F.S.D .....N° D'ordre .....

Université Saad DAHLAB de Blida

Faculté des Sciences

Département d'Informatique

Mémoire présenter par :

M^elle METIDJI Sanaa
M^elle MECHIEKH Fatma

En vue d'obtenir le diplôme de Master

Domaine: MI

Filière: Informatique

Spécialité: Informatique

Option: Ingénierie de logiciel

Sujet :

Conception et réalisation d'un médiateur de métadonnées : XML, RDF,RuleML

Soutenue le: , **devant le jury composé de :**

M : boumahdi fatma	Président
M : nahal	Rapporteur
M :yakhlef hadjer	Examinateur
M^me M .FAREH	Promotrice

MA-004-92-1

REMERCIEMENT

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

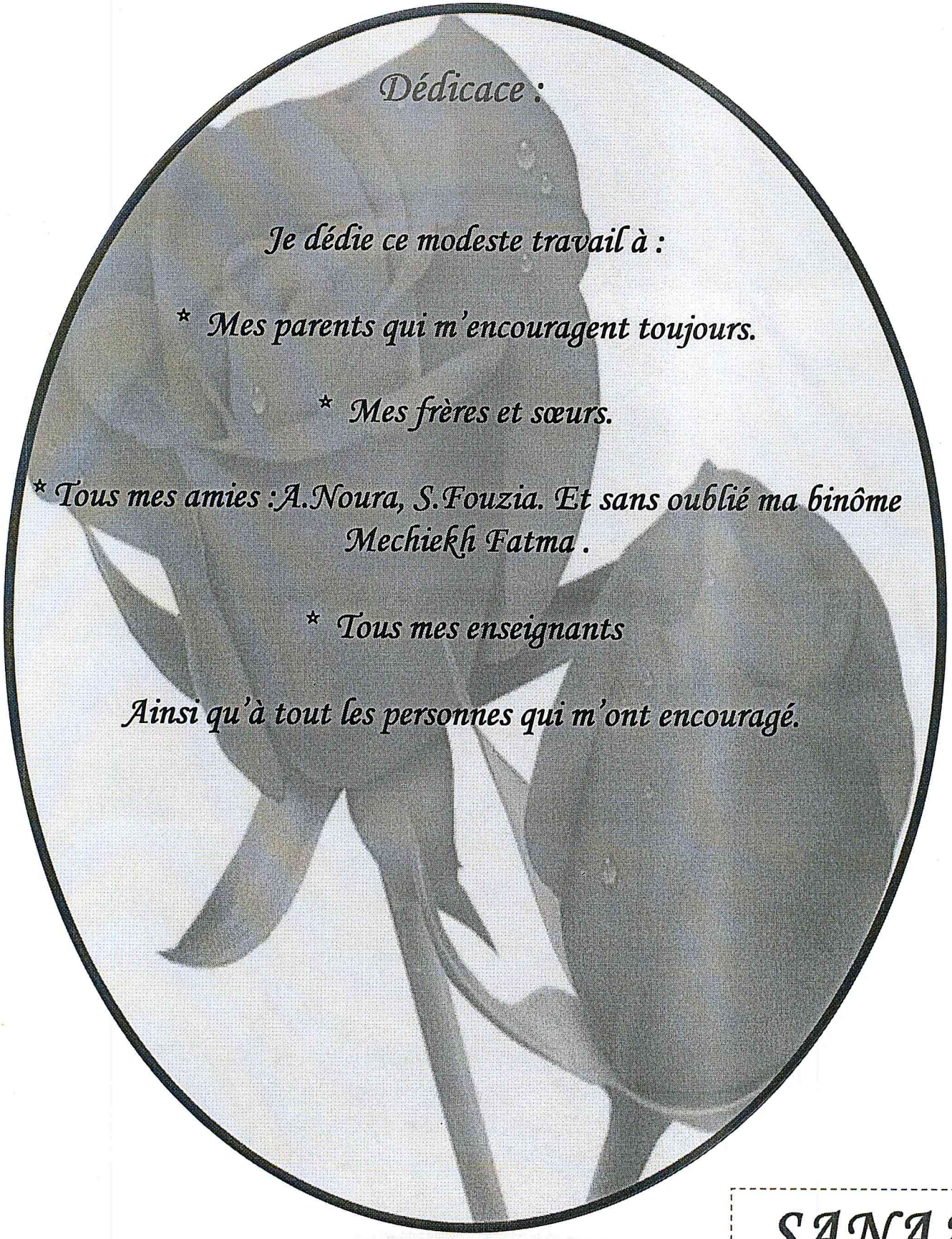
Nous tenons à remercier sincèrement tous les personnes qui ont apporté leur contribution à l'aboutissement de ce projet .

Nous remercions notre promotrice M^me^ FAREH pour son soutient et ses précieux conseils.

Nous remercions aussi tous les enseignants de département d'informatique pour les efforts consacré pour nous transmettre le savoir

Notre reconnaissance s'adresse à toutes les personnes qui chères et ceux qui nous ont aidée de prés ou de loin.

En fin, on tient à remercier les membres de jury qui vont faire l'honneur d'apprécier notre travail.



Dédicace :

Je dédie ce modeste travail à :

* Mes parents qui m'encouragent toujours.

