dF.S.D ... N° D'ordre :

Université Saâd DAHLAB de Blida

Faculté des sciences

Département d'Informatique

Mémoire présenté par :

M^r **CHERIFI Hamid et SOURI Youcef**

Pour l'obtention du diplôme de Master

Domaine : Mathématique et Informatique

Filière : Informatique

Spécialité : Ingénierie des logiciels

Sujet :

Optimisation Des Requêtes Dans Un Système De Médiation Des Données Hétérogènes

Soutenu le : Juillet 2012, devant le jury composé de :

M^me **FAREH**	Promotrice
	Présidente
	Examinateur
	Examinateur

MA-004-85-1

Remerciement

Nous remercions avant tout dieu le tout puissant qui nous a aidés à réaliser ce travail

Nous tenons à remercier notre promotrice Mme ***FAREH*** *pour son aide et sa patience et sa disponibilité et sa compréhensibilité*

Nous remercions le membre de jury pour nous avoir fait l'honneur de juger notre travail.

Nous sommes reconnaissantes à Tous nos enseignants qui nous ont facilité la compréhension et la maitrise.

Nous tenons à remercier également et énormément nos très chers parents ainsi toute personne qui nous a aidés de pré ou de loin.

Dédicace

e 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 ;

A mes chères grands mères et mes grands pères

A tout ma famille, mes tantes, mes oncles, mes cousins(es) et mes voisins ;

A mes amis Mohamed, Amroun, Ahmed Amine.

A tous mes camarades d'étude Abdelhek, Rachid

,Sid ali, Ayoub, Amel

A mon binôme CHERIFI Hamid et à toute sa famille

A notre promotrice M^{me} Fareh. ;

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

YOUCEF

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 ;

A mes chères grands mères et mes grands pères

A tout ma famille, mes tantes, mes oncles, mes cousins(es) et mes voisins ;

A mes amis Fateh, Bachir, Bassem, Anes, Ali, Youcef...

A tous mes camarades d'étude Abdelhek, Mohamed Abdelrazek, Youcef

A mon binôme SOURI Youcef et à toute sa famille ;

A notre promotrice M^me^ Fareh. ;

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

HAMID

Sommaire

PARTIE 1 : Etat de l'art

Chapitre 2 : Etat de l'art des systèmes de médiation.

Chapitre 2 : Etat de l'art des différentes approches existantes pour l'optimisation des requêtes.

PARTIE 2 : Conception et Implémentation

Chapitre 3 : Conception du système d'optimisation des requêtes

Liste des acronymes

GAV	Global As Views
LAV	Local As Views
XML	Extended Markup Language
RDF	Ressource Description Framework
MOMIS	Mediator Onvironment for Multiple Information Source
PAT	Phisical Algebra Tree
LAT	Logical Algebra Tree
TGV	Tree Graph View
SGBD	System Gestion Base De Données.
UML	Unified Modeling Language
OMT	Object Modeling Technique
OOSE	Object Oriented Software Engineering
OMG	Object Management Group
IDE	Environnement de développement intégré
JDBC	Java DataBase Connectivity

Les Figures

Les tableaux

Résumé

Nées des besoins de l'intégration de données, les systèmes de médiations sont à l'heure actuelle au cœur des plusieurs travaux de recherches, Un médiateur donne à l'utilisateur l'illusion d'interroger un système homogène et centralisé en lui évitant d'avoir à trouver les sources de données pertinentes pour sa requête, de les interroger une à une, et de combiner lui-même les informations obtenues

L'Optimisation des requêtes est un module fondamental dans les systèmes de médiations. Une requête posée sur un système de médiation de bases de données sera divisée et envoyée à toutes les bases contenant les attributs figurant dans la requête. Cependant, il n'y a que certaines bases qui contiennent des réponses.
Le but de ce travail est d'améliorer l'exécution des requêtes sur un système de médiation de sources de données hétérogènes (Relationnelle, XML). Pour permettre au médiateur de sélectionner les sources concernées par une requête et évite l'envoi de la requête à toutes les sources.

Mots clés : Médiation, Optimisation, requête, sources de données hétérogènes.

Abstract

Born of the needs of data integration systems, mediations are currently at the heart of many research works, a mediator gives the user the illusion of a seamless query and centralized by avoiding d having to find the sources of data relevant to its application to question them one by one, and combine the information itself obtained
The Query optimization is a fundamental unit in mediation systems. A query posed on a mediation system databases will be divided and sent to all databases containing the attributes in the query. However, there are only certain databases that contain answers.
The aim of this work is to improve query performance on a system of mediation of heterogeneous data sources (relational, XML). To allow the Ombudsman to select relevant sources for a query and avoids sending the query to all sources

Keywords: Mediation, Optimization, Query, heterogeneous data.

1. Contexte de travail

L'énorme quantité d'informations disponibles sur des sources de données autonomes, distribuées et hétérogènes, nécessite une stratégie de recherche, de sélection et d'analyse, de plus en plus performants, pour localiser et d'extraire précisément l'information désirée. Les systèmes d'intégration d'informations offre un accès à de multiples sources de données à travers des requêtes sur un schéma global. Une requête posée au système de médiation de bases de données sera divisée et envoyée à toutes les bases contenant les attributs figurant dans la requête. Mais bien évidemment, il y'a certaines base qui ne contiennent aucune réponse.

2. Problématique et motivations

-L'explosion du nombre de sources d'information multiplie les besoins de Techniques d'intégration des sources de données autonomes et hétérogènes. L'intégration des données est le processus par lequel plusieurs sources de données autonomes, réparties et hétérogènes (où chaque source est associée à un schéma local) sont intégrées sous forme de source unique représentée par un schéma global.

-Un système d'intégration est, soit un entrepôt de données (approche matérialisé), soit un médiateur (approche virtuelle).nous intéressons aux systèmes de médiations Une requête posée sur un système de médiation de bases de données sera divisée et envoyée à toutes les bases contenant les attributs figurant dans la requête. Cependant, il n'y a que certaines bases qui contiennent des réponses. Ce projet s'intéresse au traitement des requêtes, et plus particulièrement à l'optimisation des requêtes.

La question principale est de savoir comment optimiser l'exécution des requêtes de médiation.

3. Objectifs de travail

-Le projet rentre dans la problématique générale de la médiation des données hétérogènes, parmi ces données hétérogènes on s'intéresse aux données structurés qui sont représentés par le modèle relationnel, et semi structuré qui sont présentées par XML. Notre but est de comment optimiser la requête d'une manière qu'elle ne sera envoyée qu'aux sources contenant sûrement des réponses.

Le but de ce travail est d'améliorer l'exécution des requêtes sur un système de médiation de sources de données hétérogènes et autonomes. Pour permettre au médiateur

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 des données hétérogènes.

4. Organisation du mémoire

Afin d'atteindre les objectifs cité ci-dessus, notre mémoire s'articulera autour de deux parties

- Etat de l'art qui contiendra deux chapitres :

- Chapitre I: Nous présentons dans le premier chapitre les Aspects caractéristiques de la médiation de données.
- Chapitre II: Dans ce Chapitre nous présentons quelques Approches existants pour l'optimisation des requêtes.

- Conception et implémentation qui contiendra trois chapitres :

- Chapitre III: Ce chapitre comporte la conception de notre système de médiation par le langage de modélisation UML, ainsi qu'une architecture générale de notre système.
- Chapitre IV: Dans ce chapitre nous décrirons les différents outils utilisées dans notre projet, ensuite nous présenterons une démonstration globale de différentes taches à accomplir par notre application

1. Introduction

Aujourd'hui différentes organisations doivent faire face au challenge de faire interopérer des systèmes de bases de données afin de pouvoir réaliser des fonctions critiques.

La répartition et le couplage de très grandes bases de données posent des problèmes qui sont, d'une part, étroitement liés à l'hétérogénéité des données, à leur différence sémantique, mais aussi aux différences en termes de fonctions offertes, de disponibilité de coopération (autonomie), et d'accessibilité des sources. Il est donc nécessaire de concevoir des solutions (infrastructures, protocoles d'accès, architectures) permettant d'offrir en permanence sur le réseau des ressources de manière transparente ; et de construire des systèmes de traitement

de données sur de grandes quantités de données multiformes stockées dans des bases interconnectées

Pour cela différent 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.

2.1.1. Distribution et hétérogénéité

Dans plusieurs systems d'information les données sont réparties dans des sources différentes, elles-mêmes également réparties dans des machines géographiquement distribuées.

Un système d'information est homogène si le logiciel qui gère les données est le même sur tous les sites, les données ont le même format et structure et appartiennent a un même univers de discours.

Un système hétérogène est celui qui n'adhère pas a toutes les caractéristiques d'un systeme homogène, c'est-a-dire, le systeme d'information utilise des langages de programmation et d'interrogation, des modèles, des SGBD différents.

L'hétérogénéité provient des différents choix qui sont faits pour représenter les informations issues du monde réel dans un format informatique.

La classification proposée par Fabrice JOUANOT, DILEMMA présente les conflits des données en trois niveaux : conflits syntaxiques, conflits structurels et conflits sémantiques **[GEN, 04]**

a) Conflits syntaxiques

C'est l'utilisation de modèles de données différents. La même information est représentée par des syntaxes et des concepts différents d'un modèle à un autre (relation dans le modèle relationnel, classe dans lemodèle objet, balise dans XML, etc.)

b) Conflits structurels

Résultent d'une structuration et classification différente des informations. Ils sont étroitement liés aux choix de conception :

- **Les conflits de schémas :** résultent de l'utilisation de différents concepts pour représenter le même objet. Une information peut être représentée par une entité dans un système (S1) et par une relation dans un autre système (S2)

- **Les conflits de généralisation / spécialisation :** résultent des différences de

Hiérarchisation des informations.

Exemple : dans un système (S1) les données sont regroupés dans un seul objet EMPLOYEE alors que dans (S2) on utilise deux objets DIRIGENT et EMPLOYEE.

- **Les conflits d'agrégation :** résultent d'un niveau de granularité différent de deux systèmes. La valeur d'un attribut sur un système correspond à une agrégation des valeurs de plusieurs attributs sur un autre.

Exemple : Nous disposons de moyennes annuelles des étudiants dans un système (S1) et le classement des étudiants dans (S2)

- **Les conflits de typage :** résultent des différences de typage pour le même objet dans les différents systèmes de la coopération.

Exemple : la valeur d'un attribut âge est représenté comme numérique sur le système (S1) et comme chaine de caractère sur le système (S2).

- **Les conflits de complétude :** apparaissent lorsque des objets se correspondent partiellement sur les différents systèmes. C'est-à-dire qu'une partie d'un objet du système (S1) trouve une correspondance sur le système (S2).

Exemple :

(S1) : Véhicule (Code_V, marque, catégorie, couleur, date, année)

(S2) : Véhicule (N°_V, marque, catégorie, date, couleur)

Dans (S1) l'attribut Date contient seulement Moins, Jours alors que dans (S2) la date contient Année, Moins, Jours.

c) Conflits sémantiques

Proviennent des différences d'interprétation des informations partagées entre différents domaines d'application. Plusieurs types de conflits sémantiques apparaissent :

Conflits de noms : problèmes taxonomiques et linguistiques

Conflits de valeurs : problèmes d'unités et d'échelle

Conflits cognitifs : problèmes de signification

Exemple : La table Employé dans deux systèmes différents

(S1) : Joueur (id_joueur, Nom, prénom, Adresse, Poids)

Poids : exprimé en kg

(S2) : Player (player_id, fname, lname, Adr, Poids)

Poids: exprimé en gramme

Les systèmes S1 et S2 sont hétérogènes et divers conflits apparaissent :

- Conflits de noms : joueur et player, id_joueur et player_id, etc.
- Conflits de valeurs : Poids (en kg) et Poid (en gramme) **[BA, 07]**

2.1.2. Autonomie

Les bases de données sont souvent gérées par de systèmes autonomes. il est important de comprendre l'autonomie et la manière de la gérer lorsque une base de données partage ses données avec d'autres bases. Il y a différents niveaux d'autonomie :

- **Conception** : les bases de données locales ont leurs propres modèles de données, langage d'interrogation, interprétation sémantique des données, contraintes, fonctions, etc.

- **Communication** : les bases de données locales décident quand et comment répondre aux requêtes.
- **Exécution** : les bases de données locales n'ont pas besoin d'informer d'autres systèmes de l'ordre d'exécution des transactions locales ou des opérations externes. Un système local ne distingue pas les opérations locales et globales.
- **Association** : les bases de données locales décident quand se connecter et se déconnecter du systeme global. Elles décident également quelles fonctions et données partagé, avec quels types d'utilisateurs. Des informations statistiques (cout, performance, vitesse d'exécution) ne sont pas toujours partagées par les bases de données, ce qui complique l'optimisation de requêtes locales.

2.1.3. Interopérabilité

L'interopérabilité implique la possibilité de pouvoir demander et recevoir des services entre des systèmes interopérables et pouvoir utiliser leurs fonctions. Une forme limitée d'interopération est l'échange de données ou un systeme envoie périodiquement des données vers un autre systeme. Dans ce cas les données peuvent être interdépendantes, ce qui implique que les données et les fonctions de systemes différents sont dépendantes, même si cela est transparent au niveau applicatif. La gestion de données interdépendantes implique la gestion de contraintes d'intégrité.

De manière générale, on considère que deux systemes d'informations sont interopérables si les conditions suivantes sont vérifiées :

(i) Ils peuvent échanger des messages et des requêtes.
(ii) ils peuvent opérer comme une unité pour réaliser une tache commune.

Ces conditions sous entendent que les systemes interopérables peuvent utiliser les fonctions des uns et des autres, fonctionner comme des clients et des serveurs et communiquer même si leurs composants internes sont incompatibles.

Le partage de données et de fonctions offertes par des différentes bases de données peut être réalisé selon des stratégies différentes : intégration, interopérabilité, interdépendance et échange.

2.2. Taxonomies des systèmes de médiation

Les systèmes capables de supporter l'interopérabilité et plusieurs niveaux d'intégration entre bases de données sont les systèmes de bases de données distribuées et hétérogènes. Un système de médiation fournit un point d'accès unique a un ensemble de sources de données.

Les sources locales sont intégrées dans le sens qu'un seul schéma global (médiateur) est construit a partir des schémas exportes par les sources locales.

Les systèmes de médiation souvent classifies par rapport a des paramètres tels que le degré d'intégration des données, la gestion des données, l'expression et le traitement de requêtes. A. Sheth et J. Larson. propose une classification de ces systèmes selon les dimensions distribution, hétérogénéité et autonomie. D'autres classifications ont été proposées. nous présentons dans la suite deux taxonomies des systèmes de médiation ayant des critères d'analyse différents.

2.2.1 Du point de vue intégration de données

L'intégration de données implique généralement l'accès transparent et uniforme des données gérées par différentes bases de données. Le mécanisme clé qui permet d'intégrer les données est le *schéma global* compose complètement ou partiellement de schémas locaux. Selon le niveau d'intégration des données entre plusieurs bases de données les systèmes sont faiblement ou fortement couples.

Des exemples de systèmes fortement couples sont les systèmes multi bases et les fédérations de bases de données.

Les systèmes de bases de données hétérogènes et distribuées sont des systèmes faiblement couplés.

2.2.2 Du point de vue utilisation de données

Il y a des systèmes orientes vers l'interrogation et des systèmes qui permettent la modification des données. R. Domenig et K. R. Dittrich propose une classification des premiers en prenant comme critère l'intégration physique des données :

- **Système virtuellement intègre**

Les données restent dans leurs sources et sont récupérées à la demande lors des interrogations. Selon la stratégie utilisée pour intégrer les données :

(i) les systèmes faiblement couplés intègrent des données structurées, non structurées ou semi-structurées (par ex.*mediated query systems*) ;

(ii) les systèmes fortement couplées intègrent, en général, des données structurées (p. ex. fédérations de bases de données) ;

(iii) les systèmes de recherche d'information intègrent des données natives sans prendre en compte leur structure (p. ex. les méta-moteurs de recherche).

- **Système matérialise**

Les données provenant de sources différentes sont intégrées sur une seule base. Selon le type de données intégrées les systèmes sont des SGBD universels qui intègrent des données structurées ou des entrepôts de données qui intègrent des données structurées et dérivées **[GEN, 04]**

2.3. Systèmes de médiation de données

Les systèmes d'intégration de données offrent des architectures d'interopérabilité sur une fédération de sources de données distribuées, autonomes et hétérogènes. Ils permettent d'accéder à 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. **[BOU, 08]**

2.3.1. 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

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

-G représente 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) décrivant la structure des sources participantes au processus d'intégration

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

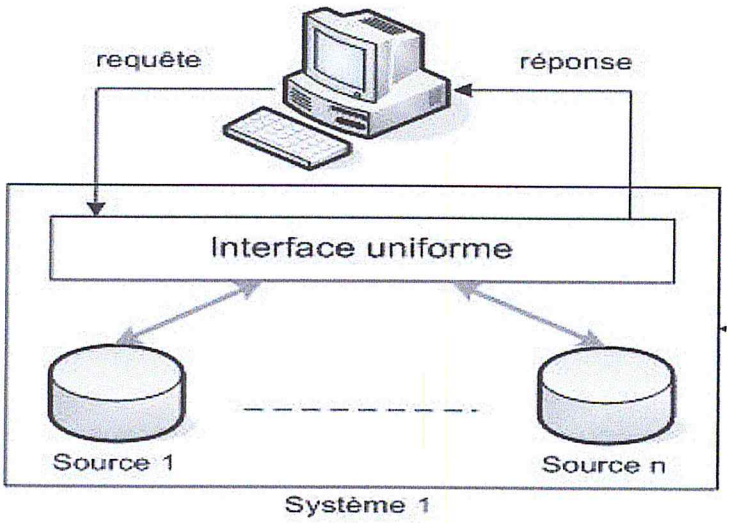


Figure 1 : Composante d'un system de médiation.

2.3.2. 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 intègre (ou schema global) qui sera utilise comme interface d'accès aux sources de données. La construction du schema global à partir des schémas locaux est une tache difficile. Cette difficulté est liée au fait que les sources stockent différentes sortes de données, en différents formats, ayant différentes significations et associées aux différents noms.

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

- **La préintégration** : Cette phase vise à préparer l'intégration des schémas en les rendant plus homogènes. Elle consiste principalement à traduire les schémas initiaux dans un modèle de données commun (réduction de l'hétérogénéité syntaxique). Elle s'attache également à enrichir leur sémantique.
- **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).

2.3.3. 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 a ce niveau sont la perte d'information, la taille des données générées et la Performance des traitements sur ces données. L'exploitation de la structure de données a transformer (types, schémas) joue un rôle crucial dans ce contexte.

- **Traduction de requêtes:** La traduction de requêtes d'un langage (exp. XQuercy) 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 Médiateurs

L'approche d'intégration par médiation constitue sans doute aujourd'hui la solution la plus courante pour relier différentes sources qui cette fois, ne correspondent pas nécessairement a des bases de données. La notion de médiateur a été initialement proposée par Wiederhold G. Il définit un médiateur comme suit :

Un médiateur doit être vu comme une couche logicielle permettant d'accéder de manière transparente Pour l'utilisateur a différentes ressources reparties et hétérogènes. Pour cet accès, le médiateur exploite des connaissances qui sont utiles à différents services (interrogation, localisation des ressources notamment) et concerne le type d'accès (accès en lecture pour la médiation), le type De données qu'il est possible d'intégrer (structurée, semi structurée ou non structurée pour la médiation)

3.1. Caractéristiques

La médiation repose sur un composant essentiel, appelé médiateur, chargé de répondre à des besoins à partir de connaissances mises à sa disposition. Le médiateur permet de localiser l'information et de résoudre les conflits schématiques et sémantiques. Un composant secondaire, appelé Wrapper, sert d'interface avec les SI composants. Le Wrapper résout les conflits syntaxiques en présentant les données dans le modèle de médiation.

3.1.1. Médiateur (Schéma global)

Un médiateur est un logiciel qui offre une interface unique et transparente à plusieurs bases de données hétérogènes et distribuées. Il est chargé de la localisation des données pertinentes et de la résolution des conflits (structurels et sémantiques). Un ensemble de connaissances sont mises à la disposition du médiateur pour lui permettre la génération d'un plan d'exécution pour traiter une requête exprimée dans le langage de médiation. Le médiateur constitue l'interface entre les utilisateurs et les données partagées par la coopération. Pour accéder aux bases de données locales, le médiateur fait appel à des Wrappers.

3.1.2. Wrapper (adaptateur)

Un Wrapper de données est un logiciel qui convertit les requêtes exprimées dans le modèle de médiation provenant d'un ou de plusieurs médiateurs vers le modèle des bases de données locales. Il convertit les données, résultats d'une requête, du modèle des bases de données locales vers le modèle de médiation. Un Wrapper de données offre une interface homogène locale d'accès aux données sources (résolution des conflits syntaxiques).

3.1.3. 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. Un standard de langage de médiation est propos

3.2. Type de médiation

Selon la manière avec laquelle le médiateur traite les requêtes et utilise la sémantique des sources de données, nous pouvons distinguer deux types de médiation : médiation de schéma et médiation de contexte

3.2.1. Médiation de schéma

La médiation de schéma associe au médiateur un ensemble de connaissances qui localisent et intègrent les informations pertinentes à un contexte d'utilisation. Les requêtes sont exécutées sur ces connaissances (préétablies) qui indiquent au médiateur la localisation physique des données et les correspondances entre les données pour permettre leur combinaison et transformation afin de restituer des résultats homogènes et exploitables. La notion de contexte est ici implicite puisqu'il n'y a aucune information supplémentaire sur la source de données locale qui renseigne sur la sémantique des données. Le traitement des requêtes se fait alors de manière statique. L'aspect dynamique du médiateur se retrouve dans sa capacité de gérer la disparition des sources de données incomplètes et de s'adapter aux capacités d'interrogation différentes de chaque site. La médiation de schéma est une extension directe de l'approche fédérée avec une meilleure extensibilité.

3.2.2. Médiation de contexte

Cette approche repose sur une intégration dynamique des informations. Chaque application et chaque source de données locale doivent exprimer leur domaine sous forme de contexte d'utilisation des informations et ce pour permettre au médiateur, lors du traitement d'une requête, de comparer le contexte de l'application aux contextes des systèmes participant à la coopération. Le médiateur est guidé alors par des informations à caractère sémantique pour résoudre dynamiquement une requête, sans connaissance préalable des SI participants.

3.3. Architecture

L'approche de type médiation se présente en trois seulement : Niveau système d'information disposant des sources de données locales, niveau médiation composé du couple

Médiateur/Wrapper. ainsi que les connaissances nécessaires à l'interopération des sources de données et le niveau application qui correspond aux applications.

