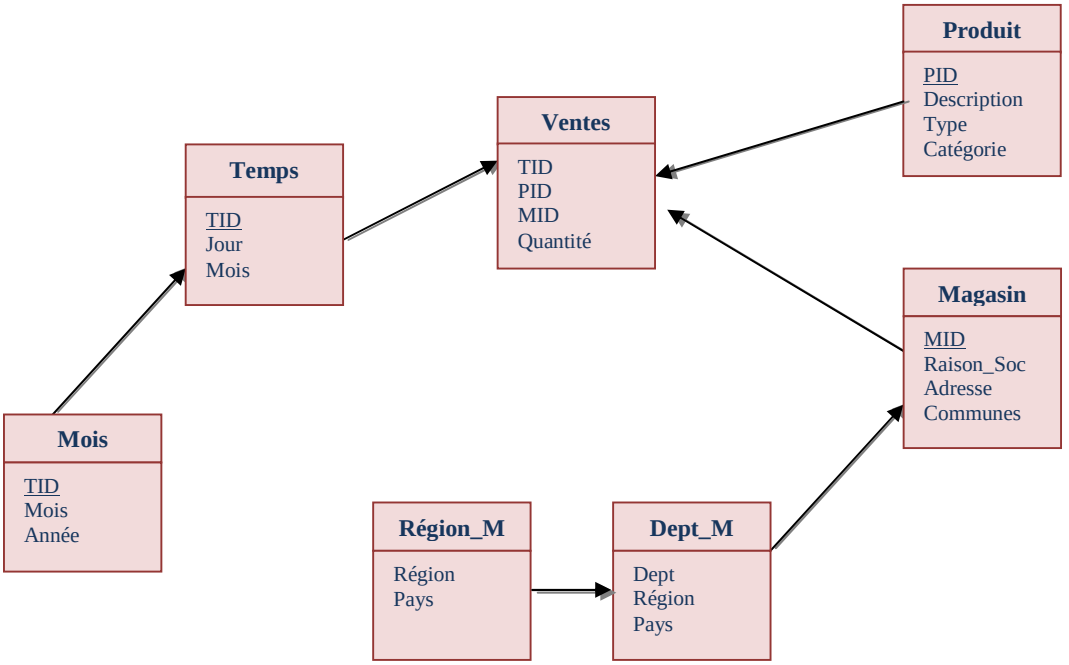


Figure I-4:Un exemple d'un schéma en flocon en neige

I.1.3.4 Schéma en constellation

Ce schéma comprend plusieurs tables de faits et des dimensions communes ou non. Ce modèle résulte certainement de la fusion de plusieurs modèles en étoiles.

I.1.4 Problématique liée aux entrepôts de données

La volumétrie de la table de faits et la multitude de jointures entre elles et les tables de dimensions rendent l'extraction des informations, via des requêtes décisionnelles, une tâche coûteuse en temps et en espace. Pour réduire le temps de réponse des requêtes décisionnelles, il y a lieu d'utiliser des structures et des techniques d'optimisation, comme les index, les vues matérialisées, les techniques de fragmentation horizontale, etc. Dans la littérature, les chercheurs ont démontré que le problème de sélection de chaque structure (technique) est un problème NP-complet [1]. Cette complexité de l'opération de sélection a obligé les chercheurs à recourir à des méthodes approchées ou «métaheuristiques» pour résoudre ce problème en un temps polynomial.

L'exploitation de l'entrepôt de données génère plusieurs problématiques comme :

(1) L'augmentation du volume de données peut nécessiter l'augmentation de l'espace de stockage. Le grand leader en matière d'entrepôts de données « *Teradata* » propose une solution de stockage qui offre la possibilité d'ajouter une capacité de stockage à très moindre coût et de rentabiliser la performance afin de répondre aux besoins des utilisateurs commerciaux en entreprise [2].

(2) Les requêtes définies sur les entrepôts de données sont très complexes : Toute jointure doit passer par la table des faits, ce qui rend le coût d'exécution de ces

requêtes très important. Sans technique d'optimisation, leur exécution peut prendre des heures, voire des jours.

(3) La sélection d'une structure d'optimisation (index, vue matérialisée, fragmentation) génère un nouveau problème au niveau de l'entrepôt de données qui se traduit par l'explosion de l'espace de recherche ou l'espace de solution des techniques d'optimisation.
(4) La sélection d'une seule technique d'optimisation est problème NP complet [4], et elle n'est pas toujours suffisante pour optimiser toute la charge de requêtes. Pour satisfaire le maximum de requêtes, il est indispensable d'opter pour une sélection multiple de structures ayant une forte similarité, ce qui est engendre une complexité énorme. Dans ce cas elle est plus complexe que la sélection isolée vu la complexité de son espace de recherche englobant ceux des structures concernées.
(5) L'explosion de l'espace de recherche rendre l'opération de sélection très complexe et le temps de réponse de l'algorithme de sélection de plus en plus long. Il faut donc prévoir une stratégie afin de permettre de réduire la complexité de problème d'optimisation.

Afin de palier ce problème il est intéressant de mettre en œuvre une sélection combinée des techniques d'optimisation pour réduire la complexité des requêtes.

Dans les sections qui suivent, nous allons introduire les techniques d'optimisation des requêtes dans les entrepôts de données. Nous allons définir leurs concepts, leurs complexités et leurs espaces de recherche.

I.2 Techniques d'optimisation des entrepôts de données

La réduction de la complexité d'exécution des requêtes et du temps et coût d'exécution repose sur une politique d'optimisation d'un ED qui dépend de la technique d'optimisation adéquate, la nature de sélection nécessaire et l'algorithme de sélection.

Il existe deux classes de structures d'optimisation : les structures redondantes et les structures non redondantes. La première famille impose une redondance de données qui sont les index et les vues matérialisées. Ces structures redondantes partagent la même ressource de stockage et impliquent une surcharge de maintenance pour le système, mais cette redondance peut servir aussi pour accélérer les accès. La deuxième famille est l'ensemble des techniques non redondantes du fait qu'elles ne répliquent pas de données et réorganisent leur représentation physique. On trouve comme structures non redondantes : la fragmentation des données, le traitement parallèle et la manière d'implémenter la jointure.

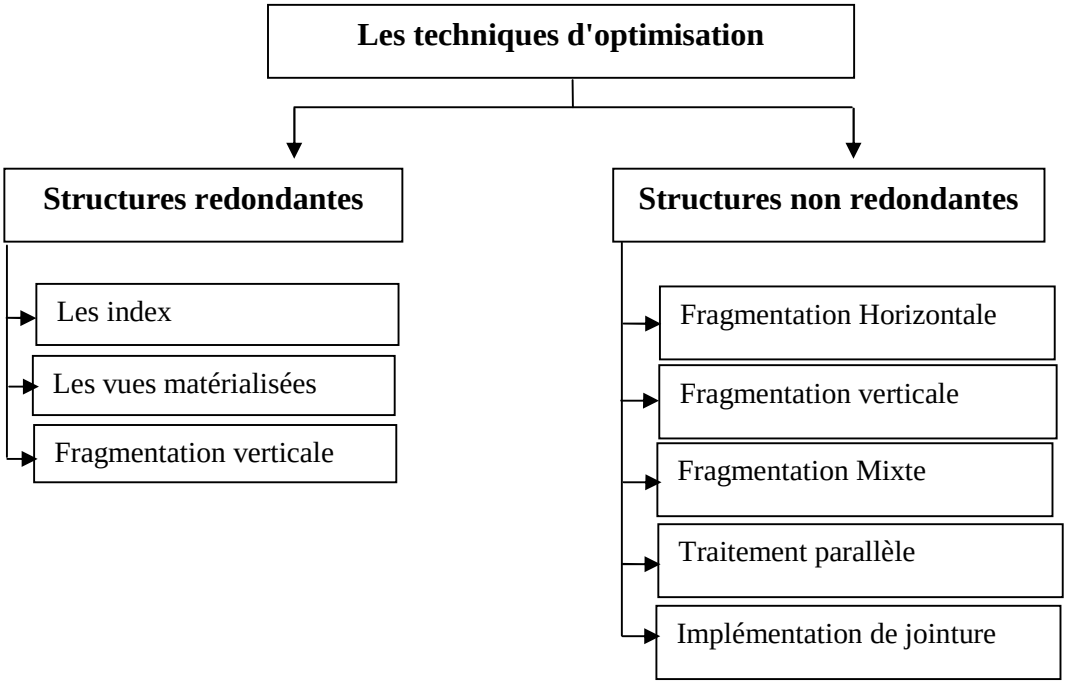


Figure I-5:Classification des techniques d'optimisation

Nous présentons dans ce qui suit, deux techniques d'optimisation rentrant dans le cadre de ce travail : l'indexation et la fragmentation horizontale.

I.2 .1 Techniques d'optimisation redondantes

I.2 .1.1 ***Les index :***

L'indexation est l'une des techniques d'optimisation redondantes qui minimise le volume de données à exploiter dans les calculs. La création d'un index permet d'améliorer considérablement le temps d'accès aux données en créant des chemins d'accès directs. Deux types d'index sont disponibles : les mono-tables (B-tree, index binaire, projection, etc.) et les multi-tables (index de jointure). Les index mono-table sont des index définis sur un ou plusieurs attributs de la même table, et les index multi-tables sont des index définis sur plusieurs tables. Nous présentons dans les sections suivantes les principales techniques d'indexation utilisées dans les SGBD relationnels et les entrepôts de données.

- ✓ **Techniques d'indexation**

Plusieurs techniques d'indexation peuvent implémenter sur les SGBD commerciaux dont les plus connues sont : les index B-arbre, les index binaires, les index de jointure, etc. Un administrateur de base de données a besoin d'information sur la façon dont les données sont stockées pour optimiser l'exécution des requêtes.

1. ***Les index simples :*** Regroupent les index B-arbres et les index de projection. L'index B-arbre est défini sur un attribut d'une table. Les nœuds de l'arbre renferment

les valeurs ordonnées de l'attribut et les feuilles contiennent des clés vers les tuples de données stockées sur le disque.

L'index B-arbre est l'index par défaut pour la plupart des SGBD commerciaux. Cet index est organisé sous forme d'index hiérarchique équilibré (toutes les feuilles sont au même niveau). Chaque nœud d'un niveau pointe vers le niveau inférieur. Les nœuds feuilles (niveau le plus bas) contiennent les entrées d'index ainsi qu'un pointeur vers l'emplacement physique de l'enregistrement correspondant (généralement un identifiant physique, ROWID). Figure I.6 représente un exemple d'index B-arbre.

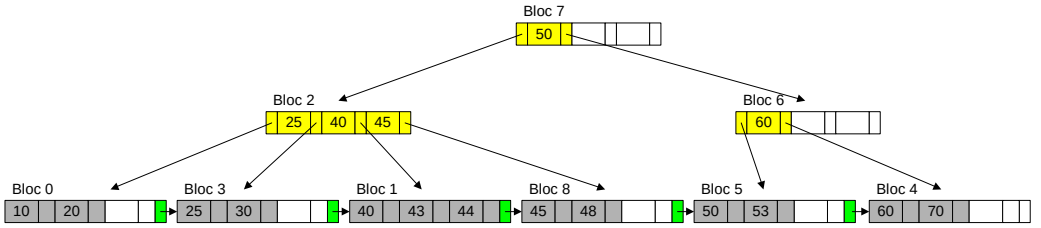


Figure I-6:Index B-arbre

2. ***Les index de jointure :***

Etant toujours présente dans les requêtes OLAP et très coûteuse en temps d'exécution, la jointure entre deux tables peut être précalculée et stockée dans un index proposé par Valduriez [18] qui matérialise les liens existant entre deux tables en utilisant une table à deux colonnes chacune représentant l'identifiant d'une table (Voir Figure I-7).

Notons que la taille de l'index de jointure dépend de la sélectivité de la jointure. Si la jointure est très sélective alors la taille de l'index est très petite.

Table Client

RID	Nom	Age	Sexe	Ville
200	Abdelkader	26	M	Tiaret
202	Kamel	30	M	Alger
300	Sabrina	22	F	Oran
100	Ali	27	M	Alger
212	Amina	19	F	Tiaret
310	Nacer	50	M	Tiaret

Index de jointure

RID	RID
1	200
2	200
3	200
4	202
….	…

Table Vente

RID	CID	PID	TID	Montant
1	200	260	11	300
2	200	300	14	260
3	200	220	16	1000
4	202	220	66	700
5	100	190	22	800
6	100	190	25	150
7	212	101	30	500
8	310	100	11	650
9	300	101	66	140
10	300	101	66	800

Figure I-7:Index de jointure

3. *Index de jointure en étoile :*

L'index de jointure en étoile est adapté aux requêtes définies sur les entrepôts de données modélisés en étoile. Il permet de stocker le résultat de la jointure entre la table des faits et les tables de dimensions (Voir Figure I-8). Il est dit complet s'il regroupe toutes les tables de dimension sinon partiel. Il accélère l'opération de jointure mais exige beaucoup d'espace de stockage et un coût de maintenance très élevé.

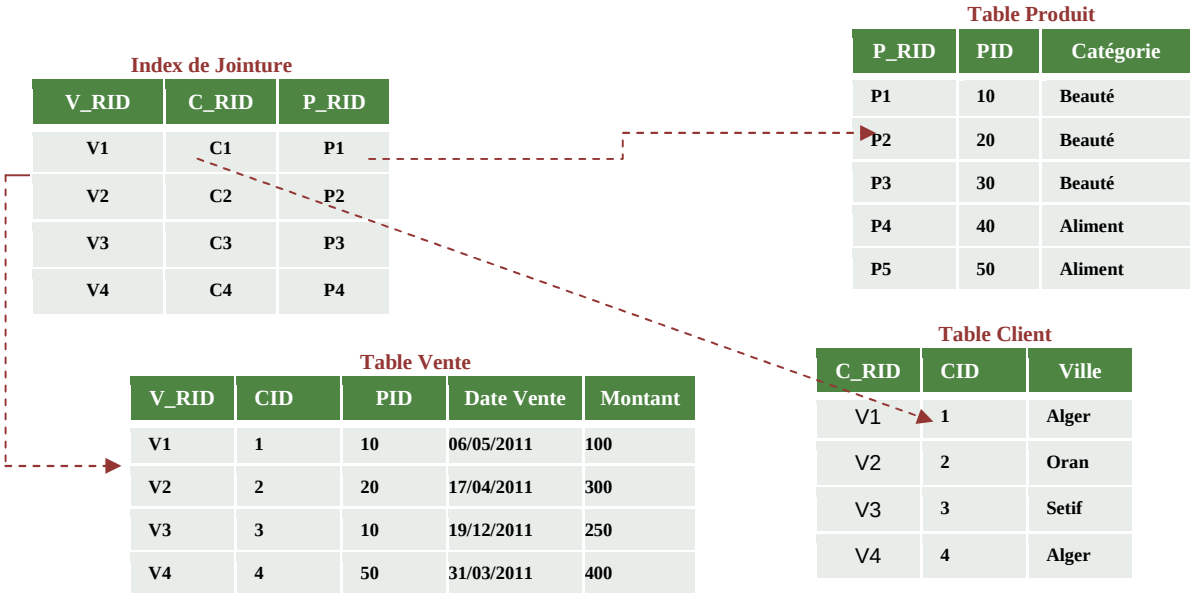


Figure I-8:Index de jointure en étoile

4. *Les index bitmap :*

Les index Bitmap (IB) stockent les données de chaque dimension (ou d'attribut) séparément, et permettent l'accès rapide aux différentes dimensions qui sont nécessaires pour répondre à une requête.

Par exemple, un index bitmap simple (IBS) sur un attribut Genre, avec le domaine {masculin, féminin}, représenté sous forme de deux vecteurs bitmap noté B*m*, B*f*. Pour B*m*, le bit est un ensemble de 1 si le correspondant de tuple à la valeur « masculin» pour l'attribut sexe, sinon le bit est un ensemble de 0. De même pour le B*f*, le bit est un ensemble de 1 si le tuple associé à la valeur « féminin » pour l'attribut Genre, sinon le bit est un ensemble des zéros, Pour un autre attribut, par exemple AnnéesTravail ayant des valeurs de 1-5, il y a un vecteur bitmap correspondant à chacune des cinq valeurs, noté B1-B5. Tuples associés à la valeur « 5» pour l'attribut AnnéesTravail sont représentés par des vecteurs binaires « B1=0, B2=0, B3=0, B4=0, B5=1 ».

	Genre		AnnéesTravail				
Tuple	*Bf*	*Bm*	B1	B2	B3	B4	B5
1	**1**	0	0	**1**	0	0	0
2	**1**	0	**1**	0	0	0	**1**
3	**1**	0	0	0	0	**1**	0
4	0	**1**	0	**1**	0	0	0
5	**1**	0	0	0	**1**	0	0

Figure I-9:Exemple de deux index bitmap simple

La figure I-9 montre un IJB sur le Genre et AnnéesTravail pour 5 tuples avec deux vecteurs bitmap pour l'attribut Genre et cinq pour l'attribut AnnéesTravail.

Pour illustrer le traitement de la requête avec les d'index Bitmap, considérons une requête SQL simple de recherche tous les tuples correspondant aux employés féminins ayant exactement 5 ans d'expérience professionnelle :

SELECT *
FROM Employee
WHERE Genre = "féminin" *AND AnnéesTravail = "5"*

Afin d'évaluer cette requête on utilise l'exemple SBIs, un optimiseur de requête prend le bitmap pour le «*Genre* = féminin » et « AnnéesTravail = 5 » et effectue une opération ET Logique. Le tableau donne le résultat, où le tuple 2 est le seul tuple dans la réponse de requête.

Résultat de sélection

Tuple	*Bf*	*Bm*	Résultat
1	**1**	0	0
2	**1**	**1**	**1**
3	**1**	0	0
4	0	0	0
5	**1**	0	0

Figure I-10:Résultat de requête avec l'utilisation de l'Index Bitmap

5. **Index de jointure binaire *(IJB) :***

Dans les entrepôts de données les requêtes impliquent des opérations de jointure, construire un index de jointure sur les colonnes de jointure améliore le temps de d'exécution de la requête. L'index de jointure binaire (IJB) est construit en créant n'importe quel type d'index Bitmap sur une table T basé sur une colonne A de la table S, où A est généralement l'attribut joindre. Il est habituellement utilisé avec de données de cardinalité réduite. Supposons, un attribut 'A' ayant n valeurs distinctes, et appartenant à une table de dimension D, et une table des faits F composée de m instances, la construction d'un index de jointure binaire IJB défini sur l'attribut A se fait de la manière suivante :

- Créer n vecteurs composés chacun de m entrées ;

Le i'ème bit du vecteur correspondant à une valeur *vk* 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 *vk*. Il est mis à 0 dans le cas contraire.

Client

ID	Sexe
1	F
2	F
3	F
4	M
5	F
6	F
7	F
8	M
9	M

Vente

Tuple	Id_client	IJB sur Genre	
		M	F
T1	4	**1**	0
T2	1	0	**1**
T3	3	0	**1**
T4	9	**1**	0
T5	6	0	**1**
T6	10	0	**1**
T7	3	0	**1**
T8	5	0	**1**
T9	1	0	**1**
T10	8	**1**	0

Figure I-11:Exemple d'un Index de jointure binaire

Pour un entrepôt modélisé par un schéma en étoile, les index de jointure binaires sont souvent utilisés pour accélérer les requêtes de jointure en étoile connues par leur nombre d'opérations de jointure. 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 [1].

Exemple 6: Considérons la dimension « Client » et la table de fait « Vente » qui contient les ventes pour les clients. L'index de jointure binaire définis sur « Vente » par rapport à l'attribut « Genre » est spécifié par la Figure I.11.

La création des index de jointure binaire se fait par l'exécution de la requête de création (ORACLE) pour chaque attribut de dimension sur la table de fait :

```
CREATE BITMAP INDEX IJB
ON Vente (Client.Genre)
FROM Vente, Client
WHERE Vente.Id_C=Client.Id ;
```

La sélection des *IJBs* est un problème difficile vu le nombre important d'attributs candidats participant à la construction des index. Le PSI est connu *NP-Complet* [5].

La sélection d'une configuration d'*IJB* est généralement une tâche difficile comparée à d'autres types d'index. La taille d'un IJB est proportionnelle à sa cardinalité. Un attribut de forte cardinalité rend l'index volumineux, donc difficile à stocker et de le maintenir.
Soit A = {*A1, A2, ..., AK*} un ensemble d'attributs indexables candidats pour la sélection d'une configuration d'*IJB*. Chaque *IJB* dans cette configuration est constitué

d'un sous-ensemble d'attributs de A, par conséquent cette configuration constitue une partition de A en un ensemble :

1. Sélection d'un seul index de jointure : si on veut utiliser un seul *IJB*, le nombre de possibilité est donné par des groupes. Chaque groupe d'attributs représente un *IJB* potentiel. Nous pouvons considérer deux scenarii.
a. Sélection d'un seul index de jointure : si on veut utiliser un seul *IJB*, le nombre de possibilité est donné par [1] :

$$N = \binom{k}{1} + \binom{k}{2} + \cdots . + \binom{k}{k} = 2^K - 1$$

Si le nombre d'attributs indexables est égal à 5 (K = 5), alors le nombre d'index possible est égal à 31.
b. Sélection de plus d'un index de jointure : pour sélectionner plus d'un IJB, le nombre de possibilités est donné par :

$$N = \binom{2^K - 1}{1} + \binom{2^K - 1}{2} + \cdots . + \binom{2^K - 1}{2^K - 1} = 2^{2^K} - 1$$

Par exemple, si le nombre d'attributs indexables est égal à 5, alors $N = 2^{31} > 1,2 \times 10^9$.

Peu de travaux de sélection d'*IJB* qui ont été effectués et qui vont être présentés dans le chapitre suivant.

I.2 .2 Techniques d'optimisation non redondantes

I.2 .2.1 La fragmentation horizontale

La fragmentation horizontale consiste à diviser une relation R en sous-ensembles de n-uplets appelés fragments horizontaux résultant de l'application de l'opération de restriction. De plus, la reconstruction de la relation R à partir de ces fragments horizontaux est obtenue par l'opération d'union de ces fragments. Cette stratégie est utile pour les situations où les utilisateurs doivent pouvoir accéder aux différentes lignes différentes. Dans les bases de données la fragmentation horizontale se scinde en deux versions.

a)La fragmentation primaire : s'effectue grâce à des prédicats de sélection définis sur la relation. Elle favorise le traitement des requêtes de restriction portant sur les attributs utilisés dans le processus de la fragmentation.

Exemple7 : Soit la table Client (Client_id, Nom, Ville) partitionnée comme suit :

- Client1: $\sigma_{\text{Ville='Alger'}}$(Client)
- Client2: $\sigma_{\text{Ville='Paris'}}$(Client)

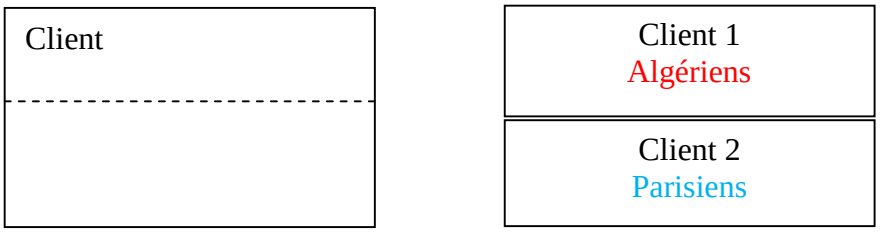


Figure I-12: Exemple d'une fragmentation horizontale primaire

b) La fragmentation horizontale dérivée

La FH dérivée permet de fragmenter une relation suivant la fragmentation horizontale primaire d'une seconde relation, à condition de l'existence d'une clé étrangère (relation père-fils) qui pointe de la première table (relation membre) à la seconde table (relation propriétaire). Le calcul des fragments dérivés se fait par la semi-jointure des fragments propriétaire avec la relation membre.

Exemple 8 : Soit la table Ventes (Client_id, Produit_id, Date, Quantité)

- Ventes1=Ventes ⋉Client1
- Ventes2=Ventes ⋉Client2

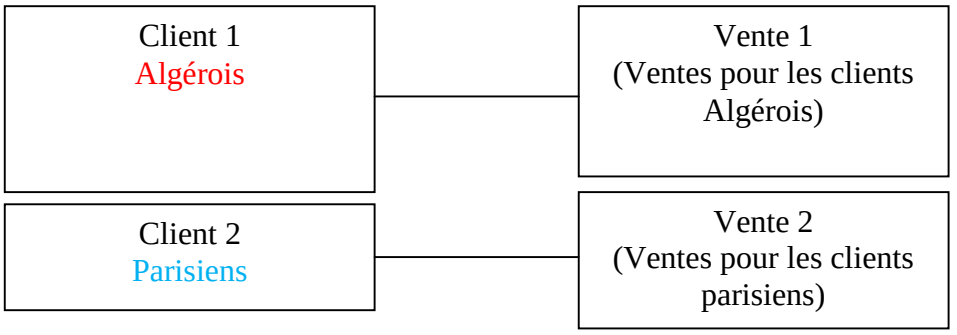


Figure I-13: Exemple d'une fragmentation horizontale dérivée

A partir de ces deux définitions, nous constatons que la fragmentation primaire pourrait accélérer les opérations de sélection tandis que la fragmentation dérivée accélérerait les opérations de jointure.
La complexité du problème de sélection d'un schéma de FH à été étudiée d'une façon détaillée dans [7] pour un schéma en étoile composé d'une table des faits F et d tables de dimension, *D1, D2,.., Dd*. L'explosion du nombre de fragments $N = \prod_{i=1}^{k} M_i$ où

Mi : nombre de fragments de la table de dimension Di et k : nombre de tables de dimension fragmentées

Le nombre des schémas de FH est exponentiel par rapport au nombre de sous-domaines utilisés pour fragmenter l'entrepôt. Par conséquent, une énumération exhaustive de tous les schémas de fragmentation est quasi impossible pour des valeurs élevées du nombre de sous domaines [7].

c) Scénarii de fragmentation horizontale d'un entrepôt de données

Dans le contexte d'entrepôt de données, les opérations les plus coûteuses en temps et en ressources sont les jointures en étoile entre table de fait volumineuse et plusieurs tables de dimensions. Il a été montré dans la littérature que la meilleure façon de fragmenter horizontalement d'un ED consiste à fragmenter par la FH primaire les tables de dimension ensuite utiliser leurs schémas de fragmentation pour fragmenter la table des faits par la FH dérivée. Avec cette procédure de fragmentation, les

opérations de sélection définies sur les tables de dimension ainsi que les jointures entre ces dernières et la table des faits seront optimisées [7].

Exemple 9 : Soit un entrepôt de données avec une dimension « Client » et une table de faits « Vente ». La table « Client » est fragmentée avec une FH primaire sur l'attribut « Genre » en clients féminins et clients masculins. La fragmentation est réalisée par opération de restriction comme suit :

-Client1: σGenre='M'(Client)

-Client2: σGenre='F'(Client)

Le résultat de la FH primaire est illustré sur la figure I-14 :

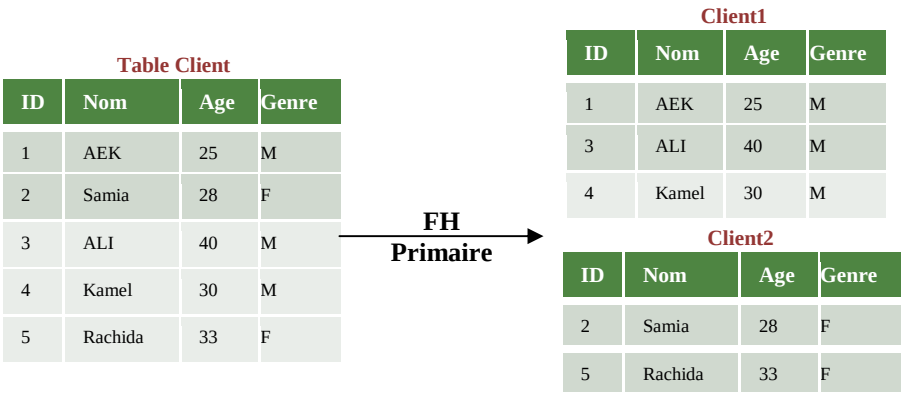


Figure I-14 : Exemple de fragmentation horizontale primaire

La création de la fragmentation horizontale sur la table de dimension Client pour attribut Genre se fait par le script de création (ORACLE) suivant:

```
CREATE TABLE CLIENT (Id number(6), Nom varchar(30), Age number(3),
Genre char(1),
PARTITION BY RANGE(Genre)
PARTITION Client1 VALUES LESS THAN (M) TABLESPACE TBS1,
PARTITION Client2 VALUES LESS THAN (F) TABLE SPACE TBS2,
```

La table « Vente », contenant la clé étrangère vers Client « Id_C », est fragmentée par une FH dérivée suivant la FH primaire de « Client »; cette fragmentation est obtenue par semi- jointure comme suit :

- Vente1= Vente ⋉Client1
- Vente2= Vente ⋉Client2

Les résultats de FH dérivée sont illustrés dans la figure I-15:

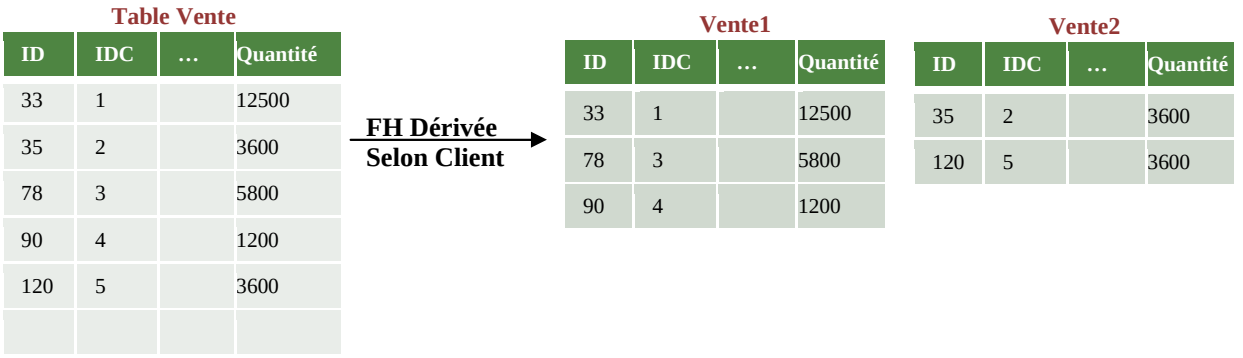


Figure I-15 : Exemple de fragmentation horizontale dérivée

La requête qui permet de créer la table Vente fragmentée est la suivante :

CREATE TABLE VENTES (IdC number(6), Date DATE , Montant Number(8,2)
CONSTRAINT Client_fk FOREIGN KEY (IdC) REFERENCES Client(Id))
PARTITION BY REFERENCE(Client_fk)

Ce scénario de FH permet d'accélérer les requêtes contenant le prédicat de jointure

« Client.Id=Vente.Id_C » entre la table de fait Vente et la dimension Client avec prédicat de sélection sur Client « Client.Genre = valeur », où valeur = F ou M.

I.2.3 Similarité entre la fragmentation horizontale dérivée (FHD) et IJB

Dans le contexte d'entrepôt de données, les index de jointure binaires et la fragmentation horizontale sont deux techniques d'optimisation similaires. Elles permettent d'accélérer l'exécution de requêtes contenant des opérations de jointures en étoile, entre table de faits volumineuse et plusieurs dimensions, et opérations de sélection sur les dimensions. Nous allons montrer cette similarité à travers l'exemple suivant :

Exemple 10: Soit un ED avec les dimensions « Client », « Produit » et la table de fait « Vente ».

Nous considérons deux cas d'optimisation que nous résumons sur la figure I-16:

1. Optimisation par FH dérivée de « Vente » suivant la FH primaire de « Client » avec l'attribut « Genre »

2. Optimisation par IJB définie sur « Vente » suivant le même attribut « Genre ».

Considérons la requête suivante qui permet de calculer, pour chaque client féminin, la quantité totale de produits vendus.

SELECT Nom_Client, Sum(Quantité)
FROM Vente V, Client C
WHERE V.Id_C=C.Id AND C.Genre='feminin'
GROUP BY Nom_Client

Les deux cas d'optimisation permettent d'optimiser la requête comme suit :

- Cas avec FH dérivée, seul le fragment Vente2 (ventes des clients féminin) sera chargé. Sur ce fragment, les deux opérations de regroupement et d'agrégation sont effectuées.

- Cas avec IJB. Le bitmap qui vérifie « Genre='féminin' » est chargé. Les lignes contenant des uns seront choisies pour permettre de charger les tuples faits des ventes des clients féminins. Sur ces tuples chargés, le regroupement et agrégation sont réalisés.

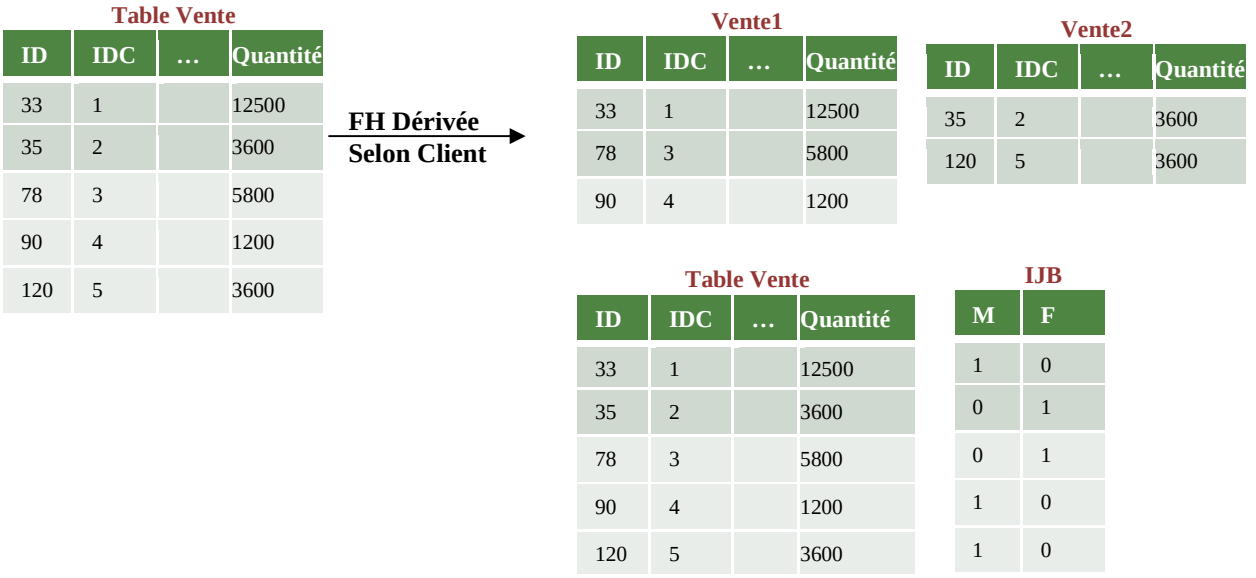


Figure I-16 : FH dérivée et IJB sur la table Vente

Nous signalons que les tâches d'indexation et de fragmentation sont effectuées par l'administrateur de l'entrepôt qui a aussi d'autres tâches. Nous présentons dans ce qui suit, le rôle qu'il peut jouer pour effectuer une bonne optimisation.

I.3 Rôle du concepteur de la stratégie de sélection des techniques d'optimisation

Aujourd'hui la tâche de l'administrateur devient très difficile face à l'explosion des données ces dernières années d'où la nécessité de développer des outils automatiques qui aident à l'administration de l'ED.

En présence de plusieurs techniques d'optimisation, avec un espace de recherche très grand, l'AED dispose de deux modes de sélection : la sélection isolée et la sélection combinée. Dans la sélection isolée, il choisit une seule technique qui pourra être redondante ou non redondante, cette sélection a été étudiée par exemple dans [5, 6, 7,8]. Mais il est souvent insuffisant pour une meilleure optimisation de l'entrepôt. La

sélection combinée permet de sélectionner plusieurs techniques à la fois. Elle est principalement motivée par les fortes similarités entre les techniques d'optimisation. Les travaux majeurs dans cette catégorie sont principalement concentrés sur la sélection des vues matérialisées et les index [7, 8,14]. La sélection multiple peut être effectuée de deux manières : séquentielle ou conjointe.

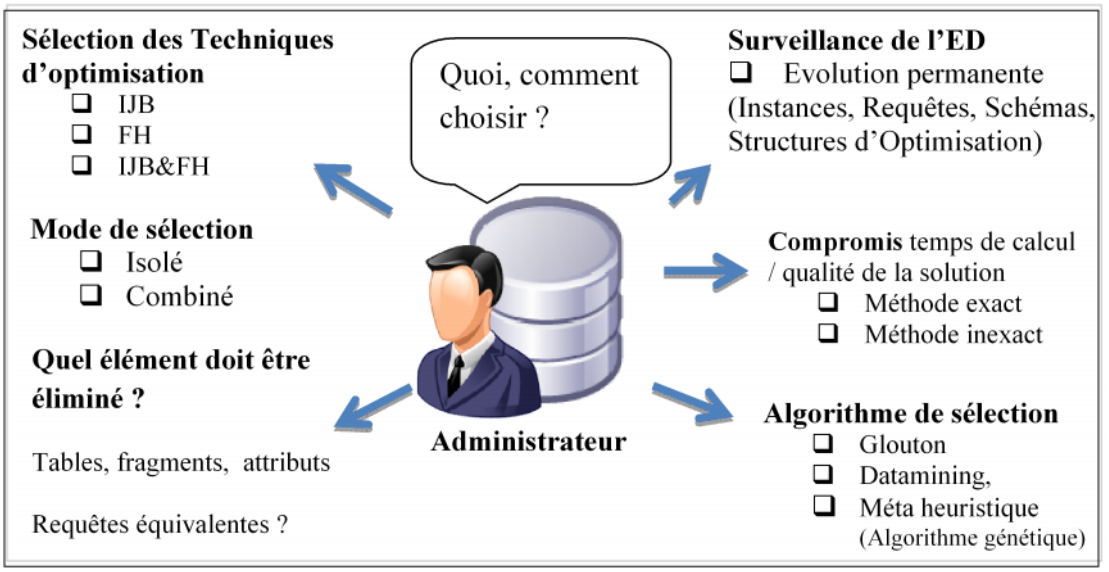


Figure I-17 : Choix effectués par administrateur

Pour mieux comprendre ces difficultés, considérons le scénario suivant dans lequel un administrateur a un ensemble de requêtes à optimiser. Cette tâche est complexe pour plusieurs raisons ;

- La complexité des requêtes, un schéma ED très grand engendre un nombre exponentiel des structures d'optimisations (index, vues, fragments).
- Présence de plusieurs techniques d'optimisation candidates.
    1. Quelles techniques d'optimisation dois-je choisir ?
    2. Quel est le mode de sélection des techniques d'optimisation (isolé, combiné)
- La technique d'optimisation est associée à un espace de recherche très grand.
    3. Quelles tables, fragments, attributs doivent être éliminés ?
- Deux techniques d'optimisation peuvent partager la même ressource, donc engendrent la notion de concurrence. Par exemple un attribut est choisi pour être utilisé soit par la technique d'index soit par les vues matérialisées.