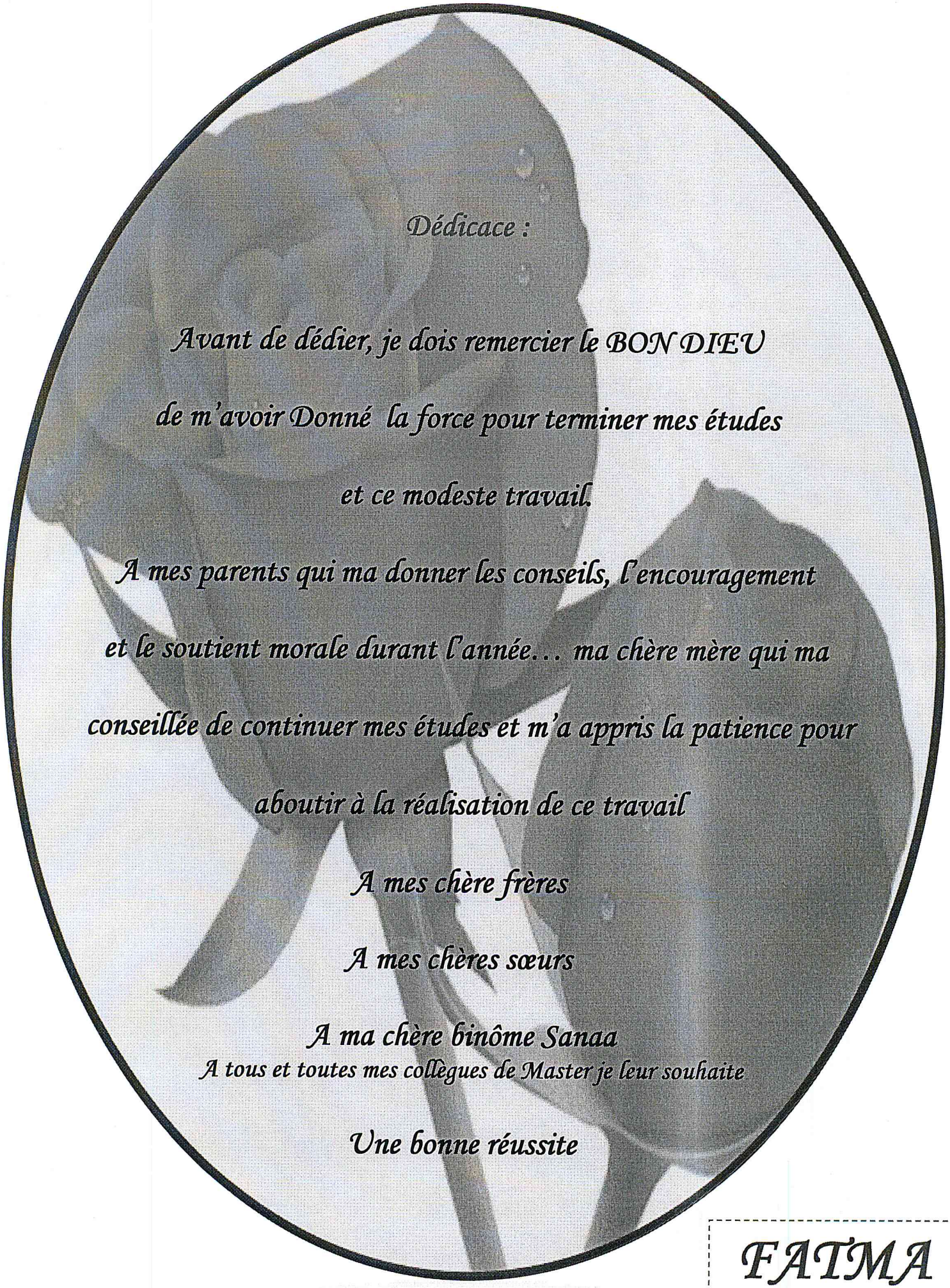
* Mes frères et sœurs.

* Tous mes amies :A.Noura, S.Fouzia. Et sans oublié ma binôme Mechiekh Fatma .

* Tous mes enseignants

Ainsi qu'à tout les personnes qui m'ont encouragé.

SANAA



Dédicace :

Avant de dédier, je dois remercier le BON DIEU

de m'avoir Donné la force pour terminer mes études

et ce modeste travail.

A mes parents qui ma donner les conseils, l'encouragement

et le soutient morale durant l'année… ma chère mère qui ma

conseillée de continuer mes études et m'a appris la patience pour

aboutir à la réalisation de ce travail

A mes chère frères

A mes chères sœurs

A ma chère binôme Sanaa

A tous et toutes mes collègues de Master je leur souhaite

Une bonne réussite

FATMA

Résumé

La masse énorme d'informations disponibles sur des sources de données distribuées et hétérogènes, nécessite une politique ou un plan de recherche, de sélection et d'analyse, de plus en plus performants, pour permettre à l'utilisateur de localiser et extraire précisément l'information désirée d'une manière simple et efficace.

Notre travail consiste à proposer une architecture de médiation des sources de métadonnées hétérogènes représentées par le XML, RDF et RuleML, assurant à l'utilisateur la transparence de l'hétérogénéité. Ceci en intégrant des données de structures et de natures fondamentalement différentes et en permettant la décomposition d'une requête faisant intervenir plusieurs sources en des requêtes spécifiques à ces sources, puis savoir recomposer le résultat.

