

République Algérienne Démocratique et Populaire  
Ministère de l'enseignement supérieur et de la recherche scientifique

UNIVERSITE SAAD DAHLEB DE BLIDA

USDB

Faculté des Sciences  
Département d'informatique

*Mémoire de fin d'étude pour l'obtention  
D'un diplôme de Master en Informatique  
Option : ingénierie des logiciels*

Thème :

*Intégration sémantique des données  
hétérogènes Structurées (Relationnelles,  
Objet), et semi structurées (XML)*

Présenté par : - *KETTAB ASMA.*  
- *SALHI FOUZIA.*

Soutenu le: 03 / 07 /2011

devant le jury composé de :

M<sup>me</sup> M.FAREH  
M<sup>R</sup> Ferfera  
M<sup>elle</sup> Attaf  
M<sup>elle</sup> Oukid.

Promotrice  
Président  
Examineur  
Examineur

Promotion :  
2010 /2011

MA-004-32-1



## Remerciements

*Nous remercions tout d'abord ALLAH tout puissant  
qui nous a donné la force de mener à bon terme ce modeste  
travail.*

*Nous tenons à remercier sincèrement toutes les personnes qui  
ont apporté leur contribution à l'aboutissement de ce projet.*

*Nous remercions notre promotrice M<sup>me</sup> FAREH pour son  
soutien et ses précieux conseils.*

*Nous remercions aussi tous les enseignants de département  
informatique pour tous les efforts consacrés pour nous  
transmettre le savoir.*

*Nous remercions également les membres de jury,  
M<sup>r</sup> FERFERA. M<sup>elle</sup> ATTAF et M<sup>elle</sup> OUKID pour  
avoir accepté d'évaluer ce travail.*

*Notre reconnaissance s'adresse à toutes les personnes qui sont  
chères et ceux qui nous ont aidé de près ou de loin.*





## *Dédicace*

*Je dédie ce modeste travail à la femme qui a sacrifié sa vie à notre éducation ma très chère mère 'Yassmine'*

*A l'homme qui signifie le mot qui m'encourage et me donne de la force de croire mon très cher père 'Mohamed'.*

*La lumière de mes yeux mes parents.*

*mes chères sœurs 'Houria, Nessrine' et 'Fadila , Ratiba, Zahra' et leurs maris ainsi mon chers et unique frère 'Abd Enour' que j'aime énormément.*

*A mes très chères grandes mères et mon grand père ainsi à la mémoire de mon grand père.*

*A mes tantes et mes oncles qui m'ont soutenu beaucoup. oublier mes amies et toute personne qui m'aime*

**FOUZIA**

## *Dédicace*

*Je dédie ce modeste travail à la femme qui a  
sacrifié sa vie à notre éducation ma très chère  
mère 'Fatima Zohra'*

*A l'homme qui signifie le mot qui  
m'encourage et me donne de la force de  
croire mon très cher père 'Mohamed'.*

*La lumière de mes yeux mes parents.*

*A ma chère sœur 'Sabrina' et mes deux chers frères  
'Imad Eddine' et 'Akram' que j'aime énormément.*

*A mes très chères grands-mères et mon grand-père ainsi  
à la mémoire de mon grand père*

*A mes tantes et mes oncles qui m'ont soutenu beaucoup.*

*Sans oublier mes amies et toute personne qui m'aime.*

**ASMA**

# Sommaire

<b>Introduction Général</b> .....	1
I.1 Contexte Général .....	1
I.2 Problématique .....	2
I.3 Objectif.....	3
I.4 Organisation du mémoire.....	4

## **PARTIE I :Etat de l'art**

### **Chapitre I : Approches d'intégration de données hétérogènes**

Introduction .....	5
I Intégration des données hétérogènes .....	5
I.1 Problématique de l'intégration des données .....	5
I.2 Systèmes d'intégration .....	8
I.3 Classification des systèmes d'intégration .....	11
I.4 Architecture virtuelles VS Architecture matérielle et Comparais entre les SIF .....	15
I.5 Synthèse .....	16
II Système de médiation .....	17
II.1 Définition .....	17
II.2 Schéma global .....	18
II.3 L'adaptateur (wrapper).....	19
II.4 Métadonnée .....	19
II.5 Communication Adaptateur /Médiateur.....	19
II.6 Classification des approches de médiation .....	19
II.7 Processus de médiation .....	20
II.8 Panorama des médiateurs existant .....	21
Conclusion.....	21



## **Chapitre II : Processus de développement**

Introduction.....	21
I Les processus de développement.....	21
II Les processus de développement Unifiés.....	21
III Choix méthodologique.....	23
IV Le processus de développement 2TUP.....	23
Conclusion.....	25

## **Chapitre III : Sources de données traitées et modèle pivot**

Introduction.....	26
I Les sources de données traitées.....	27
I.1 Les bases de données relationnelles.....	27
I.2 Les bases de données objet.....	30
I.1 Les de données semi structurées.....	33
II Le modèle commun.....	37
Conclusion.....	39

## **PARTIE II : Conception et mise en œuvre**

### **Chapitre IV : Etude fonctionnelle**

Introduction.....	40
I. Capture des besoins fonctionnels.....	40
1.1 L'étude préliminaire.....	40
1.2 Identification des cas d'utilisation.....	42
1.3 Les diagrammes des cas d'utilisation.....	43
II. L'analyse.....	50
2.1 Formalisation des scénarios.....	51
2.1.1 diagramme de séquence du cas d'utilisation lancement d'une requête.....	51
2.1.2 diagramme de séquence du cas d'utilisation Création de schéma global.....	52
2.1.3 diagramme de séquence du cas d'utilisation traitement de la requête.....	52
2.1.4 diagramme de séquence du cas d'utilisation Traduction.....	53
2.1.5 Diagramme de séquence du cas d'utilisation Fusion des résultats.....	53
2.2 Diagrammes d'états transition.....	54

## Chapitre V : Etude technique

Introduction.....	56
I Présentation des outils de développement.....	56
I.1 Choix du langage de programmation JAVA.....	56
I.2 Système de gestion de base de données (SGBD).....	56
I.2.1 SGBD Objet .....	56
I.2.2 SGBD Relationnel .....	57
I.2.3 Manipulation des données semi structurées .....	57
I.2.4 JDBC (Java Data Base Connectivity) .....	58

## Chapitre VI : Conception

Introduction .....	59
I Conception du système.....	59
I.1 Diagramme de classe .....	59
I.2 Concevoir les classes et les attributs.....	60
I.3 Concevoir les opérations.....	61
II Architecture du système .....	62
II. 1 Architecture Générale .....	62
II.1.1 Caractéristiques de la solution .....	62
II.1.2 Vue générale .....	63
II.1.3 Description des couches de système .....	64
II.2 Architecture détaillée.....	70
II.2.1 Architecture technique du système .....	70
II.2.2 Description des modules .....	70

## Chapitre VII : Test et implémentation

Introduction.....	77
I Implémentation.....	77
I.1 Environnement de développement .....	77
I.2 Le passage de modèle de conception au modèle relationnel .....	77
II Tests.....	78
II. 1 Fenêtre d'accueil .....	78
II. 2 Fenêtre menu .....	79
II. 3 Fenêtre création schéma global .....	80
II. 4 Fenêtre traitement de requête.....	81
Conclusion.....	82
<b>Conclusion générale</b> .....	83
1. conclusion générale .....	83
2. Perspectives.....	84

<b>Figure VI.5:</b> Schéma local de la base Objet .....	65
<b>Figure VI.6:</b> Fragment de schéma global .....	67
<b>Figure VI.7:</b> Fragment de fichier méta donnée.....	68
<b>Figure VI.8:</b> Architecture technique du système .....	69
<b>Figure VI.9:</b> Génération de schéma global .....	70
<b>Figure VI.10:</b> Exemple génération de schéma global.....	71
<b>Figure VII.1:</b> Exemple de passage d'une classe à une table .....	71
<b>Figure VII.2:</b> Fenêtre d'accueil.....	77
<b>Figure VII .3:</b> Fenêtre menu .....	78
<b>Figure VII .3:</b> Fenêtre Création schéma globale .....	79
<b>Figure VII .3:</b> Fenêtre Traitement de requête .....	78

## Liste des tableaux

<b>Tableau I.1 :</b> Comparaison entre les systèmes d'information .....	14
<b>Tableau I.2 :</b> Architecture virtuelle vs architecture matérielle .....	15
<b>Tableau IV.1 :</b> Les acteurs du système .....	41
<b>Tableau IV.2 :</b> Les messages reçus par le système .....	41
<b>Tableau IV.3 :</b> Les messages émis par le systèmes .....	42
<b>Tableau IV.4 :</b> Identification des cas d'utilisation .....	43
<b>Tableau VI.1 :</b> description des classes .....	59
<b>Tableau VI.2 :</b> description des opérations .....	60



## Résumé

Le nombre croissant de données a mis à jour les difficultés pour retrouver l'information pertinente désirée par un utilisateur, en effet, il pose le problème de l'intégration de sources d'informations préexistantes, ces informations sont représentées et stockées dans une multitude de sources de données et cela de façon très hétérogène.

Notre travail se situe dans le cadre de l'intégration de données hétérogènes (Relationnelle, Objet, XML), en se basant sur l'approche médiateur, afin de faciliter l'interrogation de ces données pour un large public. 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. Le système d'intégration de données que nous proposons est un système à base *médiateur* en utilisant l'approche *GAV* (Global as View).

Les données seront stockées au niveau source, et le médiateur sauvegarde à son niveau leurs descriptions sous format de schéma global, ce dernier sera construit d'une manière automatique avec l'utilisation d'un fichier des métas données exposant l'aspect sémantique des données sources.

**Mots clés :** intégration, médiateur, adaptateur, sources de données hétérogènes, schéma global, requête, sémantique.

## Abstract

The great number of data makes a lot of difficulties to find information desired by users, in effect, it put a problem of integrating existing information.

This information is represented and stored in a multitude of data sources and this is very diverse.

Our work situated on the integration of heterogeneous data (Relational, Object, Xml), based on the mediator approach to facilitate querying data by a wide a public.

A mediator gives the user illusion to querying homogenous system. The system of integration data that we proposed is a system based of mediator and using the approach *GAV* (Global As View).

Data will be stored at the source and a mediator save a description of source, under size a global view, the latter is will be built automatically and use the file of meta data.

**Keywords:** integration, mediator, adapter, heterogeneous data sources, schema, query semantics.

## ملخص

العدد المتزايد من المعلومات قام بزيادة الصعوبات للعثور على المعلومات المتعلقة بالموضوع من قبل المستخدم، و هو في الواقع يثير مشكلة دمج مصادر المعلومات المتوفرة حاليا، يتم تمثيل هذه المعلومات وتخزينها في عدد وافر من مصادر البيانات وهي متنوعة جدا

عملنا هو جزء من تكامل البيانات غير المتجانسة (XML ، Objet ، Relationnel) ، استنادا إلى النهج الوسيط ، لتسهيل الاستعلام عن البيانات من قبل جمهور واسع. الوسيط يعمل بمثابة واجهة بين طلبات المستخدمين ومصادر البيانات. انه يمد المستخدم استعمال سلس ومركزي عن طريق تجنب الاضطراب إلى العثور على مصادر البيانات ذات الصلة لهذا الطلب ، لسؤالهم واحدا تلو الآخر ، والجمع بين المعلومات نفسها التي تم الحصول عليها.

نظام تكامل البيانات الذي نقترحه هو نظام يستند إلى الوسيط باستخدام نهج (Global as View) GAV. وسيتم تخزين البيانات في المصدر ، والعودة إلى الوسيط في شكل أوصاف مستوى النمط العام ويتم تكوينه آليا مع استخدام ملف بيانات التعريف

كلمات رئيسية : التكامل، الوسيط ، غير متجانسة، مصادر البيانات، المخطط ، دلالات الاستعلام.

---

# *Introduction Générale*

---

## *Sommaire*

- 1. Contexte Général*
- 2. Problématique*
- 3. Objectif*
- 4. Organisation du mémoire*



## I. Introduction générale

### I.1 Contexte Général

Quotidiennement, nous faisons appel à différentes sources d'information (journaux, livres, personnes, télévision...) pour obtenir des éléments de réponse à des questions qui nous intéressent ou nous sont utiles. En général, nos recherches d'information aboutissent rapidement car nous savons à quelles sources nous adresser et comment interagir avec elles.

De plus en plus des informations sont stockées sur support informatique sous forme de fichiers, bases de données ou pages web.

Ainsi, avec l'avènement du web et des réseaux informatiques tout comme l'accroissement des données et des services produits font que les utilisateurs finaux se trouvent confrontés à des problèmes d'accès à l'information pertinente qu'ils requièrent. L'hétérogénéité peut provenir du format ou de la structure des sources (Sources structurées : base de données relationnelles, base de données objet, sources semi-structurées : documents XML), la diversité de ces sources de données conduit à des modes de consultation qui peuvent être très différents.

Dans un tel contexte, le besoin d'intégration se fait de plus en plus sentir. Cependant, pour répondre à ce besoin, le développement des applications d'intégration est incontournable avec la répartition des sources et l'hétérogénéité de leurs structures et de l'intégration entre les données en différents formats manipulés.

Afin d'exploiter ces informations pertinentes deux activités de recherche principales ont vu le jour :

- *La recherche d'information* : dans laquelle plusieurs moteurs de recherche ont été développés, comme Google, Yahoo....
- *L'intégration des sources de données* : afin de faciliter leur exploitation, notre travail se focalise sur ce dernier domaine de recherche et traite le cas de l'intégration des bases de données hétérogènes, afin de pouvoir les intégrer dans un système de médiation, ou des systèmes d'aide à la décision.

## I.2 Problématique

La constitution d'un système d'intégration de données est une réponse au problème de l'intégration d'une grande quantité de données variées, relatives à un certain domaine d'application, et stockées physiquement dans différentes sources de données.

Intégrer des sources de données dans le but de fournir aux utilisateurs une interface d'accès uniforme est une tâche difficile.

Cette difficulté concerne les aspects suivants:

- ✓ l'hétérogénéité des données qui comprend :
  - Les conflits syntaxiques
  - Les conflits sémantiques
- ✓ l'autonomie des sources.

De manière générale, la problématique peut se résumer dans l'hétérogénéité de données qu'il est nécessaire de résoudre pour permettre de mettre en correspondance les sources et autoriser l'interrogation et la réponse aux requêtes de façon transparentes.

Dans notre mémoire nous nous intéressons particulièrement à l'hétérogénéité des données ; plusieurs modèles de données existent (Relationnels, Objets, XML ) et chacun possède sa propre syntaxe de modélisation, lorsque le même objet est modélisé dans des modèles différents, les différents schémas obtenus présentent des conflits syntaxiques, ainsi un même objet peut être désigné différemment ce qui présente des conflits sémantiques. L'hétérogénéité sémantique présente un défi majeur dans le processus d'élaboration d'un système d'intégration. Elle est due aux différentes interprétations des objets du monde réel. En effet, les sources de données sont conçues indépendamment, par des concepteurs différents, ayant des objectifs applicatifs différents. Chacun peut donc avoir un point de vue différent sur le même concept.

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

### **I.3 Objectif**

Avec le développement des applications qui partagent des informations provenant de diverses sources de données autonomes et hétérogènes, il est souvent nécessaire pour une telle application d'accéder simultanément à plusieurs sources, du fait qu'elles contiennent des informations pertinentes et complémentaires.

Pour ce faire, la solution des systèmes d'intégration a été proposée. Elle consiste à fournir une interface uniforme et transparente aux données pertinentes via un schéma global. La diversité des sources d'information et leur hétérogénéité est une des principales difficultés rencontrées par les utilisateurs aujourd'hui. Intégrer les sources de données dans le but de fournir aux utilisateurs une interface d'accès uniforme

Comme l'hétérogénéité syntaxique et sémantique sont des aspects importants dans l'intégration des sources de données, l'objectif est donc d'étudier les différents systèmes d'intégration afin d'utiliser le plus adapté à nos exigences, de proposer son architecture pour pouvoir répondre aux besoins des utilisateurs en cachant l'hétérogénéité des sources.

Donc notre objectif est de concevoir et de réaliser un système d'intégration de données hétérogènes, structurées présentés par le modèle relationnel et le modèle objet, et semi structuré présenté par le modèle XML. Pour permettre à un simple utilisateur de trouver des réponses à ses requêtes en lui cachant l'hétérogénéité syntaxique et sémantique de ces sources.



## I.4 Organisation du mémoire

Le présent mémoire est structuré comme suit :

- ❖ La première partie est composée de trois chapitres
  - Nous présentons dans le premier chapitre les différentes approches d'intégration de données, ainsi une étude sur le système de médiation.
  - Dans le deuxième chapitre nous présentons un bref aperçu sur le processus de développement adopté.
  - Nous présentons dans le troisième chapitre les sources de données traitées, et le modèle pivot.
  
- ❖ La deuxième partie composée de quatre chapitres
  - Dans le quatrième chapitre nous présentons une étude fonctionnelle du système que nous allons réaliser.
  - Dans le cinquième chapitre nous présentons une étude technique de notre système.
  - Le sixième chapitre présente la conception de notre solution.
  - Et le septième chapitre est consacré à présenter notre prototype testé sur des applications différentes pour le domaine médicale.
  
- ❖ Conclusion générale

---

# *Partie I*

# *Etat de l'art*

---

## *Sommaire*

- 1. Approches d'intégration de données*
- 2. Sources de données traitées et modèle pivot*
- 3. Processus de développement*

---

# *CHAPITRE I*

Approches d'intégration de  
sources de données hétérogènes

---



## Introduction

Un système d'intégration de données fournit une vue unifiée de données provenant de sources multiples et hétérogènes qui permettent aux utilisateurs d'accéder, à travers un schéma global unifié, à plusieurs sources de données ayant chacune un schéma local.

Bien que les systèmes actuels puissent surmonter la difficulté principale d'intégration qui est l'hétérogénéité des sources, leur mise en œuvre pose un certain nombre de problèmes, en ce qui concerne la génération des liens sémantiques.

Il existe principalement deux approches d'intégration des données, l'approche entrepôt de données qui matérialise les vues et l'approche de médiation qui ne construit que des vues virtuelles des données et les interroge là où elles se trouvent.

### I. Intégration des données hétérogènes

L'hétérogénéité est non seulement due aux différents formats de structuration des systèmes d'information mais également aux multiples interprétations que des systèmes autonomes peuvent avoir de la même donnée. Ainsi, l'intégration de sources de données hétérogènes, autonomes et réparties passe par la résolution de ces conflits sémantiques et différences syntaxiques.

#### I.1 Problématique de l'intégration des données

Garantir la conformité mutuelle des données hétérogènes et leur conformité par rapport au schéma global est cruciale dans un système multi-source en général. Il est difficile d'envisager l'intégration de données sans tenir compte des conflits liés à l'hétérogénéité des sources.

Les conflits sont présents lorsque les sources contiennent des erreurs et des incohérences (erreurs de saisie, redondances, violation de contraintes d'intégrité, valeurs contradictoires,...). Les problèmes se multiplient lorsqu'il s'agit d'intégrer des données depuis plusieurs sources dans un système d'informations global, un système de bases de données fédérées ou encore dans un entrepôt de données.

En effet, en plus des données erronées que peut contenir chaque source individuellement, le problème de correspondance entre objets doit être pris en compte. En d'autres termes lorsque des données proviennent d'origines diverses, et saisies à des moments distincts par plusieurs personnes qui utilisent des conventions différentes, la question capitale est de savoir quelles sont les données qui correspondent à la même entité du monde réel ?

Par exemple une société peut avoir des informations sur ses clients stockées dans des tables différentes, car chaque client peut acheter des produits dans plusieurs départements.

Une fois la décision de centraliser l'information sur tous les clients dans une même source (ex : dans un entrepôt de données), un même client peut être représenté différemment dans les sources : « John Smith », « Smith John », « J.Smith ».

La constitution d'un système multi-source est une réponse au problème de l'intégration d'une grande quantité de données variées, relatives à un certain domaine d'application, et stockées physiquement dans différentes sources de données. Un système multi-source est régulièrement alimenté et mis à jour via des extractions depuis des sources et la saisie de données quotidiennement, ce qui augmente la probabilité de présence de données erronées.

Un système global étant une solution pour faire face aux besoins des entreprises en termes de capitalisation de connaissances et d'aide à la décision, il est primordial que les données stockées soient correctes et fiables. Ces systèmes ont alors besoin d'un support de nettoyage de données très avancé qui permettra de détecter et de résoudre les conflits présents dans les sources et de mettre en conformité les données sources avant de les intégrer et de les délivrer à l'utilisateur.

Les premières approches traditionnelles se sont focalisées sur les conflits d'intégration de schéma appelés conflits sémantiques liés au schéma qui découlent de l'utilisation d'une terminologie différente pour décrire le même concept. Par exemple dans la relation de médiation **produit** (Num-produit, désignation, *prix*) et dans la relation source **produit** (Num-produit, désignation, *prix-produit*), les attributs *prix* et *prix-produit* ont deux terminologies différentes mais une sémantique identique.

En conclusion l'hétérogénéité se situe à deux niveaux : **syntactique et sémantique**.

**a. L'hétérogénéité syntactique :** se trouve dans les formats de stockage des données (XML, Relationnel, Objet, ... etc.) dans les langages d'interrogation ( XQUERY, SQL, OQL, etc.) sera différents.

**b. L'hétérogénéité sémantique :** représente les différences entre les interprétations du monde réel selon le contexte et l'utilisation des données. Ceci a généralement lieu durant le processus de conception des systèmes d'information. Les conflits sémantiques peuvent survenir au niveau des schémas et des données .



Les conflits au niveau des données résultent de l'utilisation de domaines de données différents selon le type d'application (e-gouvernement, e-commerce, bioinformatique, etc.). En effet, des données similaires peuvent être représentées et interprétées différemment dans chaque domaine.

Les conflits des schémas sont, quant à eux, caractérisés par des différences dans les structures logiques ou méta données.

Dans ce contexte, (Bressan et al, 1999) a identifié quatre principaux types de conflit : conflits de nom, conflits d'échelle, conflits d'indéterminisme (*confounding conflicts*) et conflits de représentation.

- Les conflits de nom sont liés aux différences dans la désignation de concepts. Le cas le plus fréquent est la présence de **synonymes**, d'**homonymes**, d'**hyperonymes** et d'**hyponymes** (Mena, Illarramendi et al., 1996). Les **synonymes** sont deux mots distincts ayant le même sens.