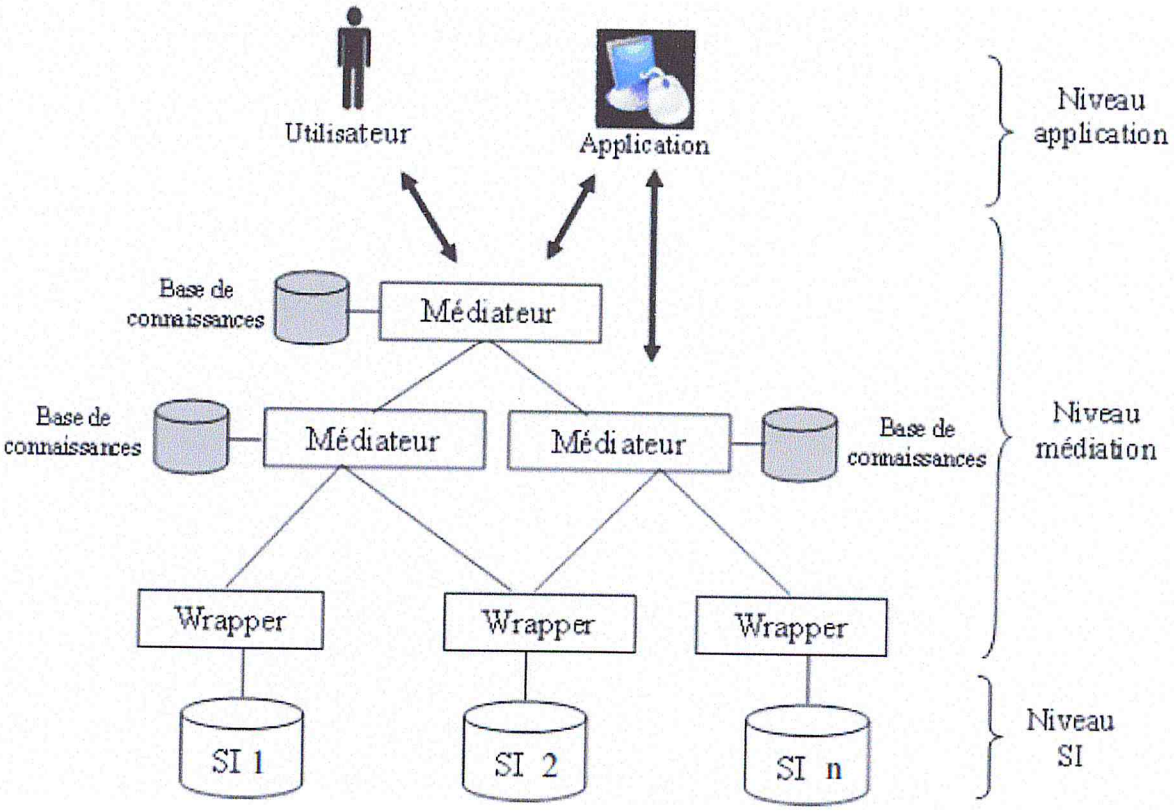


Figure 1 : Architecture D'un Médiateur ***[BA, 07]***

3.4. Classification des médiateurs

Les déférents médiateur existants se distinguent principalement par la façon dont établie la correspondance entre le schéma globale et les schémas des sources des données à intégrer.

On distingue deux manières distinctes :

3.4.1. L'approche Global As Views(GAV) : a été la première à être proposé pour l'intégration d'information et provient du monde de la base de données fédérée. Elle consiste à définir le schéma global en fonction des schémas des sources de

données à intégrer. Pour cela, les prédicats du schéma globale sont définis comme des vues sur les prédicats des schémas des sources à intégrer .cette approche suppose donc que les sources à intégrer soient connues a l'avance .comme les requêtes d'un utilisateur s'exprimes en termes des prédicats du schéma global. On obtient facilement une requête en termes du schéma des sources de données intégrées.

L'inconvénient de l'approche GAV est qu'elle est peu adaptée a l'ajout de nouvelles sources de données :cela peut nécessiter de mètre a jour la définition de l'ensemble des prédicats du schéma global

3.4.2. L'approche Local As Views(LAV) : est l'approche duel qui consiste a définir les schema des sources de données a intégrer en fonction du schéma globale.les avantage et inconvenants de cet approche sont inversés par rapport a l'approche GAV. L'approche LAV est très flexible par rapport a l'ajout (ou la suppression)de sources de données a intégrer :cela n'a aucun effet sur le schéma global, seules des vues doivent être ajoutées (ou supprimées).le prix a payer pour cette flexibilité et cette simplicité de mise a jour est la complexité de la construction des réponses a une requête dans un médiateur conçu selon l'approche LAV.la réécriture de requêtes en termes de vue est la seul possibilité pour obtenir des plans de requêtes exprimes en termes de vues(la réécritures).que l'on doit ensuite exécuter pour interroger les sources de données.

3.5. Panorama du médiateur existant

Nous structurons le panorama des principaux médiateurs en fonction du langage de représentation des connaissances utilisée pour exprimer le schéma globale. Pour chaque systèmes, nous indiquons s'ils suivent une approche GAV ou LAV

3.5.1. Systèmes fondés sur un schéma global a base de règles

Les systèmes les plus représentatifs de cette famille sont RAZOR,INTERNET SOFTBOT ,INFOMASTER et INFORMATIN MANIFOLD

Les system Razor et Internet Softbot sont les seuls qui utilisent le langage DataLog pur pour modéliser le schéma globale et pour exprimer les requête de l'utilisateurs

Information Manifold utilisent des extensions de DataLog. Infomaster permet d'exprimer, en plus des règles DataLog,des contraintes d'intégrité

Le système HERMES est un système de bases de données fédérées et peut être vu comme un médiateur suivant une approche GAV.il ne repose pas sur la description sémantique d'un domaine d'application ou du contenu de déférentes bases de données.son objectif est simplement de pouvoir combiner des requêtes posées a divers systèmes de gestion de bases de données.

Il repose sur un langage de requête uniforme permettent d'englober et de combiner dans une même requête l'appel a des bases de données relationnelles,orientee-objets,spatiales,d'image.son langage de requête est un langage a base de règles qui est sorte d'extension de Prolog.

3.5.2 Systèmes fondés sur un schéma global à base de classe

Les systèmes TSIMMIS est fondé sur un langage orienté-objet appelé OEM, pour le schéma global et les vues, ainsi que le langage OEM-QL pour les requêtes. Ce système suit une approche GAV

Les Systems SIMS et OBSERVER utilisent une logique de description pour décrire le schéma globale, les vue et les requêtes.

Ils relèvent d'une approche LAV puisqu'ils décrivent le contenu des sources a intégrer en fonction des descriptions de concepts du schéma global

Les MOMIS est fondé sur l'utilisation d'une logique de description tres riche pour décrire les schémas des sources de données a intégrer .l'approche suivie est de type GAV puisque le schéma globale est inféré a partir du schéma des sources.

3.5.3 Systèmes fondés sur un schéma global a base d'arbres

Avec l'avènement de XML, des médiateurs commencent a voir le jour au dessus de données semi-structures ayant le format de documents XML.la médiation sémantique dans C-WEB est fondée THESAURI et le deuxième est XYLEME. Les deux systèmes relèvent a la fois de l'approche GAV et LAV **[ROUSS, 02]**

4. Conclusion

Nous avons étudié dans ce chapitre les déférents approches et aspects des systèmes de médiations, mais notre but est la conception d'un system d'optimisation de requête dans un system de médiation pour cela nous aurons étudié les déférent approches des systèmes d'optimisation des requêtes dans un system de médiation dans le chapitre suivant.

1. Introduction

L'optimisation des requêtes constitue l'un des volets les plus importants dans le domaine des bases de données, d'où l'intérêt des chercheurs pour ce domaine, et ce depuis les années 70. Plusieurs travaux de recherche ont proposé diverses solutions s'inspirant, entre autre, de l'intelligence artificielle, la théorie des graphes, etc. Il y a eu également l'apparition de plusieurs articles établissant un état de l'art sur l'optimisation des requêtes.

2. Optimisation de requêtes

Le traitement de requêtes vise à traduire une requête utilisateur en une structure compréhensible par le médiateur, lui permettant de produire une réponse adéquate. Ce processus de traduction est complexe, et repose sur cette structure qui doit être conçue pour faciliter les différentes étapes du traitement de requêtes. Ces étapes décomposent la requête en un plan d'exécution compose de sous-requêtes mono-source et d'une requête de synthèse.

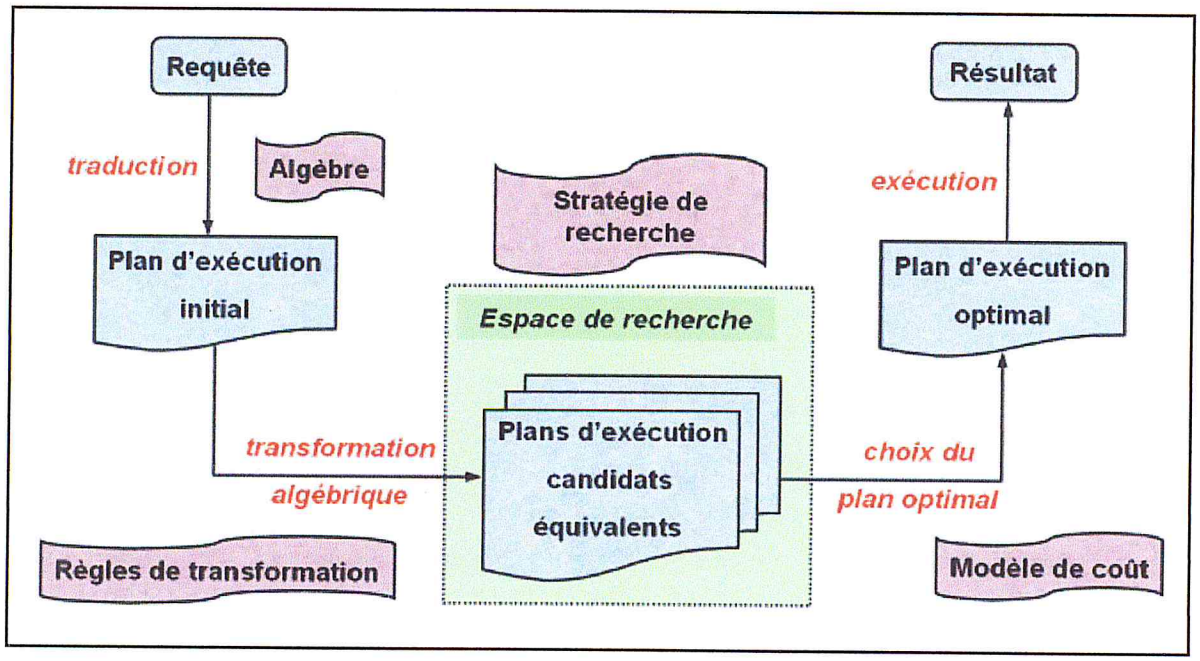


Figure 3 : Processus d'optimisation de requête.

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 proposé pour XML dans **[Widom, 99]**.

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. **[TIA, 11]**

3. Les Approches existants pour l optimisation des requêtes

Dans le présent chapitre, qui constitue un état de l'art sur Les Approches existants pour l'optimisation des requêtes, sera défini les approches suivant afin de comparaissent entre elles

- Annotation des Attributs pour l'Optimisation des Requêtes dans un Système de Médiation de Bases de Données

- Proposition d'un Cadre Générique d'Optimisation de Requêtes dans Les Environnements Hétérogènes Répartis.

- Optimisation Extensible dans un Médiateur de Données Semi Structurées

- Une Approche pour l'Optimisation des Requêtes Dirigée par la Réutilisation des Plans d'Exécution.

3.1 Annotation des Attributs pour l'Optimisation des Requêtes :

3.1.1. Introduction :

Cette approche ou bien méthode basée sur l'annotation des attributs de types numérique et énuméré. Chacun de ces attributs est annoté par son domaine de définition. Nous stockons ces informations dans le système de médiation pour permettre au médiateur d'anticiper les requêtes de façon à n'envoyer les requêtes qu'aux sources concernées. Cette approche utilise deux algorithmes :

-Le premier génère les sous domaines d'un attribut.

-Le second affecte les sources aux sous domaines générés.

3.1.2. Formalisation de l'approche proposée :

Formellement, un système d'intégration de données est un triplet I : <G, S, M>, où G représente le schéma global (défini sur un alphabet AG) qui modélise le schéma intégré, S est l'ensemble des schémas des sources (définis sur un alphabet AS) décrivant la structure des sources participantes au processus d'intégration, et M est une correspondance entre G et S qui établit la connexion entre les éléments du schéma global et ceux des sources. Pour interroger le système intégré, les requêtes sont exprimées en termes de constructions du schéma global. Cette approche consiste à annoter chaque attribut de type numérique ou énuméré par le domaine dans lequel l'attribut prend ses valeurs. Nous regroupons les domaines pour générer un ensemble de sous domaines auxquels sera assénées l'ensemble des sources. Ces annotations seront stockées au niveau du médiateur (Figure. 4).

Par la suite, une requête contenant un prédicat portant sur un attribut annoté ne sera envoyée à une source que si elle contient des instances satisfaisant le prédicat.

Dans cette formalisation nous allons tout d'abord définir les attributs numériques et énumérés avant de préciser l'objectif, les entrées et les sorties de l'approche proposée.

- un attribut est numérique si ses valeurs sont des nombres (Ex : entier, réel,)

- un attribut est énuméré s'il prend ses valeurs dans un ensemble fini

(Ex : 48 wilayas de l'Algérie).

- **Objectif :**

L'objectif est de réduire le temps d'exécution des requêtes globales posées au médiateur, spécifiquement, celles contenant des prédicats portant sur des attributs numériques ou énumérés.

- **Les entrées :**

Nous avons en entrée :

1. Un ensemble de requêtes globales fréquentes : QG

2. Un ensemble de sources S = {S1, S2, …, Sn}. Chaque source Si contient

un ensemble de relations locales ${}_{i}^{j}RL$ ${}_{i}S = \{{}_{i}^{i}RL, {}_{2}^{i}RL, \ldots\ldots, {}_{m}^{i}RL\}$

Chaque relation locale ${}_{i}^{j}RL$ contient un ensemble d'attribut ${}_{f}^{j,i}A$,

${}_{i}^{j}RL$(Cléi, ${}_{1}^{j,i}A$, ${}_{2}^{j,i}A$, …,), D(${}_{f}^{j,i}A$): est le domaine de définition de l'attribut ${}_{f}^{j,i}A$. ${}_{1}^{j,i}A$,
Si l'attribut est numérique, son domaine de définition est un ou plusieurs intervalles.

S'il est énuméré, son domaine de définition est un ensemble fini de valeurs.

- **Les sorties:**

Nous avons en sortie un schéma global *SG* qui peut être

défini comme un quadruplet : SG = <*RG*, *S*, *SD*, *SourceSD*>

1. *RG* et un ensemble de relations globales qui se définit comme suit :

RG = {*RG1*, *RG2*,…, *RGm*} Chaque relation globale *RGi* contient un

ensemble d'attributs ${}_{j}^{i}AG$ ${}_{i}RG$ (*Clé*, ${}_{1}^{i}AG$, ${}_{2}^{i}AG$,…,${}_{3}^{i}AG$) **[BAKH, 05]**

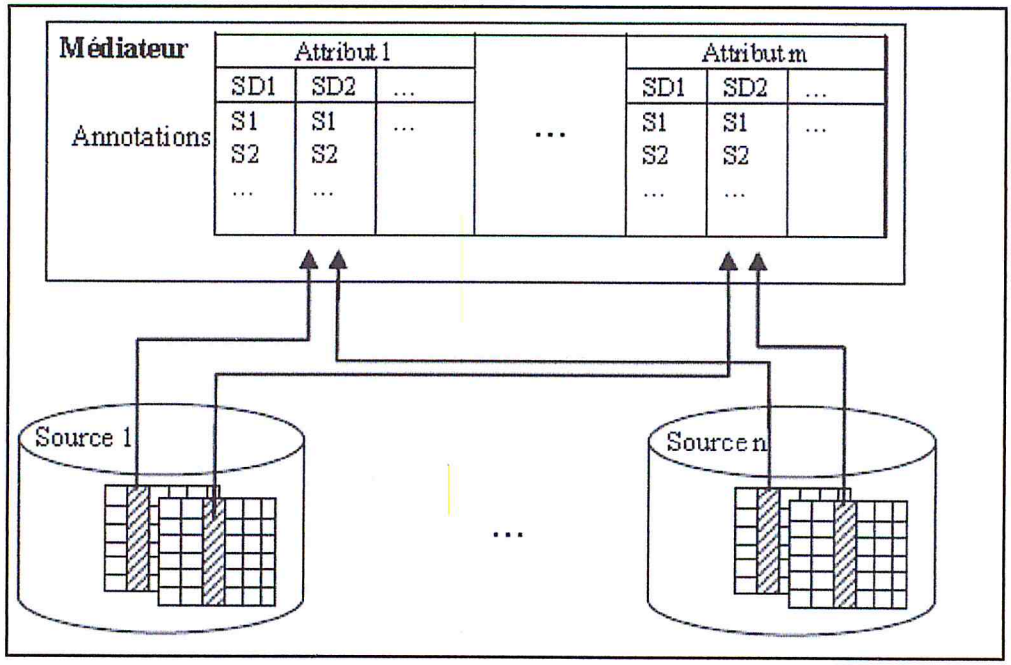


Figure 4 : Schéma d'un médiateur avec annotation des sources. **[BAKH, 05]**

3.1.3. Evaluation

L'objectif de cette approche est d'améliorer l'exécution des requêtes sur un système de médiation de sources de données hétérogènes et autonomes. La méthode recommande de sauvegarder les domaines de définition des attributs de type numérique et énuméré dans le médiateur. Cette sauvegarde permettra au médiateur de sélectionner les sources concernées par une requête et évite l'envoi de la requête à toutes les sources. L'algorithme de génération des sous domaines qui a pour objectif d'isoler les sources afin d'avoir un minimum de sources par sous domaine ; et l'algorithme d'affectation des sources aux sous domaines qui associe à chaque sous domaine la liste de sources possédant des instances dans ce sous domaine.

3.2.Cadre Générique d'Optimisation de Requêtes dans les Environnements Hétérogènes Répartis

En expliquant cette approche : nous allons tout d'abord discuter de l'espace de recherche en analysant les caractéristiques des règles de transformation équivalentes. Ensuite, nous allons présenter le cadre générique d'optimisation avec ses fonctions unitaires.

Pour montrer la généricité du cadre, nous allons illustre l'application de certains algorithmes de stratégie de recherche dans notre cadre générique d'optimisation, allant des

algorithmes naïfs comme Exhaustif aux algorithmes plus complexes comme Génétique. Enfin, nous expliquerons comment utiliser ce cadre générique pour construire des optimiseurs ayant différents critères d'optimisation.

- **Data Guide**

Les schémas de données peuvent être représentés avec la structure d'arbre sous l'aide de Data Guide. Le Data Guide est essentiellement une représentation concise de la structure de la collection qui est sous forme arborescente appelée arbre de la base de données, dans laquelle chaque chemin d'étiquette de la collection apparaît exactement une seule fois.
Le Data Guide est souvent plus petit que l'arbre de la base de données, ce qui permet de le maintenir en mémoire centrale lors de la recherche. Sans identification des nœuds de la collection, le Data Guide ne peut que renseigner sur l'existence d'un chemin d'étiquettes mais ne peut en aucun cas identifier de manière unique ce chemin dans la collection. Pour cela, un identifiant est attribué à chaque nœud du Data Guide. Le Data Guide et la table des annotations permettent ensemble d'indexer toute la structure de la base de données,

3.2.1. Espace de recherche

L'espace de recherche est une collection de solutions possibles à une problématique. Dans un processus d'optimisation, l'espace de recherche est dédié à trouver le plan d'exécution optimal pour une requête envoyée au système, chaque variable correspond donc à un plan d'exécution candidat Phisical Algebra Tree (PAT).

La taille de l'espace de recherche dépend des contraintes du problème.

Dans le contexte d'optimisation de requêtes, pour le PAT correspondant à la requête en question, il existe un ensemble de règles de transformation pouvant être appliquées sur le PAT pour générer d'autres PAT candidats.

Un opérateur du PAT peut être exécuté soit sur le médiateur soit sur la source, cela enrichit l'ensemble de règles possibles pour transformer le PAT, et ainsi agrandit l'espace de recherche.

- **Règles de transformation**

Une règle de transformation consiste à modifier un PAT afin de générer un autre PAT équivalent en termes du résultat de l'exécution de PAT.

Certaines règles de transformation sont atomiques, cela signifie que chaque règle est dédiée à modifier un seul opérateur du PAT. Ce sont les règles qui modifient les attributs d'un opérateur. Par exemple, une règle qui change l'algorithme d'exécution d'une jointure, de boucle imbriquée à tri par fusion ;

Certaines règles permettent de modifier l'ordre des opérateurs dans un PAT (déplacement, ajout, suppression, fusion ou scission des opérateurs). Par exemple, une règle qui fait descendre une restriction en dessous d'une jointure.

- **Poids d'une règle**

il est intéressant de mesurer cet effet d'une manière numérique et statistique. Nous introduisons ainsi une notion poids pour chaque règle.

Pour une règle de transformation équivalente, si les coûts d'exécution avant et après l'application de la règle sur un PAT sont respectivement *costbefore* et *costafter*, le poids de cette règle est la moyenne de *costafter* par rapport à *costbefore* des *n* PAT sur lesquels la règle peut s'appliquer

$$\frac{\sum_{i=1}^{n} \frac{cost_{after_i} - cost_{before_i}}{cost_{before_i}}}{n}$$

[TIA, 11]

Cette valeur moyenne est calculée en se basant sur les historiques de calcul de coût sur les PAT pendant les processus d'optimisation. Quand la valeur est positive, cela signifie que l'application de la règle améliore le PAT en termes de coût d'exécution.

3.2.2. Cadre générique d'optimisation

Le processus générique pour toutes les stratégies de recherche peut ainsi être résumé :

1. Annoter un LAT (Logical Algebra Tree) avec les informations liées à l'exécution et au coût ;
2. Estimer le coût d'exécution du PAT à partir des annotations ;
3. Trouver une règle de transformation pour générer un nouveau PAT;
4. Concevoir un algorithme de déroulement défini par la stratégie de recherche;
5. Définir les conditions d'arrêt pour que le processus d'optimisation se termine en considérant que le PAT optimal est retrouvé.