Mots Clés: Métadonnée, Médiateur, XML, RDF,Ruleml,Traitement de requêtes, ontologie.

Abstract

The enormous mass of information available on distributed data sources, heterogeneous, requires a policy or plan of research, selection and analysis, more efficient, to allow the user to precisely locate and extract the desired information in simple and effective.

Our job to propose a mediation architecture of heterogenous sources of metadata represented by XML,RDF and Ruleml, providing the user the transparency of the heterogeneous .This by integrating data structures and fundamentally different natures and allowing the decomposition of a query involving multiple source of applications specific to these sources, then recompose know the result.

Keywords: Metadata , Mediator, XML,RDF,Ruleml, Query processing, Ontology

ملخص

الكتلة الهائلة من المعلومات المتاحة عن مصادر البيانات الموزعة والغير متجانسة ، يتطلب سياسة أو خطة للاختيار، والبحث والتحليل، وأكثر كفاءة، للسماح للمستخدم لتحديد بدقة واستخراج المطلوب من المعلومات

بطريقة بسيطة وفعالة.
مهمتنا هي أن تقترح هندسة وساطة من المصادر متجانسة من البيانات الفوقية ممثلة بـXML، RDFوRuleML، وتوفير للمستخدم شفافية و تجانس .هذا عن طريق دمج هياكل البيانات وطبائع مختلفة جوهريا والسماح بتفكيك الاستعلام الذي ينطوي على مصادر متعددة للتطبيقات محددة لهذه المصادر، أعاده التركيب ثم تعرف النتيجة.
كلمات البحث: البيانات الفوقية، المحقق، XML، معالجة الاستعلام، الأنطولوجيا.

Sommaire

Chapitre 3: Conception du système médiateur

Chapitre 4: Implémentation et Validation du système

Liste de figure

Liste de tableaux

Introduction générale

Introduction générale

1. Contexte

De nos jours, les systèmes multi-source sont de plus en plus développés.Ils sont définis comme l'intégration de plusieurs sources hétérogènes .

Parmi ces systèmes d'informations, nous distinguons les entrepôts de données, les systèmes d'informations basés sur le web, les systèmes de bases de données fédérées, ou encore les systèmes de médiation.

Notre travail se focalise principalement sur les systèmes de médiation dans un contexte des métadonnées par les modèles XML, RDF, et RuleML.

Les métadonnées permettent de donner du sens au contenu des sources de manière à ce que leur localisation et interrogation soient plus aisées et plus pertinentes.

2. Problématique

Intégrer les sources de métadonnées dans le but de fournir aux utilisateurs une interface d'accès uniforme est une tâche difficile. Cette difficulté concerne trois aspects :

- ✓ l'hétérogénéité des données
- ✓ l'autonomie des sources
- ✓ l'évolution des sources.

La problématique première des systèmes de médiation est l'hétérogénéité des données et l'hétérogénéité des schémas qu'il est nécessaire de résoudre pour mettre en correspondance les sources et autoriser l'interrogation et la réponse aux requêtes de façon transparente. L'hétérogénéité des données est due au fait que deux bases de données n'utilisent pas le même vocabulaire ou référentiel pour représenter une même donnée.

De manière générale, la problématique peut se résumer dans l'hétérogénéité de métadonnées 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.

3. Objectif

Réaliser un système de médiation qui permet d'accéder à des données à travers une interface uniforme, sans se soucier de leur structure ni de leur localisation.

Notre objectif est de concevoir et de réaliser un système de médiation sémantique de métadonnées hétérogènes, présentées par les modèles XML, RDF et RuleML pour permettre à

un simple utilisateur de trouver des réponses à ses requêtes en lui cachant l'hétérogénéité syntaxique et surtout sémantique de ces sources.

3. Organisation du mémoire

Ce mémoire est organisé autour de quatre chapitres :

- **Le chapitre I : Etat de l'art sur la médiation des données**

Dans ce chapitre nous avons parlé de système d'intégration de données et ces problèmes ainsi que les différents conflits de données , on a basé sur l'approche médiateur de système d'intégration.

- **Le chapitre II : Les modèles des sources de métadonnées : XML, RDF et RuleML**

Dans ce chapitre on a définit les différents types de métadonnées, ainsi que les trois modèles de sources de métadonnées (XML, RDF, RuleML) que nous avons utilisé pour représenter ces dernières.

- **Le chapitre III : Conception du système d'intégration**

Ce chapitre comporte la conception de notre système de médiation par le langage de modélisation UML, ainsi qu'une architecture générale de notre système.

- **Le chapitre IV : Implémentation et Test du système**

Dans ce chapitre nous décrirons les différents outils utilisées dans notre projet, ensuite nous présenterons une démonstration globale de différentes taches à accomplir par notre application.

Chapitre I

la médiation des données: Etat de l'art

1. Introduction

L'expansion du nombre de sources d'information disponibles sur le Web et la quantité de données gérées au sein des bases de données des entreprises a rendu indispensable l'intégration de données provenant de différentes sources. Ces sources de données sont le plus souvent réparties, autonomes et hétérogènes.