C'est l'exemple de "publication" et "article" qui capturent la même information sur les articles de recherche publiés. Les **homonymes** sont des mots partageant la même graphie et la même prononciation mais n'ayant pas le même sens.

Par exemple, "mémoire" peut faire référence à des entités différentes: "mémoire informatique" et "mémoire de thèse". Les **hyperonymes** et les **hyponymes** indiquent des niveaux d'une hiérarchie désignés par le concept plus "général" ou le concept moins "général". C'est le cas de "personne" qui est un hyperonyme de "employé" car c'est un terme plus "général".

- Les conflits d'échelle ont lieu quand des systèmes de référence différents sont utilisés pour mesurer une grandeur. C'est l'exemple de "Fahrenheit" ou "Celsius" pour la température.

- Les conflits d'indéterminisme surgissent quand les concepts semblent avoir le même sens alors qu'ils sont différents. Ceci peut être dû à des contextes temporels différents.

- Les conflits de représentation surviennent quand les schémas de deux sources décrivent différemment un même concept. Par exemple, le nom d'un étudiant peut être représenté par deux champs "prénom" et "nom" dans une source et par un seul champ "identité" dans une autre source. Nous pouvons également citer l'exemple d'un concept défini comme une classe dans une source de données et comme attribut dans une autre, etc...



*Masquer l'hétérogénéité sémantique revient à faire communiquer les systèmes d'information via une connaissance commune qui permet d'explicitier et de préciser le sens des données pour être interprétées correctement par différents systèmes. Cette connaissance peut être capturée grâce à des ontologies formelles.*

*Et masquer l'hétérogénéité syntaxique revient à faire la traduction des langages de données en un langage commun.*

## **I.2 Systèmes d'intégration**

L'intégration de données consiste à éliminer d'abord les conflits entre les données et ensuite les représenter dans un seul schéma cohérent.

Tout système d'intégration doit fournir les solutions aux problèmes suivants :

- l'identification et la spécification des correspondances entre des données sémantiquement liées,
- fournir une vue globale intégrée des données représentées à travers des conceptualisations différentes,
- la gestion des mises à jour des données de différentes sources et leurs répercussions sur le schéma global.

Les systèmes d'intégration de données offrent des architectures d'interopérabilité (coopération des systèmes d'information) sur une fédération de sources de données distribuées, autonomes et hétérogènes. Les entrepôts de données, les systèmes de médiation et les architectures P2P sont des exemples d'infrastructures qui permettent l'intégration de données, c'est-à-dire l'accès à des données produites par des sources autonomes.

A travers des schémas virtuels, des métadonnées et des correspondances sémantiques, 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. L'hétérogénéité des données extraites des sources nécessite leur réconciliation, en d'autres termes, leur mise en correspondance par rapport au schéma global, avant de les présenter à l'utilisateur.

### I.2.1 Définition et composants

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 une interface uniforme, sans se soucier de leur structure ni de leur localisation.

Formellement, un système d'intégration de données est un triplet  $I : \langle G, S, M \rangle$ , où :  $G$  représente le schéma global modélisant le schéma intégré,  $S$  est l'ensemble des schémas des sources 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, Pour interroger le système intégré, les requêtes sont exprimées en termes de constructions du schéma global  $G$  [BOUSSIS 08].

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

- le niveau schéma, qui consiste à consolider tous les schémas des sources en un seul schéma global, ou schéma de médiation, qui sera utilisé pour supporter les requêtes.
- le niveau données (population du schéma intégré).

Un système d'intégration se compose de deux parties [RAH 05] (voir figure 1.1) :

- Une partie correspond aux utilisateurs du système intégré (décideurs) ou autres système.
- Une deuxième partie interne et comprend des sources d'informations et une interface uniforme qui permet a la partie externe d'interroger d'une manière transparente les sources de données. comme s'il n'y avait qu'une source unique.

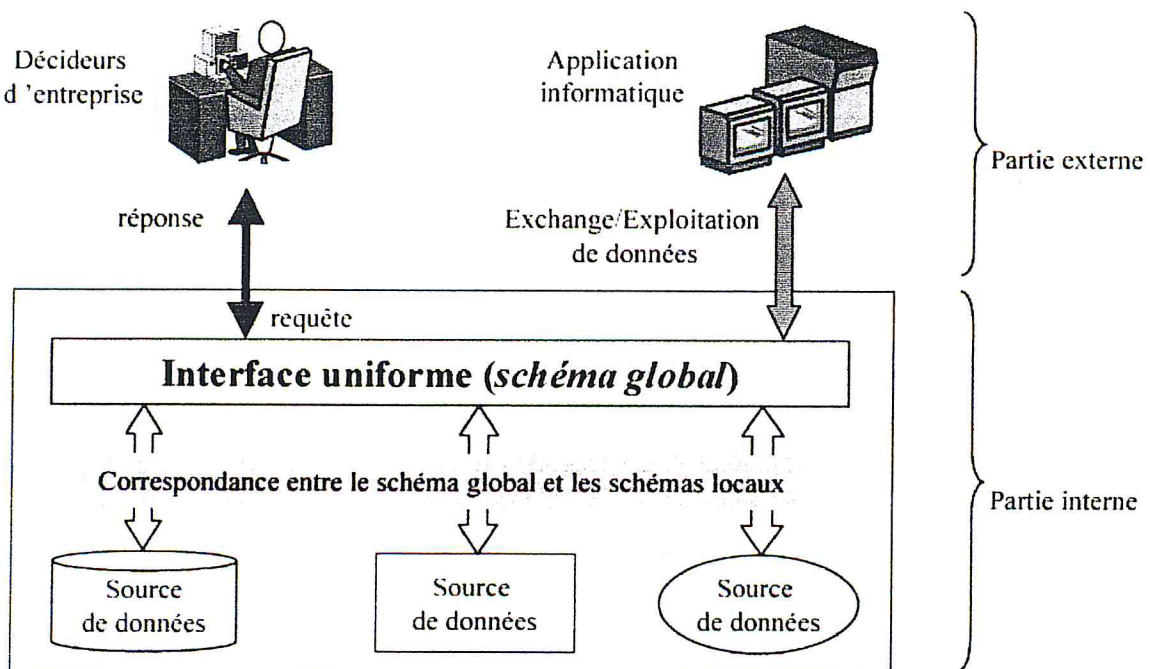


Figure I.1 Système d'intégration de données



### I.2.2 Processus d'intégration

La plupart des approches d'intégration décomposent le processus d'intégration de données en plusieurs phases dont les principales sont [JOU 99]:

- **La pré intégration** : 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 (Comparaison)** : lorsque les schémas initiaux ont atteint le niveau de conformité souhaité, l'étape suivante consiste à identifier les éléments communs des bases existantes.

Parmi les travaux dédiés à cette phase il existe les approches suivantes :

- Calcul d'une fonction de similitude entre paires d'éléments [YU91].
- Méthodes manuelles : les paires de noms sont proposées à l'expert de la base qui doit déterminer lui-même les correspondances à retenir.

- **Fusion** : Une fois effectué le travail de description sémantique des schémas (Pré-intégration), et de recherche de correspondance entre leurs éléments (Comparaison), le processus d'intégration proprement dit peut être effectué. Il consiste de manière semi-automatique en général, à construire un schéma intégré à partir :

- Des schémas sources.
- Des correspondances entre schéma (déterminer lors de la comparaison).
- De règles d'intégration (propre à la méthode et au modèle de données).

- **Restructuration** : cette phase se retrouve dans les cas d'intégration de vues ou lorsque la modification ultérieure des schémas sources est admise, les approches récentes de type fédérées n'admettent pas de telles modifications, afin de préserver l'environnement local des bases composants la fédération et de garantir leur autonomie.



### **I.3 Classification des systèmes d'intégration**

Actuellement de nombreux systèmes communiquent entre eux en échangeant des informations. La plupart de ces systèmes sont hétérogènes, d'où la difficulté de partager les informations. Parmi les systèmes existantes il y'a :

#### **I.3.1 Systèmes multibases**

Les systèmes Multibases sont des systèmes dits faiblement couplés [BUS 99]. On les caractérise de cette manière car ils n'offrent pas une vision unifiée des données. Il n'existe pas de schéma global permettant un accès transparent aux différentes sources de données. La coopération est seulement assurée par l'intermédiaire d'un langage commun : le langage multibase de type SQL notamment [LIT 89].

L'utilisateur peut poser une requête aux différents systèmes à l'aide de ce langage commun, sans se soucier de l'hétérogénéité des systèmes sources. Ceci ne signifie pas qu'une requête nécessitant l'accès à diverses sources est exécutée en une seule fois. L'utilisateur doit envoyer autant de requêtes qu'il y a de sources impliquées, c'est donc à l'utilisateur de relier les différentes réponses aux requêtes formulées.

#### **I.3.2 Systèmes fédérés**

A l'inverse des systèmes Multibases, les systèmes fédérés sont dits fortement couplés [BUS 99]. Ils se caractérisent par l'existence d'un schéma unifié appelé schéma fédéré qui constitue l'interface d'accès au système intégré. L'intégration se situe au niveau des schémas. La conception de ce schéma fédéré suppose d'unifier les schémas source et de traiter leur hétérogénéité. Il est nécessaire d'identifier les correspondances et de résoudre les conflits entre les éléments des schémas. Ces correspondances peuvent être exprimées à l'aide de différents langages, au moyen de règles ou à travers une ontologie. Il y a donc cette fois une vision unifiée des sources. L'intégration offre un accès commun aux sources et une représentation commune.

Les bases de données fédérées utilisent la stratégie ascendante, le processus de développement ascendant est utilisé pour faire coopérer des bases de données existantes, le schéma global est obtenu à partir des schémas locaux, c'est le cas le plus courant mais le plus compliqué car il nécessite de gérer l'existant qui est généralement hétérogènes.

### I.3.3 Approche médiateur

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 à des bases de données. La notion de médiateur a été initialement proposée par [WIE 92]. IL définit un médiateur comme suit: "A mediator is a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications".

Un médiateur doit être vu comme une couche logicielle permettant d'accéder de manière transparente pour l'utilisateur à différentes ressources (Bases de données, fichiers) réparties et hétérogènes. Pour cet accès, le médiateur exploite des connaissances (métadonnées) qui sont utiles à différents services (interrogation, localisation des ressources notamment).

Les systèmes de médiation sont assez proches des systèmes fédérés. Ce qui les distingue concerne notamment 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) et leur capacité d'évolution (en générale plus grande pour la médiation car la conception suit souvent une approche descendante, nous y reviendrons) [BUS 99].

L'approche par médiation est fondée sur la définition de vues [ROU 02]. Les données ne sont pas stockées dans le système de médiation mais résident dans leur source d'origine (comme pour les systèmes fédérés). L'utilisateur a une vision unifiée des données sources : l'interrogation se fait par l'intermédiaire d'un schéma global. Il n'a pas connaissance des schémas locaux.

L'architecture générale d'un système de médiation est présentée en figure 2.1. Une requête globale est posée via le schéma global et celle-ci est ensuite décomposée en sous requêtes, traduites pour être exécutées sur les différentes sources concernées. Le médiateur est chargé de localiser les données pertinentes pour répondre à la requête (en utilisant les métadonnées).

L'interrogation effective des sources se fait par des adaptateurs (ou « wrappers ») qui constituent une interface d'accès aux différentes sources. Ces adaptateurs traduisent les sous requêtes exprimées dans le langage de requête spécifique de chaque source. Les résultats sont ensuite renvoyés au médiateur qui se charge de les intégrer avant de les présenter à l'utilisateur. Par analogie à l'architecture des systèmes fédérés, on peut considérer que le schéma global du médiateur correspond au schéma fédéré et que l'adaptateur inclut les schémas d'export et les schémas pivots.



### I.3.4 Entrepôt de données

L'approche entrepôt de données consiste à voir l'intégration comme la construction de base de données réelles appelées entrepôt regroupant les informations pertinentes pour les applications considérées, l'utilisateur lance un traitement directement sur les données stockés dans l'entrepôt c'est une approche dite matérielle.

Un entrepôt de données est l'espace de stockage centralisé d'un extrait de sources de données pertinentes pour les décideurs. Son organisation doit faciliter la gestion des données sous la forme d'une vision unifiée et doit permettre la conservation des évolutions nécessaires pour les prises de décisions.

Un entrepôt de données offre une vision uniforme des données qui seront extraites en fonction de besoins d'analyse pour alimenter les magasins de données.

Bill Inmon est l'un des premiers à avoir employé le terme entrepôt de données, ce dernier le définit comme suit : « une collection de données orienté sujet, intégrées, variant selon le temps et non volatiles, qui sert de support au processus de prise de décision des acteurs de managements (les décideurs) ou autre système.

**Orientées sujets** : les données collectées doivent être orientées « métier » et donc triées par thèmes.

**Intégré** : les données sont centralisées dans un entrepôt à partir d'un ensemble de sources de données variées, les données sont fusionnées et agencées dans une vision cohérente.

**Variant selon le temps** : toutes les données d'un entrepôt sont identifiées par des périodes temporelles spécifiques, on parle d'historisation de données.

**Non volatiles** : les données d'un entrepôt sont stables, il est possible d'ajouter des données mais on ne modifie pas les données déjà intégrés.

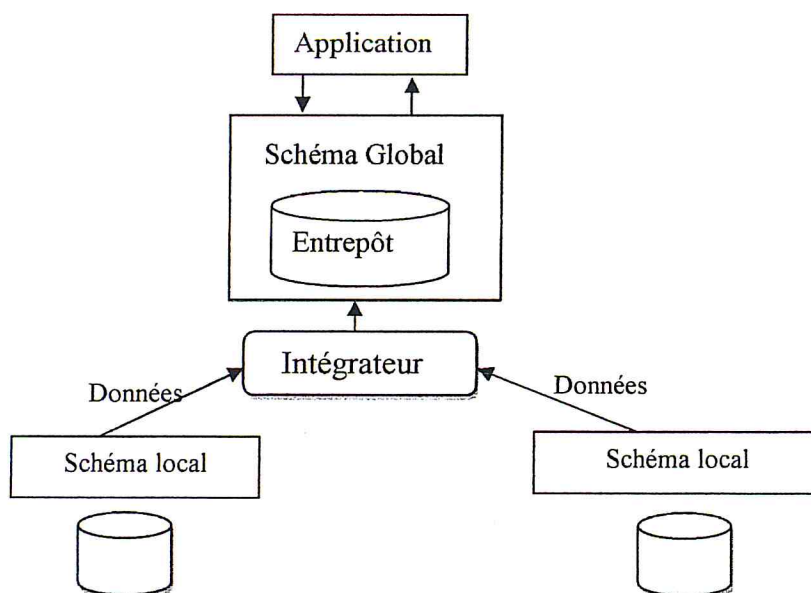


Figure I.2 Architecture « Entrepôt »



### a. Etape d'intégration

Nous distinguons deux niveaux dans la construction des entrepôts de données. Le premier niveau correspond à la <sup>9e</sup> <sup>11e</sup> <sup>Chargés</sup> construction des sources de données opérationnelles, et de l'entrepôt de données global. Le second niveau englobe tous les entrepôts de données locaux. La raison de cette distinction est, qu'à chaque niveau, sont associées différentes étapes de traitement et différentes difficultés techniques. Au premier niveau, le processus de construction est décomposé en quatre étapes principales, qui sont : (1) l'extraction des données des sources de données opérationnelles, (2) la transformation des données aux niveaux structurel et sémantique, (3) l'intégration des données, et (4) le stockage des données intégrées dans le système cible.

On parle de processus ETL (*Extract, Transform, Loading*). L'étape d'extraction des données se fait généralement directement sur les systèmes de production, dans les bases de données en créant un fichier de données. Ce fichier sera par la suite téléchargé sur un deuxième système qui se chargera du reste des manipulations de données. Les transformations sont utilisées pour nettoyer les données et pour créer les clés qui serviront dans l'entrepôt. Il peut arriver que des données concernant la même entité (personne, entreprise, etc.) soient présentes dans différents systèmes, mais qu'il n'y ait pas de façon de joindre ces données automatiquement.

Par exemple, un des systèmes peut identifier des personnes avec leur numéro d'assurance sociale, et un autre avec un code arbitraire à partir de leur nom et de leur date de naissance. Afin de pouvoir joindre les données et de les insérer dans l'entrepôt, il est nécessaire de créer des clés supplémentaires pour chaque enregistrement (comme un nombre), ce qui permet de les identifier uniquement et de les joindre correctement. Ces clés substituts deviennent alors les clés utilisées pour les jointures dans l'entrepôt. Il existe des logiciels qui sont spécialisés dans le processus ETL d'un entrepôt de données.

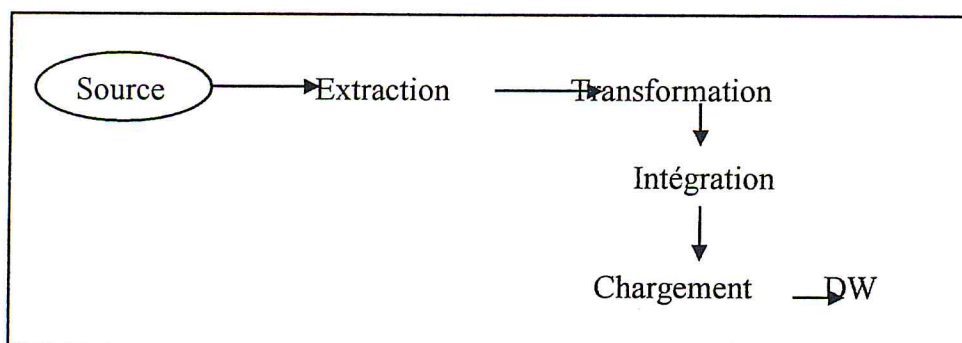


Figure I.3 Etapes d'alimentation de l'entrepôt.

**b. Représentation conceptuel d'entrepôt de données :**

Souvent représentés par une structure à plusieurs dimensions On parle de modèle multidimensionnel, souvent représenté sous forme de cube, parce que les données seront toujours des faits à analyser suivant plusieurs dimensions.

- On entend par **dimensions** les axes avec lesquels on veut faire de l'analyse, par exemple il ya une dimension client, dimension produit, une dimension est tout ce qu'on utilisera pour faire nos analyse.

- Les **faits** sont des tables qui contiennent des informations opérationnelles, le fait modélise le sujet d'analyse, il est formé de mesures correspondant aux fait de l'analyse.

L'implantation classique consiste à considérer *un modèle en étoile* avec au centre la table des faits et les dimensions comme étant de branches à l'étoile. Les branches de l'étoile sont des relations de 1 à plusieurs, la table des faits est énorme contrairement aux tables des dimensions.

**I.3.5 Médiateur/ Entrepôt de données**

Avec le développement du Web et des technologies de l'information ces dernières années, d'autres approches d'intégration, tels que les systèmes hybrides (Approche mixte), ont été proposés. Ces systèmes combinent, à la fois, l'approche Médiateur et l'approche Entrepôt. Il s'agit, par exemple, d'un système médiateur qui intègre plusieurs sources de données externes et qui exploite un entrepôt de données contenant des données conformes au schéma global du médiateur.

**I.4 Architecture virtuelles VS Architecture matérielle et Comparaison entre les Système d'information**

	<b>Système multibase</b>	<b>Base de données fédérés</b>	<b>Système d'information basé sur le mediateur</b>
Type d'hétérogénéité concernée	Technologique	toutes	toutes
Type de composants	Structuré	Structuré	N'importe
Méthodes d'accès	Langage de requête	Langage de requête	N'importe
Virtuelle ou matérialisé	Virtuelle	Virtuelle	Virtuelle
Ascendantes ou descendantes	/	Ascendantes	descendantes x Ascendant

**Tableau I.1** Comparaison entre les Systèmes d'information



	<b>Approche virtuelle (médiateur, base fédérée, multi-base)</b>	<b>Approche matérielle (entrepôt de données)</b>
Localisation des données	Les données restent dans leurs sources.	Les données provenant de sources différentes sont intégrées dans une seule base.
Mises à jour des Sources	Les sources sont mises à jour fréquemment	mise à jour périodique des sources
Traitement des requêtes	Décomposition des requêtes en sous requête adaptée pour chacune des sources qui peut être couteux.	Application directe de la requête sur l'entrepôt
Historique	pas d'historisation des données	Les données doivent être historisées
Prédiction des requêtes	Impossible de prédire les requêtes d'utilisateur.	Prédiction possible des requêtes d'utilisateur
Performance (Exploit)	Défi principal	Bonne performance

**Tableau I.2** Architecture virtuelle Vs Architecture matérielle

### 1.5 Synthèse

Les différentes approches présentées ont des soucis de prendre en compte les différents types d'hétérogénéité, Selon le domaine d'application les données à intégrer sont de natures différentes, les systèmes basés sur le médiateur peuvent faire face à n'importe quel type de données (structurées, semi-structurées et non structurées), contrairement aux autres systèmes, et évite la réplication des données contrairement à l'approche entrepôt de données, ce qui privilégie cette approche dans le développement d'application de recherche d'information.

Donc notre choix s'est porté sur l'approche médiateur, qui est l'objet de ce qui suit.



## II. Système de médiation

### II.1 Définition

Un *système de médiation* est un outil puissant permettant un accès simple aux différentes informations collectées de sources de données pouvant être très divers. Il doit intégrer des données diverses afin de pouvoir offrir à l'utilisateur une vue centralisée et uniforme des données en masquant les caractéristiques spécifiques à leur localisation, méthode d'accès et formats [GAR 06].

L'approche médiateur [GIO 1992] consiste à définir une interface entre l'agent (humain ou logiciel) qui pose une requête et l'ensemble des sources accessibles via le Web potentiellement pertinentes pour y répondre. L'objectif est de donner l'impression d'interroger un système centralisé et homogène; alors que les sources interrogées sont réparties, autonomes et hétérogènes.

L'approche médiateur (Figure 2.1) présente l'intérêt de pouvoir construire un système d'interrogation de sources de données sans toucher aux données qui restent stockées dans leurs sources d'origine. Ainsi, le médiateur ne peut pas évaluer directement les requêtes qui lui sont posées car il ne contient pas de données, ces dernières étant stockées de façon distribuée dans des sources indépendantes. L'interrogation effective des sources se fait via des adaptateurs (wrappers), qui traduisent les requêtes réécrites en termes de vues dans le langage de requêtes spécifique accepté par chaque source [MOHAND 2004].