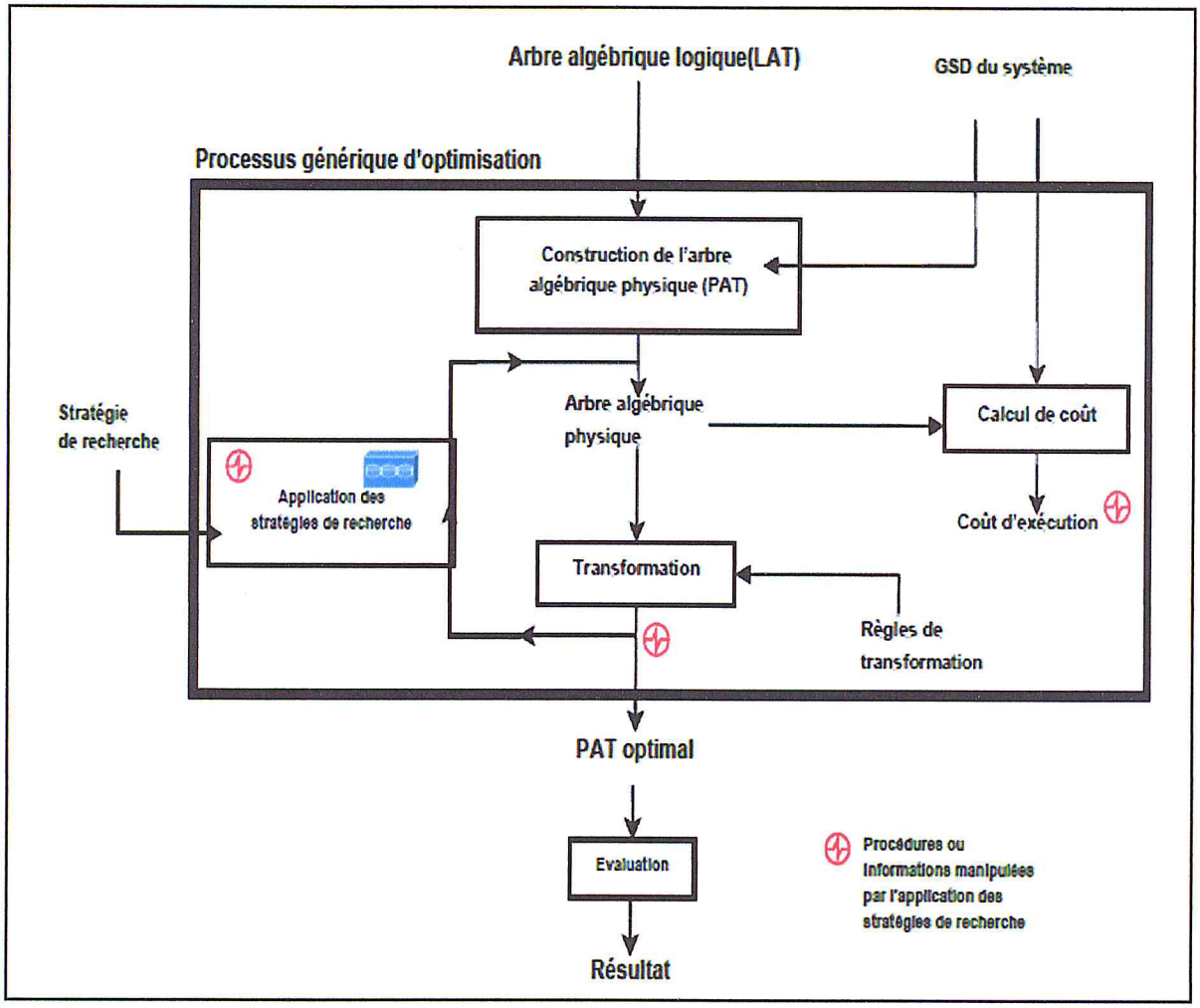


Figure 5 : Processus générique d'optimisation basé sur le coût. **[TIA, 11]**

3.2.3. Application sur différentes stratégies de recherche

Ce cadre générique d'optimisation est applicable sur six stratégies de recherche :

1. Algorithme exhaustif.
2. Algorithme incrémental.
3. Algorithme recuit simulé.
4. Algorithme programmation dynamique.
5. Algorithme génétique.
6. Algorithme colonies de fourmis.

3.2.4. Application sur différents critères d'optimisation

Pour un optimiseur de requêtes, il est possible d'avoir d'autres objectifs d'optimisation que la diminution du temps d'exécution (mémoire, CPU, réseaux, etc.).Si le développeur de l'optimiseur veut changer le critère d'optimisation, il suffit de modifier la fonction getRuleWeight, et l'implémentation de toutes les autres fonctions restent valides pour le nouveau critère d'optimisation. **[TIA, 11]**

3.2.5. Evaluation

Ce cadre générique d'optimisation a les avantages suivants :

Réduction du temps de développement pour différentes stratégies de recherche

Modèle de coût générique

Différents critères d'optimisation

Indépendance du langage pivot du médiateur.

3.3. Optimisation Extensible dans un Médiateur de Données Semi Structurées

Cette approche d'optimisation de requêtes basée sur le modèle TGV.les *TGV*s (Tree Graph View) permettent de modéliser le langage de requêtes XQuery grâce a un ensemble de définitions riches étendant les principes des *Tree Patterns*.

Pour cela, il est nécessaire de définir une algèbre d'évaluation correspondant a cet représentation de requêtes, un modèle de cout, des règles de transformation et une stratégie de génération de TGV équivalents pour nous permettre d'obtenir une représentation optimale.

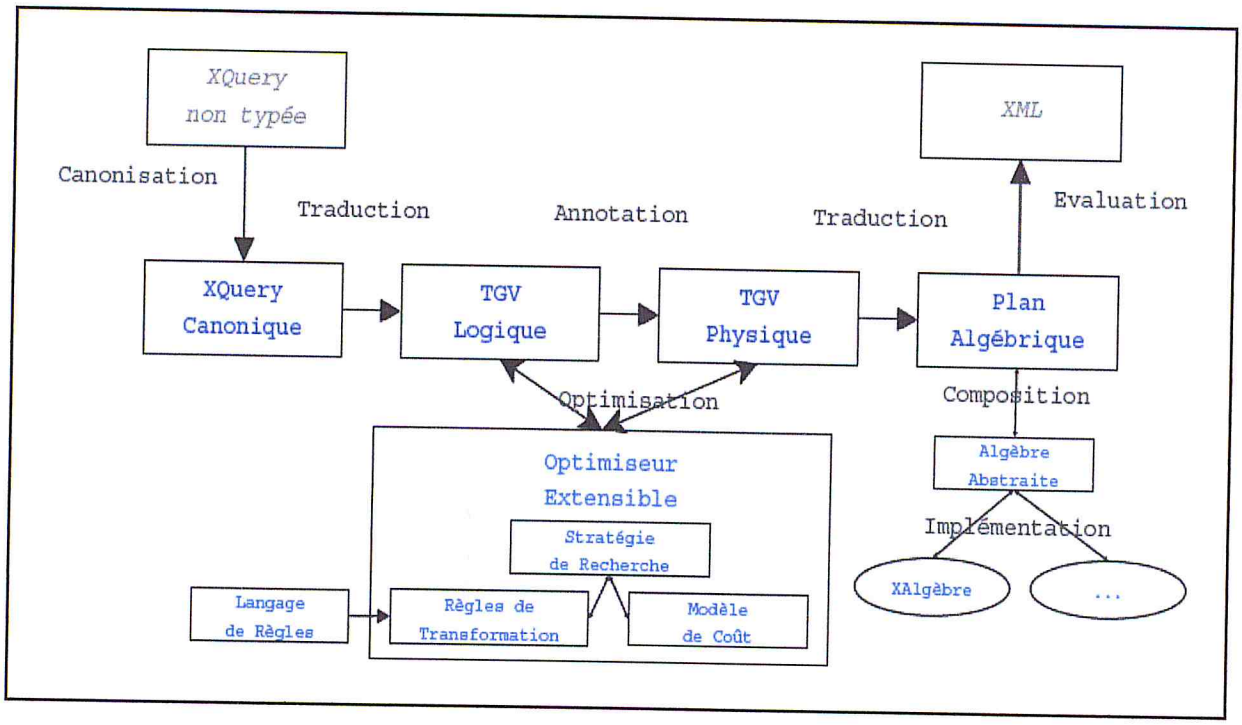


Figure 6 : Cycle de traitement d'une requête XQuery. **[TRA, 06]**

3.3.1. Algèbre Abstraite pour TGV

L'algèbre abstraite permettant de définir des expressions algébriques d'évaluation des TGV. Cette algèbre abstraite définie les primitives des opérations de traitement sur les TGV. Une simple traduction de ces opérations permet l'implantation des TGV dans les systèmes de gestion de base de données pour évaluer des documents XML.

3.3.2. Annotation des TGV

Les annotations nous permettent de définir, a un niveau de granularité quelconque, une annotation spécifique que nous pouvons définir aisément. Chaque information est annotée sur un ensemble d'éléments du TGV. Cette annotation est représentée par une couleur unique qui détermine, sur la représentation, l'ensemble des éléments associe. Il est alors aisé d'identifier les localisations des sources et les algorithmes choisis. Il est aussi possible d'ajouter des formules de cout. Cette base d'annotation permet de manipuler les TGV a un niveau supérieur dans le cadre de l'optimisation. Un langage de définition de règles basé sur les TGV et les annotations transforme les TGV pour obtenir un plan optimal. Un *TGV logique* annote est alors appelé un *TGV Physique*.

3.3.3. Règles de transformation

Les règles de transformations peuvent être classées dans trois groupes de transformations : équivalentes logiques, équivalentes physiques et utilisateurs. Les transformations équivalentes sont des transformations qui modifient un TGV sans modifier le résultat de son évaluation, une transformation logique est une transformation qui n'utilise que les connaissances internes aux TGV, alors qu'une transformation physique utilise les connaissances des TGV et de leurs annotations. Enfin, les transformations utilisateurs sont des transformations dont la base de connaissances ne peut être annotée, de plus, les résultats de ces transformations ne sont pas forcement équivalents mais répondent aux pré-requis du contexte définis lors de la conception de l'optimiseur.

3.3.4. Stratégie de recherche

La stratégie de recherche définie dans l'optimiseur s'appuie sur un modèle de cout défini par calibrage des sources avec raffinage sur historique. Cette stratégie incrémentale définie un cadre d'optimisation pour effectuer des tests de performances.

L'annotation des TGV permet de donner une estimation du coût d'évaluation sur des ensembles d'éléments présents dans le TGV. Le parcours de l'espace de recherche des plans est orienté grâce a l'attribution d'un coefficient d'amélioration des règles de transformations. Ce coefficient est orienté lui-même par calibrage de celles-ci grâce au modèle de coût avec raffinage par historique. Ces coefficients sont influencés par un facteur déterminé grâce a la catégorisation des règles de transformation. **[TRA, 06]**

3.3.5. Evaluation

Une algèbre abstraite est proposée pour définir une évaluation des TGV de manière intuitive indépendante d'une algèbre physique.

Une base d'annotation générique est associée au modèle pour représenter n'importe quelle information que nous souhaitons intégrer à la représentation

L'optimiseur modifie un TGV à l'aide de règles de transformation. Ces règles de transformation peuvent être intégrées dans le système grâce a un langage de définition de règles de transformation.

Une stratégie de recherche définie, elle permet donc de réduire l'espace de recherche. Pour cela, un modèle de coût est nécessaire pour déterminer le coût d'évaluation de chaque TGV, celui-ci est défini par calibrage des sources et raffinage par historique.

3.4. Optimisation des requêtes basée sur la réutilisation des plans d'exécution

3.4.1. Introduction

Cette solution est particulièrement intéressante car elle part d'une constatation simple: deux requêtes peuvent manipuler des tables différentes et utiliser des opérations de jointure, sélection et projection différemment; mais, au final, l'optimiseur leur générera le même plan d'exécution avec des entrées différentes propres à chacune d'elles (tables, attributs, etc.). Il est à signaler que ce type d'optimisation constitue une valeur ajoutée à l'optimiseur de requêtes permettant d'accroitre ces performances, et non un modèle d'optimisation à part entière.

Les méthodes basées sur la réutilisation des plans d'exécution, ont donc pour but de décharger l'optimiseur de requêtes, ce qui accélérera le processus d'optimisation, et par conséquent le temps de réponse. La diminution de la charge sur l'optimiseur lui permet également de travailler à son plus haut niveau, et d'offrir ainsi des plans de qualité, ce qui se répercutera sur les plans futurs générés par similarité.

3.4.2. PLASTIC (PLAn Selection through Incremental Clustering)

Cette première méthode a été proposée par des chercheurs indiens en 2002, de laquelle a découlé un outil appelé PLASTIC (PLAn Selection through Incremental Clustering) qui, comme son nom l'indique, procède à une classification des plans d'exécution afin de les réutiliser pour de futures optimisations.

Dans ce qui suit, nous procédons à une présentation détaillée de l'outil PLASTIC, ainsi que les améliorations qui lui ont été apportées en 2004, ce qui a provoqué l'apparition d'une deuxième version appelée PLASTIC2.

3.4.2.1. Description générale de PLASTIC1

L'outil PLASTIC a pour but d'amortir le coût d'optimisation par une technique de réutilisation de plans, dont le principe est d'établir des similarités entre les requêtes en les représentant sous forme de vecteurs contenant des attributs significatifs, tels que les tailles des tables participant à la requête, leur nombre, etc.

Les composants de ces vecteurs seront utilisés dans une fonction de distance, grâce à laquelle les requêtes jugées similaires (donc ayant le même plan d'exécution) seront regroupées dans des clusters construits de façon incrémentale. Dès qu'une nouvelle requête arrive, elle sera assignée au cluster le plus adéquat et, de ce fait, s'exécutera sous le plan représentant ce cluster.

3.4.2.2. Quelques concepts généraux

Query Template : Ce sont les requêtes où l'ensemble ou une partie des constantes sont remplacées par des variables, appelées variables liées (*bind variables*)

Query Template Space : C'est l'ensemble des requêtes pouvant être instanciées de *query template* en assignant différentes valeurs aux *bind variables.*

Plan Diagram for a Query Template: C'est l'énumération des plans choisis par l'optimiseur dans tous les points du *Query Template Space,* sur lesquelles sont appliqués des prédicats de sélection. Il servira dans cette méthode à la génération des clusters de départ (ensemble d'apprentissage).

Plan Template : C'est un plan d'exécution où tous les opérateurs sont retenus, mais les valeurs de leurs entrées telles que: les noms des attributs, etc. sont remplacées par des variables.

3.4.2.3. Architecture de PLASTIC1

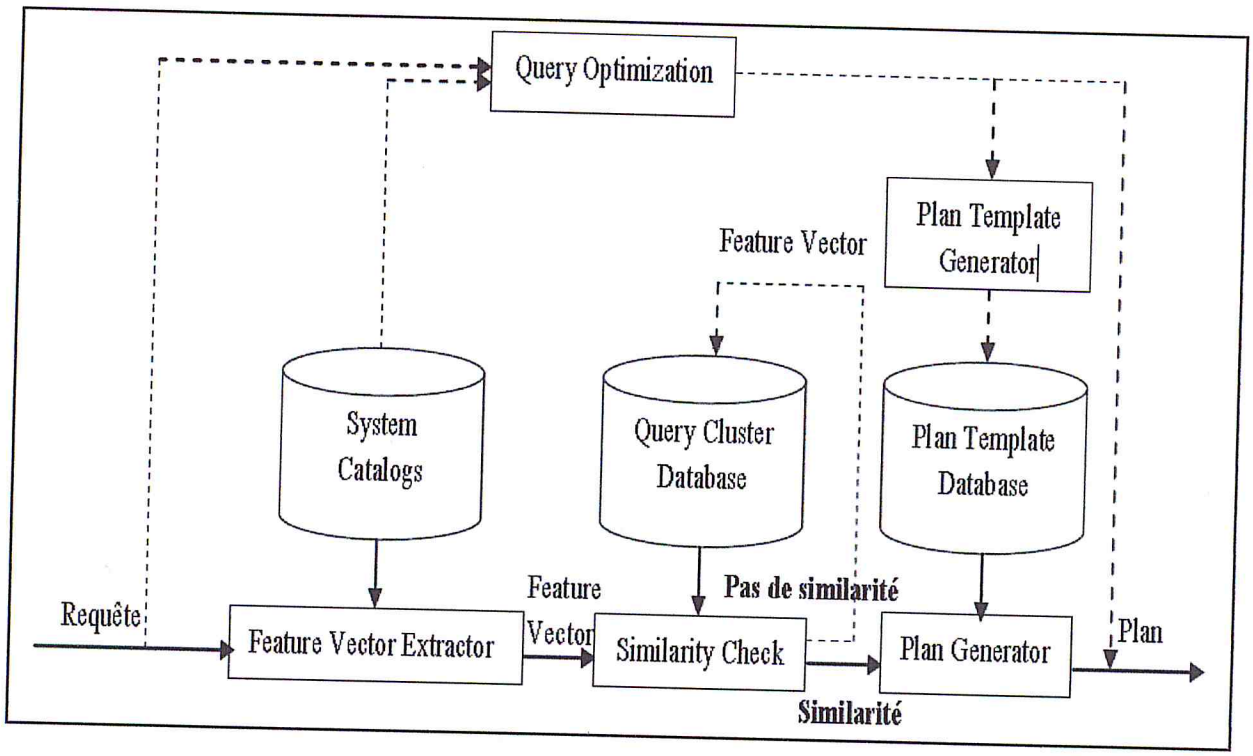


Figure 7 : Architecture de PLASTIC **[ANT , 02]**

L'architecture globale de PLASTIC est représentée sur la figure 7 :

Les lignes continues montrent le fonctionnement de l'outil dans le cas où une similarité est trouvée, tandis que les tirets montrent le cas inverse, à savoir, l'absence de similarité.

Une fois que la requête est envoyée au système, elle passe par le *Feature Vector Extractor* qui, lui-même, accède au *system catalogs* pour obtenir l'information requise afin de produire le vecteur de caractéristiques de cette requête. Ce vecteur est ensuite envoyé au module *Similarity Check* qui, comme son nom l'indique, décide si le vecteur de cette requête est suffisamment similaire au vecteur de requête de l'un des clusters déjà stockés dans la *Query Cluster Database.*

S'il existe une similarité, le *plan template* du cluster sélectionné est récupéré du module *Plan Template Database*, et sera ensuite utilisé pour exécuter la requête. Si par contre, aucune similarité n'est trouvée, l'optimiseur classique de requêtes est appelé. Le plan qu'il génèrera sera utilisé pour exécuter la requête. Ce plan sera ensuite envoyé au module *Plan Template Generator* qui le convertira en une représentation abstraite et le stockera dans le

Plan Template Database. Parallèlement, le vecteur de caractéristiques de la requête sera stocké dans la *Query Cluster Database*, et ainsi un nouveau cluster sera ajouté. **[ANT , 02]**

3.4.3. Evaluation

Dans cette approche on a présenté la méthode récente d'optimisation des requêtes, basée sur la réutilisation des plans d'exécution. Cette méthode a pour avantage de réduire le temps d'optimisation, en allégeant la charge sur l'optimiseur, lui offrant ainsi la possibilité de générer des plans de qualité, qui seront réutilisés par la suite.

4. Conclusion

Dans ce chapitre nous avons résumé quelque approche des Systems d'optimisation des requêtes.

Nous avons aussi évalué chaque méthode d'optimisation existante, dans le but de nous en inspirer pour la construction de notre approche.

Cette partie, nous a 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 la prochaine partie.

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. 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 médiateur.

2. Le Cycle de vie d'un logiciel et le langage de modélisation UML

2.1. Le cycle de vie d'un logiciel

2.1.1. Définition

Le cycle de vie d'un logiciel (en anglais software life cycle), désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition. L'objectif d'un tel découpage est de permettre de définir des jalons intermédiaires permettant la validation du développement logiciel, c'est-à-dire la conformité du logiciel avec les besoins exprimés, et la vérification du processus de développement, c'est-à-dire l'adéquation des méthodes mises en œuvre .

2.1.2. Modèle de cycle de vie en cascade

Le modèle de cycle de vie en cascade a été mis au point dès 1966, puis formalisé aux alentours de 1970.

Dans ce modèle le principe est très simple : chaque phase se termine à une date précise par la production de certains documents ou logiciels. Les résultats sont définis sur la base des interactions entre étapes, ils sont soumis à une revue approfondie et on ne passe à la phase suivante que s'ils sont jugés satisfaisants **[LAU, 06].**

Le cycle de vie du logiciel comprend généralement au minimum les étapes suivantes :

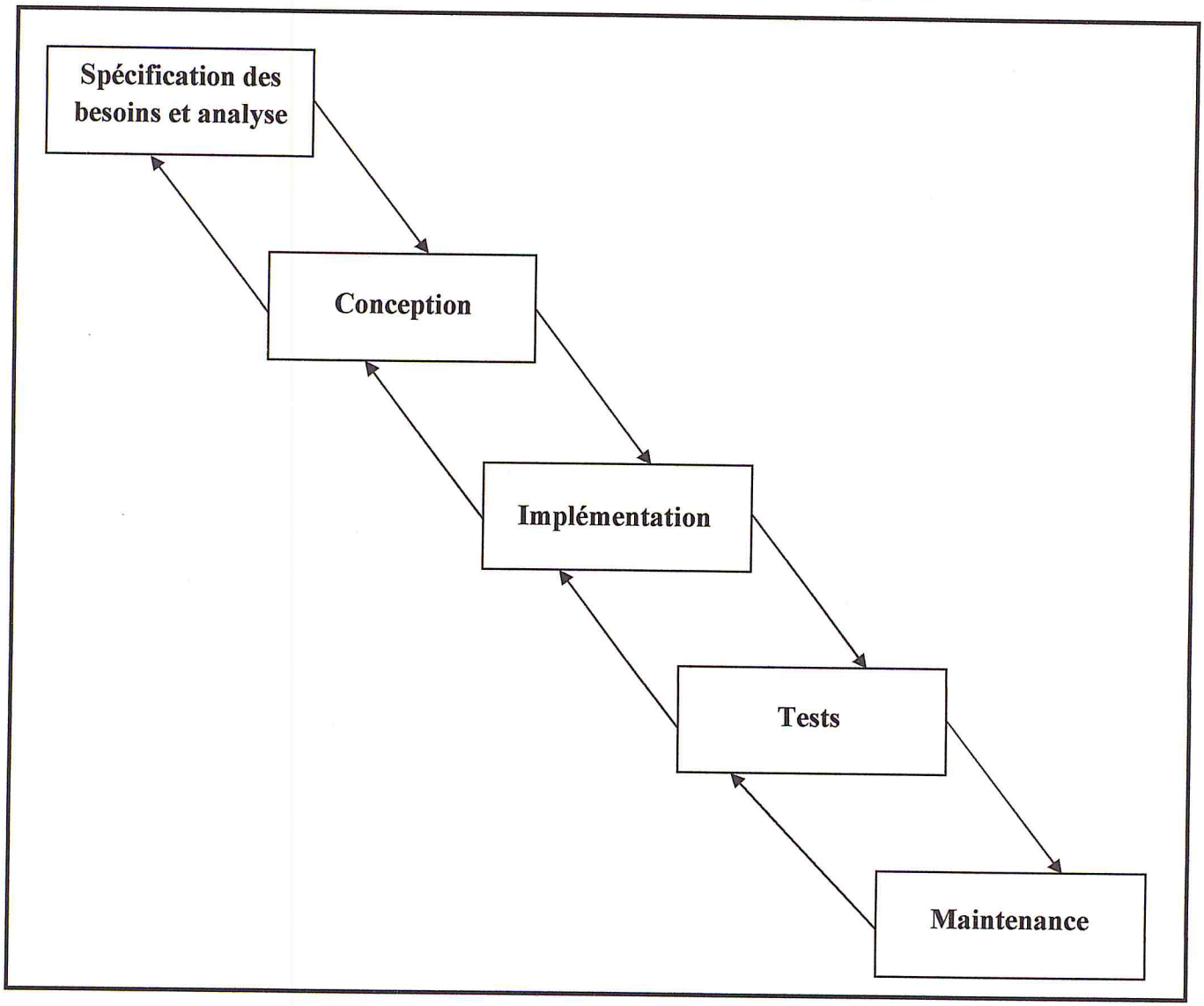


Figure 8 : Le cycle de vie de modèle en cascade.