Les sources d'informations sont réparties : de plus en plus d'informations sont créées partout dans le monde et publiées sur le Web, de nombreuses entreprises ont des ramifications dans plusieurs pays et les états décentralisent leurs administrations.

Elles sont autonomes car les sources de données sont conçues par différentes personnes, à différents moments et pour répondre à différents besoins applicatifs.

Enfin, les sources d'informations sont hétérogènes : des logiciels différents sont utilisés pour créer et gérer les données (ex : Oracle, MySQL,… etc.), les données sont publiées dans des formats divers (ex : HTML, PDF, images,… etc.) et des modèles de données différents sont utilisés pour les représenter (ex : modèles relationnel, objet, semi-structuré,… etc.).

2. Systèmes d'intégration de données

2.1. Définition

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 **[1]**.

Il existe principalement deux approches pour d'intégration des données :

-l'intégration matérialisée de données, où la vue unifié des données est matérialisée et les données sont rapatriées des sources d'origine et stockées dans un entrepôt de données,

- l'intégration virtuelle de données, où la vue unifié est virtuelle et les données restent stockées dans les sources d'origine. L'architecture type pour l'intégration virtuelle de données est l'architecture médiateur .

2.2. Différentes classes de conflits

Les conflits des données sont issus des différences de modélisation et d'interprétation des entités du monde réel par les concepteurs de système d'information.

Plusieurs travaux ont proposé des taxonomies des conflits de données **[2-5]**. Nous allons présenter une classification détaillée des conflits qui peuvent se présenter lorsqu'on intègre des sources de données hétérogènes. Cette classification est inspirée de celle proposée dans **[6]**, elle permet d'illustrer les différents conflits possibles. Les différents conflits de données sont classifier en trois niveaux: les conflits syntaxiques, les conflits structurels (schématiques) et les conflits sémantiques. Nous décrivons chaque niveau de ces conflits.

2.2.1. Conflits syntaxiques

Les conflits syntaxiques sont liés au modèle de données utilisé pour représenter les informations. Une même information peut être représentée par différents modèles avec une syntaxe et des concepts différents. De nombreux modèles de représentation existent il en résulte de très nombreux conflits syntaxiques. Les solutions d'interopération résolvent ces conflits syntaxiques en définissant un modèle commun de données **[2]**.

2.2.2. Conflits structurels (ou schématiques)

Les conflits structurels sont liés, aux choix de conception et la manière de représenter les informations pour chaque système. Ces conflits apparaissent lorsque le même élément du monde réel est perçu avec des structures différentes. Par exemple **[7]** un concessionnaire automobile modélisera le propriétaire d'un véhicule avec un sous ensemble d'attributs dans une relation Véhicule alors que le service de gestion des cartes grises modélisera un propriétaire de véhicule avec une relation Personne liée par une relation de dépendance à une table véhicule.

Il existe également des conflits de généralisation ils sont produits par des différences dans les liens de généralisation / spécialisation entre entité. Par exemple un système ne peut utiliser qu'un seul concept pour représenter tous les types de routes

tandis qu'un autre emploiera deux concepts pour distinguer les entités autoroute et route départementale liées à un concept plus général route.

Des conflits de type de données peuvent aussi intervenir lorsqu'une même valeur est représentée par des types différents. Par exemple la date de construction d'un produit peut être représenté par une chaîne de caractères « Janvier 2005 » ou par un type composé de deux champs mois et année 01/2005.

2.2.3. Conflits sémantiques

Les conflits sémantiques proviennent des hétérogénéités de description des informations. Deux grandes catégories de conflits sémantiques**[6]**: Les conflits taxonomiques sont liés aux hétérogénéités de vocabulaire et les conflits de valeurs sont liés à l'utilisation et au codage de la valeur d'une donnée.

Les conflits taxonomiques sont issus de l'autonomie des systèmes à nommer et à classifier les entités du monde réel. Ils concernant en particulier les problèmes de synonymie (des termes différents expriment la même réalité), d'homonymie (deux termes identiques expriment des réalités différentes), de polysémie (un terme change de signification en fonction du contexte), d'hyperonyme et d'hyponyme (un terme peut être plus général ou plus spécifique qu'un ou plusieurs autres termes)**[6]**.

Les conflits de valeurs sont liés à la marnière de coder la valeur d'une entité du monde réel dans différents systèmes. Des valeurs différentes d'un même information peuvent être considérées comme similaires selon le contexte.

3. Médiation de données

3.1 Définition de la médiation

Le terme **médiation** a été utilisé dans plusieurs domaines. La définition la plus couramment référencée dans la littérature est celle de G. Wiederhold dans **[8,9]**, où il définit la médiation comme "*une couche intergicielle intelligente de services dans des systèmes d'information, liant des ressources de données aux applications*". Ainsi, la médiation est une couche intermédiaire (voir Figure 1.1) qui propose des services informatiques permettant de relier des ressources d'information (bases de données, services web, appareils communicants, etc.) aux applications (services web, applications

d'entreprise, outils de supervision, etc.) et qui résoudre des problèmes tels que l'hétérogénéité, la distribution des données ou la sélection des sources de données. Cette couche est généralement constituée de chaînes de médiation
Formées par d'un élément de base appelé médiateur.