Globalement un médiateur offre un accès direct aux sources et se caractérise par

[VODISLAV 2003]:

- approche "paresseuse", pas de matérialisation.
- migration de requêtes vers les sources.
- *avantages*: cohérence, données réelles ← *naïf*
- *inconvénients*: performances, traduction de requêtes, capacités des sources.

## ❖ Architecture de médiation

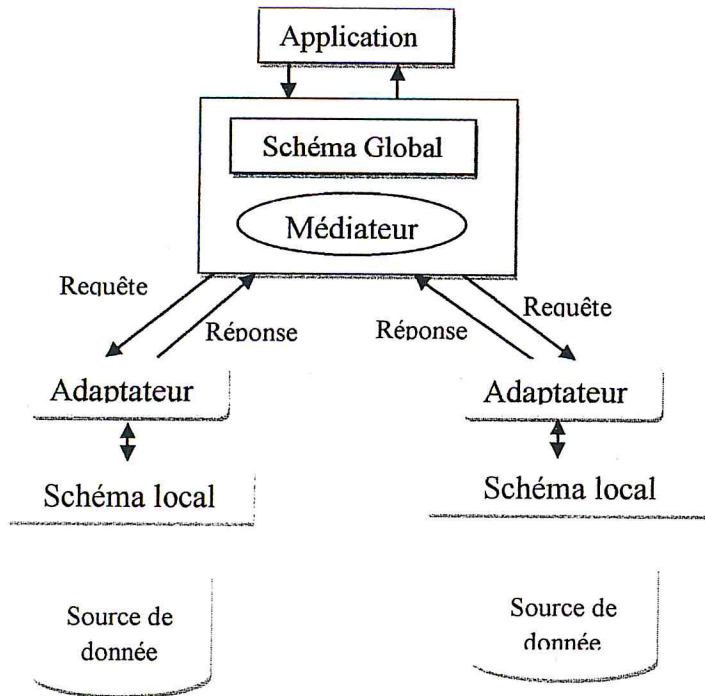


Figure I.4 Architecture de médiation

## II.2 Schéma global

Un médiateur comprend un schéma global ou ontologie, dont le rôle est central. C'est un modèle du domaine d'application du système. L'ontologie fournit un vocabulaire structuré servant de support à l'expression des requêtes. Par ailleurs, elle établit une connexion entre les différentes sources accessibles. En effet, dans cette approche, l'intégration d'informations est fondée sur l'exploitation de vues abstraites décrivant de façon homogène et uniforme le contenu des sources d'informations dans les termes de l'ontologie.

Les sources d'informations pertinentes, pour répondre à une requête, sont calculées par réécriture de la requête en termes de ces vues. Le problème consiste à trouver une requête qui, selon le choix de conception du médiateur, est équivalente ou implique logiquement, la requête de l'utilisateur mais n'utilise que des vues. Les réponses à la requête posée sont ensuite obtenues en évaluant les réécritures de cette requête sur les extensions des vues.

### II.3 L'Adaptateur (Wrapper)

Les adaptateurs sont généralisés au maximum de données fournies par la source correspondante, ce qui limite leur modification à l'extension des données de la source. Ces modules spécifiques aux différentes sources effectuent le lien et le requêtage auprès des sources, ainsi que l'analyse des réponses obtenues.

L'interrogation et la réponse d'une source donnée s'effectuent dans son propre format, l'adaptateur converti la requête dans le format de la source, et la réponse dans un format correspondant au schéma de médiation du système. C'est cette réponse au format global qui est ensuite retransmise au médiateur, pour y être regroupée avec les autres réponses des autres adaptateurs.

L'adaptateur(Wrapper) s'occupe de l'hétérogénéité des sources, c'est un traducteur, il fait la traduction du langage de requête commun en langage de requête natif (propre a la source), et la traduction des résultats natifs en résultat au format commun [GAR 06].

### II.4 Métadonnées

Les métadonnées sont des données permettant de décrire d'autres données ou des composants du système, ce concept à pour but de résoudre certains problèmes liés a l'hétérogénéité sémantique.

### II.5 Communication Adaptateur/Médiateur

Pour faciliter le travail d'intégration, on définit un langage commun dans lequel le médiateur interrogera les adaptateurs, et un format de résultat commun dans lequel les adaptateurs répondront au médiateur [GAR 06].

### II.6 Classification des approches de médiation

Les approches de médiation sont classifiées selon le type de leur schéma médiateur en deux groupes : les approches GAV (Global As View) ou le schéma médiateur est une vue sur des schémas locaux, et les approches LAV (Local As View), ou les schémas locaux sont des vues du schéma global [Boum 05].



- **GAV** : le Schéma global est défini comme une vue intégrante sur schémas locaux, C'est une Approche ascendante depuis les sources vers le médiateur et se caractérise par la difficulté pour ajouter une source.
- **LAV** : Chaque source locale est définie comme une vue locale du schéma global, C'est une Approche descendante depuis le médiateur vers les sources. [GAR 06]

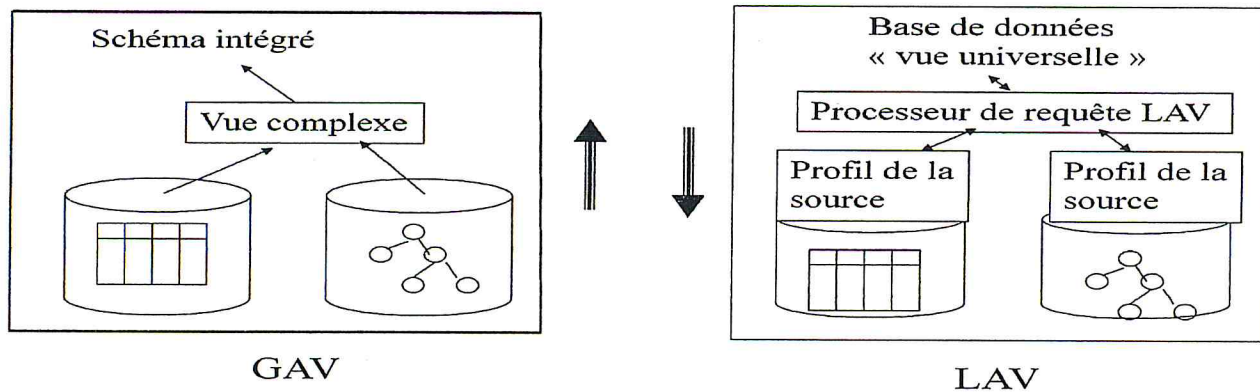


Figure I.5 Décomposition versus Recomposition

## II.7 Processus de Médiation

Mapping est la correspondance entre le schéma global et les schémas des sources, il est utilisé pour la traduction des requêtes et la structuration des résultats.

On distingue les phases suivantes :

- Analyse syntaxique et sémantique de requêtes.
- Décomposition de la requête globale en sous requêtes locales.
- Génération d'un plan optimisé pour l'exécution de la requête.
- Exécution des sous requêtes sur les différentes sources
  - transformation de la requête du langage commun vers le langage de la source appropriée.
  - transformation du résultat du format de la source vers le format commun.
- Récupération des résultats intermédiaires et recombinaison du résultat final
- Combinaison des résultats locaux.
- Requête de recombinaison sur système global.

## II.8 Panorama des médiateurs existant

- Génération relationnelle (1975-1990)
  - Souvent centré sur un SGBD qui joue le rôle d'un médiateur
  - Exemple : Multibase.
- Génération relationnelle étendue (1990-2000)
  - Fédère des BD hétérogènes autour de SQL3
  - Exemple : médiateur Objet.
- Génération XML XQuery (>2000) : utilise le standard XML
  - OLE-DB.NET (Microsoft).

### Conclusion

Nous avons étudié dans ce présent chapitre les différentes approches d'intégration. Ainsi les Systèmes de médiation, A priori, Concevoir un système dans cette même logique est l'objectif de notre travail. Dans le suivant chapitre nous allons présenter la méthode adoptée pour la réalisation de notre système.

---

*CHAPITRE II*

Processus de développement

---



## Introduction

La conception et la réalisation d'un projet s'appuie sur une approche ou démarche à suivre, qui doit être choisie minutieusement, afin d'assurer le succès du projet. Le choix est très difficile car il existe plusieurs approches, et la décision d'adopter telle ou telle approches doit se faire selon plusieurs critères concernant le projet.

### I. Les processus de développement

- Un processus définit une séquence d'étapes, en partie ordonnées, qui concourent à l'obtention d'un système logiciel ou a l'évolution d'un système existant.
- L'objet d'un processus de développement est de produire des logiciels de qualités qui émondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles. En conséquence, le processus peut se décomposer suivant deux axes de contrôle sur le développement :
  - L'axe de développement technique, qui se concentre principalement sur la qualité de la production ;
  - L'axe de gestion du développement, qui permet la mesure et la prévision des coûts et des délais.

### II. Les processus de développement unifiés

#### II.1 les principes fondamentaux des Processus Unifiés(UP)

Le Processus Unifié est un processus de développement logiciel construit sur UML « générique, itératif et incrémental, centré sur l'architecture, conduit par les cas d'utilisation et piloté par les risques ». Il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel.

- **Itératif et incrémental** : le projet est décomposé en itérations de courte durée qui aident a mieux suivre l'avancement global.  
A la fin de chaque itération, une partie exécutable du système final est produite d'une façon incrémentale.
- **Centré sur l'architecture** : tout système complexe doit être décomposé en parties modulaires afin de garantir une maintenance et une évolution facile. Cette architecture ( fonctionnelle, logique, matérielle, etc...) doit être modélisée en UML et pas seulement documentée en texte.

- **Piloté par les risques** : les risques majeurs du projet doivent être identifiés au plus tôt, mais surtout levés le plus rapidement possible. Les mesures à prendre dans ce cadre déterminent l'ordre des itérations.
- **Conduit par les cas d'utilisation** : le projet est mené en tenant compte des besoins et des exigences des utilisateurs. Les cas d'utilisation du futur système sont identifiés, décrits avec précision.

## II.2 Activités et phases

L'objectif d'un processus unifié est de maîtriser la complexité des projets informatiques en diminuant les risques.

UP est un ensemble de principes générique adapté en fonctions des spécificités des projets.

UP gère le développement selon deux axes :

- **L'axe vertical** représente les principaux enchainements d'activités, qui regroupent les activités selon leur nature. Cette dimension rend compte l'aspect statique du processus qui s'exprime en termes de composants, de processus, d'activités.
- **L'axe horizontal** représente le temps et montre le déroulement du cycle de vie du processus ; cette dimension rend compte de l'aspect dynamique du processus qui s'exprime en terme de cycles, de phases, d'itérations.

## II.3 Les meilleurs pratiques du processus Unifié

- Traitement des questions très importantes dans les premières itérations.
- La définition des besoins.
- Construction du noyau architectural cohérent dès premières itérations.
- Application des cas d'utilisation de manière approprié.
- Modélisation graphique du logiciel(UML).
- Gestion rigoureuse des spécifications.
- Pratique des demandes de changement et de la gestion de configuration

Les adaptations d'UP les plus connues sont :

- RUP : Rational Unified Process
- XP : eXtreme Programming
- 2TUP : Two Tracks Unified Process

---

# *CHAPITRE III*

Sources de données  
traitées et modèle pivot

---



## Introduction

Depuis toujours, trouver un meilleur moyen pour stocker les informations sur un support matériel pour leurs réutilisations futures, était un vrai souci, pour répondre à trois besoins, non exclusifs les uns des autres :

- Conserver l'information en lieu sûr pour répondre à une contrainte légale ou conventionnelle (archivage des données).
- Rendre l'information disponible.
- Réutiliser l'information (traitement des données).

Pour répondre à ce problème beaucoup de recherche ont été faites et qui ont conduit à l'apparition de plusieurs formats de données, les données structurées tel que les bases de données relationnelles, les bases de données Objet, les données semi structurées, les données non structurées... [ALIMA 09]

Ces différentes capacités sont prises en charge par des logiciels appelés système de gestion de base de données (SGBD). Les principaux produits commerciaux sont Oracle, DB2 (IBM) et SQL Server (Microsoft). Plusieurs SGBD gratuits comme MySQL ou PostgreSQL sont de plus en plus utilisés pour la gestion de bases petites et moyennes, et particulièrement adaptés pour s'interfacer avec des pages Web.

Tout SGBD est conçu au tour d'un modèle de données bien défini. Il s'agit de la combinaison de trois éléments :

- Une structure, qui correspond à l'organisation logique des données, la forme sous laquelle les utilisateurs vont les percevoir ou les représenter.
- Des contraintes d'intégrité, que l'on peut définir sur les données, afin d'en assurer l'intégrité et la cohérence avec le monde réel et les besoins des applications.
- Des langages de manipulation des données, pour les mises à jour et les interrogations.

On a choisit de limiter le cadre de notre étude sur le traitement de deux formats de données qui sont considéré comme des données sources de notre système.

- Les bases de données relationnelles, les bases de données Objets comme données structurées sous SGBD MySQL, SGBD DB4O sous Java.
- Fichiers XML comme données semi-structurées

**Nous détaillons dans ce qui suit** Les principes de base de chaque type de données ainsi que les plateformes choisi, déjà cité.

## I. Les sources de données traitées

### I.1 Les bases de données relationnelles

#### I.1.1 Le Modèle relationnel

Dans ce modèle, les données sont représentées par des tables, sans préjuger de la façon dont les informations sont stockées dans la machine. Les tables constituent donc la *structure logique* du modèle relationnel. Au niveau physique, le système est libre d'utiliser n'importe quelle technique de stockage (fichiers séquentiels, indexage, adressage dispersé, séries de pointeurs, compression, ...) dès lors qu'il est possible de relier ces structures à des tables au niveau logique. Les tables ne représentent donc qu'une abstraction de l'enregistrement physique des données en mémoire.

Le succès du modèle relationnel auprès des chercheurs, concepteurs et utilisateurs est dû à la puissance et à la simplicité de ses concepts. En outre, contrairement à certains autres modèles, il repose sur des bases théoriques solides, notamment la théorie des ensembles et la logique des prédicats du premier ordre.

Les objectifs du modèle relationnel sont de:

- proposer des schémas de données faciles à utiliser ;
- améliorer l'indépendance logique et physique ;
- mettre à la disposition des utilisateurs des langages de haut niveau ;
- optimiser les accès à la base de données ;
- améliorer l'intégrité et la confidentialité ;
- fournir une approche méthodologique dans la construction des schémas.

De façon informelle, on peut définir le modèle relationnel de la manière suivante :

- les données sont organisées sous forme de tables à deux dimensions, encore appelées relations, dont les lignes sont appelées n-uplets ou *tuples* en anglais ;
- les données sont manipulées par des opérateurs de l'algèbre relationnelle ;
- l'état cohérent de la base est défini par un ensemble de contraintes d'intégrité.

Au modèle relationnel est associée la théorie de la normalisation des relations qui permet de se débarrasser des incohérences au moment de la conception d'une base de données relationnelle.

[JAOUA 10].



### I.1.2 Algèbre relationnel

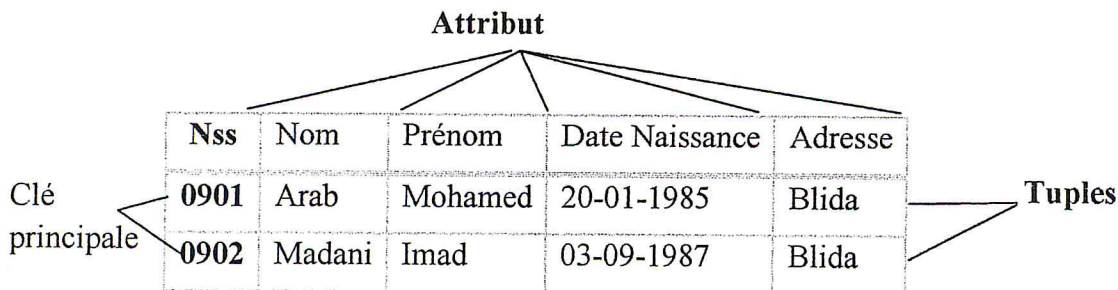
L'algèbre relationnelle est un support mathématique cohérent sur lequel repose le modèle relationnel.

La manipulation des éléments de la table se fait à l'aide d'opérations sur les ensembles, les principales opérations envisageables sont [GAR 02]:

- Les opérations ensemblistes : Union, différence, produit cartésien.
- Les opérations relationnelles : Projection, Restriction, Jointure .
- Les opérations dérivées : intersection, division.

**Exemple :** L'entité Malade pourra par exemple être représenté par :

- Nss (numéro de sécurité sociale)
- Nom
- Prénom
- Date Naissance
- Adresse



**Figure III.1** Table Malade

### I.1.3 Langage SQL

Le langage SQL (*Structured Query Language*) peut être considéré comme le langage d'accès normalisé aux bases de données. Il est aujourd'hui supporté par la plupart des produits commerciaux, que ce soit par les systèmes de gestion de bases de données tels que Microsoft *Access* ou par les produits plus professionnels tels que *Oracle*. Il a fait l'objet de plusieurs normes ANSI/ISO dont la plus répandue aujourd'hui est la norme SQL2 qui a été définie en 1992.

Les instructions SQL sont regroupées en catégories en fonction de leur utilité et des entités manipulées. Nous pouvons distinguer cinq catégories, qui permettent :



- la définition des éléments d'une base de données (tables, colonnes, clefs, index, contraintes, ...),
- la manipulation des données (insertion, suppression, modification, extraction, ...),
- la gestion des droits d'accès aux données (acquisition et révocation des droits),
- la gestion des transactions,
- et enfin le SQL intégré.

#### I.1.4 Forces du modèle Relationnel

Les principales forces du modèle relationnel ce sont :

- Les fondements mathématiques rigoureux sur lesquels le modèle repose ; ceux-ci permettent en particulier au moteur du SGBDR d'optimiser automatiquement les requêtes qui lui sont soumises en appliquant des théorèmes de l'algèbre relationnelle.
- l'indépendance des données et des programmes de traitement, permettant un partage aisé d'informations entre les applications.
- un langage de programmation non procédural puissant, concis et normalisé, le SQL, permettant de décrire, de mettre à jour et d'interroger les données mémorisées, d'effectuer des opérations algébriques complexes sur ces données, de spécifier des contraintes d'intégrité ainsi que de contrôler les accès aux données et de gérer des utilisateurs.
- un langage de programmation universel qui rend la base active.
- la spécification logique des liens entre tables qui affranchit le schéma de base de données de tout chemin d'accès prédéfini.
- le concept de transaction garantissant l'intégrité de la base de données à la suite de l'exécution d'une séquence d'instructions de mise à jour [JAOUA 10].

#### I.1.5 Faiblesses du modèle Relationnel :

Les principales faiblesses du modèle relationnel mises en évidence sont :

- l'indépendance des données et des programmes de traitement qui dissocie les données de leur comportement.
- le respect des formes normales qui a pour conséquence de multiplier le nombre de tables et donc le nombre d'opérations de jointures dont le coût en ressources mémoire et processeur est élevé.

- le nombre de types de base restreint et non extensible (il n'est pas possible de définir de nouveaux types de données par composition de types existants).
- Structure de données limités faible capacité de modélisation : il est difficile de représenter directement des structures de données complexes (hiérarchies, graphes, collections,...) [JAOUA 10].

**Synthèse :** Une solution pour les limites du modèle Relationnel c'est l'approche Objet.

## **I.2 Les bases de données Objet**

### **I.2.1 Le modèle Objet**

Les modèles a objets sont issus des réseaux sémantique et des langages de programmation orientés Objet. Ils regroupent les concepts essentiels pour modéliser de manière progressive des objets complexes encapsulés par des opérations de manipulation associées. Ils visent à permettre la réutilisation de structures et d'opérations pour construire des entités plus complexes.

Les modèles de données à objets ont été créés pour modéliser directement les entités du monde réel avec un comportement et un état.

Ce modèle est proche du modèle de classe de C++ qui peut être vu comme une implémentation des types abstraits de données. Il est aussi très proche du modèle Objet du langage JAVA.

### **I.2.2 Modélisation des Objets**

#### **a) Notion d'Objet**

Abstraction informatique d'une entité du monde réel caractérisé par une identité, un état et un comportement. Un objet est donc une instance d'entité, il possède une identité qui permet de le repérer [GAR 02].

Un objet peu changer de valeur mais pas d'identifiant

Deux objets ayant les mêmes valeurs et les identifiants différents sont considérer différent.

Une personne malade est un objet caractérisé par un nom, prénom, âge, et comme comportement associé : Ajouter (), modifier (), supprimer ().

L'objet de type personne malade d'identité M1 peut être décrit comme suit :

```
M1 {  
    Nss :0901, Nom: Arab, Prénom: Mohamed, Age: 25;  
    Ajouter (), modifier (); supprimer ().  
}
```

Un objet peut être très simple et composé seulement d'une identité et d'une valeur. Il peut aussi être très complexe et lui-même composé d'autres objets.

### b) Notion de classe

En orienté objet une classe décrit un ensemble d'instances d'objets, d'une forme donnée.

Une classe est un modèle définissant les méthodes et les variables pour un type particulier d'objet ayant des caractéristiques similaires. Un objet appartenant à une classe donnée est appelé instance de cette classe.

Les classes permettent de définir des objets d'une manière efficace. Les méthodes et les variables d'une classe ne sont définies qu'une seule fois, lors de la définition de la classe, sans que l'on ait à les répéter pour chaque instance de la classe.

Une classe spécifie la structure et le comportement communs des objets qu'elle permet de créer.

#### Exemple :

```
Class Malade {  
    Int nss  
    String nom ;  
    String prénom ;  
    Int âge;  
}
```

### c) Opération

Modélisation d'une action applicable sur un objet, caractérisée par un en-tête appelé signature définissant son nom, ses paramètres d'appel et ses paramètres de retour.

Le terme méthode sera aussi employé pour désigner une opération.

Le principe d'encapsulation hérité des types abstraits cache les structures de données (les attributs) et le code des méthodes en ne laissant visible que les opérations exportées, appelées **opérations publiques**. Par opposition, les opérations non exportées sont appelés **privés**.

