dF.S.D … N° D'ordre : ……..

Université Saâd DAHLAB de Blida

Faculté des sciences

Département d'Informatique

Mémoire présenté par :

M^lle^ **BENKHALED Sihem et AID Soumia**

Pour l'obtention du diplôme de Master

Domaine : Mathématique et Informatique

Filière : Informatique

Spécialité : Ingénierie des logiciels

Thème : Optimisation Des Requêtes Dans Un Système De Médiation Des Données Hétérogènes en utilisant l'optimisation combinatoire

Soutenu le : Septembre 2015, devant le jury composé de :

Promotrice : M^me^ **FAREH**

Président :Mme Toubaline

Examinateur : Mr Derrar

Examinateur :Mr Zair

REMERCIEMENTS

Nous remercions avant tout ELLAH le tout puissant qui nous a données la volenté et la patience pour accomplir ce travail.

Nous souhaitons adresser nos remerciements les plus sincères à notre promotrice Mme FAREH pour sa précieuse aide, sa patience, sa générosité et disponibilité et surtout son soutien morale.

Nous tenons à remercier aussi tous nos enseignants qui nous ont facilité la compréhension et la maîtrise.

Nous tenons à exprimer notre profonde gratitude aux membres de jury qui nos font le grand honneur d'avoir accepté de juger notre travail.

Enfin un grand merci à nos amis et collègues et tous ceux qui ont contribué de près ou de loin à la réalisation de ce mémoire.

Sihem & Soumia

Dédicace

Je dédie ce travail à mes très chers parents qui m'ont toujours soutenu pendant toute ma vie, je leur souhaite le bonheur et la bonne santé J'espère que vous seriez toujours fiers de moi ;

Je tiens à remercier également toute la famille derrière moi qui m'ont supporté depuis ces années;

A mes chers frères Oussama , Amir.

A mes chères soeurs. .

A tous mes amies sans exception.

A mon binôme BENKHALED Sihem et à toute sa famille ;

A notre promotrice Mme Fareh. ;

A tous ceux que j'aime et qui m'aiment

SOUMIA

Dédicace

Je dédie ce travail à mes très chers parents qui m'ont toujours soutenu pendant toute ma vie, je leur souhaite le bonheur et la bonne santé J'espère que vous seriez toujours fiers de moi ;

Je tiens à remercier également toute la famille derrière moi qui m'ont supporté depuis ces années;

A mon cher frères Zakaria .

A mes chères soeurs.

A tous mes amies sans exception.

A mon binôme AID Soumia et à toute sa famille ;

A notre promotrice Mme Fareh. ;

A tous ceux que j'aime et qui m'aiment

Sihem

ملخص

في وقتنا الحالي **أنظمة الوساطة** جدا معروفة و متطورة. ومع ذلك، فإن استخدامها يطرح عددا من المشاكل، لاسيما من حيث مشكلة زمن الاستجابة ل**لاستعلام**. بشكل عام، هنالك العديد من الخطط لتشغيل نفس الاستعلام بعضها يكون أسرع بكثير من البعض الآخر، المشكل الذي يطرح نفسه هو كيفية اختيار **خطة التنفيذ المثلى والفعالة**.

الهدف الرئيسي من هذا المشروع هو تنفيذ الاستعلامات الأكثر فعالية في نظام الوساطة **الغير المتجانسة** و **الموزعة** من اجل تقليل زمن الاستجابة للمستخدمين. وتقليل تكاليف الاتصالات الإجمالية المرتبطة بالاستعلام باستخدام **الاندماجي الأمثل**.

الكلمات المفتاحية: أنظمة الوساطة, استعلام, خطة التنفيذ المثلى, الغير المتجانسة, الموزعة, الاندماجي الأمثل .

Résumé

De nos jours, les **systèmes de médiation** sont de plus en plus développés et connus. Cependant, leur mise en oeuvre pose un certain nombre de problèmes en particulier le problème de performances en termes de temps de réponse de **requête**.

En général, il existe de nombreux plans pour exécuter la même requête, parmi lesquelles certaines sont souvent beaucoup plus rapides que les autres, la problématique qui se pose c'est comment choisir le **plan d'exécution optimal** et efficace.

Le but principal de ce projet est d'exécuter les requêtes les plus efficacement possibles dans un système de médiation **hétérogène** et **distribué** afin de minimiser le temps de réponse aux utilisateurs. Et de minimiser les coûts de communication totale associée à une requête en utilisant l'**optimisation combinatoire**.

Mots Clés: systèmes de médiation, requête, plan d'exécution optimal, hétérogène, distribué, optimisation combinatoire.

Abstract

Nowadays, **mediation systems** are increasingly developed and known. However, their use poses a number of problems, in particular the problem in terms of performance of **query** response time,In general, there are many plans to run the same query, some of which are often much faster than others, the problem which arises is how to choose the efficient and **optimal execution plan**.

The main purpose of this project is to run the most efficient possible queries in a **heterogeneous** and **distributed** mediation system to minimize the response time for users. And minimize the total communication costs associated with a query using **combinatorial optimization**.

Keywords: Mediation systems, query, optimal execution plan, heterogeneous, distributed, combinatorial optimization.

Sommaire

Chapitre 5 : Implémentation

Chapitre 6 : Test et Validation du système

LES TABLEAUX

LES FIGURES

Introduction Générale

Depuis plus de quarante années, l'accès aux données demeure toujours un sujet important dans le monde informatique. Beaucoup de travaux ont été réalisés pour faciliter l'accès aux diverses données réparties partout dans le monde, d'une manière efficace et stable. Des entreprises ou établissements ont développé les systèmes d'informations pour stocker, interroger et mettre à jour les données. Chacun de ces systèmes dispose d'une façon pour définir le format de données et surtout un mode d'accès aux données permettant aux utilisateurs d'interagir avec les données. Ce type de système d'information est appelé « **source de données** ».

L'intérêt de faire collaborer des systèmes ne cesse de croître. Aujourd'hui différentes entreprises et organisations doivent faire face au challenge de faire interopérer des systèmes de bases de données afin de pouvoir réaliser des fonctions critiques. Ces entreprises et organisations possèdent une grande latitude pour définir le format des données et le mode d'accès les plus adéquats de leurs données.

Problématique

Le système d'information est construit grâce à plusieurs méthodes et technologies : plateformes, protocoles de communication, environnements et outils de développement, modèles de conception des données, types de SGBD, etc., la complexité pour faire interopérer des systèmes d'information réside au niveau de cette diversité de méthodes de conception et de technologies d'implémentation.

Nous nous intéressons dans ce projet par les systèmes de médiation des données hétérogènes. Notre problématique peut se résumer comme suit : comment traiter et optimiser le temps de réponse d'une requête dont les éléments de réponse se trouvent dispersés sur différents systèmes d'informations distribuées et hétérogènes

Malgré le nombre important de travaux effectués, les solutions proposées restent insuffisantes et la problématique continue à rester d'actualité.

Objectif

L'optimisation combinatoire occupe une place très importante en mathématique et en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation.

Le but de ce travail est d'optimiser des requêtes sur un système de médiation de source de donnée hétérogène et autonome. Pour permettre au médiateur de choisir le plan d'exécution le plus optimal pour une requête donnée et de sélectionner les sources concernées par une requête et évite l'envoi de la requête à toutes les sources. Un outil doit développer, permettant d'optimiser l'exécution des requêtes dans un système de médiation, Parmi un ensemble d'alternatives qui satisfont une certaine propriété, choisir celle qui optimise une certaine fonction de coût en utilisant l'optimisation combinatoire.

Pour réaliser notre objectif et arriver à concevoir un système d'optimisation de requêtes dans les systèmes de médiation, nous avons passé par plusieurs étapes. Ces étapes sont présentées dans les chapitres ci-dessous :

Chapitre I: Médiation des Données Qui permet d'obtenir une bonne compréhension des aspects caractéristiques de la médiation de données ainsi que les différents composants de ces systèmes.

Chapitre II: Etat de l'art des approches existantes pour l'optimisation des requêtes Dans ce chapitre nous présentons quelques définitions et approches concernant le domaine l'optimisation combinatoire.

Chapitre III: La solution proposée Dans ce chapitre nous présentons une solution en utilisant l'optimisation dynamique et combinatoire.

Chapitre IV : Conception du Système Ce chapitre comporte la conception de notre système d'optimisation de requêtes dans les systèmes de médiation en utilisant l'optimisation combinatoire. Pour modéliser notre système nous avons utilisé le langage de modélisation UML.

Chapitre V : Implémentation du Système Dans ce chapitre nous décrirons les différents outils utilisés pour la réalisation de notre projet, ensuite nous présentons une démonstration globale des différentes fonctionnalités de notre application.

Chapitre VI: Tests et Validation du Système Ce chapitre vise à démontrer que notre système répond à l'objectif souhaité par notre projet.

Chapitre 1 :

Médiation des données

1 Introduction

L'intégration d'informations est un domaine de recherche qui a pour but de proposer des architectures pour la construction de systèmes de questions/réponses sur des informations distribuées et hétérogènes provenant de multiples sources de données pouvant être disparates. Ces architectures de données permettent aux utilisateurs d'accéder travers un schéma global unifié à plusieurs sources de données ayant chacune un schéma local. Ces sources de données sont le plus souvent réparties, autonomes et hétérogènes.

- Les sources d'informations sont réparties : de plus en plus d'informations sont-crées partout dans le monde et publiées sur le Web, de nombreuses entreprises ont des ramifications dans plusieurs pays et les états décentralisent leurs administrations.
- Elles sont autonomes car les sources de données sont conçues par différentes personnes, a différents moments et pour répondre a différents besoins applicatifs.
- Enfin, les sources d'informations sont hétérogènes : des logiciels différents sont utilises pour créer et gérer les données (Oracle, MySQL, SQL Server), les données sont publiées dans des formats divers (HTML, PDF,…etc.) et des modèles de données différents sont utilisés pour les représenter (Relationnel, Objet, Semi structure).

De ce fait, l'accès « transparent » aux ressources et de manière plus générale à L'information constitue un des challenges actuels majeurs de l'informatique.

Pour cela différentes approches d'intégrations ont été mis en œuvre pour assurer la cohérence des données et d'homogénéiser les différents formats trouvés. Les systèmes de médiation offrent des solutions intéressantes pour ce type de défis.

2. Médiations des données

2.1. Aspect caractéristique de la médiation de données

Le problème de la médiation peut être résumé de la manière suivante : étant donné un ensemble de sources (niveau local) et d'applications, il s'agit de construire une infrastructure intermédiaire facilitant l'accès (expression, traitement et optimisation de requêtes) et la manipulation des données (niveau global). La médiation des données est guidée par trois aspects : la distribution et l'hétérogénéité des données, l'autonomie des sources, et l'interopérabilité des systèmes qui gèrent les bases.

2.2 Systèmes d'intégration de données

Les systèmes d'intégration de données offrent des architectures d'interopérabilité sur une fédération de sources de données distribuées, autonomes et hétérogènes. Les entrepôts de données, les systèmes de médiation et les architectures P2P sont des exemples d'infrastructures qui permettent l'intégration de données, c'est-à-dire l'accès a des données produites par des sources autonomes. A travers des schémas virtuels, des méta données et des correspondances sémantiques, ils permettent d'accéder a ces sources de données de façon uniforme et transparente, en transformant par réécriture les requêtes d'un utilisateur en sous requêtes envoyées aux sources de données les plus appropriées. L'hétérogénéité des données extraites des sources nécessite leur réconciliation, en d'autres termes, leur mise en correspondance par rapport au schéma global, avant de les présenter a l'utilisateur.

2.2.1. Définition & Composante :

Un système d'intégration de données fournit une vue unifiée de données provenant de sources multiples et hétérogènes. Il permet d'accéder à ces données à travers d'une interface uniforme, sans se soucier de leur structure ni de leur localisation [BAR 03].

Formellement, un système d'intégration de données est un triplet I : <G, S, M>, où :

G - est le schéma global (défini sur un alphabet AG) modélisant le schéma intégré.

S - est l'ensemble des schémas des sources (définis sur un alphabet AS) .

M - est une correspondance entre G et S qui établit la connexion entre les éléments du schéma global et ceux des sources.

Tout système d'intégration de données doit considérer l'intégration à deux niveaux:

1 -le niveau schéma: qui consiste à consolider tous les schémas des sources en un seul schéma global, ou schéma de médiation, qui sera utilisé pour supporter les requêtes.

2 -le niveau donné (population du schéma intégré).

Un système d'intégration se compose de deux parties :

_ Une partie (1) externe et correspond aux utilisateurs du système intègre (décideurs) ou autres systèmes.

_ Une partie (2) interne et comprend des sources d'informations et une interface uniforme qui permet a la partie externe d'interroger d'une manière transparente les sources de données, comme s'il n'y avait qu'une source unique.

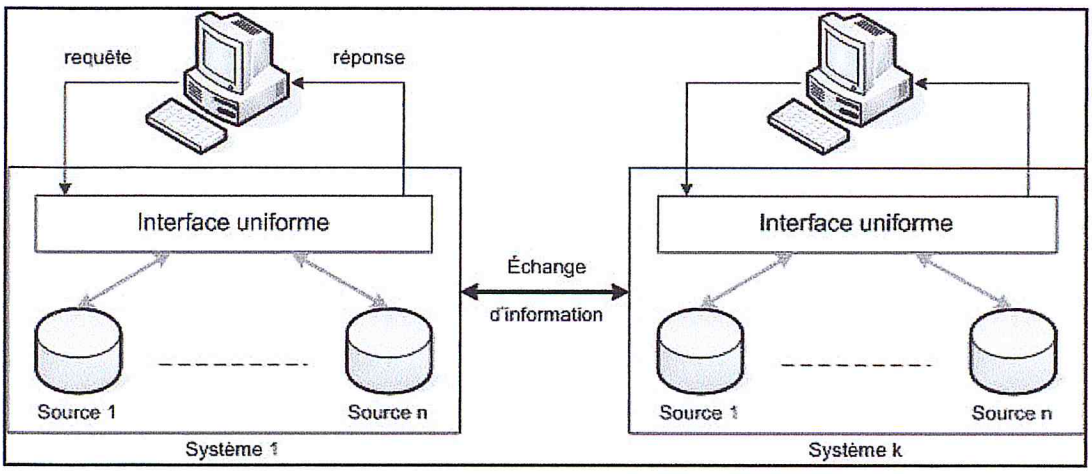


Figure 1: Système d'intégration d'information **[RAH 05]**.

2.2.2 Processus d'un system de médiation

Etant donne un ensemble de sources de données hétérogènes {S1, S2, …, Sn}, le problème d'intégration consiste à construire un schéma intégré (ou un schéma global) à partir des schémas locaux .Ce problème est lié au fait que les sources stockent des différents types de données, en différent format, ayant différentes significations et associées aux différents noms. Il convient d'abord d'indiquer qu'il existe plusieurs méthodologies permettant l'intégration des bases de données classiques.

Le processus d'intégration est ainsi décomposé en trois phases distinctes :

- **La pré -intégration** : Cette phase vise à préparer l'intégration des schémas en les rendant plus homogènes. Elle consiste à traduire les schémas initiaux dans un modèle de données commun.

- **L'identification des correspondances** : Durant cette phase, les correspondances entre les éléments des schémas source sont détectées et formalisées de même que les différents conflits.

- **L'intégration** : Cette phase finale produit le schéma intègre et fournit les règles de traduction permettant de passer des schémas source au schéma intègre et inversement (mapping).

2.2.3. Tâche d'un system de médiation

On peut distinguer quatre taches principales d'un système d'intégration. Les Deux premières concernent la traduction de données provenant de sources différentes et Résolvent le problème de l'hétérogénéité physique/logique des sources en fournissant une Interface d'accès uniforme. Les deux dernières sont des taches d'intégration sémantique et Résolvent le problème de l'hétérogénéité sémantique en reliant chaque source au schéma Global. Ces quatre taches sont décrites ci-après :

- **Transformation de données**: Un exemple typique de cette tache est la transformation de Données relationnelles en XML et inversement. Les

problèmes importants qui doivent être Résolus à ce niveau sont la perte d'information, la taille des données générées et la Performance des traitements sur ces données.

- **Traduction de requêtes:** La traduction de requêtes d'un langage (exp. XQuery) en un autre langage (exp. SQL) est liée au problème de transformation de données. Elle doit également prendre en compte la puissance d'expression du langage cible et nécessite souvent des extensions spécifiques afin d'obtenir la puissance d'expression du langage source.
- **Réécriture de requêtes:** Cette tache est différente de la tache précédente et généralement plus complexe car elle doit prendre en compte l'hétérogénéité structurelle et sémantique entre les schémas. Elle joue un rôle primordial dans l'intégration de données sur le Web.
- **Fusion de données:** La fusion de données essaye de répondre au problème de la représentation multiple d'une même information dans différentes sources. Elle fait partie de la tache de réécriture de requêtes, mais se pose également dans un environnement ou les données sont matérialisées. **[BOU, 08]**

3. Approche de type Médiation

3.1. Caractéristiques

Cette approche étende la fédération en ajoutant davantage de flexibilité. La médiation est basée sur deux concepts : le médiateur et le wrapper.

- **Le médiateur** est une interface entre l'utilisateur et la collection de sources de données lui donnant l'illusion d'interroger un système d'information homogène. Le médiateur est chargé de répondre aux requêtes à partir de connaissances mises à sa disposition. Il permet de localiser l'information et de résoudre les conflits des données.

- **Le wrapper** est une interface permettant la traduction des informations entre le médiateur et les sources de données. Il traduit les requêtes posées par le médiateur en des requêtes

compréhensibles par les sources de données. Ensuite, il traduit les réponses envoyées par les sources de données en des réponses compréhensibles par le médiateur.

Plus :

- **Langage de médiation** : le médiateur adopte un langage unique appelé langage de médiation pour permettre aux différents utilisateurs et applications d'interroger les différentes sources de données de la coopération de manière uniforme en leur masquant les détails d'hétérogénéité et de localisation.

4. Traitement des requêtes sur les systèmes à base d'un médiateur

Le traitement d'une requête posée indépendamment de la localisation des différentes données intervenant pour calculer le résultat, introduit les difficultés de la reformulation d'une requête, la décomposition d'une requête et la recomposition des résultats et le niveau de l'optimisation.

On va présenter le problème de traitement des requêtes dans les systèmes à base d'un médiateur et étudier les différentes approches proposées.

4.1 Traitement d'une requête

Le rôle d'un système de médiation consiste à reformuler la requête d'un utilisateur en termes de schémas sources, puis à la décomposer en sous requêtes, qui seront envoyées aux différentes sources. Les résultats sont ensuite renvoyés au médiateur, qui les intègre avant de les renvoyer à l'utilisateur.

4.2 La reformulation d'une requête

Une des différences principales entre un système d'intégration de données à base d'un médiateur et un système traditionnel de base de données est que les utilisateurs posent leurs requêtes en termes du schéma global. Les données sont stockées dans les sources de données, organisées sous des schémas sources. Par conséquent, pour répondre aux requêtes, il est nécessaire d'établir une correspondance entre le schéma global et les schémas des sources de données. Le processeur de requête doit être capable de reformuler une requête écrite sur le schéma global en une requête écrite sur les schémas des sources de données. On distingue dans la littérature trois approches [LENZ 03][LEVY 00] pour établir la correspondance entre le schéma global et les schémas des sources de données à intégrer (règles de mapping).

4.2.1 L'approche GAV

Un schéma global est considéré comme une vue sur des schémas sources. GAV (Global As View), a été la première à être proposée pour l'intégration de données. Dans l'approche GAV, pour chaque relation utilisée dans le schéma global, on définit une vue composée des termes des relations des schémas sources. La transformation d'une requête écrite en utilisant le schéma global en une requête écrite en utilisant des schémas sources est une opération simple, cette simplicité est considérée comme avantage de l'approche GAV. Les inconvénients de cette approche sont qu'il suppose que les sources à intégrer soient connues à l'avance, ainsi une modification sur l'ensemble des sources ou sur leurs schémas entraîne une reconsidération complète du schéma global.

4.2.2 L'approche LAV

LAV (Local As View) Chaque relation d'un schéma source est définie comme vue sur le schéma global. Dans le cas d'une approche LAV, la requête sur le schéma global doit être reformulée suivant les schémas sources. Dans cette approche, chaque source est spécifiée de manière indépendante.

Cette approche a l'avantage qu'elle est très flexible par rapport à l'ajout (ou la suppression) de sources de données à intégrer, cela n'a aucun effet sur le schéma global, seules des vues doivent être ajoutées (ou supprimées). De l'autre coté, le prix à payer pour cette flexibilité et cette simplicité de mise à jour est la complexité de la construction des réponses à une requête dans un médiateur conçu selon l'approche LAV. La reformulation de requêtes en termes de vues est la seule possibilité pour obtenir des requêtes exprimées en termes de vues, que l'on doit ensuite exécuter pour interroger les sources de données.

4.2.3 L'approche GLAV

GLAV (Global and Local As View) est une combinaison entre GAV et LAV prend les avantages des deux approches prétendantes. GLAV offre plus de flexibilité à la mise à jour par les utilisateurs ou les sources locales. GLAV associe une requête conjonctive écrite sur le schéma global à une requête conjonctive écrite sur des schémas sources.