    4. Un attribut est bénéfique pour quelle technique d'optimisation ?
    5. Comment surveiller les changements effectués durant le cycle de vie de l'entrepôt de données

L'objectif dans ce cas, l'optimisation des requêtes décisionnelles exécutées sur entrepôt de données modélisé en étoile. Une relation de coopération peut s'identifier

entre ces techniques d'optimisation pour optimiser telle ou telle requête. Une situation de concurrence sur des ressources communes peut aussi être constatée.
Derrière ces stratégies, il existe des outils implémentés dans les SGBD commerciaux. Le but de ces outils est de remplacer l'administrateur en effectuant automatiquement certaines de ses tâches d'administration.

I.3.1 Formalisation générale de problème de sélection

Etant donné une charge de requêtes *Q* (récupérée généralement à partir du journal de transaction du SGBD ou spécifiée par l'administrateur) et une ou plusieurs contraintes d'optimisations *B* (espace de stockage, coût de maintenance, temps d'exécution). Il faut trouver une configuration de structures physiques en un temps raisonnable, qui, une fois implémentées, permettent de réduire le coût d'exécution des requêtes *Q* sans violer la contrainte *B*

Le problème de sélection d'une technique d'optimisation TO peut être formalisé comme suit :

Entrées :

1. Un entrepôt de données ayant une fait F et n dimensions D = {D1…Dn},
2. Un ensemble de requêtes Q
3. Une ensemble des contraintes *B* liées au TO (espace disque disponible, coût de maintenance).
4. Une fonction qui renvoie pour chaque requête et chaque ensemble de structure d'optimisation, le coût de l'évaluation de la requête en utilisant ces structures.

Sortie: Un sous-ensemble de structure d'optimisation qui :

1. Respectent l'ensemble de contraintes liées au TO
2. Minimisent le coût d'évaluation de l'ensemble requête donnée.

I.3.2 Type d'algorithmes de sélection des structures d'optimisations

Etant donné une structure d'optimisation, plusieurs configurations sont possibles pour optimiser les requêtes. Pour les index par exemple, on peut créer autant d'index que d'attribut existants dans le schéma des tables. Puisque les index nécessitent un espace de stockage, on ne peut pas tout indexer. Il faut impérativement choisir les index qui permettent d'optimiser au mieux les traitements et requêtes exécutés. La sélection manuelle peut être effectuée par l'administrateur qui choisit un certain nombre de solution. Cette sélection dépend largement de l'expérience de l'administrateur et nécessite un temps d'administration non négligeable.

Afin d'automatiser la sélection des techniques d'optimisation, plusieurs algorithmes de sélection ont été proposés pour choisir une configuration de structures optimale. Nous pouvons les classer en plusieurs approches : approches basées gloutons, approches basées heuristiques, approches basées méta-heuristiques, approches basées datamining et récemment un travail se base sur les Systèmes Multi-agents (SMA)

[52]. Ces approches ont été souvent adoptées pour la sélection d'index [20, 21,22] ou pour la fragmentation [7,15, 24,25].

La figure I-18 montre une classification des différentes méthodes de sélections.

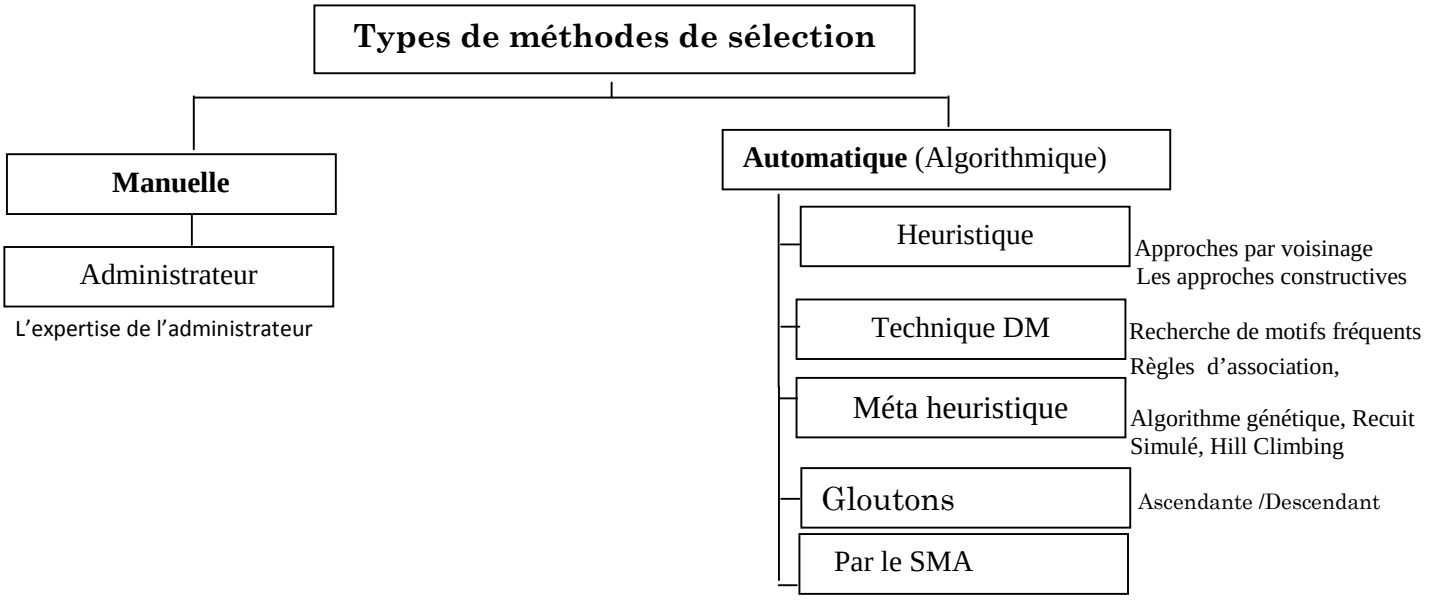


Figure I-18 : Classification des méthodes de sélection

Nous nous sommes intéressés dans le chapitre III de ce mémoire à une approche de sélection de schéma d'indexation et de fragmentation basée sur une architecture Multi-agent. A notre connaissance, un seul travail a étudié l'optimisation par la FH par le paradigme SMA dans le contexte des entrepôts de données distribués [52].

Nous trouvons nécessaire de définir les concepts relatifs au Système Multi-agent dans les sections suivantes.

I.5 Le paradigme Multi-agents

Nous présentons, dans ce qui suit, la notion d'agents, les Systèmes Multi-agents, leurs caractéristiques et leurs méthodologies de conception

I.5.1 Les Agents

Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents [23].

I.5.2 Caractéristiques des agents

Afin de mieux saisir la notion d'agent, nous pouvons relever quelques propriétés qui peuvent être attribuées aux agents [42] :

- Adaptabilité : capacité à apprendre et à s'améliorer avec l'expérience ;
- Autonomie : capacité d'agir seul, pour atteindre des buts ;
- Comportement collaboratif : capacité à travailler avec d'autres agents pour un objectif commun ;
- Capacité inférentielle : capacité d'agir avec des spécifications abstraites des tâches ;
- Capacité de communication au niveau des connaissances : capacité de communiquer avec les autres agents avec un langage comme celui des êtres humains ;
- Personnalité : capacité à manifester des attributs d'un caractère humain crédible ;
- Réactivité : capacité à détecter et réagir ;
- Continuité temporelle : persistance d'une identité et d'un état sur une longue période.

I.5.3 Les systèmes Multi-agents (SMA)

« On appelle Système Multi-agents (ou SMA), un système composé des éléments suivants :

- un environnement E, c'est-à-dire un espace disposant généralement d'une métrique ;

- un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents;

- un ensemble A d'agents, qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système ;

- un ensemble de relations R qui unissent des objets (et donc des agents) entre eux;

- un ensemble d'opérations Op, permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O;

- des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers [23]

I.5.4 Domaines d'application :

Les systèmes Multi-agents sont à l'intersection de plusieurs domaines scientifiques : informatique répartie et génie logiciel, intelligence artificielle, vie artificielle. Ils s'inspirent également d'études issues d'autres disciplines connexes notamment la sociologie, la psychologie sociale, les sciences cognitives et bien d'autres [50].

Les SMA présentent des avantages potentiels dans la résolution des problèmes liés à des systèmes distribués et complexes [50]. En effet, il est raisonnable de traiter ces derniers par la décomposition modulaire et fonctionnelle, pour que des « agents »

soient spécialisés dans la résolution d'un aspect particulier du problème. Les SMA permettent de réduire la complexité de la résolution d'un problème en divisant le savoir nécessaire en sous-ensembles, en associant un agent intelligent indépendant à chacun de ces sous-ensembles et en coordonnant l'activité de ces agents [42]. On parle ainsi d'intelligence artificielle distribuée.

I.5.5 La communication :

La communication est à la base des interactions et des organisations sociales d'un système multi agents. Elle permet aux agents d'échanger des informations et des demandes de services. Les agents communiquent entre eux à l'aide d'un langage de communication et en utilisant des protocoles de communication. Elle permet aussi à un agent d'agir sur un autre agent en lui fournissant des informations qui auront pour conséquence la remise en question de son comportement ou encore en lui demandant de modifier son comportement.

1. Types de protocoles de communication :

La communication entre les agents est réalisée avec l'envoi des messages asynchrones suivant les protocoles de communication définis. Les connaissances et les mécanismes de traitement pour la résolution d'un problème sont distribués dans les agents. La communication s'établit point à point, par diffusion totale ou restreinte.

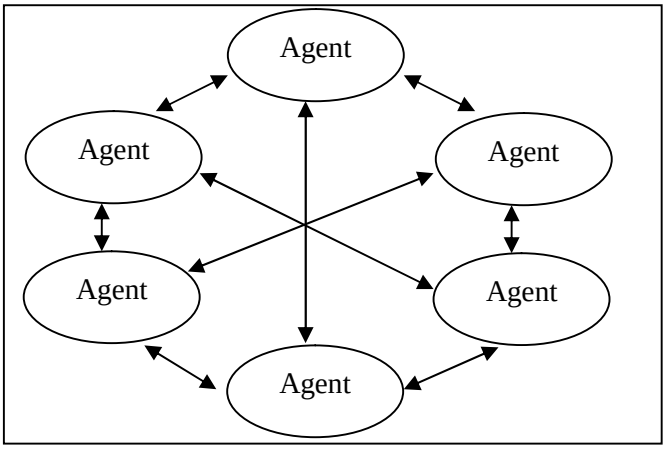


Figure I -19: La communication par messages [58]

2. Langages de communication entre agents

L'utilisation d'un langage commun implique que tous les agents comprennent son vocabulaire sous tous ses aspects :

- **la syntaxe,** qui précise le mode de structuration des symboles;

- **la pragmatique,** pour pouvoir interpréter les symboles;

On peut citer deux langages de communication entre agents : KQML[1] et FIPA-ACL[2]. Dans notre cas nous allons utiliser le langage de communication FIPA-ACL (*Foundation for Intelligent Physical Agents-Agent Communication Language*) basé sur la théorie des actes de langage FIPA-ACL est un langage qui s'appuie sur la définition de deux ensembles :

- Un ensemble d'actes de communication primitifs, auquel s'ajoutent les autres actes de communication pouvant être obtenus par la composition de ces actes de base;
- Un ensemble de messages prédéfinis tel que tous les agents peuvent comprendre.

FIPA-ACL possède 22 actes communicatifs groupés selon leurs fonctionnalités.

I.5.6Les méthodologies de conception des SMA :

Suite au développement des systèmes multi-agents, de nombreuses méthodologies sont apparues, ces méthodologies couvre les différentes étapes du processus de développement de SMA : analyse, développement, implémentation, déploiement [51]. Parmi eux, on peut citer : la méthodologie Voyelles, la méthodologie GAIA et la méthodologie Aalaadin [51].

Nous présentons brièvement la méthodologie Voyelles puisqu'elle sera utilisée pour la modélisation de notre solution (voir chapitre III).

La méthodologie Voyelles « AEIO » :

La méthodologie Voyelles repose sur quatre concepts, identifiés par les quatre voyelles : **A** pour Agents, **E** pour Environnements, **I** pour Interactions et **O** pour Organisations. Chaque voyelle correspond à un ensemble de modèles qui devront être combinés pour aboutir à la réalisation du système multi-agents.

Les Agents, concernent les modèles (ou les architectures) utilisés pour la partie active de l'agent.

L'Environnements, sont les milieux dans lesquels sont plongés les agents, ils permettent de définir l'ensemble des capacités de perception et d'actions des agents

L'Interactions, concernent les infrastructures, les langages et les protocoles d'interactions entre agents, depuis de simples interactions physiques à des interactions langagières par actes de langage.

L'Organisations, structurent les agents en groupes, hiérarchies, relations, etc.

La méthodologie « AEIO » se décompose en trois phases qui sont :

- **La phase d'analyse**, qui permet d'identifier et de décomposer le problème en

[1]www.fipa.org
[2]www.cs.umbc.edu/kqml/papers

quatre composantes fondamentales : Agents, Environnement, Interactions et Organisation, pour dégager l'architecture générale du système.

- **La phase de conception**, qui permet de choisir les modèles opérationnels des composantes pour aboutir à la spécification du fonctionnement global du système exprimé sous forme de diagrammes de comportement.
- **La phase de programmation** : (ou implémentation) consiste en l'instanciation des modèles, en utilisant des platesformes et des langages choisis.

I.5.7 Caractéristiques des Systèmes Multi-Agents

Le SMA repose sur deux concepts: **modularité**: pour réduire la complexité ; **vitesse**: due au parallélisme. L'idée est de décomposer le problème sous forme de sous problèmes, chaque sous problème est résolu par une entité autonome d'une manière distribuée; de plus, les algorithmes proposés dans littérature peuvent implémenter d'une manière distribuée dans l'architecture SMA.

L'objectif d'une exécution parallèle est de réduire le temps de réponse, la figure I -20 montre l'approche Top down (haut vers le bas), les paramètres (P1…Pn) sont affectés aux agents individuels (A1…An) et l'environnement. Les paramètres décrivent l'interaction entres les agents et leur environnement qui aboutit au niveau Macro à la résolution d'un problème complexe.

Il y a un but initial où une tâche initiale est décomposée en sous-buts ou sous-tâches et qui est répartie entre plusieurs agents (décomposition de problème "top-down"). Cette décomposition est considérée comme une façon d'élagage c.-à-d. réduire la complexité des problèmes.

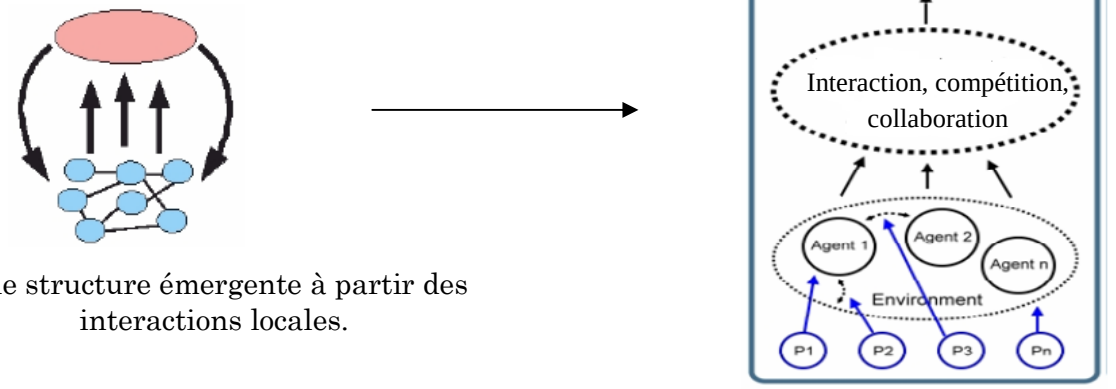


Figure I -20: l'approche Top Down

Le détail de la mise en œuvre des étapes réalisées par les Système Multi-agents sera l'objet du chapitre III.

Conclusion

Nous avons présenté dans ce chapitre, le concept général d'un entrepôt de données, son architecture, le modèle multidimensionnel des données (cube de données), les différents modèles de stockage (modèle ROLAP et MOLAP) et les types de données qui y sont présentes.

Nous avons également mis le point sur la conception physique des entrepôts de données en présentant et en classifiant les différentes techniques d'optimisations des requêtes définies dans le contexte d'entrepôts de données relationnels selon la nécessité ou pas d'un espace de stockage supplémentaire: les index, les vues matérialisées, et la fragmentation verticale comme techniques redondantes, ainsi que la fragmentation horizontale primaire et dérivée, et le traitement parallèle comme techniques non redondantes.

Nous avons donné une classification des techniques de sélections des structures d'optimisation en deux familles manuelle et automatique. Ainsi nous avons défini les concepts de bases des Systèmes Multi-agents.

Dans notre travail, nous nous sommes intéressés à deux techniques d'optimisation : l'IJB et la FH. Nous avons également mis le point sur les similarités entre ces deux techniques. Dans le chapitre qui suit, nous abordons le concept de sélection des techniques d'optimisation à savoir la sélection isolée et la sélection multiple. Une formalisation de problèmes de sélection des deux technique IJB et la FH chaque fera aussi l'objet de ce chapitre, ainsi que les principaux travaux traitant chaque problématique.

Chapitre II : Problèmes de sélection des techniques d'optimisation

Introduction

La taille des entrepôts de données peut varier de quelques giga-octets à plusieurs téraoctets dont la majorité des données est stockée dans un nombre restreint de grandes tables de faits. Le processus d'analyse est réalisé à l'aide de requêtes complexes comportant de multiples jointures et des opérations d'agrégation sur des tables volumineuses. Les décideurs souhaiteraient avoir de temps de réponse plus rapide à leurs requêtes, lorsqu'ils interrogent la base de données.

Les principales techniques utilisées pour répondre à ce besoin sont : les vues matérialisées, les index et la fragmentation. Le choix de l'AED d'une politique d'optimisation des requêtes et opérations exécutées sur les données devient difficile. Cette optimisation s'effectue durant la phase de conception physique et la solution doit être la plus optimale possible. Elle nécessite de choisir les techniques d'optimisation (index, fragmentation, vues, etc.) à implémenter. Plusieurs travaux se sont intéressés à la sélection des techniques d'optimisation dans le but de choisir les techniques les plus adéquates, qui permettent d'optimiser les requêtes et réduire le coût d'accès aux données.

Parmi les techniques adoptées dans le contexte d'entrepôts de données relationnels afin d'optimiser les performances, on peut citer, les index de jointure binaires *IJB* et la fragmentation horizontale *FH*. La première technique est une technique redondante qui nécessite un espace de stockage avec une surcharge de coût de maintenance et la seconde technique est non redondante; elle permet de répartir les tuples des tables en plusieurs sous ensembles de n-uplets pouvant être accédés indépendamment. Rappelons que les index et la fragmentation horizontale sont des structures qui améliorent l'accès aux données en optimisant les jointures en étoiles entre table de faits volumineuse et plusieurs dimensions.

L'une des difficultés les plus importantes dans la conception physique des entrepôts de données est la sélection combinée d'un ensemble d'index et de fragments, appelée configuration de IJB-FH, qui minimise le coût total d'exécution des requêtes, sous les deux contraintes: espace de stockage et nombre de fragments. Plusieurs travaux traitent la sélection des index et d'un schéma de fragmentation d'une manière isolée [20, 21,22]. Par contre peu de travaux se sont penchés sur la sélection des index de jointure binaire et de la fragmentation horizontale de manière combinée [1, 32,46] cette sélection est intéressante car elle permet de couvrir l'optimisation d'un

maximum des requêtes mais elle engendre un espace énorme ce qui est rend la tache de sélection difficile.

Dans ce chapitre, nous allons présenter un état de l'art sur le problème de sélection d'index et de schéma de fragmentation ; nous allons détailler les principaux algorithmes qui ont été proposés dans la littérature qui s'intéresse à la sélection des index de jointure binaires et la fragmentation horizontale afin d'améliorer les performances des entrepôts de données, ainsi qu'une comparaison des différentes approches.

II.1 **Espace de recherche des techniques d'optimisation de l'ED**

Nous allons présenter dans cette section les principaux travaux de sélection des index de jointures binaires et de la fragmentation horizontale basés sur le concept d'élagage. Certaines approches fournissent une solution quasi optimale (voir [12,15, 14]) par un algorithme glouton.

L'avantage de ces approches est qu'elles sont habituellement de complexité inférieure, en comparaison avec les méthodes exactes, et peuvent fournir un résultat optimale ou quasi optimale avec la croissance de la taille du problème. L'inconvénient de tels algorithmes est que la solution n'est pas garantie d'être optimale. Des travaux prouvent des limites sur l'erreur de la solution optimale et essayent d'améliorer ces limites.

Les travaux de sélection des techniques d'optimisation formalisent l'espace de recherche selon divers formalismes (treillis, arbre, graphe, etc.). La taille de ces structures peut varier de quelques dizaines de sommets à des millions de sommets. L'élagage est destiné pour résoudre les problèmes complexes, là où on ne peut pas explorer tout l'espace de recherche qui augmente exponentiellement avec le nombre d'instances. Cette technique consiste à réduire l'espace de recherche et implique la réduction de la complexité du problème.

Les travaux utilisent le treillis qui est parcouru d'une manière gloutonne, en exploitant un modèle de coût, pour sélectionner les meilleures structures. Le parcours du treillis peut être très coûteux en termes de temps de calcul, surtout si le cube de données OLAP est de forte dimensionnalité. Le passage à l'échelle est donc délicat. D'autres études modélisent les relations entre les structures d'optimisation à l'aide d'un graphe AND-OR [29]. Certains travaux assimilent le problème de sélection de vues matérialisées au problème du sac à dos ou à un problème d'optimisation en utilisant les algorithmes génétiques [13,32]. L'élagage de l'espace de recherche dans ces travaux se base sur des approches différentes comme le fusionnement, la classification et le regroupement.

Le tableau II-1 montre l'espace de recherche des techniques d'optimisation d'ED en fonction de nombre d'instance de problème.

Technique d'optimisation	**Nombre d'instance**	**Taille de l'espace de recherche**
IJB	n attributs	$2^n - 1$
Vues matérialisées	d dimensions	2^d
Fragmentation horizontale	m attributs de dimension n sous-domaines	$\frac{1}{e}\sum_{m=1}^{2n}\frac{m^n}{m!}$
Fragmentation verticale	n attributs	n^n
Implémentation de la jointure	n relations	$(2(n-1))!/\,(n-1)!$

Tableau II-1 *:* Espace de recherche des techniques d'optimisation d'ED

Les travaux qui traitent du problème de sélection de techniques d'optimisation sont différenciés selon :

- La méthode de construction des structures candidates : treillis, graphe, plan d'exécution des requêtes(MVPP).
- L'utilisation d'une présélection pour réduire la complexité du problème (élagage)
- Type de solution (optimale ou quasi optimale) fournie par l'algorithme de sélection.
- La méthode de construction de la configuration finale : algorithme glouton, algorithme génétique ou résolution du problème du sac à dos.

II.2 **Problème de sélection d'index**

Aujourd'hui l'index joue un rôle crucial dans le système de gestion de base de données (SGBD) pour déterminer la performance des requêtes. Le travail d'optimisation des performances effectué par l'AED se porte en grande partie sur la sélection d'index pertinent dans les entrepôts de données, qui présentent une volumétrie très importante et sont interrogés par des requêtes complexes. La présence d'un bon ensemble d'index peut considérablement réduire le temps de l'évaluation de la requête en réduisant la quantité de données qui doivent être accédées pour répondre à la requête. Naturellement, les index coûtent chères en termes d'espace et de temps de maintenance, ce qui explique l'impossibilité de créer tous les index candidats. Afin d'assurer une bonne exécution d'une charge de requêtes, il est important de choisir un ensemble d'index optimal.

La tâche du choix d'index, c.-à-d., choisit automatiquement un ensemble approprié d'index pour une base de données et une charge de requêtes, est motivé pour plusieurs raisons [10]. D'abord, puisque les schémas de base de données des applications réelles peuvent être grands (beaucoup de tables, de colonnes) et des index peuvent être définis sur un ordre d'une ou plusieurs colonnes. L'espace des index qui est approprié pour la charge de requêtes peut être très grand. En second lieu, les optimiseurs de requêtes d'aujourd'hui peuvent exploiter des index disponibles par des techniques avancées telles que l'intersection de deux index ou plus, etc. Ainsi, il devient important de tenir compte des interactions parmi des index. Enfin, le choix des index doit respecter certaines contraintes de ressource (Espace de stockage).

Le problème de sélection d'index a été assimilé au problème du sac à dos [12] [19]. Les index sont alors assimilés à des objets, l'espace de stockage des index aux poids des objets, le coût de la charge en présence d'un index au bénéfice de l'objet du sac à dos, et l'espace disque alloué au stockage de la configuration d'index finale à la taille du sac à dos.

II.2.1 Formulation de problème de sélection d'index (PSI)

Les entrées de problème de sélection d'index (PSI) peuvent être formulées par (D, F, Q, b) comme suit :

D: l'ensemble de tables des dimensions de l'entrepôt de données $\{D_1, D_2,..,D_n\}$

F : La table de faits

Q : L'ensemble de toutes les requêtes $\{Q_1, Q_2,.... Qn\}$ avec des poids $\{w_1, w_2,..., w_n\}$ qui représente leurs fréquences d'accès.

b : la contrainte de la ressource (espace de stockage etc.)

L'objectif du PSI est de sélectionner une configuration d'index C permettant de maximiser le gain apporté par l'utilisation de C tel que la taille de C ne dépasse pas b (Taille(C)<=b). La fonction de gain est calculée comme suit :

$$\text{Gain}(Q; C) = \sum_{q \in Q} \left(\text{coût}(q) - \text{coût}(q; C)\right) \qquad (4)$$

Le PSI est un problème d'optimisation NP-Complet [15]. Il peut être regardé comme un problème combinatoire d'optimisation parce que sa solution doit être choisie parmi un nombre fini de configurations possibles. Par conséquent, l'énumération de tous les sous-ensembles possibles d'index est une manière possible de résoudre l'PSI; mais cette méthode est impraticable dans la plupart des cas, car le temps de calcul se développe exponentiellement avec le nombre d'index candidat. Dans ce cas, on parle d'élagage qui intervient comme une étape à priori pour éliminer les index inutiles. La

figure II.1 illustre l'intervention de l'élagage dans l'approche générique de sélection d'index.

Les travaux proposent des algorithmes pour la sélection d'index dans les bases et entrepôts de données comportant généralement deux étapes : (1) la sélection des attributs candidats à l'indexation et (2) la construction d'une configuration d'index.

(1) **Génération d'index candidats à l'indexation** : En identifiant l'ensemble d'index candidat pour une charge de requêtes données, les premières approches de sélection d'index qui ont été proposées dans la littérature font appel à l'administrateur de l'entrepôt de données pour constituer manuellement l'ensemble d'index candidats [15, 16, 17, 19, 21,22]. Cette tactique a l'avantage de faire appel à l'expertise humaine, mais la tendance dans les travaux plus récents est de recourir à une approche automatique, plus susceptible d'être déployée à grande échelle. Pour cela, une analyse syntaxique des requêtes de la charge Q est réalisée afin d'identifier les attributs intéressants à indexer, qui sont principalement présents dans les clauses Where et Group by des requêtes SQL [10, 12, 18,23]. Dans la phase de détermination des attributs candidats, l'élagage peut se faire de deux manières : manuelle ou automatique. Notons que la taille de l'espace de recherche est exponentielle par rapport au nombre d'attributs candidats. Les approches proposées utilisent les techniques datamining ; des algorithmes heuristiques ; des règles à priori pour réduire l'espace de recherche d'index.

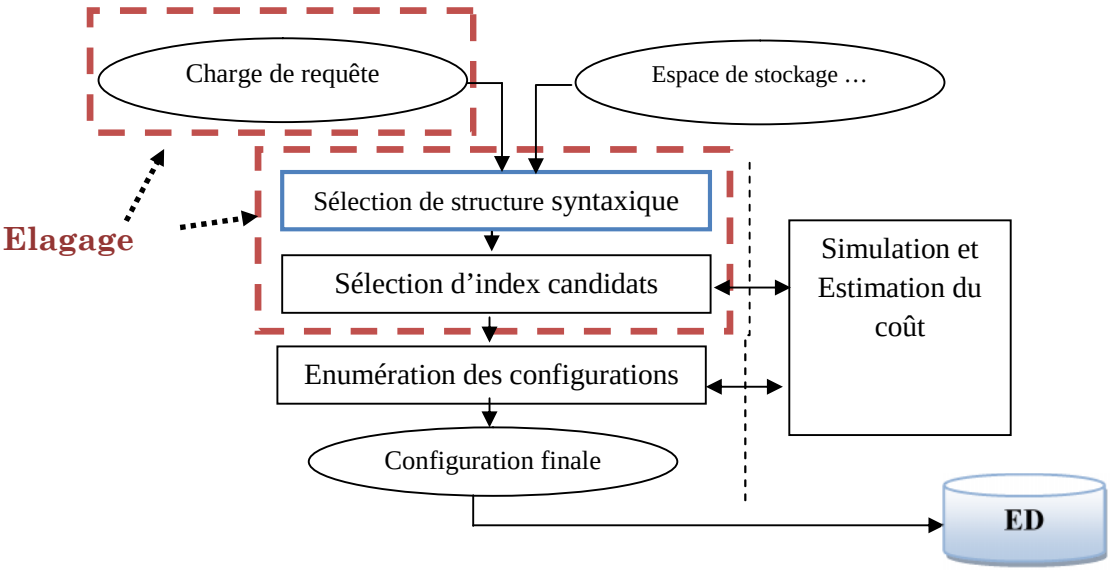


Figure II-1 : Approche générique de sélection d'index

(2) **La construction d'une configuration d'index**

La sélection de la configuration d'index à partir de l'ensemble des index candidats est basée sur un algorithme glouton. Des méthodes ascendantes partent d'une

configuration d'index vides et y ajoutent itérativement des index qui minimisent le coût de la charge jusqu' à ce qu'il ne diminue plus. Au contraire, les méthodes descendantes ont comme point de départ l'ensemble complet des index candidats. A chaque itération, des index sont élagués jusqu'à ce que le coût de la charge ne diminue plus finalement, certaines méthodes adoptent une approche mixte ascendante et descendante.

Dans la sélection d'un ensemble d'index candidats, un élagage peut intervenir pour réduire l'ensemble des index candidats à la sélection. Cette opération peut se faire de deux manières :

- Manuelle : en se basant sur son expérience, l'administrateur de la base de données peut manuellement sélectionner un ensemble d'index candidats.
- Automatique : par un algorithme pour mesurer s'il bénéfique l'attribut serait élu au processus de l'indexation.

II.2.2 Elagage manuel de l'espace de recherche d'index :

L'AED se base sur une connaissance à priori et son expertise pour déterminer les index candidats à la sélection. L'administrateur, avant d'indexer un attribut, doit considérer un certain nombre de points, par exemple :

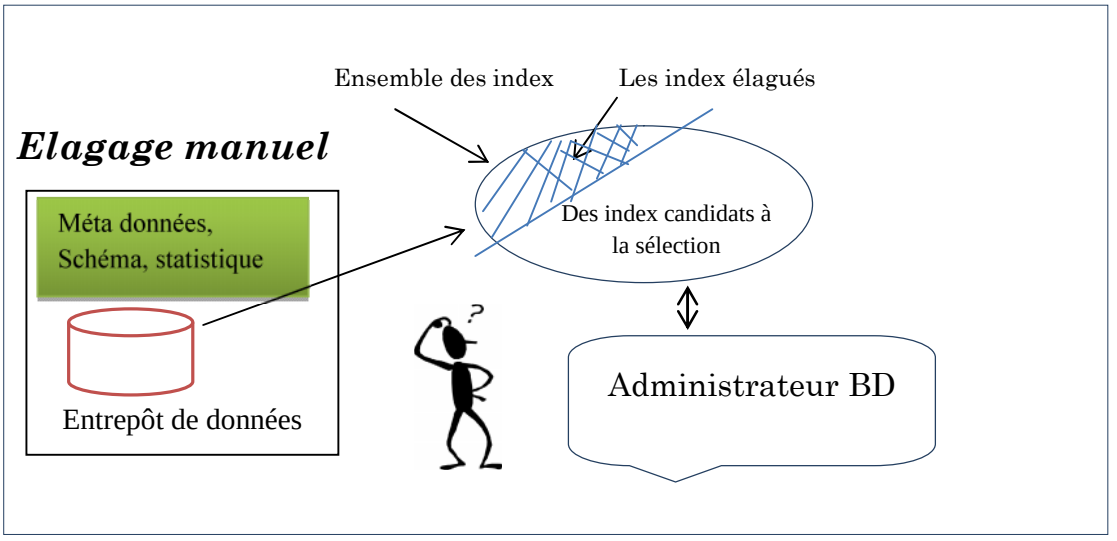


Figure II-2 : Elagage manuel

- Éviter d'indexer les clés étrangères, les attributs fréquemment modifiés car ceci entraînerait la mise à jour fréquente de l'index

-Pour créer un index composé de plusieurs attributs en même temps, l'AED doit considérer par exemple le point suivant et ceci afin de réduire l'accès aux données sur disque : considérer les attributs qui sont souvent sélectionnés ensemble dans les requêtes.

-Les index de jointure binaires sont définis en utilisant les attributs de sélection de faible cardinalité (nombre de valeurs distinctes), et les index encodés sur une hiérarchie d'attributs à forte cardinalité.

L'élagage manuel reste toujours limité dans les problèmes de grande échelle d'instance. Il nécessite un temps d'administration non négligeable. La solution est d'aller vers une approche automatique.

II.2.3 Modèles de coûts :

Les méthodes de sélection d'index exploitent deux types de modèles de coût : soit une fonction mathématique ad-hoc, soit l'optimiseur de requêtes du SGBD. L'utilisation d'une fonction ad-hoc a l'avantage de la rapidité, mais implique des hypothèses simplificatrices parfois trop irréalistes. Faire appel à l'optimiseur de requêtes est plus fiable, mais rend la sélection d'index dépendante d'un SGBD donné et s'avère plus gourmande en temps de calcul.

Plusieurs travaux antérieurs traitent de la sélection des index seulement (ISP) (Voir, par exemple, [3, 4, 5, 6]) pour l'entrepôt de données. Certains travaux commencent par élaguer l'espace de recherche des index afin de réduire la complexité du problème. Par la suite, un algorithme glouton sélectionne un ensemble optimum; cette sélection d'index fait appel à un modèle de coût pour élaguer les index moins avantageux.

Nous présentons dans ce qui suit les travaux sur la sélection d'index de jointure binaire dans le contexte de l'entrepôt de données.

II.3 Problème de sélection des index de jointure binaires

Peu de travaux ont traité la sélection isolée des IJB. Nous pouvons citer les travaux dans [6, 7, 8]). Nous présentons dans cette section une formalisation du problème de sélection des IJB, et les principaux travaux existants.

II.3.1 Travaux existant

II.3.1.1Travaux de Aouiche et al

Le travail d'Aouiche et al. [20], est parmi les travaux qui traitent le problème de sélection des *IJB* dans le contexte d'entrepôts de données modélisés par un schéma en étoile. L'approche proposée se base sur une technique de datamining (recherche des motifs fréquents) pour élaguer l'espace de recherche des index de jointure. Un algorithme glouton est utilisé pour la sélection d'une configuration d'index. Figure III.6 représente les principales étapes de l'approche proposée.

Pour comprendre l'approche utilisée, nous présentons la notion de motifs fréquents ainsi que l'algorithme CLOSE qui permet de rechercher ces motifs à partir d'un contexte d'extraction donné.

Définition 1 (Motif fréquent)

Soient $I = i_1, \ldots, i_m$ un ensemble de m items et $B = t_1, \ldots, t_n$ une base de données de n transactions. Chaque transaction est composée d'un sous-ensemble d'items $I' \subseteq I$. Un sous-ensemble I′ de taille k est appelé un k-itemset. Une transaction t_i contient un motif I′ si et seulement si $I' \subseteq t_i$. Le support d'un motif I′ est la proportion de transactions de B qui contiennent I′. Le support est donné par la formule suivante :

$$Support\ (I') = \frac{|\{t \in B,\ I' \subseteq t\}|}{|\{t \in B\}|}$$

Pour décider si un motif est fréquent ou non, l'utilisateur fixe un seuil minimum du support *minsup*. Si *Support*(I′) <*minsup* alors I′ est un motif fréquent, sinon il est non fréquent.

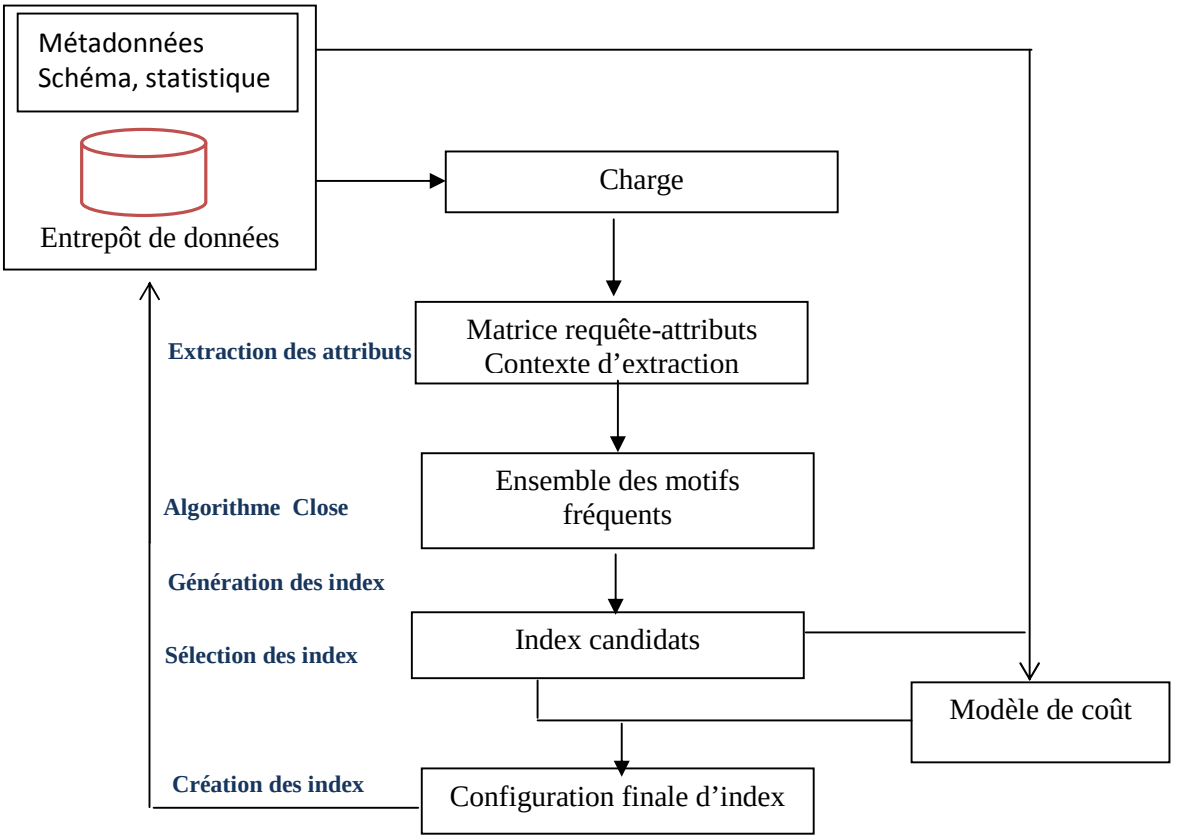


Figure II-3 : Architecture fonctionnelle de l'approche de Aouiche

Définition 2 (Motif fréquent fermé)

Un motif fermé est un ensemble maximal de motifs communs à un ensemble d'objets. Un motif i′ $\subseteq I$ tel que *support*(i) $\geq$ *minsup* est appelé motif fréquent fermé.