Un **médiateur** comme défini par G. Wielderhold, est *un module logiciel qui exploite des connaissances sur certains ensembles ou sous-ensembles de données pour créer de l'information pour une couche supérieure d'applications* **[9]**.

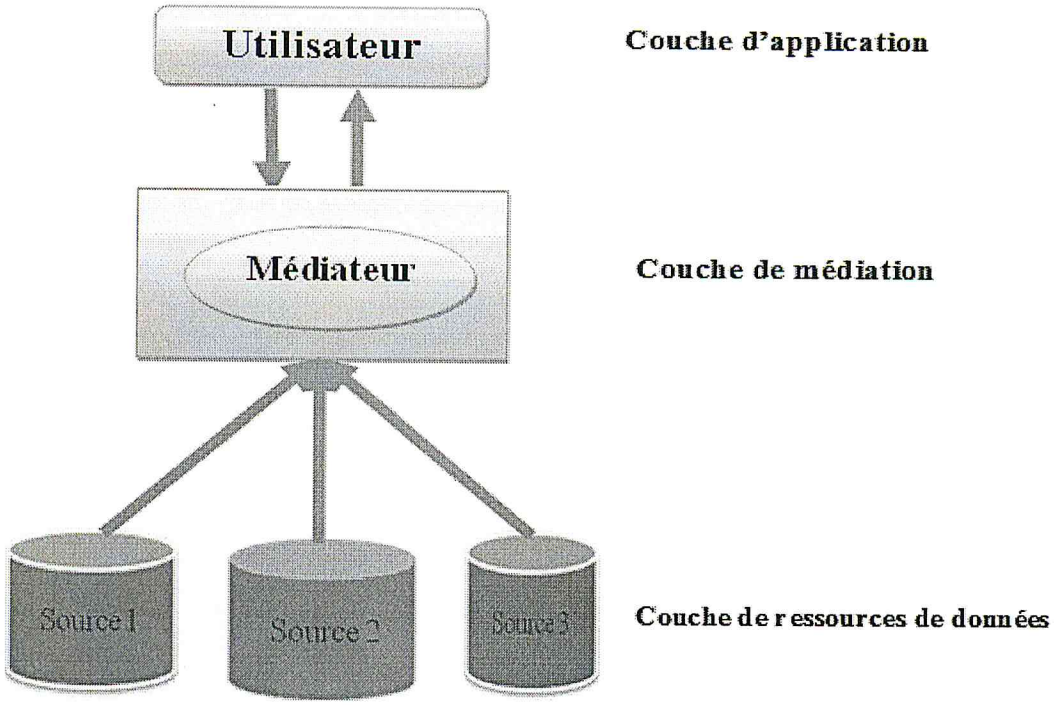


Figure. 1.1: Couche de médiation [8,9]

3.2 Importance de la médiation

L'objectif principal de la médiation est l'abstraction des données pour en extraire la connaissance du domaine **[8]** afin de faciliter la prise de décision, en fournissant des opérations telles que :

– Sélection de données

– Transformation d'un type de données en un autre

– Intégration de différentes sources de données

– Sélection de source de données lorsqu'il y a plusieurs sources disponibles.

4. Architecture de médiation

Dans l'approche médiateur, l'intégration de données est fondée sur la définition d'un schéma global unifiant les schémas hétérogènes des sources à intégrer et sur la description homogène et abstraite, du contenu des sources par des vues. Une première proposition d'architecture médiateur proposée par Gio Wiederhold **[8]** représentée dans la Figure 1.2.

Cette architecture se compose de trois niveaux :

1. Le niveau source : comporte les différentes sources de données. Les sources de données communiquent avec le médiateur à l'aide d'*adaptateur* (*wrapper*) publiant de manière homogène les données de la source.

L'*adaptateur* est chargé de traduire une requête exprimée dans le langage du médiateur en une requête exprimée dans le langage de la source, faire évaluer la requête par la source et renvoyer les résultats au médiateur.

2. Le niveau médiateur : comporte un ou plusieurs médiateurs permettant d'intégrer les données transmises par les *adaptateurs* des sources. Il s'occupe de faire l'interface entre une requête utilisateur et les sources évaluant la requête suivant le modèle *décomposition, recomposition, optimisation.*

Il présente les données de manière homogène et centralisée à la couche supérieure, résolvant le problème d'hétérogénéité et de distribution des données.

3. Le niveau client : comporte les applications clientes pour accéder aux médiateurs (i.e. navigateur, interface graphique, API cliente).

L'ajout d'une nouvelle source à un système de médiation entraîne la définition d'un nouvel *adaptateur* pour la rendre accessible **[10]**.