**Exemple :** Ajouter () ; de l'exemple précédent.



### I.2.3 SGBD Orienté Objet

Les bases de données orientées objet existent depuis le milieu des années 1980. Leur objectif principal est d'unifier deux technologies : les systèmes de gestion de bases de données et la programmation orientée objet. Un langage de programmation orienté objet est en général très expressif mais manque d'outils performants de gestion de la persistance ; un SGBD, pour sa part, a comme principale caractéristique de gérer la persistance des données en manquant d'expressivité dans la formulation des traitements appliqués aux données.

Un SGBD orienté objet (SGBDOO) implante toutes les caractéristiques des langages de programmation orientés objets et est capable de fournir des services de persistance performants pour des applications traitant de grandes quantités de données. Dans un tel SGBD, les liens entre les objets sont bien sûr implantés sous forme de références à d'autres objets (en d'autres termes, sous la forme de pointeurs physiques) et l'accès aux informations est principalement de type navigationnel. Un accès sous forme de requêtes permettant d'extraire un ensemble d'objets satisfaisant un certain nombre de caractéristiques est en général également possibles.

### I.2.4 Forces Du modèle Objet

Les principales forces du modèle objet qui se dégagent de ce qui précède sont :

- l'unité entre les données et les programmes de traitement : un objet est l'association de structures de données et d'opérations spécifiant les capacités de l'objet ;
- le principe de la liaison dynamique, qui permet de spécifier des algorithmes de traitement qui pourront être appliqués à une famille de classes, certaines d'entre-elles encore inconnues au moment de la conception de l'algorithme ;
- un langage de programmation et un modèle de données uniques, lorsque la persistance peut être implantée de façon transparente à l'aide d'un SGBDOO ;
- la réduction des différences entre le monde à modéliser et le modèle de données ;
- la richesse des types de données à disposition ainsi que la possibilité, via les concepts de type de données abstrait et de classe, d'en créer de nouveaux ;
- la grande capacité de modélisation, en particulier celle de structures de données complexes telles que les hiérarchies, les graphes, les collections, ... ;
- la possibilité de concevoir des composants logiciels réutilisables, permettant ainsi de construire des applications par composition.

### I.2.5 Faiblesses Du modèle Objet

Les faiblesses du modèle objet qu'on peut relever sont principalement :

- le manque de bases théoriques dans la systématisation de la conception d'un modèle objet ;
- le problème de la gestion de la persistance dans un environnement sans SGBDOO, seul système de gestion de données vraiment adapté ;
- la navigation entre objets du modèle est définie une fois pour toutes ; si de nouveaux problèmes posés sur les mêmes données doivent être résolus postérieurement à la conception du modèle, celui-ci devra vraisemblablement être modifié ;
- la tenue de charge des moteurs de SGBDOO, lorsque le nombre d'utilisateurs concurrents augmente d'une manière importante .

### I.3 Les Données Semi structurées

Les données semi structurées sont les données qui possède une structure flexible et qui n'ont pas un schéma à priori mais plutôt dont le schéma peut être extrait à partir de la données. La plupart du temps, un ensemble de données semi structurées est représenté sous la forme d'un graphe dont les feuilles contiennent les données et dont les nœuds et les liens représentent la structure de l'ensemble. [KEZ 07]

La modification, l'ajout ou la suppression d'une donnée entraîne une modification du graphe, c'est-à-dire la structure de l'ensemble.

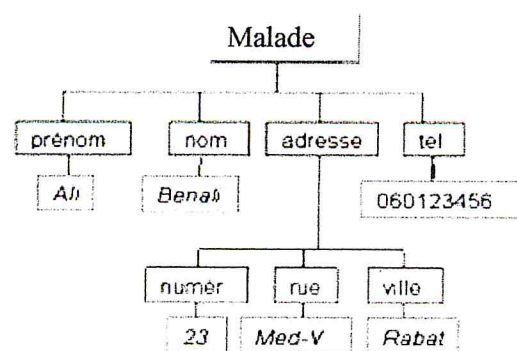


Figure III.2 Exemple de graphe de données semi structuré



### I.3.1 Le Modèle XML

XML a été mis au point par le XML Working Group sous l'égide du World Wide Web Consortium (W3C) dès 1996. Depuis le 10 février 1998, les spécifications *XML 1.0* ont été reconnues comme recommandations par le W3C, ce qui en fait un langage reconnu.

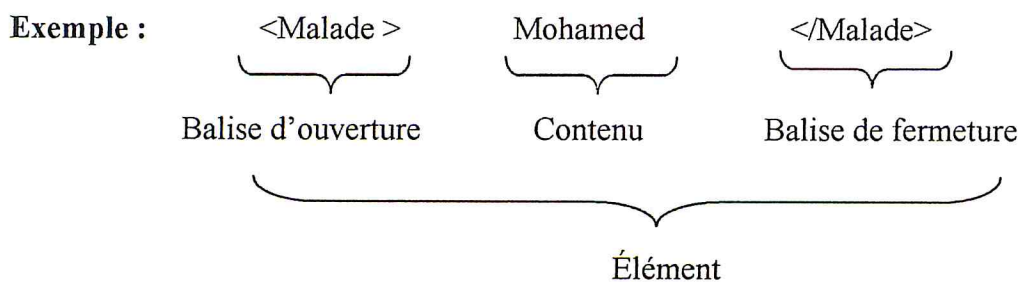
XML (entendez *eXtensible Markup Language* et traduisez *Langage à balises étendu*, ou *Langage à balises extensible*) est en quelque sorte un langage HTML amélioré permettant de définir de nouvelles balises. Il s'agit effectivement d'un langage permettant de mettre en forme des documents grâce à des balises.

XML peut être considéré comme un métalangage permettant de définir d'autres langages, c'est-à-dire définir de nouvelles balises permettant de décrire la présentation d'un texte XML est à la fois un format de description des données et un conteneur de ses données. Il a trouvé deux utilisations principales: en tant que format de stockage et comme format de transmission entre différentes applications.

### I.3.2 La syntaxe d'un document XML

Un fichier XML est présenté sous la forme d'un simple fichier texte contenant des balises de type `<balise>`, son composant principal est l'élément « Élément ».

Un document XML est composé d'éléments complexes (en contenant d'autres) et d'éléments simples contenant du texte, les éléments sont délimités par des balises ouvrantes et fermantes et peuvent être précisés par des attributs



Les éléments de données peuvent être imbriqués

**Exemple :**

```

<adresse>
  <rue>Abed Elah</rue>
  <ville>Blida</ville>
</adresse>

```



**La déclaration d'entête d'un document XML a la forme suivante :**

**Exemple :** `<?xml version="1.0" encoding="UTF-8"?>` ← Partie 1  
`<!DOCTYPE bd SYSTEM "Exemple.dtd">` ← Partie 2 } Entête de document

Cette expression indique au processus qui va traiter le document : La version du langage XML utilisé dans le document, le codage des caractères utilisés dans le document, si le document XML fait référence à une DTD ou à un schéma xml définit dans un autre fichier, il faut mettre "no". Sinon, on utilise "yes".

### I.3.3 Les qualifications d'un document XML

Un document XML peut avoir deux qualifications, il peut être :

**Bien formé** : quand il respecte la syntaxe du langage XML définie par le W3C.

**Valide** : quand il est associé à une définition de type de document et qu'il la respecte (nom des éléments, type, répétition et ordre d'apparition dans le document).

Un document XML bien formé est un document XML qui respecte certaines règles simples :

- Il existe un et un seul élément racine qui contient tous les autres éléments Les balises sont correctement imbriquées : chaque balise ouvrante a une balise fermante associée.
- Le nom des balises est libre mais il contient au moins une lettre.
- Les attributs des balises, lorsqu'ils existent, doivent comporter obligatoirement une valeur qui doit toujours apparaître entre double apostrophes.
- Quand un élément est vide, les balises peuvent être simplifiées :  
`<balise></balise>` est identique à `<balise/>`.

### I.3.4 Interrogation des documents XML

L'interrogation de documents XML demande un langage spécifique pour traiter les particularités de XML. Ce langage doit être structuré pour permettre de naviguer dans un document, de poser des questions et d'utiliser la réponse. Pour cela plusieurs langages ont été mis en œuvre : XPath, XSLT, XQuery

- **Xpath** est un langage qui permet de naviguer dans un document XML et dans certains cas de l'interroger ; il existe en deux versions 1.0 et 2.0. Xpath permet de sélectionner facilement une partie d'un document XML en utilisant un chemin dans l'arbre. Ce chemin est constitué de métadonnées du document et de prédicats. Nous pouvons interroger un document avec utilisation des métadonnées, Support des prédicats

- **XSLT** (eXtensible Stylesheet Transformation) :

Nous avons déjà vu qu'on peut interroger un document XML avec Xpath . L'une des principales insuffisances de ce langage est le manque de possibilité de mise en forme et de traitement des résultats. Or c'est la fonction de XSLT. Son utilisation dans ce contexte semble intéressante.

Le processeur XSLT (composant logiciel chargé de la transformation) crée une structure logique arborescente à partir du document XML et lui fait subir des transformations selon les *template rules* contenues dans la feuille XSL pour produire un arbre résultat .il représente aussi un langage de transformation d'un document XML en un autre document (pas uniquement XML). XSLT a deux cas d'utilisation ; l'affichage d'un document et sa conversion d'un format en un autre.

- **XQuery** : Le W3C vient de proposer Xquery comme recommandation d'interrogation de documents XML. Cela semble indiquer que XSLT a un certain nombre de carences dans ce domaine.

Tout d'abord, le langage en lui-même a quelques limites, les performances ne sont pas très bonnes et il manque encore un certain nombre de fonctionnalités.

On peut le caractériser XQuery comme étant le «SQL de XML»;il est le dernier né des langages d'interrogation XML, il date du 21 novembre 2006.

C'est est un nouveau langage créé spécifiquement pour l'interrogation de documents XML et pour répondre aux limites des langages précédents. Par exemple, l'un des problèmes de Xpath 1.0 est l'absence de système de types évolué: il n'est pas possible de sélectionner toutes les dates présentes dans un document. XQuery est doté d'un système de types très évolué. Pour le mettre en œuvre au sein d'un document XML, XQuery est doté d'un modèle de données qui est le plus complexe du monde XML.

Xquery est la recommandation qui offre le plus d'avantage pour l'interrogation de documents XML.



### 1.3.5 Forces du modèle XML

- La lisibilité : aucune connaissance ne doit théoriquement être nécessaire pour comprendre un contenu d'un document XML
- Auto descriptif et extensible
- Une structure arborescente permettant de modéliser la majorité des problèmes informatiques
- Universalité et portabilité : les différents jeux de caractères sont pris en compte
- Déployable : il peut être facilement distribué par n'importe quels protocoles à même de transporter du texte, comme HTTP
- Intégrabilité : un document XML est utilisable par toute application pourvue d'un parseur (c'est-à-dire un logiciel permettant d'analyser un code XML)
- Extensibilité : un document XML doit pouvoir être utilisable dans tous les domaines d'applications.
- Offre un grand nombre de standards associés : XML Schéma, XQuery, XPath

Ainsi, XML est particulièrement adapté à l'échange de données et de documents.

L'intérêt de disposer d'un format commun d'échange d'information dépend du contexte professionnel dans lequel les utilisateurs interviennent. C'est pourquoi, de nombreux formats de données issus de XML apparaissent (il en existe plus d'une centaine) :

- OFX : Open Financial eXchange pour les échanges d'informations dans le monde financier
- MathML : Mathematical Markup Language permet de représenter des formules mathématique
- CML : Chemical Markup Language permet de décrire des composés chimiques
- SMIL : Synchronized Multimedia Integration Language permet de créer des présentations multimédia en synchronisant diverses sources : audio, vidéo, texte,...

En raison de la versatilité de XML il n'y a pas de langage parfait.



## **II Le modèle commun**

La construction du schéma médiateur (approche GAV ou LAV) s'effectue en définissant des vues sur des sources de données hétérogènes. Certains problèmes liés à l'hétérogénéité du format des sources (bases de données relationnelles, objet, document XML) sont résolus grâce à l'utilisation d'un modèle commun (également appelé modèle pivot ou global). Ce modèle doit permettre de représenter des données provenant de sources hétérogènes pour les intégrer d'une façon uniforme dans un schéma médiateur.

Les premiers systèmes d'intégration de données ont utilisé les modèles relationnels ou objet comme modèle commun. Cependant, l'intégration de sources de données peu ou pas structurées est difficile avec ces modèles permettant de représenter des bases de données.

Les modèles de données semi-structurés (XML) ont été conçus pour représenter facilement des données irrégulières provenant de sources hétérogènes, structurées ou non.

Nous avons choisi XML comme formalisme de représentation, pour sa capacité à représenter tout type de donnée.

## Conclusion

L'étude de ces trois modèles devrait servir de base à une réflexion sur la position du modèle XML dans le contexte actuel de l'intégration de données.

Le choix de l'utilisation de XML comme modèle d'intégration (et non pas le modèle relationnel ou le modèle objet utilisés dans la plupart des études sur les systèmes de gestion de données distribuées et hétérogènes) se justifie par la richesse de ce langage, à savoir l'abondance des descriptions et le typage des données, la clarté et l'extensibilité

Un autre critère participant au choix de XML comme modèle d'intégration est que son caractère général rend aisée la traduction des modèles de données existants. Pour chaque objet du modèle (réseau, hiérarchique, relationnel ou objet), il est possible de construire l'arbre XML associé. Les nombreux standards associés à ce langage en font le candidat idéal comme modèle d'intégration dans une architecture d'accès à des données distribuées et hétérogènes.

Ainsi, XML Schéma permet de fournir un modèle de métadonnées uniforme entre les adaptateurs et le médiateur. Le langage XQuery offre l'interrogation efficace de requêtes sur XML et les feuilles de style XSL permettant de présenter les données à l'utilisateur [ BAKH 06].

En fin, la standardisation de XML par W3C et son succès dans l'industrie informatique ont le fait un bon candidat comme langage commun pour un système d'intégration.

---

# *Patric II*

## *Conception et*

### *mise en œuvre*

---

#### *Sommaire*

- 1. Etude fonctionnelle*
- 2. Etude technique*
- 3. Conception*
- 4. Mise en œuvre*



---

# *CHAPITRE IV*

## L' étude fonctionnelle

---

## Introduction

Ce chapitre présentera *l'étude fonctionnelle* du système que nous allons réaliser, Cette étape va nous servir à poser les bases de la capture des besoins du système à réaliser, nous allons parler des fonctionnalités que peut offrir ce dernier a fin de bien connaître les acteurs qui interagissent avec lui par la suite nous allons les modéliser en utilisant le diagramme des cas d'utilisation d'UML.

### 1. Capture des besoins fonctionnels

#### 1.1 L'étude préliminaire

L'étude préliminaire c'est une pré\_étude, elle consiste à effectuer un premier repérage des besoins fonctionnels et opérationnels, en utilisant principalement du texte, ou des diagrammes très simples, elle prépare les activités plus formelles de capture des besoins fonctionnels.

##### 1.1.1 Objectifs de cette étude

L'étude préliminaire a pour objectifs de :

- Etablir un recueil initial des besoins fonctionnels et technique.
- Modéliser le contexte du système, considéré comme une boîte noire en
  - Identifiant les entités externes au système qui interagissent directement avec lui (acteurs).
  - Répertoire les interactions (émission / réception de message) entre ces acteurs et le système.

##### 1.1.2 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'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 fournis par les adaptateurs à l'utilisateur.

### 1.1.3 Identification des acteurs du système

#### *Qu'est qu'un acteur ?*

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.

Nous allons présenter dans un tableau, les différents acteurs et les missions de chacun.

<b>Acteurs</b>	<b>Mission</b>
Utilisateur	L'utilisateur s'occupe du lancement de la requête
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 IV.1** les acteurs du système

### 1.1.4 Identification des messages

#### *Qu'est qu'un message ?*

Un message représente la spécification d'une communication unidirectionnelle entre les objets, qui transporte de l'information avec l'intention de déclencher une activité chez le récepteur.

<b>Emetteur</b>	<b>Message reçu par le système</b>
Utilisateur	Requête
Médiateur	Demande de traitement de la requête
Adaptateur	Demande de traduction de la requête
Adaptateur	Demande de traduction du résultat

**Tableau IV.2** les messages reçus par le système



Message émis par le système	Récepteur
résultat	Utilisateur
Ensemble sous requête	Adaptateur
Ensemble sous résultat	Médiateur

Tableau IV.3 les messages émis par le système

1.1.5 Modélisation du contexte

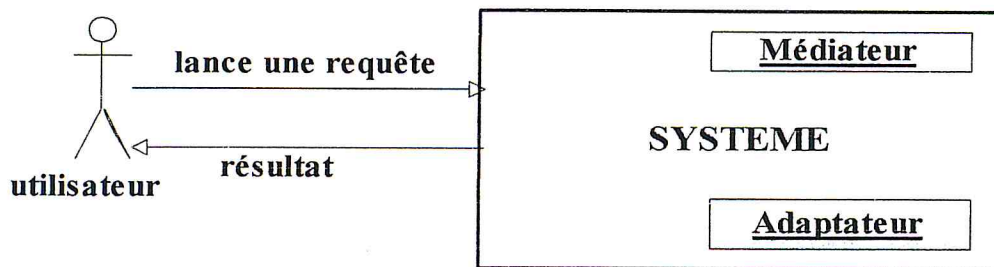


Figure IV.1 diagramme de contexte du système

1.2 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.

Le tableau qui va suivre englobe les cas d'utilisations identifiés dans cette étude.

Acteur	Cas d'utilisation	Message émis/reçus par les acteurs
Utilisateur	Lancer une requête	<i>Emet</i> : saisir une requête <i>Reçoit</i> : résultat (réponse de la requête)
<b>Création de schéma global</b>		
Médiateur	consulter les schémas locaux	<i>Emet</i> : consultation des schémas locaux <i>Reçoit</i> : un ensemble d'attribut
	consulter le fichier des métas donnés	<i>Emet</i> : concept local <i>Reçoit</i> : concept global
	Utiliser l'approche GAV	<i>Emet</i> : construction des vues sur les schémas sources <i>Reçoit</i> : schéma global

Traitement des requêtes		
Médiateur	Décomposer la requête	<i>Emet</i> : reformulation de la requête Globale <i>Reçoit</i> : sous requêtes
	Localiser les sources pertinentes	<i>Emet</i> : recherche dans la table de correspondance <i>Reçoit</i> : emplacement de chaque attribut recherché
	Connecter chaque sous requête a l'adaptateur approprié	<i>Emet</i> : sous requêtes <i>Reçoit</i> : source appropriée
	<b>Fusion des résultats</b>	
	Recomposer le résultat	<i>Emet</i> : sous réponses. <i>Reçoit</i> : Résultat.
Traduction		
Adaptateur	Traduire les sous requêtes	<i>Emet</i> : sous requête en langage global <i>Reçoit</i> : sous requête en langage local
	Traduire les sous résultats	<i>Emet</i> : sous résultats en langage local <i>Reçoit</i> : sous résultats en langage global

Tableau IV.4 Identification des cas d'utilisation

### 1.3 Les diagrammes des cas d'utilisation

#### 1.3.1 Diagramme des cas d'utilisation Global du système

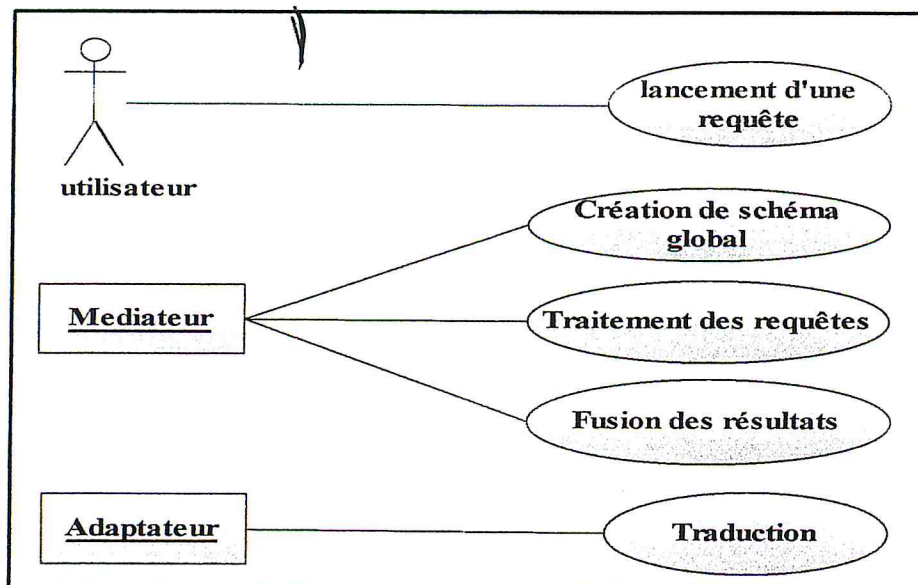


Figure IV.2 diagramme de cas d'utilisation global de système

## a. Le cas d'utilisation Lancement d'une requête

### a.1 Sommaire d'identification

**Titre :** Lancement d'une requête  
**But :** consulter un ensemble de donnée  
**Résumé :** Lancement d'une requête pour consulter un ensemble de donnée  
**Acteurs :** utilisateur du système

### a.2 Description des enchainements

- **Pré-conditions:** aucune
- **Scenario nominal :** ce cas d'utilisation commence lorsque l'utilisateur accède à une interface dans laquelle il pose sa requête.
- **Poste-condition :** une requête en schéma globale est crée.
- **Exceptions :** Requête erroné.

### 1.3.2 Diagramme des cas d'utilisation créer un schéma global

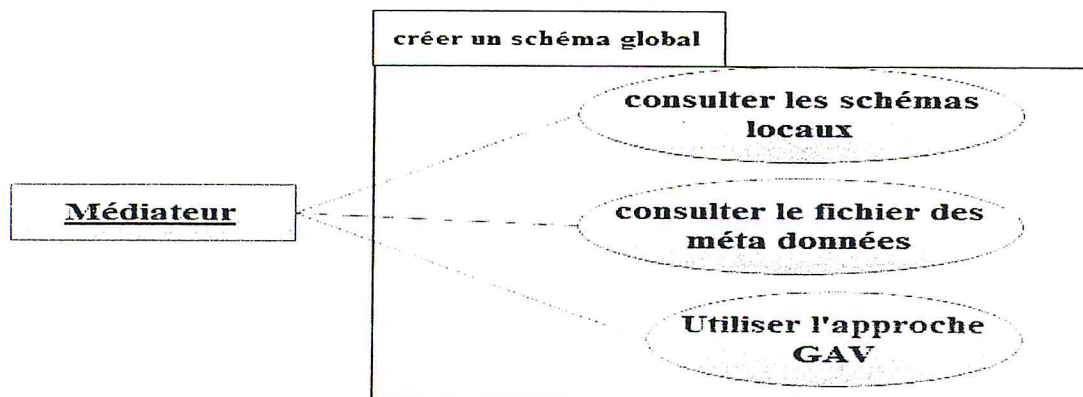


Figure IV.3 Diagramme de cas d'utilisation créer un schéma global

## a. Le cas d'utilisation consulter les schémas locaux

### a.1 Sommaire d'identification

**Titre :** consulter les schémas locaux  
**But :** déterminer l'ensemble des concepts globaux à partir des concepts locaux  
**Résumé :** consulter le fichier des métras donnés pour déterminer l'ensemble des concepts globaux  
**Acteurs :** Médiateur

### a.2 Description des enchainements

- **Pré-conditions:** existence des schémas locaux.