L'algorithme CLOSE pour la recherche des motifs fréquents :

Chapitre II : Problèmes de sélection des techniques d'optimisation

Plusieurs algorithmes ont été proposés pour la génération d'un ensemble de motifs fréquents. La génération des motifs fréquents est une tâche compliquée, car le nombre de motifs fréquents peut exploser lorsque le nombre de motifs initial est important. Pour réduire cette complexité, l'algorithme CLOSE génère les motifs fréquents fermés. L'ensemble des motifs fréquents peut être généré directement à partir de l'ensemble des motifs fréquents fermés. Dans le contexte de la sélection d'index, la génération des motifs fréquents fermés permet de générer moins d'index candidats et par conséquent réduire la complexité de la sélection des index.

L'approche de sélection des *IJB* proposée par les auteurs dans [20] comporte six étapes :(1) extraction de la charge de requêtes, (2) analyse de la charge, (3) construction d'un contexte de recherche des motifs fréquents, (4) application de l'algorithme CLOSE sur ce contexte, (5) construction de l'ensemble des index candidats et (6) construction de la configuration d'index finale.

L'approche commence par extraire la charge de requêtes à partir du journal de SGBD. Ensuite les attributs indexables font l'objet des prédicats de sélection dans les clauses WHERE des requêtes. L'étape suivante est la construction d'un contexte de recherche des motifs fréquents sous forme d'une matrice binaire < requête-attributs> Les lignes dans cette matrice représentent les requêtes et les colonnes les attributs indexables utilisés par chaque requête. La matrice formalise l'absence ou la présence d'un attribut dans une requête par les valeurs {0,1} respectivement.

Pour réduire cette complexité, l'algorithme CLOSE génère les motifs fréquents fermés. Chaque motif extrait est composé d'un ensemble d'attributs de l'entrepôt de données, le but d'un motif fréquent est de générer un index s'il vérifie les conditions suivantes.

(a) Il contient des clés étrangères de la table des faits. Ces clés sont nécessaires pour la construction des clauses FROM et WHERE de la requête de création de l'index.
(b) Il contient des clés primaires des tables de dimension. Ces attributs sont nécessaires pour construire la clause WHERE pour joindre leurs tables de dimension à la table des faits. Ces tables sont ajoutées dans la clause FROM.
(c) Un ensemble d'attributs non clés des tables de dimension. Ces attributs sont nécessaires pour construire la clause ON.
A la fin de cette étape, un ensemble d'index candidats est construit à partir des motifs fréquents.

La construction de la configuration d'index finale est basée sur un algorithme glouton et un modèle de coût qui permet de calculer la taille des index sélectionnés ainsi que le coût d'exécution des requêtes en présence de ces index.

Afin d'évaluer le coût des index Aouiche et *al* proposent un modèle de coût .Le modèle évalue le coût d'accès aux données avec les index *IJB*, et le coût de maintenance de ces index.

II.3.1.2 Travaux de Bellatreche et al.

Dans [21, 22], Les travaux de Bellatreche et al. présentent une amélioration des travaux de Aouiche et al [20]. L'approche précédente peut éliminer des index sur des attributs non fréquemment utilisés mais qui appartiennent à des tables de dimension volumineuses, ce qui ne permet pas d'optimiser une opération de jointure. Pour pallier ce problème, Bellatreche et al. proposent d'inclure d'autres paramètres dans la génération des motifs fréquents comme la taille des tables de dimension, la taille de la page système, etc.

Les auteurs proposent deux algorithmes *DynaClose* et *DynaCharm* qui constituent une adaptation des algorithmes *Close* et *Charm* utilisés dans la phase d'élagage de l'espace de recherche des index. Au lieu d'utiliser le support comme critère de détermination des motifs fréquents, les algorithmes proposés reposent sur une fonction fitness permettant de pénaliser chaque motif fréquent en prenant en compte les paramètres cités ci-dessous.

Pour un motif fréquent mi, cette fonction est définie comme suit :

$$Fitness(m_i) = \frac{1}{n} \times \left(\sum_{j=1}^{n} \alpha_j \times Sup_j \right)$$

Où n représente le nombre d'attributs non clés A*j* dans mi. Sup*j* représente le support de A*j* et αj est un paramètre de pénalisation défini par l'équation suivante : $\alpha j = \frac{|D_j|}{|F|}$ où $|D_j|$, $|F|$ représentent respectivement le nombre de pages nécessaires pour stocker la table de dimension Dj et la table des faits *F*.

Etant donné un support minimum minsup, une valeur minimum de la fonction fitness *minfitness* est calculée comme suit : $minfit = \frac{minsup}{|F|} \times \left[\left(\sum_{j=1}^{d} \frac{|D_j|}{d} \right) \right]$ où d représente le nombre de tables de dimension.

Les algorithmes *DynaClose* et *DynaCharm* sont utilisés dans la phase d'élagage de l'espace de recherche des index. Une fois les motifs fréquents générés, une étape de purification permet d'éliminer les motifs qui ne peuvent pas générer un index de jointure. Par exemple, un motif fréquent ne contenant aucun attribut non clé des tables de dimension sera supprimé. La purification permet de générer un ensemble d'attributs indexables candidats. Cet ensemble est défini par l'union des attributs non clés appartenant aux motifs fréquents générés. A partir de l'ensemble d'attributs

candidats, un algorithme glouton proposé par les auteurs permet de sélectionner une configuration d'index finale sous une contrainte d'espace de stockage. L'algorithme glouton commence par l'index défini sur l'attribut ayant la cardinalité minimum, ajouter ensuite d'autres index itérativement jusqu'à ce que l'espace de stockage soit consommé ou tous les index sélectionnés.

II.3.1.3 Travaux de Boukhalfa et al

Dans [1], Boukhalfa et *al*. proposent une approche de sélection d'une configuration d'IJB visant à réduire le coût d'exécution d'une charge de requêtes sous une contrainte de stockage. Les auteurs ont proposé deux types d'algorithmes de glouton pour la sélection. Le premier sélectionne une configuration d'index mono-attributs et le deuxième un ensemble d'index multi-attributs.

L'algorithme commence par une configuration composée d'un index mono-attribut défini sur l'attribut ayant une cardinalité minimum noté *BJImin*. La configuration initiale est améliorée itérativement par l'ajout d'index définis sur d'autres attributs non encore indexés (figure II-4). L'algorithme s'arrête lorsque les deux conditions suivantes sont satisfaites: aucune amélioration n'est possible, et l'espace de stockage est consommé.

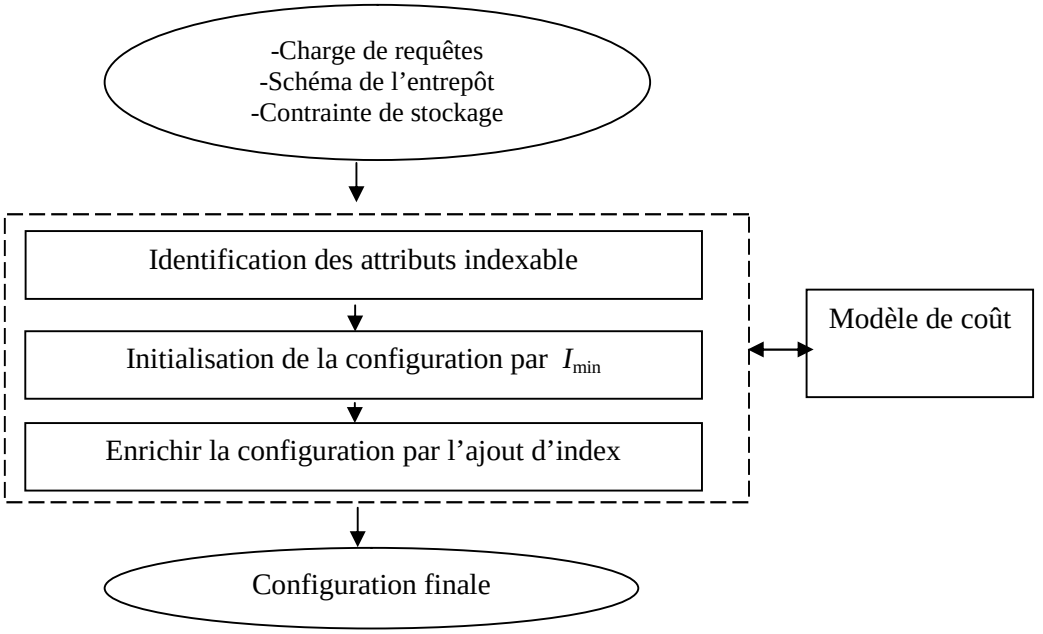


Figure II-4 : Architecture de l'approche de sélection d'IJB mono-attribut

L'approche proposée se déroule en trois étapes : (1) l'identification des attributs indexables : (2) l'initialisation de la configuration, (3) l'enrichissement de la configuration actuelle par l'ajout de nouveaux index. Dans la première étape, Les attributs indexables candidats sont choisis parmi les attributs de sélection de faible et

de moyenne cardinalité noté *BJImin*. La deuxième et la troisième étape sont réalisées au sein d'un algorithme glouton. Cet algorithme commence par une configuration composée d'un index mono-attribut défini sur l'attribut ayant une cardinalité minimum. La configuration initiale est améliorée itérativement par l'ajout d'index définis sur d'autres attributs non encore indexés.

L'algorithme s'arrête lorsque deux conditions sont satisfaites : aucune amélioration n'est possible et l'espace de stockage est consommé.

Dans la sélection d'index multi-attributs, la configuration d'index est représentée par une matrice binaire (index, attribut). L'identification de l'ensemble des attributs indexables *BJISET* représentent tout attribut figure dans le prédicat de sélection qui possède une faible ou une moyenne cardinalité. Pour chaque requête *Qi*, l'index qui couvre tous les attributs indexables est sélectionné; l'union de ces index représente la configuration initiale *CI* pour la construction d'une configuration finale; les auteurs ont proposé plusieurs stratégies d'élimination d'attributs en se basant sur le modèle de coût proposé par Aouiche dans [20]. La figure II-8 représente l'approche de sélection d'IJB multi-attribut.

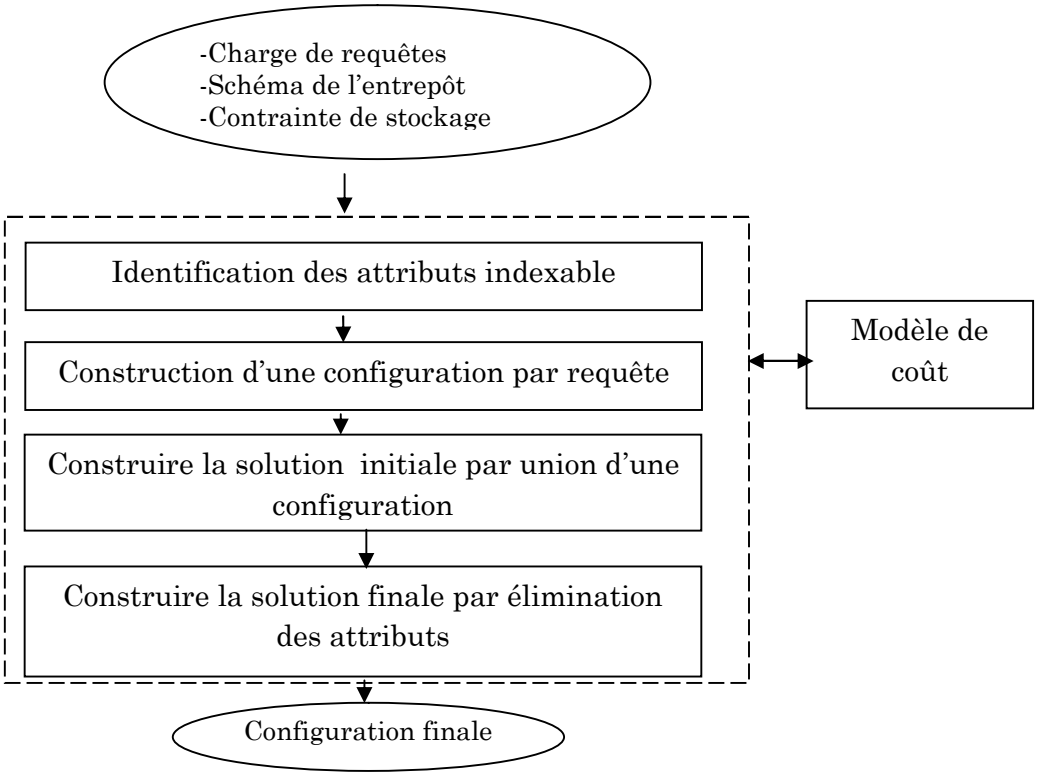


Figure II-5 : Architecture de l'approche de sélection d'IJB multi-attribut

L'approche proposée dans [1] prend en entrée un schéma de l'entrepôt, une charge de requêtes Q composée de m requêtes les plus fréquentes {Q1, Q2, · · ·,Qm} et un espace disque limité et renvoie en sortie une configuration d'IJB multi-attributs permettant

de réduire le coût d'exécution de la charge sous la contrainte de l'espace de stockage. La figure III.5 représente l'architecture de sélection d'*IJB* multi-attribut. L'approche de sélection passe par quatre étapes : (1) l'identification des attributs indexables, (2) la construction d'une configuration par requête, (3) la construction d'une configuration initiale et (4) la construction d'une configuration finale.

La configuration d'index est représentée par une matrice binaire *index-attributs* qui désigne si, l'index I_i est construit sur l'attribut A_j ou non. Les colonnes représentent les attributs indexables et les lignes représentent les index définis sur ces attributs. L'ensemble des attributs indexables est l'union de tous les attributs de sélection de chaque requête possédant une cardinalité faible ou moyenne.

La configuration initiale obtenue peut être très volumineuse. Ce grand volume est dû à deux considérations :

1. Un grand nombre d'index est créé pour satisfaire toutes les requêtes.
2. Les index multi-attributs sont volumineux en fonction du nombre d'attributs indexés et de leurs cardinalités

Si la taille de la configuration initiale est inférieure à la capacité de stockage, alors elle est automatiquement choisie comme configuration finale. Dans le cas contraire, les auteurs proposent des stratégies d'élimination d'attributs afin de réduire la taille de la configuration.

II.3.2 Bilan et discussion

Nous avons abordé dans ce qui a précédé le problème de sélection d'index et présenté quelques travaux qui existent dans la littérature traitant ce problème.

Le tableau II.2 résume les principaux travaux effectués sur la sélection des IJBs. Nous allons résumer et classer ces travaux dans le tableau 3.2 suivant plusieurs critères :

- Méthode de sélection des index candidats.
- Algorithme de sélection d'une configuration finale d'index.
- Catégorie du modèle de coût utilisée pour cette seconde sélection.
- Approche de construction d'index

Travaux	**Sélection des attributs candidats**	**Technique d'élagage**	**Construction de la configuration finale**				**Modèle de coût**
			Gloutton	DM	AG	MSD	
Bellatreche et *al* [21][22]	Manuelle	DATAMING DynaClose et DynaCharm	Descendant	**x**			Fonction de coût
Boukhalfa et *al* [1]	Automatique	Règle a priori	Ascendante				Fonction de coût
Aouiche et *al* [20]	Automatique	DATAMIING Recherche des motifs fréquents(CLOSE)	Descendant	**x**		**x**	Fonction de coût
Chaudhuri et al. 2004[14]	Automatique	CLIUSTRING	Ascendante				Appel à l'optimiseur
Golfareli et al. [25]	Automatique		Ascendante				Appel à l'optimiseur

Tableau. II.2 Comparaison des travaux effectués sur les index

Dans le tableau on désigne par : *DM* : Datamining, AG : algorithme génétique, IJB : index de jointure binaire, MSD : Modèle Sac à dos et le caractère (**x**) indique «oui» et un espace vide indique que «non».

II.4 **Problème de sélection d'un schéma de fragmentation horizontale**

La fragmentation est une technique d'optimisation de requêtes qui a été largement étudiée dans le domaine des bases et des entrepôts de données [26]. Les méthodes proposées exploitent une fragmentation horizontale primaire et dérivée qui a été très efficace pour le traitement des requêtes décisionnelles. Elle réduit le coût de traitement d'une requête en minimisant le nombre d'accès aux enregistrements non nécessaires [27]. Plusieurs travaux commerciaux et académiques ont montré l'intérêt de la fragmentation horizontale. Plusieurs SGBDs commerciaux ont proposé des langages de définition de données pour supporter la fragmentation. La fragmentation horizontale primaire et dérivée étant un problème NP-complet [1] en raison de la complexité de ce problème, les travaux effectués dans ce cadre sont basés sur des méthodes heuristiques.

II.4.1 **Formalisation du problème**

Sélectionner le schéma de fragmentation optimal est une tâche difficile. Le problème de sélection de la FH est formalisé dans [7] comme suit :

Entrées:

- Un entrepôt de données ayant une table de faits F et n tables de dimensions D = {D1…Dn},

- Une charge de requêtes Q = {Q1,Q2, ...,Qm}, où chaque requête Qj possède une fréquence d'accès fj

- W : le nombre de fragment maximum générés

Le problème de sélection d'un schéma de FH consiste à partitionner F en p fragments horizontaux F1, F 2,…,Fp tel que le coût d'exécution de Q en terme E/S sur la table fragmentée est minimale et que la contrainte (P<=W) soit vérifiée.

Les travaux qui traitent la fragmentation dans les entrepôts de données relationnels s'inspirent de ceux proposés dans les bases de données relationnelles [38],[33] et orientées objets [26]. Cette fragmentation peut être horizontale, verticale ou mixte.
Nous présentons dans ce qui suit les principaux algorithmes utilisés pour la sélection d'un schéma de FH.

II.4.2 Algorithmes de sélection d'un schéma de FH

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 coût. Tous ces algorithmes commencent par un entrepôt 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.

a. Algorithmes basés sur les prédicats :

L'approche basée sur les prédicats est composée de cinq étapes principales, à partir d'un ensemble de transaction. L'algorithme COM-MIN génère un ensemble complet et minimal de prédicats (respectant la notion de complétude et de minimalité). Les entrées de cet algorithme : une relation *R* et un ensemble de prédicats simples *Pr*. La sortie: un ensemble de minterm *M (*plusieurs travaux se sont basés sur cet algorithme [30], [31], [34]), l'étape suivante consiste à éliminer les minterms contradictoires, ensuite on génère des fragments de telle sorte que chaque minterms m_i corresponde un à fragment Ti =σ m_i(T).

L'approche basée sur les prédicats est caractérisée par une grande complexité par rapport au nombre de prédicats. Pour n prédicats simples, cette approche génère 2^n minterms

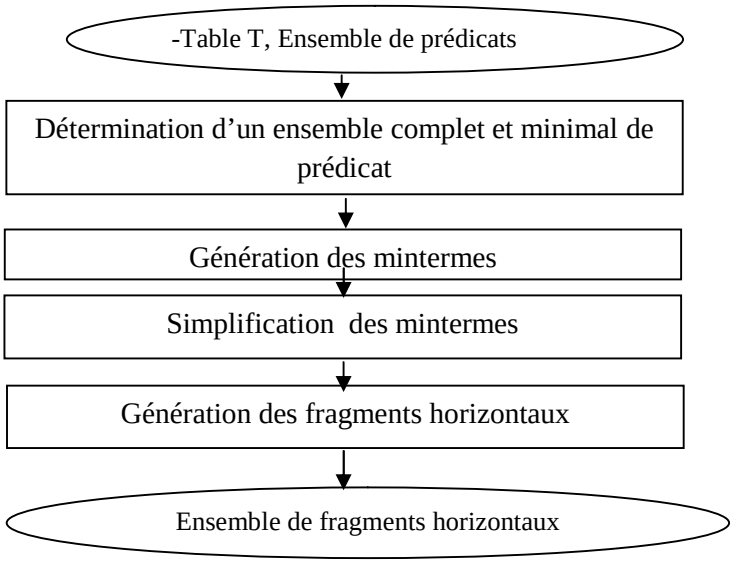


Figure II-6 : Approche basée sur les prédicats

b. Algorithmes basés sur l'affinité:

Cette approche a été proposée dans le souci de réduire la complexité de l'approche basée sur les prédicats. Cette approche est composée de cinq étapes :
(1) Enumération des prédicats simples, (2) Construction de la matrice d'usage des prédicats, (3) Génération de la matrice d'affinité des prédicats, (4) Regroupement des prédicats et (5) Génération des fragments horizontaux.

(1) Enumération des prédicats simples de la sélection définis sur les attributs de la table à fragmenter.
(2) La construction de la matrice d'usage < requête, prédicat > par la fonction suivante :

$$Use(qj,Pj)=\begin{cases}1 & \text{Si le prédicat } Pj \text{ est utilisé par la requete } qj\\ 0 & \text{sinon}\end{cases}$$

(3) A partir de cette matrice on génère la matrice d'affinité des prédicats <m x m> notée *aff* où m est le nombre de prédicats. La valeur de chaque élément x_{ij} correspond à la somme des fréquences d'accès des requêtes utilisant simultanément les deux prédicats P_i,P_j. Bellatreche et al. [30] ont utilisé deux autres valeurs dans la matrice d'affinité :
" □ " qui indique que le prédicat Pi implique Pj
" □ " qui indique qu'il y a une similarité entre Pi et Pj .

(4) Regroupement des prédicats : Zhang et al [40] ont utilisé l'algorithme *BEA*. Cet algorithme est appliqué sur la matrice d'affinités des prédicats et permet de générer un ensemble de semi-blocs diagonaux. Bellatreche et al [34, 35] exploitent un algorithme graphique proposé par Navathe*et al* [36] pour la fragmentation verticale et adapté à la fragmentation horizontale. Cet algorithme prend comme entrée la matrice d'affinité *aff* en la considérant comme un graphe complet. Un nœud de cet arbre représente un prédicat de sélection *Pi* dans la matrice et un arc entre deux prédicats *Pi,Pj*est la valeur de la cellule *aff (Pi,Pj)*. L'algorithme détecte et extrait un ensemble de cycles *C* appelés composantes.

(5) Génère des fragments horizontaux: la génération des fragments horizontaux dans les travaux de Zhang et al consiste à associer chaque fragment à un semi-bloc diagonal. Dans [26], la génération des fragments est effectuée par l'association de chaque composante *Ci* à un fragment.

La complexité de cet algorithme est : O ($n \times N + N^2$) [34]. Cette approche est donc moins complexe que celle basée sur les prédicats dont la complexité est $O(2^n)$.

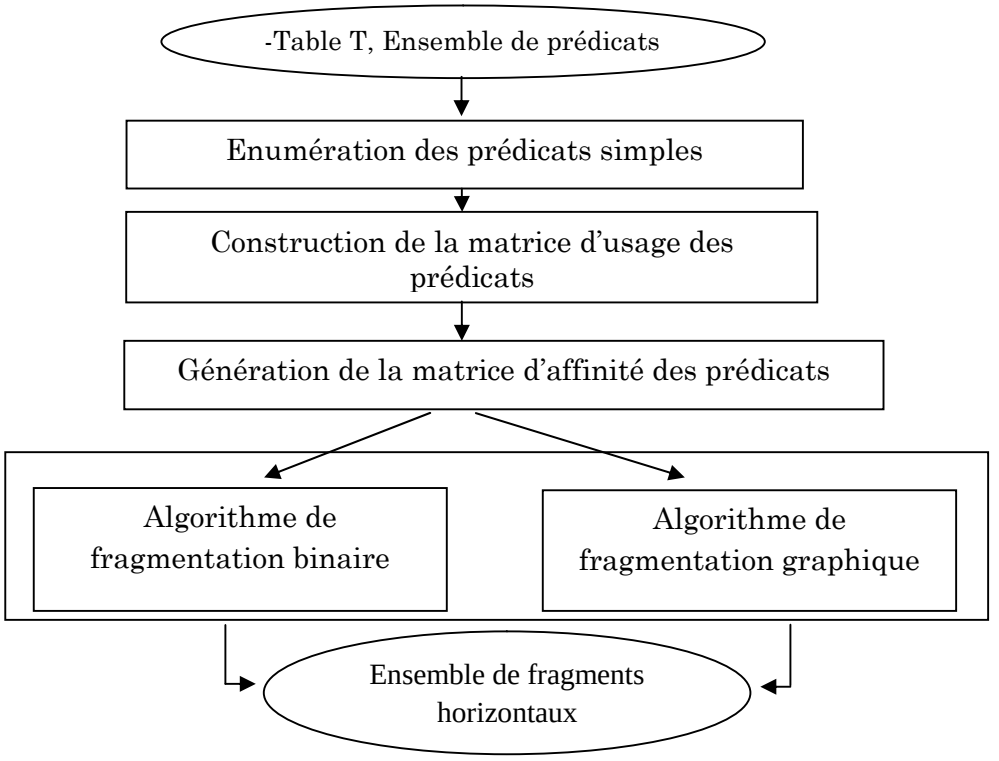


Figure II-7: Approche basée sur l'affinité

c. Algorithmes basés sur un modèle de coût :

Cette approche repose sur l'évaluation de la qualité de chaque schéma de fragmentation candidat en utilisant un modèle de coût mathématique [34]. Trois phases caractérisent cette approche :

1. La génération des schémas de fragmentation possibles à partir les minterms primitifs et leurs combinaisons
2. L'évaluation des schémas générés en utilisant un modèle de coût
3. La sélection du schéma de fragmentation quasi optimal ayant un coût minimum. Les auteurs de cette approche ont proposé deux algorithmes, un algorithme exhaustif et un algorithme approximatif. L'algorithme exhaustif énumère tous les schémas de fragmentation possibles, chaque schéma de coût minimum sera sélectionné. L'algorithme approximatif proposé est de type Hill Climbing permet de parcourir un sous-ensemble des schémas de fragmentation possibles et renvoie par conséquent à un schéma de fragmentation quasi optimal.

Cette approche permet de sélectionner un schéma de fragmentation quasi optimal mais elle est limitée par le fait qu'elle ne permet aucun contrôle sur le nombre de fragments générés.

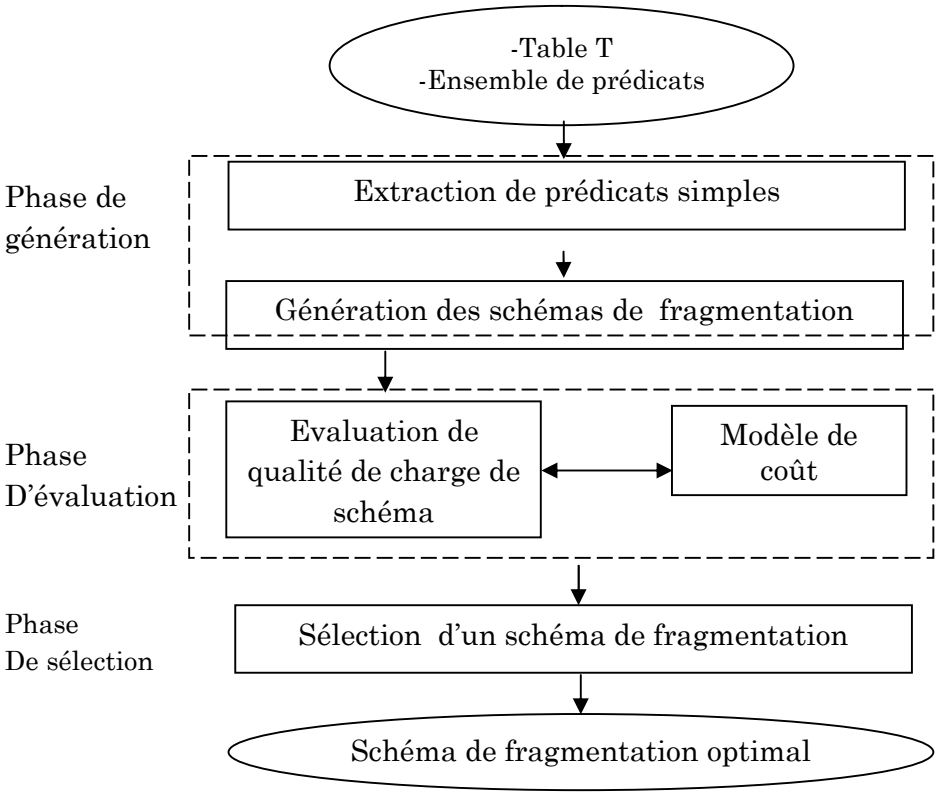


Figure II-8: Approche basée sur un modèle de coût

Le principal inconvénient de ces méthodes, est que l'administrateur n'a aucun contrôle sur le nombre des fragments générés. Les travaux de Boukhalfa et *al* [31],

assurent la contrainte W (W fixé par l'administrateur), qui contrôle le nombre des fragments générés. Les auteurs ont proposé une démarche de fragmentation qui permet à partir d'un ensemble de requêtes les plus fréquentes, un schéma de l'entrepôt et un seuil W représentant le nombre de fragments de la table des faits à générer, de sélectionner un schéma de fragmentation permettant de partitionner le schéma en étoile initial en un ensemble de sous-schémas en étoile. Le schéma de fragmentation sélectionné réduit le temps d'exécution de l'ensemble des requêtes d'entrée et génère un nombre de fragments inférieur ou égal à W. Cette démarche est caractérisée par quatre principales phases:

(1) La préparation de la fragmentation par la collecte des informations quantitatives (la fréquence d'accès des requêtes, facteur de sélectivité des prédicats de sélection et de jointure) et qualitative (les prédicats utilisés par les requêtes les plus fréquentes). L'étape de préparation de la fragmentation est composée de quatre sous-étapes [37] :

(i) L'extraction des prédicats de sélection, (ii) l'identification des tables de dimensions concernées par le processus de fragmentation, (ii) la généralisation d'un ensemble des prédicats complet et minimal pour chaque table candidate et (iv) le découpage du domaine de chaque attribution en sous-domaines,

(2) La sélection d'un schéma de fragmentation : le problème de sélection d'un schéma de fragmentation est NP-Complet [1]. Les auteurs ont proposé trois heuristiques : hillclimbing, un algorithme génétique et un recuit simulé.

(3) La fragmentation des tables de dimension : Le schéma de fragmentation obtenu à l'issue de l'étape précédente propose un nouveau découpage des domaines des attributs de sélection en un ensemble de sous-domaines. La fragmentation des tables de dimension repose sur ce découpage.

(4) La fragmentation de la table des faits : La fragmentation dérivée de la table des faits est effectuée en utilisant les schémas de fragmentation des tables de dimension. Pour une table de dimension fragmentée on génère les fragments de la table de faits. Toutes les combinaisons possibles entre les fragments de dimension sont considérées [37,38]. Chaque fragment de faits est obtenu par une semi-jointure entre la table des faits et une combinaison des fragments de dimension comme suit:

$Fi = F \ltimes D1j \ltimes D2k \ltimes \cdots \ltimes Dgl$ $(1 \leq j \leq m1, 1 \leq k \leq m2, \cdots, 1 \leq l \leq mg)$.On désigne par : F : la table de faits, Fi :fragments de la table de faits et Djk :schémas de fragmentation des tables de dimension.

Ces travaux montrent l'amélioration des performances obtenues en appliquant la technique de la fragmentation sur un entrepôt de données. La performance d'un entrepôt de données peut être améliorée en réduisant le nombre de fragments non pertinents accessibles par les applications. L'optimiseur dans ce cas ne charge que les fragments pertinents. La fragmentation horizontale primaire et dérivée réduite la sélection et la jointure.

Travaux de Mahboubi

Dans [9], l'auteur s'intéresse à la fragmentation horizontale des entrepôts de données XML afin de les répartir sur plusieurs sites. Pour ce faire, il se base sur l'adaptation des techniques de fragmentation existantes sur les entrepôts XML comme la fragmentation horizontale primaire basée sur les prédicats et celle basée sur les affinités de prédicats. Ensuite, il présente une nouvelle méthode à base du concept de fouilles de données à savoir la fragmentation basée sur la classification des prédicats. Elle est effectuée en trois étapes :

- Codage des prédicats de sélection de la charge de requêtes dans une matrice binaire qui représente le contexte de classification. Elle est similaire à la matrice d'usage des prédicats.
- Classification des prédicats par application de la technique des k-means qui permet de partitionner l'ensemble des prédicats en k classes disjointes.
- Construction des fragments. Cette étape exploite l'ensemble de classes et le document XML qui représente le schéma de l'entrepôt pour construire un autre document XML sur le schéma de fragmentation. Afin d'assurer la complétude, un fragment supplémentaire est ajouté par négation de la disjonction de tous les prédicats.

II.4.3 **Bilan et discussion**

Nous pouvons conclure que les approches guidées par les prédicats et celles guidées par les affinités ne proposent aucun mécanisme permettant d'estimer la qualité du schéma de fragmentation obtenu. Les approches guidées par les affinités supposent que la fréquence d'accès est le paramètre le plus important dans la définition d'un schéma de fragmentation. Cela n'est pas toujours vrai, car dans le contexte d'entrepôts de données, certaines requêtes non fréquentes peuvent engendrer un coût d'exécution important [1].

Le tableau II.3 résume les principaux travaux effectués sur la fragmentation horizontale. Il montre pour chaque travail l'approche de fragmentation suivie (basée prédicats, affinité ou modèle de coût), l'algorithme utilisé et le modèle de coût utilisé.

Travaux	Approche de fragmentation	Algorithme de sélection	Modèle de coût
Bellatreche et al. [25]	Basée sur un modèle de coût	Algorithme Glouton	mathématique
Boukhalfa et al. [1]	Basée sur un modèle de coût	Heuristique (Hill Climbing, recuit simulé, algorithmes génétiques)	mathématique
Ozsu et al. [15]	Basée sur les prédicats	----------	Non
Zhang et al. [24]	Basée sur les affinités des prédicats	BEA + Groupement graphique	Non
Mahboubi et al. [9]	basée sur la classification des prédicats	algorithme de fouille de données (k-means)	mathématique

Tableau II.3: Comparaison des travaux effectués sur la fragmentation horizontale

Nous avons présenté dans cette section, des méthodes de sélection de la FH et des IJBs selon le mode isolé, où chaque technique est sélectionnée toute seule. Cependant, dans les applications réelles d'entreposage, l'utilisation d'une seule technique n'est pas suffisante pour optimiser toutes les requêtes [1]. En conséquence, l'administrateur doit choisir plus d'une technique d'optimisation. Après ce constat, la question à se poser est de savoir comment offrir aux administrateurs une sélection multiple.

II.5 **Sélection Multiple de Schémas de Fragmentation et d'Index de Jointure binaire:**

La sélection des techniques d'optimisation, est faite en deux modes : le mode isolé, et le mode multiple. La sélection isolée des techniques d'optimisations n'est pas suffisante pour satisfaire le maximum de requêtes [1]. Pour résoudre ce problème, deux solutions principales s'offrent aux administrateurs : une solution naïve (séquentielle) et une solution conjointe. Dans la sélection séquentielle, l'administrateur sélectionne une technique après l'autre d'une manière indépendante, la sélection multiple dans ce cas est un problème global décomposé en plusieurs sous-problèmes, où chacun est traité indépendamment des autres. Finalement, l'administrateur exécute l'ensemble des recommandations issues des différents sous-problèmes dans n'importe quel ordre.

Pour satisfaire un maximum de requêtes, il est préférable d'opter pour une sélection conjointe de plusieurs techniques, en exploitant la similarité donnée par une fonction d'interdépendance. Cette sélection apporte de meilleures améliorations, car chaque technique permet de palier aux manques de l'autre. La fragmentation horizontale et les index de jointure binaires partagent les mêmes attributs et optimisent la sélection et la jointure. Les deux problèmes (FHD, IJB) sont NP-Complet, ce qui implique Leur combinaison à augmenter la complexité du problème.

II.5.1 Formalisation du problème

Boukhalfa et al [1], ont formalisé ce problème comme un problème d'optimisation avec contraintes comme suit :
Étant :
- Un entrepôt de données modélisé par un schéma en étoile composé d'une table de faits F
et de d tables de dimension {*D1,D2, ⋯,Dd*} ;
- Une charge de requêtes *Q*= {Q_1,Q_2, ⋯,Q_m} où chaque requête *Qi* possède une fréquence d'accès *fi*;
- Un seuil W représentant le nombre de fragments générés par le processus de fragmentation ;
- Un espace de stockage S réservé aux *IJB*.

L'objectif recherché est de sélectionner simultanément un schéma de *FH* (*SF*) et une configuration d'IJB (*CI*) tel que :

- Le coût d'exécution de Q en présence de *SF* et *CI* soit réduit ;
- Le nombre de fragments générés par *SF* soit inférieur ou égal à W (NSF ≤W)
- La taille de *CI* est inférieure ou égale à l'espace de stockage réservé (Taille(*CI*) ≤*S*).

Deux principaux travaux ont traité le problème de sélection combinée de la FH et des IJB [31][32]. D'autres travaux sur la sélection conjointe ont été proposés, la sélection des vues matérialisées et d'index (voir [42, 43, 44]), la sélection de la fragmentation horizontale et verticale (voir [36]), et la sélection de la fragmentation horizontale, les index et le traitement parallèle (voir [46]).

II.5.2 Approche de Boukhalfa et al

Boukhalfa et al [1] ont proposé une approche de sélections simultanée basée sur les dépendances entre la *FHD* et les *IJB*. Elle consiste à sélectionner un schéma de FH en se basant sur une charge de requêtes, ensuite sélectionner une configuration d'*IJB* en prenant en compte le schéma de fragmentation sélectionné et les requêtes non bénéficiaires du processus de fragmentation. La sélection d'un schéma de FH avant celle des *IJB* permet de réduire la complexité du problème de sélection des *IJB*. L'approche proposée est la sélection simultanée d'un schéma de FH et d'une configuration d'*IJB*. Elle est constituée de cinq étapes principales :

1. **Énumération de l'ensemble des attributs de sélection** : énumérer l'ensemble des attributs de sélection qui sont définis dans la clause WHERE notés EAS. Les *IJB* sont définis sur les attributs de faible cardinalité, donc l'ensemble d'attributs de départ noté EAFC (EAFC □EAS).