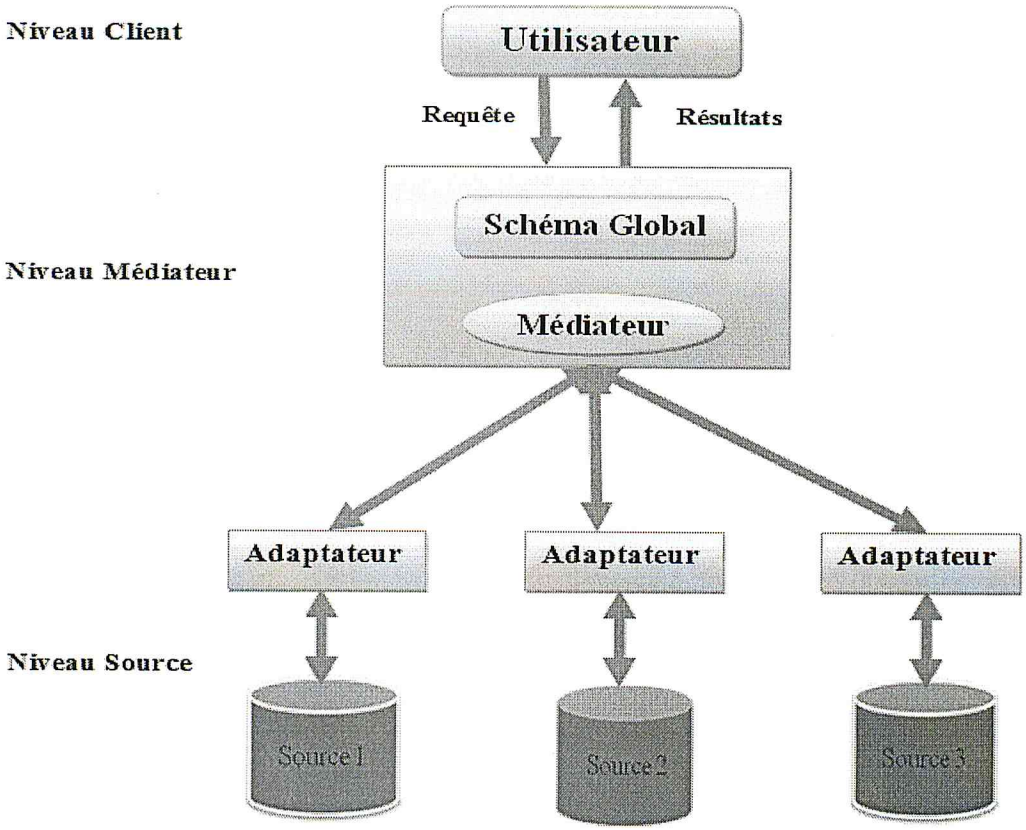


Figure. 1.2 -Architecture de médiation [8]

5. Médiateurs existants [11]

- Génération relationnelle (1975-1990)
  - Souvent centré sur un SGBD qui joue le rôle d'un médiateur
  - SDD1, Sirius Delta, R*, Ingres/Star
  - Mermaid, Multibase, MSQ
- Génération relationnelle étendue (1990-2000)
  - Fédère des BD hétérogènes autour de SQL3
  - Objet : OLE-DB, pegasus, IRO-DB
  - XML : Medience Server, Information Integrator (IBM)
- Génération XML XQuery (2000- …)
  - OLE-DB.NET (Microsoft), Nimble, Xquark Fusion,
  - Liquid Data (BEA), Enosys Software

6. Conclusion

Dans ce chapitre nous avons parlé de système d'intégration de données et ces problèmes ainsi que les différents conflits de données, on a basé sur l'approche médiateur de système d'intégration.

Dans le chapitre suivant nous concentrerons sur Les modèles des métadonnées : XML, RDF, et RuleML, utilisés pour homogénéiser la représentation des données hétérogènes.

Chapitre II

Les modèles des sources de métadonnées : XML, RDF et RuleML

1. Introduction

La maîtrise sur des grands ensembles d'information devient de plus en plus complexe et de plus en plus fastidieux. Les métadonnées constituent une voie pour aider l'utilisateur ou le gestionnaire d'information à comprendre, retrouver, comparer des informations sans forcément avoir recours directement au contenu de celles-ci. En effet, les métadonnées peuvent être vues comme étant des données structurées qui décrivent les données et qui peuvent s'appliquer à tous types de données.

L'objectif de ce chapitre est de définir les différents modèles de métadonnées XML, RDF, RuleML que nous utilisons dans notre projet.

2. Généralités sur les métadonnées

2.1. Qu'est-ce qu'une métadonnée ?

Le terme "meta" vient du grec et dénote quelque chose de nature plus élevée ou plus fondamentale. Les métadonnées sont littéralement "des données relatives à d'autres données" (data about data : données sur des données). Toutefois, l'importance des métadonnées aujourd'hui mérite quelques précisions dans leur définition.
Nous citons ici la définition donnée par le National Information Standards Organisation (NISO), dans un article paru en 2004, intitulé "Understanding Metadata" :
« Une métadonnée (du Grec, "méta", ce qui dépasse, englobe) est une donnée à propos d'une autre donnée. En sciences de l'information, les métadonnées sont des ensembles de données structurées décrivant des ressources physiques ou numériques, ou, sur un plan plus fonctionnel, "de l'information structurée qui décrit, explique, localise la ressource et en facilite la recherche, l'usage et la gestion" » **[13]**.

2.2. Intérêts des métadonnées

Le terme « métadonnée » est utilisé depuis longtemps dans certains domaines d'activité, comme la description des documents géographiques, la gestion des ressources

images et multimédias ou les bases de données. Ce concept est aussi au cœur de certains métiers comme ceux des bibliothèques et de l'archivistique. Il concerne aujourd'hui tous les acteurs de l'environnement numérique.