2.1.3. Les activités

a) Spécification des besoins et analyse

- **Spécification des besoins**

L'expression des besoins comme son nom l'indique, permet de définir les différents besoins:

- inventorier les **besoins principaux** et fournir une liste de leurs fonctions
- recenser les **besoins fonctionnels** (du point de vue de l'utilisateur) qui conduisent à l'élaboration des modèles de cas d'utilisation
- appréhender les **besoins non fonctionnels** (technique) et livrer une liste des exigences.

Le modèle de cas d'utilisation présente le système du point de vue de l'utilisateur et représente sous forme de cas d'utilisation et d'acteur, les besoins du client.

- **Analyse**

L'objectif de l'analyse est d'accéder à une compréhension des besoins et des exigences du client. Il s'agit de livrer des spécifications pour permettre de choisir la conception de la solution.

Un modèle d'analyse livre une spécification complète des besoins issus des cas d'utilisation et les structure sous une forme qui facilite la compréhension (scénarios), la préparation (définition de l'architecture), la modification et la maintenance du futur système.

Il s'écrit dans le langage des développeurs et peut être considéré comme une première ébauche du modèle de conception.

b) Conception

La conception permet d'acquérir une compréhension approfondie des contraintes liées au langage de programmation, à l'utilisation des composants et au système d'exploitation.

Elle détermine les principales interfaces et les transcrit à l'aide d'une notation commune.

Elle constitue un point de départ à l'implémentation :

- elle décompose le travail d'implémentation en sous-système.
- elle créée une abstraction transparente de l'implémentation.

c) Implémentation

L'implémentation est le résultat de la conception pour implémenter le système sous formes de composants, c'est-à-dire, de code source, de scripts, de binaires, d'exécutables et d'autres éléments du même type.

Les objectifs principaux de l'implémentation sont de planifier les intégrations des composants pour chaque itération, et de produire les classes et les sous-systèmes sous formes de codes sources.

d) Test

Les tests permettent de vérifier des résultats de l'implémentation en testant la construction. Pour mener à bien ces tests, il faut les planifier pour chaque itération, les implémenter en créant des cas de tests, effectuer ces tests et prendre en compte le résultat de chacun.

e) Maintenance

Elle comprend toutes les actions correctives (maintenance corrective) et évolutives (maintenance évolutive) sur le logiciel.

2.2. Le langage UML

Définition

UML ou Unified Modeling Language, est un langage de modélisation qui est né au milieu des années 90 de la fusion de trois méthodes objets: OMT , BOOCH1 (GRADY BOOCH le concepteur de la méthode) et OOSE . L'idée de cette fusion est partie du constat qu'à l'époque ils existaient plusieurs méthodes objets liées par un consensus autour d'idées communes: objets, classes, sous-systèmes etc.

C'est à partir de 1997 que l'OMG2 (Object Management Group) qui standardise les technologies de l'objet, s'est attachée à la définition d'un langage commun unique, utilisable par toutes les méthodes objets dans toutes les phases du cycle de vie et compatible avec les techniques de réalisation du moment. D'où la naissance d'UML. UML offre des éléments pour décrire les différents aspects d'un système: les diagrammes.

Ses points forts sont les suivants :

- C'est un langage formel et normalisé : il permet un gain de précision, de stabilité et encourage l'utilisation d'outils.
- C'est un support de communication performant : il cadre l'analyse et facilite la compréhension de représentations abstraites complexes. De plus, son caractère polyvalent et sa souplesse en font un langage universel .

UML 2.0 comporte ainsi treize types de diagrammes représentant autant de vues distinctes pour représenter des concepts particuliers du système d'information. Ils se répartissent en trois grands groupes : **[LAU, 06]**

- **Diagrammes structurels ou diagrammes statiques (UML Structure)**
  - diagramme de classes (Class diagram)
  - diagramme d'objets (Object diagram)
  - diagramme de composants (Component diagram)
  - diagramme de déploiement (Deployment diagram)
  - diagramme de paquetages (Package diagram)

- diagramme de structures composites (Composite structure diagram)

- **Diagrammes comportementaux ou diagrammes dynamiques (UML Behavior)**
  - diagramme de cas d'utilisation (Use case diagram)
  - diagramme d'activités (Activity diagram)
  - diagramme d'états-transitions (State machine diagram)
- **Diagrammes d'interaction (Interaction diagram)**
  - diagramme de séquence (Sequence diagram)
  - diagramme de communication (Communication diagram)
  - diagramme global d'interaction (Interaction overview diagram)
  - diagramme de temps (Timing diagram)

3. Spécification des besoins

3.1. Recueil des 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.
- l'utilisateur lance sa requête dans un langage commun.
- Le médiateur est chargé de traiter la requête
- Le médiateur est chargé de traiter les conflits sémantiques entre les données
- 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.

3.1.1 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 de notre système et décrit la mission de chacun.

Acteurs	Mission
Utilisateur	L'utilisateur s'occupe du 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. • L'administrateur peut jouer le rôle d'un utilisateur. • Consulter l'ontologie. • Générer Data Guides. • Construire la table de correspondance.
Médiateur	Le médiateur est l'un des acteurs principaux du système, il est chargé du traitement de la requête et de reproduire le résultat.
Adaptateur	L'adaptateur est l'acteur qui s'occupe de la traduction langage à un autre.

Tableau 1 : les acteurs du système.

3.1.2. Modélisation du contexte

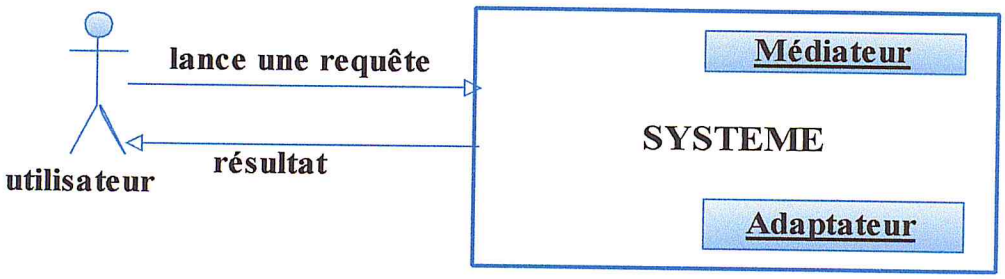


Figure 9 : diagramme de contexte du système.

3.1.3 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.

3.1.3.1 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 (use case) 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

Tableau 2 : Modèle de représentation des descriptions détaillées des cas d'utilisations.

c. Le cas d'utilisation Génération table de correspondance

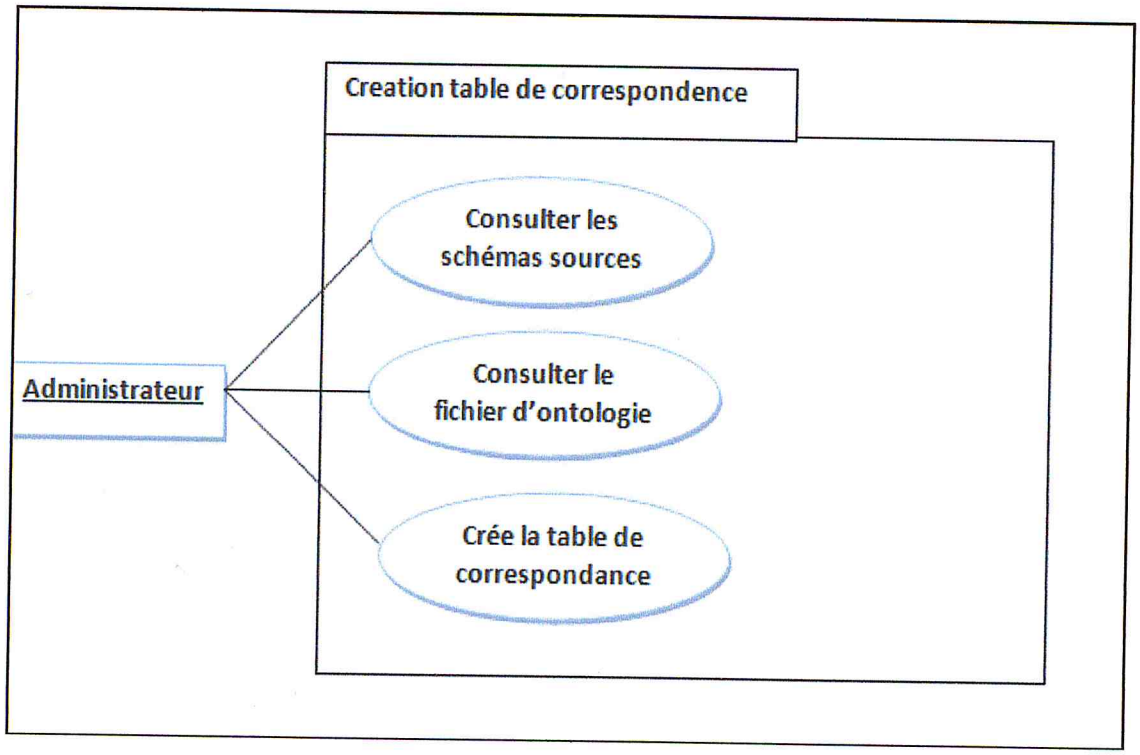


Figure 11 : Diagramme de cas d'utilisation Génération table de correspondance

c.1 Le cas d'utilisation Consulter les schémas sources

Cas d'utilisation	Consulter les schémas sources.
Acteur	Administrateur.
Objectifs	déterminer l'ensemble des concepts globaux à partir des concepts locaux.
Description	Le médiateur doit tirer l'ensemble des éléments constituant les schémas locaux.
Pré condition	existence des schémas locaux.
Post condition	un ensemble d'élément est tiré.
Exception	Aucune.

Tableau 5 : Description du cas d'utilisation « Consulter les schémas sources»

c.2 Le cas d'utilisation Consulter le fichier d'ontologie

Cas d'utilisation	Consulter le fichier d'ontologie.
Acteur	Administrateur.
Objectifs	Consulter un ensemble de données.
Description	Le médiateur doit tirer l'ensemble des concepts globaux à partir des concepts locaux.
Pré condition	fichier d'ontologie non vide. .
Post condition	concept global pour chaque ensemble de concepts locaux.
Exception	Aucune.

Tableau 6 : Description du cas d'utilisation « Consulter le fichier d'ontologie»

c. 3 Le cas d'utilisation Création table de correspondance

Cas d'utilisation	Création table de correspondance.
Acteur	Administrateur.
Objectifs	Crée table de correspondance.
Description	Ce cas d'utilisation commence lorsque l'utilisateur de système accède à une interface dans laquelle il pose sa requête.
Pré condition	consulter les schémas locaux, fichier d'ontologie.
Post condition	Création de table de correspondance.
Exception	Aucune.

Tableau 7 : Description du cas d'utilisation « Création table de correspondance»

d. Le cas d'utilisation génération le data guides

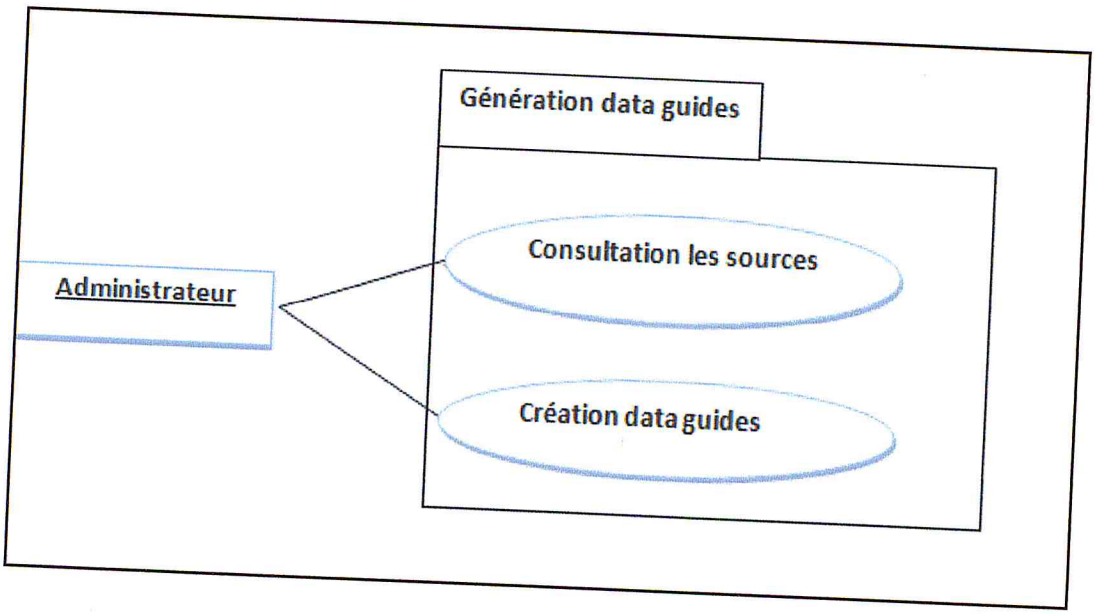


Figure 12 : Diagramme de cas d'utilisation Génération le data guides

d.1 Le cas d'utilisation Consultation les sources

Cas d'utilisation	Consultation les sources.
Acteur	Administrateur.
Objectifs	déterminer l'ensemble des tables, des attributs.
Description	ce cas d'utilisation commence lorsque le médiateur généré le data guides. Le médiateur doit tirer l'ensemble des éléments constituant les schémas locaux.
Pré condition	existence des schémas locaux.
Post condition	data guides.
Exception	Aucune

Tableau 8 : Description du cas d'utilisation « Consultation les sources»

d.2 Le cas d'utilisation Création data guides

Cas d'utilisation	Création data guides.
Acteur	Administrateur.
Objectifs	localiser les sources pertinentes.
Description	après la détermination de l'ensemble d'attribut, tables, le médiateur insérer dans le data guides pour chaque source de données les tables, pour chaque table les attributs.
Pré condition	Data guides.
Post condition	L'implémentation data guides.
Exception	aucune.

Tableau 9 : Description du cas d'utilisation « Création data guides»

e. Diagramme de cas d'utilisation Traitements des requêtes

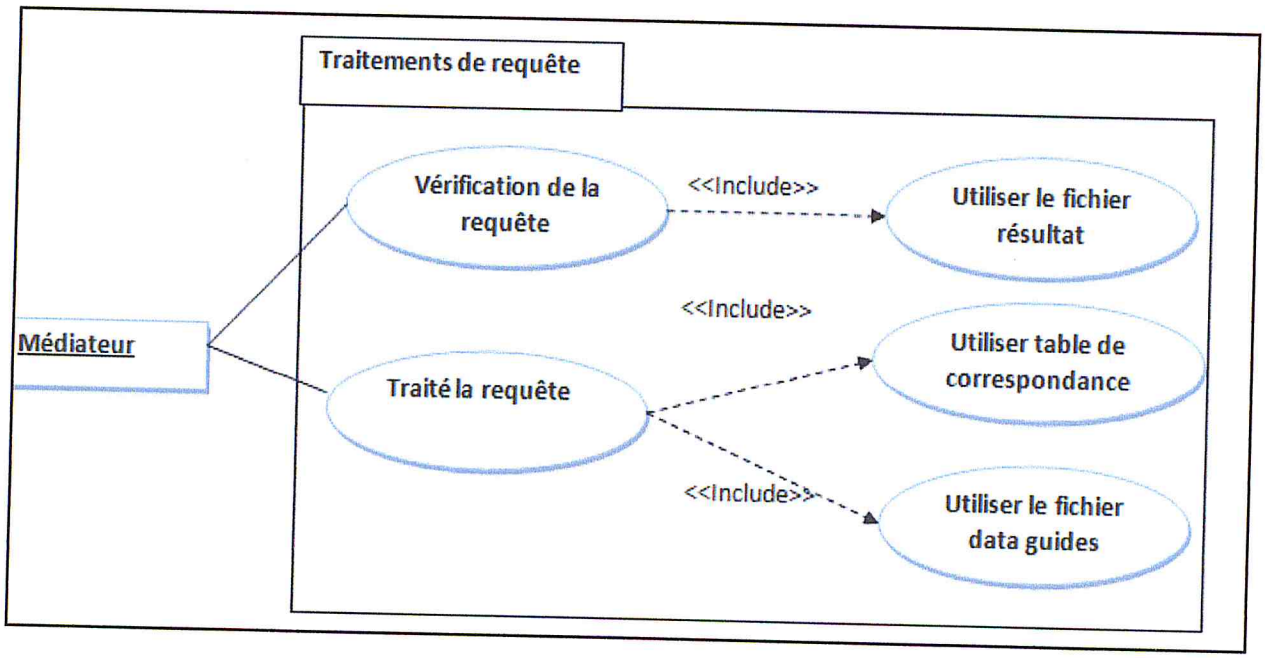


Figure 13 : Diagramme de cas d'utilisation Traitement de requête

e.1 Le cas d'utilisation vérification de la requête

Cas d'utilisation	vérification de la requête.
Acteur	Médiateur.
Objectifs	obtenir des résultats.
Description	Si cette requête existe dans le fichier résultat (est déjà saisi), le Médiateur afficher le résultat.
Pré condition	lancement d'une requête.
Post condition	résultat.
Exception	aucune.

Tableau 10 : Description du cas d'utilisation «vérification de la requête»

e.2 Le cas d'utilisation Traité la requête

Cas d'utilisation	Traité la requête.
Acteur	Médiateur.
Objectifs	Obtenir des requêtes adaptées aux sources.
Description	Analyser l'ensemble des attributs constituant la requête, remplacer les attributs générales par leurs définitions, obtenir des sous requêtes adaptés aux sources approprie.
Pré condition	Lancement d'une requête.
Post condition	Résultat.
Exception	Aucune.

Tableau 11 : Description du cas d'utilisation «Traité la requête»

f. Diagramme de cas d'utilisation Fusion des résultats

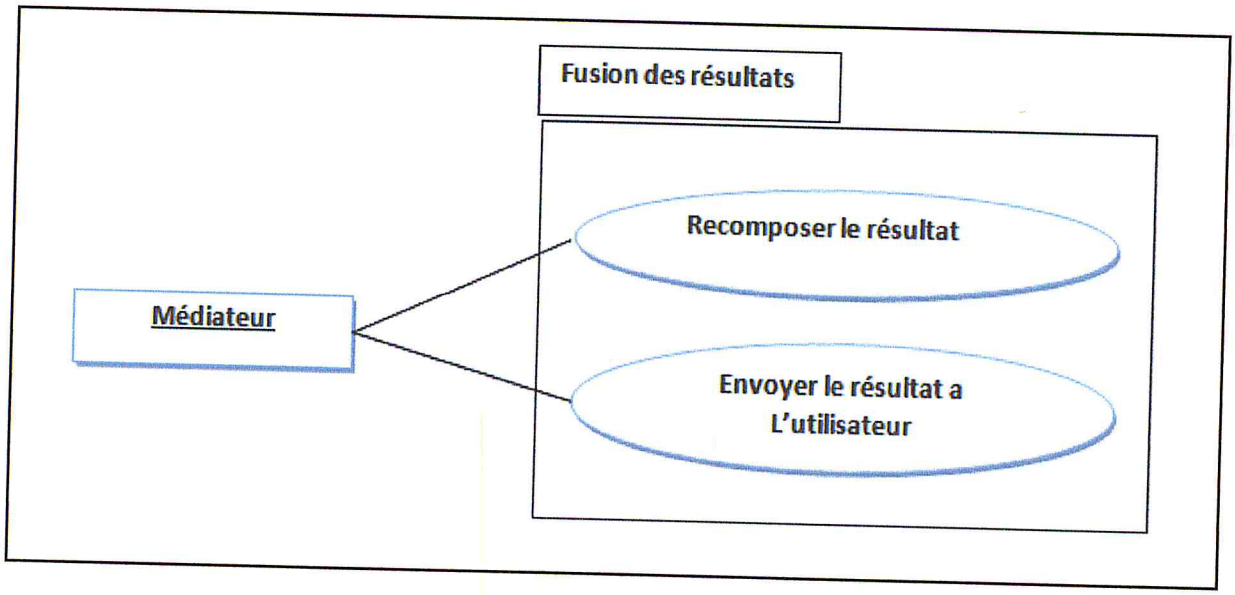


Figure 14 : Diagramme de cas d'utilisation Fusion des résultats

f.1 Le cas d'utilisation Recomposer le résultat

Cas d'utilisation	Recomposer le résultat.
Acteur	Médiateur.
Objectifs	Recomposer le résultat.
Description	ce cas d'utilisation commence après la réception des sous réponses, le médiateur fait la jointure des résultats pour avoir une seule réponse.
Pré condition	- le nombre de sous résultat égale le nombre de sous requête - Existence de sous résultat.
Post condition	Résultat.
Exception	aucune.

Tableau 12 : Description du cas d'utilisation «Recomposer le résultat»

f.2 Le cas d'utilisation envoyé le résultat à l'utilisateur

Cas d'utilisation	envoyer le résultat à l'utilisateur.
Acteur	Médiateur.
Objectifs	fournir un résultat.
Description	ce cas d'utilisation commence après la recomposition des sous résultats de chaque adaptateur.
Pré condition	existence résultat.
Post condition	résultat.
Exception	aucune.