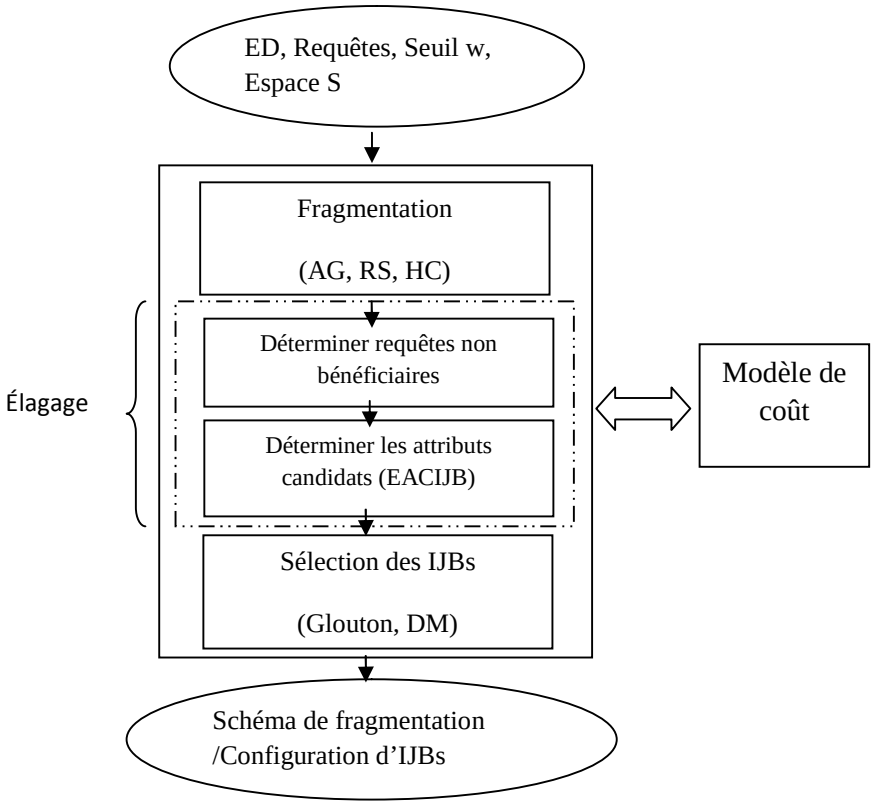


Figure II-10 : Approche de sélection de FHD & IJB [1]

2. **Génération d'un ensemble de prédicats minimal et complet *:*** on génère l'ensemble EASCM (EASCM □EAS) avec EASCM l'ensemble de prédicats respectant la notion de la complétude et de la minimalité.

3. **Sélection d'un schéma de fragmentation** : La sélection d'un schéma de fragmentation est effectuée par l'un de ces Algorithme (Génétique AG, Recuit Simulé RC ou Hill. Climbing HC). L'algorithme prend en entrée la charge de requête initiale *Q = {Q1, Q2, ….,Qm},* l'ensemble *EASCM* et le seuil W fixé par l'administrateur.
Le schéma de fragmentation obtenu est défini par :
-L'ensemble de tables de dimension fragmentées.
-Pour chaque table de dimension fragmentée, l'ensemble des attributs utilisés pour la fragmenter.
Donc parmi les attributs dans *EASCM*, un sous-ensemble EAF est utilisé pour fragmenter l'entrepôt (EAF □EASCM).
-Pour chaque attribut de fragmentation, le découpage de son domaine en partitions

4. **Identification des requêtes bénéficiaires** : Le but de cette étape est de réduire le nombre de requêtes pris en compte dans la charge *Q* sans perdre de la qualité des recommandations générées. La sélection des IJB repose sur l'ensemble des requêtes non bénéficiaires. Une requête est bénéficiaire du processus de fragmentation si son

coût d'exécution après fragmentation est significativement réduit, cette réduction est quantifiée par métrique appelée taux de réduction (T*R*) défini comme suit :

$$TR(Q_j) = \frac{Cost\ (Q\,j\ ,SF)}{Cost\ (Q\,j,\square)}$$

Où *Cost*(*Qj* ,*SF*) et *Cost*(*Qj*, □) représentent respectivement le coût d'exécution de la requête *Qj*sur l'entrepôt fragmenté selon le schéma SF et sur le schéma non fragmenté.

L'ensemble des requêtes non bénéficiaires *Q'* est défini par :

$$Q' = \{Q'j \ / Q'j \square\ Q\ \square TR\ (Q'_k) < \lambda\} \quad \text{avec } \lambda \square\ [0,1]$$

5. **Identification des attributs candidats à l'indexation**: A partir de l'ensemble précédent Q', on identifie l'ensemble des attributs de sélection noté $EAS_{Q'}$. L'ensemble des attributs candidats pour la sélection d'une configuration d'index (noté EACIJB) défini par : $EACIJB = (EAFC - EAF) \cap EASQ'$.

L'élagage de l'espace de recherche d'IJB dépend de deux paramètres : le nombre d'attributs de fragmentation et le nombre de requêtes bénéficiaires.

6**. Sélection d'une configuration d'index** : L'algorithme de sélection d'une configuration d'*IJB* reçoit en entrée l'ensemble *EACIJB*, l'ensemble des requêtes *Q'* et l'espace disque S réservé aux *IJB*. Il donne en sortie une configuration d'index *CI*. Deux algorithmes de sélection d'index peuvent être utilisés dans cette approche, un algorithme glouton et un algorithme basé sur les techniques de data Mining. Les algorithmes de sélection d'un schéma de FH et d'une configuration d'index basés sur deux modèles de coûts, le premier permet d'évaluer la qualité des différents schémas de fragmentation générés et le deuxième permet de calculer la taille des différents IJB générés ainsi que le coût d'exécution.

II.5.3 Élagage de l'espace de recherche des *IJB*s

Le nombre d'IJB possibles augmente de manière exponentielle par rapport au nombre d'attributs candidats pour la sélection. Si le nombre d'attributs est important, il est quasiment impossible d'énumérer toutes les configurations possibles [1]. Pour sélectionner un ensemble d'index, plusieurs travaux proposent deux étapes, la première consiste à éliminer un ensemble d'attributs candidats, et la deuxième consiste à sélectionner une configuration en utilisant le nouvel ensemble d'attributs obtenu après élagage [11,32]. La première étape permet d'élaguer l'espace de recherche des index et par conséquent diminue sa complexité.

Dans [32] l'étape d'élagage est effectuée en utilisant des techniques de datamining. Ce qui différencie l'approche d'élagage proposée [1] dans ces travaux réside dans la

façon dont cet élagage est effectué. Au lieu d'utiliser des techniques de datamining, leur approche utilise la fragmentation horizontale pour élaguer l'espace de recherche d'une deuxième technique (IJB) en se basant sur les dépendances existant entre elles. Puisque le schéma de fragmentation sélectionné dans la première étape utilise un sous ensemble d'attributs d'index, alors cet ensemble ne doit pas être inclus dans l'espace de recherche des IJB; ce qui implique que la taille de l'espace de recherche des IJB diminue de manière très significative, ce qui permet de réduire la complexité du problème de leur sélection.

Les auteurs dans [32] proposent une démarche qui cherche à trouver un compromis entre les deux contraintes (nombre maximum de fragments, le stockage des index), ils ont proposé de répartir les attributs entre deux classes : *Classe_IJB* et *Classe_FH*, avant de sélectionner les deux techniques. Cette classification réduit le nombre d'attributs à introduire dans la fragmentation ; l'approche proposée par les auteurs (*figure* II.13**)** comporte quatre étapes: (1) Extraire les attributs de sélection à partir d'une charge de requêtes,(2) Classifier ces attributs entre FH et IJB par la méthode de classification « k-means » avec k=2. (3) Définir un schéma de fragmentation sur la classe FH. (4) et définir les IJB sur la classe IJB à laquelle on ajoute les attributs non sélectionnés par FH.

La classification des attributs de sélections s'effectue en deux phases : (1) pondération des attributs (le poids c'est la somme de nombre de requêtes, le facteur de sélectivité et la cardinalité de l'attribut). (2) A travers « k-means » et ce poids, classifier l'*attribut* de sélection.

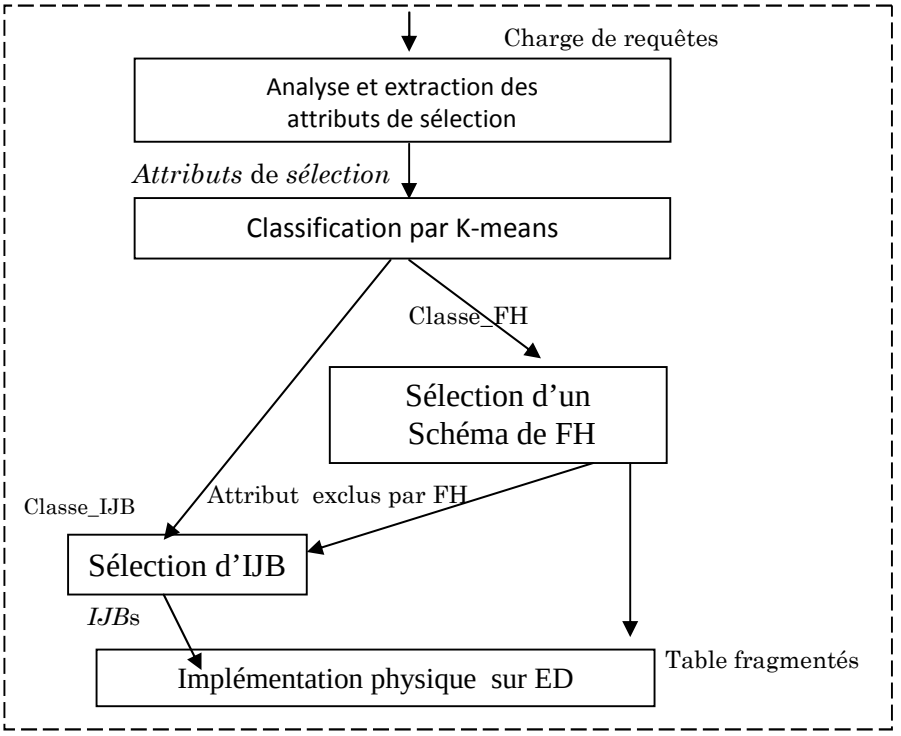


Figure II-9 : processus de sélection d'IJB et FH avec classification

II.5.4 Comparaison des travaux effectués sur la FH et l'IJB

Peu de travaux traitent la sélection FH&IJB. L'interaction entre ces deux techniques pourra être exploitée pour élaguer l'espace de recherche des *IJBs*. Dans [32], l'élagage est effectué en utilisant des techniques de datamining. Dans [1] l'élagage est effectué en utilisant la technique de la fragmentation horizontale.

Travaux	Type de fragmentation horizontale	Elagage	Technique d'élagage	Algorithme de sélection	Algorithme de sélection	Modèle de coût	Contrôle de W
Boukhalfa et al[1]	Primaire et dérivée	Espace IJB	FH	Basée sur un modèle de coût	Hill Climbing Algorithme génétiques Recuit simulé	Mathématique	Oui
Rym et *al* [32]	Primaire et dérivée	Espace IJB	Datamining (k-means)	Basée sur un modèle de coût	Algorithme génétiques	Mathématique	Non

Tableau II.3 Comparaison des travaux effectués sur la FH et l'IJB.

Conclusion

Nous avons présenté dans ce chapitre un état de l'art sur les principales techniques d'optimisation de requêtes définies dans le contexte des entrepôts de données relationnels : les index, la fragmentation horizontale primaire et la fragmentation horizontale dérivée. Pour chaque technique, des algorithmes de sélection sont présentés.

Nous avons constaté par les travaux de littérature, l'intérêt de l'élagage de l'espace de recherche des techniques d'optimisation et les insuffisances de la sélection énumérative qui ne peuvent pas fournir un rapport d'exécution performant. Pour mettre en œuvre l'élagage afin de réduire la complexité, deux modes d'implémentation sont décrites : l'élagage inter-technique et l'élagage intra-technique. Chaque implémentation est basée sur un algorithme spécifique. Nous distinguons deux types d'élagage inter-technique : l'élagage inter-technique séquentiel et élagage conjoint mutuel. L'élagage inter-technique est motivé par la nature des dépendances détectées entre ces techniques. Par exemple, la forte dépendance entre les vues et les index a mené la combinaison de ces deux techniques dans le même module. Chaque module génère ses recommandations avec ses propres contraintes. La sélection des techniques dans chaque module est implémentée par un algorithme spécifique. Les modules ne sont pas totalement indépendants, puisque chaque module met à jour les statistiques qui seront utilisées par les autres modules.

La complexité de la sélection combinée d'IJB et de FH ainsi que le processus d'élagage, nous a motivés à s'intéresser à une approche basée sur les SMA pour

sélectionner les schémas d'indexation et de fragmentation. Les SMA modélisent un système complexe qui se caractérise par la notion d'objectif commun et la coopération entre plusieurs entités. A notre connaissance, cette approche n'a jamais été étudiée pour sélectionner les techniques d'optimisation dans le contexte d'entrepôt de données. Nous proposons, pour résoudre ce problème, une approche basée sur les SMA, qui sont particulièrement bien adaptés à la modélisation des interactions entre ces techniques d'optimisation.

Le chapitre suivant présente notre approche de sélection d'IJB et de FH dirigée par une architecture multi-agents.

Chapitre III : Nouvelle démarche dirigée par les SMA pour la sélection d'index et de fragmentation

Introduction

Rappelons que la sélection isolée des structures d'optimisation c'est problème un NP complet, et elle n'est pas toujours suffisante pour optimiser toute la charge de requêtes. Pour satisfaire le maximum de requêtes ; il est indispensable d'opter pour une sélection multiple de structures ayant une forte similarité, ce qui engendre une complexité énorme. La sélection multiple est plus complexe que la sélection isolée vu la complexité de son espace de recherche composé des différentes combinaisons des structures concernées.

Nous proposons dans ce travail une nouvelle approche de sélection des IJB et de la fragmentation horizontale dirigée par une architecture Multi-agents. Cette approche se concentre sur l'élagage des espaces de recherches des deux problèmes de sélection afin de réduire leurs complexités. Ce système vise par ailleurs à minimiser le temps d'exécution de la charge de requêtes.

Une grande partie du travail de ce memoire se concentre sur la conception, le développement et la validation notre approche qui permet de donner des solutions optimales ou quasi optimales.

La modélisation de notre problème est représentée par une architecture Multi-agent. Cette architecture nous montre l'aspect distribué du système, les interactions et les relations qui peuvent avoir lieu entre ces techniques d'optmisation. Les agents de notre système SMA utilisent les modules de processus de sélection des techniquesd'optmisation (extration, sélection, génération, évalutaion) développés dans la plupart de travaux cités dans l'etat de l'art.

Nous allons présenter dans la première partie les différents agents qui composent notre SMA ainsi que leurs fonctionnements internes. Nous allons décrire aussi l'organisation du système considéré, afin de situer les interactions, la communication et les coordinations entre ces agents. La deuxième partie présente la projection de notre système sur le problème de sélection l'IJB et de la FH.

Plusieurs niveaux aident à la réduction de la complexité de problème de sélection des techniques d'optimisations :

- Aborder la complexité par une démarche de modélisation systémique utilisant les deux concepts, modularité et parallélisme.

- Répartition des attributs entre les techniques d'optimisation, où chaque technique est représentée par un agent qui traite un espace de solution restreint.

- En utilisant généralement des heuristiques qui cherchent une bonne solution en un temps de réponse raisonnable. Aussi les méta-heuristiques sont adaptables et applicables à une large classe de problèmes.

Nous présentons dans ce qui suit notre motivation d'utiliser le SMA et la modélisation de processus de sélection des structures d'optimisation de l'ED par la méthodologie Voyelles « AEIO ».

Nous nous inspirons des approches de sélection des structures d'optimisation proposées dans littérature. Afin de garantir une bonne coopération, chaque agent de système a intérêt à connaître les compétences de ses collègues, pour accéder directement, en cas de besoin. Il peut y avoir des communications inter techniques.

III .1 **Motivation (Pourquoi SMA ?)**

Le processus de sélection des structures d'optimisation est distribué, coopératif coordonné et se caractérise par une complexité (NP complet). Les avantages d'utiliser le SMA dans la modélisation des systèmes de sélection des techniques d'optimisation afin d'automatiser les tâches de l'AED peuvent être récapitulés comme suit :

(1) Capacité de modéliser des systèmes complexes réellement :

Puisque nous employons une collection d'agents intelligents interconnectés aussi bien que les entités autonomes pour modéliser l'environnement, le modèle en résultant est une approximation plus réaliste de l'environnement fondamental qui est constitue d'un ensemble des techniques d'optimisation (index, vues matérialisées, fragmentions horizontale et verticale).

(2) L'intégration des techniques d'optimisation d'ED

Avec une meilleure interaction entre les agents représentant de diverses fonctionnalités d'une approche générique de sélection des techniques d'optimisation de l'ED, le temps de réponse opérationnel peut être rigoureusement ravalé et la coordination opérationnelle inter et intra-technique est améliorée. Ceci mène pour à améliorer le temps d'exécution de la charge de requêtes.

(3) Décomposition de processus de la sélection en plusieurs étapes

Le SMA s'intéresse à des systèmes dans lesquels des agents opèrent collectivement et de façon décentralisée pour accomplir une tâche complexe à appréhender de manière globale. La résolution doit s'opérer par composition de solutions élaborées de manière locale.

(4) Exécution agile des différentes opérations

La distribution des calculs, et la décentralisation de la décision par les systèmes Multi-agent peuvent mener à des modules plus flexibles qui peuvent efficacement exécuter les opérations (extraction, évaluation, vérification des contraintes) et traiter le changement à de divers niveaux.

(5) L'existence d'un système d'apprentissage améliore la qualité de prise de décision :

L'utilisation des Systèmes Multi-agent comme circuit principal de processus décisionnel améliore la qualité globale des décisions en raison de leur approche holistique[3].

(6) Caractéristique de l'ED :

- données volumineuses : table de fait de plusieurs Giga ou Téra octets
- nombre important des tables de dimensions et des attributs.
- les requêtes sont nombreuses (plusieurs dizaines) et très complexes

Lors de notre étude, les concepts fondamentaux des SMA comme la complexité, l'autonomie, l'interaction et les aspects coopératifs nous sont apparus très présents et importants pour résoudre le problème traité.

a) Objectif commun: les techniques d'optimisation de l'ED réalisent une tâche collective : optimiser les requêtes des décideurs.

b) Complexe : la complexité du système en raison (schéma de ED très grand, requêtes complexes, nombre important des structure d'optimisation, les contraintes des techniques d'optimisation).

c) Interaction : les techniques d'optimisation présentent une interaction inter et intra technique tout dépend le degré de similarité entre ces techniques.

d) Autonome : l'autonomie apparait en terme de contrôle de l'agent, ce contrôle est distribué dans le système, chaque technique à un contrôle local (nombre de fragments, limite de l'espace de stockage et de maintenance..) et un objectif commun (optimiser la charge des requêtes). Chaque agent agit selon leurs buts, compétences, préférences.

e) Environnement commun : les techniques d'optimisation partagent un environnement commun qui présente les différentes ressources partagées(les attributs de sélections et l'espace disque).

f) Evoluer dans l'environnement : L'entrepôt de données évolue au cours de sa durée de vie. Cette évolution concerne plusieurs aspects : (i) le contenu des tables, (ii) la taille des techniques d'optimisation sélectionnées lors de la conception physique (les index et les partitions), (iii) les fréquences des requêtes/mises à jour, (iv) l'ajout et la suppression de requêtes, etc. Ces changements nécessitent une dynamique pour

[3]Chaque partie d'un système est un reflet du tout

pouvoir maintenir une meilleure performance de l'entrepôt et surtout éviter sa dégradation.
Nous présenterons dans ce qui suit une modélisation de notre approche de sélection des structures d'optimisation de l'ED à base de SMA.

III .2 Modélisation globale du système

Un système complexe, nécessite la distribution des tâches entre des " entités " autonomes afin d'atteindre son objectif d'une manière optimale, et établi à base de Système Multi-agents. Le SMA met en lumière une forte composante interactionnelle. Parmi les méthodes qui couvrent le mieux le cycle de développement d'un système Multi-agents, nous avons choisi la méthodologie Voyelles pour modéliser notre système.

Cette méthode repose sur la décomposition du système en cinq dimensions : Agent, Environnement, Interaction, Organisation, Utilisateur. Cette décomposition modulaire du système permet de simplifier sa construction et offrir une meilleure réutilisation du code.

Voyelles n'est couplée a aucune notation ni plateforme, ce qui offre la possibilité d'utiliser le langage AUML (Agent Unified Modeling Language), qui est une extension du langage UML, dans la phase de conception du système et la plateforme Jade (3) pour réaliser le système.

La décentralisation de ce processus de sélection présente un avantage qui améliore les performances globales du système. Par analogie, nous proposons un SMA composé de groupes d'agents, où les éléments en interaction de chaque technique sont fortement liés.

L'objectif principal de notre travail est de réaliser un système SMA afin d'élaguer l'espace de recherche de notre problème qui permettra à un AED d'optimiser la charge de requêtes. Le système devra répondre aux questions suivantes que l'AED posera certainement lorsqu'il veut optimiser son ED :

- Quelles sont les techniques à choisir ?
- Quelles sont les structures d'optimisation à choisir ?
- Quelles sont les préférences ?
- Quel sera le taux de réduction de la charge?
-Quels sont les ressources nécessaires pour créer ces instances (espace disque)?

III .3 La démarche de la modélisation multi-agent (Les décompositions Voyelles)

La méthodologie VOYELLES **[**49**]** (également notée *AEIOU*) décrit le système par approche systémique du haut vers le bas (top-down) et approche est de bas en haut (bottom-up); l'accent étant mis sur le comportement des agents individuels. Cette méthodologie considère qu'un SMA est composé de quatre dimensions Agents,

Environnements, Interaction et Organisation auxquels il convient également d'ajouter une prise en compte des Utilisateurs. Elle est généralement employée pendant la phase d'analyse d'un problème pour identifier les sous-problèmes propres à chaque dimension, puis pour leur appliquer des modèles appropriés de résolution.

Cependant, dans des systèmes de taille importante, le sous-problème attaché à une dimension peut rester un problème complexe, lui-même composé de plusieurs facettes.

La modélisation de notre système consiste essentiellement en la définition de l'ensemble des agents, de leur architecture et du protocole de négociation conformément au modèle VOYELLES. Cette méthode consiste à déterminer les cinq composantes du SMA qui sont : A : Agents, E: Environnement, I : Les Interactions, O: l'Organisation plus (U) : Utilisateurs.

On modélise notre système avec deux visions : vision Marco (système), et vision Micro (centrée agent). Notre système est illustré dans la figure III-1.

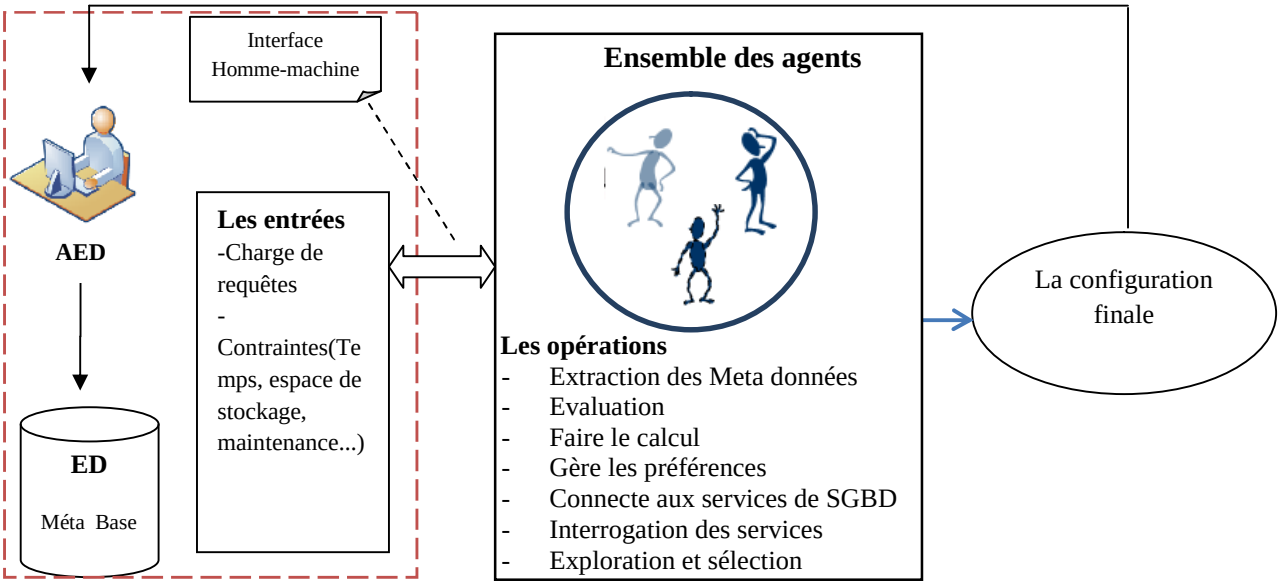


Figure III -1: Présentaion générale de notre systeme SMA

On présente dans ce qui suit la vision système et agent de notre solution.

III .3.1 **Vision système :**

La vision système regroupe les éléments suivants : les agents, l'architecture globale, les interactions, l'organisation et l'utilisateur du système.

III .3.1.1 ***Les Agents (A) :***

Qui sont les agents du système ? Notre système comporte cinq types d'agents : agent extracteur, agent préférence, agent évaluateur, agent sélectionneur utilisé dans le cas d'une sélection intra-technique et un agent négociateur utilisé dans le cas d'une sélection inter-technique.

Chaque agent dans cette approche produit des résultats qui peuvent être finaux (comme les index finaux sélectionnés) ou intermédiaires utilisés comme entrées pour d'autres agents (comme les index candidats).

- ✓ **Agent « Extracteur »**

L'agent EXTRACTEUR gère tout le flux d'informations et la mise à jour de la base de connaissance(BC). C'est un agent cognitif et modulaire. L'extracteur se connecte au SGBD (journal de requêtes) et la base de données pour extraire les requêtes fréquentes et les métas-données de schéma de l'entrepôt. L'extraction de la charge en entrée de notre approche s'effectue à partir du journal des requêtes exécutées sur les données de l'entrepôt. Cet agent analyse syntaxiquement les requêtes pour extraire les différentes structures d'optimisation de l'ED. Les statistiques représentent toutes sortes d'informations quantitatives nécessaires pour évaluer le coût d'exécution des requêtes, comme la cardinalité des tables, le nombre de valeurs distinctes de chaque attribut, etc. Ces informations sont stockées dans une BC bien structurée pour pouvoir être utilisées par les autres agents.

Les statistiques et les métas-données chargées par l'agent extracteur dépendent de l'algorithme de sélection utilisé par l'agent sélectionneur et les paramètres de modèle de coût utilisé par l'agent évaluateur pour réaliser la sélection d'une configuration finale.

- ✓ **Agent « Préférence»**

Il est intéressant de remplacer totalement l'expertise de l'AED par un agent préférence (dans le but d'une auto-administration). Cet agent s'occupe de gérer les souhaits et les préférences de l'administrateur ; il répond aux questions suivantes : pour cette charge de requêtes, quelles sont les techniques que je dois sélectionner pour le processus d'optimisation ?, quelles sont les contraintes initiales ? , comment repartir les ressources et les attributs ?

L'agent préférence permet de recommander des techniques d'optimisation : IJB, FH ou les deux en même temps. D'autres techniques peuvent être intégrées dans les versions futures de notre système. Il raisonne grâce à ses croyances, ses préférences et par sa capacité d'apprentissage, pendant qu'il travaille au fur et à mesure. L'évolution artificielle par le processus d'apprentissage apporte également une amélioration sur leur qualité de décision.

La gestion de préférence est basée sur un nombre de critères : Caractéristiques des attributs de prédicat *WHERE*, Type de requêtes, Fréquence d'accès…

- ✓ **Agent « Evaluateur »**

La sélection d'une structure d'optimisation doit tenir compte des limites imposées par le matériel, tel que l'espace disque réservé pour les index. Le rôle de l'agent évaluateur est d'estimer le coût de chaque structure d'optimisation et de s'assure si les

contraintes spécifiées par l'administrateur (espace de stockage et de maintenance, nombre de fragment, seuil de pénalité) sont vérifiées. L'évaluation de coût dans ce cas elle se base sur l'optimiseur de SGBD ou une fonction de coût mathématique. La deuxième tâche de l'agent évaluateur est la vérification des contraintes suivantes : (1) Le coût d'exécution de la charge de requêtes doit être réduit. (2) L'espace alloué pour les index ne dépasse pas une limite de stockage S. (3) Le nombre de fragment générés ne dépassent pas un seuil W (4) D'autres contraintes imposées par le SGBD comme le nombre d'index par table: nombre d'index autorise par table est limité (Oracle autorise 16 index par tables).

- ✓ **Agent « Sélectionneur »**

L'agent sélectionneur est un agent cognitif et modulaire, son rôle est de sélectionner une structure (index, vues matérialisées, fragment) élue à la sélection en utilisant un algorithme de sélection pour cette structure. Nous utilisons dans notre SMA un algorithme basé sur une technique de datamining. Il peut être facilement remplacé par un autre algorithme de sélection. Si la structure vérifié les contraintes spécifiées et apporte un gain, alors le résultat est inscrit et mis à la BC par l'agent extracteur. La sélection d'une structure d'optimisation se base sur une BC enrichie et mise à jour suite à ses interactions avec son environnement et les autres agents. L'algorithme de sélection exécuté par l'agent sélectionneur nécessite souvent un modèle de coût.

- ✓ **Agent « Négociateur »**

La tâche de l'agent négociateur est d'arriver à un accord commun entre les autres agents quand ils doivent coordonner leurs actions pour résoudre le problème de la ressource partagée et la répartition de l'espace. Quand il s'agit d'une sélection Inter technique, l'agent négociateur a un rôle important pour la répartition de l'espace de stockage entre deux techniques d'optimisation redondante, et gère le conflit dans le cas ou deux techniques partagent les structures d'optimisation. Dans ce cas l'agent négociateur négocie l'allocation de la ressource pour telle ou telle technique d'optimisation. Cet agent gère le système, maintient la cohérence, coordonne l'ensemble et joue aussi le rôle d'un négociateur.

L'attribution d'un attribut partagée x à la configuration *Config 1 ou config 2, peut* conduire à plusieurs alternatives résumées dans le Tableau III.2. En effet, l'ajout d'un attribut donné à la configuration *Config* peut améliorer de façon directe le coût des requêtes de la charge ou indirectement la deuxième technique optimisation.

$$Use_x(TO1,TO2)=\begin{cases}1 & \text{x utilisé par TO1 et TO2}\\ 0 & \text{x est utilisé par TO1 ou TO2}\end{cases}$$

Exemple11 :
Soit un ensemble de fragments faits (F_1, F_2, F_3, F_4, F_5, F_6) et les deux techniques d'optimisation l'IJB et la FH. La matrice *Use*(Q,F) montre l'utilisation ou non de fragment faits Fi par une requête Qj.

	F_1	F_2	F_3	F_4	F_5	F_6
Q1	**1**	0	0	0	0	0
Q2	**1**	**1**	**1**	0	0	0
Q3	0	1	0	**1**	0	**1**
Q4	1	1	1	0	0	0
Q5	0	0	0	**1**	**1**	**1**
Q6	**1**	0	0	0	0	0
Q7	0	0	0	0	0	0

Tableau III.1: matrice *Use* (Q,F)

Le coût de la charge de requêtes Q est calculé par l'utilisation de deux techniques FH et IJB indépendamment (Cost(Qj,FH) et simultanément (Cost(Qj,FH)$_{IJB}$. L'algorithme ci-dessous illustre ce calcul.

Algorithme : Impact de l'indexation sur la fragmentation

Pour chaque fragment Fi
  Bénéfice =0
  Pour chaque requête Qj
    Si *Use(Qj,Fi)=1* ***A*lors**
        bénéfice **+=** Cost(Qj/FH) - Cost(Qj/FH)ijb
    Si bénéfice >0 ***A*lors**
    Creér un index local sur Fi
   Fin
Sinon
     FH Seule
Fin.

Le gain d'un attribut *Ai* est calculé localement par apport à la configuration de chaque technique, et aussi par rapport aux deux configurations. Si la pertinence globale est supérieure à la pertinence locale on parle d'une amélioration coopérative sinon on dit que l'amélioration est concurrentielle.

Améliorer de façon coopérative veut dire que l'attribut doit être utilisé par les deux techniques afin d'apporter un gain à la charge de requêtes. Dans l'amélioration concurrentielle l'objet doit être utilisé par la technique de coût minimum. La sélection combinée des deux techniques, définies chacune sur son ensemble d'instance, permet de réduire le coût d'exécution des requêtes. Dans ce cas, il est intéressant de répartir les instances entre les deux techniques pour identifier pour chacune d'elles un espace réduit (voir [45]).

L'avantage de négociateur dans ce cas : La prise en compte de l'interaction c.-à-d l'impact de chaque technique sur l'autre afin d'améliorer la qualité de décision d'une façon globale.

Algorithme de prise en compte de l'interaction inter techniques (Algorithme 1)

Entrée :
A : ensemble d'attributs de sélection
Q : ensemble de requêtes
TO : ensemble des techniques d'optimisation
Sortie :
Mode d'amélioration
Bénéfice apporté
Début
Pour *chaque attribut partagé Ai* **faire**
Si $Gain_{Ai}$(Q/TO1, TO2) >Max {$Gain_{Ai}$(Q/TO1),$Gain_{Ai}$(Q/TO1)} **Alors**
Amélioration coopérative
Sinon
Amélioration concurrentielle
Finsi
FinPour
Fin

	Use_x(TO1 ,TO2)=0	Use_x(TO1 ,TO2)=1
Amélioration coopérative	Bénéfice à TO1	Bénéfice à TO2 et TO2
Amélioration concurrentielle	-	Min(bénéfice de TO1, benefice de TO2)

Tableau III.2 : Mode d'amélioration

III .3.1.2 Architecture du système

L'architecture du système est présentée dans la figure III.2.

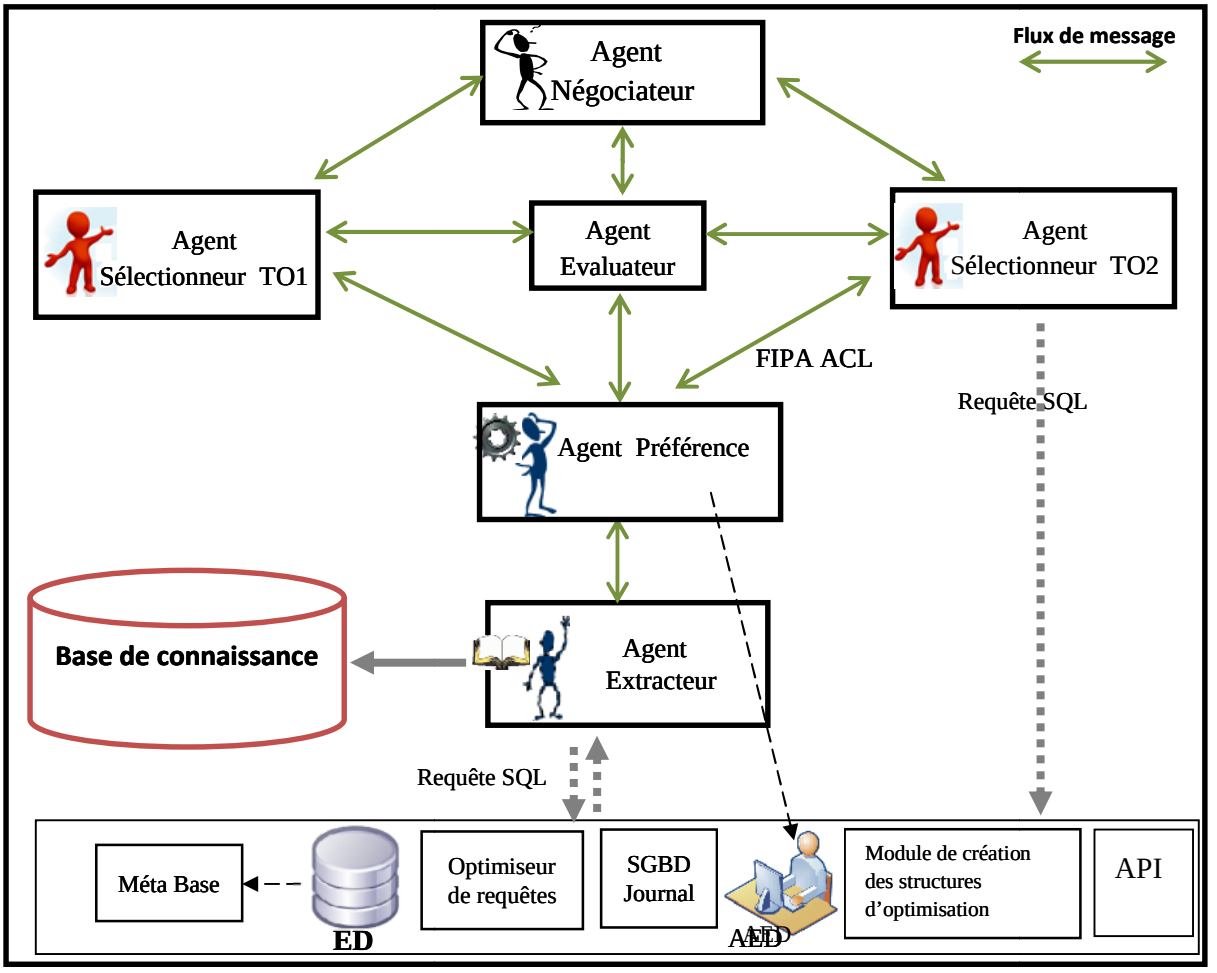


Figure III -2: Architecture de notre système

III .3.1.3 Les Interactions (I)

Nous présentons dans ce qui suit les interactions sous forme des scénarii entre les différentes entités durant la procédure d'envoi d'un message.

1. Variation de l'ordre d'exécution pour les agents sélectionneur :

Nous identifions trois scénarii pour l'ordre de sélection des techniques d'optimisation : TO1 puis TO2, TO1 puis TO2 et TO1 en parallèle avec TO2. Dans certains cas comme par exemple la sélection de index / vues matérialisées, la sélection simultanée permet d'avoir une gestion dynamique de l'espace de stockage, et la prise en compte de l'interaction.

4. Description de l'interaction entre les agents

Après la réception d'une demande d'optimisation de la charge de requêtes par l'AED, les agents sélectionneurs (IJB ; FH ; FH ; VM) signalent leurs présences ; l'agent extracteur charge toutes les données et les contraintes dans une BC bien structurée selon les techniques d'optimisation sélectionnées par l'AED. Pour la réaliser l'agent extracteur se connecte au service SGBD et à la base de données pour extraite les

métadonnées, les requêtes fréquentes, la fréquence d'accès aux attributs et les structures d'optimisation (Index, IJB, VM.FH, FV).

La sélection d'une structure d'optimisation nécessite une analyse syntaxique de la requête en étoile. Ces informations sont insérées dans une BC partagée entre les agents.

Pour extraire la configuration candidate à la sélection, chaque structure d'optimisation est envoyée à l'agent évaluateur pour la vérification avant de mettre la configuration initiale. Chaque structure qui ne vérifie pas une contrainte à priori (les contraintes spécifiées par l'AED comme un seuil de pénalité, condition à priori...) doit être supprimée de la configuration initiale.

L'agent préférence gère les préférences de l'AED ; il sélectionne les techniques d'optimisation impliquées dans le processus d'optimisation, envoie un message de diffusion vers les agents sélectionneurs concernés. Il intervient en amont pour repartir les attributs et les ressources entre les agents sélectionneurs.

L'agent négociateur intervient avant ou durant le déroulement de l'algorithme de sélection local par les agents sélectionneurs. L'agent sélectionneur sélectionne un objet et l'envoie à l'agent négociateur ; quand il s'agit d'une ressource partagée entre les deux techniques, l'agent gestionnaire préfère a quelle technique doit être alloué cet objet. Dans le cas d'une sélection des deux techniques redondantes, l'agent négociateur réalise une répartition dynamique de l'espace de stockage entre eux.