4.3 La décomposition d'une requête et recomposition des résultats

Dans un système de médiation la requête est décomposée en sous requêtes destinées aux sources de données. Une fois les sous requêtes soumises à chacune des sources, il s'agit de savoir recomposer les différents résultats entre eux pour obtenir un résultat final. Le médiateur doit décider quelle est la part du travail qui doit être effectuée par les sources de données et quelle est la part du travail qui doit être effectuée par lui.

5. Conclusion

Nous avons étudié dans ce chapitre les différents approches et aspects des systèmes de médiations et le traitement d'optimisation des requêtes, mais notre but est la conception d'un système d'optimisation de requête dans un système de médiation pour cela nous aurons étudié les différents systèmes d'optimisation des requêtes dans un système de médiation avec les algorithmes utilisés dans le chapitre suivant.

Chapitre 2 :

Etat de l'art des approches existantes pour l'optimisation des requêtes

1. Introduction

L'utilisation d'un système de médiation permet d'exécuter des requêtes simples et complexes. En conséquence, le traitement de ces requêtes nécessite une grande puissance de calcul du côté du système. Le médiateur global est obligé d'optimiser le processus de traitement de requêtes afin de répondre le plus rapidement possible aux utilisateurs. Les systèmes d'information pouvant traiter les requêtes contiennent toujours un module chargé de l'optimisation de requêtes, appelé optimiseur. Ce dernier analyse les requêtes pour déterminer un moyen efficace de les traiter. En général, il existe de nombreuses solutions pour exécuter la même requête, parmi lesquelles certaines sont souvent beaucoup plus rapides que les autres. Comment choisir la solution la plus efficace devient la problématique essentielle d'un optimiseur.

2. **Processus d'optimisation de requêtes** : L'optimiseur a deux tâches principales à accomplir en un temps acceptable: d'abord, il faut trouver les solutions alternatives pour une même requête donnée. Ensuite, parmi ces alternatives, il faut choisir, en se basant sur certains critères, la plus performante pour exécuter la requête.

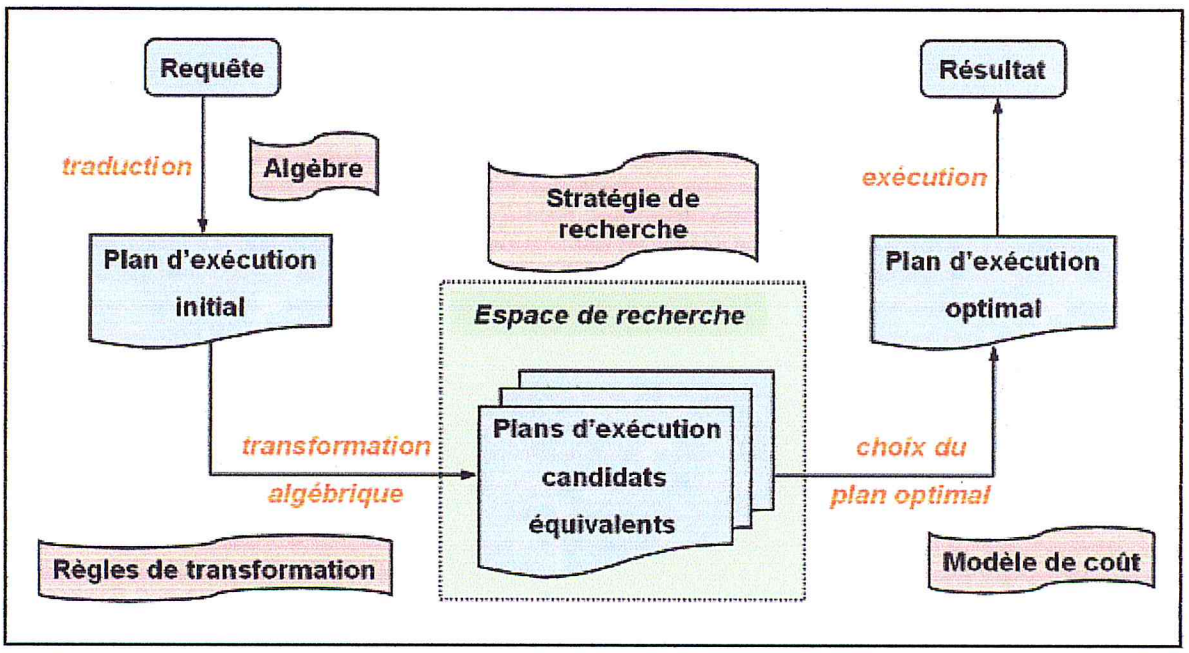


Figure 2 : Processus d'optimisation de requête [Liu 11].

2.1. Algèbre

Une algèbre utilisée par l'optimiseur est constituée d'opérateurs relationnels simples (ex. Sélection, projection, jointure, union, etc.). Et d'un opérateur de communication entre les sources et les autres composants du système. Les opérateurs de l'algèbre acceptent en opérande des relations composées de tuples, et produisent des relations en tant que résultat de l'opérateur. Dans les systèmes relationnels, l'algèbre a été définie par Codd.

Chaque requête est traduite dans les opérateurs de l'algèbre. Il existe d'autres algèbres pour les systèmes non-relationnels, par exemple, une algèbre LORE a été proposée pour XML.

2.2. Plan d'exécution

La requête peut être présentée en utilisant l'algèbre de l'optimiseur, par un arbre d'opérations dont les nœuds sont des opérateurs de l'algèbre, et les feuilles représentent les données des sources (opérandes).

Cet arbre est appelé le plan d'exécution logique de la requête parce qu'il spécifie la méthode pour exécuter la requête, en indiquant l'ordonnancement de l'exécution des différents opérateurs, et la communication des données entre les sources et le système qui traite la requête.

2.3. Règle de transformation

Les équivalences supportées par les opérateurs induisent des réécritures algébriques conservant la sémantique de la requête mais potentiellement plus simples à évaluer. Ces équivalences sont décrites par des règles de transformations. Certaines règles ont pour but de reformuler directement la requête en un plan d'exécution. D'autres utilisent un plan d'exécution initial pour générer un ensemble de plans équivalents.

2.4. Espace de recherche

L'ensemble des plans d'exécution équivalents constitue l'ensemble des possibilités d'exécution d'une requête. Le nombre de possibilités peut s'avérer rédhibitoire. Pour résoudre ce problème, l'optimiseur a besoin d'une stratégie de recherche.

2.5. Modèle de coût

Le but d'un modèle de coût est d'estimer le coût d'exécution des plans. Il permet ainsi de choisir le meilleur plan d'exécution ayant le moindre coût d'exécution. Le modèle de coût contient, d'une part, des statistiques sur les données et sur le système SGBD et, d'autre part, des formules pour évaluer la taille des résultats intermédiaires et le coût des plans.

Ces formules reposent en général sur un certain nombre de paramètres et d'hypothèses simplificatrices. L'unité de mesure du coût dépend de l'objectif d'optimisation. Le coût peut être mesuré :

- en unité de temps si l'objectif est de réduire le temps de réponse du plan.
- en unité de travail si l'objectif est de réduire la consommation de ressources du système.
- en nombre de connexions si l'objectif est de réduire les appels aux sources de données.
- en unité monétaire si l'objectif est de réduire le prix d'exécution de la requête, dans le cas certains accès aux sources sont coûteux.
- en unité de flux de données si l'objectif est de réduire la taille de données circulées dans le réseau à un moment donné.

2.6. Stratégie de recherche:

La stratégie de recherche explore l'espace de recherche pour trouver le meilleur plan d'exécution en utilisant le modèle de coût. Cette stratégie définit quels sont les plans explorés parmi l'ensemble des plans de l'espace de recherche, et l'ordre dans lequel ces plans sont explorés. La programmation dynamique est la stratégie

déterministe la plus couramment utilisée pour construire les plans qui sont des arbres de jointures. **[Liu 11]**

3. Optimisation dynamique des requêtes sur de bases de données hétérogènes

Au contraire de l'optimisation statique qui précède l'exécution de la requête, l'optimisation dynamique a lieu pendant l'exécution de la requête. Nous distinguons dans la littérature [NAAK 99] quatre approches : Réécriture dynamique du plan d'exécution, Prise en compte des sources indisponibles, Ordonnancement dynamique des jointures et Optimisation adaptative.

3.1. Réécriture dynamique du plan d'exécution

L'objectif est de réduire l'impact des sources les plus lentes sur le temps de réponse total de la requête [NAAK 99]. Cette approche propose d'éviter les situations de blocage en modifiant dynamiquement le plan d'exécution pour permettre au médiateur d'effectuer du travail utile en attendant la fin du blocage. Le travail utile est :

- La matérialisation des résultats intermédiaires,
- La construction de nouveaux opérateurs pour manipuler les résultats intermédiaires disponibles. Lorsque le travail utile est réalisé, le médiateur attend que les sources bloquées reprennent la production des résultats.

Cette technique fait l'hypothèse que toutes les sources de données répondent dans un délai raisonnable. Mais, certaines sources de données peuvent être temporairement indisponibles, ou fournir leur réponse dans un délai qui n'est pas raisonnable [BONN 99].

3.2. Le traitement incrémental des requêtes

Le problème est de traiter la requête en tenant compte des sources indisponibles sans attendre que toutes les sources soient disponibles pour débuter le traitement de la requête [NAAK 99]. Cette approche [BONN 99] vise à obtenir efficacement le

résultat d'une requête en commençant le traitement avant que toutes les sources soient disponibles. La requête est traitée partiellement et les résultats partiels sont matérialisés dans le médiateur.

La requête initiale est réécrite pour prendre en compte les résultats partiels (nommée requête incrémentale). La requête incrémentale est exécutée lorsque les sources indisponibles deviennent accessibles. Le résultat est équivalent à celui de la requête initiale mais plus rapide. Un des avantages de cette méthode est de pouvoir traiter une requête bien qu'à aucun instant l'ensemble des sources ne soient disponibles simultanément [NAAK 99].

3.3. Ordonnancement dynamique des jointures.

L'idée est ici de traiter une jointure inter site le plutôt possible, dès que deux sous requêtes ont produit leur résultat. La méthode se déroule en quatre étapes [EVRE 95] .En préliminaire à l'exécution, le coût des jointures entre deux sites est estimé au moyen des formules traditionnelles basées sur la sélectivité des jointures. (2) Toutes les sous requêtes sont envoyées en parallèle aux wrappers, pour être exécutées. (3) Dès qu'une sous requête a produit son résultat, une valeur de seuil est calculée en fonction du coût des jointures qui sont reliées à cette sous requête. Parmi toutes les jointures impliquant les résultats des sous requêtes, celles qui ont un coût supérieur au seuil sont éliminées; la jointure de moindre coût est exécutée. Reprendre l'étape (3) jusqu'à ce que toutes les jointures soient exécutées. Cette approche a deux avantages: elle remplace la stratégie d'optimisation statique qui détermine un plan d'exécution à partir d'un modèle de coût, et elle prend en considération le temps de réponse réel des sources, et ainsi offre une amélioration du temps de réponse total de la requête.

3.4. Optimisation adaptative

(1) L'approche suppose que l'exécution d'une requête se déroule selon trois phases[NAAK 99]: (1) la phase de contrôle (monitoring) pendant laquelle l'état d'avancement de l'exécution est mesuré; (2) la phase de prise de décision au cours de laquelle l'optimiseur décide

éventuellement de corriger le plan courant s'il n'est plus optimal à cause de la perte de précision des estimations utilisées pour son choix; (3) une phase de correction pendant laquelle le plan courant est abandonné au profit d'un nouveau plan.

Le problème de cette approche est la phase de prise de décision qui est basé sur un modèle de coût

4. L'optimisation combinatoire :

Les problèmes d'optimisation combinatoire (POCs) sont complexes et difficiles. On distingue deux grandes classes de méthodes de résolution des POCs : les méthodes exactes et les méthodes approchées ou heuristiques. Les méthodes exactes permettent de trouver des solutions optimales. Cependant, elles deviennent très vite inutilisables pour des instances de taille importante. Par contre, les heuristiques tentent de trouver en un temps raisonnable de bonnes solutions sans garantie d'optimalité mais acceptables au regard des contraintes des POCs et des délais souvent imposés pour leur résolution. Elles regroupent les heuristiques spécifiques à un POC particulier et les métaheuristiques. Les premières sont plus pointues, en général plus « efficace » mais peu réutilisable (les méthodes constructives etc.). Les métaheuristiques sont générales et indépendantes des POCs traités. Elles sont classées en deux catégories : les métaheuristiques à population de solutions (algorithmes évolutionnaires,...) favorisant une diversification de la recherche ; les métaheuristiques à solution unique (recuit simulé,...) manipulant une solution à la fois et caractérisées par une exploitation de la région englobant la solution initiale dans l'espace de recherche du problème traité.

Les algorithmes évolutionnaires (AEs) simulent le processus de l'évolution naturelle pour trouver une meilleure solution à travers la sélection et la re-création sur plusieurs générations. L'utilisation de ces derniers est devenue de plus en plus populaire pour la résolution de problèmes d'optimisation dans différents domaines.

De manière générale, le calcul évolutionnaire est une approche de recherche dans laquelle des opérateurs génétiques, tels que la reproduction, le croisement et la

mutation, sont utilisés pour atteindre une solution satisfaisante dans un espace dont la dimension est déterminée par la longueur des chromosomes (solutions).

4.1. Les métaheuristiques :

Les métaheuristiques ont connu un essor considérable depuis leur apparition dans les années 1970. Elles sont présentées par **[OSLA 96]**comme étant des méthodes d'approximation conçues dans le but de s'attaquer à des problèmes complexes d'optimisation comme les problèmes d'optimisation dans un système de médiation qui n'ont pu être résolus de façon efficace par les heuristiques et les méthodes d'optimisation classiques. Ces mêmes auteurs définissent formellement la notion de métaheuristique comme étant un processus itératif qui guide une heuristique subordonnée en combinant intelligemment différents concepts pour explorer et exploiter l'espace de recherche, et qui utilise des stratégies d'apprentissage pour structurer l'information dans le but de trouver efficacement des solutions les plus rapprochées possible de la solution optimale. Le développement des métaheuristiques fait partie d'un effort soutenu investi dans le domaine de l'optimisation combinatoire. Ce dernier est défini comme étant l'étude mathématique de la recherche d'un arrangement, d'un groupement, d'un ordonnancement ou d'une sélection d'objets discrets habituellement finis en nombre **[OSLA 96]**Malgré les progrès remarquables qu'ont connus les algorithmes exacts durant ces dernières années, l'optimisation combinatoire constitue toujours un défi de taille pour eux. Par conséquent, les algorithmes d'approximation sont devenus une sphère importante de recherche et d'applications dans ce domaine.

Un problème d'optimisation se définit formellement par un triplet P = (I;R; f) avec :

– I l'ensemble des instances de P.

– Étant donné une instance x ϵ I; R(x) désigne l'ensemble fini des solutions réalisables et donc candidates.

5. Quelques systèmes de médiation existants

5.1. Approche pour l'intégration et l'interrogation des sources de données hétérogènes et distribuées [Benh 05]

La solution est proposée, pour résoudre le problème de l'intégration de données hétérogènes et distribuées, est une architecture multi agents décentralisée qui évite les points faibles des architectures centralisées. Cette architecture permet d'offrir aux utilisateurs les moyens de manipuler de manière transparente des données issues d'un ensemble de sources de données hétérogènes, autonomes et distribuées. Elle combine entre l'approche de type médiation et l'utilisation du paradigme agent.

Décideur : décide la meilleure stratégie de coopération qu'il faut appliquer pour traiter efficacement une requête.

Wrapper : joue principalement le rôle d'un traducteur tq il fait :

- La traduction des sous requêtes écrites en utilisant le langage commun vers le langage propriétaire à la source locale.

-La traduction des résultats natifs en résultat au format commun du médiateur.

5.2. TlexIS [PetH 08]

FLEXIS est un système de médiation suivant une approche LAV. Ainsi, il est formé (i) d'un médiateur, (ii) de wrappers et, (iii) de sources de données. Le médiateur dispose d'un schéma global S, de vues logiques V et d'un moteur d'évaluation de requêtes qui calcule les réécritures d'une requête à partir de V puis, qui évalue les réécritures sur les sources afin de générer l'ensemble des réponses à retourner. Le moteur d'évaluation de requête est basé sur un algorithme qui étend celui du MINICON. L'algorithme du MINICON repose sur une approche en deux phases. La première consiste à générer un ensemble de réécritures partielles de la requête utilisateur, appelées MCD. La seconde vise à combiner ces MCDs pour obtenir les réécritures de la requête utilisateur. Une nouvelle extension a été proposée permettant de résoudre le problème de réécriture contenant des prédicats de comparaison dans le cas général.

Dans FLEXIS, les deux phases principales du MINICON ont été étendues afin de sélectionner les vues et de calculer les réécritures de façon plus flexible.

5.3. Piazza [AMLY]

Permet aussi bien l'échange de données relationnelles, XML que RDF. Il est basé sur une architecture Pair à Pair pure. En présence de différents schémas et de différentes représentations, les pairs intéressés par l'échange de données définissent des correspondances sémantiques entre eux, deux à deux ou entre petits groupes de pairs. Chaque pair exprime ses requêtes sur son propre schéma. Les requêtes sont dans ce cas évaluées globalement sur un réseau de pairs sémantiquement liés par les correspondances. Piazza combine et généralise les formalismes LAV (Local As View) et GAV (Global-As-View) proposés dans la médiation de schémas dans les systèmes d'intégration de données et les étend aux documents XML. Le langage d'expression des correspondances pour les données relationnelles est PPL (Peer Programming Language) tandis que celui utilisé pour les documents XML est basé sur XQuery. La réécriture des requêtes est basée sur un pattern matching entre les expressions XQuery et les correspondances sémantiques et elle est faite de manière centralisée. L'approche Piazza présente cependant des insuffisances liées à la difficulté de décrire les correspondances, de les construire mais aussi à la maintenance de ces dernières. A noter que les reformulations des requêtes sont faite par un noeud central.

5.4. GLUE :

Le système GLUE utilise une approche par machine learning pour classifier les concepts d'une ontologie afin de les mettre de manière semi automatique en correspondance avec les concepts définis dans les ontologies distantes. Cependant, bien que tout repose sur la phase d'apprentissage, cette approche suppose qu'un nombre important d'utilisateurs collabore pour définir les mappings sémantiques entre les ontologies.

5.5. PeerDB :

Permet le partage de données relationnelles distribuées sans partage de schéma. Il combine les propriétés des systèmes multi-agents avec celles des systèmes Pair-à-Pair. Chaque pair fournit une base de données relationnelle décrite grâce à des méta-données (mots-clés). La reformulation des requêtes est faite par des agents grâce à une mise en correspondance des méta-données associées aux schémas. Qui permet le traitement de requetes et une interface SQL qui permet a l'utilisateur de soumettre

des requêtes SQL. Pour chaque relation et pour chacun de ses attributs, l'utilisateur crée des mots clés (des synonymes). Ces mots cles sont stockes localement sur les noeuds.

Table 1 : Comparaison entre systèmes de médiation :

Système / Caractéristique	**Approche pour l'intégration et l'interrogation des sources de données**	**Piazza**	**FlexIS**	**GLUE**	**PeerDB**
Architecture	Multi agent (Décentralisée)	Pair-à-Paire (Centralisée)	Médiation (Décentralisée)	Pair-à-Paire (Centralisée)	Pair-à-Paire / Multi agent
Langage de requête	XQuery	XQuery	SQL	XQuery	SQL
Langage de schéma Global	XML	XML	DataLog	XML	DataLog
Langage de sources locales	FichierXML/ BDD Relationnel/ BddObjet/ application	XML/RDF	FichierXML/ BDD Relationnel/Objet / application	FichierXML/ BDD Relationnel/Objet application	FichierXML/ BDD Relationnel application
Type de correspondance	GLAV	GLAV	LAV	GLAV	GLAV
Type d'optimisation	Dynamique	Statique	Dynamique	Semi-automatique	Dynamique

Table 2: Les avantages et les inconvénient des systèmes de médiation

Système/ Caractéristique	Avantages	Inconvénients
Approche pour l'intégration et l'interrogation des sources de données	évite les points faibles des architectures centralisées. Offre une transparence d'accès à la localisation, aux schémas sources et aux langages de requêtes des données sources. Offre une intégration sémantique des données en utilisant les nouvelles technologies pour la représentation de la sémantique des informations	Nécessite de nouvelles méthodes de découverte automatique des correspondances (alignement des ontologies).et des algorithmes efficaces de réécriture et d'optimisation des requêtes. la complexité d'intégration et de traitement des requêtes.
Piazza	Permet bien l'échange de données relationnelles, XML que RDF. La réécriture des requêtes est basée sur un pattern matching entre les expressions XQuery et les correspondances sémantiques.	des insuffisances d'architecture centralisée .La résolution de nombreux conflits sémantiques reste guider par l'utilisateur .Le médiateur offre le seul point d'accès et contrôle l'exécution de toutes les requêtes. La programmation statique
FlexIS	L'algorithme permet de prendre en compte des requêtes disposant de contraintes arithmétiques. la solution proposée est peu flexible.Permet également de générer les K meilleures réécritures. Mise à jour automatique.	l'algorithme gère uniquement la réécriture des requêtes impliquant des sélections, des projections et des jointures. la complexité du processus de réécriture de requêtes.
GLUE	Basée sur l'approche ontologie pour la représentation de la sémantique des informations	des insuffisances liées à la difficulté de décrire les correspondances La réécriture des requêtes est faite de manière centralisée. la complexité du processus d'intégration
PeerDB	Basée sur une amélioration de routage de requêtes. permet à l'utilisateur de modifier la partie partagée de ses données selon ses besoins.	N'est pas basée sur méta-données stockées autoriser des correspondances entre motsclés pouvant aboutir à de fausses reformulations. un système soit centralisé peut provoquer des problèmes de blocages et un embouteillage.