Tableau 13 : Description du cas d'utilisation «envoyer le résultat à l'utilisateur»

g. Diagramme de cas d'utilisation Traduction

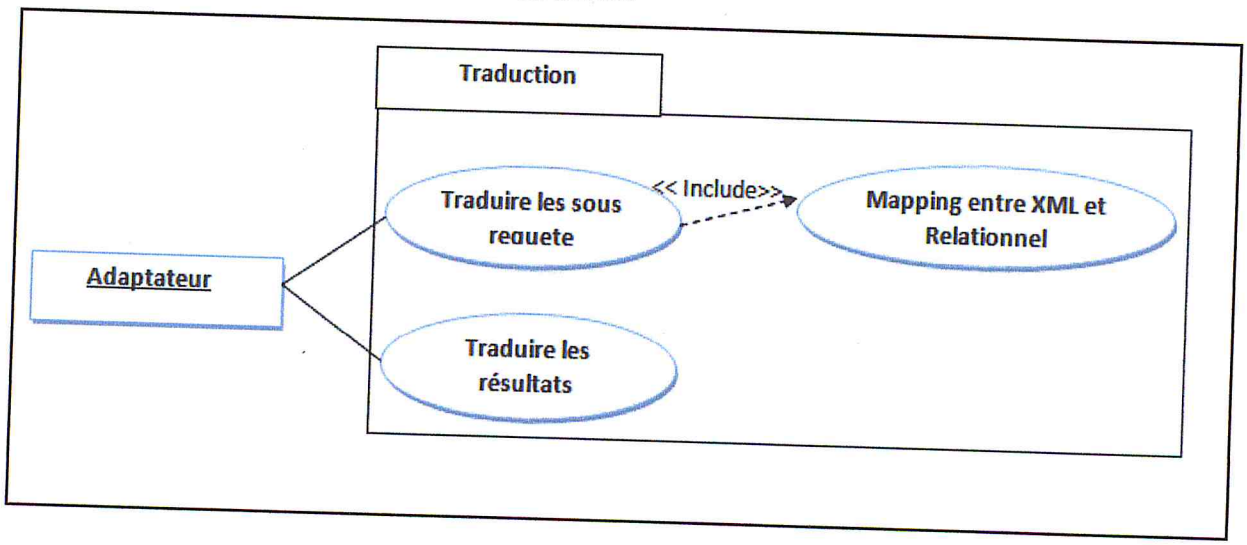


Figure 15 : cas d'utilisation Traduction

g.1 Le cas d'utilisation Traduire les sous requête

Cas d'utilisation	Traduire les sous requêtes.
Acteur	Adaptateur.
Objectifs	Traduire les sous requêtes.
Description	Traduire les sous requêtes du langage global aux langages locaux adaptés aux sources.
Pré condition	sous requêtes en langage global.
Post condition	sous requêtes en langage local.
Exception	aucune.

Tableau 14 : Description du cas d'utilisation «Traduire les sous requêtes»

g.2 Le cas d'utilisation Traduire les résultats

Cas d'utilisation	traduire les résultats.
Acteur	Adaptateur.
Objectifs	traduire les résultats.
Description	Traduire les résultats du langage local au langage global adapté au médiateur.
Pré condition	sous résultat en langage local.
Post condition	sous résultat en langage local.
Exception	aucune.

Tableau 15 : Description du cas d'utilisation «Traduire les résultats»

3.2. Analyse

L'analyse permet de lister les résultats attendus, en terme de fonctionnalités, de performance, de robustesse, 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]**

Qu'est qu'un scénario ?

Un scénario représente un ensemble ordonné de messages échangés par des objets. On parle ici d'objet au sens large : instance de classe d'analyse ou instance d'acteur.

Les échanges de messages entre objet peuvent être représentés en UML dans une sorte de diagramme complémentaire appelé *diagramme de séquence*.

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.

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

- **Scenario de l'authentification (cas normal)**
  1. L'administrateur entre au Système.
  2. Le système affiche la page d'authentification.
  3. L'administrateur saisit le nom d'utilisateur et le mot de passe.
  4. Le système vérifie le nom d'utilisateur et le mot de passe.
  5. Le système affiche la page d'accueil.

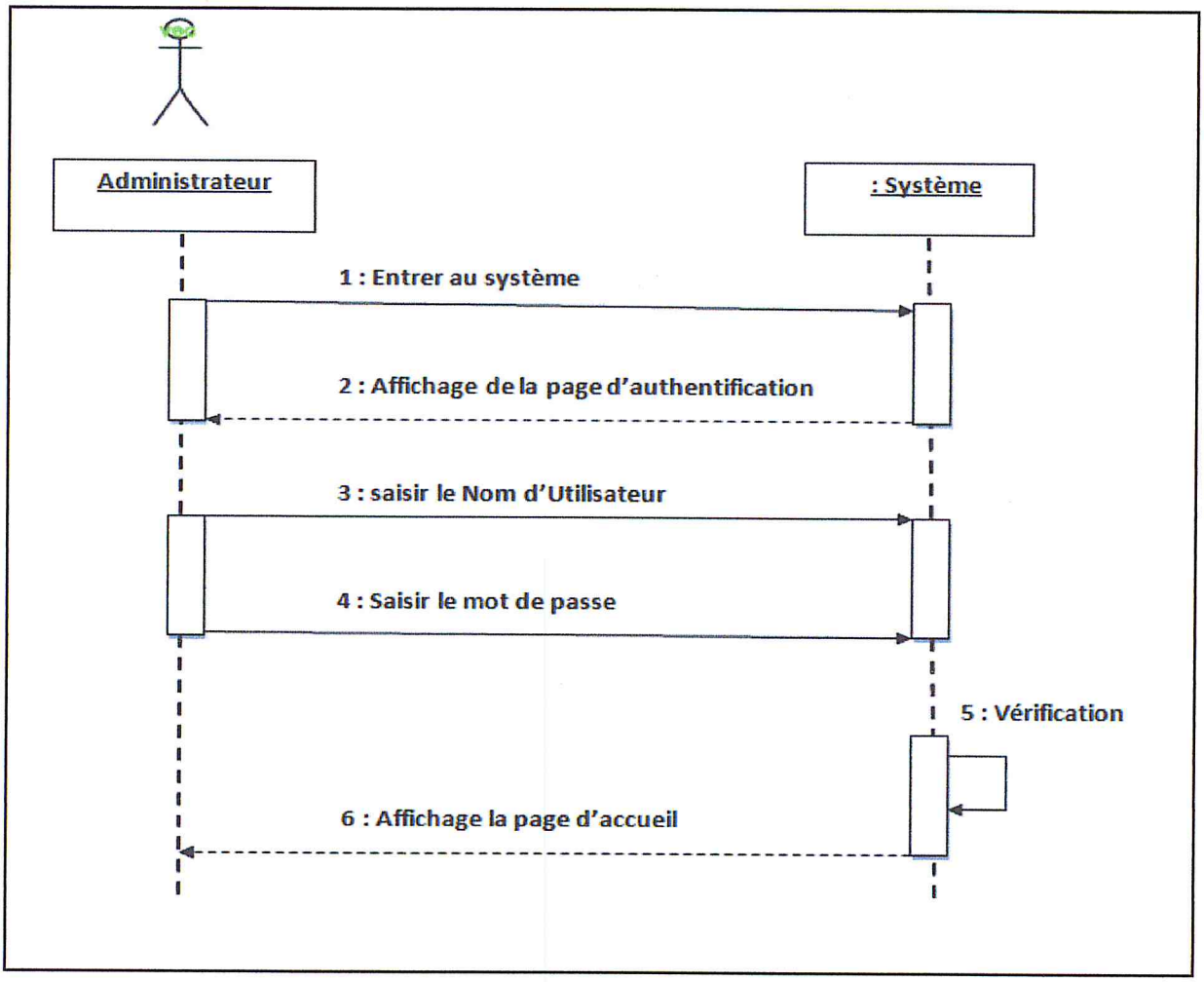


Figure 16 : Diagramme de séquence de processus d'authentification (cas normal)

- **Scenario de l'authentification (cas d'erreur)**
  1. L'administrateur entre au Système.
  2. Le système affiche la page d'authentification.
  3. L'administrateur saisit le nom d'utilisateur et le mot de passe.
  4. Le système vérifie le nom d'utilisateur et le mot de passe.
  5. Le système affiche «le nom d'utilisateur et le mot de passe n'est pas valide».

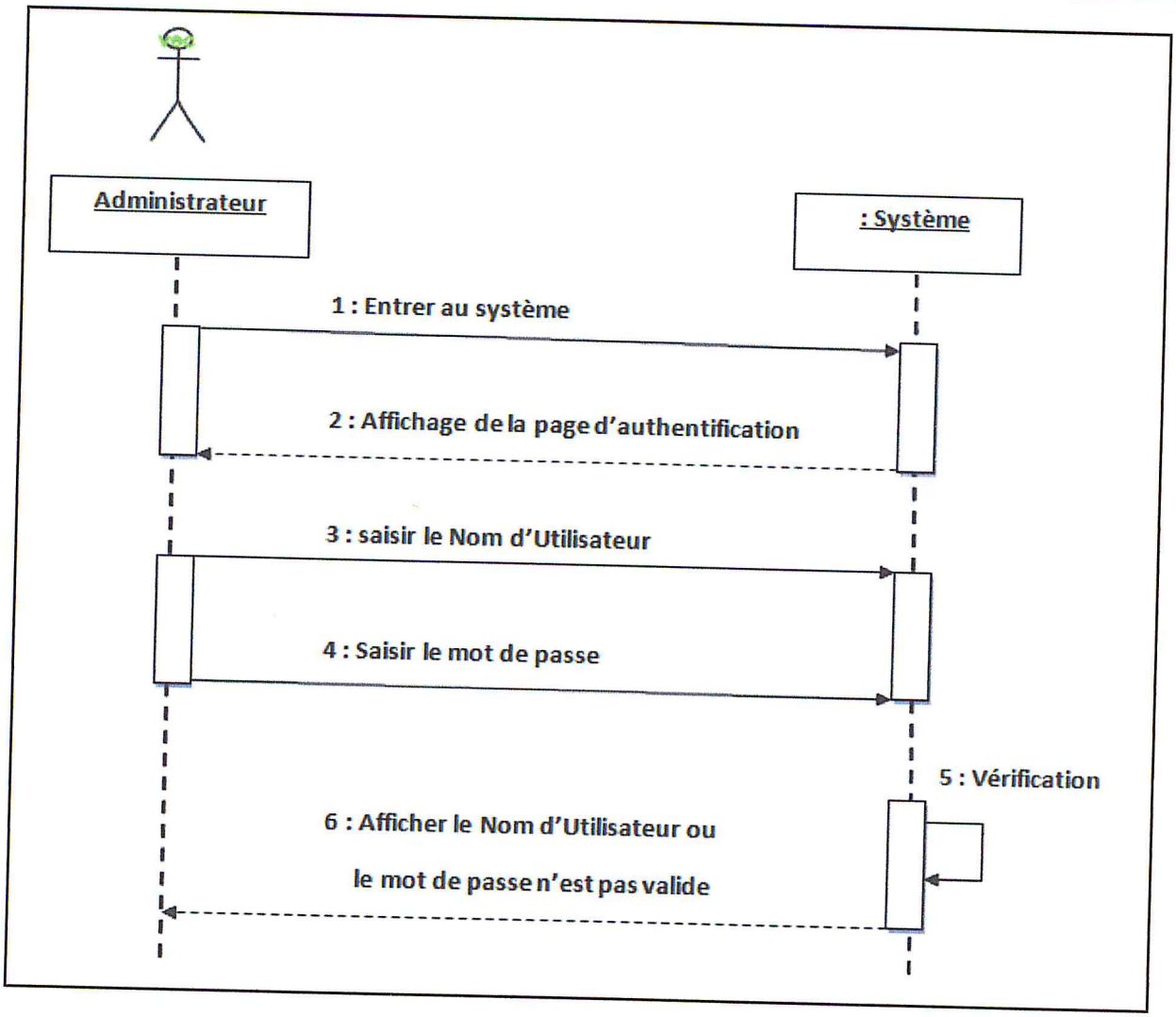


Figure 17 : Diagramme de séquence de processus d'authentification (cas erreur)

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

- **Scenario de lancement d'une requête**
  1. L'utilisateur doit poser une requête globale sur la page de lancement.
  2. Envoyer la requête au Médiateur.
  3. Le Médiateur produit le résultat.
  4. Le Médiateur envoi le résultat de la requête.
  5. Affichage de résultat a l'utilisateur.

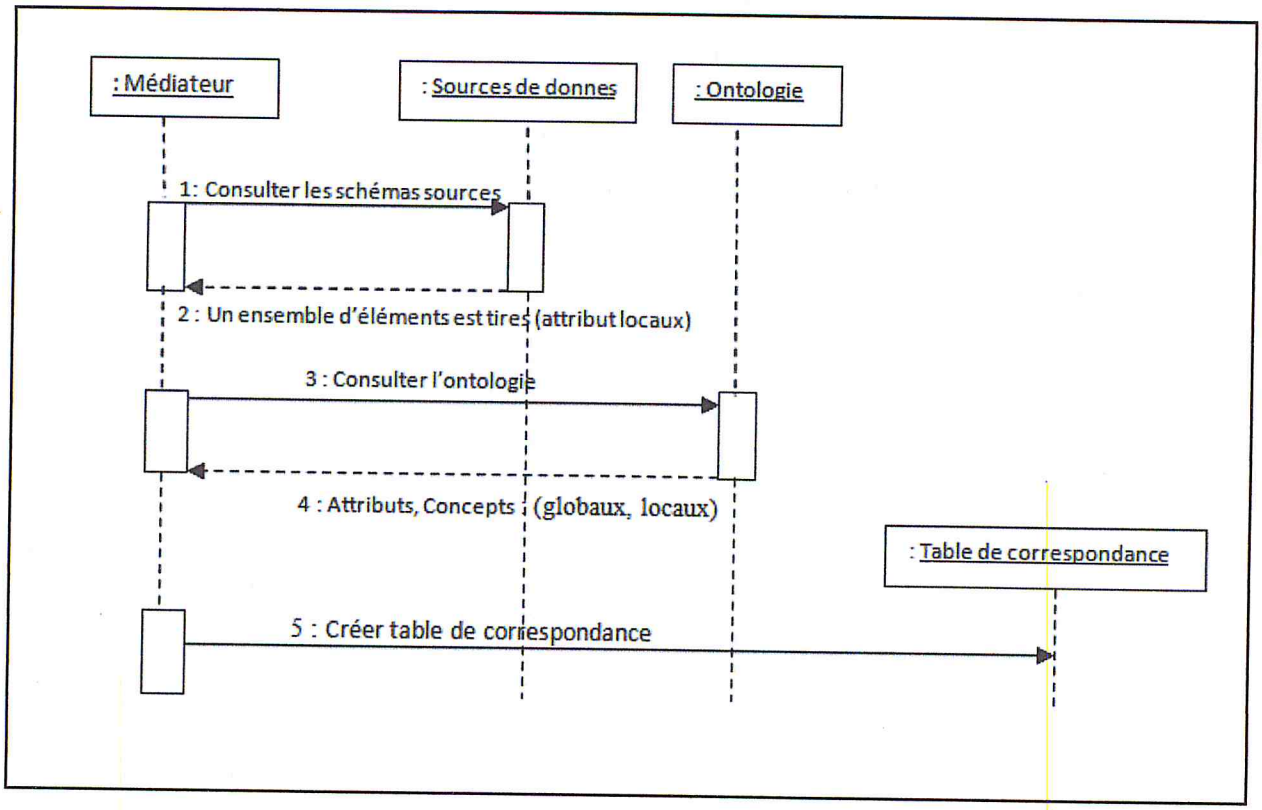


Figure 19 : diagramme de séquence du cas d'utilisation génération table de correspondance

3.2.4. Diagramme de séquence du cas d'utilisation Génération Data Guides

- **Scenario de Génération Data Guides**
  1. Consulter les sources des données.
  2. Un ensemble d'éléments est tires.
  3. Créer table Data Guides.

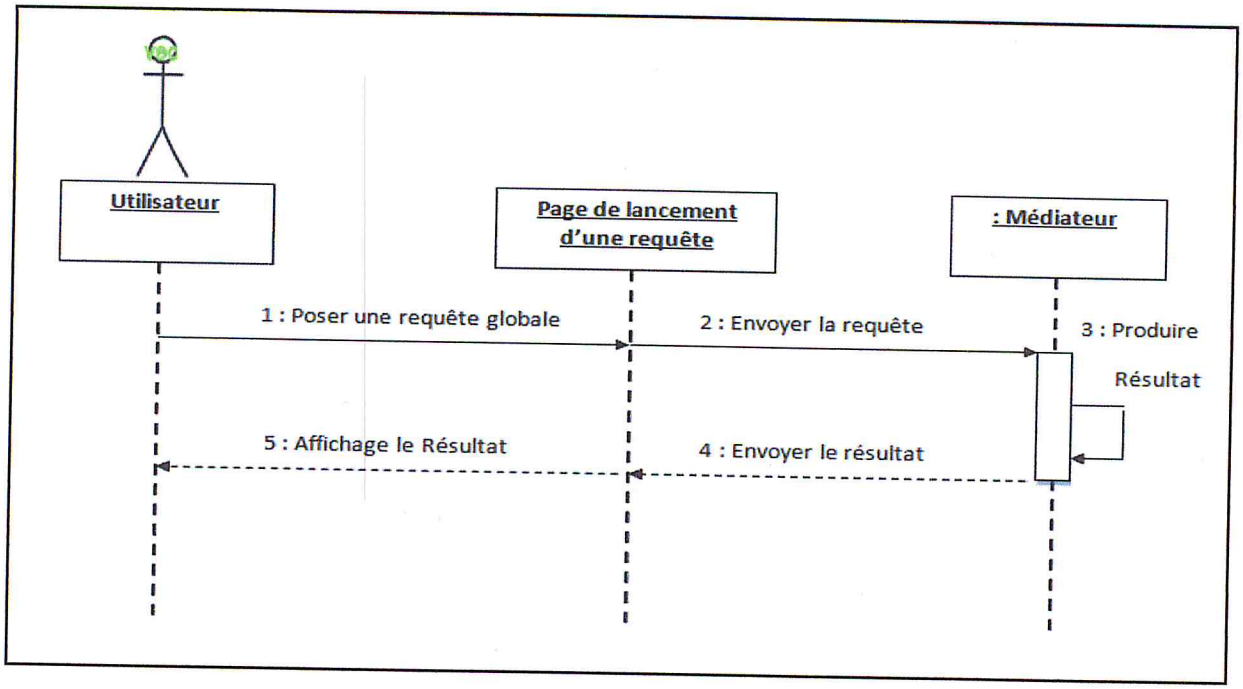


Figure 18 : diagramme de séquence du cas d'utilisation lancement d'une requête

3.2.3 diagramme de séquence du cas d'utilisation Génération Table de Correspondance

- **Scenario de Génération Table de Correspondance**
  1. Consulter les schémas sources.
  2. Un ensemble d'éléments est tires (attribut locaux).
  3. Consulter l'ontologie.
  4. Tirer Attributs, Concepts : (globaux, locaux).
  5. Créer table de correspondance.

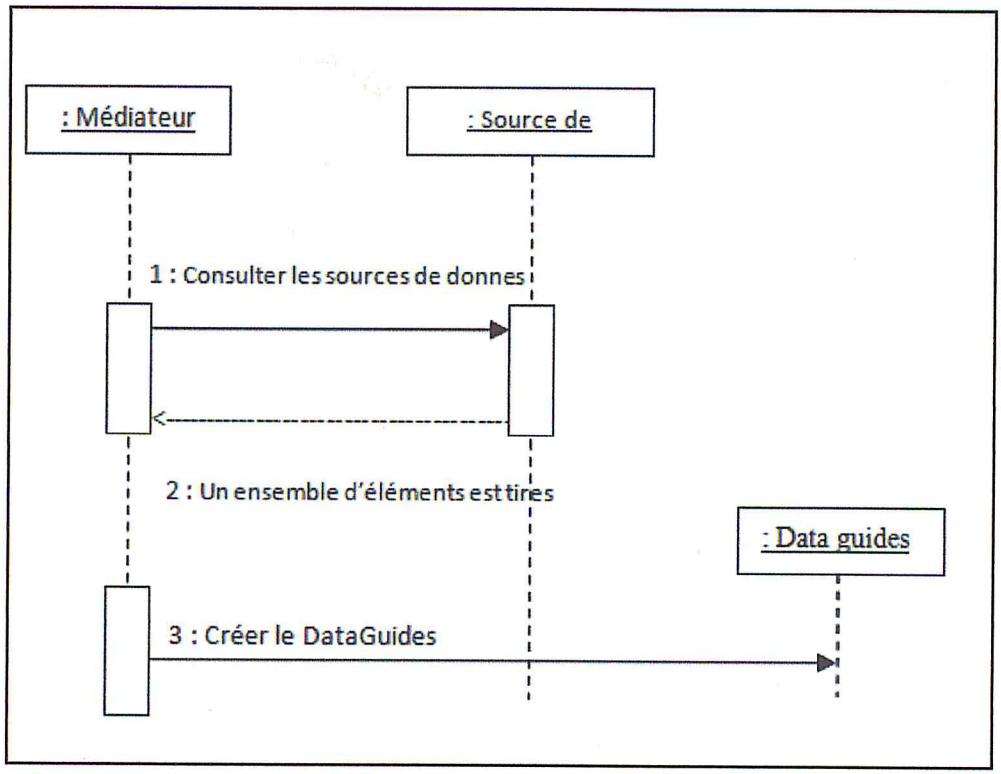


Figure 20 : diagramme de séquence du cas d'utilisation génération Data guides

3.2.5 diagramme de séquence du cas d'utilisation Traitement de la requête

- **Scenario de « Traitement de la requête » d'une manière détaillée.**

1 : Le système affiche la page lancement d'une requête.

2: Saisir la requête.

3 : Vérification de la requête.

4 : Envoie la requête au médiateur.

5 : Chercher le résultat de cette requête si existe dans fichier résultat.

6 : Affichage du résultat.

7 : Tirer les attributs locaux pour chaque attribut global et les concepts locaux pour le concept global.

8 : Tirer les sources approprie pour chaque table sélectionné

9 : Décomposer la requête en sous requête.

10: Envoyer le sous requête à l'adaptateur approprie.

11 : Traduire la source Relationnel en XML.

12 : Demande la réponse de la sous requête envoyée par l'adaptateur approprie.

13 : Renvoyer la réponse de la requête envoyée par l'adaptateur approprie.

14 : Renvoyer le résultat de la sous requête par l'adaptateur approprie.

15 : Recomposer les résultats.

16 : Renvoyer le résultat final.

17 : Affichage du résultat.