Les fonctions des métadonnées peuvent en effet être déclinées en six groupes **[14]** :

1. Améliorer la recherche d'information et la « découverte » des ressources
2. Gérer les ressources
3. Gérer les « archives »
4. Faciliter le partage de données et leur réutilisation
5. Participer à la pérennité des ressources numériques
6. Décrire les utilisateurs pour gérer les accès

2.3. Typologie des métadonnées

Le terme « **métadonnée** » regroupe plusieurs typologies. Cette diversité est relative à plusieurs critères. Anne J. Gilliland-Swetland donne un résumé intéressant sur ces différentes typologies, dans son article : Introduction to Metadata **[15]**, où elle répartie les métadonnées en s'appuyant sur sept critères :

la source, le mode de création, la nature, le statut, la structure, la sémantique et le niveau. Emmanuël Colinet et Inge Alberts **[16]** ont repris ces différents attributs auxquels ils ont rajouté un huitième attribut : La granularité.

La figure 2.1 portée dans la page suivante illustre ces différentes typologies :

- **La source :** Cet attribut peut prendre deux valeurs, comme mentionné au début de cette section (Typologie des métadonnées) :

- Métadonnée interne, générée au moment de la création du document ou sa numérisation par l'auteur ou le créateur de la ressource en général.
- Métadonnée externe, créée plus tard, par une personne autre que l'auteur du document **[17]**.

- **Le mode de création :** Nous distinguons principalement deux modes de création :

- Automatique : Les métadonnées sont générées automatiquement par le système informatique : indexation et condensation automatiques. Elles sont regroupées par exemple dans des fichiers logs.

- Manuel : Les métadonnées sont générées à partir de schémas de métadonnées maison ou standardisés, tel que le Dublin Core ou le LOM.

La nature :

- Non spécialisée : Les métadonnées sont créées par des non spécialistes du domaine.
- Spécialisée : Les métadonnées sont générées par des bibliothécaires, archivistes ou professionnels de l'information.

Le statut :

- Statique : Les métadonnées accompagneront le document tout au long de son cycle de vie.

Exemple : titre, provenance, date de création.

- Dynamique : Les métadonnées peuvent évoluer en fonction du cycle de vie : structure des répertoires, résolution des images, information sur les modes d'accès.
- Long Terme : Les métadonnées accompagneront le processus de préservation : informations procédurales et techniques, informations légales, documentation sur la gestion à long terme.

La granularité :

- Large : La description du document sera exhaustive (25 mots clé, par exemple), ce qui aura une incidence au niveau du repérage de l'information (augmentation du bruit et diminution de la pertinence. Il y aura également incidence sur les modes de production (coûts plus élevés) et les modes de gestion (stockage, élimination…). Le choix de la granularité doit se faire en fonction des besoins informationnels.
- Pointue : La description du document sera pointue (5 à 10 mots clé, par exemple).

Le niveau :

- Collection : Les métadonnées sont attribuées pour définir et préserver une collection.
- Document : Les métadonnées sont attribuées pour définir et préserver un document.