L'agent négociateur communique avec les agents sélectionneurs des techniques par deux messages Delete ou Save qui indiquent l'utilisation ou non de l'attribut. Quand un message Delete/ Save est reçu, par l'agent sélectionneur, ce dernier est renvoyé à l'agent extracteur pour mettre à jour la BC plus particulièrement la configuration finale.

L'agent sélectionneur applique un algorithme de sélection afin d'identifier les configurations possibles, chaque configuration est envoyée à l'agent évaluateur sous forme d'un groupe de messages afin d'évaluer leur coût, et envoyée à l'agent sélectionneur pour l'inscrire dans la BC; à la fin d'algorithme l'agent compare ces coûts et propose à l'AED la configuration de coût minimum. L'AED matérialise la configuration finale et l'intègre dans l'entrepôt de données.

Les interactions entre les agents sont illustrées dans le diagramme d'interactions. Un message représente dans ces diagrammes une information envoyée d'un agent vers un autre. Un groupe de messages inclut une série de messages échangés entre deux agents pour répondre à une tâche. La section suivante décrit l'interaction entre les agents pour répondre au besoin de l'AED.

5. Communication et négociation inter-agents

Pour modéliser des agents en coopération [50], on est souvent obligé de définir une suite d'échange de messages, c'est-à-dire une conversation d'agents. La communication entre ces agents est modélisée sous forme des diagrammes d'interaction d'UML.

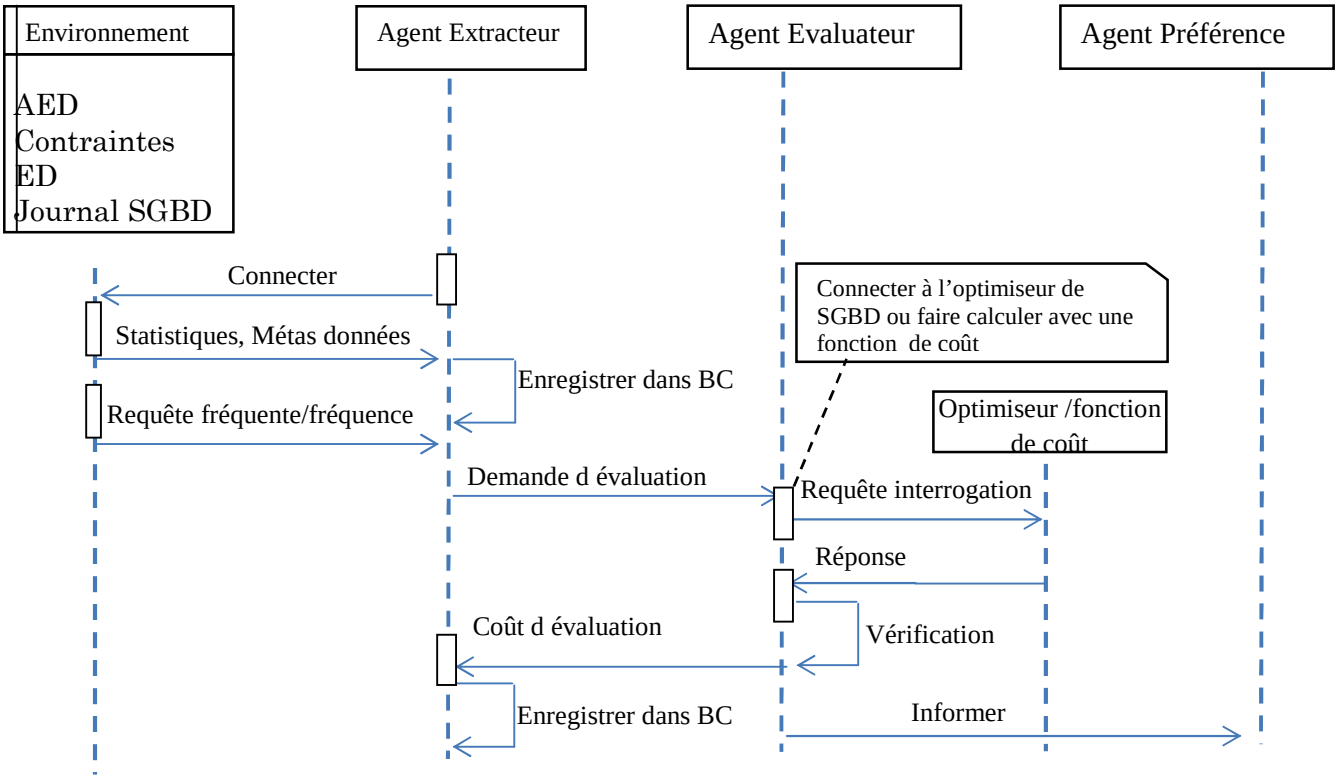


Figure III -3: Génération des instances candidates à la sélection

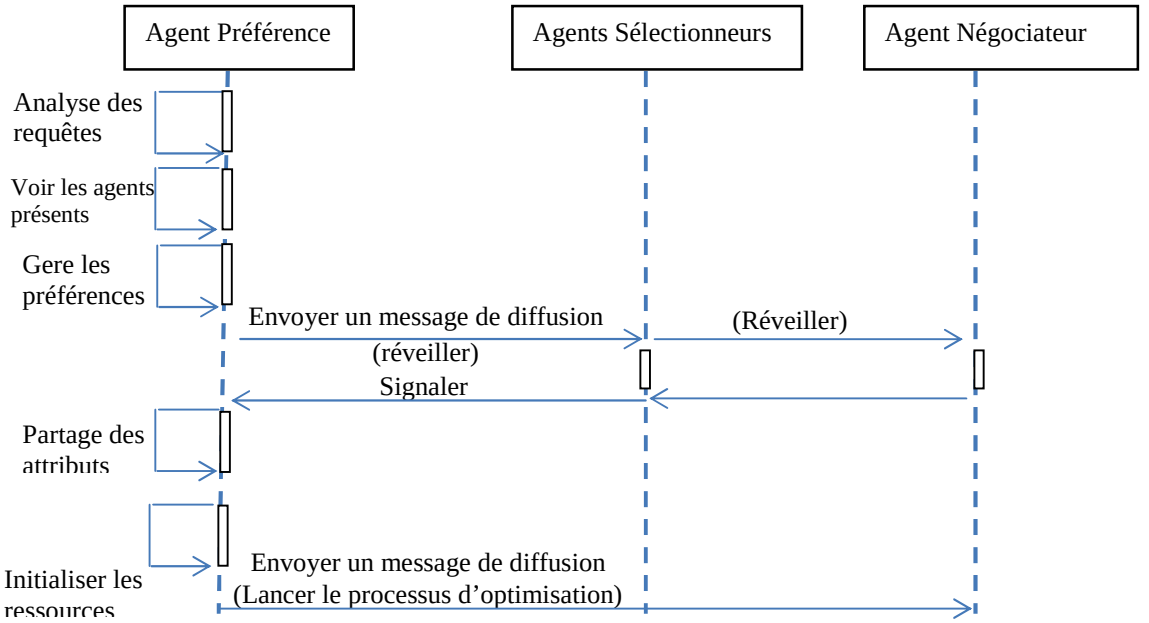


Figure III -4: Sélection des techniques d'optimisation

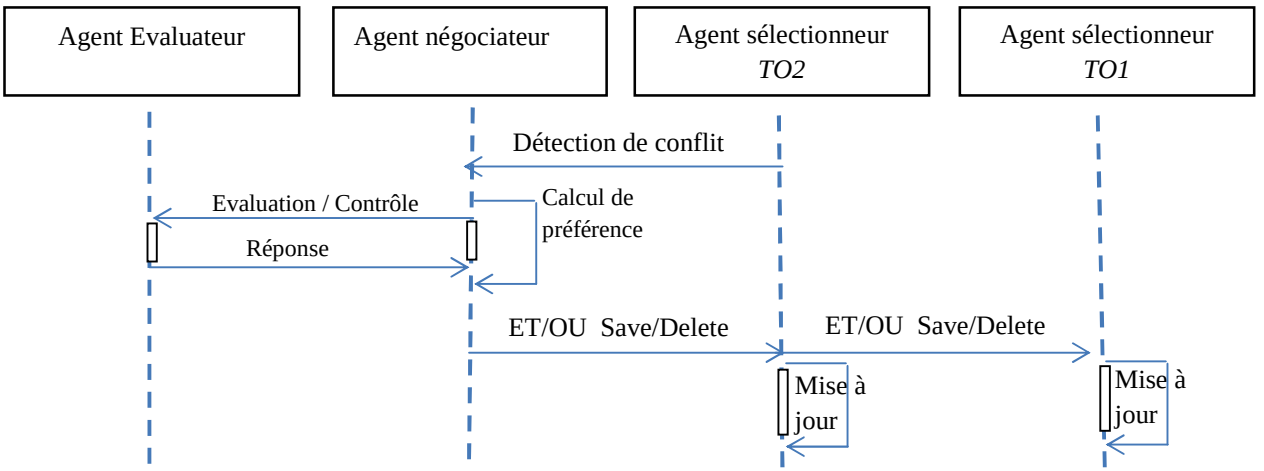


Figure III -5: Communication inter technique

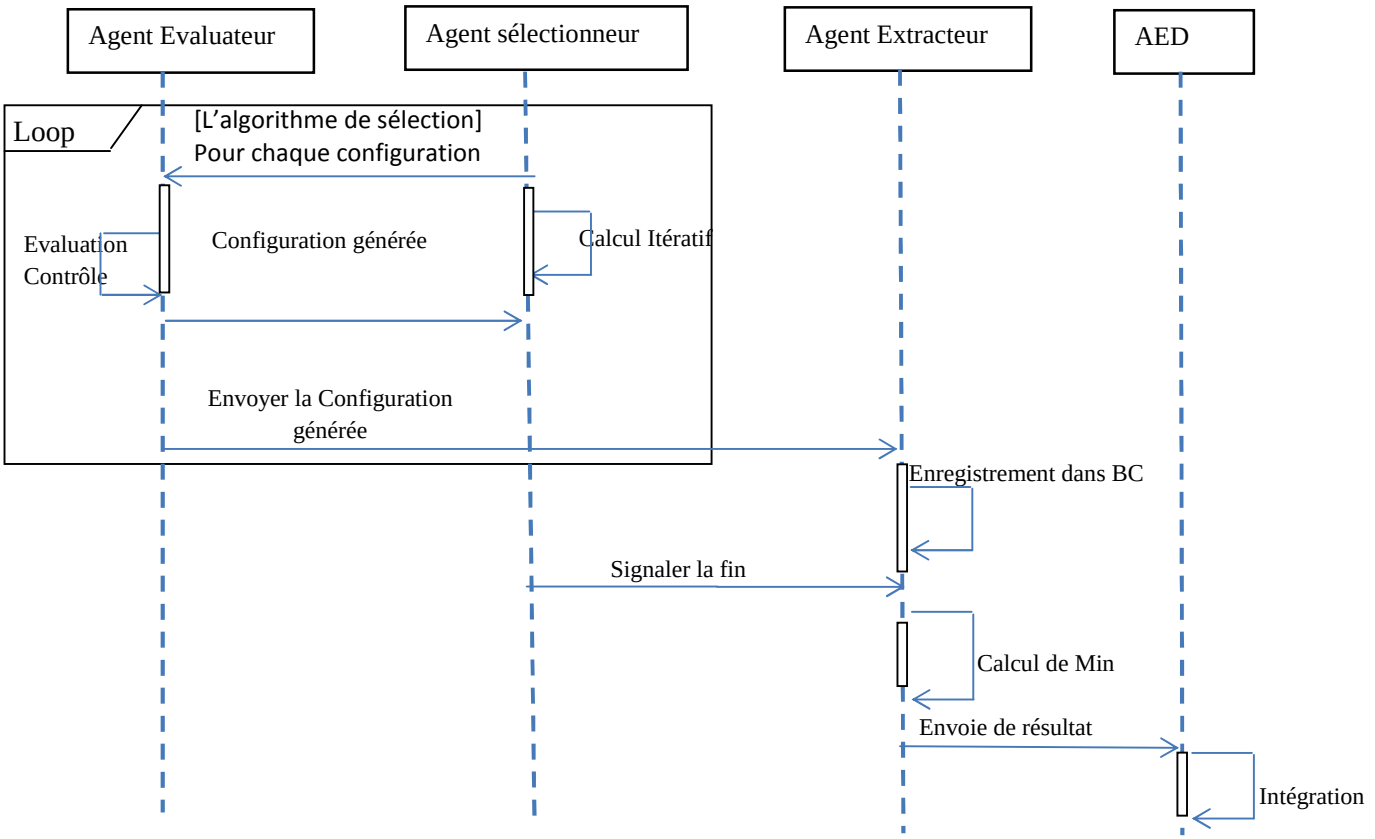


Figure III -6: Génération de la configuration finale

6. Moyen de l'interaction

Les moyens de l'interaction dépendent de la topologie de l'agent (cognitif / réactif) c.-à-d. que l'interaction entre ces agents est basée sur un langage de communication, dans le cas des agents cognitifs et via l'environnement dans le cas des agents réactifs.

L'interaction entre ces agents repose sur : ressource (BC), et l'envoi des massages (langage de communication ACL).

Les ressources désignent dans notre cas :

- Méta données extraites à partir de l'ED.
- Espace disque
- Structures d'optimisation (vues, index, fragments)
- Services du SGBD (journal, optimiseur.)

7. Type d'interaction

L'interaction dans notre cas est divisée en deux types : Inter technique et Intra technique**.**

L'interaction est intra technique dans le cas où les agents collaborent localement pour sélectionner une seule technique, dans ce cas la communication est directe (envoi de message). Dans le cas d'une sélection multiple, on parle d'une interaction inter technique. L'agent négociateur est intéressant, dans le cas où les agents sélectionneurs sont fortement liés autrement dit si les techniques d'optimisation nécessitent beaucoup de communications entre-eux (mesure de degré de similarité). Pour un système où les agents sont hétérogènes l'agent négociateur n'est pas intéressant du moment où les agents mettent à jour leur représentation à chaque modification ; cette mise à jour peut être inutile dans ce cas où les agents n'ont pas besoin de communiquer.

Chaque agent qui peut être intégré au processus d'optimisation n'a besoin que de connaître le négociateur et de se faire connaître auprès de lui. Tout agent quittant le SMA, n'a qu'à signaler son départ au négociateur.

8. Situation d'interaction :

Les agents peuvent être en compétition sur les ressources du système comme l'espace de stockage et les attributs de sélection. On parle ici de compétition lorsqu'une forte similarité est identifiée entre ces techniques. Elle exprime la volonté d'un agent d'atteindre un but ou d'effectuer une action localement sans prise en charge l'impacte sur l'autre technique, alors l'agent négociateur est dans le besoin de maintenir la cohérence de la solution globale.

Par exemple, le sélectionneur d'index et le sélectionneur de vues raisonnent en terme local (maximum d'espace, maximum d'attributs), l'agent Négociateur raisonne au

niveau global c.-à-d. en terme de but " optimiser la charge de requêtes". Il a pour but déprendre en compte l'impact inter technique.

Le Tableau III.3 montre le classement de situation d'interaction [42] entre les différents agents, selon les buts, les ressources et les compétences des agents

- Buts compatibles ou incompatibles
- Ressources suffisantes ou insuffisantes
- Compétences suffisantes ou insuffisantes
- Type de collaboration

Remarque : on dit que l'objectif est compatible si l'objet de l'agent *i* aide à réaliser l'objectif de l'agent *j*. La ressource est suffisante si un agent à toutes les ressources pour réaliser le but et la compétence est suffisante si l'agent a la capacité d'atteindre le but.

Agents	**But**	**Ressources**	**compétences**	**Situation**
Intra technique	Compatible	Insuffisantes	Insuffisantes	Collaboration simple cordonnée
Sélectionneurs TO1, TO2...TOn	Incompatible	Insuffisantes	Insuffisantes	Conflits collectifs pour des ressources
Négociateur - sélectionneur	Incompatible	Insuffisantes	Insuffisantes	Collaboration cordonnée

Tableau III.3 : Situation de l'interaction entre les agents

Jusqu'à ce niveau nous avons défini la vision système, dans la section suivante, on va définir la vision centrée agent.

III .3.1.4 L'Organisation (O)

L'organisation est définie à priori sous forme d'un ensemble de règles, l'objectif est de déterminer les limites, les rôles et les obligations de chaque agent ainsi que les relations entre eux.

III .3.1.5 Les Utilisateurs (U)

Le décideur et l'AED sont les utilisateurs du système. Ce dernier se charge de la maintenance, l'administration et l'optimisation de l'ED. L'intervention de l'administrateur est primordiale pour effectuer certaines tâches comme l'identification de la charge de requêtes sur laquelle il serait intéressant d'effectuer le tuning, les techniques d'optimisation utilisées, les recommandations intéressantes à matérialiser.

Il est intéressant d'enrichir notre système par l'interaction avec l'administrateur qui donne la possibilité à l'administrateur de personnaliser sa conception physique et d'utiliser son expérience afin d'améliorer la qualité de la solution finale.

III .3.1.6 L'Environnement (E)

L'objectif global de l'environnement est d'exercer un ensemble d'activités, dont l'exécution est temporellement liée tout en optimalisant une fonction objective (réduire au minimum le coût de la charge avec la contrainte de stockage et de maintenance)

L'environnement du SMA est constitué par:
- L'entrepôt de données contenant les informations concernant les acteurs, l'historique des requêtes, le journal de SGBD, etc.
- Les différentes structures d'optimisation (index, vues matérialisées, fragment Horizontaux et verticaux).

Le système que nous avons décrit est un système composé d'agents communiquant dans un environnement. Son environnement peut être modélisé par une représentation topographique représentant des objets passifs, des services et les ressources que manipulent les agents. La figure III.7 ci-dessous présente une représentation topographique de l'environnement SMA.

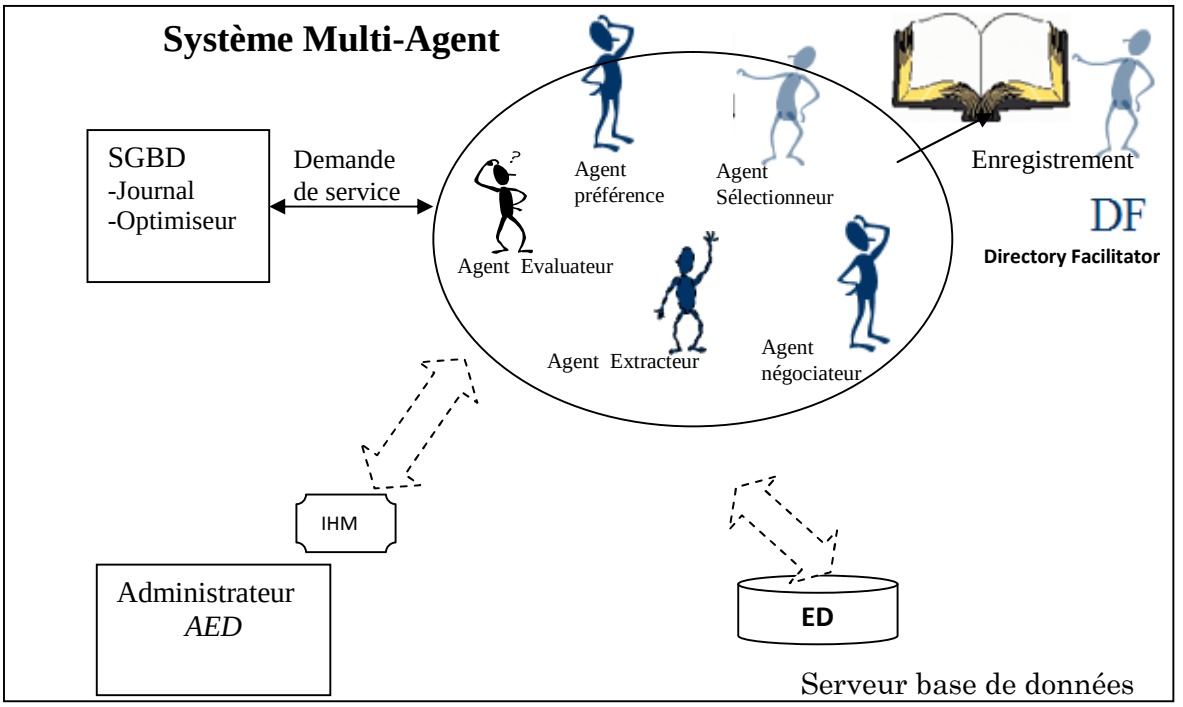


Figure III -7: Représentation topographique de l'environnement du SMA

III .3.2 Vision centrée agent :

Après avoir présenté les différents messages créés pour la communication entre les agents, on présente maintenant les agents. Chaque agent est décrit premièrement par sa structure à savoir, type d'agent (sans ou avec connaissance au sens symbolique), perception, action. Puis on détaillera le comportement, l'organisation, et leur environnement de perception.

La méthodologie voyelles (AEIO) [49] (Figure III -11) se décompose en quatre parties :

La facette A permet de représenter l'ensemble des fonctionnalités du raisonnement interne (planification par exemple) de l'agent.

La facette E permet de définir l'ensemble des capacités de perception et d'action d'un agent sur son environnement.

La facette I permet de définir l'ensemble des interactions avec les autres agents (protocoles de communication par exemple).

La facette O est liée aux capacités de structuration et de gestion des relations des agents entre eux.

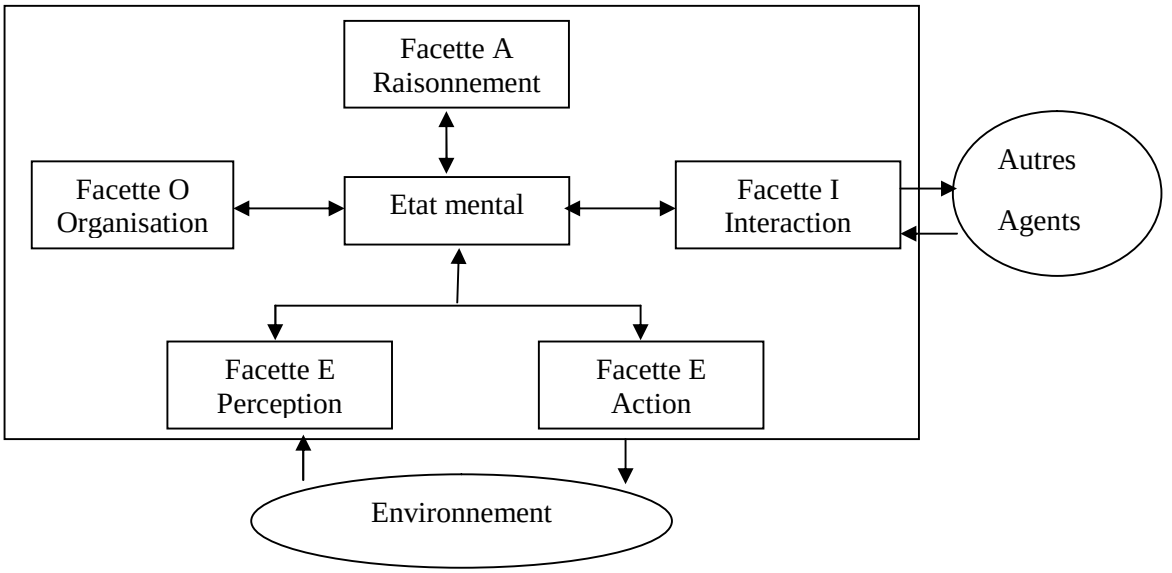


Figure III -8: Facette AEIO au sein d'un agent [57]

III .3.2.1 Choix de modèle fonctionnel

Le choix d'un modèle d'agent se fait essentiellement en fonction des besoins exprimés au cours de la phase d'analyse. Notons, en premier lieu, que tous les agents disposent d'un module de communication (boîte où les agents reçoivent leurs messages).

Pour notre système, le modèle d'agent que nous avons recensé est :

- Soit **cognitifs** : du fait de sa capacité à apprendre et à raisonner.

-Soit **modulaire** : il possède une architecture modulaire; chaque module est spécialisé dans un aspect du fonctionnement de l'agent. Les modules interagissent entre eux (Figure III -9:).

.

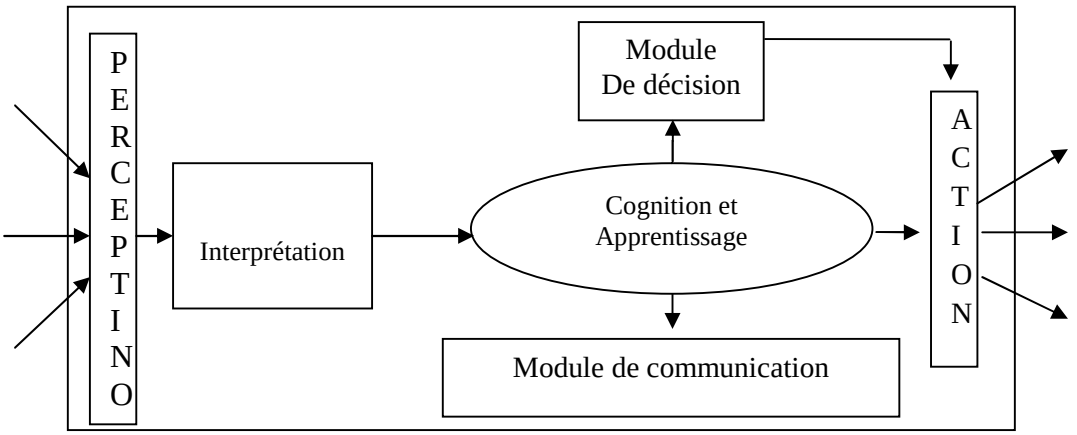


Figure III -9:Structure de l'agent

- ✓ **Module d'interprétation**

Pour pouvoir analyser des événements traduits par des messages reçus.

- ✓ **Module cognition croyances et Apprentissage**

Les croyances sont réalisées en mettant à jour les informations au niveau de la BC suite à la perception de l'environnement (réception de messages).
On peut envisager un apprentissage de l'Agent préférence relativement au type de requêtes, type d'attributs. Par exemple, après un certain nombre d'interactions, l'Agent préférence réussira à construire une vision globale sur l'ensemble des préférences pour sélectionner les différentes techniques d'optimisation.

- ✓ **Module de décision**

Permet à l'agent de choisir entre différents plans dont il dispose pour atteindre ses objectifs. Un plan est une séquence d'actions que l'agent peut prendre quand un événement surgit. Il sera déclenché lorsqu'un ensemble de conditions est vérifié.

- ✓ **Module de communication** : lui permettant de communiquer avec les différents agents.

III .3.2.2 La perception/Action des agents

Chaque agent de notre système reçoit l'information et génère une action. La Figure III-13 montre la perception /action de chaque agent. La perception c'est une porte entre le monde et l'agent qui lui donne accès à une certaine « conception de monde » dans laquelle ses raisonnements et ses actions sont significatifs. C'est par la perception que l'agent acquiert des informations sur le monde pour lui permettre d'élaborer son action en poursuivant ses buts.

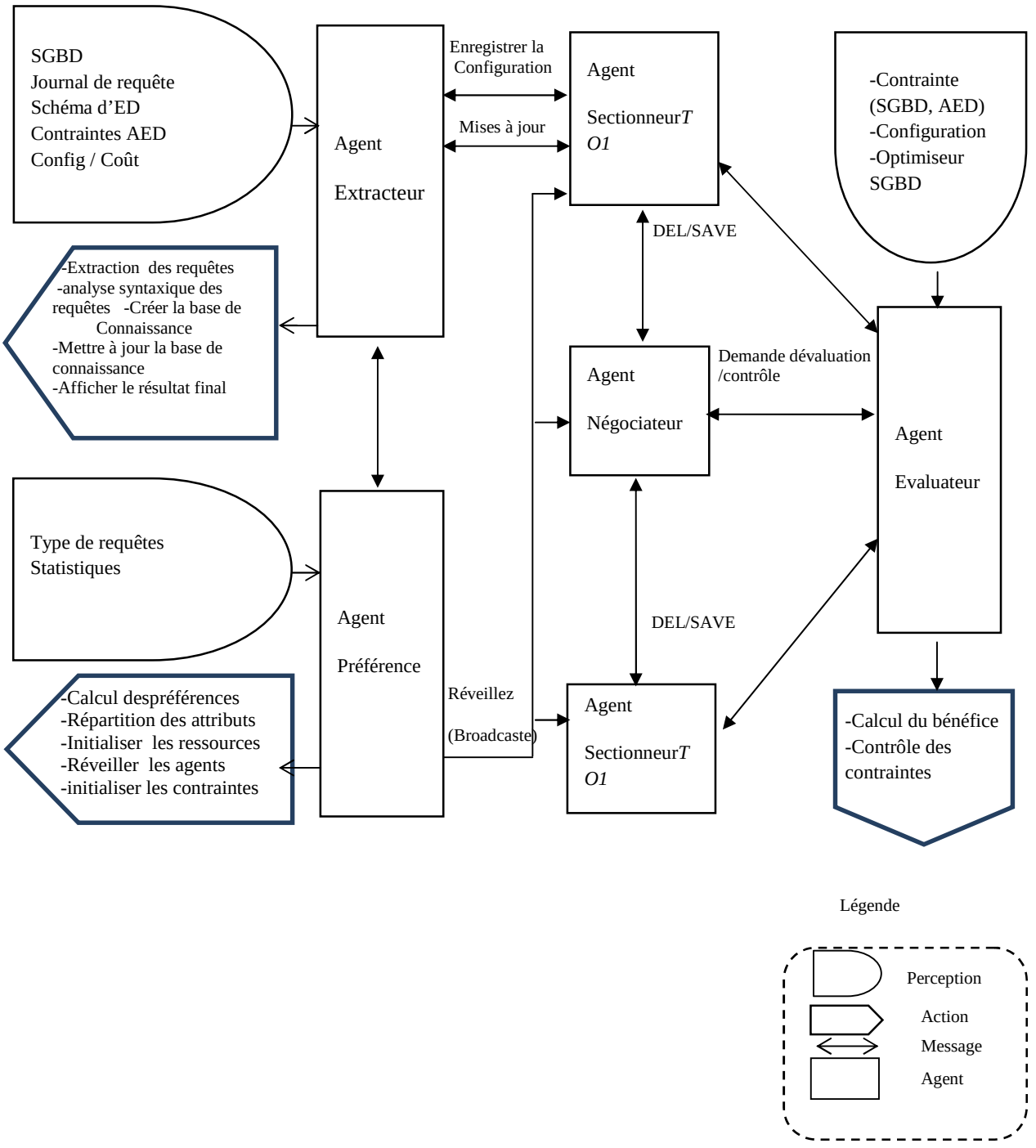


Figure III -10: Perception/Action des agents

III .3.2.3 Comportement des agents

- ✓ **Comportement de l'agent extracteur**

Les actions réalisées par cet agent sont alors :

- Fournir les informations nécessaires pour les autres agents;
- Sauvegarder les solutions trouvées par l'agent sélectionneur;
- Mettre à jour les informations de la BC (en cas d'ajout d'une nouvelle requête, modification de fréquence de requête);

- ✓ **Comportement de L'agent préférence**

Les tâches réalisées par cet agent sont alors :

- Analyse de la charge
- Détermination des techniques d'optimisation
- Initialiser les ressources ente les techniques ; Espace disque ; nombre de fragment
- Répartition d'attributs entre les techniques :
- Fournit un ensemble de recommandations sous forme de règles conditionnelles
- Envoyer un message de diffusion pour réveiller les agents sélectionneurs inclus dans le processus de l'optimisation

Répartition des attributs entre les techniques :

Soit A = *{a1, a2,…..an}* un ensemble d'attributs extraits à partir d'une charge de requêtes Q. Le partage des attributs de sélection entre deux techniques d'optimisation TO_1, TO_2 est effectué par classification de l'ensemble d'attributs en deux ensembles. Deux scénarios peuvent être considérés.
Cas1 : Si deux techniques TO1 et TO2 sont faiblement dépendantes alors l'intersection est vide, on parlera d'une sélection isolée.

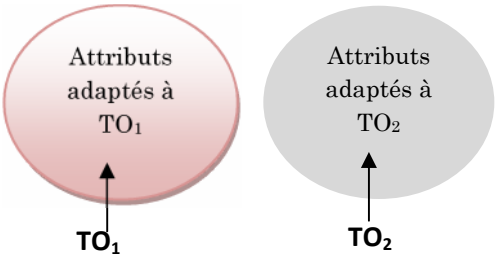


Figure III -11: Sélection isolée (séquentielle)

Cas1 : Si deux techniques d'optimisation TO1 et TO2 sont fortement dépendantes alors l'intersection est non vide, on parlera d'une sélection combinée.

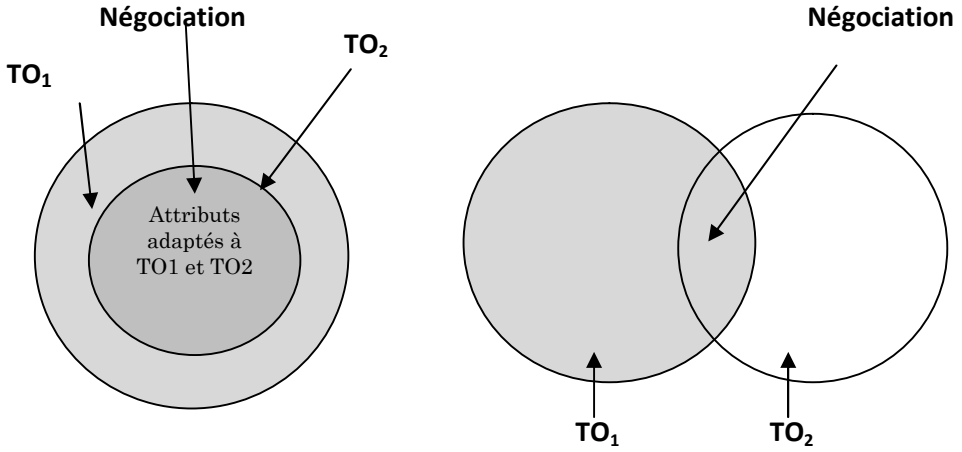


Figure III -12: Sélection combinee

- ✓ **Comportement de l'agent sélectionneur**

Pour chaque technique d'optimisation faisant intervenir un agent sélectionneur (IJB, FH, FV, Vues), la détection de la possibilité de coopération consiste en un raisonnement sur cette présentation. Il faut donc que l'agent dispose de certaines connaissances et capacités, pour pouvoir participer à la résolution des problèmes.

Nous regroupons celles-ci dans un module présent dans chaque agent que nous baptisons 'Mode d'emploi'. La Figure IV -12: montre l'architecture de l'agent que nous avons proposée. La modification de ce mode d'emploi est dépendante de la technique d'optimisation sélectionnée. Pour toute modification d'un agent, ce dernier informe tous ses collègues qui mettent à jour leurs représentations ce qui modifie le mode d'emploi.

Mis à part le mode d'emploi, l'agent possède un *module de perception* lui permettant de percevoir son environnement et mettre à jour ses représentations internes de l'environnement et des autres agents.

Les actions réalisées par cet agent sont alors

- Faire le calcul
- Envoyer le résultat de calcul
- Signaler la fin

- ✓ **Comportement de l'agent négociateur**

Le négociateur est un médiateur entre les agents sélectionneurs. Chaque sélectionneur lui envoie alors sa description et le négociateur prend en considération le groupe des agents qu'il peut intégrer au processus d'optimisation. Le négociateur suit alors le comportement de n'importe quel agent sélectionneur. Ilpossède une architecture modulaire qui contient:

- Perception des messages envoyés par les agents sélectionneurs de chaque technique.
- Action qui représente la décision et le choix de préférence (une ressource préférable peut être utilisée par telle technique)
- Un modèle d'interprétation lui permettant de communiquer les différentes techniques.
- Mode d'emploi pour s'adapter aux techniques d'optimisation choisies par l'AED.

Les actions réalisées par cet agent sont alors :

- Assurer la cohérence des interactions (contrôle)
- Gérer les conflits entre agents
- Coordonner l'exécution des agents

- ✓ **Comportement de L'agent évaluateur**

Les actions réalisées par cet agent sont alors :

- Faire appel l'optimiseur de SGBD

- Evaluation du coût d'une structure d'optimisation par une fonction mathématique
- Vérification des contraintes

III .4 Nouvelle démarche de sélection d'index et de fragmentation, dirigée par notre architecture SMA

La sélection des deux techniques d'optimisation IJB et FH s'effectue à partir des attributs de sélection extraits de la charge de requêtes. Les deux structures sont donc concurrentes sur la même ressource représentant l'ensemble des attributs de sélection définis sur les tables de dimension.

D'abord, en tant que premier objectif de ce travail nous avons modélisé notre système par un Système Multi-agents afin d'identifier les différents agents avec leurs interactions (voir la section précédente). Le deuxième objectif de ce travail est de mettre en place une approche de la sélection des structures d'optimisation (IJB et FH). L'approche de résolution que nous préconisons est basée sur le Système Multi-agent où l'algorithme de sélection utilisée : l'algorithme génétique (AG) pour la sélection de la FH et les IJBs améliorés par l'interaction entre les deux techniques d'optimisation. Le nombre important des choix rend les tâches de conception physique et tuning de plus en plus compliquées. Le développement d'outils afin de résoudre ce problème d'optimisation doit reposer sur la réduction de l'espace de recherche puisque nous somme devant un problème de complexité énorme.

Afin de simplifier la complexité de problème de sélection de FH & IJB. Nous proposons une nouvelle approche de sélection combinée d'index de jointure binaires et schéma de fragmentation qui se base sur le paradigme SMA.
Nous présentons, dans ce chapitre, le principe de cette nouvelle démarche. Nous commençons par montrer la motivation de notre choix de sélection par analyse des approches de sélection combinées existantes. Par la suite, nous détaillons la mise en œuvre de cette nouvelle démarche avec spécification des algorithmes utilisés.

Pour mieux comprendre notre contribution, nous allons projeter ce problème d'optimisation sur notre système SMA proposé précédemment.

III .4.1 Motivation

Dans nos travaux, nous nous intéressons à la mise on œuvre d'une nouvelle démarche d'élagage de l'espace de recherche de la FH et l'IJB dirigée par le SMA, cela pour les motivations suivantes :

- La FH et IJB sont similaires du fait quelle permettent d'optimiser les requêtes de jointures en étoiles avec sélection sur les dimensions.
- Ces deux techniques sont définies sur un même ensemble d'attributs de sélections.

- Dans le contexte d'entrepôts de données, les attributs dimensions servent à effectuer des analyses décisionnelles. Le nombre d'attributs peut être du nombre de 100, 200 attributs ou plus. Par conséquent, définir un schéma d'optimisation sur un tel ensemble d'attribut devient une tâche difficile.
- Peu de travaux se sont intéressés à la mise en œuvre de démarche de sélection combinée de FH et IJB. Rappelons que dans la littérature on trouve les travaux [1,39] qui traitent la sélection de fragmentation horizontale et d'IJB d'une manière combinée. Dans le contexte de l'ED un travail porte sur la partage de ces attributs sélection, entre index et fragmentation, en se basant sur une classification par l'algorithme de datamining « k-means »[32]. Bellatarche et Boukhalfa [1] ont proposé une approche de sélection simultanée basée sur les dépendances entre la FHD et les IJB. Elle consiste à sélectionner un schéma de FH, ensuite sélectionner une configuration d'IJB en prenant en compte le schéma de fragmentation sélectionné et les requêtes non bénéficiaires du processus de fragmentation. La sélection d'un schéma de FH avant celle des IJB permet d'élaguer l'espace de recherche des IJB.