6. Analyse générale des systèmes de médiation

Le système "Approche pour l'intégration et l'interrogation des sources de données" pour résoudre le problème de l'intégration de données hétérogènes et distribuées, est une architecture multi agents décentralisée qui évite les points faibles des architectures centralisées**(Piazza).**Cette approche offre une intégration sémantique des données en utilisant les nouvelles technologies pour la représentation de la sémantique des informations.

1- Les données hétérogènes supportées sont les données relationnelles, les objets et XML.

2- Utilise le langage d'interrogation expressive XQuery comme langage d'interrogation commun et le modèle XML Schéma comme modèle de données commun.

3-une adaptation des règles de mapping GLAV offre une flexibilité aux changements contrairement au **Piazza** qui a des insuffisances comme la résolution de nombreux conflits sémantiques reste guider par l'utilisateur.et Le médiateur offre le seul point d'accès et contrôle l'exécution de toutes les requêtes par une optimisation statique(qui utilise un modèle de coût) De ce fait l'utilisation d'une méthode d'optimisation statique s'apparaît inutile par rapport à l'optimisation dynamique utilisée dans le système **FlexIS** car L'algorithme de l'optimisation dynamique permet de prendre en compte des requêtes disposant de contraintes arithmétiques, et la solution proposée est peu flexible. Permet également de générer les K meilleures réécritures, et une mise à jour automatique. Ainsi que le type de correspondance dans **FlexIS** est LAV Cette approche a l'avantage qu'elle est très flexible par rapport à l'ajout (ou la suppression) de sources de données, De l'autre coté, le prix à payer pour cette flexibilité et cette simplicité de mise à jour est la complexité de la construction des réponses à une requête dans un médiateur contrairement au GAV la simplicité est considérée comme avantage de l'approche GAV et Les inconvénients de cette approche est qu'il suppose que les sources à intégrer soient connues à l'avance, mais l'approche GLAV utilisée dans **Glue et PeerDB** est une combinaison entre GAV et LAV prend les avantages des deux approches.

7. Conclusion

Dans ce chapitre nous avons étudié quelque System de médiation d'optimisation des requêtes en utilisant l'optimisation combinatoire. Nous avons aussi évalué chaque approche par des algorithmes d'optimisation existants, dans le but de nous en inspirer pour la construction de notre système.

La prochaine étape consiste a proposé notre propre solution qui va nos permettre par la suite de réaliser notre système de médiation.

Chapitre 3:

La Solution Proposée

1. Introduction :

Nous avons étudié dans l'analyse précédente (chapitre 2) les avantages et les inconvénients des systèmes basés médiateur. D'après cette étude l'approche proposée est:

- **Contexte de travail**

1.1. Sources de données

Une base de données distribuée (DDB) est une collection de bases de données multiples, interdépendantes logiquement réparties sur un réseau informatique. Cette répartition des ressources améliore les performances, la fiabilité, la disponibilité et la modularité qui sont inhérents à systèmes distribués et hétérogène comme il est montré dans la majorité des systèmes ci-dessus.

Les données supportées par notre solution sont de types relationnels, objets et XML. Pour cela nous définissons des règles du passage d'un schéma relationnel vers XML schéma et d'un schéma objet relationnel vers XML schéma.

1.2. Type de correspondance

L'approche GLAV est la meilleure par rapport aux autres (GAV et LAV). En effet, les deux avantages principaux de cette approche sont : 1- Elle est flexible aux changements des sources de données (comme LAV et au contraire que GAV).

2- Dans la reformulation des requêtes suivant cette approche, chaque règle de mapping est représentée par une requête conjonctive écrite sur le schéma global associée à une requête conjonctive écrite sur des schémas sources. Ces requêtes sont des vues virtuelles ne représentent pas des résultats stockés sur des sources, au contraire que LAV (dans LAV chacune des sources peut être regardée comme elle contient une réponse à une requête écrite sur le schéma global. Par conséquent, les sources représentent des réponses matérialisées aux requêtes écrites sur le schéma global).Ces règles permettent de reformuler la requête d'une manière plus efficace.

1.3. Langage

Dans notre solution nous avons adapté le langage Xquery à cause de sa :

- **Expressivité** : XQuery fonctionne avec toutes sortes de structures de données et sa nature récursive le rend idéal pour les requêtes sur les arbres et les graphes.
- **Flexibilité** : XQuery marche avec des données hiérarchiques et tabulaires.
- **Consistance** : XQuery possède une syntaxe consistante et peut être utilisé avec d'autres standards XML comme XML Schema

1.4. Optimisation

-L'optimisation des requêtes dans un environnement hétérogènes et distribuées est une tâche difficile, à cause l'autonomie des sources. Dans notre travail nous considérons que les sources sont autonomes elles n'exports pas leurs modèles de coûts. De ce fait l'utilisation d'une méthode d'optimisation statique s'apparaît inutile. Nous adoptons d'utiliser une méthode d'optimisation dynamique qui remplace les méthodes D'optimisations statiques. Nous adaptons la méthode d'ordonnancement dynamique des jointures.

1.4.1. La méthode d'ordonnancement dynamique des jointures.

L'idée est ici de traiter une jointure le plutôt possible, dès que deux sous requêtes ont produit leur résultat. La méthode se déroule en quatre étapes. En préliminaire à l'exécution, le coût des jointures entre deux sites est estimé au moyen des formules traditionnelles basées sur la sélectivité des jointures. (2) Toutes les sous-requêtes sont envoyées en parallèle aux wrappers, pour être exécutées. (3) Dès qu'une sous-requête a produit son résultat, une valeur de seuil est calculée en fonction du coût des jointures qui sont reliées à cette sous-requête. Parmi toutes les jointures impliquant les résultats des sous-requêtes, celles qui ont un coût supérieur au seuil sont éliminées, la jointure de moindre coût est exécutée.(4) Reprendre l'étape (3) jusqu'à ce que toutes les jointures soient exécutées.

Cette approche a deux avantages: elle remplace la stratégie d'optimisation statique qui détermine un plan d'exécution à partir d'un modèle de coût, et elle prend en considération le temps de réponse réel des sources, et ainsi offre une amélioration du temps de réponse total de la requête.

1.4.2 Algorithme génétique

-Concernant l'algorithme utilisé dans cette solution est l'algorithme génétique.

Le grand avantage des algorithmes génétiques est qu'ils parviennent à trouver de bonnes solutions sur des problèmes très complexes, et trop éloignés des problèmes combinatoires classiques pour qu'on puisse tirer profit de certaines propriétés connues. Ils doivent simplement déterminer entre deux solutions quelle est la meilleure, afin d'opérer leurs sélections. On les emploie dans les domaines où un grand nombre de paramètres entrent en jeu, et où l'on a besoin d'obtenir de bonnes solutions en quelques itérations seulement.

Par ailleurs, les algorithmes génétiques se prêtent bien, du fait de leur traitement simultané de solutions, à la recherche d'**optimum multiples.**

L'algorithme génétique :

Avec les algorithmes évolutionnaires, nous passons à une catégorie de méta heuristiques, celles des méthodes dites évolutionnaires, qui manipulent un ensemble de plusieurs solutions simultanément.

Principe de l'algorithme génétique :

L'algorithme génétique repose sur une boucle qui enchaîne des étapes de *sélections* et des étapes de *croisements*. Dans un premier temps, à partir d'une population de α individus (Les solutions réalisables), on désigne ceux autorisés à se reproduire.

On croise ensuite ces derniers, de façon à obtenir une population d'enfants, dont on peut faire muter aléatoirement certains gènes.

La performance des enfants est évaluée, grâce à la fonction *fitness*, et l'on désigne, dans la population totale résultante parents+enfants, les individus autorisés à survivre, de telle manière que l'on puisse repartir d'une nouvelle population de α individus.

La boucle est bouclée, et l'on recommence une phase de sélection pour la reproduction, une phase de mutation, et ainsi de suite.

Un critère d'arrêt permet de sortir de la boucle, par exemple un certain nombre d'itérations sans amélioration notable de la performance des individus.

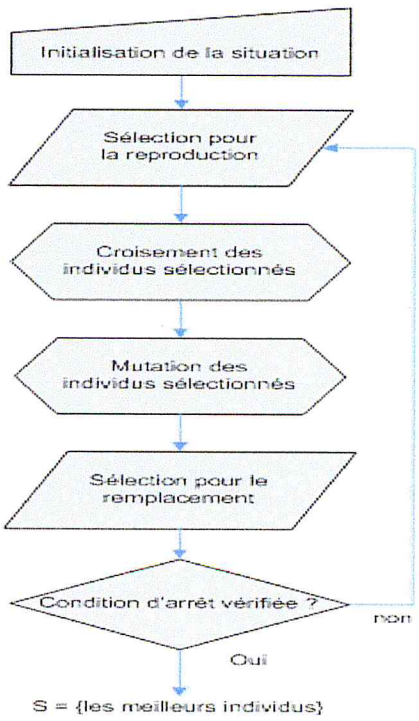


Figure 3 : Principe de fonctionnement d'un algorithme génétique

La solution proposée :

Le but de ce travail est d'exécuter les requêtes le plus efficacement possible afin de minimiser le temps de réponse aux utilisateurs. Et de minimiser les coûts de communication totale associée à une requête. De ce fait on va faire une hybridation entre la méthode d'ordonnancement dynamique des jointures avec l'algorithme génétique à cause des avantages des deux méthodes sur l'amélioration du temps de réponse total de la requête.

Dans notre solution nous avons adapté l'algorithme génétique :

1. En préliminaire à l'exécution La requête peut être présentée en utilisant l'algèbre de l'optimiseur, par des plans d'exécution chaque plan est représenté par un arbre d'opérations dont les nœuds sont des opérateurs de l'algèbre, et les feuilles représentent les données des sources (opérandes).l'ensemble des arbres (*α individus)* est appelé l'ensemble de solution réalisable.

2. Après la décomposition de la requête en des sous-requêtes.

3. Dès que deux sous requêtes ont produit leur résultat, tous les arbres d'exécution (ensemble réalisable) qui ne commencent pas par cette jointure sont éliminés.

4. Appliquer l'algorithme génétique sur l'ensemble réalisable en douze itérations :

-Sélectionner un arbre parmi l'ensemble des solutions réalisables aléatoirement par une fonction random.

-Faire le croisement des individus sélectionnés de façon à obtenir une population d'enfants.

-Faire la mutation des individus sélectionnés aléatoirement (certains gènes).

- Evaluer la performance des enfants, grâce à la fonction ***fitness***, et l'on désigne, dans la population totale résultante parents+enfants, les individus autorisés à survivre, de telle manière que l'on puisse repartir d'une nouvelle population de α individus.

- La boucle est bouclée, et l'on recommence une phase de sélection pour la reproduction, une phase de mutation, et ainsi de suite.

-Un critère d'arrêt permet de sortir de la boucle, est le nombre d'itérations jusqu 'à ce que toutes les jointures soient évaluées.

5. Exécuter le plan optimal.

2. Conclusion

Cette partie, nous avons permis de nous situer dans le contexte général de notre projet de fin d'étude. Les détails spécifiques quant à l'élaboration de notre système seront abordés dans le prochain chapitre.

Chapitre 4:

Conception du système d'optimisation dynamique des requêtes

1. Introduction

Apres avoir définir dans les chapitres précédents respectivement l'état de l'art sur la médiation des données ainsi que les approches existantes pour l'optimisation combinatoire. Nous passons à l'étape conception et modélisation afin de concevoir les schémas généraux qui permettent la modélisation du fonctionnement de l'application, cette dernière est basée sur l'approche dynamique d'ordonnancement dynamique des jointures, Les deux avantages de cette méthode sont : elle remplace la stratégie d'optimisation statique qui détermine un plan d'exécution à partir d'un modèle de coût, et elle prend en considération le temps de réponse réel des sources, et ainsi offre une amélioration du temps de réponse total de la requête.

2. Spécification des besoins

2.1. Besoins Fonctionnels

Il s'agit des fonctionnalités du système. Ce sont les besoins spécifiant un comportement d'entrée / sortie du système. Les besoins fonctionnels déduits à partir de notre étude se résument ci-dessous :

- Lors de sa connexion à l'interface de l'application l'administrateur lance sa requête dans un langage commun (sql).
- Le médiateur est chargé de traiter la requête.
- Le médiateur est chargé de connecter la requête à l'adaptateur attaché à la source approprié.
- L'adaptateur est chargé de traduire les requêtes du langage global au langage local associé à la base de données.
- L'adaptateur est chargé de traduire le résultat fournit par les sources de données du langage local au langage global.
- Le médiateur est chargé de fusionner les résultats fournit par les adaptateurs. a l'utilisateur.
- L'optimiseur est chargé de évalué les plans d'exécution pour obtenir le meilleur plan possible afin d'optimiser le coût d'exécution des requêtes.

2.1.1. Identification des Acteurs

La première étape de cette phase est d'énumérer les acteurs susceptibles d'interagir avec le système.

Définition

Un **Acteur** représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système), qui interagissent directement avec le système étudié **[LAU 06]**.

Le tableau ci-dessous identifie les acteurs et décrit la mission de chacun.

Acteurs	Mission
Utilisateur	L'utilisateur s'occupe de l' lancement de la requête.
Administrateur	L'application doit permettre aux administrateurs de : • S'authentifier : L'administrateur doit s'authentifier avec un nom et un mot de passe. • consulter et de modifier les schémas.

Table 3 : les acteurs du système.

2.1.2. Modélisation du contexte

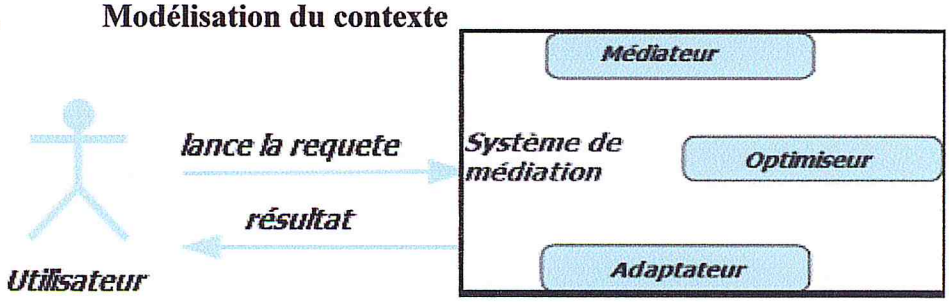


Figure 4 : diagramme de contexte du système.

2.1.3. Identification des cas d'utilisation

L'identification des cas d'utilisation donne un aperçu des fonctionnalités futures que doit implémenter le système.

Nous allons effectuer cela en considérant l'intention fonctionnelle de l'acteur par rapport au système dans le cadre de l'émission et de la réception de chaque message.

2.1.3.1. Diagramme de cas d'utilisation: Le diagramme de cas d'utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système. C'est le premier diagramme du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre.

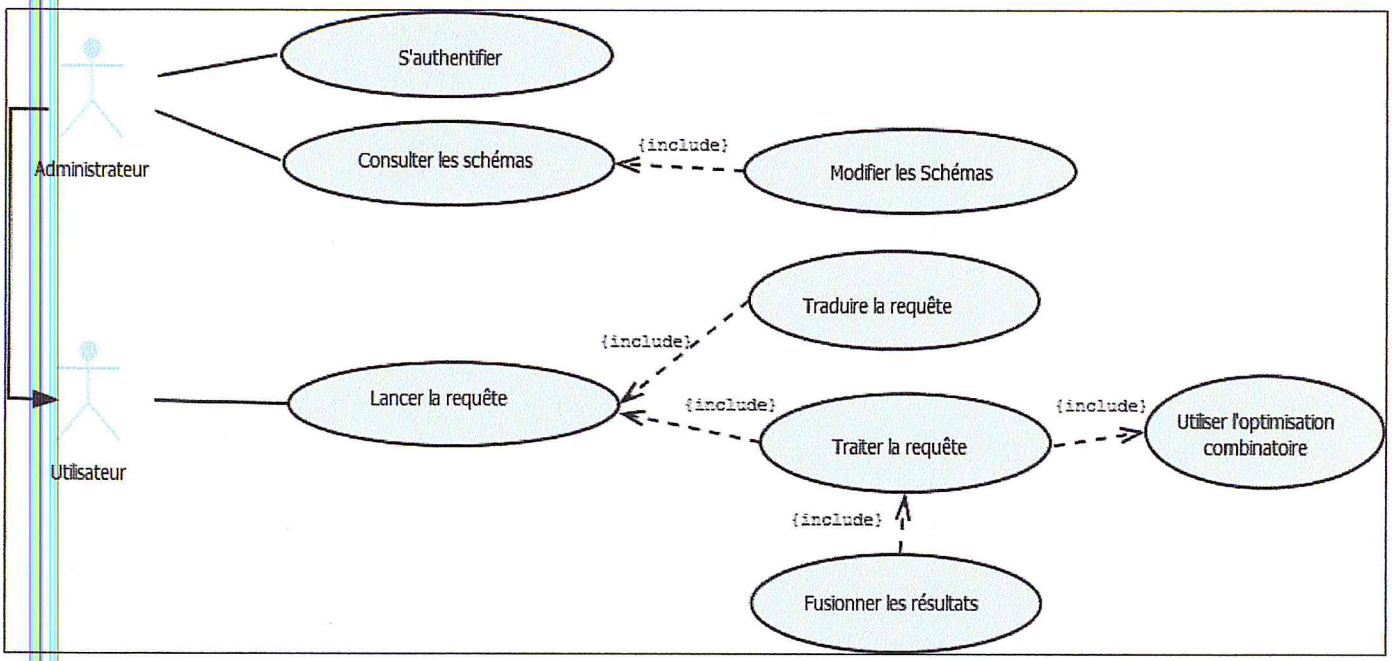
Un cas d'utilisation représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier. Un cas d'utilisation modélise un service rendu par le système. Il exprime les interactions acteurs/système et apporte une valeur ajoutée « notable » à l'acteur concerné **[LAU, 06]**.

Remarque : Les descriptions détaillées vont être organisées dans des tableaux de la façon suivante :

Elément	**Signification**
Cas d'utilisation	Le nom de cas d'utilisation
Acteur	L'acteur qui réalise ce cas d'utilisation
But	Le but de cas d'utilisation
Description	Une explication de cas d'utilisation
Pré condition	Les conditions qu'elles doivent être vérifiées afin de démarrer le cas d'utilisation
Post condition	Les résultats de cas d'utilisation
Exception	Les informations entrées par l'acteur

Table 4 : Modèle de représentation des descriptions détaillées des cas d'utilisations.

Diagramme des cas d'utilisation Global du système



a. Le cas d'utilisation Authentification

Cas d'utilisation	Authentification.
Acteur	Administrateur.
Objectifs	Permettre à l'administrateur de s'authentifier pour ouvrir sa session et accéder à ses privilèges.
Description	L'accès à l'espace approprié doit passer par le formulaire qui contient les renseignements adéquats.
Pré condition	L'administrateur doit fournir les informations convenables (email et le mot de passe doivent être corrects).
Post condition	L'administrateur peut effectuer les taches qui lui sont permises.
Exception	Annulation ; si l'administrateur tape l'email d'utilisateur ou un mot de passe qui ne convient pas, le système affiche un message d'erreur.

Table 5 : Description du cas d'utilisation « Authentification ».

b. Le cas d'utilisation Lancer une requête

Cas d'utilisation	Lancement d'une requête.
Acteur	Utilisateur du système.
Objectifs	Consulter un ensemble d'informations sur l'interface utilisateur.
Description	Ce cas d'utilisation commence lorsque l'utilisateur de système accède à une interface dans laquelle il interroge le système en posant sa requête.
Pré condition	Aucune.
Post condition	Une requête en schéma globale est crée.
Exception	Requête erroné ou ne correspond pas aux données de domaine.

Table 6 : Description du cas d'utilisation « Lancer une requête».

c. Le cas d'utilisation Consulter les schémas

Cas d'utilisation	Consulter les schémas.
Acteur	Administrateur.
Objectifs	Consulter les schémas pour les modifier.
Description	Le médiateur doit tirer l'ensemble des éléments constituant les schémas.
Pré condition	existence des schémas.
Post condition	un ensemble d'élément de concept est tiré.
Exception	Pas d'exception.

Table 7 : Description du cas d'utilisation «Consulter les schémas».