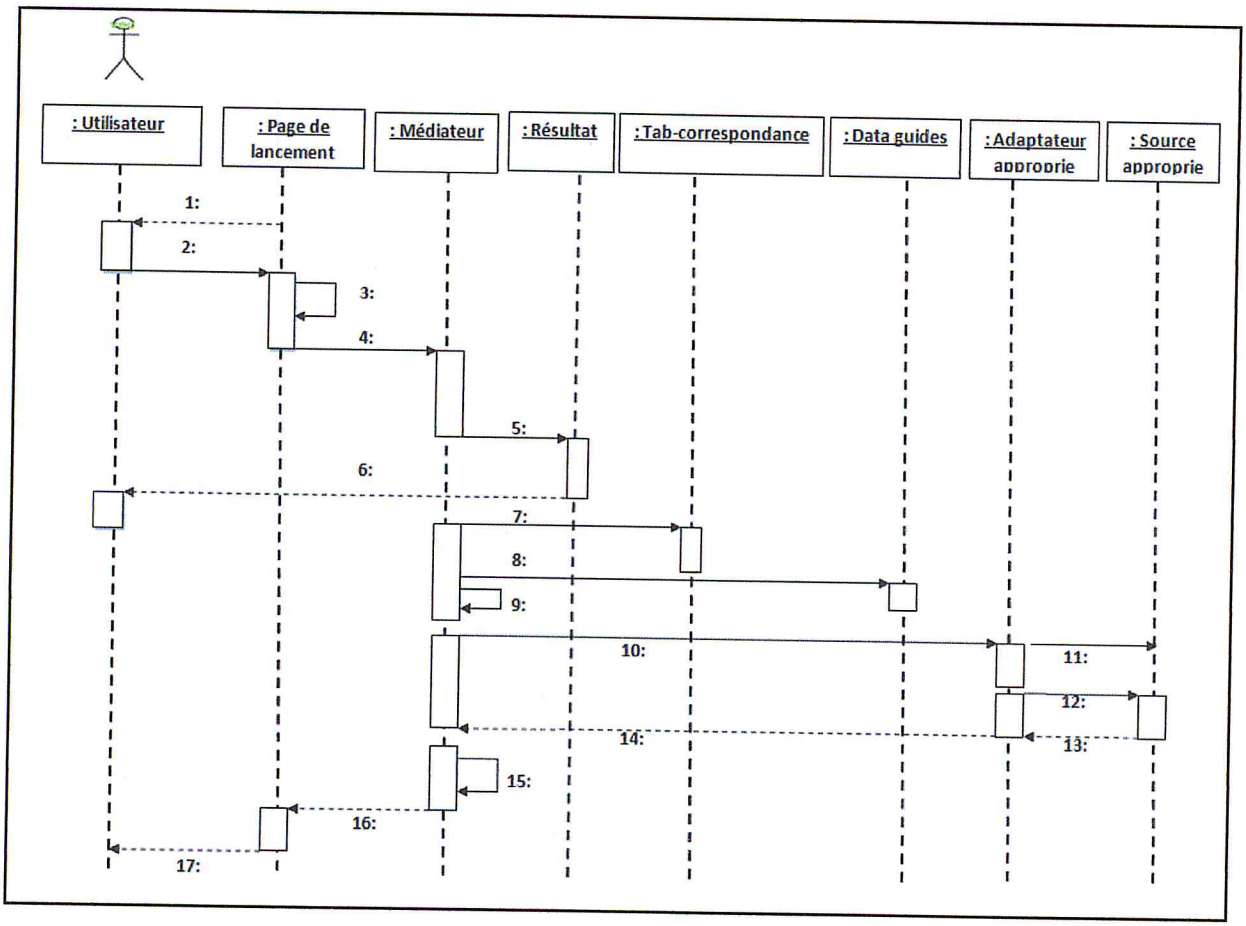


Figure 21 : diagramme de séquence du cas d'utilisation traitement de la requête global

3.2.6 diagramme de séquence du cas d'utilisation Traduction

- **Scenario de Traduction**

1. Sous requête en langage global envoyer par le Médiateur.
2. Traduire les sous Requête en langage local.
3. Interroger les sources de données.
4. Sous résultat envoyer par les sources des données.
5. Traduire les sous requête en langage global.
6. Sous résultats par les adaptateurs.

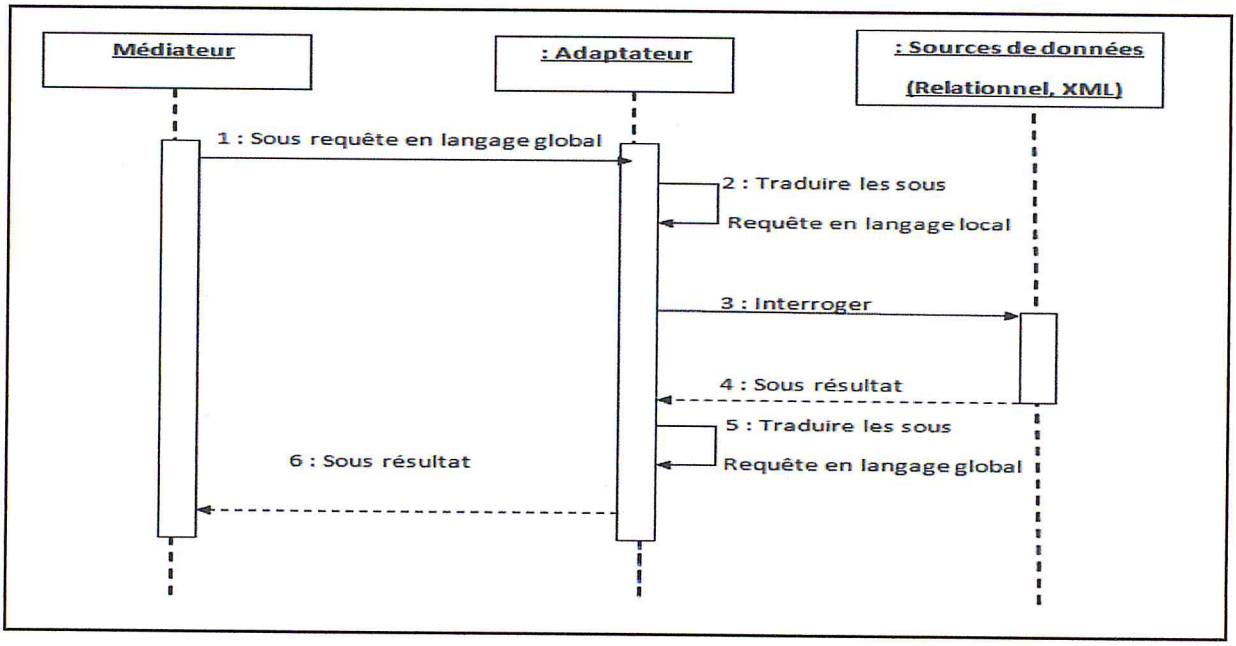


Figure : 22 diagramme de séquence du cas d'utilisation Traduction

3.2.7 diagramme de séquence du cas d'utilisation Fusion des résultats

- **Scenario de Fusion des résultats**

1. Sous réponses.
2. Recomposition les résultats.
3. Affichage les résultats.

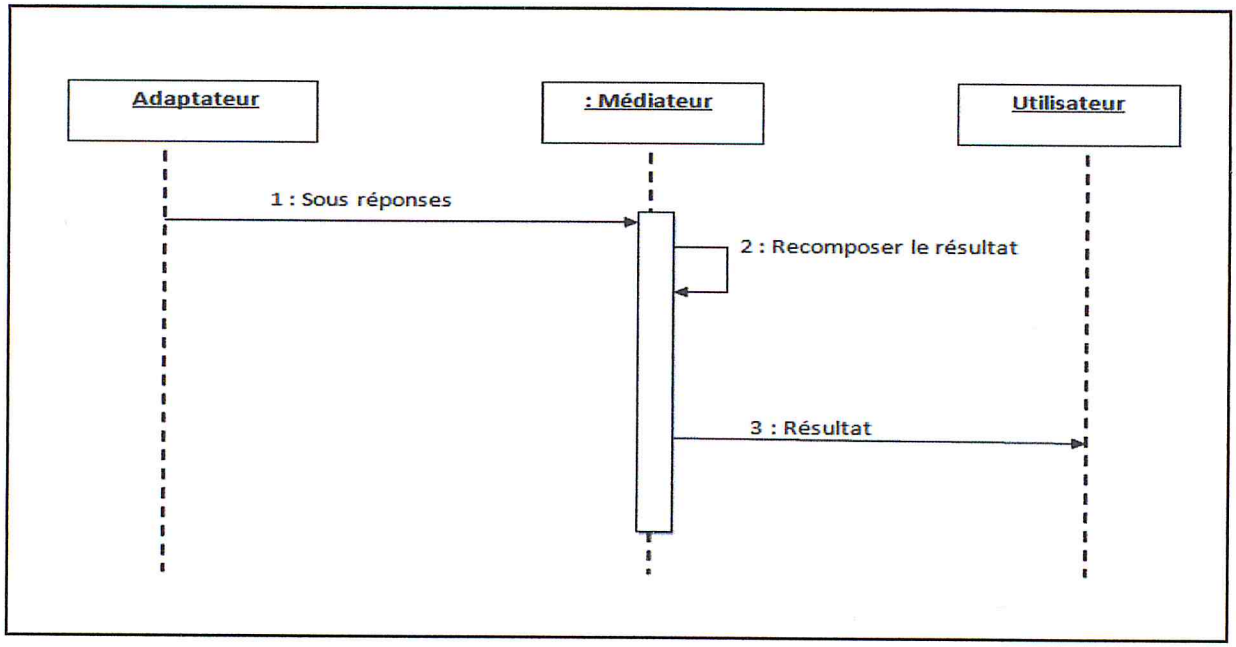


Figure 23 : diagramme de séquence du cas d'utilisation Fusion des résultats

4. Conception

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

4.1 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].**

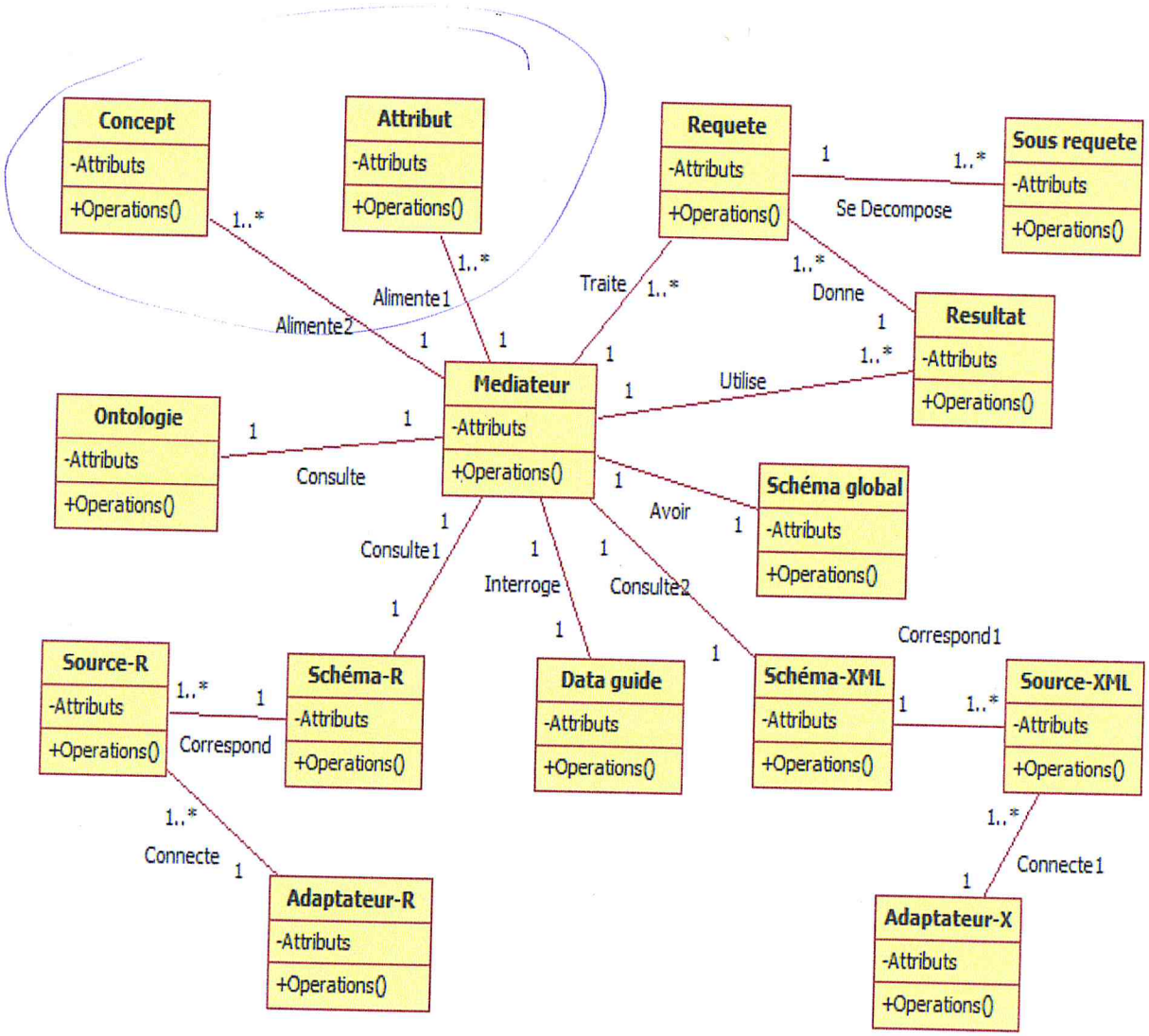


Figure 24 : Diagramme de classe

4.2 Concevoir les classes et les attributs

Classe	Attributs	Désignation	Type
Requête	Select	Les attributs globaux de la clause Select	String
	From	Les Concepts globaux de la clause From	String
Sous RequêteG	Select	Les attributs locaux de la clause Select	String
	From	Les Concepts locaux de la clause From	String
Schéma Global	Nom_Cpt	Nom concept	String
	Nom_attribut	Nom attribut	String
Schéma Relationnel	Nom_fichier	Nom fichier	String
	Emplacement	Emplacement	String
Schéma XML	Nom_fichier	Nom fichier	String
	Emplacement	Emplacement	String
Ontologie	Nom_fichier	Nom fichier	String
	Emplacement	Emplacement	String
DataGuides	Nom_fichier	Nom fichier	String
	Emplacement	Emplacement	String
Concept	Num_Cpt_L	Numéro concept local	Integer
	Nom_Cpt_L	Nom concept local	String
	Nom_Cpt_G	Nom concept global	String
	Nom_Source	Nom source	String
Attribut	Num_AttributL	Numéro attribut local	Integer
	Nom_AttributL	Nom attribut local	String
	Nom_AttributG	Nom attribut global	String
	Nom_Cpt_L	Nom concept local	String
Résultat	Requête_G	Requête Global	String
	Résultat	Résultat	String

Tableau 16 : Description des classes

4.3 Concevoir les opérations

Le développement du modèle dynamique a permis d'extraire les méthodes qui seront ajoutées aux classes définies ci-dessus. A ce niveau, nous allons donner une image assez précise du contenu des méthodes de notre projet, le tableau ci-dessus regroupe la description des différentes méthodes.

Classe	Méthode	Description
Requête	Créer_requête ()	Créer une requête
	Décomposer_requête ()	Requête décomposé
	Exécuter_requête ()	Exécuter Requête
Sous Requête	Créer_Srequête ()	Créer une sous requête
	Décomposer_Sreqête ()	décomposé sous Requête
	Exécuter_Srequête ()	Exécuter sous Requête
Ontologie	Consulter_Ontologie ()	Consulter le fichier des Ontologies
Concept	Consulter_Concept ()	Consulter la table concept
Attribut	Consulter_attribut ()	Consulter la table attribut
Médiateur	Consulte_dataG()	Le médiateur consulte le DataGuides
	Consulte_Résultats()	Consulter le fichier des Résultats
	Traite_requête()	Le médiateur Traite la requête
	Recomposer_résultat()	Le médiateur recompose le résultat
	Alimenter_Concept()	Le médiateur Alimente la table des concepts
	Alimenter_Attribut()	Le médiateur Alimente la table des attributs
Adaptateur	Traduire_SousRequête()	L'adaptateur traduit la sous requête du langage global au langage local
Dataguides	Interroger_Dataguides()	Interroger le fichier Dataguides
Resultat	Consulter_Resultat()	Consulter la table de résultat

Tableau 17 : Description des opérations

5 . Architecture du système

5.1. Architecture générale

5.1.1. Caractéristiques de la solution

Notre solution est caractérisée par :

1- Architecture centralisée qui offre une transparence d'accès à la localisation, aux schémas sources et aux langages de requêtes des données sources. Nous avons défini des stratégies de module pour l'efficacité du traitement des requêtes et l'amélioration du rendement global du système.

2- Offre une intégration sémantique des données en utilisant les nouvelles technologies pour la représentation de la sémantique des données (ontologies).

3- Localiser les sources pertinentes en utilisant la table d'annotation (dataguides).

4- Les données hétérogènes supportées sont les données relationnelles, XML;

5- Approche d'intégration descendante et une adaptation des règles de mapping GAV offrent une flexibilité aux changements;

6- Traitement des sous requêtes destinées aux sources de données permettant de supporter des systèmes anciens ;

7- Résoudre le problème de l'intégration de données hétérogènes et en prenant en compte les conflits des données (syntaxiques, sémantiques) à l'intégrer

5.1.2. Vue générale de l'architecture

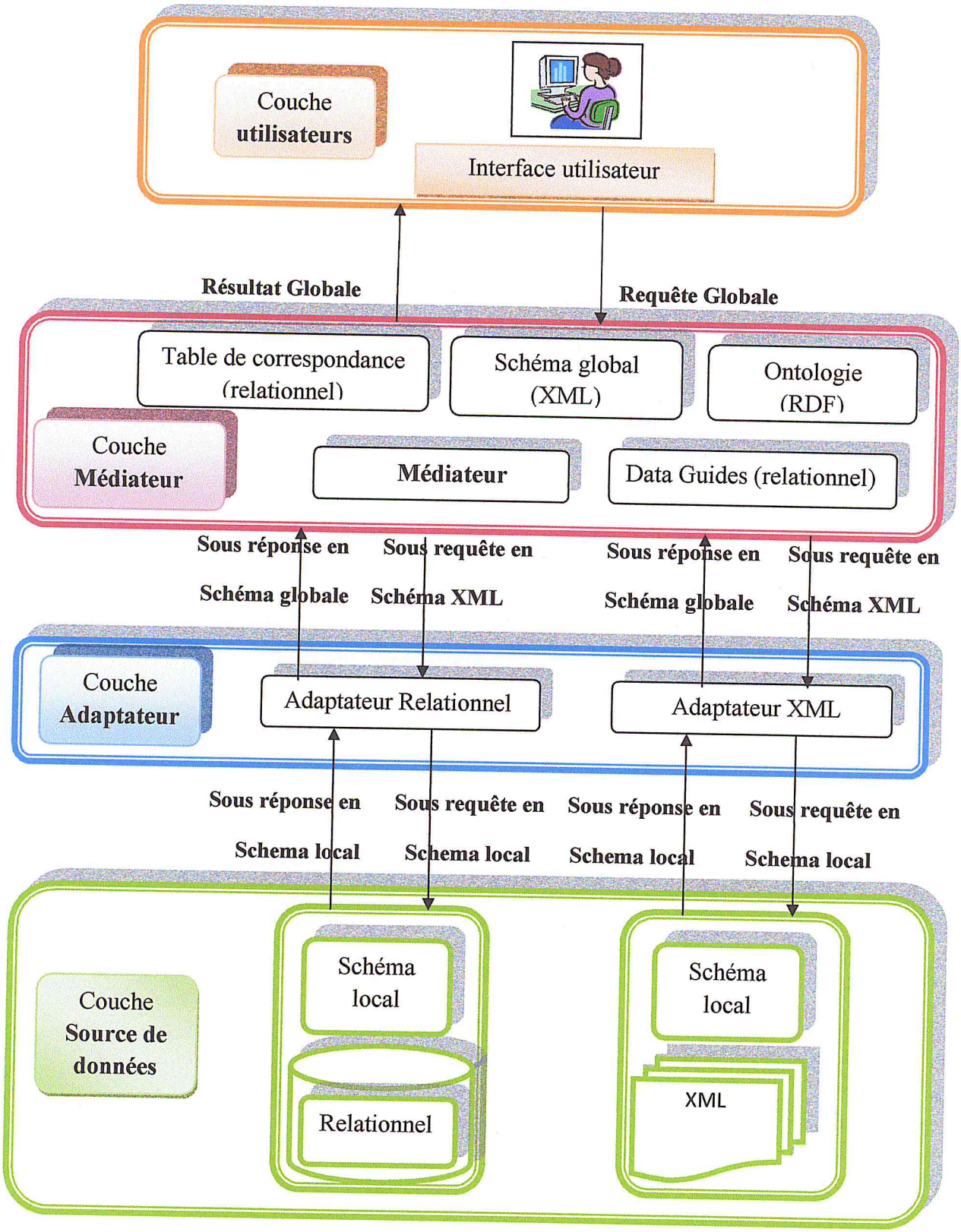


Figure 25 : Architecture générale du système

5.1.3. Description des couches de système

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

- **Couche Source de données**

Les sources de données utilisées peuvent être de nature structurée, semi structurée ou bien non structurée. Dans notre approche 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 " BdRMédicale" : pour concevoir cette base on a utilisé un SGBDR (Système de Gestion de Base de Données Relationnelle) MySQL. Cette base contient 5 tables

Malade (NSSMal, NomMal ,PreMal ,DateNaissMal).

Médecin (NSSMed,NomMed ,PreMed).

Maladie (Num,NomMad).

Atteint (NSSMal, Num,) .

Consultation (NSSMal , NSSMed ,dateCon, Prix).

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

```
<?xml version="1.0"?>
<Medecintraitant>
<NSSMedecin> 0123456789 </NSSMedecin>
<MENom>  cherifi      </MENom>
<MESpecialite> geniraliste </MESpecialite>
<MEPrenom>   hamid       </MEPrenom>
<MEAdresse>  boufarik       </MEAdresse>
<METelephone>  0779614308    </METelephone>
<MEDate_Naiss> 27/05/1988     </MEDate_Naiss>
</Medecintraitant>

<Patient>
<NSSPatient> 987654321      </NSSPatient>
<PANom>     souri         </PANom>
<PAPrenom>  youcef        </PAPrenom>
<PAStatut>  normale         </PAStatut>
<PASexe>    homme         </PASexe>
<PAAdresse>     tablat      </PAAdresse>
<PATelephone>  0775224263     </PATelephone>
<PADate_Naiss>  27/03/1988     </PADate_Naiss>
</Patient>
<Maladie>
<NumeroMaladie> 9</NumeroMaladie>
<NomMaladie> diabete    </NomMaladie>
```

Figure 26 : Fragment de la source XML

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.