- **Scenario nominal** : ce cas d'utilisation commence lorsque le médiateur crée le schéma global avant de commencer le traitement de la requête car c'est dans ce schéma global la requête de l'utilisateur est formulée. Le médiateur doit tirer l'ensemble des éléments constituant les schémas locaux.
- **Poste-condition** : un ensemble d'éléments est tiré.
- **Exceptions** : aucune.

## b. Le cas d'utilisation consulter le fichier des métas donnés

### b.1 Sommaire d'identification

**Titre** : consulter le fichier des métas donnés  
**But** : déterminer l'ensemble des éléments des schémas locaux  
**Résumé** : consulter le fichier des métas donnés pour déterminer l'ensemble des attributs globaux.  
**Acteurs** : Médiateur

### b.2 Description des enchainements

- **Pré-conditions**: fichier des métas donnés non vide.
- **Scenario nominal** : ce cas d'utilisation commence lorsque le médiateur crée le schéma global. Le médiateur doit tirer l'ensemble des concepts globaux à partir des concepts locaux.
- **Poste-condition** : concept global pour chaque ensemble de concepts locaux.
- **Exceptions** : aucune.

## c. Le cas d'utilisation utiliser l'approche GAV

### c.1 Sommaire d'identification

**Titre** : Utiliser l'approche GAV  
**But** : créer le schéma Global  
**Résumé** : création de vue sur les schémas des sources.  
**Acteurs** : Médiateur

### c.2 Description des enchainements

- **Pré-conditions**: consulter les schémas locaux.
- **Scenario nominal** : le médiateur connaît tous les attributs des schémas locaux, il peut maintenant déterminer les attributs communs en les généralisant en un attribut général, c'est cet attribut général qu'on doit trouver dans le schéma global.
- **Poste-condition** : création d'un schéma global.
- **Exceptions** : aucune.

### 1.3.3 Diagramme de cas d'utilisation Traitement de requête

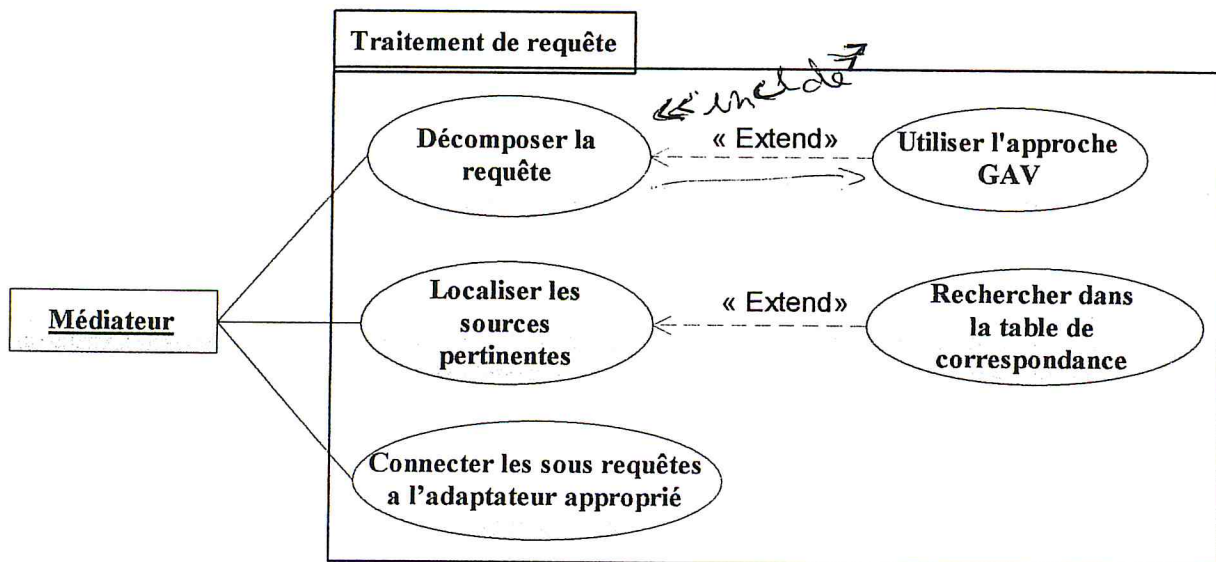


Figure IV.4 Diagramme de cas d'utilisation Traitement de requête

#### a. Le cas d'utilisation décomposer la requête

##### a.1 Sommaire d'identification

<p><b>Titre :</b> décomposer la requête  <b>But :</b> obtenir des requêtes adaptées aux sources  <b>Résumé :</b> remplacer chaque attribut par sa définition dans l'approche GAV  <b>Acteurs :</b> Médiateur</p>
--

##### a.2 Description des enchainements

- **Pré-conditions:** lancement d'une requête.
- **Scenario nominal :** analyser l'ensemble des attributs constituant la requête, remplacer les attributs générales par leurs définitions dans l'approche GAV, obtenir des sous requêtes adaptés aux sources.
- **Poste-condition :** création des sous requêtes.
- **Exceptions :** déclencher une exception si les attributs n'existent pas ou sont mal écrits donc il faut mentionner un message en indiquant qu'il faut ressaisir l'attribut erroné.

**b. Le cas d'utilisation Localiser les sources pertinentes****b.1 Sommaire d'identification**

**Titre :** localiser les sources pertinentes

**But :** localiser les sources pertinentes

**Résumé :** recherche des éléments dans la table de correspondance pour localiser les sources.

**Acteurs :** Médiateur

**b.2 Description des enchainements**

- **Pré-conditions:** requête valide
- **Scenario nominal :** après la détermination de l'ensemble d'attribut de la requête, le médiateur détermine l'emplacement de chaque attribut à l'aide d'une table de correspondance.
- **Poste-condition :** l'emplacement de chaque attribut.
- **Exceptions :** aucune.

**c. Le cas d'utilisation Connecter les sous requête à l'adaptateur approprié****c.1 Sommaire d'identification**

**Titre :** Connecter les sous requête à l'adaptateur approprié

**But :** Connecter les sous requête à l'adaptateur approprié

**Résumé :** connecter chaque source à l'adaptateur approprié pour la traduire

**Acteurs :** Médiateur

**b.2 Description des enchainements**

- **Pré-conditions:** chaque attribut est localisé.
- **Scenario nominal :** le médiateur envoie chaque sous requête à l'adaptateur connecter à la base de données approprié pour la traduire en langage local,
- **Poste-condition :** connexion avec les adaptateurs appropriés.
- **Exceptions :** aucune.



### 1.3.4 Diagramme de cas d'utilisation Fusion des résultats

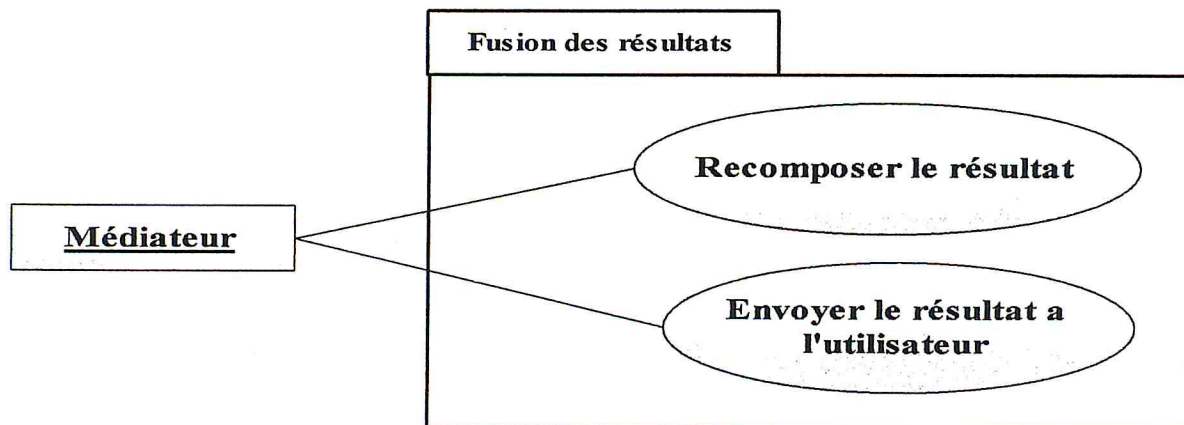


Figure IV.5 Diagramme de cas d'utilisation Fusion des résultats

#### a. Le cas d'utilisation Recomposer le résultat

##### a.1 Sommaire d'identification

**Titre :** Recomposer le résultat  
**But :** Recomposer le résultat  
**Résumé :** Recomposer le résultat pour avoir une seule réponse.  
**Acteurs :** Médiateur

##### a.2 Description des enchainements

- **Pré-conditions:**
  - le nombre de sous résultat égale le nombre de sous requête.
  - Existence de sous résultat.
- **Scenario nominal :** ce cas d'utilisation commence après la réception des sous réponses, le médiateur fait la jointure des résultats pour avoir une seule réponse.
- **Poste-condition :** résultat.
- **Exceptions :** aucune.

**b. Le cas d'utilisation envoyer le résultat à l'utilisateur**

**b.1 Sommaire d'identification**

**Titre :** envoyer le résultat à l'utilisateur.  
**But :** fournir un résultat .  
**Résumé :** fournir un résultat à l'utilisateur pour sa requête.  
**Acteurs :** Médiateur

**b.2 Description des enchainements**

- **Pré-conditions:** Existence résultat.
- **Scenario nominal :** ce cas d'utilisation commence après la recombinaison des sous résultats de chaque adaptateur.
- **Poste-condition :** résultat.
- **Exceptions :** aucune.

**1.3.5 Diagramme de cas d'utilisation Traduction**

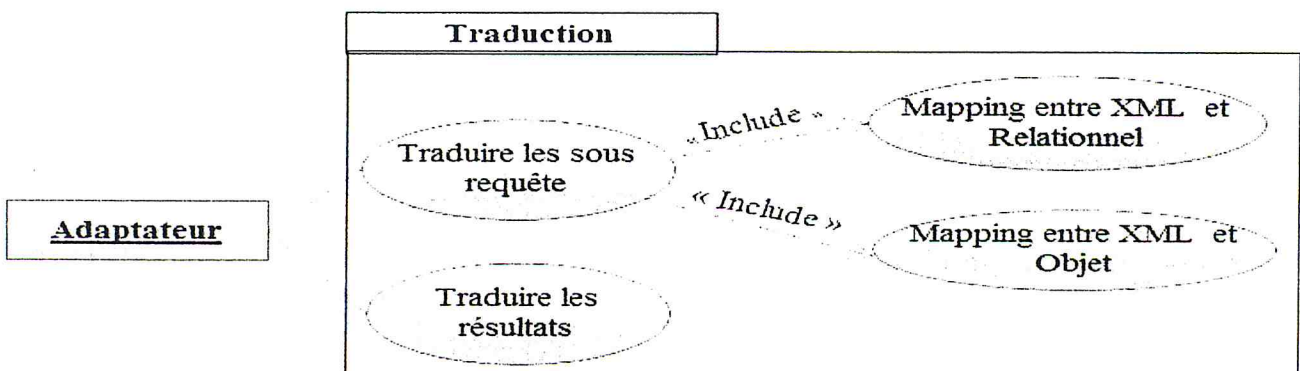


Figure IV.6 cas d'utilisation Traduction

**a. Le cas d'utilisation Traduire les sous requête**

**a.1 Sommaire d'identification**

**Titre :** Traduire les sous requêtes  
**But :** Traduire les sous requêtes  
**Résumé :** Traduire les sous requêtes du langage global aux langages locaux adapté aux sources.  
**Acteurs :** Adaptateur

## a.2 Description des enchainements

- **Pré-conditions** : sous requêtes en langage global.
- **Scenario nominal** : Traduire les sous requêtes du langage global aux langages locaux adaptés aux sources.
- **Poste-condition** : sous requêtes en langage local.
- **Exceptions** : aucune.

## b. Le cas d'utilisation traduire les résultats

### b.1 Sommaire d'identification

**Titre** : traduire les résultats

**But** : traduire les résultats

**Résumé** : Traduire les résultats du langage local au langage global adapté aux sources

**Acteurs** : Adaptateur

### b.2 Description des enchainements

- **Pré-conditions**: sous résultat en langage local.
- **Scenario nominal** : Traduire les résultats du langage local au langage global adapté au médiateur.
- **Poste-condition** : sous résultat en langage global.
- **Exceptions** : aucune.

## 2 L'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]**



## 2.1 Formalisation des scénarios

### 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.

#### 2.1.1 diagramme de séquence du cas d'utilisation lancement d'une requête

→ Diagramme de séquence

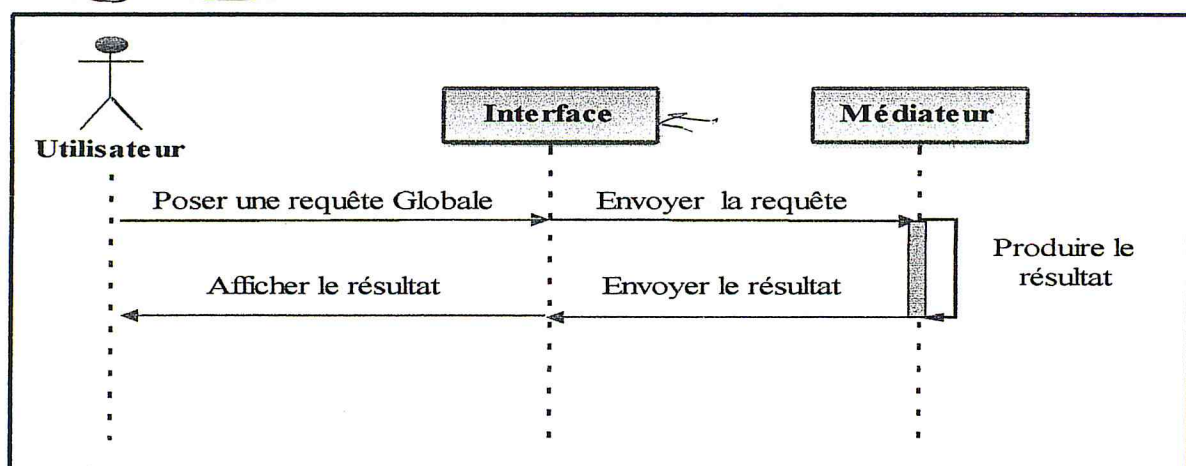


Figure IV.7 diagramme de séquence du cas d'utilisation lancement d'une requête

2.1.2 diagramme de séquence du cas d'utilisation Création de schéma global

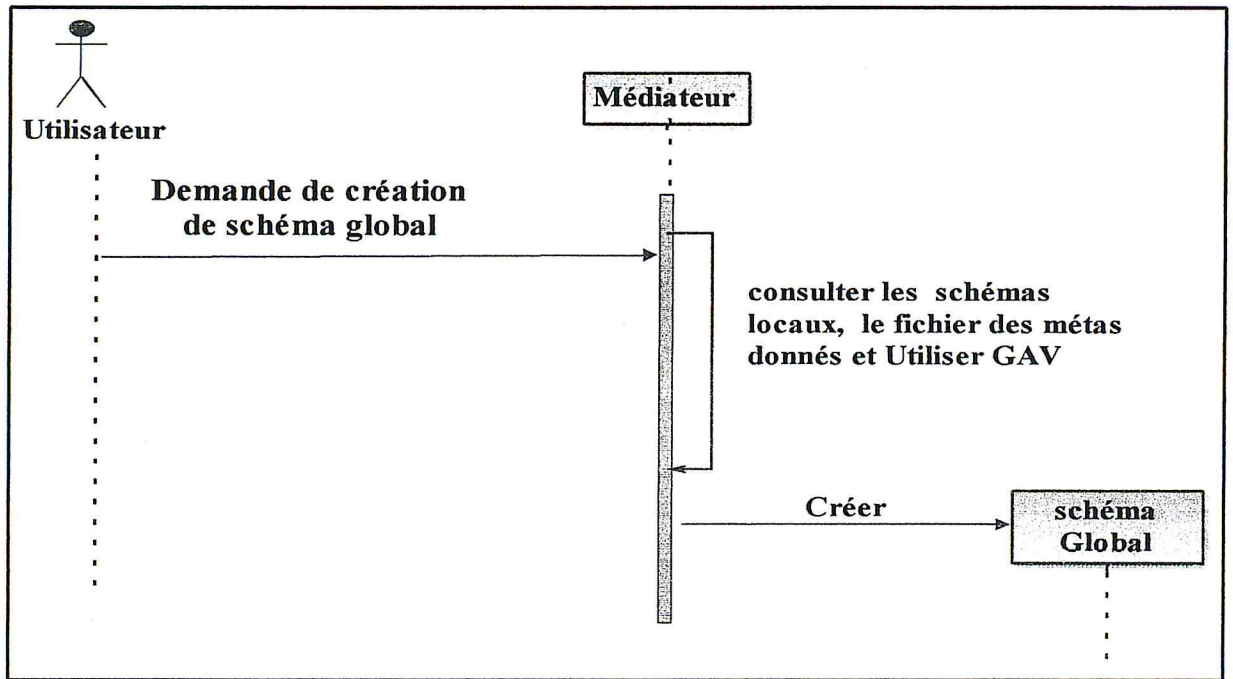


Figure IV.8 diagramme de séquence du cas d'utilisation Création de schéma global

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

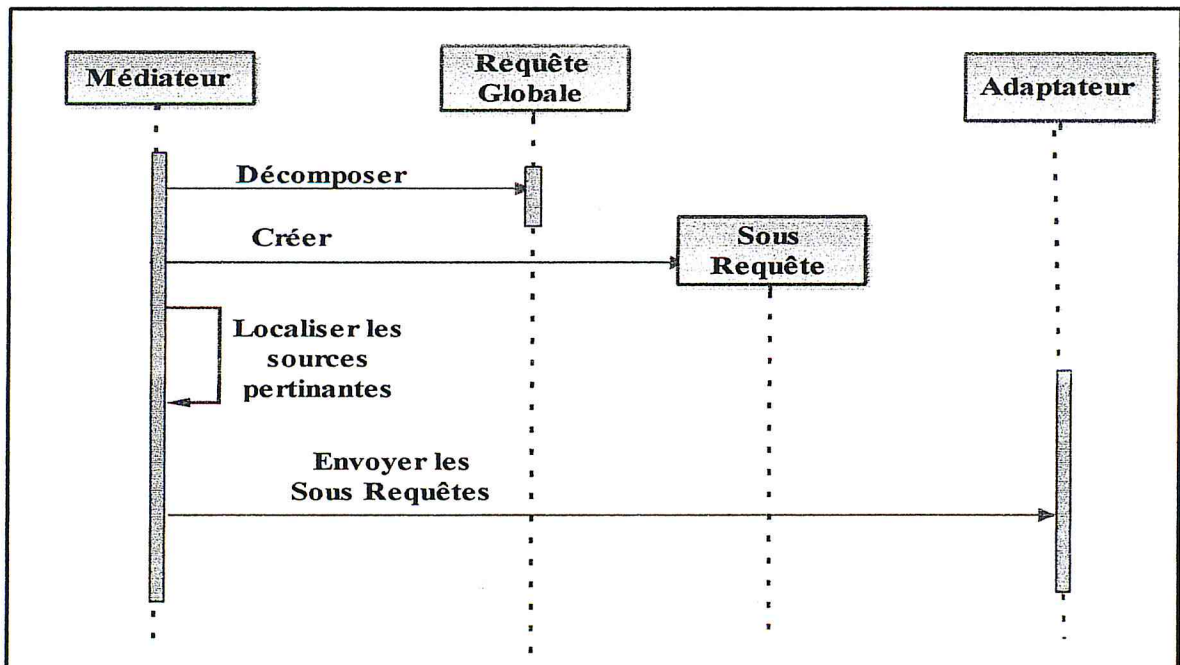


Figure IV.9 diagramme de séquence du cas d'utilisation traitement de la requête global