2.2. Analyse

L'analyse permet de lister les résultats attendus, en terme de fonctionnalités, de performance, de maintenance,… etc.

L'analyse répond donc à la question « *que faut-il faire ?* » et a pour but de se doter d'une vision claire et rigoureuse du problème posé et du système à réaliser en déterminant ses éléments et leurs interactions.

L'analyse livre une spécification plus précise des besoins grâce à l'utilisation du diagramme de séquence. Elle peut être envisagée comme une première ébauche du modèle de conception. **[Proc, 08]**

2.2.3 Qu'est qu'un diagramme de séquence ?

Un diagramme de séquence est une représentation séquentielle du déroulement des traitements et des interactions entre les éléments du système et / ou de ses acteurs.

Les diagrammes de séquences permettent de représenter des collaborations entre objets selon un point de vue temporel, on y met l'accent sur la chronologie des envois de messages.

Dans ce qui suit nous allons présenter les diagrammes de séquence afin de formaliser les scénarios des cas d'utilisation vus précédemment. Nous allons voir le système comme un ensemble d'objet en interaction.

2.2.1. Diagramme de séquence du cas d'utilisation Authentification.

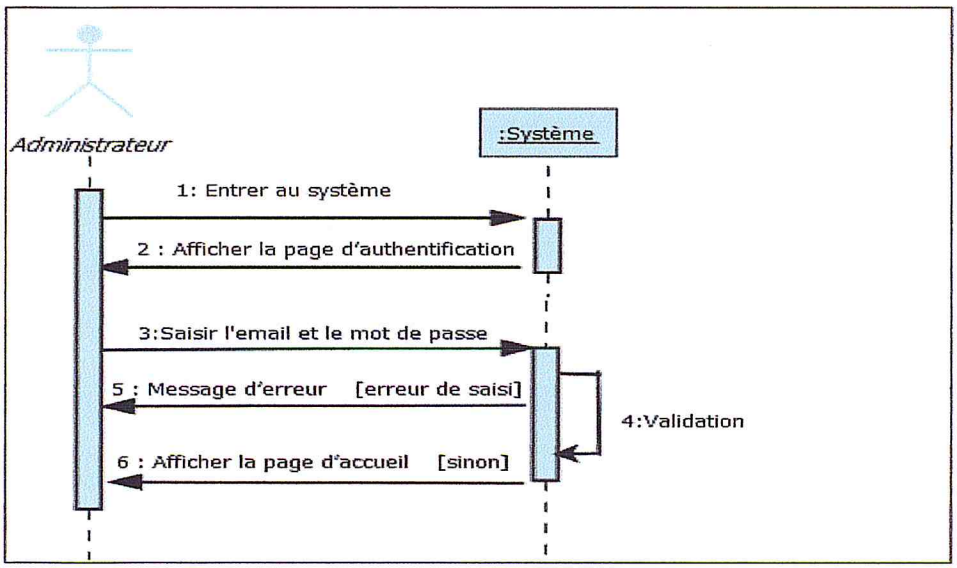


Figure 6 : Diagramme de séquence de processus d'authentification.

2.2.2. diagramme de séquence du cas d'utilisation lancement d'une requête

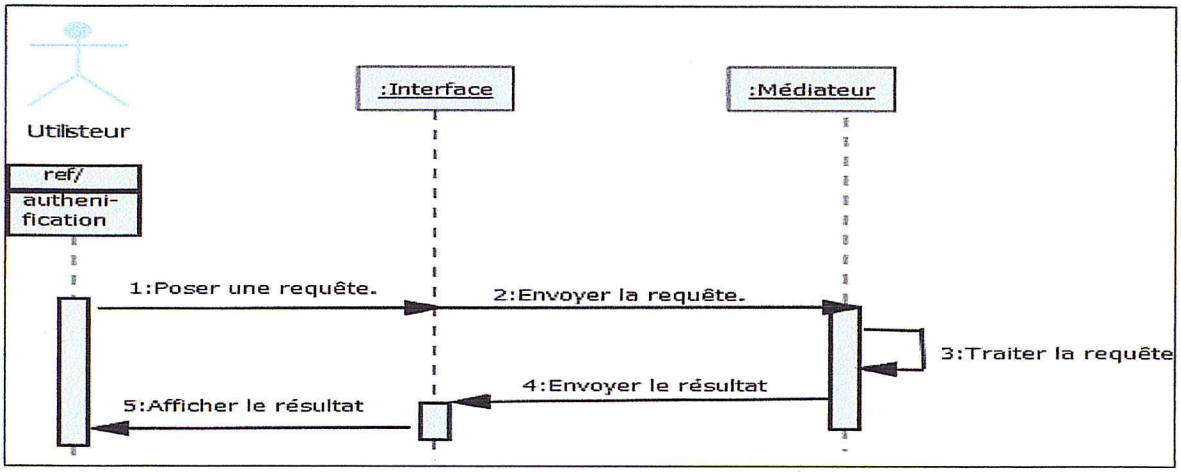


Figure 7 : diagramme de séquence du cas d'utilisation lancement d'une requête.

2.2.3. Diagramme de séquence du cas d'utilisation Traitement de la requête

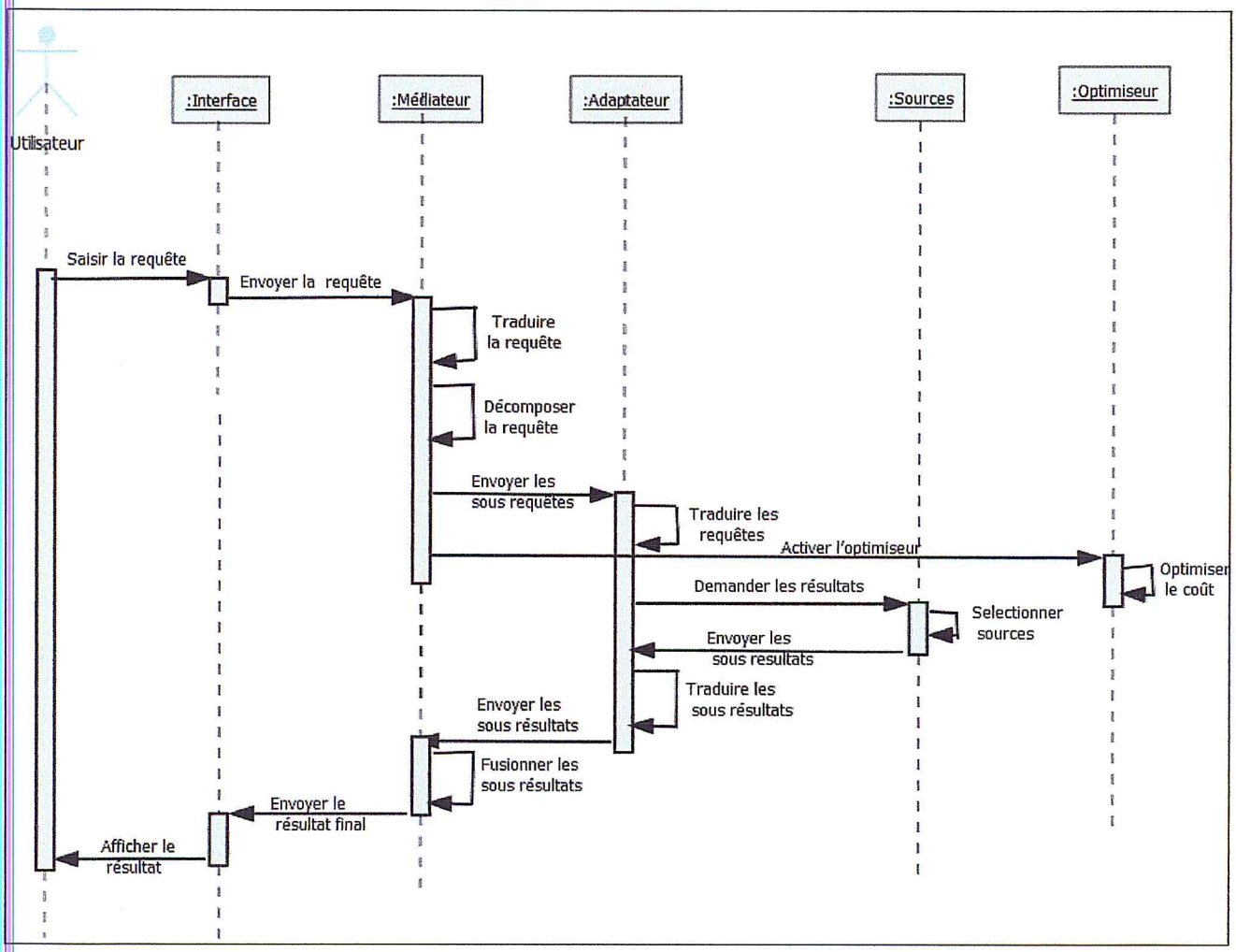


Figure 8 : diagramme de séquence du cas d'utilisation Traiter la requête.

2.2.4. Diagramme de séquence du cas d'utilisation Traduire la requête

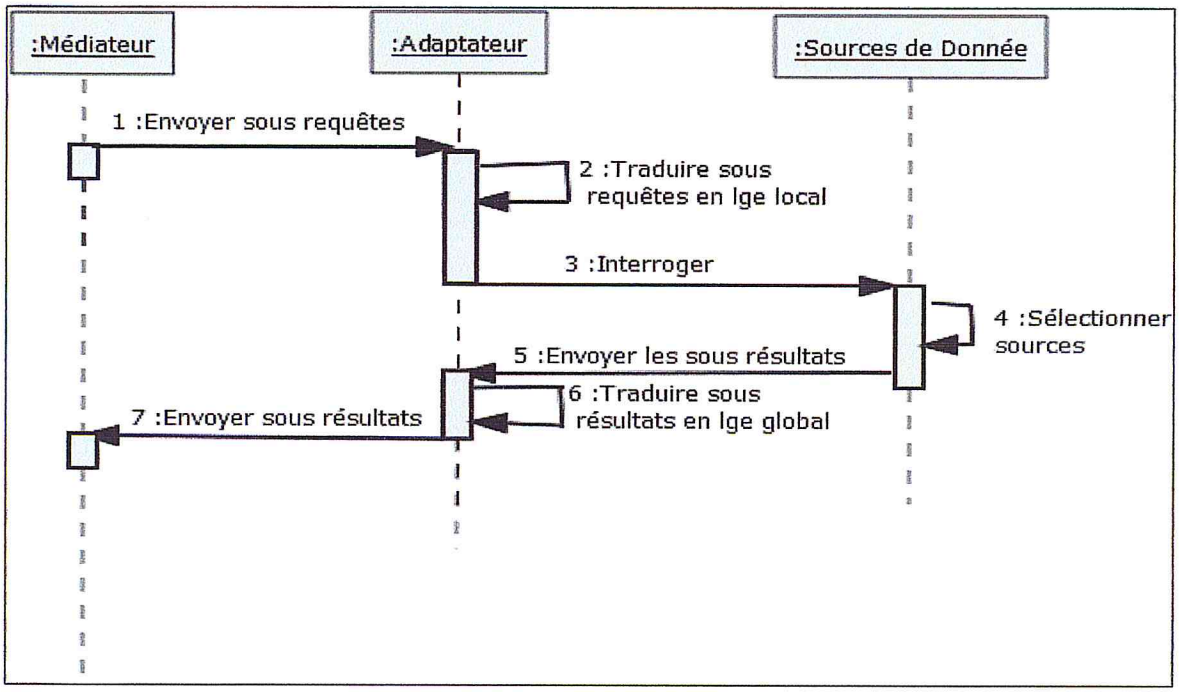


Figure 9 : Digramme de séquence du cas d'utilisation traduire la requête.

2.2.5. Diagramme de séquence du cas d'utilisation Fusion des résultats

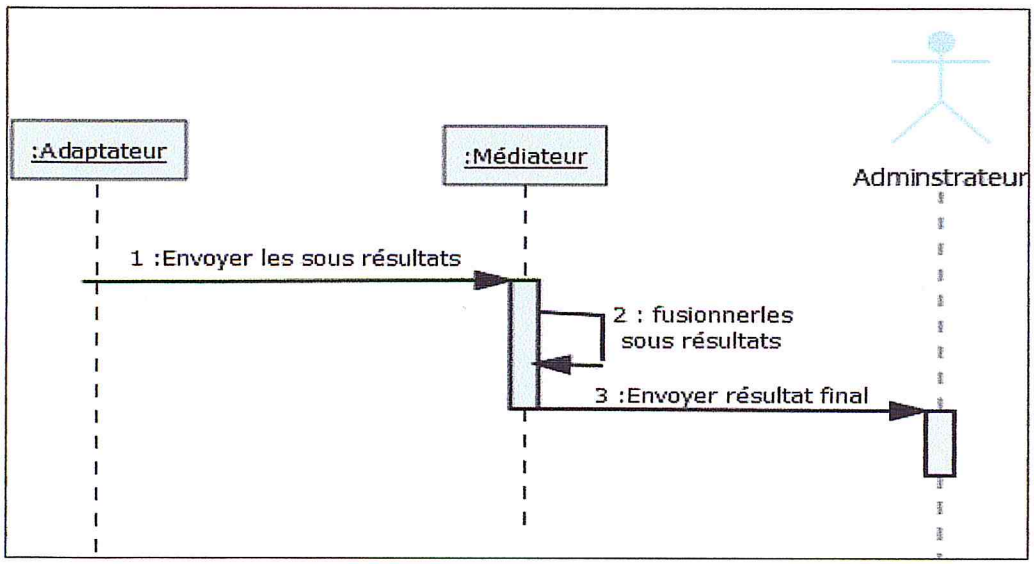


Figure 10 : diagramme de séquence du cas d'utilisation Fusion des résultats.

2.2.6. diagramme de séquence du cas d'utilisation Optimisation de coût de traitement

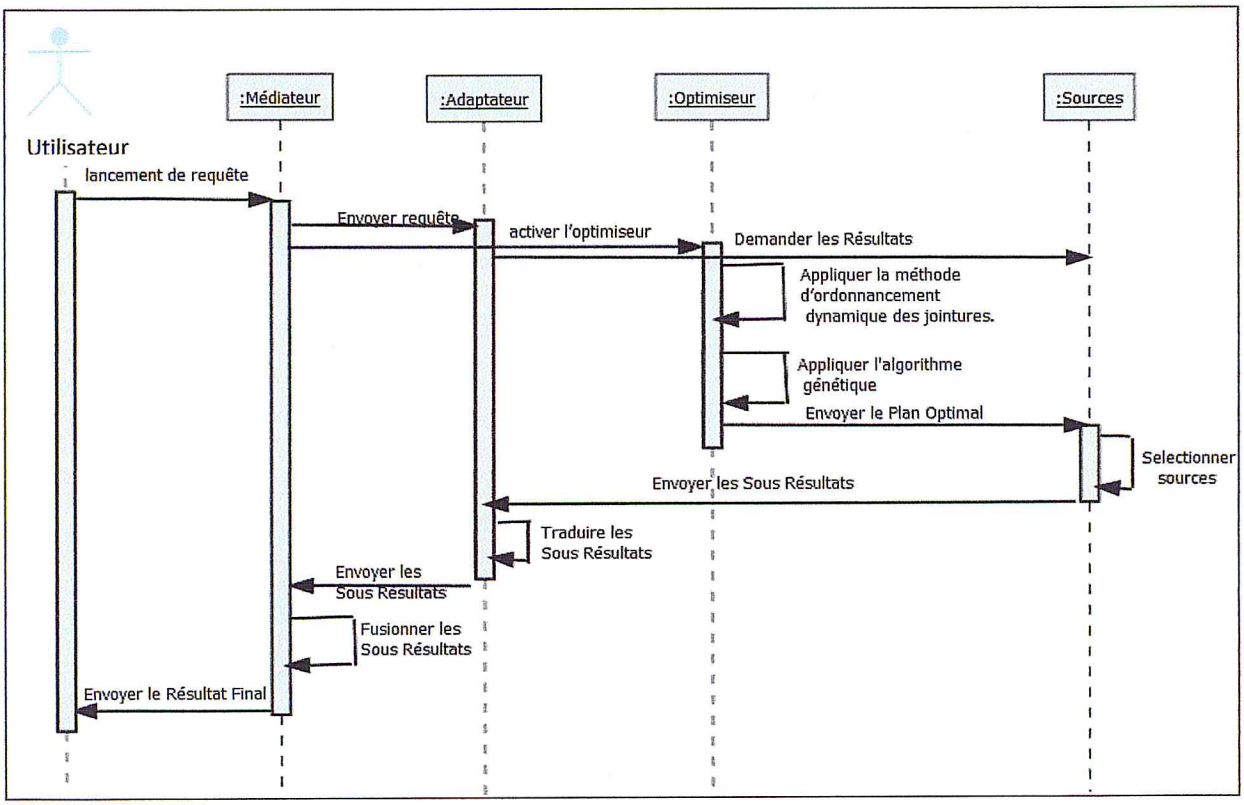


Figure 11 : diagramme de séquence du cas d'utilisation Optimiser le coût.

3 Conception

La conception permet de manière non ambiguë le fonctionnement futur de système, afin de faciliter la réalisation. La conception menée à la suite a base d'analyse répond donc à la question « *comment faut-il faire ce qu'il faut faire ?* » . **[Proc, 08]**

3.1. Diagramme de classes

Le diagramme de classes est généralement considéré comme le plus important dans un développement orienté objet. Il représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens, que ceux-ci représentent un emboîtage conceptuel (héritage) ou une relation organique (agrégation) **[LAU, 06].**

3.1.1. Diagramme de classe de Médiateur :

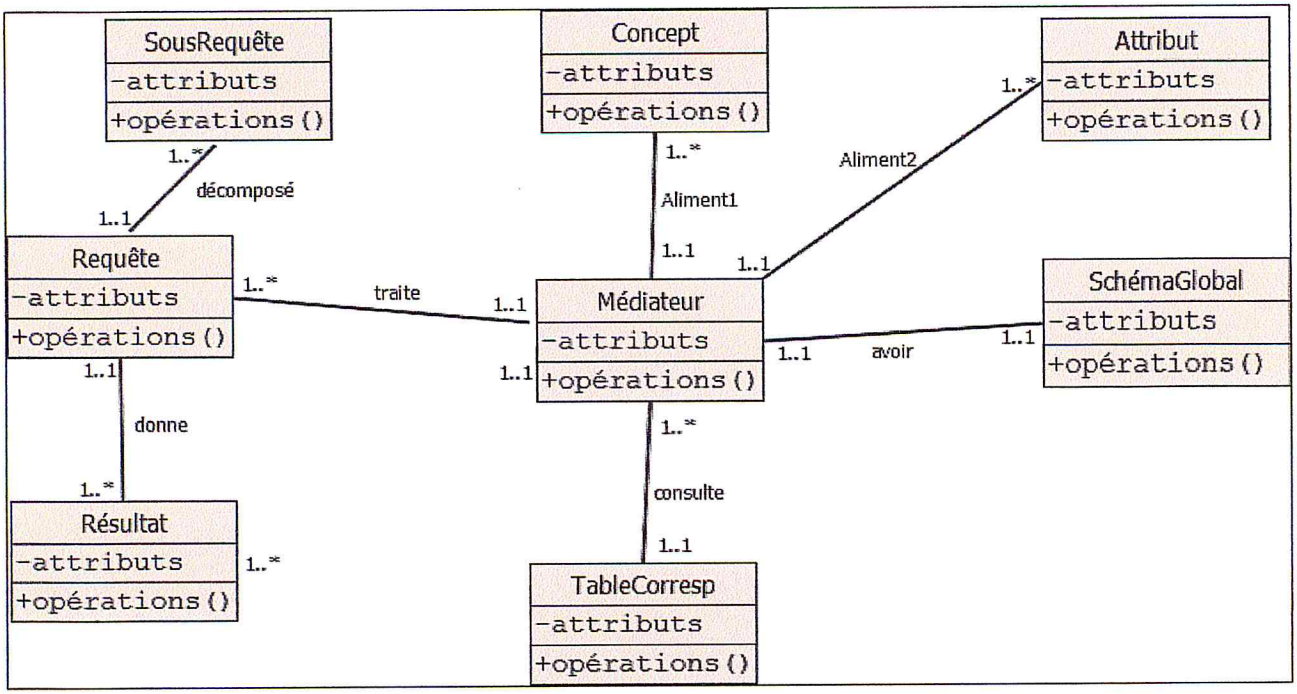


Figure 12 : Diagramme de classe du Médiation.