Ces schémas locaux sont créés de la manière suivant :

1. Le nom de la base est représenté par élément dont le nom est celui de la base et qui imbrique tous les autres éléments.
2. Chaque table est représentée par un élément dont le nom est « Table »,
3. Le nom de la table est représenté par un élément dont le nom est « NomTable »,
4. Les attributs d'une table sont représentés par des sous éléments dont le nom est « NomAttribut».

```
<?xml version="1.0"?>
<BaseXml>
<Table>
    <NomTable> Medecintraitant</NomTable>
     <NomAttribut1>NSSMedecin</NomAttribut1>
     <NomAttribut2>MENom</NomAttribut2>
     <NomAttribut3>MESpecialite</NomAttribut3>
     <NomAttribut4>MEPrenom</NomAttribut4>
     <NomAttribut5>MEAdresse</NomAttribut5>
     <NomAttribut6>METelephone</NomAttribut6>
    <NomAttribut7>MEDate_Naiss</NomAttribut7>
</Table>
<Table>
    <NomTable>Patient</NomTable>
    <NomAttribut1>NSSPatient</NomAttribut1>
    <NomAttribut2>PANom</NomAttribut2
    <NomAttribut3>PAPrenom</NomAttribut3>
    <NomAttribut4>PAStatut</NomAttribut4>
    <NomAttribut5>PASexe</NomAttribut5>
    <NomAttribut6>PAAdresse</NomAttribut6>
    <NomAttribut7>PATelephoneNomAttribut7>
    <NomAttribut8>PADate_Naiss</NomAttribut8>
</Table>
<Table>
     <NomTable>Maladie</NomTable>
     <NomAttribut1>NumeroMaladie<NomAttribut1>
     <NomAttribut2>NomMaladie</NomAttribut2>
     <NomAttribut3>Description</NomAttribut3>
</Table>
<Table>
     <NomTable>Visite</NomTable>
     <NomAttribut1>NumVisite</NomAttribut1>
     <NomAttribut2>DateVisite</NomAttribut2>
     <NomAttribut3>PrixVisite</NomAttribut3>
      <NomAttribut4>NSSPatient</NomAttribut4>
      <NomAttribut5>NSSMedecin</NomAttribut5>
</Table>
<Table>
    <NomTable>Atteint</NomTable>
    <NomAttribut1>NSSPatientNomAttribut1>
    <NomAttribut2>NumeroMaladie</NomAttribut2>
</Table>
```

Figure 27 : Schéma local de la base XML

- **Couche Adaptateur**

L'adaptateur cache l'hétérogénéité au médiateur. Il est associé à une seule source et joue le rôle d'intermédiaire entre cette source 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 Relationnelle** : cet adaptateur est chargé d'exécuter la requête sur la source relationnelle appropriée 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

➢ **Couche Médiateur**

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

1. Accepter les requêtes des utilisateurs.
2. Localiser les sources de données pertinentes.
3. décomposer et optimiser les requêtes.
4. Envoyer les sous requêtes à faire exécuter par les adaptateurs sur les sources.
5. Combiner (recomposer) les résultats des adaptateurs et effectuer éventuellement quelques opérations supplémentaires.
6. 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 création le table de correspondance.
2. Module génération le data guides.
3. Module de traitement de requêtes.
4. Module de fusion des résultats.

Afin de bien accomplir ses tâches le médiateur doit être doté d'un schéma global, un fichier d'ontologie.

➢ **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 GAV (Global as View), c'est-à-dire que le schéma global est considéré comme étant une vue sur les schémas des sources

```
<?xml version="1.0" encoding="UTF-8"?>
<schemGlobal>
 <Concept>
      <NomCpt>Medecin</ NomCpt >
      <NomAttribut>NSSMedecin</ NomAttribut >
      <NomAttribut>NomMedecin</ NomAttribut >
      <NomAttribut>PrenomMedecin</ NomAttribut >
      <NomAttribut>MESpecialite</ NomAttribut >
      <NomAttribut>MEAdresse</ NomAttribut >
      <NomAttribut>METelephone</ NomAttribut >
      <NomAttribut>MEDate_Naiss</ NomAttribut >
 </ Concept >
 < Concept >
      < NomCpt >Malade</ NomCpt >
      < NomAttribut >NSSPatient</ NomAttribut >
      < NomAttribut >NomMalade</ NomAttribut >
      < NomAttribut >PrenomMalade</ NomAttribut >
      < NomAttribut >PAStatut</ NomAttribut >
      < NomAttribut >PASexe</ NomAttribut >
      < NomAttribut >PAAdresse</ NomAttribut >
      < NomAttribut >PATelephone</ NomAttribut >
      < NomAttribut >Date_NaissMalade</ NomAttribut >
 </ Concept >
 < Concept >
      < NomCpt >maladie</ NomCpt >
      <NomAttribute>NumeroMaladie</NomAttribute>
      <NomAttribute>NomMaladie</NomAttribute>
      <NomAttribute>Description</NomAttribute>
 </ Concept >
....
</schemGlobal>
```

Figure 28 : Fragment de schéma global

➢ **Ontologie :**

Dans la littérature **[SHET , 98][CULL , 03]**, on trouve trois types d' outils utilisés pour la représentation de la sémantique des informations : les méta-données, les ontologies et le contexte (considéré comme combinaison des méta-données et d' ontologies).

Qu'est qu'une ontologie ?