2.1.4 diagramme de séquence du cas d'utilisation Traduction

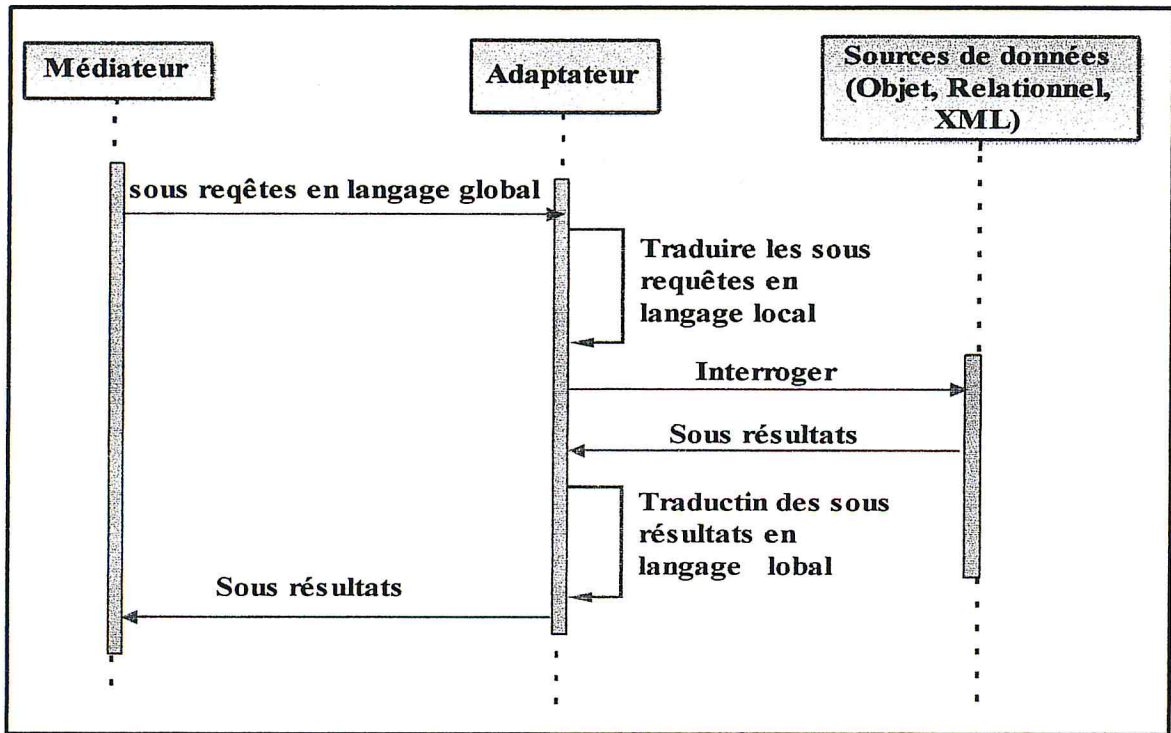


Figure IV.10 diagramme de séquence du cas d'utilisation Traduction

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

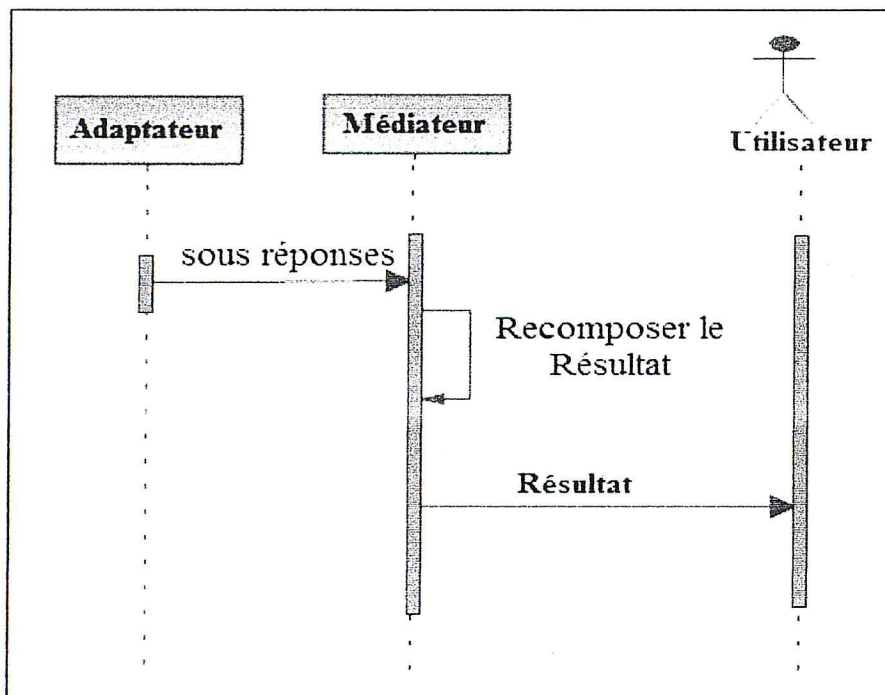


Figure IV .11 diagramme de séquence du cas d'utilisation Fusion des résultats



## 2.2 Diagramme d'état transition

Les diagrammes d'états-transitions d'UML décrivent le comportement d'un objet à l'aide d'un automate à états finis. Ils présentent les séquences possibles d'états et d'actions qu'une instance de classe peut traiter au cours de son cycle de vie en réaction à des événements discrets.



Figure IV.12 Diagramme d'état transition pour la classe requête



Figure IV.13 Diagramme d'état transition pour la classe adaptateur

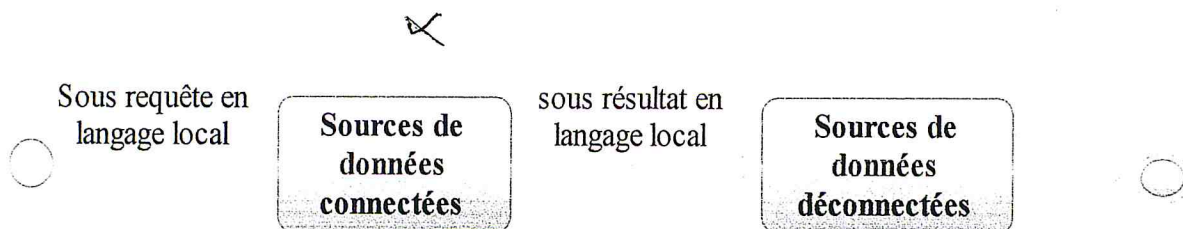


Figure IV.14 Diagramme d'état transition pour la classe source de données

---

# *CHAPITRE V*

## L' étude technique

---

## Introduction

Dans ce chapitre nous allons présenter l'étude technique de notre système. Nous allons en premier extraire et choisir ce qui peut être bénéfique de point de vue logiciel et matériels de notre système. 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 facilite le travail.

## I. Présentation des outils de développement

### I.1 Choix du langage de programmation (JAVA)

Le langage utilisé pour la réalisation de notre logiciel est le JAVA qui est un langage de programmation orienté Objet développé 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

*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.*

### I.2 Système de Gestion de Base de données (SGBD)

#### I.2.1 SGBD Objet

Le SGBD choisit est DB4O qui signifie « database for objects » ; C'est un système de gestion de base de données orienté objet (OODBMS ou SGDBOO en français) écrit en Java et en .NET et de ce fait destiné à ces deux plateformes. Ce logiciel a été initialement publié en 2001 par db4Objects, et depuis lors s'est taillé une part majeure du marché de ce qu'on peut appeler les bases de données objet de deuxième génération.



- DB4O est un système qui
  - Fournit un stockage persistant de données complexes (Tableaux, Collections , Objets... ).
  - Fournit des fonctionnalités avancé ( gestion des transaction, cryptage, interface utilisateur)
    - Véritable mode d'utilisation client-serveur.
- Comme produit DB4O il y a :
  - Une Base de données (DB4O Database Engine )
  - Une interface graphique d'administration Object Manager.
- Db4O supporte un système de requête simple QBE( Query By Example) [**Jean Jack**]

### **I.2.2 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.

### **I.2.3 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.

### I.2.4 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.

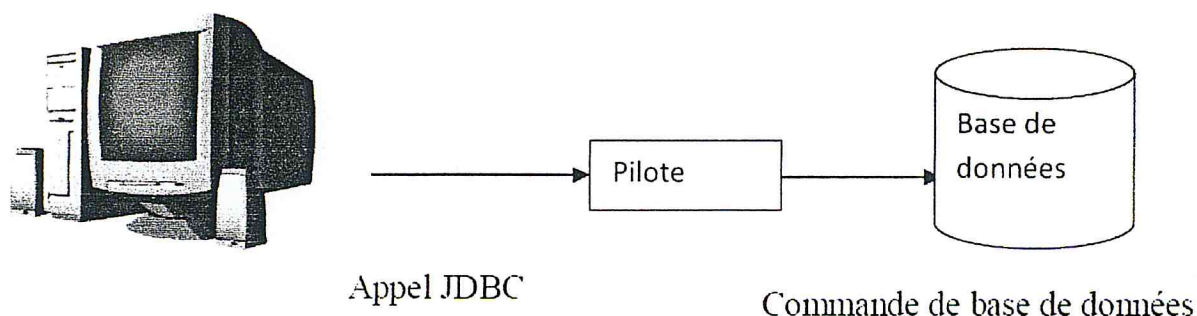


Figure V.1 Architecture de JDBC

Les accès à une base de données doivent être effectués en sept 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.

---

*CHAPITRE VI*

Conception

---



**Introduction**

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

Nous arrivons maintenant à la phase ultime de modélisation avec UML, après la modélisation des besoins puis l'organisation de la structure de la solution, la conception consiste à construire et à documenter précisément les classes, les tables et les méthodes qui constituent le codage de la solution.

**I. Conception du système**

**1.1 Diagramme de Classe**

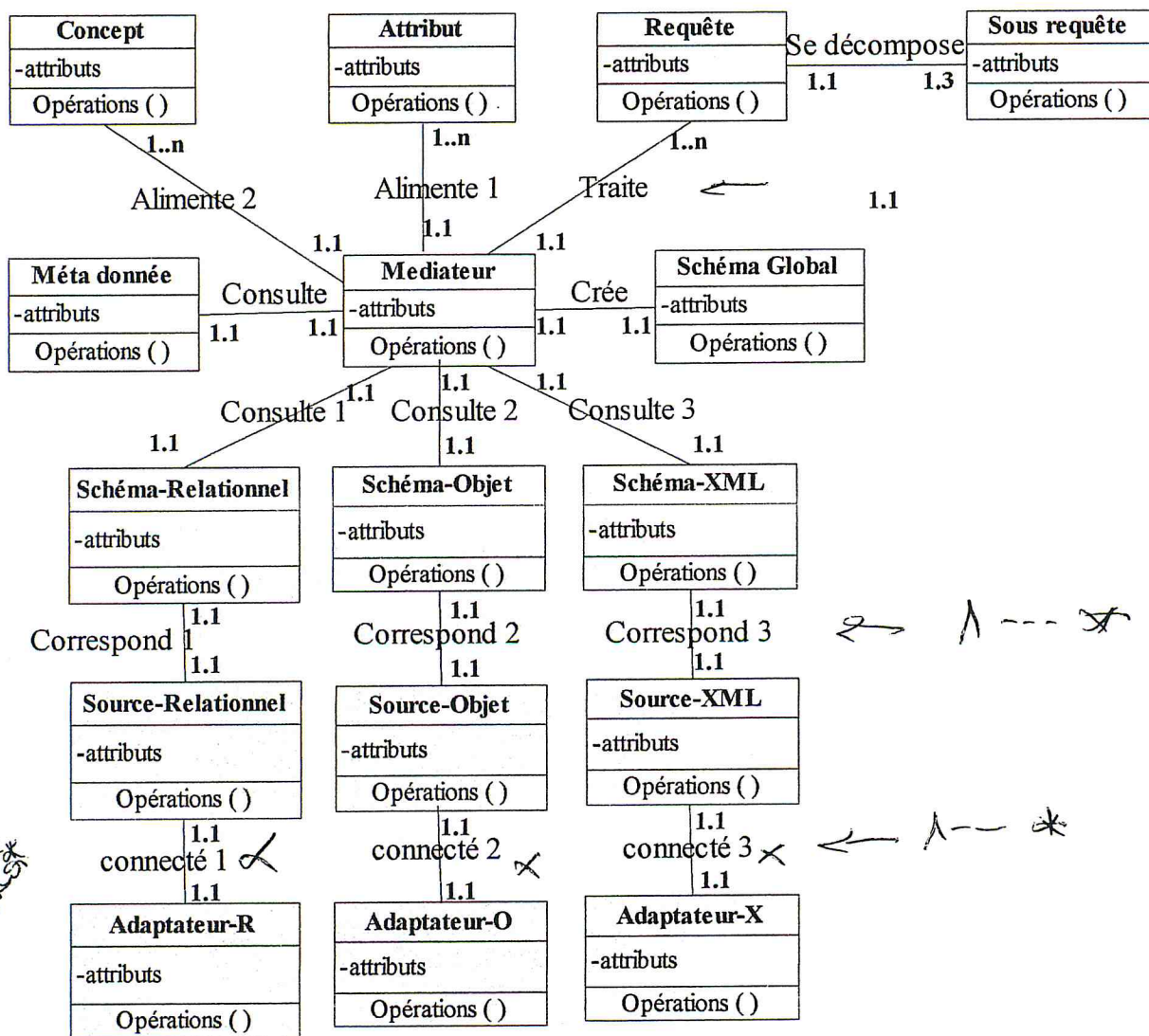


Figure VI.1 Diagramme de classe

I.1 Concevoir les classes et les attributs

Classe	Attribut	désignation	Type
Requête	Select	Les attributs globaux de la clause Select	String
	From	Les Concept globaux de la clause From	String
	Where	Les conditions de sélections des attributs	String
Sous RequêteG	Select	Les attributs locaux de la clause Select	String
	From	Les Concept locaux de la clause From	String
	Where	Les conditions de sélections des attributs	String
Schéma global	Nom_Cpt	Nom Concept	String
	Nom_attribut	Nom attribut	String
Schéma relationnel	Nom_fichier	Nom fichier	String
	Emplacement	Emplacement	String
Schéma Objet	Nom_fichier	Nom fichier	String
	Emplacement	Emplacement	String
Schéma XML	Nom_fichier	Nom fichier	String
	Emplacement	Emplacement	String
Méta données	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

Tableau VI.1 Description des classes

## I.2 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 RequêteG	Créer_Srequête ()	Créer une sous requête
	Décomposer_Srequête ()	Requête sous décomposé
	Exécuter_Srequête ()	Exécuter sous Requête
Méta données	Consulter_Métadonnée ()	Consulter le fichier des métadonnées
Concept	Consulter_Concept ()	Consulter la table concept
Attribut	Consulter_attribut ()	Consulter la table attribut
Médiateur	Créer_schémaG()	Le médiateur crée le schéma Global
	Traite_requête()	Le médiateur Traite la requête
	Recomposer_résultat()	Le médiateur recompose le résultat
	Alimenter_Concept()	Le médiateur Alimente la table des concepts
	Alimenter_Concept()	Le médiateur Alimente la table des attributs
Adaptateur	Traduire_SousRequête()	L'adaptateur traduit la sous requête du langage global au langage local

Tableau VI.2 Description des opérations



## II. Architecture du système

### II.1 Architecture générale

#### II.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- une intégration sémantique des données en utilisant les nouvelles technologies pour la représentation de la sémantique des données (méta donnés) ;
- 3- Les données hétérogènes supportées sont les données relationnelles, les objets et XML;
- 4- Approche d'intégration descendante est une adaptation des règles de mapping GAV offrent une flexibilité aux changements;
- 5- Traitement des sous requêtes destinées aux sources de données permettant de supporter des systèmes anciens ;
- 6- Notre solution permet de résoudre le problème de l'intégration de données hétérogènes et en prenant en compte les conflits des données (syntaxiques, sémantiques) à l'intégrer.

II.1.2 Vue générale de l'architecture

L'architecture générale de notre solution est présentée dans la figure suivante :

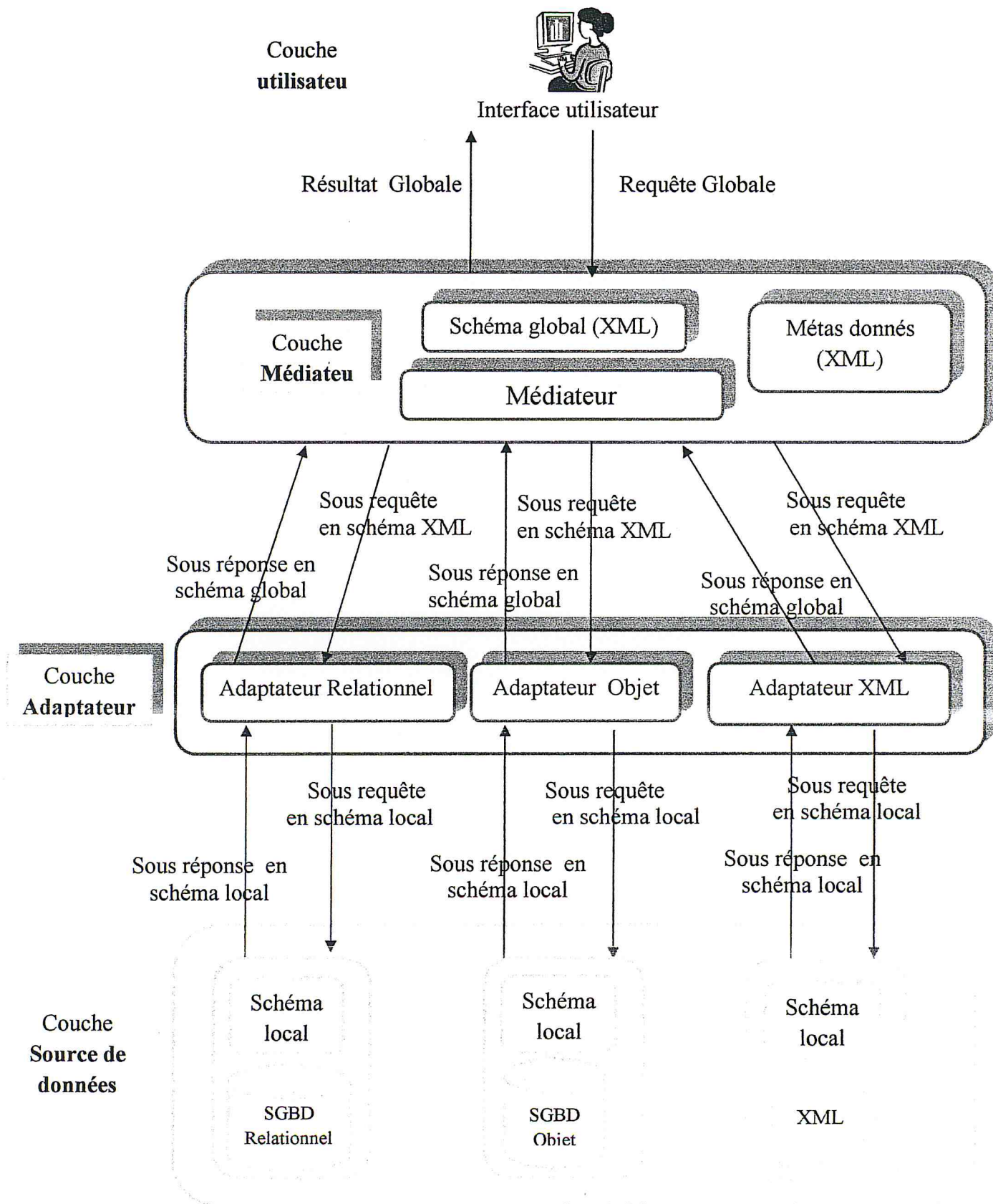


Figure VI.2 : Architecture générale du système

**II.2.1 Description des couches de système**

Notre système d'intégration a une architecture à 4 couches, de bas en haut commençons par :

**II.1.3.1 Couche Source de Données**

Dans un processus d'intégration, 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 et Objet), 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 deuxième source de données structurée** : c'est une Base de données Objet qui porte le nom " BdOMédicale" pour concevoir cette base on a utilisé un SGBO (Système de Gestion de Base de Données Objet) dbo4. Cette base contient 5 classes :

Voici trois classes parmi ce qu'on a

```

Class MedecinTraitant
{ MedNom
  MedPre
  MedNSS
  ...
}
    
```

```

Class Atteint
{ NSSPatient
  NuméroMaladie
}
    
```

```

Class Maladie
{ Description
  NuméroMaladie
}
    
```



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

```
<?xml version="1.0"?>
<BaseXml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:///d:/mes%20fichier/M2/memoire/Application/Basexml/cv.xsd">
  <CVMedecin>
    <NSSMedecin>9061000</NSSMedecin>
    <MENom>Salhi</MENom>
    <MEPrenom>Fouzia</MEPrenom>
    <MEAdresse>blida</MEAdresse>
    <METelephone>021232456</METelephone>
    <MEDate_Naiss>21/12/1980</MEDate_Naiss>
    <Diplome>
      <Date_Obtention> 23-09-2009</Date_Obtention>
      <Nom_Diplome>Medecin Generaliste</Nom_Diplome>
      <Etablissement>Universite De Blida </Etablissement>
    </Diplome>
    <Experience>
      <Organisme>Hopital IbnZiri Bologhine</Organisme>
      <Date_Debut>23-03-2010</Date_Debut>
      <Date_Fin>20-09-2010</Date_Fin>
    </Experience>
  </CVMedecin>
  ....
</BaseXml>
```

**Figure VI .3** : 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>cvmedecin </NomTable>
  <NomAttribut1>NSSMedecin</NomAttribut1>
  <NomAttribut2>MENom</NomAttribut2>
  <NomAttribut3>MEPreNom</NomAttribut3>
  <NomAttribut4>MEAdresse</NomAttribut4>
  <NomAttribut5>METelephone</NomAttribut5>
  <NomAttribut6>MEDate_Naiss</NomAttribut6>
</Table>
<Table>
  <NomTable>diplôme</NomTable>
  <NomAttribut1>Date_Obtention</NomAttribut1>
  <NomAttribut2>Nom_Diplome</NomAttribut2>
  <NomAttribut3>Etablissement</NomAttribut3>
</Table>
<Table>
  <NomTable>experience</NomTable>
  <NomAttribut1>Organisme</NomAttribut1>
  <NomAttribut2>Date_Debut</NomAttribut2>
  <NomAttribut3>Date_Fin</NomAttribut3>
</Table>
.....
</BaseXml>

```

Figure VI .4 : Schéma local de la base XML

```

<?xml version="1.0"?>
<BaseObjet>
<Table>
  <NomTable>medecintraitant </NomTable>
  <NomAttribut1>NSSMedecin</NomAttribut1>
  <NomAttribut2>MENom</NomAttribut2>
  <NomAttribut3>MEPreNom</NomAttribut3>
  <NomAttribut4>MESpecialite</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>PAAadresse</NomAttribut6>
  <NomAttribut7>PATelephone</NomAttribut7>
  <NomAttribut8>PADate_Naiss</NomAttribut8>
</Table>
.....
</BaseObjet>

```

Figure VI .5 : Schéma local de la base Objet

### II.1.3.2 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 Objet** : pour l'adaptateur Objet nous avons utilisé une méthode qui se charge de faire la traduction de la requête en requête applicable sur la base Objet.
- **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.

### II.1.3.3 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. Réécrire (décomposer) et optimiser les requêtes (optimisation répartie).
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 du schéma global.
2. Module de traitement de requêtes.
3. Module de fusion des résultats.

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

#### ➤ **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 allons construire le schéma global 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>MedecinG</ NomCpt >
    <NomAttribut>NSSMedecin</ NomAttribut >
    <NomAttribut>NomMedecinG</ NomAttribut >
    <NomAttribut>PrenomMedecinG</ NomAttribut >
    <NomAttribut>MESpecialite</ NomAttribut >
    <NomAttribut>MEAdresse</ NomAttribut >
    <NomAttribut>METelephone</ NomAttribut >
    <NomAttribut>MEDate_Naiss</ NomAttribut >
  </ Concept >
  < Concept >
    < NomCpt >MaladeG</ NomCpt >
    < NomAttribut >NSSPatientG</ NomAttribut >
    < NomAttribut >NomMaladeG</ NomAttribut >
    < NomAttribut >PrenomMaladeG</ NomAttribut >
    < NomAttribut >PAStatut</ NomAttribut >
    < NomAttribut >PASexe</ NomAttribut >
    < NomAttribut >PAAadresse</ NomAttribut >
    < NomAttribut >PATelephone</ NomAttribut >
    < NomAttribut >Date_NaissMaladeG</ NomAttribut >
  </ Concept >
  < Concept >
    < NomCpt >maladie</ NomCpt >
    <NomAttribute>NumeroMaladie</NomAttribute>
    <NomAttribute>NomMaladie</NomAttribute>
    <NomAttribute>Description</NomAttribute>
  </ Concept >
  ....
</schemGlobal>

```

Figure VI.6 : Fragment de schéma global

### ➤ Méta données :

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 méta donnée ?*

Les méta-données sont des "données sur les données", Elles sont décrites soit de façon littérale soit par des attributs qui décrivent les propriétés intrinsèques des entités considérées [CULL 03]. Une méta-donnée 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].

```

<?xml version="1.0"?>
<MetaDonnee>
<Synonymes>
    <SynonymeG>MedecinG</SynonymeG>
    <Synonyme>medecintraitant</Synonyme>
    <Synonyme>medecin</Synonyme>
    <Synonyme>cvmedecin</Synonyme>
</Synonymes>
<Synonymes>
    <SynonymeG>MaladeG</SynonymeG>
    <Synonyme>patient</Synonyme>
    <Synonyme>malade</Synonyme>
</Synonymes>
<Synonymes>
    <SynonymeG>NomMedecinG</SynonymeG>
    <Synonyme>MENom</Synonyme>
    <Synonyme>NomMedecin</Synonyme>
</Synonymes>
<Synonymes>
    <SynonymeG>NSSPatientG</SynonymeG>
    <Synonyme>NSSPatient</Synonyme>
    <Synonyme>NSSMalade</Synonyme>
</Synonymes>
    <SynonymeG>Date_NaissMaladeG</SynonymeG>
    <Synonyme>PADate_Naiss</Synonyme>
    <Synonyme>DateNaissMalade</Synonyme>
</Synonymes>
    ...
</MetaDonnee>

```

Figure VI .7 : Fragment de fichier méta donnée

**Remarque :**

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

#### II.1.3.4 Couche utilisateur

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.