Diagramme de classe d'Adaptateur :

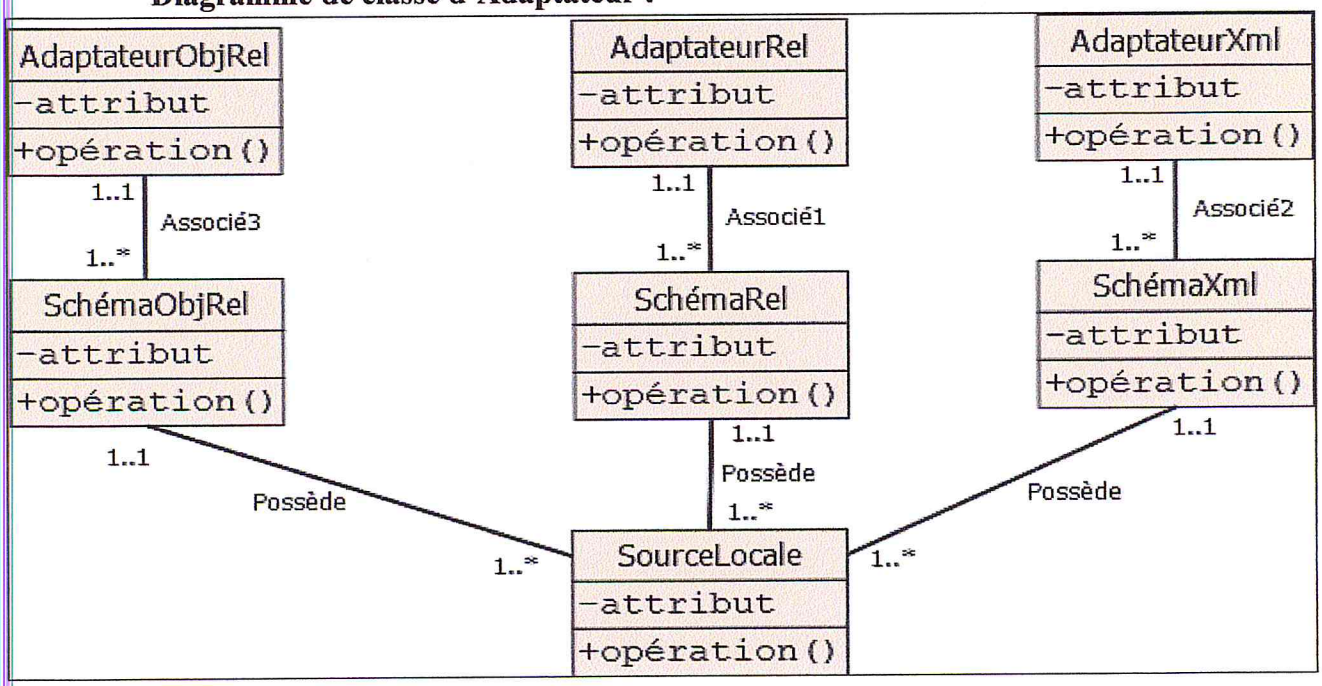


Figure 13 : diagramme de classe Adaptateur.

3.2. Diagramme de paquetage

Le diagramme de Paquetage permet le regroupement des classes logiciels en sous systèmes, leur objectif est d'encapsuler et décomposer la complexité, faciliter le travail en équipes, faciliter la réutilisation et l'évolutivité, la figure suivante représente notre diagramme de paquetage.

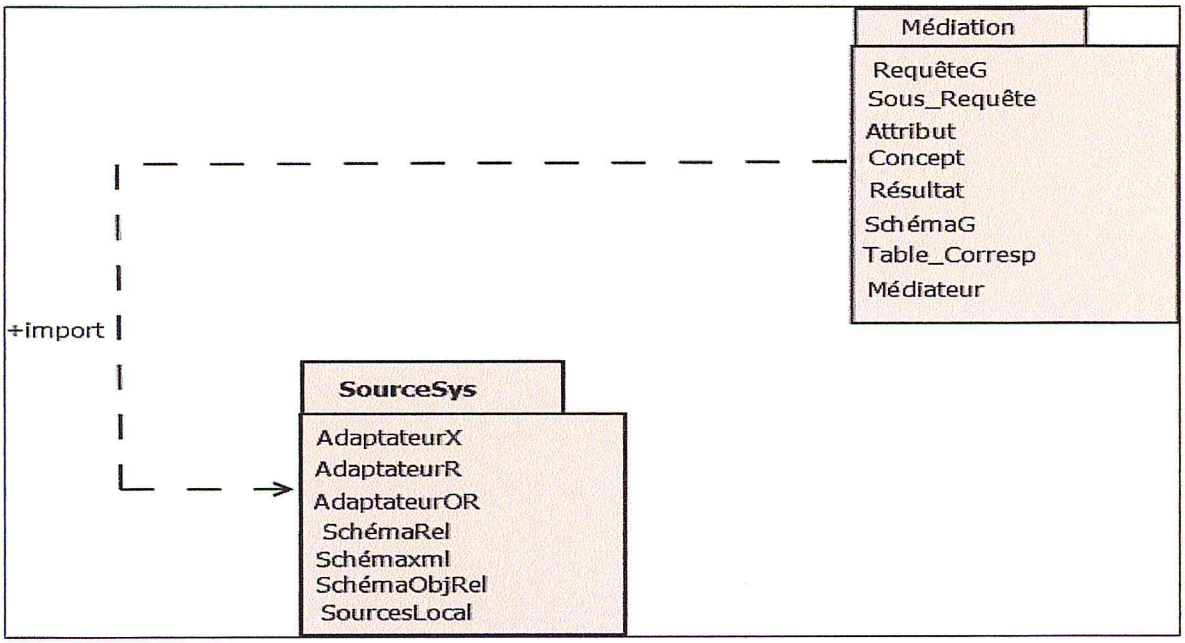


Figure 14 : diagramme de paquetage.

Conception détaillée :

4. Vue générale de l'architecture de système :

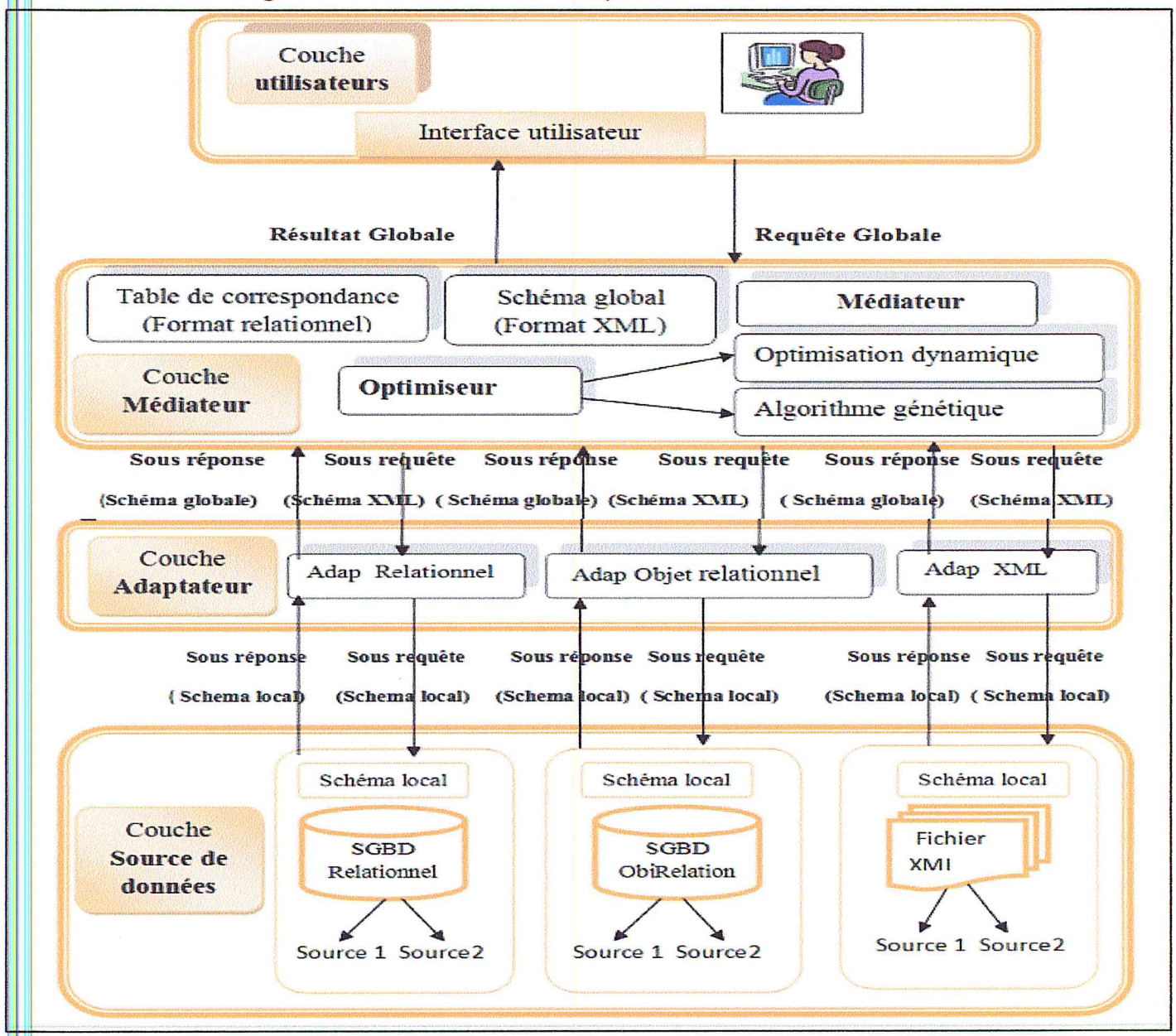


Figure 15 : Architecture générale du système.

4.1. Description de l'architecture :

Notre système a une architecture de 4 couches de bas en haut commençons par :

4.1.1. Couche Source de données

Les sources de données utilisées peuvent être de nature structurée, semi structurée ou bien non structurée. Dans notre approche on a utilisé 2 sources dans

chaque base de donnée, les données sont de type structuré et semi structuré du fait qu'on s'intéresse à des bases de données et au document XML.

Les bases de données sont structurées par leur modèle conceptuel (Relationnelle), le document XML est bien formé en respectant son XML-Schéma. Pour réaliser notre expérimentation, nous avons choisi trois sources de données, chacune est un fragment du schéma global.

➢ **Une source de données structurée** : c'est une Base de données relationnelle qui porte le nom « DBMedical » : pour concevoir cette base on a utilisé un SGBDR (Système de Gestion de Base de Données Relationnelle) SQLServer. Cette base contient 5 tables

Malade (IDMal, NomMal ,PreMal ,DateNaissMal,Adresse).

Médecin (IDMed,NomMed ,PreMed).

Maladie (IDM,NomMad,DescriptionMal).

Atteint (IDMal, IDM).

Consultation (IDMal , IDMed ,DateCon).

Et une base de donnée Objet relationnel qui porte le nom « DBMed » Cette base contient 5 tables :

Patient (IDP, NomP, PreP ,DateNaissP,AdresseP).

MédecinP (IDMedecin,NomMedecin ,PreMedecin).

MaladieP (IDMalP, NomMalP, DescriptionMalP).

AtteintP (IDMalP, IDP).

VisiteP (IDP, IDMedecin , DateConPat).

➢ **Une source données semi structurée** : c'est un document XML portant le nom "BaseXml", voici un fragment de ce document

```xml
<?xml version="1.0" encoding="UTF-8"?>
    <Malade>
        <ID>1</ID>
        <NomP>nassima</NomP>
        <PrenomP>Amrane</PrenomP>
        <Date_Nai>11-05-1988</Date_Nai>
        <Lieu_Nai>Blida</Lieu_Nai>
        <Adresse>Blida</Adresse>
    </Malade>
    <Maladie>
        <ID>2</ID>
        <Dscrption>Gripe</Dscrption>
        <Obs>Null</Obs>
    </Maladie>
    <Medecin>
        <ID>1</ID>
        <Nom>Mohammed</Nom>
        <Prenom>boubaya</Prenom>
        <Date_Nai>10-06-1991</Date_Nai>
        <Lieu_Nai>Blida</Lieu_Nai>
        <Adresse>Alger</Adresse>
    </Medecin>

      <Consultation>
        <ID>1<ID>
        <Malade_ID>1</Malade_ID>
        <Medecin_ID>1</Medecin_ID>
        <Date_Visite>12-05-2015</Date_Visite>
        </Consultation>
     <Atteint>
        <ID>1</ID>
        <Malade_ID>1</Malade_ID>
        <Maladie_ID>2</Maladie_ID>
```

Figure 16 : Une source données semi structurée.

Chaque source de données est associée à un schéma local :

➢ **schéma local** : c'est le schéma d'une source de données, il représente les structures dans laquelle les données sont stockées dans cette source. Ces schémas sont représentés par des documents XML, Un schéma local est associé à un seul adaptateur.

Ces schémas locaux sont créés de la manière suivant :

1. Le nom de la base est représenté par élément dont il imbrique tous les autres éléments.

2. Chaque table est représentée par un élément dont le nom est « NomTable ».

3. Les attributs d'une table sont représentés par des sous éléments dont le nom est « colonne».

```
<?xml version="1.0" encoding="UTF-8"?>
<BaseRelationnel>
  <Table name="malade">
    <colonne>ID</colonne>
    <colonne>Nom</colonne>
    <colonne>Prenom</colonne>
    <colonne>Date_Nai</colonne>
    <colonne>Lieu_Nai</colonne>
    <colonne>Adresse</colonne>
  </Table>
  <Table name="Maladie">
    <colonne>ID</colonne>
    <colonne>Dscption</colonne>
  </Table>
 <Table name="Medecin">
    <colonne>ID</colonne>
    <colonne>Nom</colonne>
    <colonne>Prenom</colonne>
    <colonne>Date_Nai</colonne>
    <colonne>Lieu_Nai</colonne>
  </Table>
  <Table name="Consultation">
    <colonne>ID</colonne>
    <colonne>Malade_ID</colonne>
    <colonne>Medecien</colonne>
    <colonne>Date_ID</colonne>
  </Table>
 <Table name="Atteint">
    <colonne>ID</colonne>
    <colonne>Maladie_ID</colonne>
    <colonne>Malade_ID</colonne>
 </Table>
</BaseRelationnel>
```

Figure 17 : Schéma local de la base Relationnel.

4.1.2. Couche Adaptateur

L'adaptateur cache l'hétérogénéité au médiateur. Il est associé aux plusieurs schémas locaux et joue le rôle d'intermédiaire entre les sources et le médiateur.

- C'est un « Traducteur » Traduction du langage de requête commun du médiateur en langage de requête natif propre à la source.
- Traduction des résultats natifs en résultat au format commun du médiateur.

- **Adaptateur Relationnel** : cet adaptateur est chargé d'exécuter la requête sur les sources relationnelles appropriées et de fournir le résultat au médiateur.
- **Adaptateur Objet Relationnel** : cet adaptateur est chargé d'exécuter la requête sur les sources objet relationnelles appropriées et de fournir le résultat au médiateur.
- **Adaptateur XML :** pour l'adaptateur XML nous avons utilisé Une API Java qui se charge de la traduction de la requête, et d'interroger la source pour tirer une réponse.

4.1.3. Couche Médiateur :

Le cœur de notre système se situe dans le médiateur. Il est décomposé en modules reliés entre eux. Pour bien répondre aux besoins d'un système d'intégration, un médiateur doit :

1. Accepter les requêtes des utilisateurs.
2. décomposer la requête en des sous requêtes.
3. Localiser les sources de données pertinentes.
4. optimiser les requêtes en utilisant une hybridation entre la méthode d'ordonnancement dynamique des jointures avec l'algorithme génétique.
5. Envoyer les sous requêtes à faire exécuter par les adaptateurs sur les sources.
6. Fusionner les résultats des adaptateurs.
7. Assurer le passage à l'échelle lors de l'ajout ou la mise à jour d'une source.

Afin de bien réaliser ses tâches, le médiateur possède les composants suivants, Qu'on va les détaillés par la suite :

1. Module de schéma global.
2. Module de table de correspondance.
3. Module d'optimisation de la requête.
4. Module de traitement de requêtes.
5. Module de fusion des résultats.

➢ **Schéma global :**

La présence d'un schéma global est nécessaire puisqu'il fournit un vocabulaire unique servant à exprimer les requêtes des utilisateurs. Ce schéma unifie les schémas hétérogènes des sources à intégrer en se basant sur une description homogène, uniforme et abstraite du contenu des sources par des vues.

Notre schéma global est créer en utilisant le modèle commun (XML), il est constitué des vues qui vont contenir les résultats des requêtes.

Nous avons un schéma global qui est construite par l'approche GLAV (Global Local as View), c'est-à-dire que la correspondance (Mapping) est dans les deux sens (un concept global sur un schéma global correspond à un concept local sur un schéma local et l'inverse).

```
<?xml version="1.0" encoding="UTF-8"?>
<schemGlobal>
  <Table nom="Medecin">
    <NomAttribute>IDMedecin</NomAttribute>
    <NomAttribute>NomMedecinG</NomAttribute>
    <NomAttribute>PrenomMedecinG</NomAttribute>
    <NomAttribute>MESpecialite</NomAttribute>
    <NomAttribute>MEAdresse</NomAttribute>
    <NomAttribute>METelephone</NomAttribute>
    <NomAttribute>MEDate_Naiss</NomAttribute>
  </Table>
  <Table nom="Malade">
    <NomAttribute>IDPatientG</NomAttribute>
    <NomAttribute>NomMaladeG</NomAttribute>
    <NomAttribute>PrenomMaladeG</NomAttribute>
    <NomAttribute>PASexe</NomAttribute>
    <NomAttribute>PAAdresse</NomAttribute>
    <NomAttribute>PATelephone</NomAttribute>
    <NomAttribute>Date_NaissMaladeG</NomAttribute>
  </Table>
  <Table nom="Maladie">
    <NomAttribute>NumeroMaladie</NomAttribute>
    <NomAttribute>NomMaladie</NomAttribute>
    <NomAttribute>Description</NomAttribute>
  </Table>
  <Table nom="Visite">
    <NomTable>Visite</NomTable>
    <NomAttribute>NumVisiteG</NomAttribute>
    <NomAttribute>DateVisiteG</NomAttribute>
    <NomAttribute>IDPatientG</NomAttribute>
    <NomAttribute>IDMedecin</NomAttribute>
  </Table>
```

Figure 18 : Fragment de schéma global.

Remarque :

La création des schémas sources pour la solution proposée est supposée faite par l'administrateur de l'application, car il connaît les sources.

- **Table de correspondance**

Cette table est celle utilisée pour décrire les correspondances entre schémas (schéma global /schémas des sources).C'est une base de données relationnelle qui porte le nom " TableCorrespondance" : pour concevoir cette base on a utilisé un SGBDR (Système de Gestion de Base de Données Relationnelle) SQLServer. Cette base contient deux tables :

Table 1 : Concept (ID_Cp,Nom_Cpt_G, Nom_Cpt_L).

Table 2: Attribut (ID_Att, ID_Cp , Att_Gl , Att_Loc).

- ***Couche utilisateurs***

C'est une simple Interface de communication permet à un utilisateur (administrateur ou non administrateur) de communiquer avec le système, Il envoie des requêtes au médiateur et reçoit des réponses, cette interface contient des champs de sélection qui aide l'utilisateur à sélectionner les éléments constituant la requête.

Interface 1 : c'est l'interface d'accueil.

Interface 2 : c'est l'interface d'inscription permet d'enregistrer les informations entrées.

Interface 3 : c'est l'interface d'authentification permet de vérifier les identificateurs et de connecter.

Interface 4 : c'est l'interface de consultation des schémas locaux et le schéma global permet de modifier les informations existantes dans ces schémas.

Interface 5: c'est une interface qui permet aux administrateurs de poser leurs requêtes en terme global et de sélectionner les attributs.

Interface 6 : c'est l'interface de décomposition de requête en 3 sous requêtes en terme locale pour interroger les sources de données.

Interface 7 : c'est l'interface d'affichage de résultat permet de faire une optimisation combinatoire pour afficher tout les plans possible et le plan le plus optimale avec le calcul du temps de réponse.

4.2. Description des modules :

4.2.1. Le module Traitement de requête :

Ce module est chargé de traiter une requête globale diviser en sous requêtes définies sur une relation globale envoyée et exécuter sur les sources qui contiennent seulement des réponses. Formellement, considérons une requête **Q** dont la syntaxe est la suivante : **SELECT** AG **FROM** CG **WHERE** Valeur, avec:

$CG < CG_1, CG_2, \ldots, CG_m >$ représente l'ensemble de tables (relations) et

$AG < AG_1, AG_2, \ldots, AG_n >$ représente l'ensemble des attributs (champs) d'un CG_i.

Exemple :

CG < Médecin, Malade, Maladie, Consultation, Atteint >.

$AG_{Médecin}$ <ID, Nom, Prénom, DateDeNaissance, Adresse, Spécialitée>.

La requête Q sera donc décomposée en sous requêtes.

Soit Q_i une sous requête correspond à une relation globale S_i appartenant a l'ensemble S. Soient AG_i l'ensemble d'attributs projetés de la table S_i et C l'ensemble des conditions de sélection de la sous requête Q_i, ou C_i est l'ensemble de conditions définies sur des attributs de la relation globale S_i.

Notons que : $AG_i \subseteq AG$, $Ci \subseteq C$, $CG_i \subseteq CG$.

Algorithme : Reformulation et Décomposition de requête

Entrée : - Requête en schéma global (RG)

Sortie : - Ensemble de sous requêtes en schéma local (SRL)
1 - Sous requête en schéma local destinée à la base xml.
2 - Sous requête en schéma local destinée à la base Relationnelle.
3 - Sous requête en schéma local destinée à la base Objet Relationnelle.