Dans nos travaux, nous nous sommes inspirés principalement des deux travaux de [32] pour la classification des attributs par la technique de datamining K-means et [1]pour la sélection multiple d'un schéma de fragmentation et d'index de jointures binaires dans les entrepôts de données.

Notre travail porte aussi une amélioration sur les travaux de [1].Dans ce travail les attributs exclus par la technique de fragmentation ne doivent pas tenir par la technique de l'indexation qui n'est pas toujours le cas ; autre chose les paramètres fixes par l'AED influencent sur le mode de sélection des techniques (aspect dynamique) .Notre approche doit donc prendre en compte cet aspect. Nous allons répondre à cette question en prenant en compte les paramètres de l'AED et l'interaction par la mise en place les agents préférence et négociateur pour voir leur apport pour la prise en compte d'interactions complexes. Enfin nous discutons l'intérêt d'utiliser les SMA dans ce contexte dans la section suivante.

III .4.2 Exemple de Motivation

Soit un ED représenté par la table de faits Vente et la table de dimension Client. Considérons le scénario de la FH. La table « Client » avec une FH primaire sur l'attribut « Ville » en clients Alger OR Blida*, clients* Boumerdes *et clients Autre.*

La fragmentation est réalisée par opération de restriction comme suit :

-Fragment 1 = Client1=$\sigma_{\text{ville=Alger ou Blida}}$ =(Client)
-Fragment2 = Client1= $\sigma_{\text{ville =Boumedes}}$ =(Client)
-Fragment3 =Client3 = $\sigma_{\text{ville=Autre}}$ =(Client)

Le résultat de la sélection de la fragmentation horizontale est présenté dans la figure IV -15.

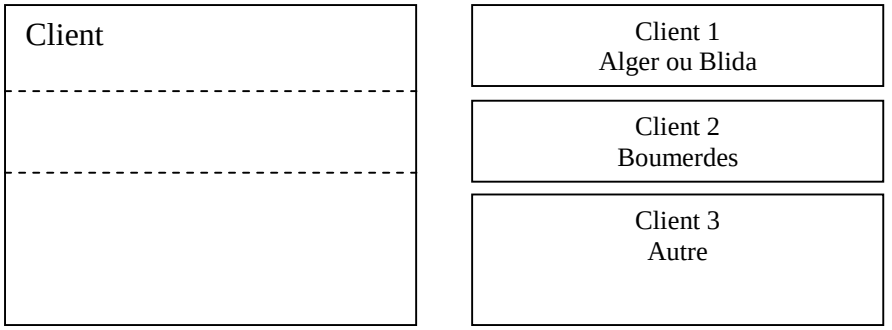


Figure IV -13: *Fragmentation primaire*

La table de Vente, contenant la clé étrangère vers Client « Id_C », est fragmentée par une FH dérivée suivant la FH primaire de « Client », cette fragmentation est obtenue par semi- jointure comme suit :

- Vente1= Vente ⋉Client1
- Vente2= Vente ⋉Client2
- Vente3= Vente ⋉Client3

Les résultats de FH dérivée sont illustrés dans la Figure III -14:

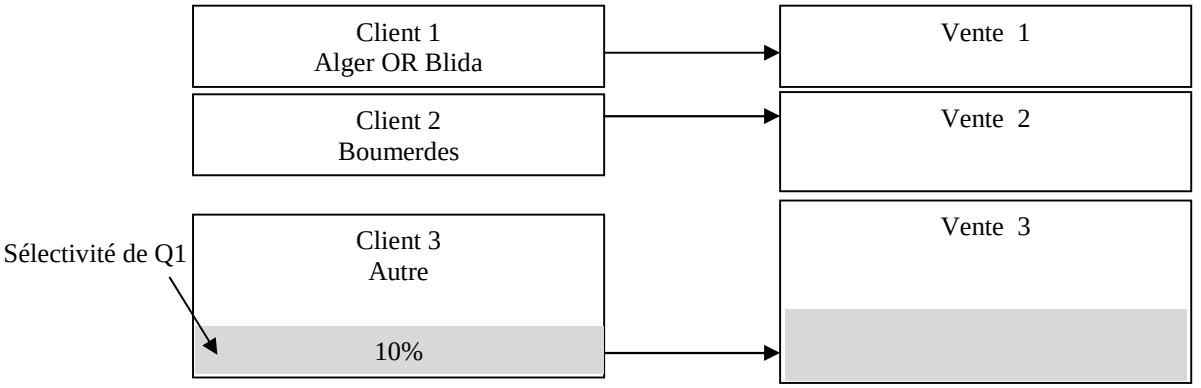


Figure III -14: Fragmentation dérivée

Considérons la requête Q1 suivante qui permet de calculer, pour chaque client de la ville Tiaret, la quantité totale de produits vendus.

```
SELECT Nom_Client, Sum(Quantité)
FROM Vente V, Client C
WHERE V.Id_C=C.Id AND C.Ville='TIARET'
GROUP BY Nom_Client
```

Les tuples sélectionnés par une requête sont en principe distribués sur le fragment de données Client3. Cependant si les tuples sélectionnés sont concentrés sur quelques blocs de données alors l'utilisation des index peut améliorer les performances des requêtes dont la sélectivité est grande.

On définit la *sélectivité* d'un prédicat comme étant le pourcentage de tuples qui satisfont ce prédicat.

Le problème qui se pose est de savoir à partir de quelle valeur de sélectivité l'AED considère l'indexation ou non des fragments faits? Pour résoudre ce problème, nous donnons la possibilité à l'AED de fixer un seuil pour lequel il est intéressant de créer un index sur le fragment ou non. Par exemple, si une requête retourne moins de 4% des tuples de la table (sélectivité inférieure à 4%) alors la fragmentation seulement sur cet attribut peut ne pas être suffisante; donc il est souhaitable d'indexer les attributs appropriés du prédicat.

Supposons que 100 blocs de données sont alloués au fragment. Si les données recherchées sont concentrées dans les blocs 1 et 2 alors l'utilisation des index peut améliorer considérablement les performances. En effet si on utilise la fragmentation seule, on doit accéder aux 100 blocs de données alors qu'en utilisant les index, on accède à seulement 2 blocs de données.

a) Chaque bloc de données alloué au fragment devrait en principe contenir un minimum de tuples. Cependant si les blocs contiennent moins que ce minimum alors l'utilisation des index peut améliorer les performances des requêtes dont la sélectivité est grande.
b) Indexer les attributs dont les valeurs sont distribuées uniformément sur la table de données. On peut estimer le pourcentage de tuples ayant des valeurs distinctes pour un attribut donné à l'aide de la sélectivité.

Dans ce cas, la fragmentation seule sur l'attribut ville, l'exécution de cette requête implique le chargement de la totalité de fragment, si on indexe sur cet attribut (seules les ventes de clients de Tiaret sont chargées). De plus, les opérations de jointures et de sélections sur dimensions sont précalculées par chaque structure d'optimisation. En résumé, les deux techniques IJB et FH doivent être définies sur le même attribut dimension, et permettent de réduire le coût de jointures en étoile entre table de fait et tables des dimensions avec opérations de sélection sur ce fragment.

Dans notre démarche de sélection, nous considérons d'autres scenarii

- Intégrer les contraintes de l'espace de stockage des index et le seuil de fragments faits générés dans le choix de mode de la sélection des techniques. Si l'espace est *réduit* et que le seuil est *grand*, favoriser la fragmentation, dans le cas contraire favoriser les index.
- Intégrer la dynamique de notre solution lors de la phase de sélection comme: Elagage si nombre d'attributs exclus par la FH est important, exploiter l'espace de stockage des IJB spécifié par l'AED si ne pas consommer totalement.

Dans cette section nous détaillons le déroulement du processus de sélection de l'IJB et la FH par le SMA.

III .4.2 L'Algorithme SMA

Dans cette section, nous présentons le principe général de notre algorithme SMA qui illustre le principe de notre solution. L'algorithme montre les entrées, les sorties, et les

hypothèses considérées. Chaque agent dans cette approche produit des résultats qui peuvent être finaux (comme les index finaux sélectionnés) ou intermédiaires utilisés comme entrées pour d'autres agents (comme les index candidats, coût d'une structure d'optimisation etc.).

Le déroulement de l'algorithme SMA de sélection des IJB et FH est présenté comme suit :

Algorithme SMA

AED

Initi(S) */* initialiser l'espace de stockage des index S*

Initi(W) */* initialiser le nombre de fragments maximum*

Initi(λ) */* initialiser le seuil pour lequel une requête est considérée bénéficiaire du processus de fragmentation.*

Initi(τ) */*initialiser valeurde la sélectivité de la requête préférable pour que le négociateur préfère l'indexation ou non de fragment.*

Agent extracteur

1. Extraction() */*Extraction requêtes fréquente& statistiques &paramètre de modèle de coût*
2. CréerBC() */*Etablir la base de connaissance*

3. *Si* Modification(ED) *Alors*
   Réveillez () **/* Réveillez les agents de système**
4. Mise à jour () */* mettre à jour la base de connaissance*

Agent préférence

1. Analyse-Charge()
2. Analyse (W, λ) */*Calcul de préférence*
3. Mode-Select () /*Choix de mode de sélection des TO : IJBSEUL ,FHSEUL,IJB&FH
4. Répartition(Attributs) /* répartition des attributs en deux classes ClasseIJB , ClasseFH
5. Réveillez (Agents IJB-FH)

Agent FH

Appel1() / Appel API (*JGAP*)*

*Appel2 ()/*Appel Négociateur si un Attribut est exclus par FH*

*Appel3 ()/*Appel Négociateur pour optimisation de la jointure entre un fragment et la table de fait*

Agent IJB

Appel4() / Appel API (*JGAP*)*

Appel5() / Appel Négociateur si un Attribut est exclus par IJB*

Appel6() / si l'espace disque non consommé totalement*

Agent Evaluateur */*Evaluer le coût*

Appel1 : Appel4 :

1. *Get* (structure)/**Reçoit la structure d'optimisation*
2. Calcul () /**Calcul de coût de la structure*
3. *Cost* ()/**Envoyer le résultat*

Agent négociateur

Appel 3 :
Si *disponible* (S) **alors**
CréerIndexlocal() /**créer un index sur le fragment*
Finsi.

Appel2 () : */*Attribut exclus par l'agent FH*
Pour i=1 à card(CI)
Si$R_{/ConfigIJB}(Ai)$<$R_{/ConfigIJB}(Attr - Exclus)$Alors /* Ratios profit/stockage
Drop(Ai)
CIJB=CIJB Attr-Exclus
Finsi

Appel6 () :
Si *disponible* (S) **alors**
CréerIndexlocal() /**créer un index sur le fragment*
Finsi.

III .4.3 Déroulement du processus de sélection par le SMA

Ce travail prend en considération les dépendances entre la fragmentation horizontale et l'IJB. En conséquence, il offre une sélection combinée. Deux principales étapes caractérisent cette approche, l'optimisation locale et l'optimisation globale qui prend en charge l'impact inter technique. L'optimisation locale permet de sélectionner un ensemble des IJBs et des FHs par les agents sélectionneurs. L'optimisation globale vise à minimiser le coût d'exécution total en considérant les deux configurations avec la réécriture des requêtes sur le nouveau schéma de l'ED.

Nous détaillons dans cette section le processus de sélection de FH&IJB dirigé par le Système Multi-agent proposé précédemment.

IV .4.4.1 Paramétrage des algorithmes de sélection

L'administrateur trouve la nécessité d'être assisté afin d'accomplir la tâche d'optimisation des requêtes décisionnelles. Il règle les paramètres physiques comme la mémoire qui constitue une opération de tuning importante, les paramètres utilisés dans chaque algorithme, etc. Cette administration repose essentiellement sur les connaissances de l'administrateur. Il fait partie de notre système SMA par l'interaction et la validation des recommandations générées par le système via une interface homme-machine. Il commence par l'initialisation des contraintes et les paramètres de l'algorithme de sélection comme le nombre de fragments exigé W et l'espace de stockage des index disponible S.

IV .4.4.2 Extraction des données

L'agent extracteur accède à l'entrepôt et obtient automatiquement la charge de requêtes à partir du journal de transactions du SGBD. Chaque requête est

caractérisée par sa fréquence d'accès ainsi que les prédicats de sélection qu'elle utilise. L'entrepôt de données, objet de ces requêtes, est modélisé par un schéma en étoile composé d'une table des faits F et un ensemble de tables de dimensions D = {D1,D2, · · · ,Dn}.

Cet agent se base sur la structure des requêtes et l'ED pour déterminer les statistiques pouvant être utiles pour l'indexation ou la fragmentation. Ces statistiques qui représentent toutes sortes d'informations quantitatives nécessaires comme la cardinalité des tables, le nombre de valeurs distinctes de chaque attribut, etc. L'agent évaluateur utilise ces informations pour évaluer le coût d'exécution des requêtes.

III .4.4.5 Mise à jour de la base de connaissance (BC)

Les opérations de maintenance ne doivent pas altérer l'exécution des requêtes de traitement et l'espace de stockage doit être optimisé. Les évolutions qui surviennent sur l'entrepôt de données rendent la configuration sélectionnée inefficace. L'agent extracteur doit percevoir son environnement en surveillant l'état de la base de données afin de décider à quel moment lancer les tâches de ré-optimisation.

Les changements effectués durant la vie de l'entrepôt de données peuvent consister à:

-Modification des paramètres comme le nombre de fragment, le seuil de tuning, etc. par l'administrateur.

-Supprimer certaines structures d'optimisation qui deviennent non utilisables.
Par exemple, un index volumineux rarement utilisé peut être supprimé et l'espace qu'il occupe récupéré pour sélectionner une autre technique redondante.

-Réaffecter l'espace de stockage alloué à chaque technique d'optimisation de façon à favoriser les techniques les plus intéressantes.

III .4.4.6 Le choix de la nature de la sélection

L'agent préférence est doté par une logique décisionnelle qui dépend du paramètre fixé par l'administrateur. Le choix de la nature de sélection des deux techniques repose sur l'analyse des paramètres W et ƛ.

Analyse de paramètre S

Afin de recommander des techniques d'optimisation (index de jointure binaires, fragmentation horizontale (FH)), la prise en compte du coût de maintenance dans le processus de sélection des index de jointure binaires est pertinente pour les applications d'entreposage en temps réel, où la mise à jour de techniques d'optimisation redondantes est souvent coûteuse.

Dans le cas où l'AED dispose de peu d'espace disque pour les IJBs, il ne peut pas satisfaire les exigences des techniques redondantes. En effet, il serait intéressant de concentrer son optimisation sur une technique non redondante, comme la FH, pour avoir une bonne réduction du coût d'exécution des requêtes. Ce scénario est aussi intéressant lorsque la charge de requêtes contient plusieurs requêtes de mise à jour, chose qui rend l'utilisation des techniques redondantes très coûteuse.
La contrainte de mise à jour : les index nécessitent leur mise à jour dans le cas de changement sur les données indexées (mise à jour, insertion, suppression). Si un index nécessite un coût de mise à jour élevé, il devient non bénéfique.

Analyse des paramètres W, λ

Les différents modes de sélection (FHSEULE, IJBSEULS ou FH&IJB) offrent une sélection mono et multi-structure d'optimisation définie comme suit :

– FHSEULE : Pour effectuer une FH de l'entrepôt de données sans l'indexer, l'AED met λ = 1. Dans ce cas, il considère que toutes les requêtes sont bénéficiaires du processus de fragmentation, par conséquent aucun index ne sera créé.

– IJBSEULS : Pour effectuer une indexation de l'entrepôt de données non fragmenté, l'AED met λ = 0 et W = 1. Dans ce cas, l'entrepôt de données n'est pas fragmenté (W = 1) et toutes les requêtes sont non bénéficiaires (λ = 0). Par conséquent, la sélection des IJB sur l'entrepôt non fragmenté se base sur tous les attributs candidats pour l'indexation (l'ensemble des attributs de fragmentation est vide) et sur toutes les requêtes de la charge.

– FH&IJB : Pour effectuer une sélection simultanée d'un schéma de fragmentation horizontal et d'une configuration d'index selon l'approche que nous avons proposée, l'AED fixe une valeur de W supérieure à 1 et une valeur de λ telle que $0 < \lambda < 1$. Dans ce cas, un schéma de fragmentation est sélectionné suivi d'une configuration d'IJB. Le paramètre λ permet d'identifier les requêtes non bénéficiaires du processus de fragmentation. Ces requêtes dépendent du schéma de fragmentation sélectionné, en particulier du seuil W utilisé.

III .4.4.7 Partage des attributs entre FH et IJB

Le partage de ces attributs, entre index et fragmentation, se base sur une classification par l'algorithme de datamining « k-means » [32]. L'agent préférence répartit les attributs entre IJB et FH en deux ensembles : *Classe_IJB* et *Classe_FH*, avant de réveiller les agents sélectionneurs IJB et FH. Cette classification repose sur trois critères : facteur de sélectivité des requêtes, la cardinalité des attributs et la fréquence d'un attribut dans la charge de requêtes.

Le but de cette étape est d'élaguer cet espace de recherche en réduisant le nombre d'attributs utilisés pour la sélection des configurations et choisir, pour chaque technique d'optimisation les attributs les plus adaptés.

III .4.4.8 Sélection d'un schéma de fragmentation et IJB par l'algorithme génétique

Afin d'effectuer la sélection d'un schéma de fragmentation et ensemble d'index de jointure binaire, les agents sélectionneurs FH et IJB doivent faire appel au APT JAVA nommée JGAP (Java Genetic Algorithms Package).

Notons que notre AG peut générer des solutions violant la contrainte d'espace et la contrainte W (nombre de fragments). Pour améliorer ces solutions, chaque chromosome (schéma de fragmentation ; index) généré par notre AG est évalué par l'agent évaluateur en utilisant la fonction de *fitness* représentée par un modèle de coût qui calcule la somme de pages accédées (entrées/sorties) nécessaires pour exécuter chaque requête dans l'entrepôt de données.

On s'intéresse généralement à *Maximiser F(x)* sous la contrainte *C(x)* ou *C(x)*qui représente un coût généré par la solution *x*. La fonction de pénalité (notée $Pen(x)$) permet de pénaliser les solutions violant la contrainte.

$$F'(x) = \begin{cases} \dfrac{F(x)}{P(x)} & si\ P(x) > 1 \\ F(x) & sinon \end{cases}$$

Afin de garantir la robustesse des algorithmes évolutifs, le calcul d'un très grand nombre de fitness (parfois de l'ordre de plusieurs centaines de milliers) est généralement nécessaire avant l'obtention d'une bonne solution. Ce nombre de calcul important peut s'avérer problématique lorsque le coût de calcul (ressources systèmes ou temporelles) de la fitness est important, ou lorsqu'on travaille en grande dimension sur des fonctions à complexité importante par exemple.

Pour que l'interaction entre l'agent évaluateur et les autres agents de notre système soit agile, deux scenarii sont considérés:

Le premier scenario fait appel à l'évaluateur : Si le noyau de SGBD est sollicité par plusieurs transactions, alors dans ce cas il est préférable d'utiliser un modèle basé sur une fonction mathématique. Ce coût est calculé en utilisant des formules qui prennent en compte un certain nombre de paramètres et de statistiques collectés sur la BC. L'avantage de ce modèle réside dans sa rapidité de calcul du coût mais il est limité par les hypothèses simplificatrices parfois trop irréalistes.

Le deuxième scenario fait appel à l'optimiseur de requêtes du SGBD : Si l'agent évaluateur est sollicité par plusieurs requêtes, dans ce cas l'optimiseur reçoit la requête, il évalue les différents plans d'exécution et retourne le meilleur plan avec son coût. Le fait d'utiliser l'optimiseur de requêtes rend le calcul du coût plus fiable mais engendre deux inconvénients majeurs : (1) le coût que l'optimiseur estimé dépend du SGBD utilisé et (2) faire souvent appel à l'optimiseur engendre un coût d'exécution supplémentaire et dégrade la qualité de l'optimiseur qui passe plus de temps à estimer le coût d'exécution des requêtes qu'à les exécuter.

Le principe de l'adaptation de l'Algorithme Génétique au problème de sélection d'une configuration de FH et d'IJB est détaillé dans » [1].

III .4.4.9 Modèle de coût

Le coût est calculé en utilisant des formules qui prennent en compte un certain nombre de paramètres et de statistiques collectées par l'agent extracteur. Les approches de sélection sont basées sur le modèle de coût. Pour cela, nous avons défini un agent évaluateur. Ce dernier permet d'estimer le coût d'exécution des requêtes sur un entrepôt de données selon les deux modèles de coût proposés dans [1] [42] respectivement pour l'estimation de la qualité du schéma de fragmentation, de la configuration d'index.

III .4.4.10 Amélioration de la performance

Il est intéressant d'enrichir notre approche par d'autres scenarii afin d'améliorer la configuration finale et considérer tous les cas possibles ce qui offre la dynamique dans notre solution. Les politiques considérées sont :

- Le gain d'un attribut est mesuré sur le nouveau schéma fragmenté
- Une requête trop sélective.
- Nombre d'attributs exclus par l'agent FH bénéfique pour l'indexation.
- Regroupe certains sous-domaines dans un seul fragment plus large.
- L'espace de stockage S ne consommé pas totalement par l'agent IJB.

Scénarios	**Résultat**	**Amélioration**
W est petit	regroupe certains sous-domaines dans un seul plus large.	Indexer sur l'attribut fragmenté
Sélectivité d'une requête sur le fragment	Jointure entre table fait et table dimension	Indexer fragments
S ne pas consommé totalement par	Indexation de l'espace vers le négociateur	Création des index locaux sur les fragments de taille important

l'IJB		
Nombre d'attributs exclus par FH est implorant	L'espace de recherche d'IJB très grand	élagage à base des règles

Tableau III.4 : Amélioration de la performance

- Des attributs peuvent être choisis par l'algorithme K means pour la FH alors que leur sélection pour un IJB donnerait un meilleur résultat. L'agent IJB et FH vérifient l'appartenance ou la non-appartenance de ces attributs à la configuration et du bénéfice qu'apportent ces attributs séparément. L'appel de négociateur par les deux agents IJB et FH améliore le premier algorithme en prenant en compte de nouveaux attributs écartés de l'espace de la fragmentation horizontale, et remplace un attribut qui engendre un coût maximum; soit ajouter à la configuration initiale, puis de construire une configuration finale en améliorant cette configuration de manière itérative.

Pour que la FH ne réalisée pas indépendamment de la sélection d'IJB ; on doit considérer le scénario suivant : il n'est pas sûr que les attributs laissés pour indexation soient les plus appropriés. Les IJB définis après fragmentation peuvent s'avérer utiles pour l'optimisation. En effet, si le volume de données chargées par l'index est tout aussi volumineux que la table elle même, l'optimiseur du SGBD jugera préférable d'utiliser la table directement.

1. Création d'un index sur le fragment

Dans le cas où l'agent IJB ne consomme pas tout l'espace occupé, alors l'espace non occupé sera transféré au négociateur pour créer des index sur fragments (index local). Le volume des fragments très grand, peut engendrer des jointures coûteuses qui sont définies sur ces fragments. Ainsi certaines requêtes deviennent trop sélectives. Par conséquente la fragmentation seulement sur cet attribut est insuffisante. Dans ce cas si la sélectivité des requêtes est inferieure au paramètre τ % alors l'agent négociateur analyse la charge Q et recommande l'indexation ou non de fragment fait.

2. Remplacement d'un attribut éliminé par l'agent FH (Ratios profit/espace)

Lors du remplacement d'un IJB par un ou plusieurs IJB*i*, la politique de remplacement considéré : l'attribut apportant moins de réduction de coût à la configuration *ConfigIJB* ; Pour k attributs candidats, k éliminations sont effectuées et celle engendrant un coût maximum sera retenue. Le gain apporté par l'attribut est mesuré par le ratio profit/espace avec les deux fonctions *Poids* et *Taille* : $Remplacer_{/ConfigIJB}(IJBi) = \frac{Poids(ConfigIJB(IJBi))}{taille(IJBi)}$. Cette fonction calcule le profit apporté par

l'index IJB*i* par apport à l'espace de stockage *taille* (IJB*i*) qu'il occupe. La fonction ratio profit/espace privilégie les index accélérant le mieux l'accès aux données tout en occupant le plus petit espace possible. Si l'espace de stockage disponible est supérieur ou égal à cette valeur, alors l'agent considère la même configuration.

3. **Réécriture des requêtes pour exécution sur schéma fragmenté**

Afin d'évaluer l'impact d'un attribut exclu par l'agent IJB sur la fragmentation de l'entrepôt, il faut réécrire les requêtes sur le nouveau schéma de données. Cette réécriture a pour but de charger uniquement les fragments nécessaires pour l'exécution d'une requête donnée. Une fois l'entrepôt de données fragmenté, il peut être représenté sous forme de plusieurs sous schémas en étoile. Avant cela, il faut identifier les sous schémas valides et leur correspondance pour une requête.

Conclusion

Dans ce chapitre, nous avons proposé une modélisation SMA constituée de cinq types d'agents, distribués et centralisés via un négociateur. L'avantage de cette approche : l'agent sélectionneur travaille localement sans connaître l'agent sélectionneur de l'autre technique ; en cas de besoin de communication inter technique il fait appel au négociateur.

Ainsi, nous avons proposé une démarche de sélection combinée d'un schéma de fragmentation et des index de jointures binaires dirigée par cette architecture SMA. Pour que notre solution soit dynamique, au cours de l'exécution de notre algorithme SMA, les agents réalisent plusieurs scénarii à travers les messages échangés.

Le modèle de conception de SMA que nous avons proposé dans ce travail est développé suivant une approche générique, indépendamment de la technique d'optimisation choisis par l'AED. Pour motiver nos travaux de recherche d'un point de vue pratique, la section suivante décrit l'application de notre modèle avec une étude de cas (sélection de FH & IJB), ainsi que sa projection sur un environnement opérationnel (c.-à-d, comment implémenter les différents agents du modèle avec la plateforme Multi-agents JADE avec un entrepôt de données ORACLE 10g).

Chapitre IV : *Étude expérimentale*

Introduction

Afin de motiver nos travaux de recherche d'un point de vue pratique, nous avons développé une application que nous appelons *AdminSMA* pour assister l'administrateur dans ses tâches de conception physique des entrepôts de données. Lors de cette phase, l'administrateur doit sélectionner un ensemble de techniques d'optimisation pour satisfaire les requêtes décisionnelles. Devant la panoplie de structures d'optimisation existantes, les approches de sélection et les algorithmes de sélection, l'administration d'un entrepôt de données devient une tâche très complexe. Ainsi, l'administrateur trouve la nécessite d'être assisté afin d'effectuer de bons choix pour améliorer les performances de son entrepôt de données. Les outils d'assistance à l'administration d'ED existant prennent en charge la sélection d'un grand nombre de structures d'optimisation : index, vues matérialisée, fragmentation horizontale et verticale. On cite l'outil AutoAdmin lancé par Microsoft SQL Server , DB2 Advisor d'IBM et Oracle Tuning Advisor pour l'administration et le tuning des entrepôts de données. Dans le domaine de recherche on peut citer l'outil ParAdmin proposé par [1].
Nous allons implémenter l'approche SMA afin de résoudre le problème de sélection d'IJB et FH.

Afin de valider nos contributions, nous avons mené des expériences qui nous ont permis d'évaluer et de comparer notre approche présentée dans le chapitre III avec l'approche séquentielle. Nous avons également évalué les performances de l'entrepôt de données . Ces expériences exploitent un entrepôt de données généré par le banc d'essais APB-1 (2) et sa charge de requêtes décisionnelles

Nous présentons dans ce chapitre l'environnement d'implémentation, l'outil implémentant notre approche SMA ainsi qu'une série d'expériences en utilisant ce dernier.

IV.1 Environnement d'implémentation

Nous présentons ici brièvement l'environnement d'implémentation et outils utilisés durant la phase de réalisation.

1 **Langage de programmation**

Nous avons implémenté notre plateforme avec le langage de programmation JAVA vu que ce dernier est un langage simple, intuitif, orienté objet, performant et dynamique. C'est aussi un langage multiplateforme disposant d'une JVM (Java

Virtual Machine) lui permettant de s'exécuter dans des environnements hétérogènes en permettant une indépendance envers les réseaux et les systèmes d'exploitation.

2 **Environnement de Développement Intégré (IDE)**

Le choix de l'IDE est souvent dicté par des préférences personnelles. Néanmoins, il est conseillé d'utiliser Eclipse ou Netbeans dans notre cas. Ces deux IDEs facilitent le développement en Java SE[4] et ME[5] et permettent l'utilisation de plateforme JADE. Les développements de ce projet ont été effectués en utilisant Netbeans.

3 **Serveur de bases de données**

Nous avons utilisé le benchmark APB-1 (2) implémenté sur Oracle 10 g. Ce benchmark utilise un schéma en étoile illustré dans la figure 9. Il contient quatre tables de dimension et une table des faits. La taille de chaque table en termes d'instances est décrite dans la Table IV.1.

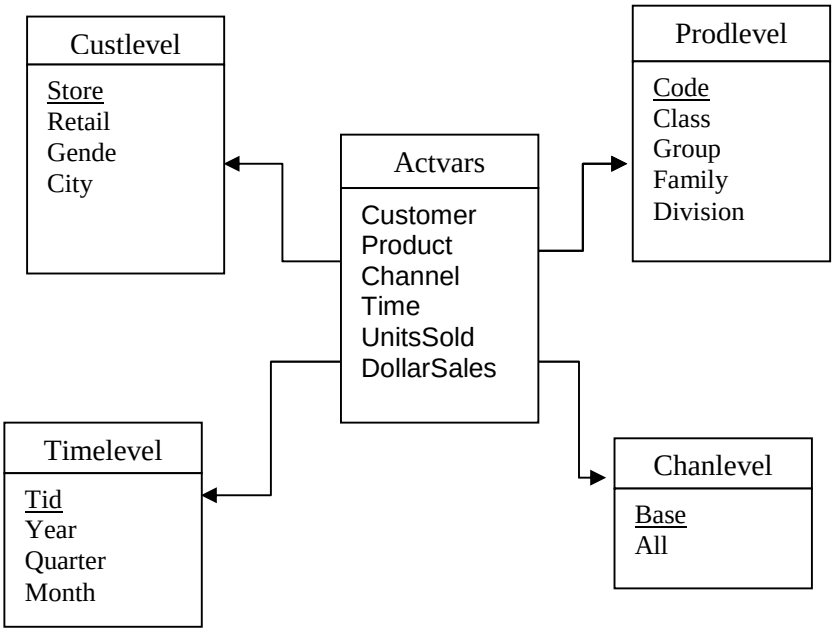


Figure IV -1 : Schéma en étoile de l'entrepôt expérimental

Table	Nombre d'enregistrement	Type de table
Actvars	24 786 000	Faits
Chanlevel	9	Dimension
Custlevel	900	Dimension
Prodlevel	9000	Dimension
Timelevel	24	Dimension

Tableau.IV.1: Taille des tables

[4]Standard Edition:plateforme Java destinée plus particulièrement aux applications pour poste de travail.

[5] Micro Edition:framework Java spécialisé dans les applications mobiles.

4 Plateforme d'agents :

Dans le choix de plateforme, on a eu le choix entre différentes plateformes d'agent Jade, Aglet, Voyager, … mais on a choisi JADE (3). Ce choix est motivé par le fait que JADE est open source et facile à manipuler vu qu'il est développé par JAVA. Nous pouvons ajouter le fait qu'il est compatible FIPA t doté d'une API puissante.

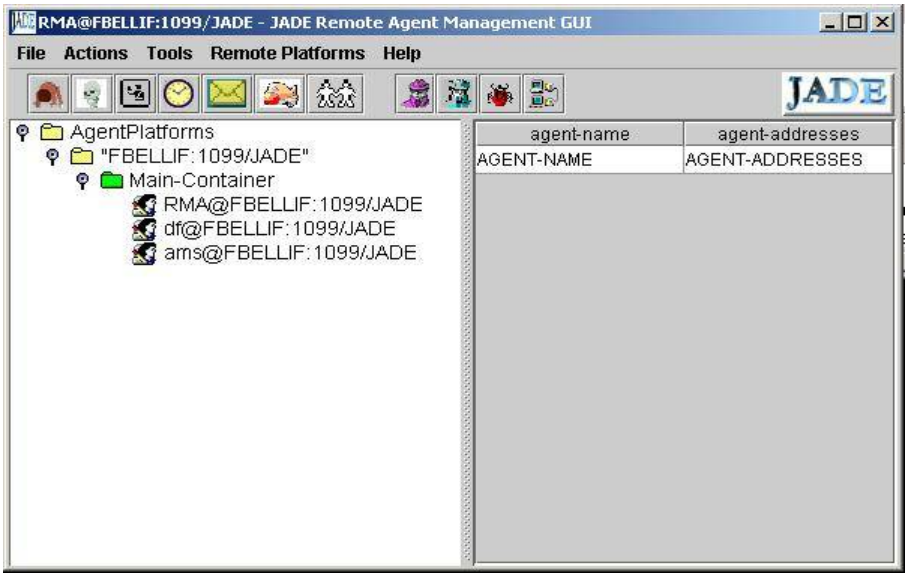


Figure IV -2: L'environnement d'exécution des agents

Nous présentons dans ce qui suit la présentation de notre outil développé AdminSMA et un scénario d'utilisation.

IV.2 Présentation de l'outil réalisé AdminSMA

L'application fonctionne sur des machines différentes ou sur une seule machine. La console d'administration JADE pourra être gérée par les agents sur une seule machine. Ce chapitre explique également la manière dont les différents agents du système vont communiquer.

Notre outil AdminSMA permet de charger un schéma en étoile d'un entrepôt de donnée et de spécifier une charge de requêtes à optimiser. AdminSMA permet de réaliser les opérations suivantes :

- Visualiser l'état de l'entrepôt ainsi que la charge de requêtes
- Choisir le mode de sélection de techniques d'optimisation (FH et/ou IJB).
- Répartition des attributs entre les deux techniques FH et IJB par l'agent préférence.

- Visualiser les résultats de sélection : attributs de fragmentation, nombre de sous domaines
- Visualiser pour chaque attribut, nombre de fragments de la table faits, attribut d'indexation.
- Visualiser le journal de communication entre les agents durant le processus de sélection.
- Visualiser graphiquement tous les échanges de messages avec une notation très proche de celle utilisée sous UML par l'agent Sniffer.

Le paramétrage de l'outil donne plus de liberté à l'administrateur pour le choix des paramètres systèmes, les paramètres de l'algorithme de sélection et les paramètres de l'agent.

IV.2.2 Scénario d'utilisation de notre Prototype

Nous présentons dans cette section les principales fonctionnalités de l'outil « AdminSMA » à travers des captures écran illustrant les interfaces de ce dernier. Pour bien décrire ces interfaces, nous considérons un scénario d'exécution et de manipulation de l'outil.

L'outil permet à l'administrateur d'effectuer les opérations de l'extraction jusqu'à la visualisions des résultats de sélection. L'administrateur intervient au niveau du paramétrage de l'application.

IV.2.2 Interface d'accueil d'AdminSMA

1. Visualiser un entrepôt de données

Après le lancement de l'outil AdminSMA l'agent extracteur se connecte au SGBD Oracle et récupère certaines informations tels que : le schéma de l'entrepôt, les tables, les attributs des tables. Il permet également de spécifier la charge de requêtes à optimiser, à partir d'un emplacement sur disque.

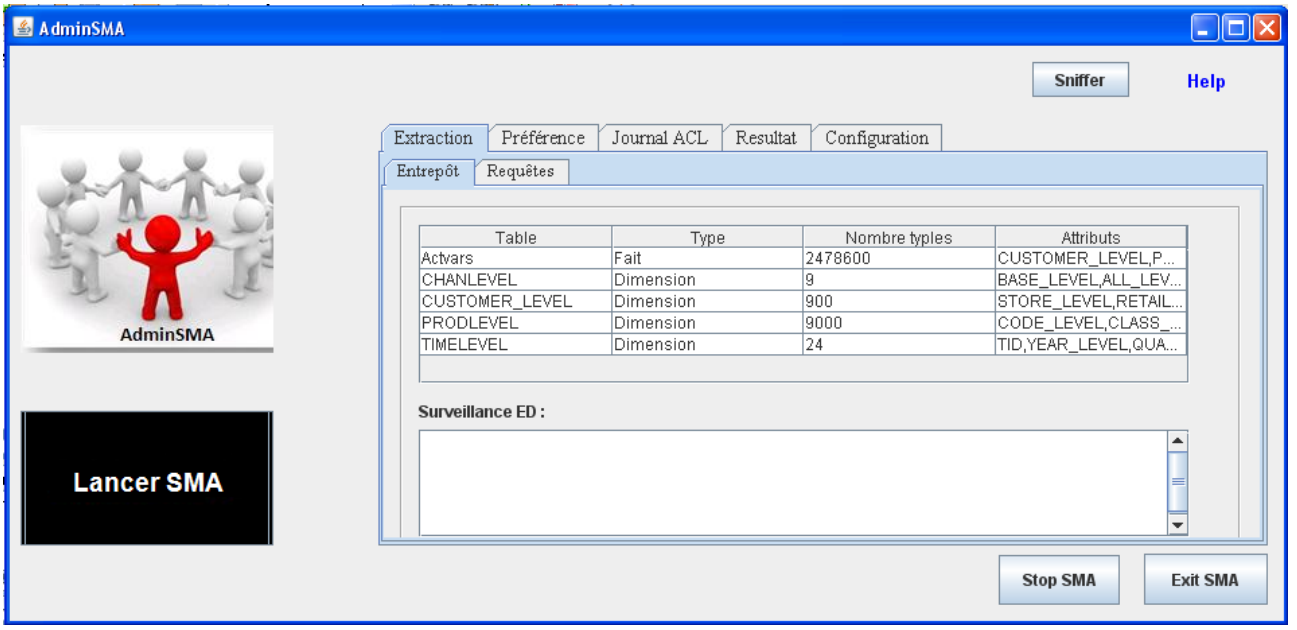


Figure IV -4:Visualiser l'entrepôt de données et les requêtes

2. Sélection des techniques d'optimisation

Après le chargement des requêtes, l'agent préférence procède à l'identification des tables et attributs candidats pour chaque technique d'optimisation choisie. Ceci est effectué automatiquement en analysant les requêtes sélectionnées.

Le choix de la nature de sélection des deux techniques repose sur l'analyse des paramètres W et ƛ. (voir la le chapitre III). Le partage de ces attributs, entre index et fragmentation, se base sur une classification effectuée par l'algorithme k-means en se basant sur une classification par l'algorithme de datamining « k-means » »[32].

La figure 5. montre le panneau de sélection automatique des tables et attributs candidats pour chaque technique d'optimisation.