## II.2 Architecture détaillée

### II.2.1 Architecture technique du système

Afin réaliser l'architecture précédente on propose l'architecture technique suivante :

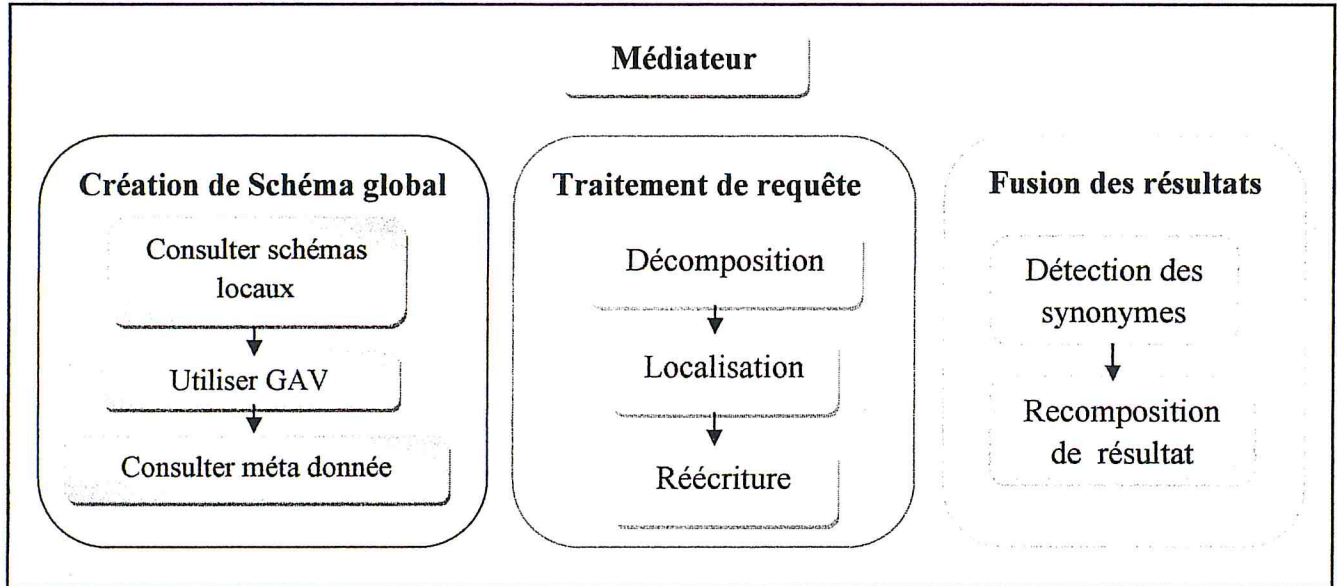


Figure VI.8 : Architecture technique du système

### II.2.2 Description des modules

#### a. Le module Création de schéma global

Afin de créer le schéma Global nous recherchons, pour chaque concept décrivant une relation ses équivalents dans les différentes bases de données à travers le fichier des métadonnées, Une fois ces concepts sont trouvés, une relation dans le schéma global portant le nom de la relation Global est créer.

Durant la deuxième étape, le même traitement est appliqué sur les concepts décrivant les attributs en gardant la relation entre les Concepts et les attributs

#### ❖ Correspondances entre schémas Global / Local ( Global as View)

L'approche GAV provient du monde de bases de données fédérées, Nous avons utilisée cette approche pour décrire les correspondances entre schémas (schéma global /schémas des sources), et consiste à définir le schéma global au niveau du médiateur en fonctions des schémas locaux des divers sources de données à intégrer ; il s'agit donc d'une approche ascendant depuis les sources vers le médiateur.

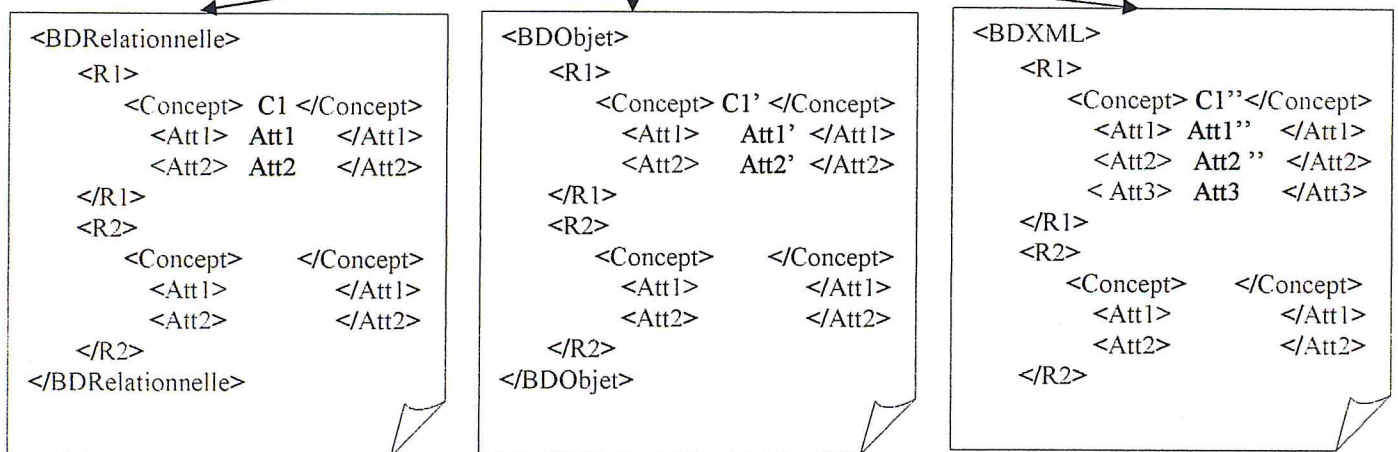


Les avantages de cette approche sont :

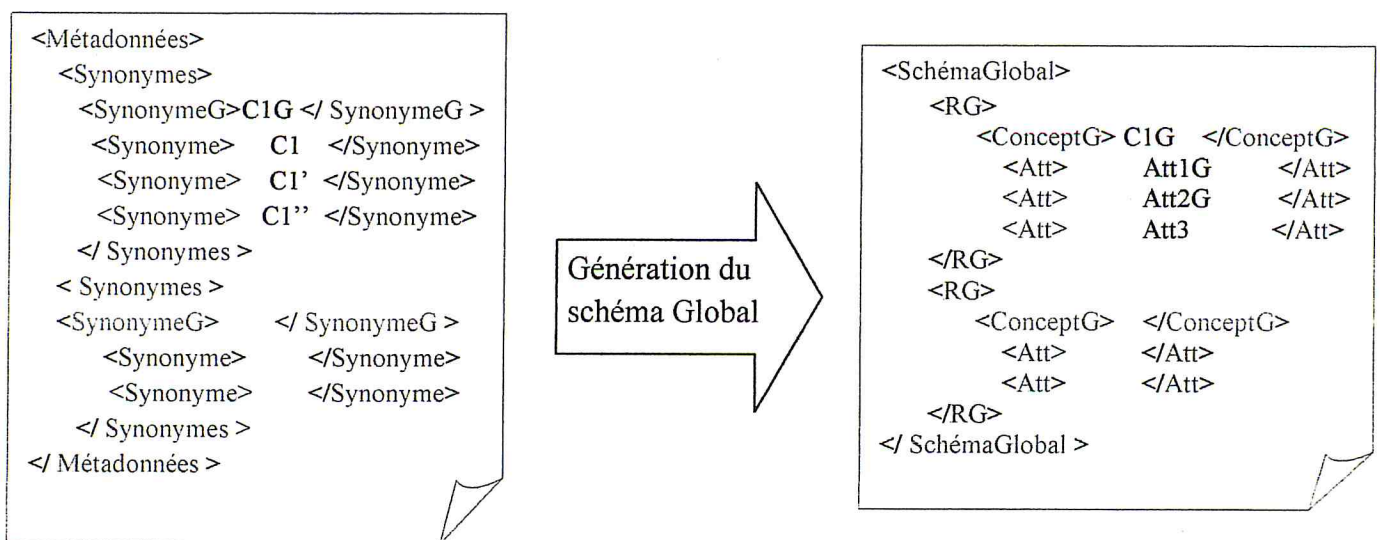
- La construction des réponses à des requêtes est simple, il suffit de remplacer les prédicats du schéma global de la requête par leur définition.
- Lorsqu'une source de données se change, ou une nouvelle source est ajoutée et doit participer au système, il suffit de mettre à jour le schéma GAV, et la phase de réécriture dans ce cas est simple.

**Exemple**

Fraction d'un Document décrivant la base de données



Fraction d'un Document des métadonnées



**Figure VI.9 : Génération du schéma Global**

❖ Algorithme de Création du schéma Global

**Algorithme : Création schéma global**  
**Entrée :** Schémas locaux, Méta données  
**Sortie :** Schéma global  
**Début**  
 Connecter aux sources de données  
 Si connexions établit **alors**  
     Consulter les schémas locaux  
     Tirer l'ensemble de concepts locaux  $E_1$   
     **Pour chaque** concept local  $CL_i$  de l'ensemble  $E_1$  **faire**  
         Chercher son synonyme  $SC_i$  dans le fichier des métras donnés  
             Si  $SC_i$  existe **alors** concept global  $CG_i$  de  $CL_i$  sera  $SC_i$   
             **Sinon**  $CG_i$  sera lui-même  
         Grouper les  $CL_i$  qui ont le même  $CG_i$   
         **Pour chaque** attribut local  $AL_i$  de l'ensemble  $E_2$  **faire**  
             Chercher son synonyme  $SA_i$  dans le fichier des métra donnée  
                 Si  $SA_i$  existe **alors** attribut global  $AG_i$  de  $AL_i$  sera  $SA_i$   
                 **Sinon**  $AG_i$  sera lui-même  
             Grouper les  $AL_i$  qui ont le même  $AG_i$   
         **Sinon** rien faire  
**Fin**

Exemple :

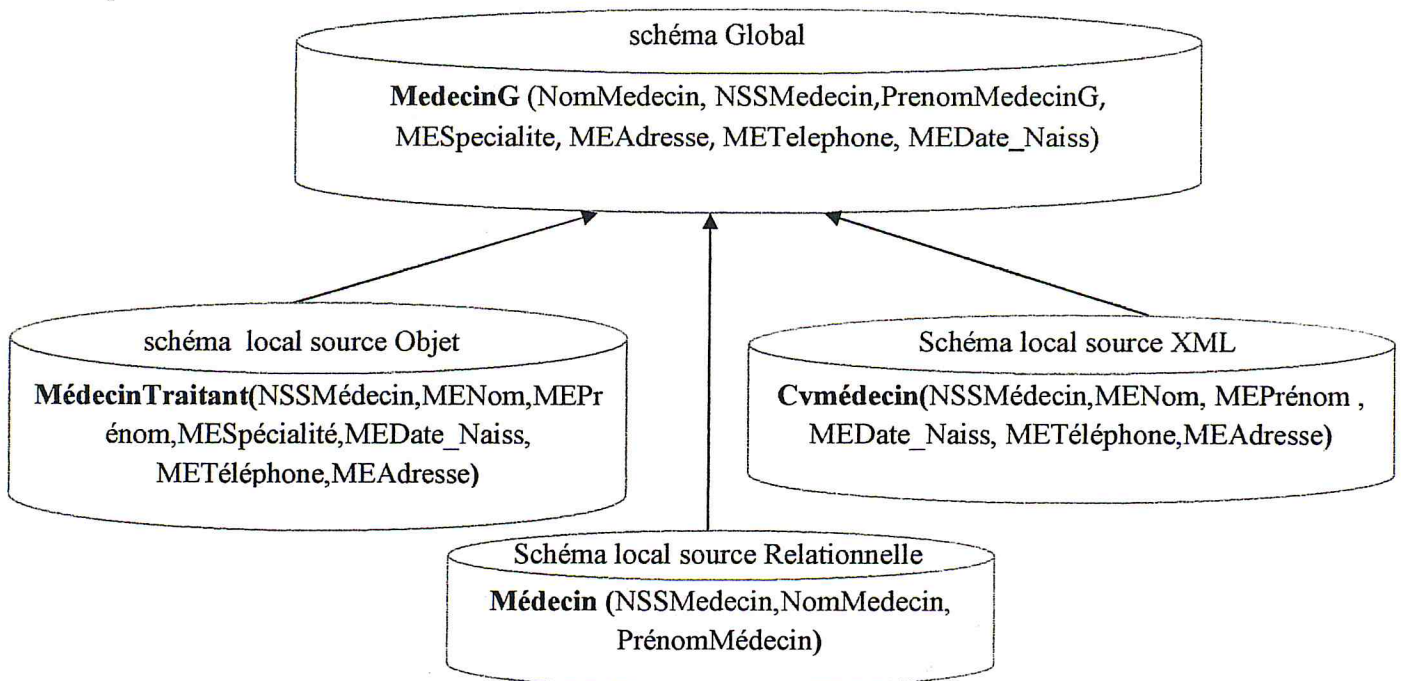


Figure VI .10 : Exemple de génération du schéma Global selon GAV

**b. Le module Traitement de requête**

Le processus traitement de requête passe par trois phases principales :

**b.1 Décomposition de requête**

Ce module est chargé de décomposer une requête globale (de l'utilisateur) en sous requêtes définies sur une relation globale.

Formellement, considérons une requête **Q** dont la syntaxe est la suivante :

```
SELECT AG FROM CG WHERE C
```

Avec:

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

$CG < CG_1, CG_2, \dots, CG_g >$  représente l'ensemble de tables (relations, ou concepts) globales,

$C < C_1, C_2, \dots, C_c >$  représente l'ensemble de conditions de sélection. Notons que chaque  $C_i$  a la structure suivante :  $< AG_i, O_i, V_i >$ , avec  $AG_i, O_i, V_i$  représentant respectivement un attribut global, une opération, et une valeur.

La requête **Q** sera donc décomposée en sous requêtes.

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

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

**Algorithme : Décomposition de requête**

**Entrée** : Requête en schéma global (RG)

**Sortie** : Sous requêtes en schéma global (SRG)

**Début**

Tirer l'ensemble des concepts locaux  $CL$  pour  $CG_i$

NombreDeSousRequête = card( $CL$ )

Créer des sous requêtes globales selon ce nombre

**Fin**



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

```
{ Select NomMedG,PrénomMedG From MedecinG where Spécialité='Généraliste' }.
```

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

Le concept global "MedecinG" a trois concepts locaux {CvMédecin, Medecin, MedecinTraitant } qui vont être utilisés pour la localisation et la réécriture.

Sous Requête 1 : **Select** NomMedG,PrénomMedG, Spécialité **From** MedecinG  
**where** Spécialité="Généraliste"

Sous Requête 2 : **Select** NomMedG,PrénomMedG, Spécialité **From** MedecinG  
**where** Spécialité="Généraliste "

Sous Requête 3 : **Select** NomMedG,PrénomMedG, Spécialité **From** MedecinG  
**where** Spécialité="Généraliste "

## b.2 Localiser les sources pertinentes :

Ce module a pour rôle d'identifier les relations locales pertinentes ainsi que leurs sources. Il a pour entrées une sous requête définie sur une relation globale (le résultat du module précédent et la relation globale concernée). Il retourne une table contenant les relations locales pertinentes et leurs sources.

### Algorithme : Localisation des sources

**Entrée :** Sous requête en schéma global (SRG)

**Sortie :** - Sous requêtes avec concept local (CL) et attributs globaux (AG)  
- La source de chaque sous requête.

#### Début

Pour chaque sous requête remplacer  $CG_i$  par  $CL_i$

Pour chaque  $CL_i$  tirer le nom de la source  $S_i$

Déterminer pour chaque sous requête local  $SRL_i$  la source local

relative

**Fin**

**Exemple :** Le résultat de la localisation des sous requêtes précédentes est ;

Sous Requête 1 : **Select** NomMedG,PrénomMedG, Spécialité **From** CvMédecin  
**Where** Spécialité="Généraliste" → Source XML

Sous Requête 2 : **Select** NomMedG,PrénomMedG, Spécialité **From** MedecinTraitant

**Where** Spécialité="Généralist " → Source Objet

Sous Requête 3 : **Select** NomMedG,PrénomMedG , Spécialité **From** Medecin

**Where** Spécialité="Généralist " → Source Relationnelle

### b.3 La réécriture des requêtes :

La réécriture des requêtes est simple, une requête sur le schéma global se traduit en termes de schémas de sources en remplaçant les vues par leurs définitions (dépliage des requêtes). C-à-d que chaque attribut global va être remplacé par son équivalent en local.

Le résultat de ce module sera des sous requêtes en schémas locaux. La requête dépliée est évaluée (exécuter) par la suite sur les sources.

#### ❖ Algorithme de réécriture de sous Requêtes

##### Algorithme : Réécriture de sous Requêtes

**Entrée :** Sous requêtes avec concept local (CL) et attributs globaux (AG)

**Sortie :** Sous requêtes en schéma local (SRL)

##### Début

Pour chaque sous requête **Faire**

Pour chaque AG<sub>i</sub> **Faire**

tirer l'ensemble d'attribut locaux (AL)

pour chaque AL<sub>i</sub> voir CL<sub>j</sub> **Si** i = j

// vérifier la correspondance Concept local / attribut local (AL<sub>i</sub> / CL<sub>j</sub>)

**Alors** réécrire la SRL<sub>k</sub> avec AL<sub>i</sub> et CL<sub>i</sub>

**Sinon** // c-à-d cet attribut n'existe pas sur la source j

Ne pas réécrire AL<sub>i</sub> pour SRL<sub>k</sub> remplacer le par la valeur *NULL*

Pour chaque condition C<sub>i</sub> **Faire**

Pour chaque AG<sub>i</sub> **Faire**

tirer l'ensemble d'attribut locaux (AL)

pour chaque AL<sub>i</sub> voir CL<sub>j</sub> **Si** i = j

// vérifier la correspondance Concept local / attribut local (AL<sub>i</sub> / CL<sub>j</sub>)

**Alors** réécrire la SRL<sub>k</sub> avec la condition C<sub>i</sub>, AG<sub>i</sub>, O<sub>i</sub>

**Sinon** // c-à-d cet attribut n'existe pas sur la source j donc ne pas

// appliquer la condition

Eliminer la sous requête car elle répond pas a la condition de sélection

**Fin**

**Exemple :** Le résultat de la réécriture des sous requêtes précédentes est ;

- Sous Requête 1 : **Select** NomMed , PréMedG, Null **From** CvMédecin

**Where** Spécialité="Généraliste "→ Cette sous requête sera négliger car elle répond pas à la condition de sélection, l'attribut Spécialité n'existe pas sur la source XML.

- Sous Requête 2 : **Select** MedNom , MedPré, Spécialité **From** MedecinTraitant **Where**

Spécialité="Généraliste "→ Cette sous requête va être exécuter sur la source Objet.

- Sous Requête 3 : **Select** MedNom, MedPré , Null **From** Medecin

**Where** Spécialité="Généraliste"→ Cette sous requête sera négliger car elle répond pas à la condition de sélection, l'attribut Spécialité n'existe pas sur la source Relationnelle .

### c. 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

**Fait**

**Fin**



## Introduction

Ce chapitre présente un aperçu général sur l'organisation du code source de notre application pour bien démontrer les outils de développement discuté dans le chapitre de l'étude technique, Ensuite nous présentons notre application implémenté dans le domaine médical.

Dans le domaine médical un des défis majeurs est l'archivage, l'accès et l'analyse de plusieurs bases de données hétérogènes qui contiennent des renseignements portant sur les patients. Les sources de données médicales sont hétérogènes (bases de données cliniques, données de laboratoires...), en plus ces bases de données existantes sont implémentées depuis longtemps et elles ne communiquent pas ensembles. Afin de faciliter l'accès à ces données, nous présenterons une approche d'intégration de ces bases de données médicales hétérogènes.