Début

Tirer l'ensemble des concepts locaux CL pour CG_i de RG (table de correspondance).

Tirer l'ensemble des attributs locaux AL pour AG_i de RG (à partir de la table de correspondance).

NombreDeSousRequête=card(AL)

Créer des sous requêtes globales selon ce nombre

// Déterminer pour chaque sous requête local SRL_i la source local relative

Pour chaque sous requête local SRL_i

Tirer l'ensemble des sources pertinentes pour le nom de table NT_i

Fin

4.2.2. Module d'optimisation de requête :

Le but de l'optimisation est d'exécuter les requêtes le plus efficacement possible afin de minimiser le temps de réponse aux utilisateurs. Et de minimiser les coûts de communication totale associée à une requête.

De ce fait, nous allons faire une hybridation entre la méthode d'ordonnancement dynamique des jointures avec l'algorithme génétique à cause des avantages des deux méthodes sur l'amélioration du temps de réponse total de la requête.

-Le coût de communication est le temps de communication entre le médiateur et les adaptateurs. Ce coût est lié à la taille des données transférées (taille de la requête, cardinalité du résultat et la taille d'un résultat) et même au paramétrage du réseau (débit de communication, protocole utilisé). Ce coût peut être estimé comme suit :

Coût –communication = temps- d'initialisation + temps-transmission

Où le temps d'initialisation est le temps d'établissement de la connexion auquel on ajoute le temps de transmission des données.

-Le coût de traitement est le temps de traitement de médiateur et le temps de traitement des adaptateurs. Ce coût est lié à la capacité de médiateur et des adaptateurs utilisés. Ce coût peut être estimé comme suit :

Coût –traitement = temps-Médiateur + temps-Adaptateur

De ce fait le coût de communication entre médiateur et l'adaptateur et le coût de traitement représentent le coût de réponse à une requête qu'on doit optimiser dans notre approche.

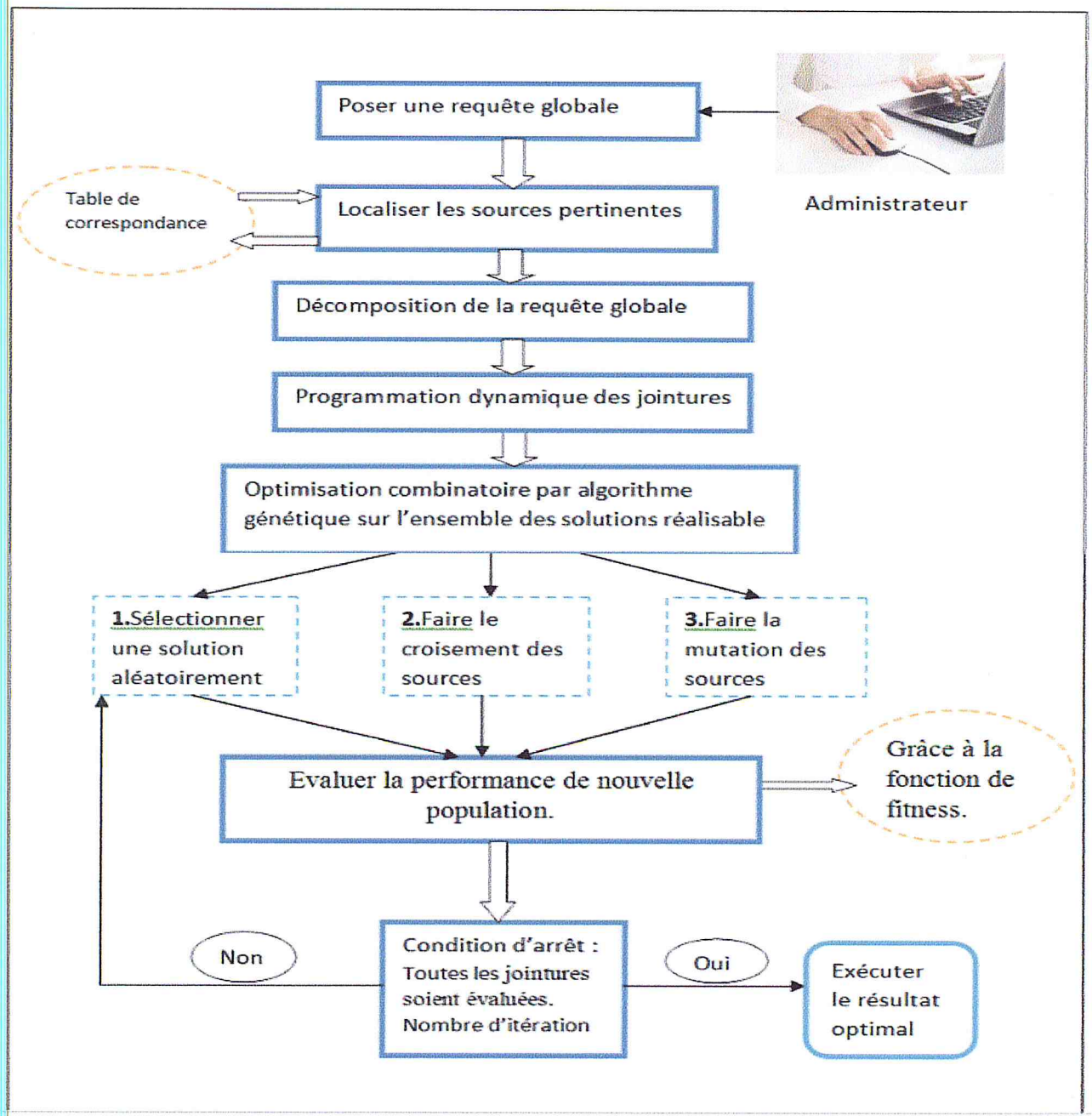


Figure 19 : Notre Processus d'optimisation de requête on utilisant la programmation dynamique et l'algorithme génétique.

4.3. Description de processus :

Etape 1 : Requête globale

On distingue 3 cas possible de poser une requête globale et l'administrateur est libre de choisir n'importe quel cas pour obtenir la réponse à une demande :

1er cas: Permet d'afficher toutes les informations possibles (attributs) d'un concept donné.

```
SELECT * FROM Concept_Global
```

2em cas: Permet d'afficher un ou plusieurs attributs d'un concept donné.

```
SELECT Attributs_Global FROM Concept_Global
```

3em cas: Permet d'afficher un ou plusieurs attributs d'un concept donné avec une spécification de valeur.

```
SELECT Attributs_Global FROM Concept_Global WHERE Attribut_Global=valeur
```

Exemple:

```
SELECT NomMalade, MalAdresse FROM Malade WHERE MalAdresse =' Boufarik '
```

Etape 2 : Localiser les sources pertinentes

```
SELECT nom_att_l, nom_cpt_l, nom_source FROM Table de correspondance WHERE
nom_att_g='NomMalade' AND nom_cpt_g= 'Malade'
```

Etape 3: Décomposition de la requête

Selon le résultat de l'étape précédente, les sous requêtes envoyées sont (**R** désigne la sous requête Relationnel, **X** la sous requête XML et **O** la sous requête Objet relationnel):

R: SELECT NomMalade, MalAdr **FROM** Malade **WHERE** MalAdr = ' Boufarik'

X: SELECT NomPatient, PatAdresse**FROM** Patient **WHERE** PatAdresse= 'Boufarik'

O: SELECT NomMal, AdresseM **FROM** Malade **WHERE**AdresseM = ' Boufarik'

Etape 4: Programmation dynamique des jointures

Dans notre travail nous considérons que les sources sont autonomes elles n'exports pas leurs modèles de coûts. De ce fait l'utilisation d'une méthode d'optimisation statique s'apparaît inutile. Nous adoptons d'utiliser une méthode d'optimisation dynamique qui remplace les méthodes D'optimisations statiques. Nous adaptons la méthode **d'ordonnancement dynamique des jointures.**

L'idée est ici de traiter une jointure le plutôt possible, dès que deux sous requêtes ont produit leur résultat tous les plans d'exécution qui ne commencent pas par cette jointure sont éliminés. Alors on va utiliser cette approche pour faire une optimisation des requêtes dans un environnement hétérogènes et distribuées grâce aux avantages de cette approche : elle prend en considération le temps de réponse réel des sources, et ainsi offre une amélioration du temps de réponse total de la requête.

Etape 5: Optimisation combinatoire par algorithme génétique sur l'ensemble des solutions réalisable

Sachant que les trois sous requêtes sont différentes de nulle, et que chaque sous requête sera envoyée aux deux sources qui correspond à son schéma local, on obtiendra plusieurs possibilités des plans d'exécution avec un ordonnancement différent des jointures.

Pour des raisons d'optimisation et pour minimiser le nombre des plans d'exécution, nous avons décidé d'exécuter les requêtes d'un même schéma local en parallèle, Dans ce cas on obtiendra six possibilités des plans d'exécution après faire le croisement entre les bases et la mutation entre les sources :

1. [(R1 // R2) ∞ (X1 // X2)] ∞ (O1 // O2)
2. [(X1 // X2) ∞ (R1 // R2)] ∞ (O1 // O2)
3. [(R1 // R2) ∞ (O1 // O2)] ∞ (X1 // X2)
4. [(O1 // O2) ∞ (R1 // R2)] ∞ (X1 // X2)

5. [(X1 // X2) ∞ (O1 // O2)] ∞ (R1 // R2)

6. [(O1 // O2) ∞ (X1 // X2)] ∞ (R1 // R2)

Algorithme génétique :

Le grand avantage des algorithmes génétiques est qu'ils parviennent à trouver de bonnes solutions sur des problèmes très complexes, Avec les algorithmes évolutionnaires, nous passons à une catégorie de méta heuristiques, celles des méthodes dites évolutionnaires, qui manipulent un ensemble de plusieurs solutions simultanément.

L'algorithme génétique repose sur une boucle qui enchaîne des étapes de *sélections* et des étapes de *croisements* et de *mutations*.

1-Sélectionner un plan parmi l'ensemble des solutions réalisables aléatoirement par une fonction random.

2 -Faire le croisement des individus sélectionnés par une fonction random de façon à obtenir une population d'enfants (entre bases).

3 -Faire la mutation des individus sélectionnés aléatoirement (entre sources).

Etape 6 : Evaluer la performance de nouvelle population

Les adaptateurs reçoivent les sous requêtes envoyées par le médiateur, chaque adaptateur fait la traduction de sous requête au langage de la source appropriée puis il interroge la source. Pour la sous requête XML une traduction complète sera établit pour aller d'une requête en SQL vers une requête en XQuery (langage de requêtes pour les documents XML). Après la traduction des sous requêtes on obtient :

R: SELECT NomMalade, MalAdr **FROM** Malade **WHERE** MalAdr= ' Boufarik'

```
X: FOR $a IN document ("C:\xmlfiles\bdXml.xml")//Patient WHERE $a/
PatAdresse = "Boufarik"
RETURN <resultat>
{$a/ NomPatient}
{$b/ PatAdresse}
</resultat>

O: SELECT "NomMal", "AdresseM"FROM "Malade" WHERE "AdresseM"
='Boufarik'
```

Pour faciliter la traduction en langage XQuery nous avons choisi d'utiliser StelsXML qui est un type JDBC 4 driver (Java DataBase Connectivity) qui permet d'exécuter des requêtes et d'autres opérations JDBC sur des fichiers XML. Avec l'API StelsXML on peut facilement extraire et traiter les données contenues dans des documents XML en utilisant la syntaxe standard de SQL.

La performance des enfants est évaluée, grâce à la fonction ***fitness,*** C'est la fonction de calcul de coût de chaque opération (temps).

-Le coût de communication est le temps de communication entre le médiateur et les adaptateurs. Ce coût est lié à la taille des données transférées (taille de la requête, cardinalité du résultat et la taille d'un résultat) et même au paramétrage du réseau (débit de communication, protocole utilisé).

Coût –communication = temps- d'initialisation + temps-transmission

-Le coût de traitement est le temps de traitement de médiateur et le temps de traitement des adaptateurs. Ce coût est lié à la capacité de médiateur et des adaptateurs utilisés.

Coût –traitement = temps-Médiateur + temps-Adaptateur

Fonction_de_Fitness = Coût –communication + Coût –traitement

Etape 7 : La condition d'arrêt

La boucle est bouclée, et l'on recommence une phase de sélection pour la reproduction, une phase de mutation, et ainsi de suite.

Un critère d'arrêt permet de sortir de la boucle, est le nombre d'itérations (12 itérations), jusqu'à ce que toutes les jointures soient évaluées pour exécuter le plan optimal.

Algorithme : Optimisation de requête

Entrée : Ensemble de solution possible (Plans d'exécutions).

Sortie : Le plan optimal.

Début

Chaque plan est représenté par un arbre l'ensemble des arbres est les solutions réalisables.

Toutes les sous-requêtes sont envoyées en parallèle aux wrappers, pour être exécutées.

Tant que deux sous requêtes ont produit leur résultat **Faire**

Une valeur de seuil est calculée en fonction du coût des jointures.

Eliminer les arbres qui ne commencent pas par cette jointure.

Tant que NbrItération <= 12 **Faire**

Appliquer l'algorithme génétique {

-Sélectionner un arbre parmi l'ensemble des solutions réalisables aléatoirement.

-Faire le croisement des individus sélectionnés (nouvelle population).

-Faire la mutation des individus sélectionnés.

- Evaluer la performance des enfants, par la fonction de fitness en éliminant les jointures qui ont un coût supérieur au seuil. }

Fin

Fin

Fin

5. Conclusion

Dans ce chapitre nous avons présenté une conception détaillée de notre système d'optimisation des requêtes dans un système de médiation des données hétérogène par l'optimisation combinatoire en utilisant le langage UML, cette étude conceptuelle nous a permet de mettre en évidence les étapes nécessaires pour la création d'un système d'optimisation des requêtes dans un système de médiation des données hétérogène. La prochaine étape consiste donc en la concrétisation de ce que nous avons proposé, en d'autres termes, la réalisation d'une optimisation des requêtes dans un système de médiation des données hétérogène par l'optimisation combinatoire et les outils que nous utilise pour l'implémenter.

Chapitre 5 :

Implémentation

1. Introduction

Après l'expression des besoins et la modélisation du système à développer, nous allons, dans cette partie, présenter en premier lieu les outils qui nous ont servis à construire nos sources de données, puis les outils de développement de notre système et a la fin les principales interfaces de notre application.

2. Les Outils de Mise en œuvre des Sources

2.1. Sql Server

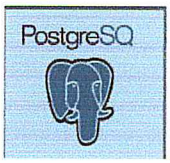
Sql Server est un Système de Gestion de Base de Données Relationnelle (SGBDR) .Ce qui lui confère une très grande capacité à gérer les données toutes en conservant leur intégrité et leur cohérence.

Sql Server est chargé de :

- Stocker les données.
- Vérifier les contraintes d'intégrité définies.
- Garantir la cohérence des données qui stocke, même en cas de panne du système.
- Assurer les relations entre les données définies par les utilisateurs.

2.2. PostgreSQL

PostgreSQL est un système de gestion de base de données relationnelles et objet (SGBDRO) fonctionnant sur des systèmes de type UNIX (par exemple Linux, FreeBSD, AIX, HP-UX, IRIX, Solaris, ...).



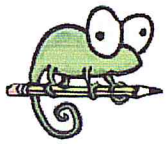
L'une des principales qualités de PostgreSQL est d'être un logiciel libre, c'est-à-dire gratuit et dont les sources sont disponibles.

Nous avons choisi cet outil pour la manipulation de nos bases de données relationnelles objet à cause des raisons suivants :

- Moteur transactionnel.
- Respect des normes SQL.
- MVCC (mécanisme permettant une concurrence efficace sans verrouiller les enregistrements pour assurer l'isolation des transactions).
- Procédures stockées dans de nombreux langages.

- Triggers.

2.3. *Notepad++*



Est un éditeur de texte générique codé en C++, qui intègre la coloration syntaxique de code source pour les langages et fichiers C, C++, Java, XML, HTML, PHP, JavaScript, makefile, art ASCII, doxygen, .bat, MS fichier ini, ASP, Visual Basic/VBScript, SQL, Objective-C, CSS, Pascal, Perl, Python, R, MATLAB, Lua, TCL, Assembleur, Ruby, Lisp, Scheme, Properties, Diff, Smalltalk, PostScript et VHDL ainsi que pour tout autre langage informatique, car ce logiciel propose la possibilité de créer ses propres colorations syntaxiques pour un langage quelconque.

Ce logiciel, basé sur la composante Scintilla, a pour but de fournir un éditeur léger (aussi bien au niveau de la taille du code compilé que des ressources occupées durant l'exécution) et efficace. Il est également une alternative au bloc-notes de Windows (d'où le nom). Le projet est sous licence GPL.

Il ne bloque pas le fichier en cours d'édition et détecte toute modification apportée à celui-ci par un autre programme (il propose de le recharger).

Il a été codé par Don Ho, un informaticien basé à Paris diplômé de l'Université Paris VII - Diderot en 2000.

3. Les Outils de développement

3.1. Langage de Programmation Java

Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton de Sun Microsystems.

Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux, Macintosh, Solaris).

Le langage Java donne aussi la possibilité de développer des programmes pour téléphones portables et assistants personnels. Enfin, ce langage peut-être utilisé sur

internet pour des petites applications intégrées aux pages web (applet) ou encore comme langage serveur (JSP).

3.2. L'EDI NetBeans

NetBeans est un environnement de développement intégré (EDI), placé en *open source* par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, NetBeans permet également de supporter différents autres langages, comme C, C++, JavaScript, XML, Groovy, PHP et HTML de façon native ainsi que bien d'autres (comme Python ou Ruby) par l'ajout de greffons. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web). Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

NetBeans constitue par ailleurs une plate forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme.

4. Présentation de l'application

Cette section a pour objectif de présenter des copies d'écran des interfaces de notre système

4.1. Fenêtre d'Accueil

- **Page d'accueil :** Dans cette page l'administrateur a le droit :
  - Inscrire.
  - Connecter s'il est déjà inscrit.
  - consulter et de modification des schémas (après faire l'authentification).
  - Poser une requête (après faire l'authentification).

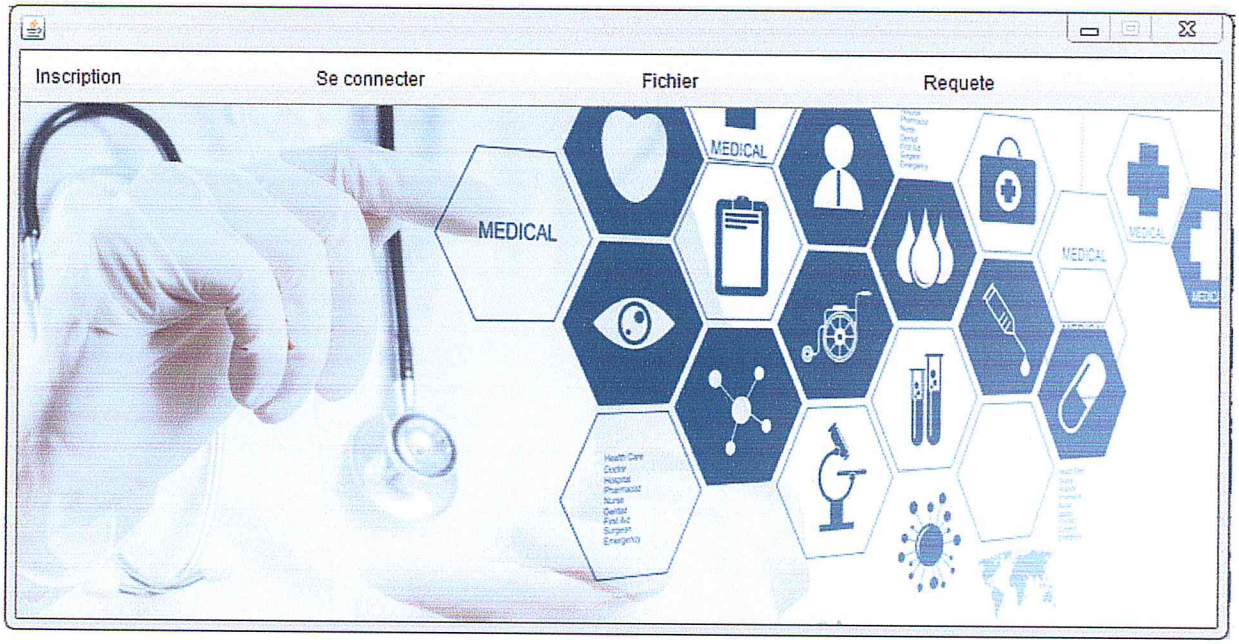


Figure 20 : interface d'Accueil.

4.2. Fenêtre d'inscription

Cette interface permet d'enregistrer les informations entrées par l'utilisateur dans la base de donnée de Système.