Une **ontologie** en informatique est un ensemble structuré de concepts permettant de donner un sens aux informations. Les concepts sont organisés dans un graphe dont les relations peuvent être : des relations sémantiques, des relations de composition et d'héritage (au sens objet).
Une Ontologie permet de donner du sens au contenu des ressources de manière à ce que leur localisation et interrogation soient plus aisées et plus pertinentes. C'est une solution potentielle importante pour représenter explicitement la sémantique des informations et aider à la détection et à la résolution des conflits sémantiques **[HAKI 03]**.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
        <rdfs:Class rdf:about="Environnment_Medicale">
        </rdfs:Class>
        <rdfs:Class rdf:about="MedecinG">
                <rdfs:subClassOf rdf:resource="Environnment_Medicale"/>
        </rdfs:Class>
        <rdfs:Class rdf:about="medecintraitant">
                <rdfs:subClassOf rdf:resource="MedecinG"/>
        </rdfs:Class>
        <rdfs:Class rdf:about="medecin">
                <rdfs:subClassOf rdf:resource="MedecinG"/>
        </rdfs:Class>
……
</rdf:RDF>
```

Figure 29 : Fragment de fichier Ontologie

Remarque :

La création de fichier ontologie, ainsi les schémas sources pour la solution proposée est supposée faits 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) MySQL. Cette base contient deux tables

Concept (Nom_Cpt_G, Nom_Cpt_L).

Attribut (Nom_Att_G, Nom_Att_L).

- **Data guides**

Cette méthode utilisé pour éliminer les sources qui ne contient pas des réponses pour une requête globale .Une Base de données relationnelle qui porte le nom " DataGuides" : pour concevoir cette base on a utilisé un SGBDR (Système de Gestion de Base de Données Relationnelle) MySQL. Cette base contient une seul table

Data guides (NomSources, NomTable, NomAttribut).

- ***Couche utilisateurs***

C'est une simple Interface de communication permet à un utilisateur 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.

5.2 Description des modules

5.2.1. Le module Génération Table de correspondance

Algorithme : Génération Table de correspondance

Entrée : Schémas locaux, Ontologie

Sortie : Table de correspondance

Début Connecter a l'ontologie

Si connexions établit **alors**

Consulter l'ontologie

Tirer l'ensemble de concepts locaux E_1

Pour chaque concept local CL_i **faire**

Tirer CG_i pour les CL_i dans le fichier d'ontologie

Insérer le CG_i et leur CL_i dans le Table de Correspondance

Sinon rien faire

Connecter aux sources de données

Si connexions établit **alors**

Consulter les schémas locaux

Pour chaque attribut local AL_i de l'ensemble E_2 **faire**

Tirer AG_i pour les AL_i dans le fichier d'ontologie

Insérer le AG_i et leur AL_i dans le Table de Correspondance

Sinon rien faire

Fin

5.2.2. Le module Génération Data Guides

Le processus Génération DataGrid passe l'algorithme suivant :

Algorithme : Génération Data guides
Entrée : Schémas locaux
Sortie : Data guides
Début
Connecter aux sources de données
Pour chaque source de donnée
Si connexions établit **alors**
Consulter le schéma local
Extraire l'ensemble des noms des tables (TA_i)
Pour chaque nom de table (TA_i) **faire**
Extraire l'ensemble des attributs locaux (AL_i)
Si AL_i existe **alors** insérer dans data guides pour chaque nom de source NS_I, nom table TA_i **et** AL_i
Sinon rien faire
Sinon rien faire
Fin

5.2.3. Le module Traitement de requête

Ce module est chargé de traiter une requête globale (de l'utilisateur) deviser 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

Avec:

AG $< AG_1, AG_2, \ldots, AG_n >$ représente l'ensemble d'attributs projetés,

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

La requête Q sera donc décomposée en sous requêtes.

Soit Q_i une sous requête correspondant 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 Ci 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 : traitement de requête

Entrée : - Requête en schéma global (RG)

Sortie : - Sous requêtes en schéma local (SRL)

Début

Chercher dans le fichier résultat si la requête global existe

Si RG existe

Alors Tirer le résultat

Sinon

Insérer RG dans le fichier resultat

Tirer l'ensemble des attriduts locaux AL pour AG_i (table de,correspondance)

NombreDeSousRequête=card(AL)

Créer des sous requêtes globales selon ce nombre

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

// vérifier la correspondance Concept local / attribut local (AL_i / CL_j)

Pour chaque AL_i voir CL_j **Si** $i = j$

Alors pour chaque AL_i tirer le nom de la table NT_i (dataguides)

réécrire la SRL_k avec AL_i et Nom de table (NT_i).

// Déterminer pour chaque sous requête local SRL_i la source local relative (dataguides)

Pour chaque sous requête local SRL_i

Tirer l'ensemble des sources pour le nom de table NT_i (data guides)

Fin

Exemple : Afin de faciliter la compréhension de notre algorithme, considérons l'exemple suivant. Soit la requête utilisateur suivante :

{ **Select** NomMedG, NSSMedecin **From** MedecinG }.

Cette dernière peut être décomposée en trois sous requêtes définie sur le schéma global, chacune va être destinée à une source.

L'attribut globale " NomMedG " a deux attribut local {MENom, NomMedecin}.

Le concept globale " MedecinG " a deux concept local {Medecintraitant, Médecin}.

Sous Requête 1 : **Select** MENom **From** Medecintraitant

⟶ Cette sous requête va être exécuté sur la source XML.

Sous Requête 2: **Select** NomMedecin **From** Medecin

⟶ Cette sous requête va être exécuté sur la Source Relationnel.

Sous Requête 3: **Select** NSSMedecin **From** Medecin

⟶ Cette sous requête va être exécuté sur la Source Relationnel.

5.2.4. Le module Fusion des résultats

Ce module construit la réponse d'une requête globale en utilisant les résultats des requêtes locales envoyées par les adaptateurs. Nous présentons l'algorithme de reconstruction de la réponse d'une sous requête Qi.

Algorithme : Fusion des sous résultats

Entrée : Un ensemble de sous réponse
Sortie : réponse globale

Début

Pour chaque sous réponse SR **Faire**
Insérer dans la réponse Global RG, SR
Insérer dans le fichier résultat la réponse Global RG

Fait

Fin

6. 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 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 du ce que nous avons proposé, en d'autres termes, la réalisation d'un d'optimisation des requêtes dans un système de médiation des données hétérogène et les outils que nous utilisons pour l'implémenter.

1. Introduction

Après l'expression des besoins et la modélisation du système à développer, nous allons, dans cette partie, présenter son implémentation et sa mise en œuvre. Donc l'utilisation des outils de développement permettent de réduire considérablement la charge de travail, parmi ces outils on trouve les langages de programmation. Et pour les données l'utilisation des SGBD simple et puissant facilitent le travail.

2. Présentation des outils de développement

2.1 Choix du langage de programmation (JAVA)

Le langage utilisé pour la réalisation du notre logiciel est le JAVA qui est un langage de programmation orienté Objet développer par SUN, le choix de ce langage du aux avantages suivant : **[JER ,08]**

- Java est un langage orienté objets
- Java est extensible à l'infini
- Java est un langage à haute sécurité
- Java est un langage simple à apprendre
- Java est portable

Mais pour avoir plus de confort il est préférable d'utiliser un environnement de développement intégré ou IDE.

Notre choix s'est porté vers L'EDI éclipse, qui nous fournit le confort et la simplicité nécessaires à un développement propre et rapide.

De plus se qui concerne notre projet, JAVA fournit tous ce qui est nécessaire à la manipulation des bases de données tels que : établir une connexion vers une base de données, exécuter des requêtes à partir de code JAVA,….etc. Et ceci est possible à l'aide de l'API JDBC que nous allons définir par la suite.

2.2. Système de Gestion de Base de données (SGBD)

2.2.1. SGBD Relationnel

Comme SGBD relationnel on a choisit MySQL (My Structured Query Language), qui fait partie des logiciels de gestion de base de données les plus utilisés au monde, C'est un logiciel libre qui développé sous double licence, produit libre ou produit propriétaire.

Le serveur de gestion de base de données MySQL permet d'assurer :

- La définition et la manipulation des données
- La cohérence des données
- La sauvegarde et la restauration des données
- La gestion des accès concurrents.

2.2.2. Manipulation des données semi structurées

La manipulation d'un document XML se fait par le biais de parseur. Un parseur est une application dont le rôle est de convertir un flux de balisage en une sortie accessible par un programme. Le parseur vérifie la conformité d'un document à la norme XML, à savoir qu'un document XML doit être bien formé.

Comme parseur on a choisit JDOM qui tire ses forces des Deux Parseur Sax et Dom, JDOM utilise des collections SAX pour parser les fichiers XML, et utilise DOM pour manipuler les éléments créer par SAX.

JDOM permet donc de construire des documents de naviguer dans leur structure et d'ajouter, modifier ou supprimer leurs contenu.

L'utilisation de l'API JDOM dans le traitement de données XML avec Java est simple malgré que cette API est toute jeune et en voie d'être améliorer.

2.2.3. JDBC (Java Data Base Connectivity)

JDBC est une API fournie par JAVA, cette API est constituée d'un ensemble d'interface et de classes qui permettent l'accès, a partir d'un programme java a des données tabulaire (triées sous forme de table).

Par données tabulaires on entend généralement les bases de données contenues dans des SGBD relationnels, Mais JDBC n'est pas restreinte a ce type de sources de données, On peut aussi accéder a des sources de données sous forme de fichiers (Fichier XML par exemple).

L'API JDBC a été développée de telle façon à permettre à un programme de ce connecté à n'importe quelle base en utilisant la même syntaxe.

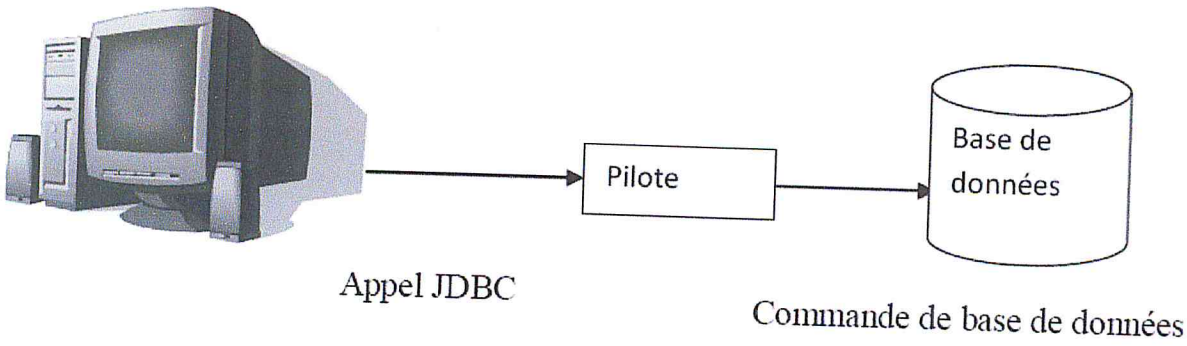


Figure 30: Architecture de JDBC

Les accès à une base de données doivent être effectués en septe étapes consécutives :

- Chargement d'un pilote JDBC
- Définition de l'URL de connexion
- Etablissement de la connexion
- Création d'une instruction
- Exécution d'une requête
- Traitements des résultats
- Fermeture de la connexion.

3. Démonstration

L'accès à notre Système peut se faire de deux manières :

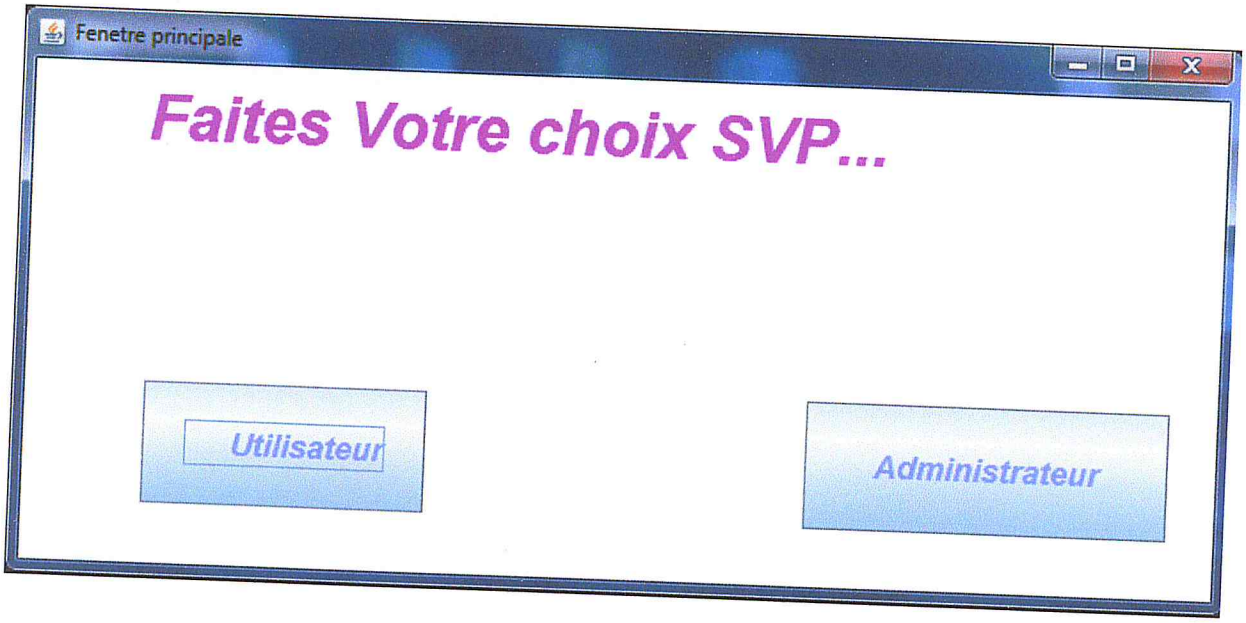


Figure 31 : page d'accueil de système.

✓ **Accès Utilisateur:**

Pour les utilisateurs qui ne disposent pas des droits d'accès administrateur, ils peuvent accéder à notre système de manière restreinte, par exemple ils ne peuvent pas générer le data guides.

✓ **Accès administrateur**:

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

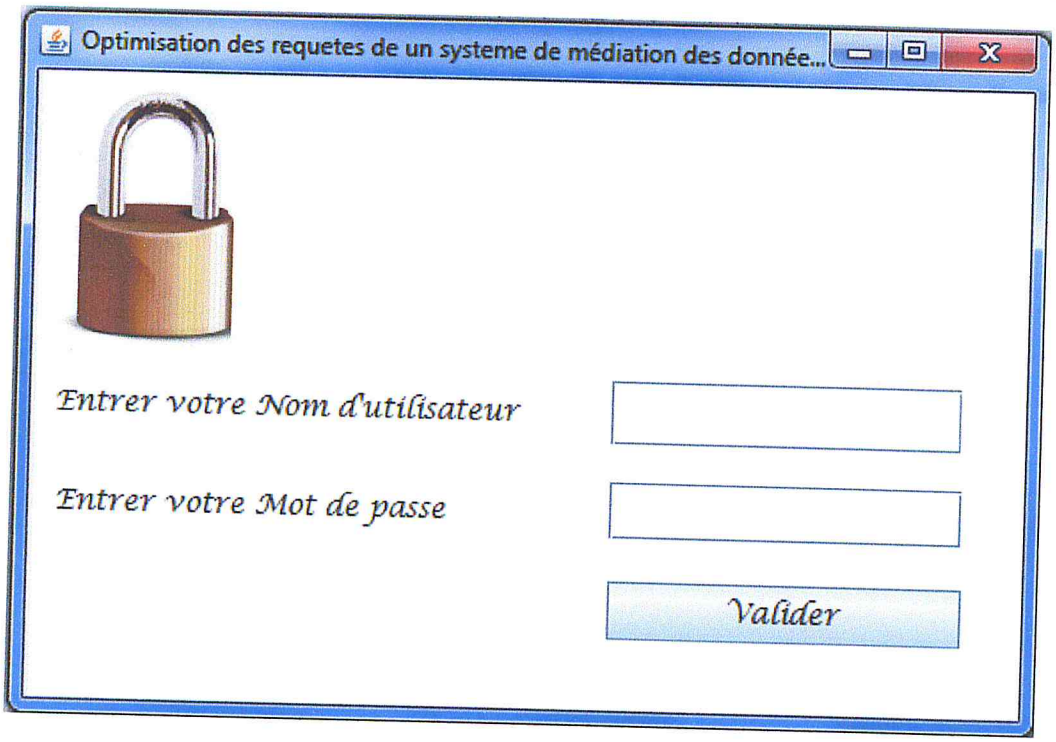


Figure 32 : page d'authentification.

L'administrateur

- **Page d'accueil :** Dans cette page l'administrateur a le droit de :
  - Consulter Ontologie.
  - Générer le DataGuides.
  - Générer le Table de correspondance.
  - Poser une requête.

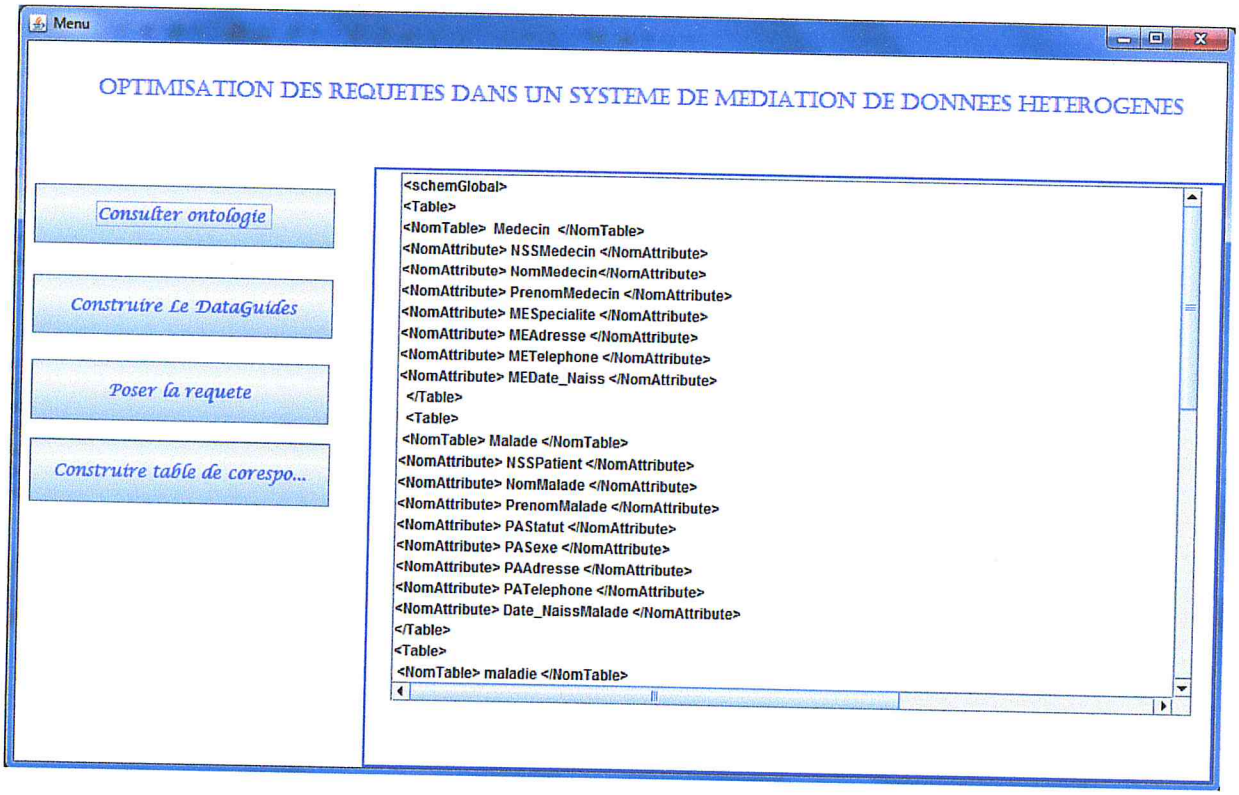


Figure 33 : page d'accueil de l'administrateur.

- **Consulter Ontologie :**

L'administrateur consulte Ontologie

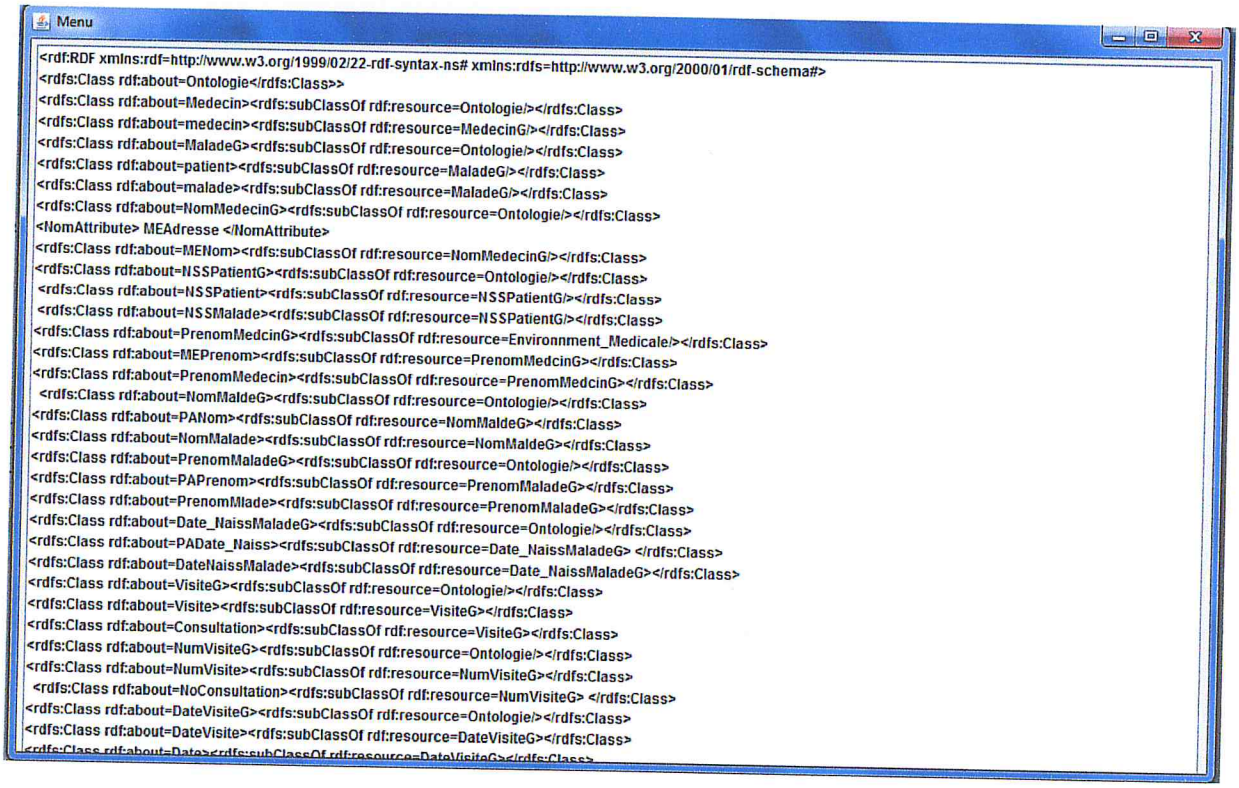


Figure 34 : Consulter ontologie.

- **Générer le Table de correspondance:**

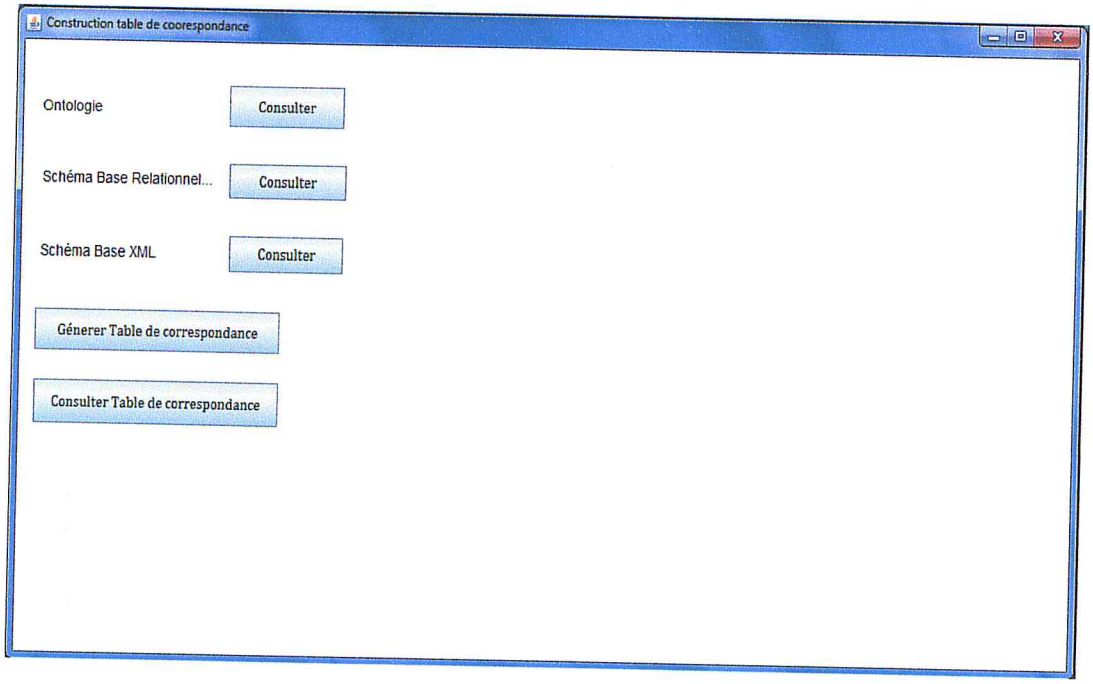


Figure 35: génération le table de correspondance.

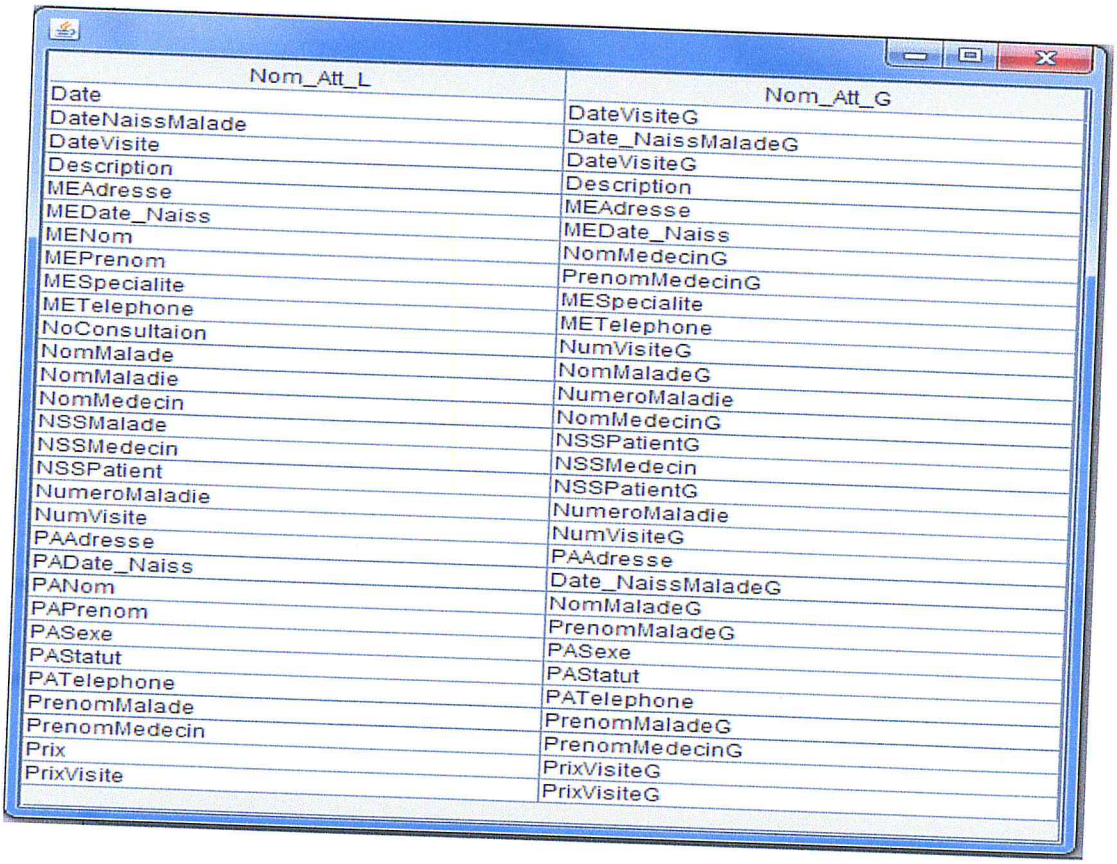
- **Consulter le Table de correspondance:**

Table concept :

Nom_Cpt_L	Nom_Cpt_G
Consultation	VisiteG
Malade	MaladeG
Medecin	Medecin
Medecintraitant	MedecinG
Patient	MaladeG
Visite	VisiteG

Figure 36: table concept.

Table attribut :



Nom_Att_L	Nom_Att_G
Date	DateVisiteG
DateNaissMalade	Date_NaissMaladeG
DateVisite	DateVisiteG
Description	Description
MEAdresse	MEAdresse
MEDate_Naiss	MEDate_Naiss
MENom	NomMedecinG
MEPrenom	PrenomMedecinG
MESpecialite	MESpecialite
METelephone	METelephone
NoConsultaion	NumVisiteG
NomMalade	NomMaladeG
NomMaladie	NumeroMaladie
NomMedecin	NomMedecinG
NSSMalade	NSSPatientG
NSSMedecin	NSSMedecin
NSSPatient	NSSPatientG
NumeroMaladie	NumeroMaladie
NumVisite	NumVisiteG
PAAdresse	PAAdresse
PADate_Naiss	Date_NaissMaladeG
PANom	NomMaladeG
PAPrenom	PrenomMaladeG
PASexe	PASexe
PAStatut	PAStatut
PATelephone	PATelephone
PrenomMalade	PrenomMaladeG
PrenomMedecin	PrenomMedecinG
Prix	PrixVisiteG
PrixVisite	PrixVisiteG

Figure 37: table attribut.

- **Générer le DataGuides:**

  L'administrateur ici consulter le schéma de DataGuides et source relationnel ainsi schéma XML après peut générer la DataGuides et le consulter.

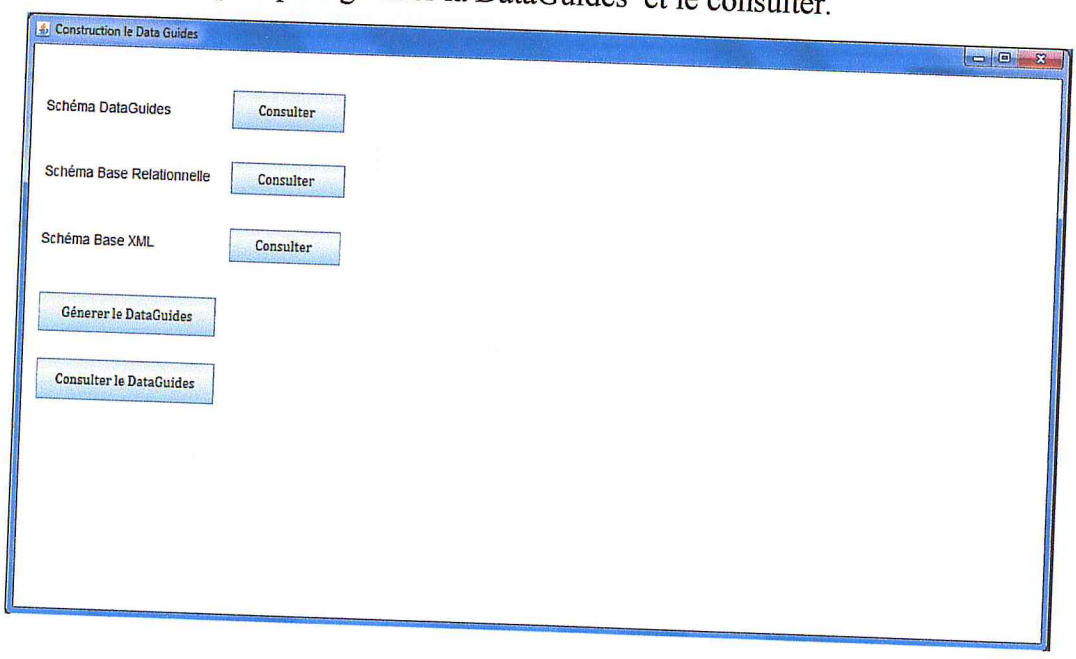


Figure 38: génération le Data Guides.

- **Consulter le Data Guides:**

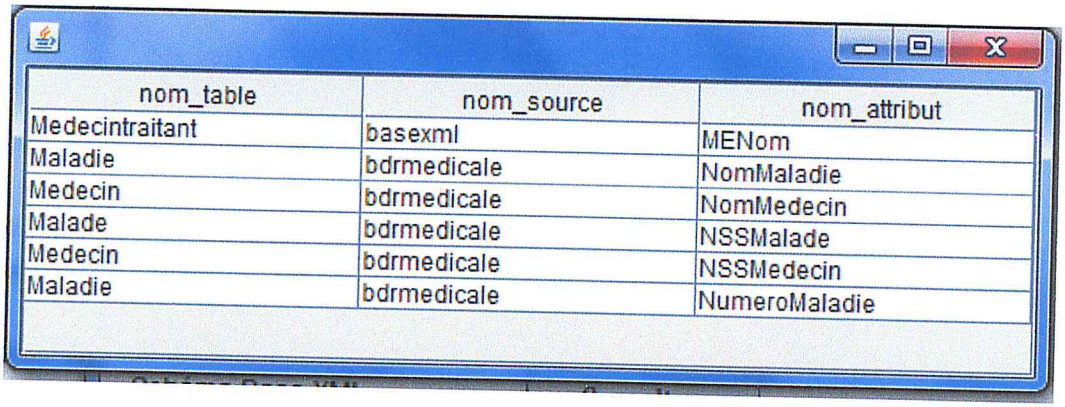


Figure 39: consulter le Data Guides

- **Poser une requête :**

Avant de poser une requête l'administrateur doit sélectionner un concept global, une fois le concept global est sélectionné, l'administrateur dois charger les attributs ensuit choisit deux attributs, validé la requête le système occupe la décomposition et localisation des sources et afficher le résultat .

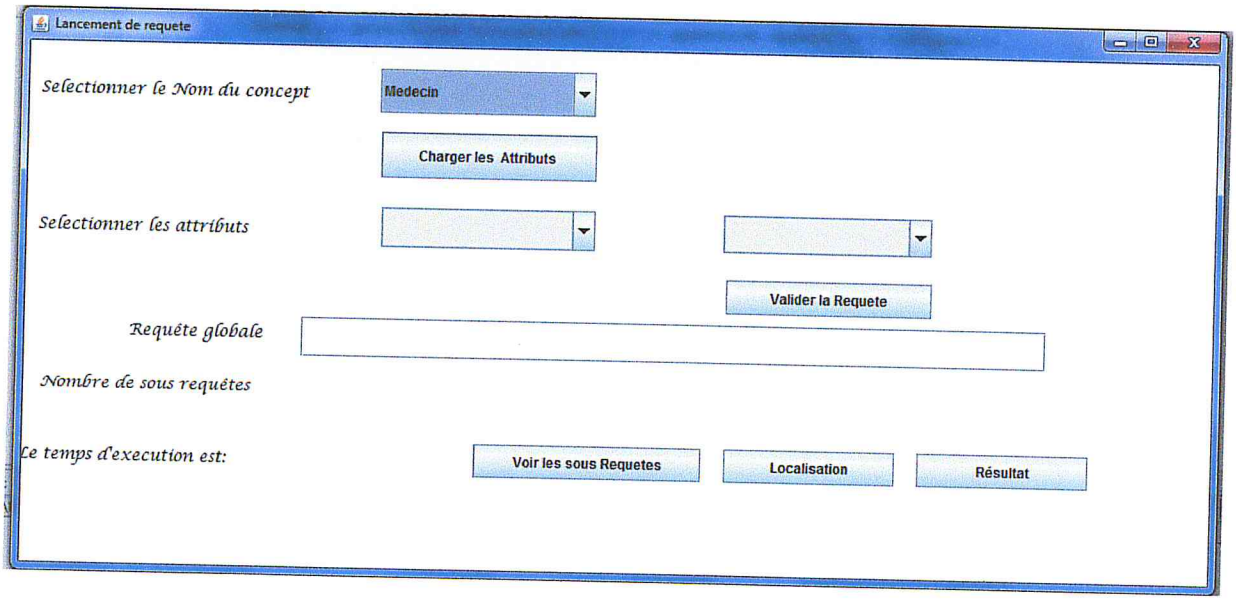


Figure 40: page pour poser une requête.

4. CONCLUSION

Dans ce chapitre, nous avons présenté les supports logiciels utilisés, ainsi que nos sources de données et les différents éditeurs qu'on a utilisé pour traiter ces sources, ensuite on a exposé l'environnement de développement choisi (langage de programmation, langage d'interrogation,…… etc.), dans le quel repose notre application.

Enfin, 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.

Références

[GEN, 04] Genoveva Vargas Solar , Anne Doucet, médiation de données :solution et problemes ouverts, Laboratoire Logiciels, systèmes , réseaux 681 rue de passerelle, (2004).

[ROUSS, 02] M-C.Rousset, A.Bidault,C Froideveaux, H.Gagliardi, F.Gouasdoué, C.Reynaud etB. Safar, Construction de mediateur pour intégrer de sources d'information multiples et hétérogènes : Le projet Picsel, (2002).

[BOU, 08] Amel Boussis, intégration de sources de donnes a base ontologique dans un envirenement P2P, These de magister .INI,(2008).

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

[Widom, 99] J. McHugh et J. Widom. Query Optimization for SemiStructured Data. Technical report, Stanford University Database Group, (1999).

[VER, 06] Laïla BENHLIMA ET Dalila CHIADMI, Vers l'interopérabilité des systèmes d'information hétérogènes, Ecole Mohammadia d'Ingénieurs BP 765 Agdal, Rabat Maroc,(2006).

[TIA, 11] Tianxiao Liu, Proposition d'un Cadre Générique d'Optimisation de Requêtes dans les Environnements Hétérogènes Répartis, Thèse de doctorat , Université de Cergy-Pontoise (2011)

[TRA, 06] Optimisation Extensible dans un Médiateur de Données Semi-Structurées, Thèse de doctorat, Université de Versailles Saint-Quentin-en-Yvelines(2006).

[BAKH, 05] Annotation des attributs pour l'optimisation des requêtes dans un système de médiation de bases de données, Abdelghani Bakhtouchi Ecole Militaire Polytechnique Alger, Ladjel Bellatreche LISI/ENSMA/ Université de Poitiers Poitiers/ France, Amar Balla Ecole Nationale Supérieur d'Informatique Alger, Algérie,(2005).

[LAU, 06] Laurent Audibert, UML 2 de l'apprentissage à la pratique, FNAC, amazon.fr, (2006).

[JER, 08] JEROM BOUGEAUT, java la maitrise édition 2008.

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

[SHET 98] Amit P. Sheth, "Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics", 1998

[CULL 03] Nadine Cullot - Fabrice Jouanot- Kokou Yétongnon, Une méthode de réconciliation sémantique pour l'extraction de connaissances. Laboratoire Electronique Informatique Image université de Bourgogne– France, 2003.

[HAKI 03] Farshad Hakimpour, Using Ontologies to Resolve Semantic Heterogeneity for Integrating Spatial Database Schemata, Mathematisch naturwissenschaftlichen Fakultat der Universitat Zürich, Zürich 2003.

[ANT , 02] PLAN SELECTION based on QUERY CLUSTERING,Antara Ghosh Jignashu Parikh Vibhuti Sengar Jayant Haritsa, Computer Science & Automation Indian Institute of Science Bangalore, INDIA, 2002

[CHA, 2004] Cours de XML - Initiation aux Schema XML, G. Chagnon, http://www.gchagnon.fr/cours/xml/schema.html, 2004

1. CONCLUSION GENERALE :

Les travaux présentés dans ce mémoire se situent dans le contexte de médiation de données, et plus particulièrement le module d'optimisation des requêtes.

Le projet qui nous a été confié, porte sur la réalisation d'un système d'optimisation des requêtes par l'approche médiateur. Donc nous avons conçus une application qui répond aux objectifs fixés dans la phase de démarrage.

Nous avons présenté une synthèse sur les différentes typologies des systèmes d'optimisation des requêtes.

Afin d'élaborer ce projet, nous avons commencé le travail par la définition des besoins en faisant la collecte d'information nécessaire à notre étude préalable ensuite nous avons entamé l'étude conceptuelle suivant le modèle en cascade qui traite les différentes phases à savoir la conception détaillée et réalisation pour aboutir à la réalisation de l'application.

Notre projet de fin d'études nous a été d'un grand apport sur plusieurs plans principalement le plan conception, Nous avons appris l'aspect essentiel dans les applications du médiation et principalement le module d'optimisation du requêtes basant sur deux méthodes Data guides et fichier résultat. Sur le plan technique nous avons acquis une certaine expérience dans le domaine de développement par les technologies suivantes : Java, DOM et Stels.

Ce projet est une base de réflexion concrète pour nous, qui peut être améliorée, corrigée ou complétée.

2. Perspectives

La solution qu'on a présenté et le projet 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.

Plusieurs perspectives pour ce travail sont à envisager :

L'intégration de nouveaux type de sources dans se système tel que les sources semi structurée (fichiers textes) et sources objet.