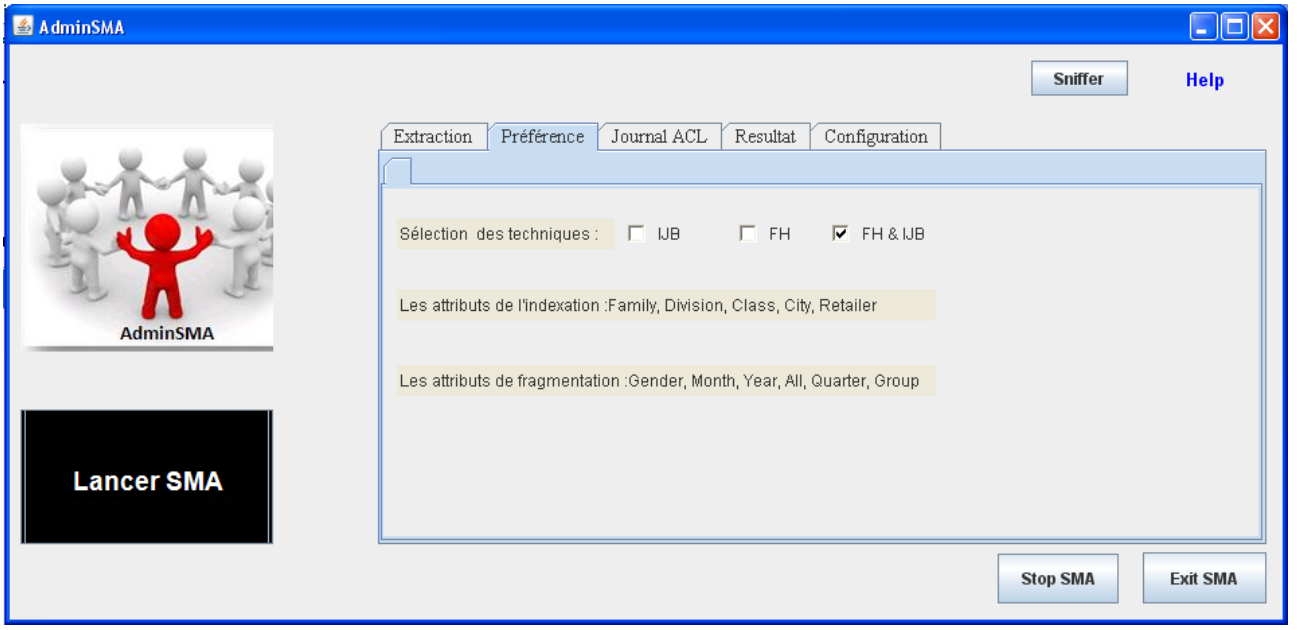


Figure IV -5: Gestion des préférences

3. Envoi d'un message JOURNAL

Les tâches sont réalisées à travers les messages échangés entre les agents. Le menu Journal ACL permet à l'AED de pouvoir consulter un message avec ses détails.

Un message est constitué du contenu du message en lui même et d'un destinataire. La méthode sendMessage du composant de communication permet d'envoyer un message. Lorsqu'un message est envoyé, l'agent retourne une chaîne de caractères qui indique si le message a pu être correctement sauvegardé. Si le destinataire du message n'existe pas, l'agent retourne faux et une alerte est affichée à l'écran de l'utilisateur.

Il est possible que l'envoi du message n'ait pas fonctionné comme il est possible aussi que l'agent n'ait pas réussi à se connecter au serveur Oracle 10g. Le message est donc sauvegardé dans la mémoire. Il est disponible dans le menu "Alerte" de l'application.

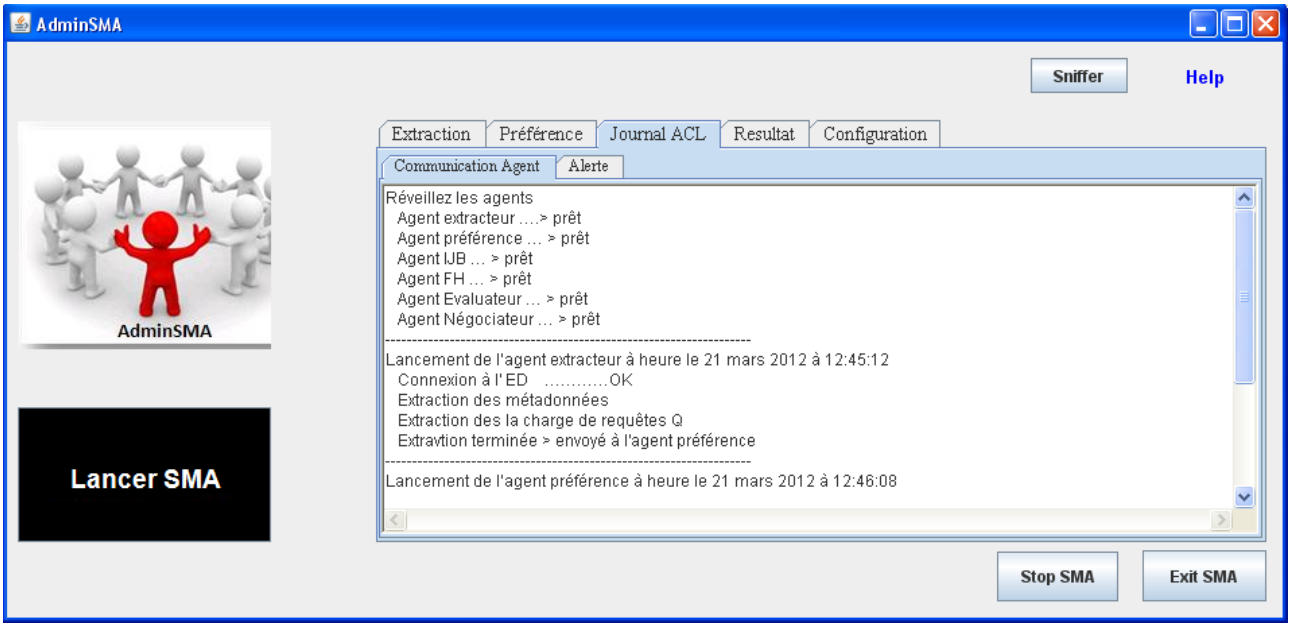


Figure IV -6: Journal de communication(Log)

La figure IV-7montre les messages échangés entre les agents dans le processus de sélection.

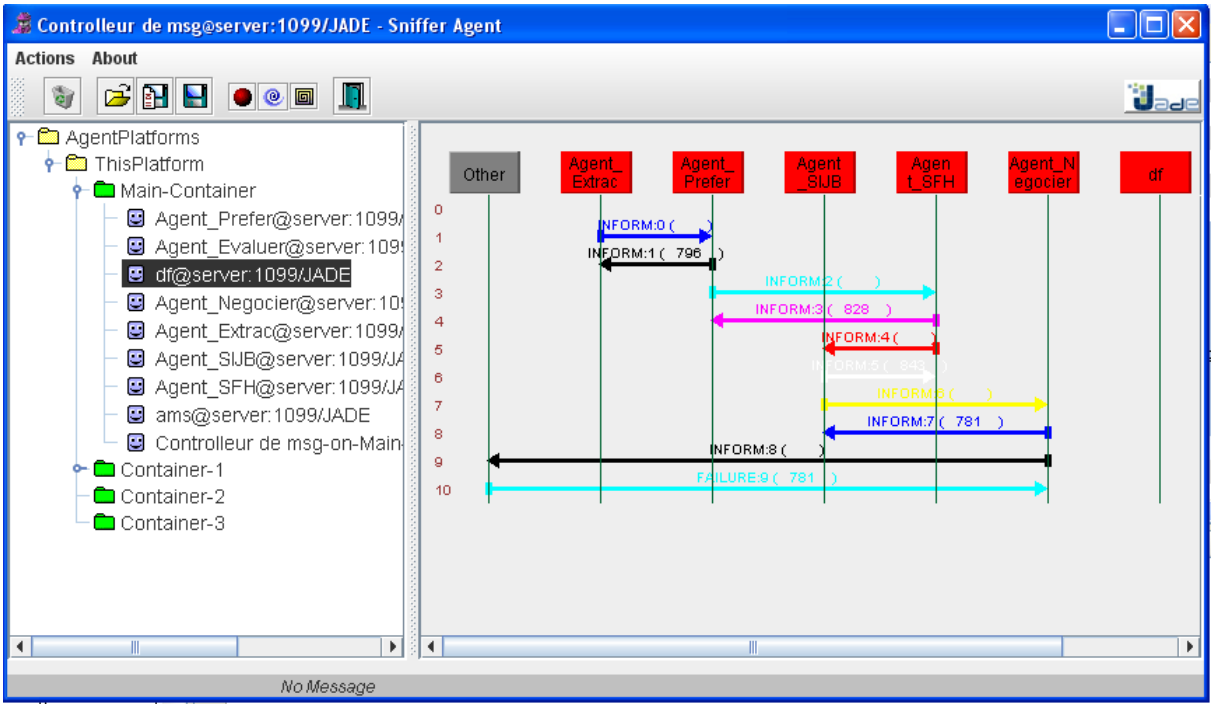


Figure IV -7: Les messages échangés

4. Configuration de l'outil

L'administrateur doit introduire les paramètres systèmes, les paramètres de l'algorithme des APIs ainsi les paramètres agents. Cette étape est décrite dans la figure IV.8..

- ✓ ***Les paramètres des API*** : paramètre de l'algorithme de classification « k-means » à savoir le nombre d'itérations pour exécuter la classification et les paramètres lié à l'algorithme génétique; la taille de la population, le nombre de génération, le taux de croisement et le temps de mutation. Ces paramètres sont introduits en entrée pour les API java relatives à ces deux algorithmes.
- ✓ ***Les paramètres système*** : Pour lancer l'algorithme de sélection, l'administrateur est amené à introduire les paramètres systèmes (taille du buffer, taille de la page système). Nombre de fragments W et limite de stockage S.
- ✓ ***Les paramètres des agents*** **:** Pour chaque agent est associée une adresse IP et un état à initialiser.

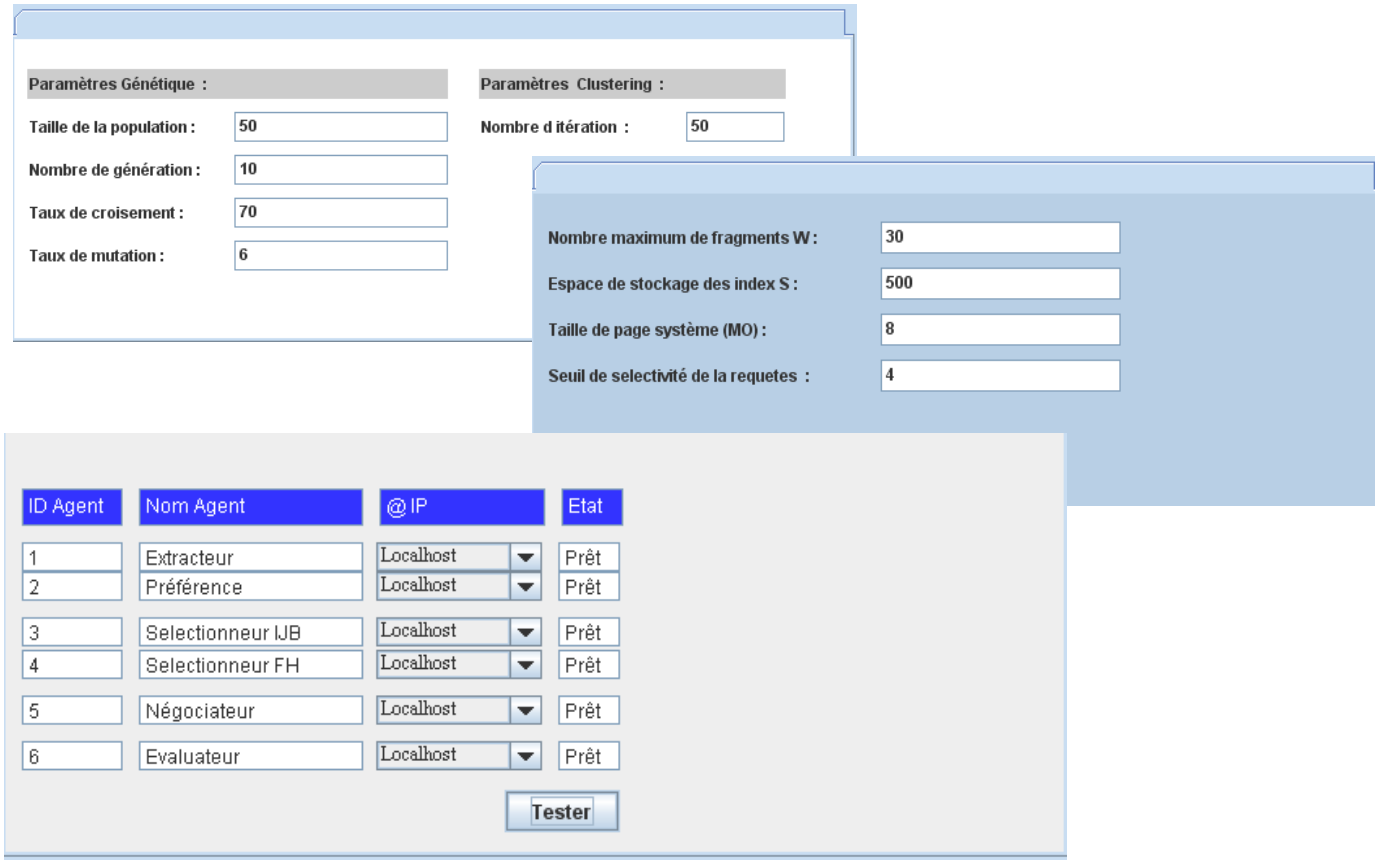


Figure IV -8:Configuration de l'outil

5. *Résultat*

Afin d'effectuer la sélection d'un schéma de FH et ensemble d'IJB ; les agents sélectionneurs FH et IJB font appel à l'API JGAP. Une fois la sélection achevée, l'outil permet de visualiser les résultats (figure IV-9), il montre les attributs de fragmentations muni chacun du nombre de sous domaine, ainsi que le nombre total de fragments faits. Il montre aussi les attributs d'indexation et les attributs exclus du processus de sélection.

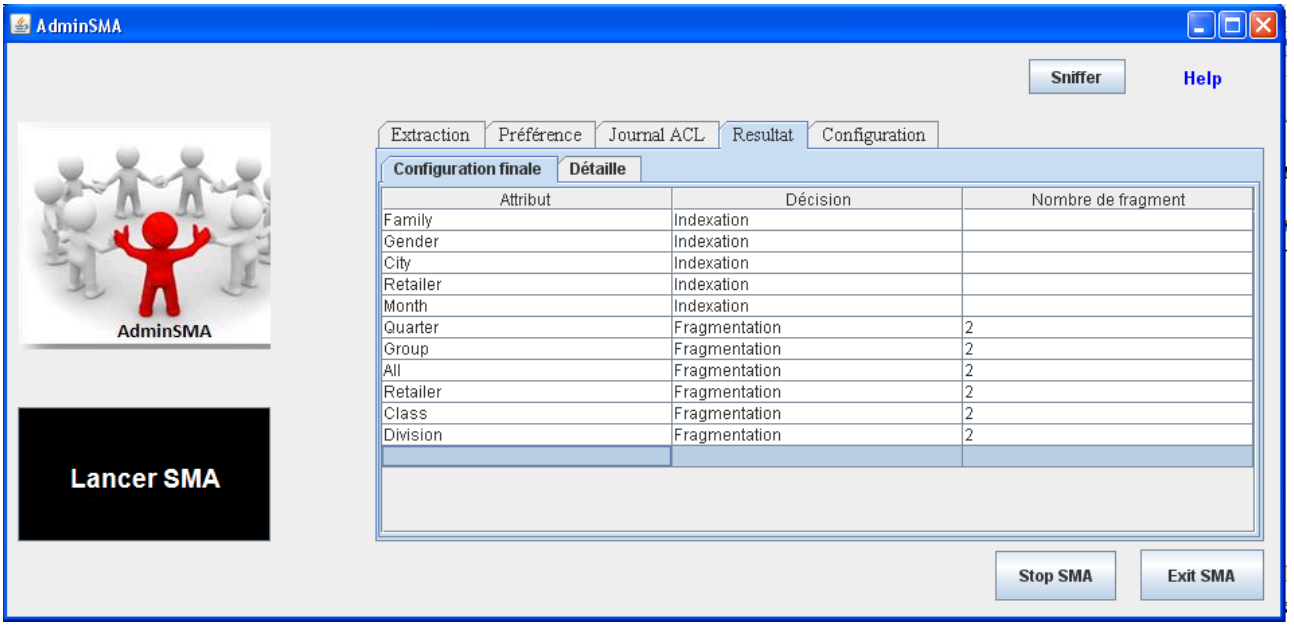


Figure IV -9: Fenêtre décrivant les détails de résultat

IV.3 Les séries d'expériences

Les tests unitaires dans un environnement multi-agents sont assez compliqués. En plus, ces tests sont lancés sur la même machine. Afin d'approuver l'utilité de notre approche SMA, nous avons effectué plusieurs mesures d'efficacités.

Les séries des expériences s'articulent autour les points suivants : (1) Comparaison de la taille de l'espace de recherche de notre approche par rapport à l'approche itérative. (2) Comparaison de la vitesse de notre approche SMA par rapport à l'approche séquentielle. (3) Etude de l'impact de paramètres W sur la taille des fragments faits et la sélectivité des requêtes (4) Etude de la Qualité de solution obtenue sans et avec l'impact de l'agent négociateur.

IV.3.1 *Impact du nombre de tables de dimension sur l'espace de recherche*

Nous nous intéressons à présent, à l'évaluation de l'impact du nombre des dimensions intervenant dans ce processus d'optimisation de requêtes sur les

résultats de la recherche. Considérons les configurations suivantes : Une seule table : (Timelevel), deux tables : (Timelevel et Prodlevel), Trois tables : (Timelevel , Prodlevel et Quatre tables : Custlevel) , (Timelevel , Prodlevel , Chanlevel et Custlevel).

Considérons les configurations suivantes ; sachant que AED Il est habituellement utilise avec des données de cardinalité réduite.

Dimension	Espace de recherche original	
	FH	IJB
Timelevel ,	4608	2^5 -1
Timelevel , Prodlevel	9216	2^{10}-1
Timelevel , Prodlevel , Custlevel	55296	2^{14}-1
Timelevel , Prodlevel , Chanlevel , Custlevel	224484	2^{16}-1

Tableau. IV.2 : Impact du nombre de tables de dimension sur l'espace de recherche

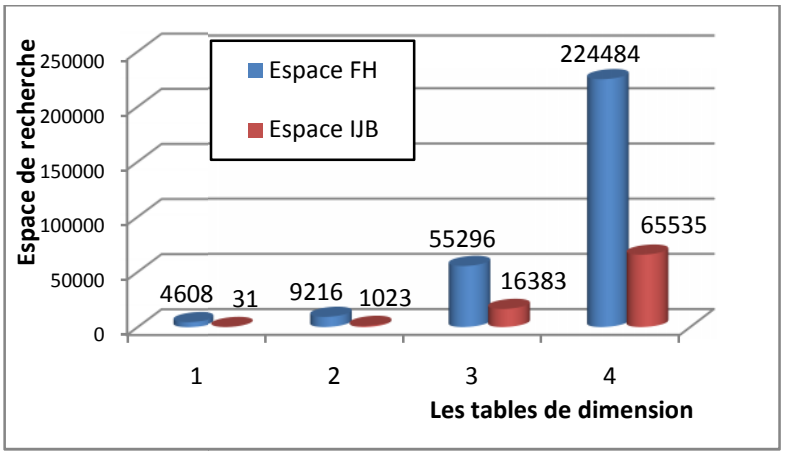


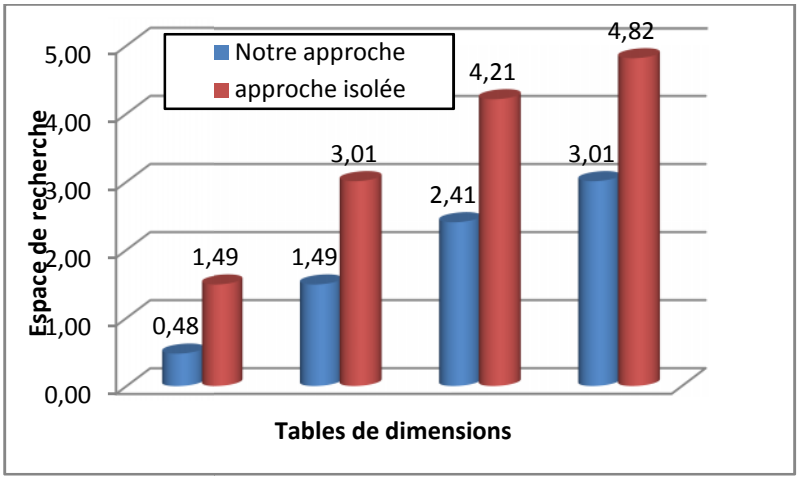
Figure IV -10 : L'impact du nombre de table de dimension sur l'espace de recherche des IJBs et la FH

On remarque que l'espace de recherche des *IJBs* et de la FH augmente d'une façon énorme.

D'après la figure 10, l'explosion du nombre de fragments est due à deux paramètres : le nombre de tables de dimension fragmentées et le nombre de fragments pour chaque table de dimension. L'élagage peut agir sur ces deux paramètres pour contrôler le nombre de fragments générés.

1) L'espace de recherche (*Approche isolée vs Approche SMA*)

L'agent préférence permet de choisir pour la FH et les IJBs les attributs les plus adéquats. Ce qui implique que chaque agent explore un espace de recherche restreint. Cette réduction de l'espace de recherche causée par la classification des attributs [32] est significative. Notons que l'indexation seule et la fragmentation seule sur tous les attributs de sélection génèrent un espace de recherche très grand.



FigureIV -11: Comparaison de la taille de l'espace de recherche des IJBs

Observations

L'intérêt de notre approche apparait clairement étant donné qu'on obtient une réduction remarquable en terme d'espace ce qui implique par la suite une optimisation de temps d'exécution de l'algorithme de sélection.

IV.3.2 Approche séquentielle vs Approche SMA

Dans la littérature, la plupart des travaux de sélection des structures d'optimisation d'ED est conçu d'une manière séquentielle**.** Dans cette étude, nous avons proposé une approche qui sélectionne la fragmentation et l'indexation d'une manière distribuée et parallèle.

Nous avons varié le nombre d'attributs de 10 à 60 et estimé le temps de résolution de ces instances par l'approche séquentielle et notre approche SMA à l'aide d'un module d'estimation.
Dans la figure 12, nous comparons la performance de l'approche séquentielle et l'approche distribuée. Sachant que la communication des agents est négligeable devant le temps d'exécution.

Hypothèse: utilisation d'un CPU de 3 GHz. On suppose que l'on peut traiter une opération primitive en 1 µs. f(n) désigne le nombre d'opérations primitives effectuées en fonction de la taille du problème.

Les tableaux.3: illustre le temps requis pour résoudre le cas et sur la base de données à l'aide les deux approches.

	f(n)		Temps d'exécution sec		
Nombre d'attributs	taille du problem IJB	taille du problem FH	Approche Séquentielle	Notre approche SMA	Réduction %
10	63	256	4,31	1,26	29,30
11	31	64	1,69	0,59	35,20
12	63	256	4.31	1,75	40,50
13	255	1024	17,37	8,39	48,33
14	127	4096	33,75	17,89	53,00
15	127	16384	119,77	71,47	59,67
20	127	1048576	7345,11	4629,62	63,03
30	32767	1,074E+09	7517503,5	4886377,24	65,00
40	1048575	1,09 E+12	-	-	-
50	33554431	1,12 E+15	-	-	-
60	1,074E+09	1,15E+18	-	-	-

Tableau 3 : Evolution de temps lorsque la taille du problème augmente

Pour pouvoir fournir une **représentation** visuelle des données des grandes valeurs sur un même graphe, nous avons utilisé une échelle **logarithmique** Log.

La figure **I**V.12: illustre le temps requis des deux approches.

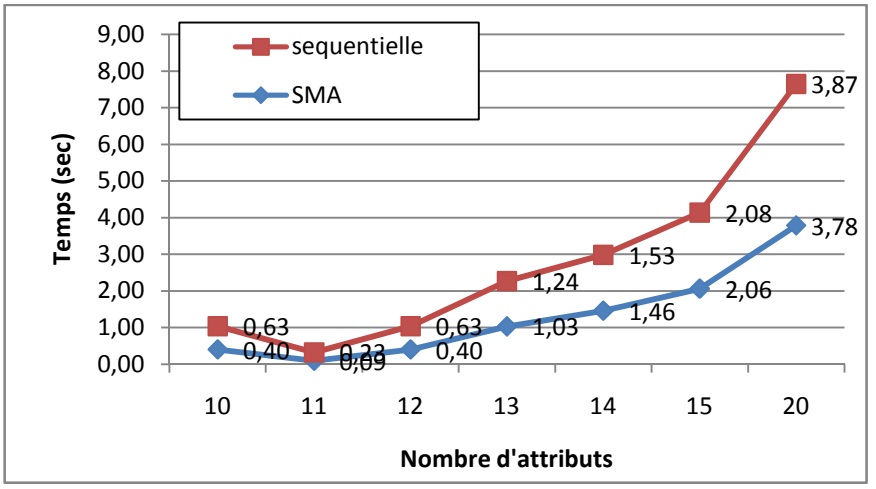


Figure IV -12 : Comparaison des speed up des approches séquentielle et SMA

Les tests séquentiels qui se basent sur la sélection d'un schéma de FH suivi de la sélection des IJB sont dans certains cas irréalistes. Dans notre approche chaque agent explore un espace restreint. L'espace de recherche des IJBset de la FH contient un ensemble réalisable de fragments et des index.

D'après l'estimation, nous observons que le temps de recherche de FH et IJB dans l'approche SMA est significativement plus petit que l'espace de recherche de l'approche séquentielle pour les situations réalistes (10, 12, 13, 14,15) de façon raisonnable.

À partir de nombre d'attributs n = 20, la valeur de gain atteint 60 %.Vu que le nombre d'attributs augmente ; par conséquent l'avantage de l'exécution parallèle apparait clairement.
En outre, les cas n = (30, 40, 50,60) sont irréalistes en raison de l'explosion d'espace de recherche.

IV.3.3 Impact de la contrainte W sur de la taille de fragments et la sélectivité.

Dans la deuxième série d'expériences, nous étudions l'effet du paramètre W sur la taille des fragments de faits et la sélectivité des requêtes. Nous avons varié la contrainte W de 20 jusqu'au 200.

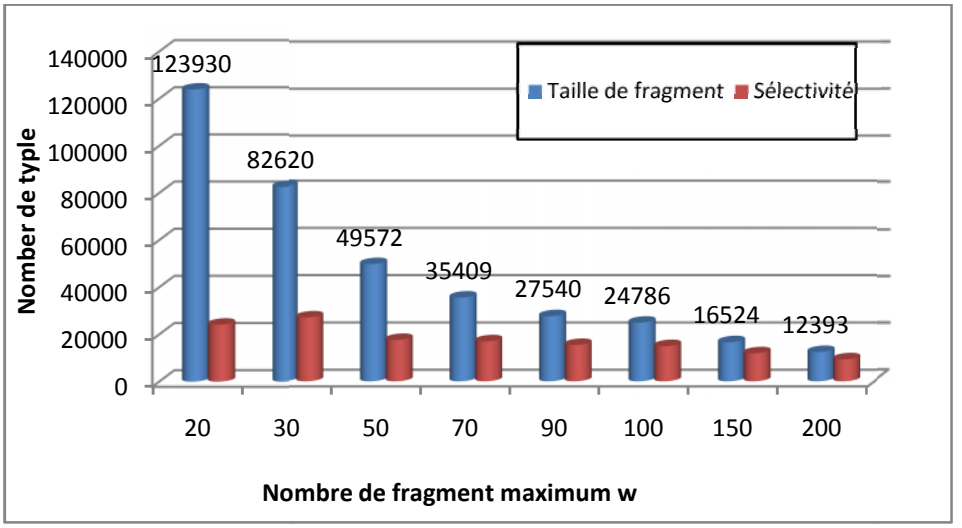


Figure III -13 : Effet de w sur la taille de fragments faits et la sélectivité.

Les résultats indiquent qu'il existe une relation inverse entre la taille de fragment fait et la sélectivité de la requête par rapport aux fragments valides.

Si l'*AED* veut un nombre de fragment réduit, les sous domaines des attributs de la fragmentation sont regroupés ; par conséquent la sélectivité de la requête devient petite. Dans ce cas, si la sélectivité des requêtes est inferieure au paramètre τ % alors l'agent négociateur analyse l'ensemble de schéma de fragmentation optimale *SF* par

rapport à la charge Q et recommande l'indexation ou non de fragment fait à l'aide de la sélectivité.

Pour W = 30, la taille de certains fragments faits sont volumineux et certaines requêtes sont trop sélectives; par conséquent la fragmentation seule sur cet attribut est insuffisante.

Nous présentons dans la section suivante des expériences visant à mesurer la qualité de la solution obtenue par notre approche.

IV.3.4 Variation des paramètres W, S

Test 1: Variation de W (S = 700mo)

Afin de voir l'influence de la contrainte W sur l'optimisation de la charge de requêtes, nous avons fait varier W avec S=700 Mo. Nous avons relevé le coût d'exécution en nombre E/S.

L'agent négociateur décide d'indexer ou non sur l'attribut selon le paramètre τ fixé par l'AED. Si une requête retourne moins que τ % des tuples de la table (sélectivité inférieure à τ %) alors la fragmentation seule sur cet attribut est insuffisante ; donc il est souhaitable d'indexer les attributs appropriés du prédicat.

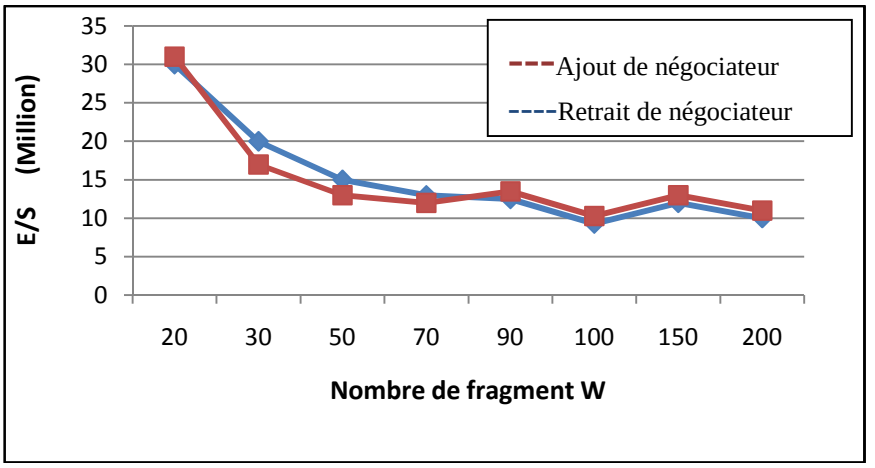


Figure IV -14 : Nombre E/S par variation de W

La figure 8 montre une meilleure optimisation pour un W = 100. L'ajout de l'agent négociateur apporte une amélioration pour 3 requêtes de sélectivité inferieure à (10%). S'il existe des requêtes trop sélectives, la présence du négociateur influe sur l'optimisation, cela réduit le nombre de lignes visitées par des requêtes et améliore le temps de réponse si les requêtes sont correctement dirigées vers une partie trop petite de ces fragments. Les mesures suivantes sont prises pour les fragments indexés et non indexés et montrent que l'indexation de fragment apporte un gain à la FH.

Test 2: Variation de S (W = 20)

Dans le deuxième test, nous avons voulu étudier l'influence de la contrainte d'espace de stockage des index S sur l'optimisation de la charge de requêtes. Nous avons fait varier S avec W=20. Pour chaque valeur, nous avons relevé le coût d'exécution de la charge de requêtes (Figure IV.15 ci dessous).

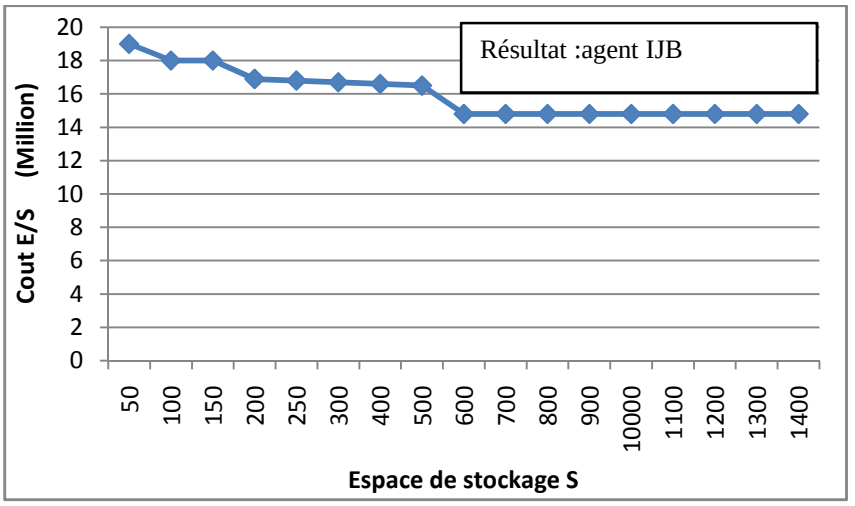


Figure IV -15 : Nombre E/S par variation de l'espace de stockage

Nous remarquons que, à partir de S = 50, le coût de la charge de requêtes est réduit de 19 millions E/S jusqu'à 16 Millions E/S pour S > 200Mo, soit une réduction de 16% du coût total. Le graphe par la suite devient linaire à partir de S > 600Mo et converge vers un coût de 15 Millions E/S.

Conclusion

Tout au long du présent chapitre, nous avons présenté notre outil *AdminSMA* et discuté les mesures de performances que nous avons mené pour valider l'approche proposée. Les résultats expérimentaux sont encourageants et montrent la faisabilité de notre approche. Néanmoins, dans le but d'avoir de meilleures performances, des améliorations peuvent être apportées, et c'est ce que la section suivante évoque.

Conclusion et perspectives

Dans notre travail, nous nous sommes intéressés à l'optimisation des requêtes décisionnelles exécutées sur l'entrepôt de données modélisé en étoile. Pour cela, nous avons proposé une nouvelle approche de sélection combinée de la fragmentation horizontale et les index de jointure binaires basée sur les Systèmes Multi-agent. Notre système multi-agent est composé de cinq types d'agents où chacun est responsable d'une tâche dans le processus d'optimisation. Ces agents sont mis en situation de coopération et parfois de concurrence afin de mieux guider le processus d'optimisation. Notre approche prend donc en considération les interactions existantes et les implémente afin de sélectionner le meilleur scénario d'optimisation et choisir les meilleures structures d'optimisation.

Notre approche prend réellement en compte l'interaction qui peut exister entre l'IJB et de la FH et les traite simultanément afin de réduire le coût d'exécution des requêtes.

Nous avons présenté et discuté dans l'étude expérimentale les tests de performances que nous avons menés pour valider l'approche proposée. Les résultats expérimentaux sont encourageants et montrent la faisabilité de notre approche. Néanmoins, dans le but d'avoir de meilleures performances, des améliorations peuvent être apportées.

En conclusion, nous pouvons résumer notre contribution aux points suivants :

1. Modélisation de problème de sélection par une architecture Multi-agents qui considère l'aspect dynamique pour pouvoir maintenir une meilleure performance de l'entrepôt et surtout éviter sa dégradation.
2. Proposition d'une démarche de sélection multiple d'IJB et de FH dirigée par l'architecture proposée : deux principales étapes caractérisent cette approche, l'optimisation locale et l'optimisation globale qui prend en charge l'impact inter technique. L'optimisation locale permet de sélectionner un ensemble des IJBs et des FHs par les agents sélectionneurs. L'optimisation globale vise à minimiser le coût d'exécution total en considérant les deux configurations avec les améliorations apportées par l'agent négociateur.

3. Implémentation du système multi-agent via un outil supportant l'approche proposée.
4. Réalisation d'une étude expérimentale pour montrer les qualités de l'approche proposée.

Perspectives

Le travail présenté dans ce mémoire ouvre plusieurs perspectives de recherche. En effet, il serait intéressant d'étudier des améliorations qui peuvent être apportées à ce travail, nous pouvons citer :

- Enrichir le fonctionnement des agents pour renforcer leur autonomie.
- Intégrer d'autres techniques d'optimisation comme les vues matérialisées.
- Doter l'agent par la capacité d'apprentissage et de perception pour pouvoir améliorer la qualité de solution au cours de la durée de vie de l'entrepôt de données.

Glossaire

AED : Administrateur de l'Entrepôt de Données.

AG : Algorithme Génétique.

BC : Base de connaissance des agents

DM : DataMining (fouille de données).

ED : Entrepôt De Données

FH : Fragmentation Horizontale.

FHD : Fragmentation Horizontale Dérivée.

FHP : Fragmentation Horizontale Primaire.

IJB : Index de Jointure Binaire.

MOLAP : Multidimensional On-Line Analytical Processing.

OLAP : On-Line Analytical Processing.

ROLAP : Rolational On-Line Analytical Processing.

RowID : Identificateur de ligne dans un SGBD relationnel.

SMA : Système Multi-agent

SF : Schéma de Fragmentation.

W : Le nombre de fragments que l'administrateur souhaite avoir

Bibliographie

[1] K. Boukhalfa, "De la conception physique aux outils d'administration et de tuning des entrepôts de données". Thèse de doctorat, Ecole nationale supérieure de mécanique et d'aéronautique, 2009.

[2] B. INMON, *« Using the datawarehouse* », Wiley & Sons Date, 1994

[3] B. FROMAN, C. GOURDON « Dictionnaire de la qualité »,

[4] L. BELLATRECHE, « *La conception physique des datawarehouses* », LISI/ENSMA Poitiers, 2006.

[5] C. D. The dificulty of optimum index selection. ACM Transactions on Database Systems (TODS), 3(4) :440–445, 1978.

[6**]** K. Aouiche. Techniques de fouille de données pour l'optimisation automatique des performances des entrepôts de données. Ph.d. thesis, Université Lumière Lyon 2, December 2005.

[7]Boukhalfa K., Bellatreche L., Richard P.,« Fragmentation Primaire et Dérivée: Étude de Complexité, Algorithmes de Sélection et Validation sous ORACLE10g », Rapport de stage, ENSMA, 2008.

[9] Mahboubi, H. « Optimisation de la performance des entrepôts de données XML par fragmentation et répartition ». Thèse pour obtenir le grade de Docteur en Informatique. 2009.

[10] Surajit Chaudhuri et al, « Index Selection for Databases: A Hardness Study and a Principled Heuristic Solution»,IEEE transactions on knowledge and data engineering, vol. 16, no. 11, november 2004.

[11] A. Caprara, M. Fischetti, and D. Maio. Exact and approximate algorithms for the index selection problem in physical database design. IEEE Transactions on Knowledge and Data Engineering, 7(6):955–967, 1995.

[12] S.Chaudhuri, M. Datar, and V. R. Narasayya. Index selection for databases: A hardness study and a principled heuristic solution. IEEE Transactions on Knowledge and Data Engineering, 16(11):1313–1323, 2004.

[13] J. Kratica, I. Ljubic, and D. Tosic.A genetic algorithm for the index selection problem. Applications of Evolutionary Computing, 2611:281–291, 2003.

[14] S. Chaudhuri and V. R. Narasayya. An efficient cost-driven index selection tool for microsoft SQL server. In Proceedings of the 23th International Conference on Very Large Data Bases, 1997.