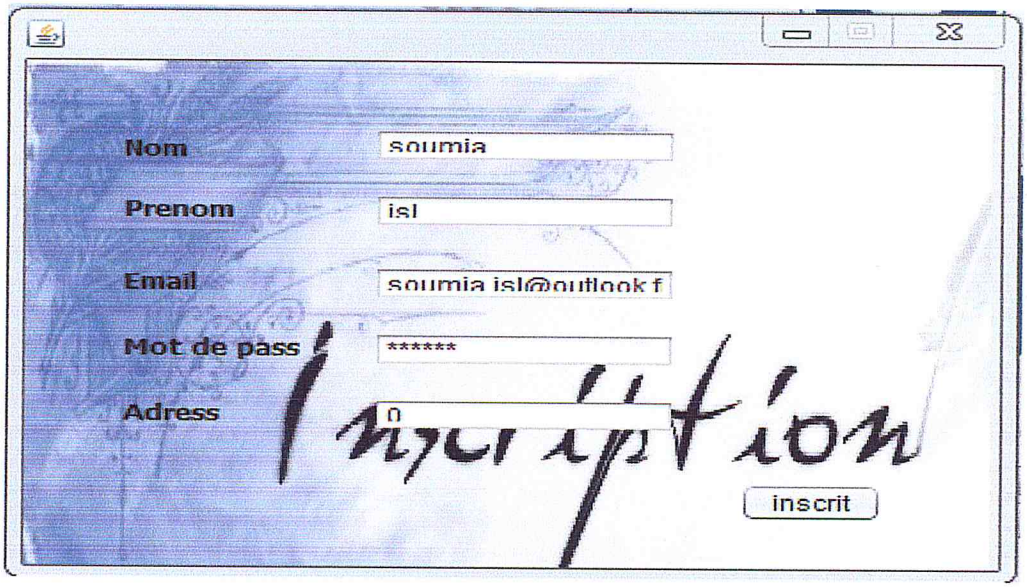


Figure 21 : interface d'inscription.

4.3. Fenêtre d'authentification

L'administrateur s'identifie en saisissant, dans la zone d'identification, un email et un mot de passe. Si ces informations existent dans la base de données de Système; le système affiche la page correspondante à l'administrateur.

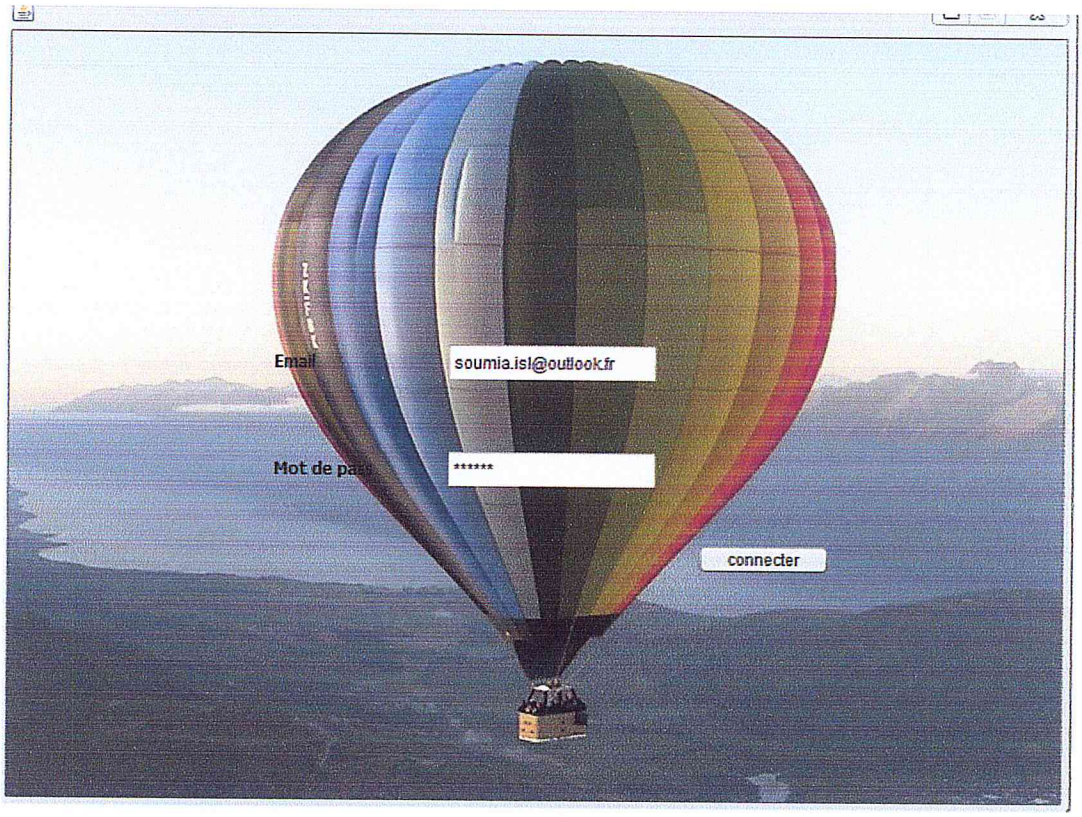


Figure 22 : interface d'authentification.

4.4. Fenêtre de consultation et de modification des schémas locaux:

Cette interface permet de consulter les schémas locaux et de modifier les informations existantes dans ces sources.

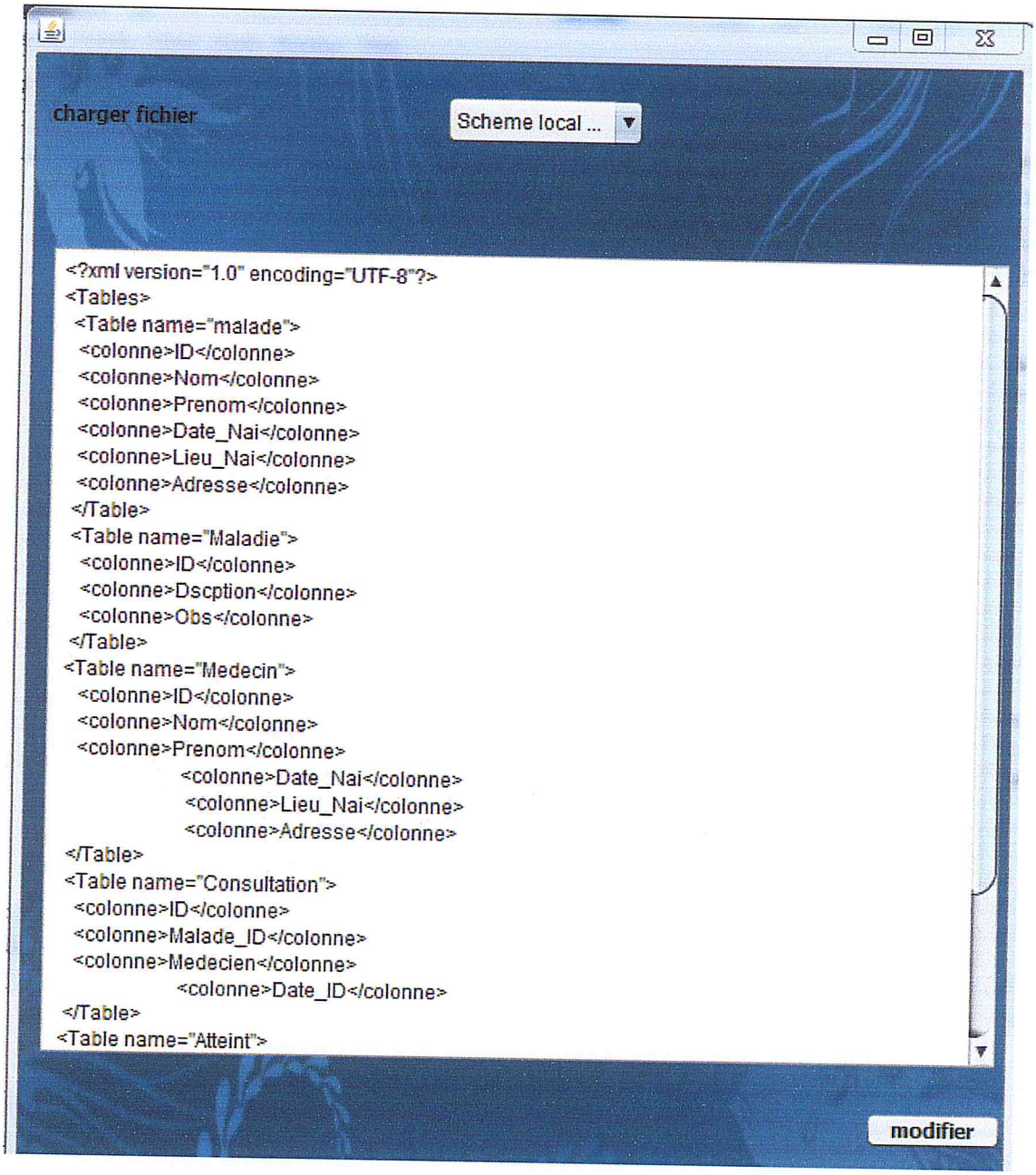


Figure 23 : Fenêtre de consultation et de modification des schémas locaux.

4.5. Fenêtre de consultation et de modification de schéma global

Cette interface donne à l'administrateur le droit de consulter les schémas et de modifier les informations existantes dans ces schémas.

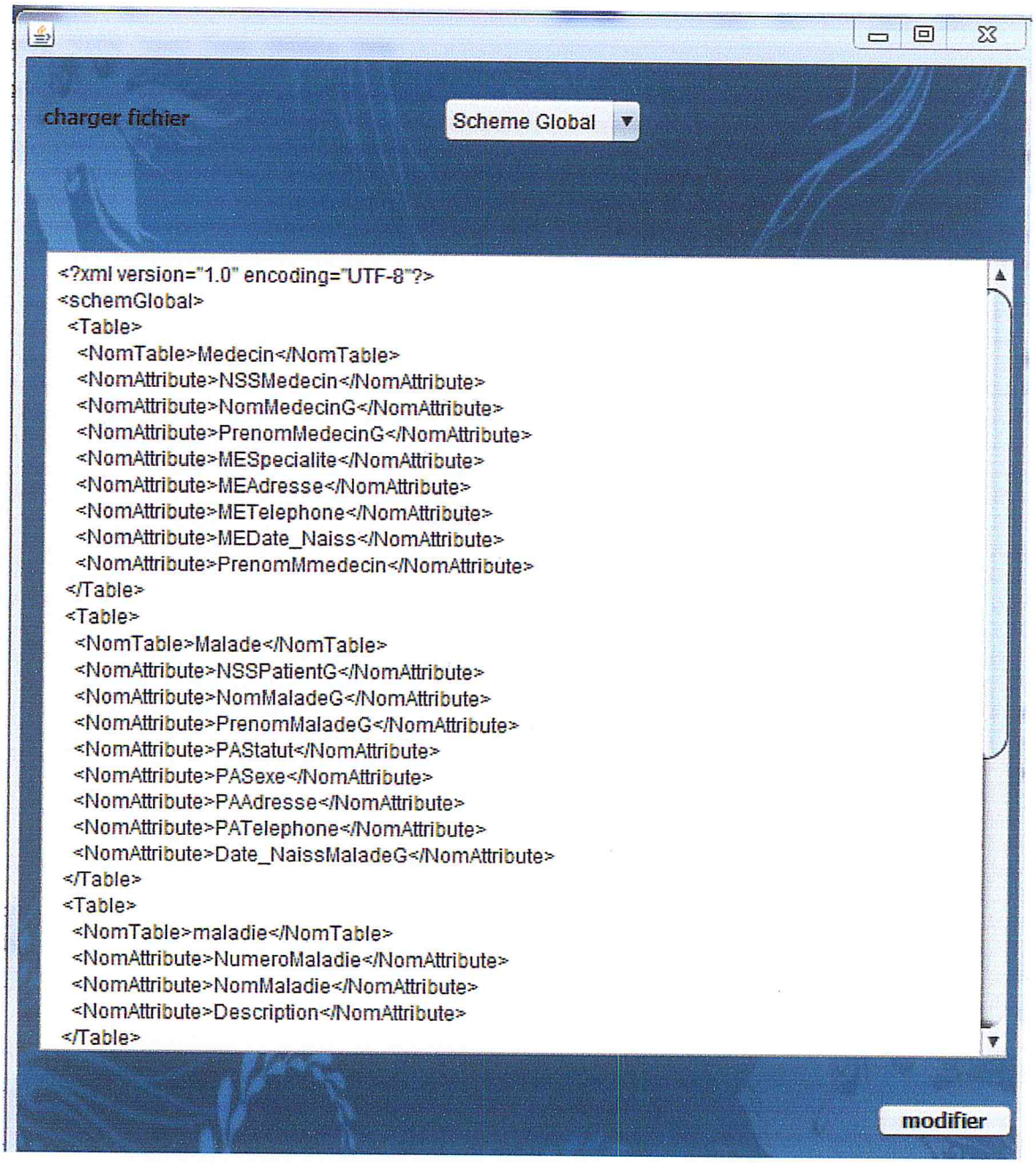


Figure 24 : Fenêtre de consultation et de modification de schéma global.

4.6. Fenêtre de Poser la requête

Cette interface donne l'administrateur le droit de choisir d'entrer:

- Le concept global seulement et on va obtenir comme un résultat toutes les informations (attributs) reliée à ce concept.
- un ou plusieurs attributs d'un concept donné.
- attributs d'un concept donné avec une spécification de valeur.

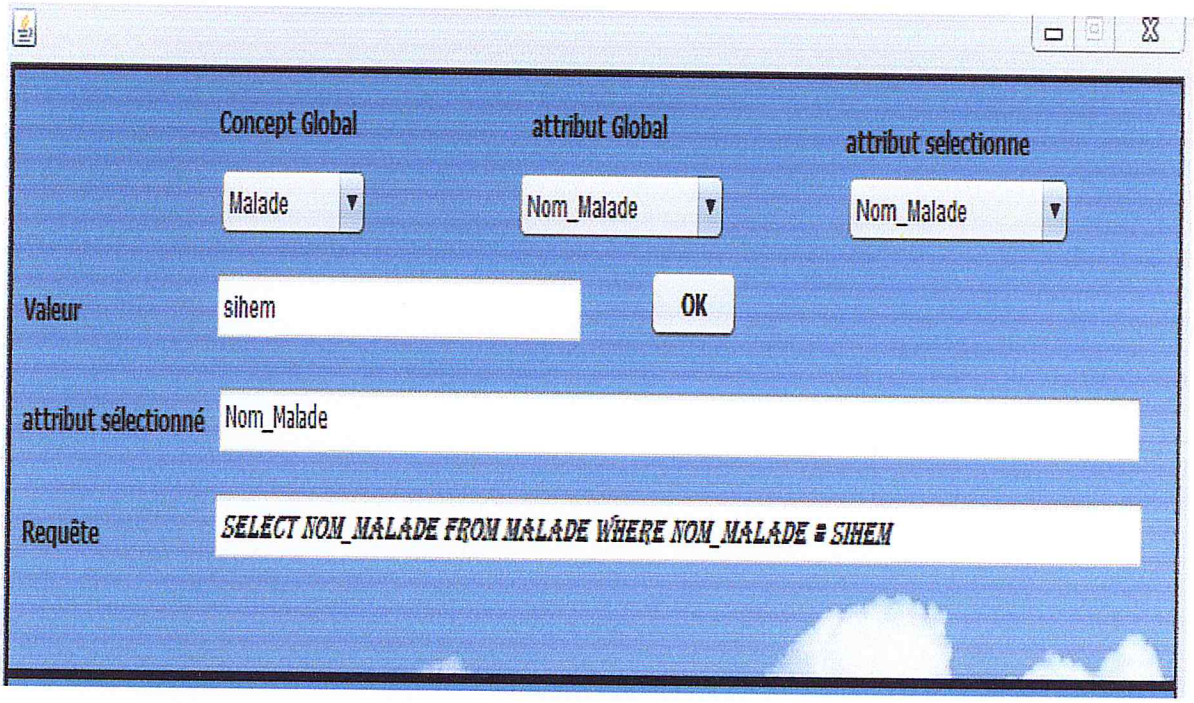


Figure 25 : Fenêtre Poser la requête.

4.7. Fenêtre de Décomposition de requête

Après avoir posé la requête, les sous requêtes envoyées sont des sous requête Relationnel, XML et Objet relationnel.

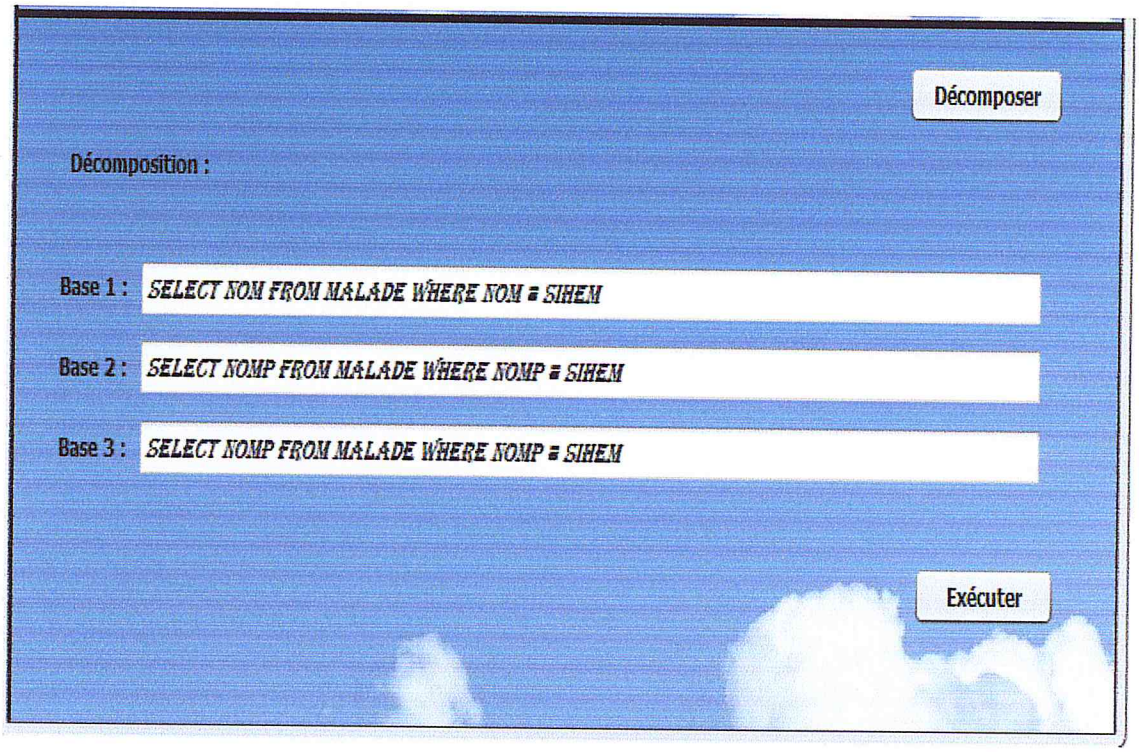


Figure 26 : Fenêtre de Décomposition de requête.

4.8. Fenêtre de Résultat Optimal

Cette interface permet d'afficher le résultat optimal, de voir le plan d'exécution choisi plus la différence entre temps d'exécution en (m/s) d'une même requête exécutée une fois sans l'utilisation de l'optimisation combinatoire et une autre fois avec.

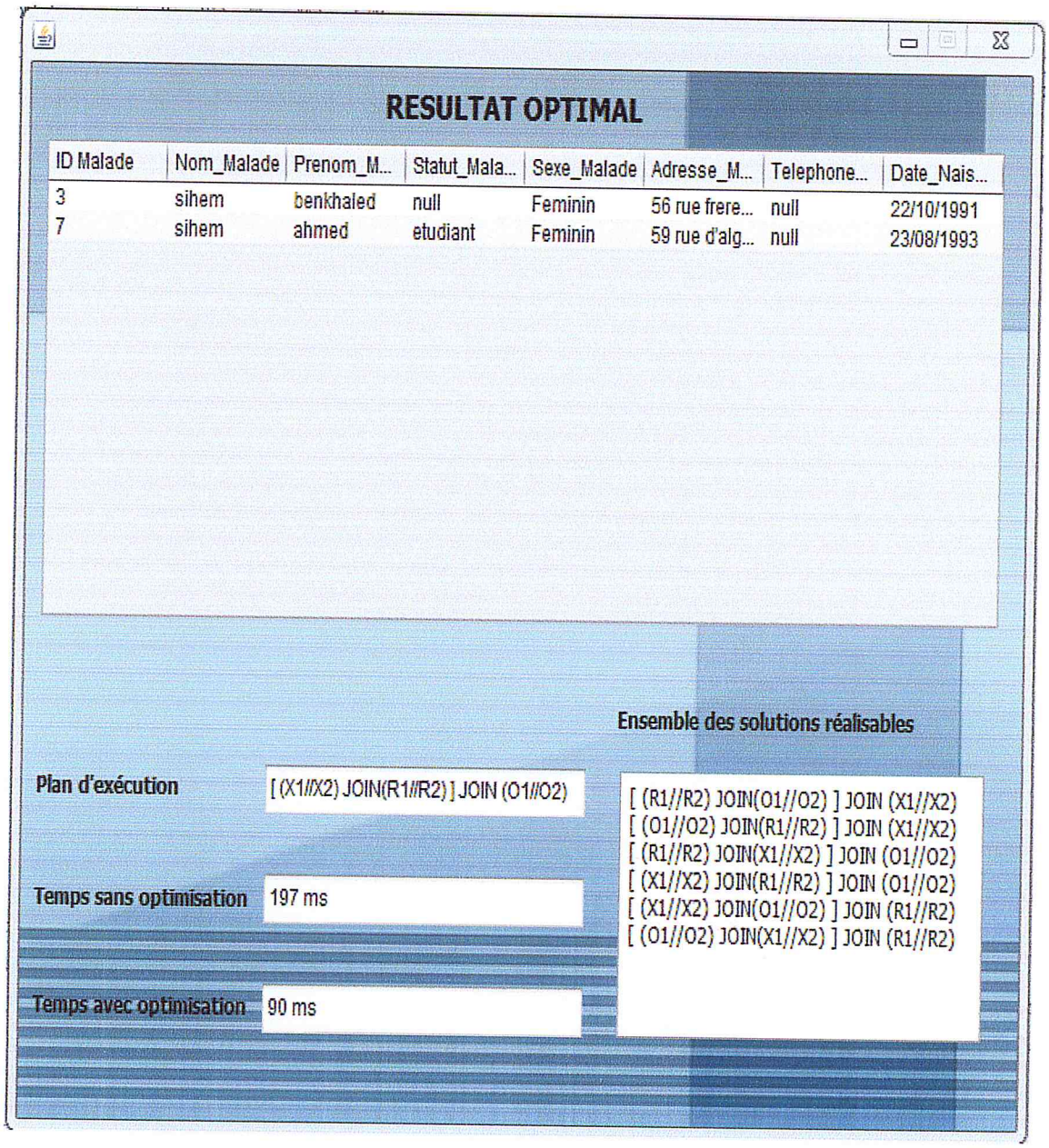


Figure 27 : Fenêtre Résultat Optimal.