## I. Implémentation

### I.1 Environnement de développement :

La programmation peut se faire pour des exemples simples avec le compilateur java 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.

### I.2 Le passage du modèle de conception au modèle relationnel

Certaines classes du diagramme de classe représente une abstraction des fichiers XML utiliser tout au long du processus de construction de schéma Global. Et d'autre nécessite La construction de tables relationnelles associées.

Pour la transformation des classes nous avons les règles suivantes :

- Règle 1 : Chaque classe du diagramme UML devient une table, il faut donc choisir un attribut de la classe pouvant jouer le rôle d'identifiant, et si aucun attribut ne convient, il faut ajouter un de telle sorte que la table dispose d'une clé primaire.
- Règle 2 : pour les associations un-à-plusieurs il faut ajouter un attribut de type clé étrangère dans la table fils de l'association
- Règle 3 : pour les associations plusieurs a plusieurs , l'association devient une table dont la clé primaire est composé par la concaténation des identifiant des classes connecté a l'association.
- Règle 4 : et pour les associations un-à-un permet d'éviter les valeurs NULL dans la base de données

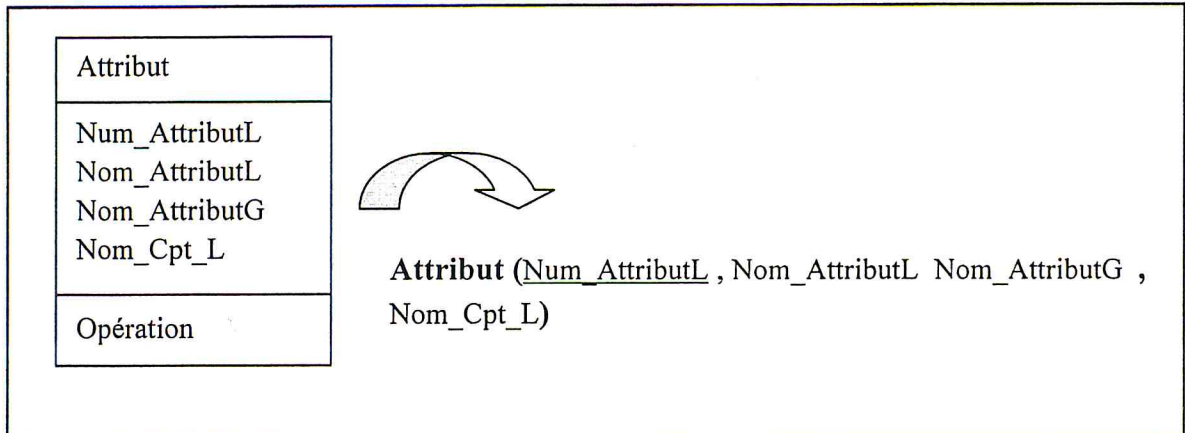


Figure VII .1 : Exemple de passage d'une classe à une table

## II. Test

L'application que nous avons développée se présente sous plusieurs aspects, selon les besoins de l'utilisateur.

### II.1 Fenêtre d'Accueil

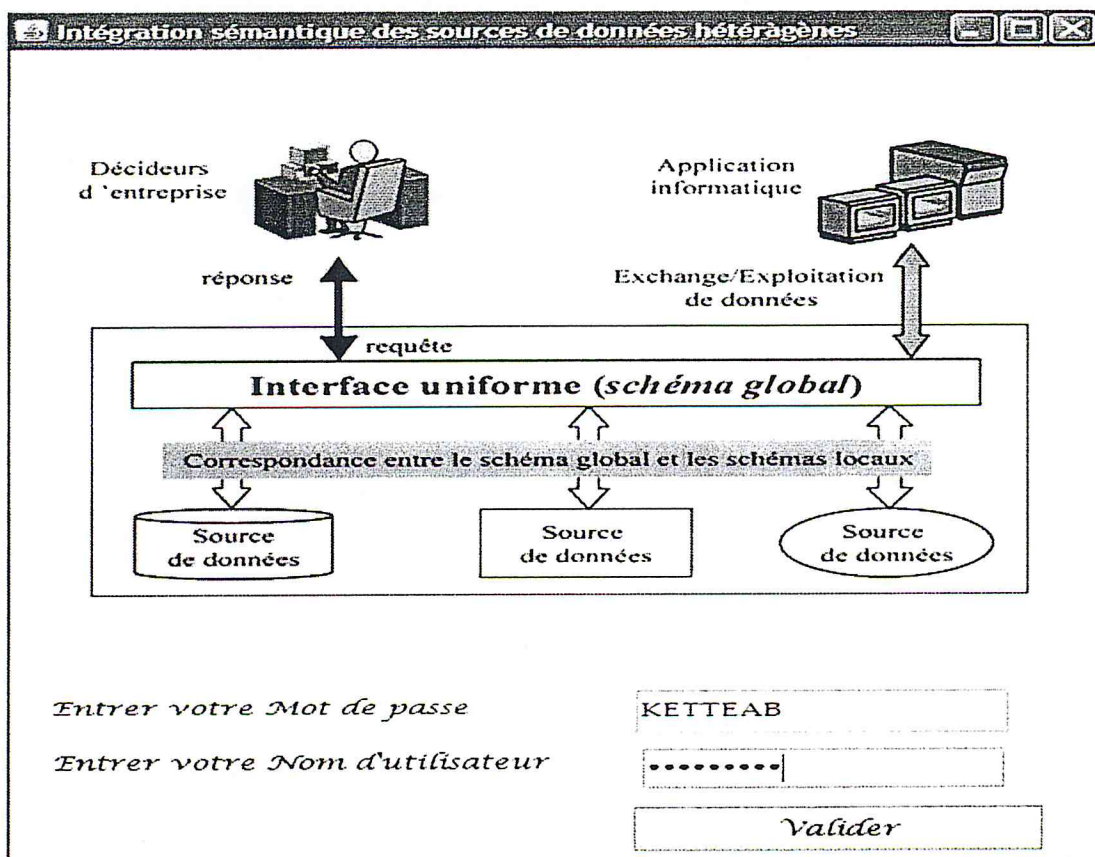


Figure VII.1 Fenêtre d'accueil

## I.2 Fenêtre Menu

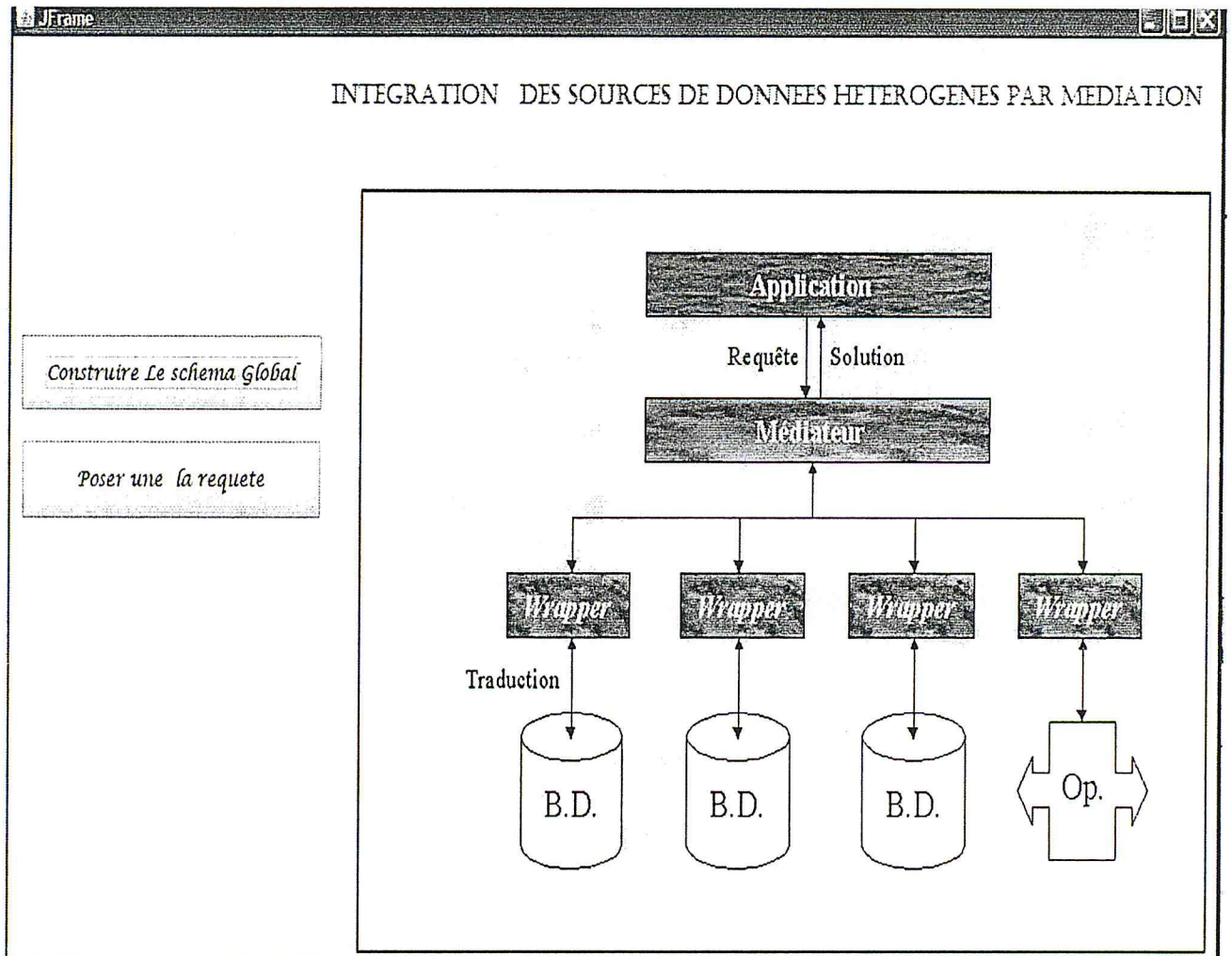


Figure VII.2 Fenêtre Menu

Cette interface donne le choix à l'utilisateur de créer le schéma global ou de poser sa requête.



## I.3 Fenêtre Création Schéma Global

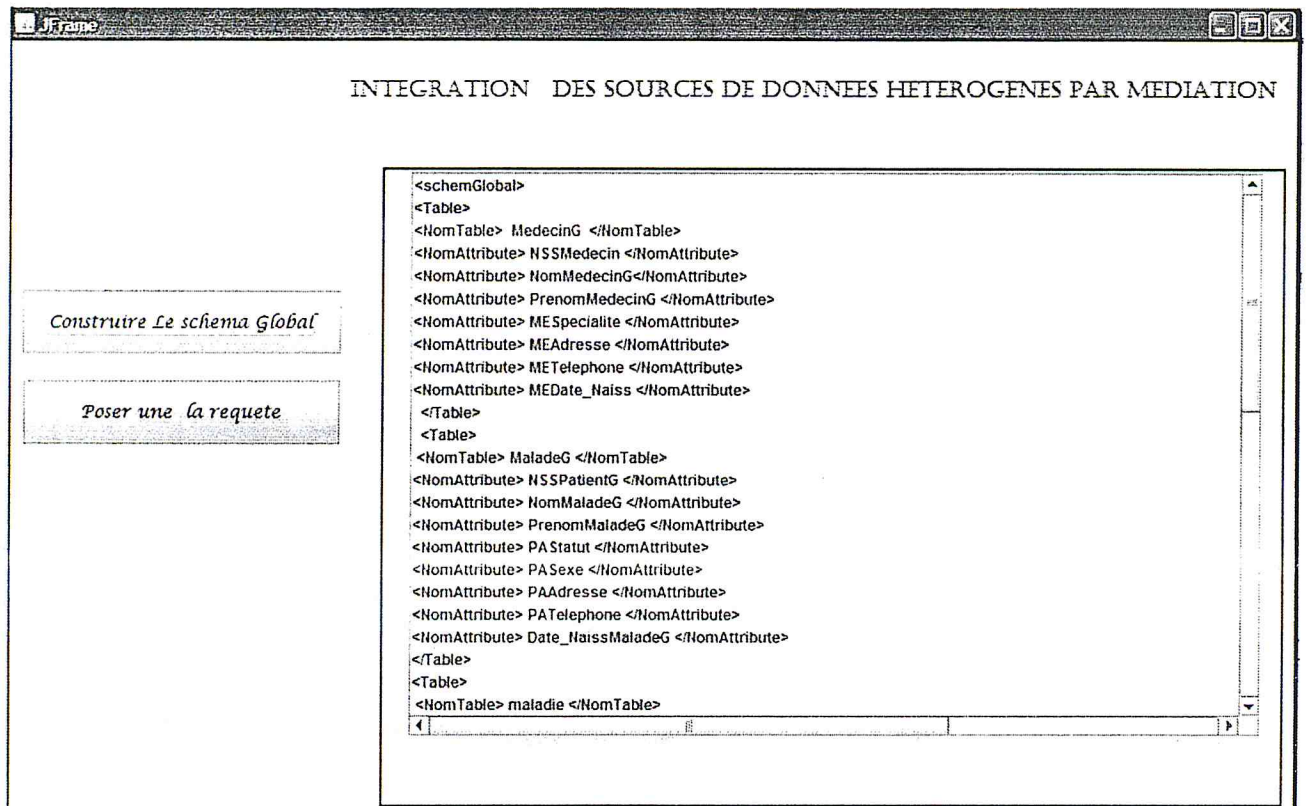
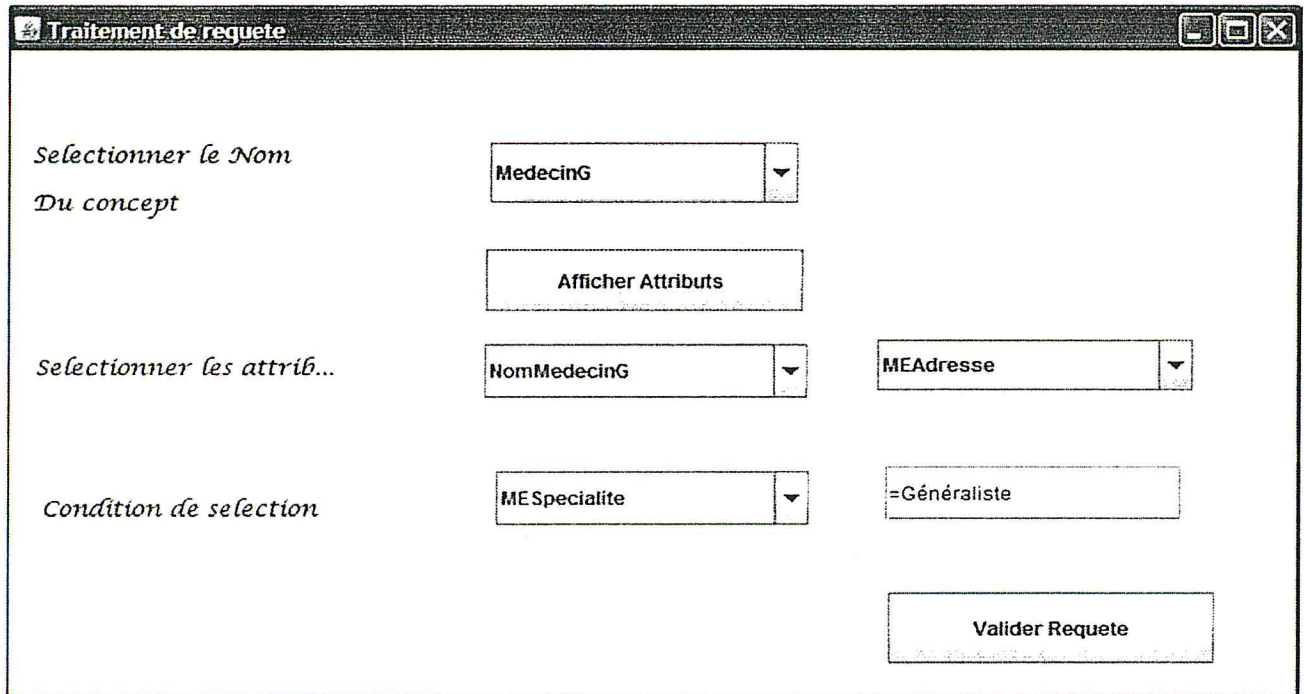


Figure VII.2 Fenêtre Création Schéma Global

#### I.4 Fenêtre Traitement de requête



The screenshot shows a window titled "Traitement de requete" with standard window controls (minimize, maximize, close) in the top right corner. The interface is organized into several sections:

- Selectionner le Nom Du concept:** A dropdown menu currently displaying "MedecinG".
- Afficher Attributs:** A button located below the first dropdown.
- Selectionner les attrib...:** Two dropdown menus. The first displays "NomMedecinG" and the second displays "MEAdresse".
- Condition de selection:** Two input fields. The first is a dropdown menu displaying "MESpecialite", and the second is a text box containing "=Généraliste".
- Valider Requete:** A button located at the bottom right of the window.

Figure VII.3 Fenêtre Traitement de requête

#### Conclusion

Dans cette partie nous avons présenté l'environnement de déroulement pour notre développement logiciel, dont le but est d'apporter une solution et répondre aux besoins des utilisateurs en leur offrant toute une ergonomie d'utilisation et de manipulation de ce système par les interfaces graphiques.

---

# *Conclusion Générale*

---

## *Sommaire*

- 1. Conclusion général*
- 2. Perspectives*



## 1. Conclusion Générale

De nos jours, de larges volumes de données sont disponibles publiquement, les types de données sont divers, et les ressources très nombreuses. Souvent les données provenant de différentes ressources se complètent. L'architecture d'un système d'intégration d'informations doit permettre leur intégration donnant ainsi l'impression à l'utilisateur qu'il utilise un système homogène, et de lui fournir une vue unifiée de ces données. Dans ce contexte, un problème essentiel de l'intégration est de gérer l'hétérogénéité des différentes sources de données. L'intégration de sources de données nécessite des outils qui prennent en charge ces problèmes et qui fournissent un accès efficace à un ensemble de sources de données.

On peut distinguer deux grandes approches pour l'intégration de sources d'information: L'approche *virtuelle* et L'approche *matérialisé*. L'approche *virtuelle* consiste à fonder l'intégration d'informations sur l'exploitation de vues abstraites décrivant le contenu des différentes sources d'information. Les données ne sont pas stockées et ne sont accessibles qu'au niveau des sources d'information. L'approche *matérialisé* consiste à voir cette intégration comme la construction de bases de données réelles, regroupant les informations pertinentes pour les applications considérées.

Afin d'exploiter au mieux les ressources des bases de données et permettre aussi une recherche efficace, notre système envisage un mécanisme qui s'occupera de l'intégration des sources de données différentes ayant une structure définie au préalable.

Nous avons opté, dans la solution proposée, pour l'approche « Médiation », qui a été étudié dans l'état de l'art, pour les motifs suivants :

- La médiation est une approche souple et elle fournit le plus d'autonomie aux bibliothèques distantes.
- L'absence d'incidence de mise à jour des données sur le système intégré. Cela permet d'éviter le problème de maintenance des données. En effet la médiation permet de construire un système d'interrogation de sources de données sans toucher aux données qui restent stockées dans leurs sources d'origine.

- Disponibilité accrue des données en cas de pannes des serveurs.
- Support de l'hétérogénéité des sources.

Dans ce mémoire nous avons présenté une architecture d'intégration de données qui repose sur l'approche médiateur, notre système offre les fonctionnalisées suivantes :

- ✓ Intégration sémantique on utilisant les métas donnés ;
- ✓ Automatisation de la création de schéma global, qui représente une vue global sur les schémas sources de données.
- ✓ Traitement de requêtes.

## 2. Perspectives

La solution qu'on a présenté et le projet réalisé dans ce mémoire pour la résolution de problème d'intégration des sources de données hétérogènes 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 <sup>ns</sup> ~~semi~~ structurée ( fichiers textes) ;
  - ✓ Rendre le système plus extensible en traitant le cas de la distribution des sources ;
- L'optimisation des requêtes et des résultats, et le temps de réponse.

## Bibliographie

- [BOU 05] Thèse par Mohand Salah Boumediene, Définition d'un système générique de partage de données entre systèmes existants.
- [Bous 08] Amel Boussis, Intégration de sources de données à base ontologique dans un environnement P2P, Thèse de magistère. L'institut national d'informatique 2008.
- [Bressan et al. 99] Bressan S., Madnick S.E., Siegel M.D., "Context Interchange: New features and formalisms for the intelligent integration of information", ACM Transactions on Information Systems, Volume 17, N°3, p.270-298.
- [BUS 99] Busse S., Kutsche R.-D., Leser U. and Weber H. Federated Information Systems: Concepts, Terminology and Architectures, Technical Report n°99-9, Technical University of Berlin, 38 p. 1999.
- [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.
- [GAR 06] Georges GARDARIN "Médiation de Données", 2006.
- [GIO 1992] Wiederhold Gio. "Mediators in the Architecture of Future Information Systems" *IEEE Computer*, March 1992.
- [HAKI 03] Farshad Hakimpour, Using Ontologies to Resolve Semantic Heterogeneity for Integrating Spatial Database Schemata, Mathematisch naturwissenschaftlichen Fakultät der Universität Zürich, Zürich 2003
- [Jean Jack] Jean Jack Le COZ , DB40.
- [JER 08] JEROM BOUGEAUT, java la maîtrise édition 2008.
- [JAOUA 10] Jaouaf mohamed amine, Génération automatique des requêtes de médiation dans un context relationnel 2010.
- [LIT 89] Witold Litwin, Abdelaziz Abdellatif, A. Zeroual, B. Nicolas, Ph. Vigier: MSOL: A Multidatabase Language.59-101 1989.
- [MOHAND 2004] Mohand-Saïd Hacid et Chantal Reynaud "L'intégration de sources de données", 2004.
- [PORC] pierre Gérard.Processus de développement logiciel (Université de Paris 13 )2008



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

[ROU 02] Wiederhold G, Mediators in the architecture of future information systems. IEEE computers, 25(3), p 38J49, March

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

[VODISLAV 2003] Dan VODISLAV "Intégration de données et XML", 2003.

[WIE 92] Wiederhold G. Mediators in the architecture of future information systems. IEEE computers, March 1992.

[ALIMA 09] Intégration des sources de données hétérogènes dans le processus de construction d'un DATA Warehouse, ALIMAZIGHI Hadjer, ATTAF Sarah. 2008-2009.