[15] Frank M.R., Omiecinski E., Navathe S.B., " Adaptive and Automated Index Selection in RDBMS ", in 3rd International Conference on Extending Database Technology, EDBT 1992, Vienna, Austria, volume 580 de Lecture Notes in Computer Science, pp. 277-292.1992.

[16] W. King.On the selection of indices for a file. Technical report TR.RJ 1341, International Business Machines (IBM), San Jose CA,

[17] K.Whang. Index selection in relational databases. In International Conference on Foundations of Data Organization (FODO 85), Kyoto, Japan, pages 487–500, 1985.

[18] Valduriez, P. Join Indices. ACM Transactions on Database Systems, 12(2) : 218-246, June.

[19] Gundem T.I.," Near optimal multiple choice index selection for relational databases ", Computers & Mathematics with Applications, 37(2) :111-120. 1999.

[20] K. Aouiche, J. Darmont, O. Boussaid, and F. Bentayeb. Automatic Selection of Bitmap Join Indexes in Data Warehouses. 7th International Conference on Data Warehousing and Knowledge Discovery (DAWAK 05), August 2005.

[21] L. Bellatreche, R. Missaoui, H. Necir, and H. Drias.Selection and pruning algorithms for bitmap index selection problem using data mining. 9th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'07), LNCS, pages 221–230, September 2007.

[22] L. Bellatreche, R. Missaoui, H. Necir, and H. Drias.A data mining approach for selecting bitmap join indices.Journal of Computing Science and Engineering, 2(1) :206–223, 2008.
[*23*] K. ZIDI ,Système Interactif d'Aide au Déplacement Multimodal (SIADM) ; Thèse de doctorat , Université des Sciences et Technologies de Lille et l'Ecole Centrale de Lille 2006].

[24] C. D**.**The dificulty of optimum index selection. ACM Transactions on Database Systems (TODS), 3(4) :440–445, 1978.

[25]M. Golfarelli, , and E. Rizzi, S. Saltarelli.Index selection for data warehousing. Proceedings 4th International Workshop on Design and Management of Data Warehouses (DMDW'2002), Toronto, Canada, pages 33– 42, 2002.

[26] L.Bellatreche, *KamalakarKarlapalem and Basak Bharat*, Query-Driven Horizontal Class Partitioning in Object-Oriented Databases, 9th International Conference on Databases and Expert Systems (DEXA'98)(1460), edited by Lecture Notes in Computer Science , August, 1998, pp. 692-701

[27] L.Bellatreche, Techniques d'optimisation des requêtes dans les data warehouses, Sixth International Symposium on Programming and Systems, 2003, pp. 81-98

[28] L. Bellatreche, K. Karlapalem, and A. Simonet. Algorithms and support for horizontal class partitioning in object-oriented databases. in the Distributed and Parallel Databases Journal, 8(2) :155–179, April 2000.

[29] Z. A. Talebi, R. Chirkova, Y. Fathi, andM. Stallmann. Exact and inexact methodsfor selecting views and indexes for olap performance improvement. 11th InternationalConference on Extending Database Techno- logy (EDBT'08), Mars 2008.

[30] M. T. Özsu and P. Valduriez.Distributed database systems :Where are we now? IEEE COMPUTER, 24(8) :68–78, August 1991.

[31] K. Boukhalfa, L. Bellatreche, and R. P. Fragmentation primaire et dérivée : Étude de complexité, algorithmes de sélection et validation sous oracle10g. Revue des Nouvelles Technologies de l'Information RNTI, 2008.

[32] R.Bouchakri, LadjelBellatreche and Kamel Boukhalfa, UneApproche par K-means de Sélection Multiple de Structures d'Optimisation dans les Entrepôts de Données, 6èmes Journées Francophones sur les Entrepôts de Données et analyse en Ligne (EDA'10), edited by RNTI, Juin, 2010, pp. 207-222

[33] Y. Zhang and M.-E.Orlowska.On fragmentation for distributed database design.InformationSciences, 1(3) :117–132, 1994.

[34] L. Bellatreche. Utilisation des vues matérialisées, des index et de la fragmentation dans la conception logique et physique d'un entrepôt de données. Ph.d. thesis, Université de Clermont-Ferrand II, December 2000.

[35] L. Bellatreche, K. Karlapalem, and A. Simonet. Horizontal class partitioning in object oriented databases.in 8th International Conference on Database and Expert Systems Applications (DEXA'97), Toulouse, Lecture Notes in Computer Science 1308, pages 58– 67, September 1997.

[36] S. Navathe and M. Ra. Vertical partitioning for database design : a graphical algorithm. ACM SIGMOD, pages 440–450, 1989.

[37] L**.** Bellatreche, K. Boukhalfa, and H. I. Abdalla.Saga : A combination of genetic and simulated annealing algorithms for physical data warehouse design. in 23rd British National Conference on Databases, (212-219), July 2006.

[38**]** S. Ceri, M. Negri, and G. Pelagatti. Horizontal data partitioning in database design. Proceedings of the ACM SIGMOD International Conference on Management of Data.SIGPLAN Notices, pages 128–136, 1982.

[39] L. Bellatreche and K. Boukhalfa.An evolutionary approach to schema partitioning selection in a data warehouse environment.Proceeding of the International Conference on Data Warehousing and Knowledge Discovery (DAWAK'2005), pages 22–125, August 2005.

[40] T. Stöhr, H. Märtens, and E. Rahm.Multidimensional database allocation for parallel data warehouses.Proceedings of the International Conference on Very Large Databases, pages 273–284, 2000.

[41] L. Bellatreche, K. Karlapalem and M. Schneider."On Efficient Storage Space Distribution Among Materialized Views and Indices in Data Warehousing Environment." Proceedings of the Ninth International Conference on Information Knowledge Management (CIKM), McLean, VA, November 2000, pp. 397-404.

[42] J. Ferber. Les systèmes multi-agents, vers une intelligence collective. InterEditions, 1995.

[43] K. Aouiche and J. Darmont. Index and materialized view selection in data warehouses.
Handbook of Research on Innovations in Database Technologies and Applications, IGI Global, Hershey, PA, USA, 2 :693–700, February 2009.

[44] S. Papadomanolakis and A. Ailamaki. Autopart: Automating schema design for large scientific databases using data partitioning. Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004), pages 383–392, June 2004.

[45] K. Aouiche, P. Jouve, and J. Darmont."Clustering-based materialized view selection in data warehouses", In ADBIS'06, volume 4152 of LNCS, pages 81–95, 2006.

[46] T. Stöhr, H. Märtens, and E. Rahm.Multidimensional database allocation for parallel data warehouses. Proceedings of the International Conference on Very Large Databases, pages 273–284, 2000.
[47] M. Kormilitsin, R. Chirkova, Y. Fathi, and M. Stallmann.Plan-based view and index selection for query-performance improvement.Technical report, NCSU, 2008.Available at http://www.csc.ncsu.edu/research/tech/reports.php.

[48] S. Agrawal, S. Chaudhuri, and V. R. Narasayya.Automated selection of materialized views and indexes in SQL databases.In VLDB, pages 496–505, 2000.

[49] Demazeau Y., «Voyelles », Habilitation à Diriger des Recherches, INP Grenoble, 2001.

[50] Boissier O. (2001) "Modèles et architectures d'agents" *Principes et architectures des systèmes multi-agent.*, J-P Briot et Y.Demazeau, (Eds.) Hermes, Vol. 1, pp. 71-107.

[51] YANN SECQ, « RIO : rôle, interaction et organisation d'une méthodologie pour les systèmes multi-agent ouverts », Thèse doctorat.LIFL, 2003
[52] KOLSI.N,A ABDELLATIF et K.GHEDIRA. Agent based data storage and distribution in data warehouses.International Journal of Software Engineering and Knowledge Engineering Vol. 18, No. 5 (2008) 597-617.

Site Web

(1) http://www.oracle.com/
(2) www.fipa.org
(3) « documentation officiel de la plateforme Jade » http://jade.tilab.com

Annexe A Charge de requêtes

SELECT Time,Count(*) *FROM Actvars A,Prodlevel P* *WHERE A.Product=P.Code AND* *P.Class='P70j55l4hybv'* *GROUP BY Time*	*SELECT Customer,Avg(Unitssold)* *FROM Actvars A,Prodlevel P* *WHERE A.Product=P.Code AND* *P.Family='Ixg0c4hqg2dw' AND* *P.Group='P66l03psafvq'* *GROUP BY Customer*
SELECT Max(Time) *FROM Actvars A,Prodlevel P* *WHERE A.Product=P.Code AND* *P.Group='P66l03psafvq'*	*SELECT Product,Count(*)* *FROM Actvars A,Timelevel T* *WHERE A.Time=T.Tid AND T.Month='1'* *AND* *(T.Quarter='Q3' Or T.Quarter='Q4')* *GROUP BY Product*
SELECT Time,Avg(Unitssold) *FROM Actvars A,Timelevel T* *WHERE A.Time=T.Tid AND (T.Quarter='Q1'* *Or* *T.Quarter='Q2')* *GROUP BY Time*	*SELECT Avg(Unitssold)* *FROM Actvars A,Prodlevel P* *WHERE A.Product=P.Code AND* *P.Family='Urt1b3vosshh' AND* *P.Group='P66l03psafvq'*
SELECT Division,Count(*) *FROM Actvars A,Prodlevel P* *WHERE A.Product=P.Code AND* *P.Group='S7jweujryiwn'* *GROUP BY Division*	*SELECT Count(*)* *FROM Actvars A,Timelevel T* *WHERE A.Time=T.Tid AND T.Year='1995'*
SELECT Time,Count(*) *FROM Actvars A,Prodlevel P* *WHERE A.Product=P.Code AND* *P.Family='Urt1b3vosshh'* *GROUP BY Time*	*SELECT Avg(Unitssold)* *FROM Actvars A,Timelevel T* *WHERE A.Time=T.Tid AND T.Year='1996'*
SELECT Year,Sum(Dollarcost) *FROM Actvars A,Prodlevel P, Timelevel T* *WHERE A.Product=P.Code AND* *A.Time=T.Tid* *AND P.Group='P66l03psafvq'* *GROUP BY Year*	*SELECT Product,Count(*)* *FROM Actvars A,Timelevel T* *WHERE A.Time=T.Tid AND T.Year = '1995'* *GROUP BY Product*
SELECT Retailer,Avg(Unitssold) *FROM Actvars A,Prodlevel P ,Custlevel C* *WHERE A.Product=P.Code AND* *A.Customer=C.Store AND P.Division =* *'E6qf1ihdev0e'* *GROUP BY Retailer*	*SELECT Count(*)* *FROM Actvars A,Prodlevel P* *WHERE A.Product=P.Code AND P.Division* *=* *'E6qf1ihdev0e'*
SELECT Customer, Time,Min(Unitssold) *FROM Actvars A,Timelevel T, Custlevel C* *WHERE A.Customer=C.Store AND* *A.Time=T.Tid* *AND (T.Quarter='Q3' Or T.Quarter='Q4')* *AND* *C.Gender='M'* *GROUP BY Customer, Time*	*SELECT Division,Sum(Dollarcost)* *FROM Actvars A,Timelevel T ,Prodlevel P* *WHERE A.Time=T.Tid AND* *A.Product=P.Code* *AND (T.Month='1' Or T.Month='2')* *AND P.Class='P70j55l4hybv'* *GROUP BY Division*
SELECT Month, Avg(Unitssold) *FROM Actvars A,Timelevel T, Custlevel C* *WHERE A.Time=T.Tid AND*	*SELECT Count(*)* *FROM Actvars A,Chanlevel Ch,Timelevel T* *WHERE A.Channel=Ch.Base AND*

A.Customer=C.Store *AND C.Gender='M'* *GROUP BY Month*	*A.Time=T.Tid* *AND (T.Quarter='Q1' Or T.Quarter='Q2')* *AND* *T.Year='1995' AND Ch.All ='Bcdefghijklm'*
SELECT Product, Sum(Dollarcost) *FROM Actvars A,Timelevel T, Prodlevel P* *WHERE A.Time=T.Tid AND* *A.Product=P.Code* *AND (T.Month='1' Or T.Month='2') AND* *(T.Quarter='Q1' Or T.Quarter='Q2') AND* *P.Division = 'E6qf1ihdev0e'* *GROUP BY Product*	*SELECT Year, Max(Unitssold)* *FROM Actvars A,Prodlevel P, Custlevel C,* *Timelevel T* *WHERE A.Time=T.Tid AND* *A.Customer=C.Store* *AND A.Product=P.Code AND P.Division =* *'E6qf1ihdev0e' AND C.Retailer* *='Zstv6mycbs7u'* *GROUP BY Year*
SELECT Product,Time,Avg(Unitssold) *FROM Actvars A,Timelevel T, Custlevel C* *WHERE A.Time=T.Tid AND* *A.Customer=C.Store* *AND (T.Month='1' Or T.Month='2') AND* *C.Gender='F'* *GROUP BY Product,Time*	*SELECT Year,Month, Max(Unitssold)* *FROM Actvars A,Prodlevel P ,Timelevel T* *WHERE A.Product=P.Code AND* *A.Time=T.Tid* *AND (T.Month='1' Or T.Month='2') AND* *P.Division = 'E6qf1ihdev0e'* *GROUP BY Year,Month*
SELECT Channel, Sum(Dollarcost) *FROM Actvars A,Custlevel C,Prodlevel P* *WHERE A.Customer=C.Store AND* *A.Product=P.Code AND* *P.Class='P70j55l4hybv'* *AND C.Retailer='M5tahn7gumlt' AND* *C.Gender='F'* *GROUP BY Channel*	*SELECT Class, Month, Min(Unitssold)* *FROM Actvars A,Custlevel C, Prodlevel P,* *Timelevel T* *WHERE A.Product=P.Code AND* *A.Customer=C.Store AND A.Time=T.Tid* *AND* *C.City='Dijon' AND C.Retailer* *='Zstv6mycbs7u'* *GROUP BY Class, Month*
SELECT Count(*) *FROM Actvars A,Custlevel C* *WHERE A.Customer=C.Store AND* *C.City='Dijon'*	*SELECT Max(Unitssold)* *FROM Actvars A,Chanlevel Ch* *WHERE A.Channel=Ch.Base AND Ch.All* *='Efghijklmnop'*
SELECT Time, Avg(Unitssold) *FROM Actvars A,Prodlevel P, Custlevel C* *WHERE A.Customer=C.Store AND* *A.Product=P.Code AND* *P.Group='S7jweujryiwn'* *AND C.Retailer ='Zstv6mycbs7u'* *GROUP BY Time*	*SELECT Time,Count(*)* *FROM Actvars A,Chanlevel Ch,Custlevel C* *WHERE A.Channel=Ch.Base AND* *A.Customer=C.Store AND C.City='Dijon'* *AND* *Ch.All ='Efghijklmnop'* *GROUP BY Time*
SELECT Sum(Dollarcost) *FROM Actvars A, Prodlevel P,Timelevel T* *WHERE A.Product=P.Code AND* *A.Time=T.Tid* *AND (T.Quarter='Q1' Or T.Quarter='Q2')* *AND* *P.Division = 'Phpet5vw6slg'*	*SELECT Sum(Dollarcost), Avg(Unitssold)* *FROM Actvars A, Custlevel C,Timelevel T* *WHERE A.Customer=C.Store AND* *A.Time=T.Tid* *AND (T.Month = '1' Or T.Month = '2') AND* *C.Retailer ='Zstv6mycbs7u' AND* *C.City='Dijon'*
SELECT Count(*) *FROM Actvars A,Custlevel C* *WHERE A.Customer=C.Store AND*	*SELECT Avg(Dollarcost)* *FROM Actvars A,Custlevel C* *WHERE A.Customer=C.Store AND*

C.Gender='F'	*C.City='Dijon' AND C.Gender='M'*
SELECT Time, Max(Unitssold) *FROM Actvars A, Chanlevel H,Custlevel C,Timelevel T* *WHERE A.Customer=C.Store AND A.Channel=H.Base AND A.Time=T.Tid AND T.Year = '1996' AND (T.Month = '1' Or T.Month* *= '2') AND C.City='Paris' AND C.Retailer ='Zstv6mycbs7u' AND H.All='Bcdefghijklm'* *GROUP BY Time*	*SELECT Month, All, Time, Sum(Dollarcost)* *FROM Actvars A, Custlevel C,Timelevel T ,Chanlevel H* *WHERE A.Customer=C.Store AND A.Channel=H.Base AND A.Time=T.Tid AND T.* *Month='2' AND T.Year='1995' AND C.Retailer* *='Coghfprojp9z' AND (C.City='Paris'or C.City='Poitiers')* *GROUP BY Month,All, Time*
SELECT Count(*) *FROM Actvars A,Chanlevel Ch,Timelevel T* *WHERE A.Channel=Ch.Base AND A.Time=T.Tid* *AND (T.Quarter='Q1' Or T.Quarter='Q2') AND* *T.Month = '7' AND Ch.All ='Bcdefghijklm'*	*SELECT Division,Avg(Unitssold)* *FROM Actvars A,Timelevel T, Prodlevel P* *WHERE A.Time=T.Tid AND* *A.Product=P.Code* *AND T.Month='7'* *GROUP BY Division*
SELECT Product, Count(*) *FROM Actvars A,Chanlevel Ch, Custlevel C* *WHERE A.Channel=Ch.Base AND A.Customer=C.Store AND Ch.All ='Bcdefghijklm'* *AND C.Retailer ='Zstv6mycbs7u'* *GROUP BY Product*	*SELECT Count(*)* *FROM Actvars A,Chanlevel Ch,Timelevel T, Custlevel C* *WHERE A.Channel=Ch.Base AND A.Time=T.Tid* *AND A.Customer=C.Store AND C.Gender='M'* *AND T.Month = '7' AND Ch.All ='Bcdefghijklm'*
SELECT Count(*) *FROM Actvars A, Prodlevel P,Timelevel T, Custlevel C* *WHERE A.Product=P.Code AND A.Customer=C.Store AND A.Time=T.Tid AND* *P.Group='S7jweujryiwn' AND (T.Quarter='Q3'* *Or T.Quarter='Q4') AND C.Gender='M'*	*SELECT Month, Sum(Dollarcost)* *FROM Actvars A,Custlevel C, Timelevel T, Prodlevel P* *WHERE A.Customer=C.Store AND A.Product=P.Code AND A.Time=T.Tid AND C.Gender='M' AND* *C.Retailer='Zstv6mycbs7u'* *AND P.Family='Urt1b3vosshh'* *GROUP BY Month*
SELECT Product, Sum(Dollarcost) *FROM Actvars A,Timelevel T, Prodlevel P, Chanlevel H* *WHERE A.Time=T.Tid AND A.Product=P.Code* *AND A.Channel=H.Base AND T.Year = '1996'* *AND P.Class= 'P70j55l4hybv' AND H.All='Efghijklmnop' AND P.Family='Hnmc1go57w3y'* *GROUP BY Product*	*SELECT Min(Unitssold)* *FROM Actvars A, Chanlevel H,Custlevel C,Prodlevel P,Timelevel T* *WHERE A.Customer=C.Store AND A.Product=P.Code AND A.Channel=H.Base AND* *A.Time=T.Tid AND (T.Month = '1' Or T.Month =* *'2') AND T.Year='1996' AND P.Family='Ixg0c4hqg2dw' AND H.All='Bcdefghijklm' AND T.Quarter='Q2'*
SELECT Count(*)	*SELECT Gender,Max(Dollarcost)*

FROM Actvars A, Custlevel C, Prodlevel P *WHERE A.Product=P.Code AND* *A.Customer=C.Store AND* *P.Family='Hnmc1go57w3y' AND* *P.Division = 'E6qf1ihdev0e' AND* *P.Group='P66l03psafvq' AND C.City='Dijon'*	*FROM Actvars A,Custlevel C, Prodlevel P* *WHERE A.Customer=C.Store AND* *A.Product=P.Code AND* *C.Retailer='Zstv6mycbs7u' AND* *P.Division = 'E6qf1ihdev0e'* *GROUP BY Gender*
SELECT Max(Dollarcost) *FROM Actvars A, Prodlevel P* *WHERE A.Product=P.Code AND* *P.Family='Hnmc1go57w3y' AND P.Division* *=* *'E6qf1ihdev0e'*	*SELECT Sum(Dollarcost)* *FROM Actvars A, Timelevel T, Custlevel C* *WHERE A.Time=T.Tid AND* *A.Customer=C.Store* *AND C.City='Dijon' AND* *(T.Quarter='Q1' Or T.Quarter='Q2')*
SELECT Product, Sum(Dollarcost) *FROM Actvars A,Timelevel T, Prodlevel P* *WHERE A.Time=T.Tid AND* *A.Product=P.Code* *AND (T.Quarter='Q1' Or T.Quarter='Q2')* *AND* *P.Group='S7jweujryiwn'* *GROUP BY Product*	

Annexe B Plateforme JADE

1. Plateforme JADE

Nous avons opté dans le cadre du développement de notre outil et plus précisément la partie qui traite des systèmes multi-agents, d'utiliser une plateforme trés connue, qui a fait ses preuves : JADE.

JADE (Java Agent DEvelopement Framework) est une plateforme développée par TILAB en 1999, codée entièrement en JAVA, dans le but de faciliter le développement de systèmes multi-agents et applications conformément au standard FIPA. La plateforme Jade se compose principalement de deux composants : une plateforme agents compatible FIPA et un package pour le développement d'agents Java. Cette plateforme est fournie avec une License open source LGPL.

2. Packages jade

Jade est composé de plusieurs packages, principalement :

Jade.core : Le noyau du système, jade.core implémente la classe agent qui est la classe principale pour l'héritage d'objet de type agent. La classe Behaviour qui est la classe permettant de modéliser le comportement des agents.

Jade.lang.acl : ce package inclut les classes de communication d'agents conformément aux standards FIPA.

Jade.domain : Contient l'ensemble des classes qui représentent les agents gestionnaires de la plateforme, tels que les agents AMS et DF qui permettent respectivement de gérer le cycle de vie d'un agent, et de fournir un service de pages jaunes. Les pages jaunes répertorient l'ensemble des agents ainsi que les différents services proposés par chaque agent.

Jade.gui : l'ensemble de classes permettant de fournir des interfaces graphiques pour la gestion des agents.

Jade.mtp : ce package fournie des interfaces que n'importe quel protocole de transport de message est sensé implémenter afin de pouvoir interagir avec la plate forme jade.

Jade.proto : Contient l'ensemble des classes qui servent à modéliser les protocoles d'interactions standards.

D'autres packages existent tels que jade.wrapper ou FIPA.

3. Outils Jade

En plus de l'organisation en packages, jade est fournie avec un ensemble d'outils de gestion, chacun incluant des sous packages de jade.tools et qui sont à leur tour des agents. Ces outils sont :

- L'agent RMA (Remote Management Agent) : C'est une console graphique permettant le contrôle et la gestion de différentes plateformes et conteneurs.

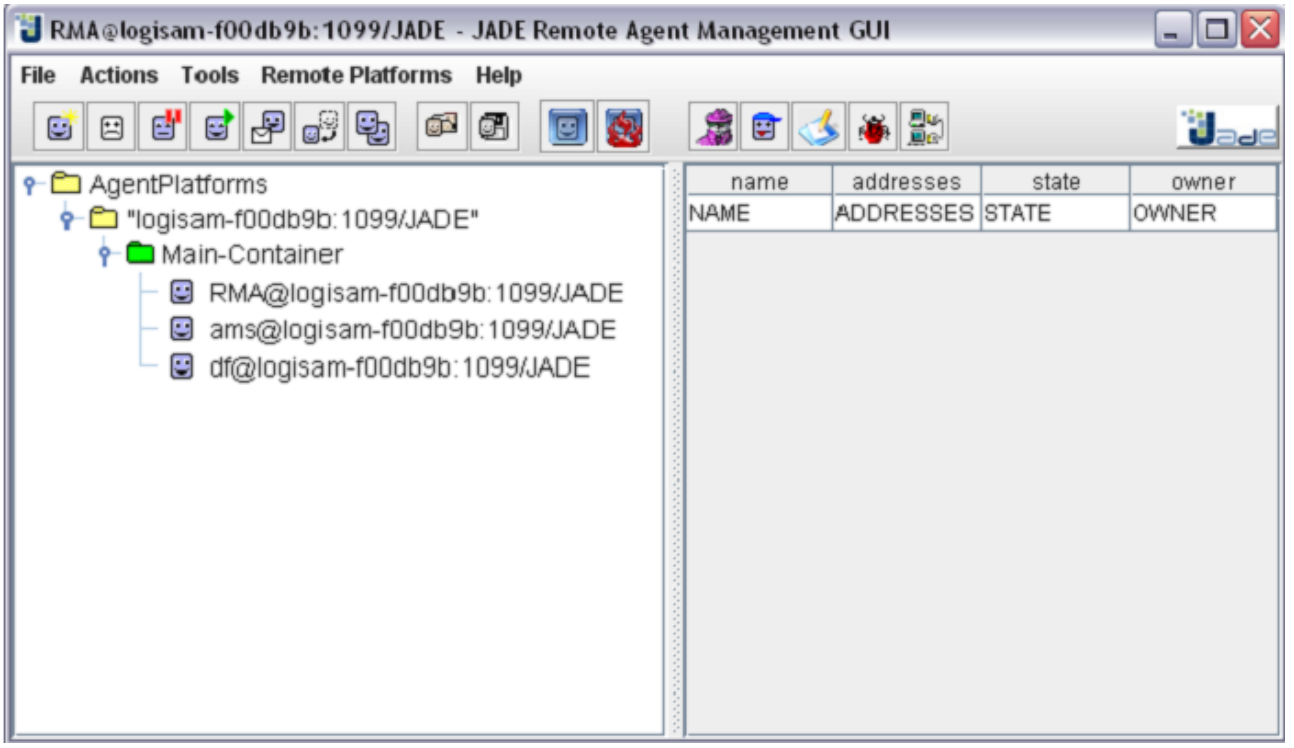


Figure B -1 : L'agent RMA de la plateforme Jade

L'agent RMA permet de contrôler le cycle de vie de la plateforme et ses agents.

- **L'agent Dummy** : C'est l'outil de débogage et de monitoring, ce dernier est en faite un agent avec une interface graphique permettant l'envoi de message ACL et la surveillance des messages échangés entre agents avec toutes les informations relatives.

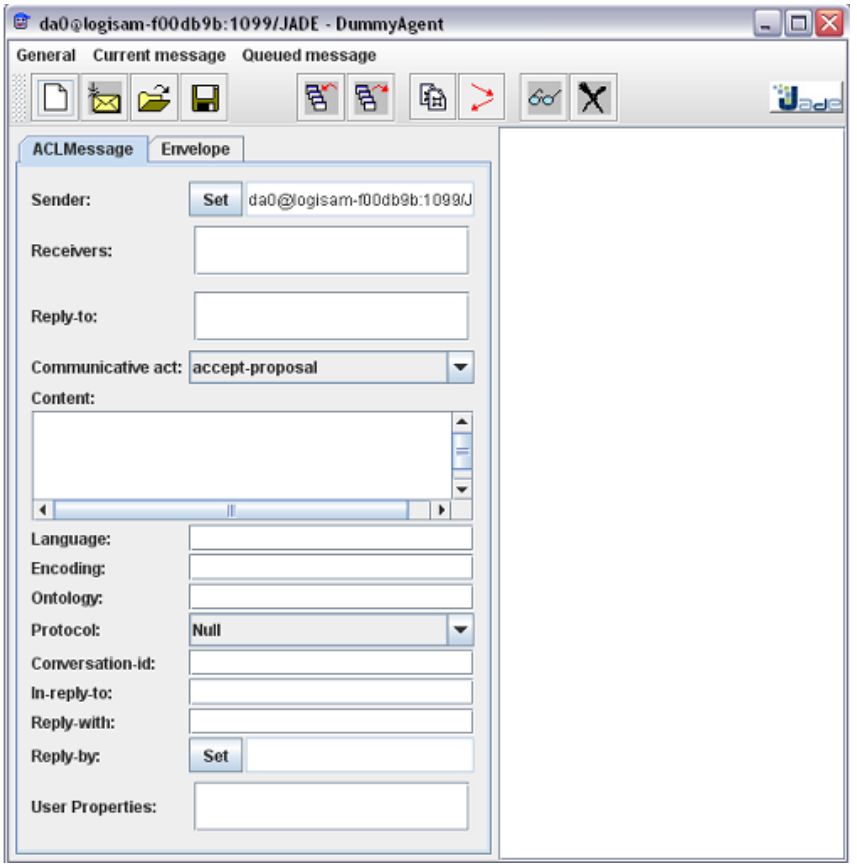


Figure B -2 : L'agent dummy de la plateforme Jade.

L'agent Sniffer : Permet de modéliser graphiquement tous les échanges de messages avec une notation très proche de celle utilisée sous UML. Cette modélisation des messages est très pratique pour le débogage des agents en observant l'échange de messages ACL.

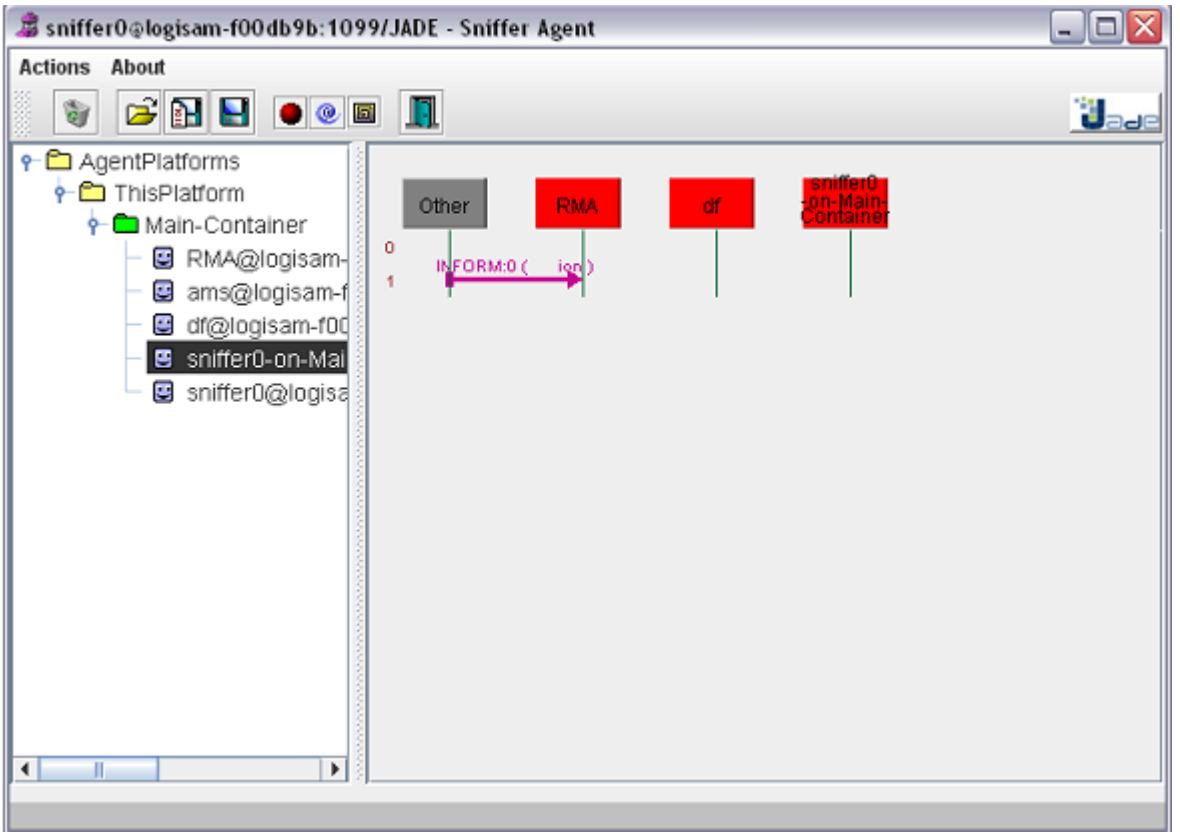


Figure B -3 : L'agent Sniffer de la plateforme Jade.

- **L'agent Introspector :** C'est l'agent permettant de surveiller le cycle de vie des autres agents, les messages échangés et les tâches en exécution.

- **L'agent DF (Directory Facilitator)** : Cet agent permet à l'aide d'interface graphique de gérer des domaines de pages jaunes. L'utilisateur peut même créer un réseau complexe de domaines et de sous domaines de pages jaunes.

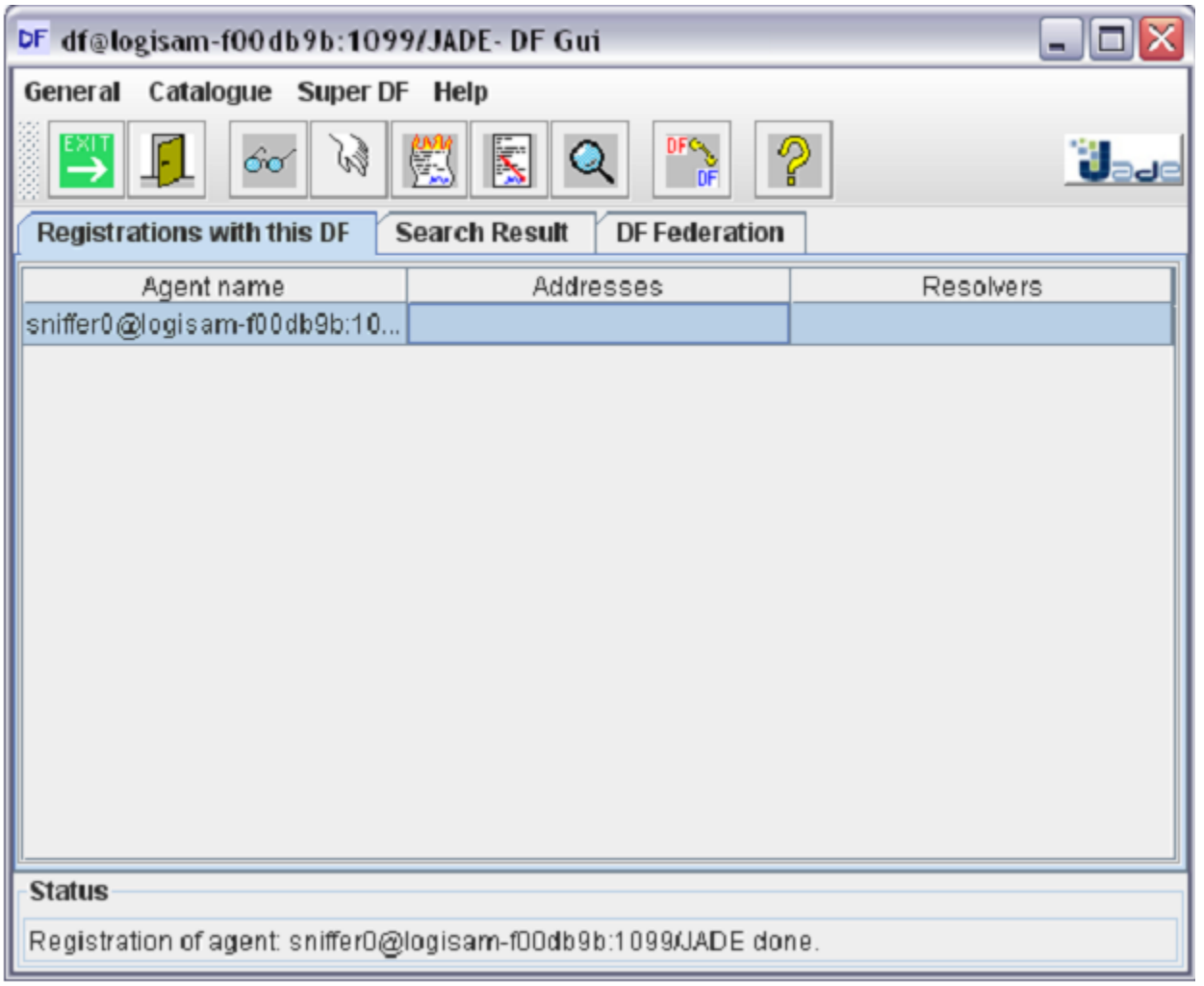


Figure B -4 : L'agent DF de la plateforme Jade.

- **L'aget LogManager** : Permet de définir les attributs et les niveaux de journalisation d'événements.

- **L'agent SocketProxy** : Cet agent agit de manière bidirectionnelle entre la plate forme Jade et les connexions TCP IP. A titre d'exemple les messages ACL transmis par la plate forme sont d'abord convertit en messages ASCII puis envoyés à travers des sockets et vice-versa. Cette conversion est à la charge de cet agent.

4. Les conteneurs

Une plateforme est composée de plusieurs instances de l'environnement d'exécution Jade.

Ces instances sont appelées conteneurs. Chaque conteneur peut contenir plusieurs agents et peut être dans différents états (actifs, suspendu …etc.). L'ensemble des conteneurs actifs constitue la plateforme. Un conteneur particulier doit être toujours actif, c'est le conteneur principal (main-container).

L'ensemble des conteneurs peut être distribué sur un réseau permettant ainsi d'avoir une plateforme multi agents distribuée. Cette fonctionnalité est permise grâce au service d'appel de méthodes à distance (Remote Methode Invocation), qui doit être lancé sur les différentes machines sur lesquelles la plateforme est distribuée.

Les conteneurs prennent en charge les tâches de gestion des cycles de vie de leurs agents et la communication (envoie et réception de message, gestion des files d'attentes). Le conteneur garde des informations relatifs à tout ses agents lui permettant de les identifier, les invoquer …etc.

Une gestion distante est possible grâce à l'interface graphique (GUI). Cette interface permet la gestion complète du conteneur et offre un contrôle complet sur ce dernier, que ce soit en local ou à distance

5. **Les comportements (behaviours)**

Les agents ont des comportements qui se modélisent par des tâches à accomplir. Chaque tâche est un behaviour qui est une classe contenant principalement deux méthodes

- Action() : qui définie les opérations à exécuter.

- Done () : qui retourne un booléen indiquant si la tâche est terminée.

Différents types de behaviours existent à titre d'exemple :

- Cyclic behaviour : C'est une tâche répétitive, qui ne se termine 'jamais', sa méthode done() retourne toujours faux.

- OneShotBehaviour : Sa méthode action() est exécutée seulement une fois et sa méthode done() retourne vrai.

6. **Communication inter-agents**

La communication entre les agents est un concept clé de la plateforme Jade.

Jade propose un package permettant d'hériter un message qui n'est rien d'autre qu'un objet avec des attributs.

```
ACLMessage message= new ACLMessage(ACLMessage.REQUEST);

message.addReceiver (new AID("Agent_1",AID.ISLOCALNAME));

message.setOntology("Etat_serveur");

message.setContent("En_Marche");

send(message);
```

Une fois le message envoyé, il sera mis dans une file d'attente jusqu'à ce que son (ses) destinataire(s) le récupère.

- **Réception de messages :** La réception du message est en fait une lecture de la file d'attente. Cette lecture se fait par un appel de la méthode receive().

```
MessageTemplate temp = MessageTemplate.and

(MessageTemplate.MatchPerformative(ACLMessage.REQUEST),

ACLMessage msg = myAgent.receive(temp);
```

Nous pouvons voir qu'avec la plate forme Jade, nous disposons de l'ensemble des outils nécessaire à la l'implémentation de notre solution.