5. Conclusion

Dans ce chapitre, nous avons présenté la vue réelle de notre application, et des principaux modules qui la font fonctionner en toute simplicité, rapidité, grâce à une interface intuitive.

Chapitre 6 :

Test et Validation du Système

1. Introduction

D'après ce qu'on a fait dans les chapitres précédents, Le but de notre système consiste à l'optimisation des requêtes du médiateur en utilisant l'optimisation combinatoire. Dans ce chapitre on va faire une validation de notre système par des graphes pour montrer la validité et l'intérêt de l'utilisation de l'optimisation combinatoire sur l'amélioration du temps de réponse dans les systèmes de médiation.

2. La fonction de fitness :

L'utilisation de fonction de fitness permet de calculer le temps de réponse total d'une requête pour chaque itération dans l'implémentation de l'algorithme génétique :

Fitness = Temps_Médiateur + Temps_Adaptateur + Temps_Communication .

On distingue 3 cas possible de poser une requête globale et calculer la fitness pour chaque cas

1 er cas : Une requête avec un concept seulement :

SELECT * FROM Malade.

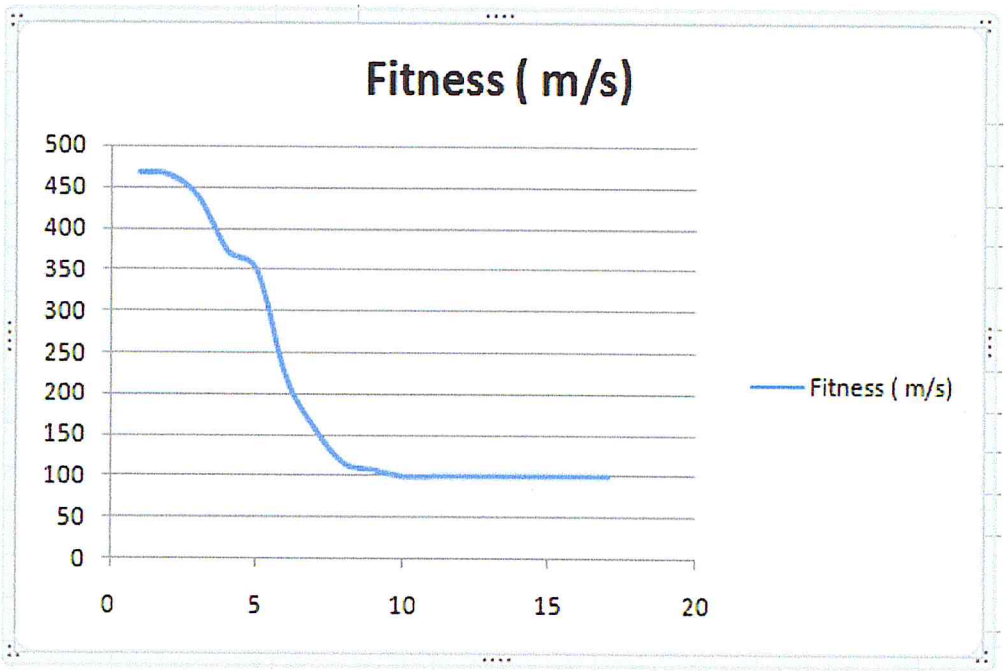


Figure 28 : La courbe du fitness en fonction de nombre d'itération pour requête 1.

2 ème cas : Une requête avec un concept et attribut :

SELECT Nom_M FROM Malade

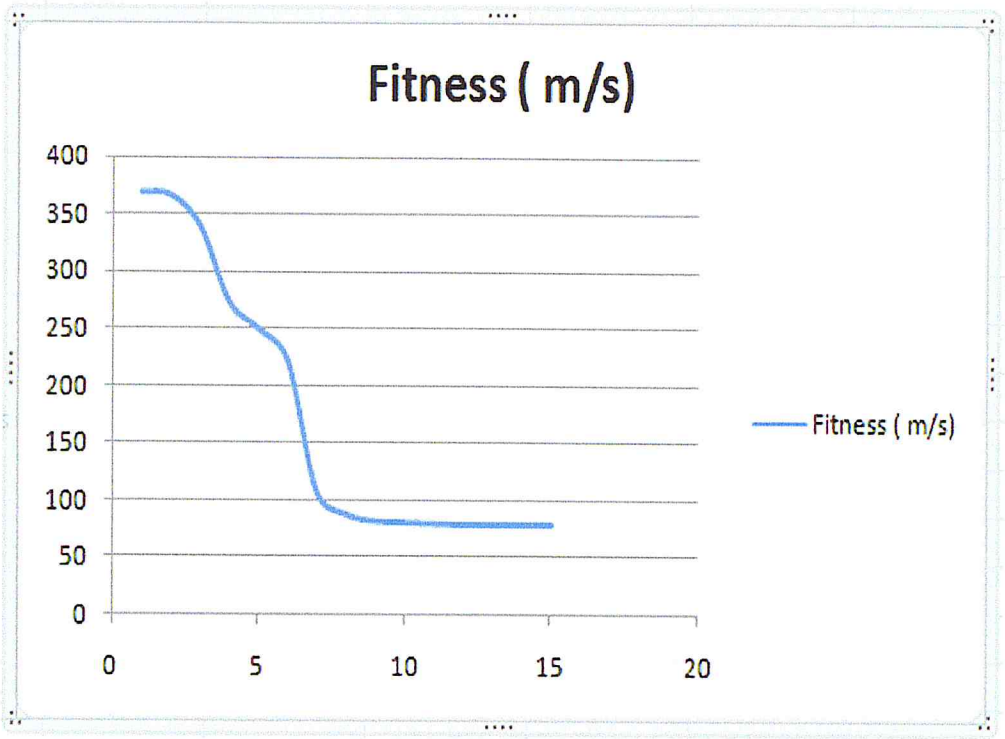


Figure 29 : La courbe du fitness en fonction de nombre d'itération pour requête 2.

3 ème cas : Une requête avec un concept, attribut et une valeur :

SELECT Nom_M FROM Malade WHERE Nom_M ="sihem" .

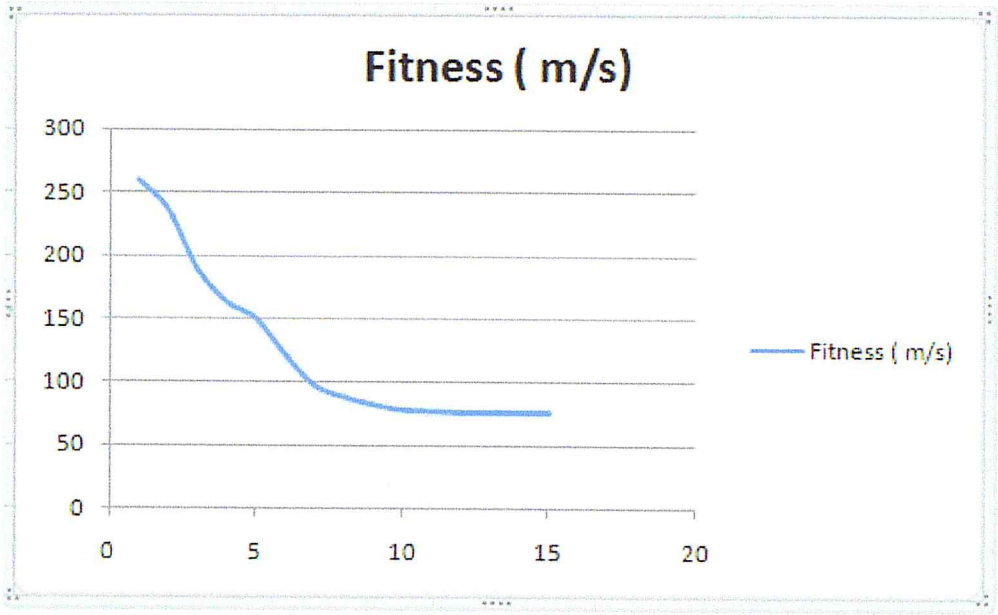


Figure 30 : La courbe du fitness en fonction de nombre d'itération pour requête 3.

Remarque :

Pour valider la fonction de fitness il faut toujours faire plusieurs tests, On remarque dans les courbes de fitness en fonction d'itération que le coût de traitement de tout les requêtes diminuant à chaque itération jusqu'il devient permanent à partir de 12 itérations.

Ce qui montre que la fonction de fitness de l'algorithme génétique offre une amélioration du temps de réponse à une requête.

3. Optimisation combinatoire :

Après l'implémentation de l'algorithme génétique dans notre système d'optimisation des requêtes qui est un des algorithmes évolutionnaire de l'optimisation combinatoire, on a besoin de faire une comparaison entre le temps de réponse à des requêtes avec optimisation et sans optimisation pour pouvoir démontrer la validité de l'utilisation de l'optimisation combinatoire sur les systèmes de médiation.

On lance un ensemble de requêtes sur notre système et on marque la valeur du temps de réponse total pour chaque requête dans le graphe, Le temps de réponse est calculer et noter dans les deux cas « **avec optimisation** » et **« sans optimisation ».**

Les requêtes de test :

Requête 1 : SELECT * FROM Médecin.

Requite 2 : SELECT Nom_M FROM Malade.

Requête 3: SELECT * FROM Malade.

Requête 4: SELECT Nom_M , Prenom_M FROM Médecin.

Requête 5 : SELECT Nom_M FROM Médecin WHERE Nom_M="sihem" .

Requête 6 : SELECT Date_V FROM Visite .

Requête 7 : SELECT Date_V FROM Visite WHERE Date_V="22-10-2015" .

Requête 8 : SELECT * FROM Maladie

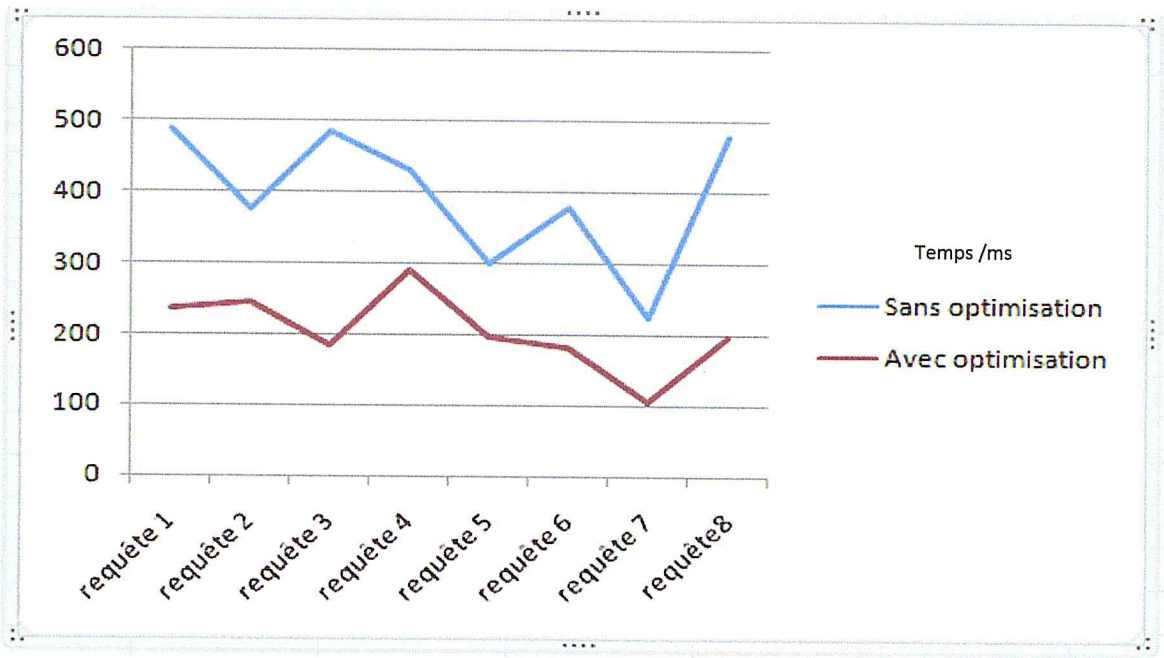


Figure 31 : Les courbes d'optimisation.

Nous remarquons dans les courbes ci-dessous que le temps est élevé sans optimisation pour toutes les requêtes par rapport au temps avec optimisation combinatoire.

Nous déduisons que le traitement des requêtes en utilisant l'optimisation combinatoire est moins couteux en unité du coût.

4. Conclusion :

Après les tests et les graphes de validation que nous avons faite qui permettent de vérifier les résultats de l'implémentation en testant la construction de notre système, on peut conclure que notre système répond aux besoins spécifiés dans les chapitres précédents.

Conclusion Générale

Quotidiennement, nous faisons appel à différentes sources d'information (journaux, livres, personnes, télévisions,…) pour obtenir des éléments de réponse à des questions qui nous intéressent ou nous sont utiles. En général, nos recherches d'information aboutissent rapidement car nous savons à quelles sources nous adresser et comment interagir avec elles. Dans ce contexte, un problème essentiel de l'intégration est de gérer l'hétérogénéité des différentes sources de données. Parmi les systèmes d'intégration nous avons focalisé sur les systèmes de médiation qui sont aujourd'hui un axe de recherche captivant non seulement pour les chercheurs, mais aussi pour les différentes entreprises.

Ce projet porte sur la réalisation d'un système d'optimisation des requêtes par l'approche médiateur en utilisant l'optimisation combinatoire. Pour ce faire nous avons passé par plusieurs étapes notamment la collecte d'information nécessaire à notre étude, la conception du système, l'implémentation et a la fin tests et validation de notre application qui répond aux objectifs.

La solution que nous avons présenté et le projet que nous avons réalisé dans ce mémoire pour la résolution de problème d'optimisation des requêtes de systèmes de médiations n'est en réalité qu'une ouverture vers d'autre travaux car notre système peu encore évoluer et se voir améliorer.

La réalisation de notre mémoire nous a permis d'avoir des notions avancées sur les systèmes de médiation et d'approfondir nos connaissances sur les systèmes de gestion de base de données ainsi sur les outils de développement. Au fait ce projet nous a permis d'intégrer dans la recherche scientifique, de plonger dans la pratique et d'activer nos connaissances acquises durant nos cinq années universitaires.

Références

[AMLY] Amina Azzaz , Mimoun Malki , Ladjel Bellatreche , Youcef Benmimoun Une Approche Ontologique d'Intégration de Sources de Données dans un Environnement de Pair à Pair.

[BA, 07] Mahfoud BALA, Interopérabilité Sémantique des Systèmes d'information Distribués, (2007).

[BAT 86] BATINI C., LENZERINI M. et NAVATHE S., A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys, Vol 18, N°4 December 1986.

[BAR 03] Xavier BARIL. Un modèle de vues pour l'intégration de sources de données XML : VIMIX. Thèse de Doctorat en Informatique, de l'Université des sciences et techniques du Languedoc. Décembre 20.

[BENH 05] Benharzallah Saber Coopération multi agents pour le traitement des requêtes sur des sources de données hétérogènes et distribuées.2005.

[BONN 99] Philippe BONNET, Prise en compte des sources de données indisponibles dans les systèmes de médiation, Thèse préparée au sein de l'INRIA Rhône Alpes (action Médiation du GIE Dyade), 25 janvier 1999.

[CrTo 98]Crainic, T. G. et Toulouse, M. (1998). Parallel metaheuristics. Dans Fleet Management and Logistics. , T. G. Crainic and G. Laporte. Norwell, MA., pages 205–251. Kluwer Academic.

[DANG 03] Tayet Tram DANG NGOC, ≪ Fédération de données semi-structruées avec XML ≫, Université de Versailles Saint-Quentin-en-Yvelines, 2003.

[EVRE 95] Cem Evrendilek Asuman Dogac Query Decomposition, Optimization and Processing in Multidatabase Systems Software Research and Development Center Scientific and Technical Research Council of Turkiye Middle East Technical University (METU) Turkiye 1995.

[GAJO 79]Garey, M. S. et Johnson, D. S. (1979). Computer and Intractability: A Guide to the Theory of NP-Completeness. New York, W.H. Freeman and Co.

[GOH 94] GOH C. H et al. Context Interchange: Overcoming the Challenges of Large-scale Interoperable Database Systems in a Dynamic Environment.In Proceeding of the 3rd International Conference on Information and Knowledge Management (CIKM'94), Gaithersburg, Maryland, USA, 1994.

[JOU 01] Fabrice JOUANOT, DILEMMA : vers une coopération de systèmes d'informations basée sur la médiation sémantique et la fusion d'objets Thèse de doctorat, 21 Novembre 2001Université de Bourgogne, UFR des Sciences et Techniques.

[KIM 91] W. KIM et J. SEO, Classifying Schematic and Data Heterogeneity in Multidatabases Systems, Journal of the ACM 1991.

[LAU 06] UML2 L'apprentissage à la pratique FNAC amazon, 2006.

[LENZ 03] Maurizio Lenzerini, Zoran Majkic , "General framework for query reformulation" , SEWASIE project, 10/02/2003.

[LEVY 00] Alon Y. Levy, Logic-based techniques in data integration, Department of Computer Science and Engineering University of Washington, 2000.

[Liu 11] Tianxiao Liu, «Proposition d'un cadre générique d'optimisation de requetes dans les environnements hétérogènes et réparties,Université de cergy-pontoise,2011.

[NAAK 99] Huber Naake, ≪Modèle de coût pour médiateur de base de données hétérogènes ≫, université de versailles saint- Quentin-en-yvelines, 28 septembre 1999.

[OUK 93] OUKSEL A. et NAIMAN C., Towards the Design of Semantic Cooperation Protocol in Heterogeneous Database Systems – In SCHEK H., SHETH A. et CZEJDO B.? Eds., 3rd International Workshop on Research Issues in Data Engineering (RIDE'93)- Interoperability in Multidatabases Systems Vienna, Austria, IEEE Computer Society Press 1993.

[OUK 99] OUKSEL A.et SHETH A., Semantic Interoperability in Global Information Systems, ACM SIGMOD Record, Vol 28, N°1 Mars 1999.

[OUS 99] Mourad Chabane OUSSALAH, Génie objet, analyse et conception de l'évolution, IRIN, Faculté des Sciences, Université de Nantes 1999 Oussalah@irin.univnantes. Fr.

[OSLA 96]Osman, I. et Laporte, G. (1996). Metaheuristics : A bibliography. Annals of Operations Research, 63(5):511–623.

[PAE 98] PAEPCKE A. et al. Interoperability for Digital Libraries World Wide. Communications of the ACM (CACM) 1998.

[PetH 08] P. Colomb, et H. Jaudoin vers un système d'intégration d'information flexible2008.

[PROC, 08] pierre Gérard.Processus de développement logiciel, (Université de Paris 13)2008.

[RAH 05] Ahmed RAHNI. AMIDHA : Une approche médiateur d'intégration de sources de données hétérogènes et autonomes. Mémoire de stage effectue à l'ENSMA Université de Poitiers. Juillet 2005.

[SHE 90] SHETH A., LARSON J. Federated database systems for managing distributed, heterogeneous, and autonomous databases, ACM Computer Surveys 22, 3 Sept. 1990.

[SHE 92] SHETH A., KASHYAP V. So Far (Schematically) yet So Near (Semantically), In Proceedings of IFIP DS-5 Conference Semantics of Interoperable Databases Systems, (Nov. 16-20. Lorne, Australia), November 1992.

[TAME 99] M. Tamer Ozsu, Patrick Valduriez, principal of distributed database systems, second edition, Prentice-Hall, Upper Saddle River, New Jersy 07458, 1999.