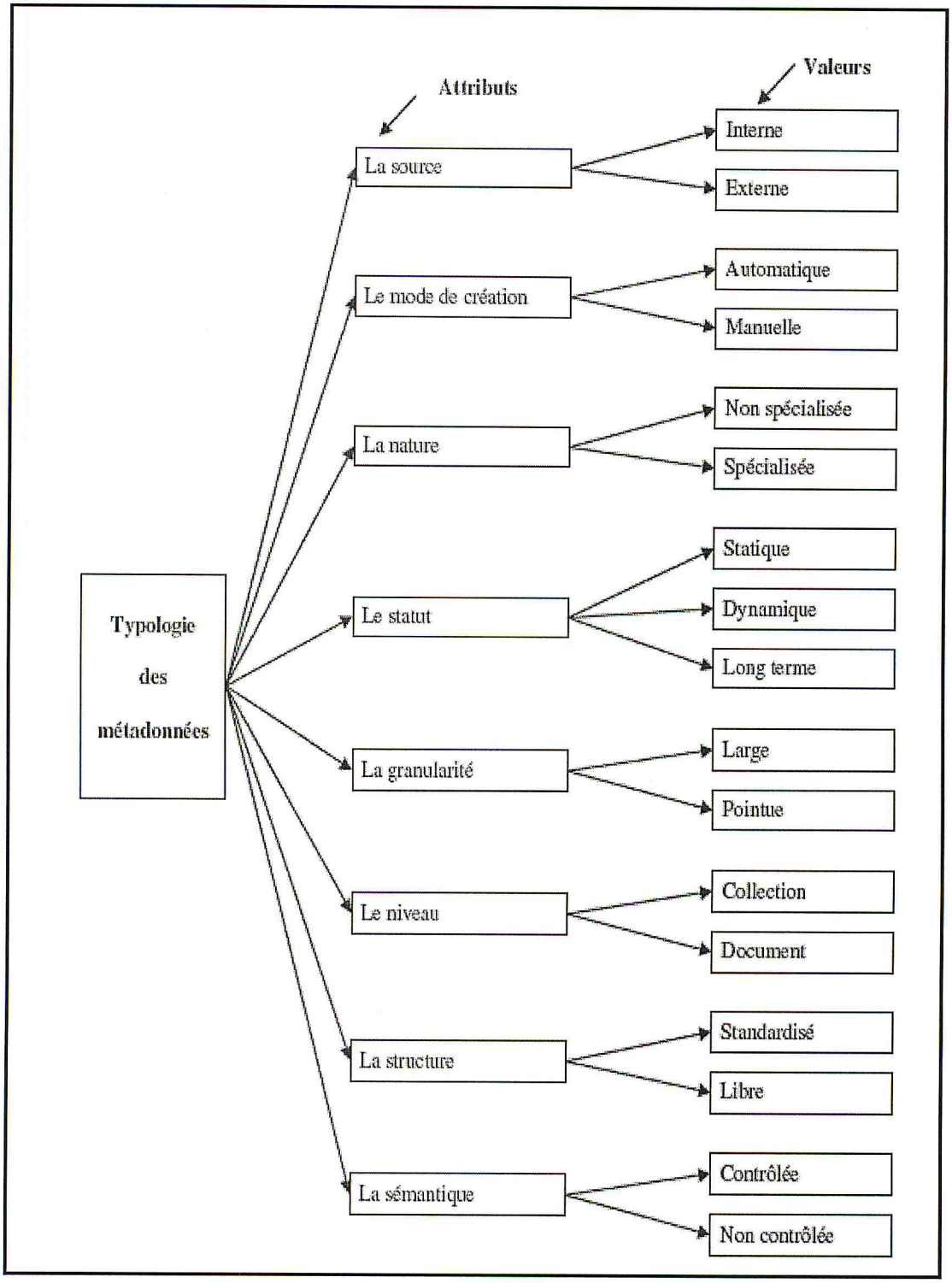


Figure 2.1 : Typologie des métadonnées

La structure :

- Standardisée : tel que : La DTD (HTML, TEI)
  Les Métalangages (XML, SGML)
  Les ensembles de métadonnées structurées (Dublin Core)
- Libre : tel que les systèmes offerts par les logiciels.

La sémantique :

- Contrôlée : La syntaxe est définie par le standard, tel que Dublin Core, MARC ou USMARC
- Non contrôlée : Comme les descripteurs dans les balises META en html.

Outre ces différentes distinctions, on dégage également la notion de métadonnées

2.4. Métadonnées utilisées [18]

2.4.1. Métadonnées au niveau des sources

Les métadonnées définies au niveau des sources décrivent le schéma de chaque source de données, l'ensemble des relations sources appartenant à chaque schéma source, les clés des relations, les attributs de chaque relation, les assertions entre les relations.

2.4.2. Métadonnées au niveau de la médiation

Les métadonnées définies au niveau de la médiation caractérisent le schéma de médiation, l'ensemble des relations de médiation appartenant à ce schéma de médiation, les clés des relations, les dépendances fonctionnelles éventuelles, et les attributs de chaque relation.

- Un schéma de médiation est constitué d'un ensemble de relations de médiation.
- L'ensemble d'assertions définies sur une relation de médiation est composé, essentiellement, de dépendances fonctionnelles qui relient l'attribut clé aux attributs non clés.

2.4.3. Métadonnées entre la médiation et les sources

Les métadonnées entre la médiation et les sources sont des correspondances linguistiques reliant un attribut d'une relation de médiation à un attribut d'une relation source.

3. Les modèles des métadonnées

Plusieurs dizaines de standards de métadonnées sont reconnus et utilisés aujourd'hui, la majorité étant implémentée en XML. Chacun répond à des besoins spécifiques.

3.1. Modèle Dublin Core

Le Dublin Core est un format descriptif à la fois simple et générique, comprenant 15 éléments différents, qui a été créé en 1995 à Dublin (Ohio) par OCLC (Online Computer Library Center) et le NCSA (National Center for Supercomputing Applications).

3.1.1. Intérêts et limites [19]

Dublin Core fait l'objet d'un large consensus et d'une large utilisation aujourd'hui grâce aux atouts suivants :

- Sa création dans un contexte international et multidisciplinaire ;
- Sa sémantique simple et "commune", facilement compréhensible, particulièrement pour les éléments de base ;
- Son extensibilité (compatible avec d'autres jeux d'éléments, évolutivité) et sa flexibilité (grande souplesse d'implémentation) ;
- Son adoption dans différents domaines, métiers et pays, et dans des applications non prévues initialement ou des domaines industriels connexes ;
- Son évolutivité au travers de groupes de travail ouverts ;
- La volonté du DCMI, de diffuser et faire adopter ce modèle ; le site officiel, www.dublincore.org, est très riche en tutoriels et recommandations, exemples, modèles et outils, et reflète bien l'activité de ce groupe d'acteurs ;
- La normalisation des 15 éléments de base à partir de 2003 par l'ISO (norme 15836-2003).

Cependant, on peut résumer au passif du Dublin Core :

- Son côté généraliste et incomplet, nécessitant souvent des extensions,
- Sa relative jeunesse et le fait qu'il évolue encore (bien que les éléments de base semblent être « gravés dans le marbre »).

3.1.2. Quelques principes du Dublin Core

Les auteurs du Dublin Core ont établi une liste de principes qui devaient guider davantage le développement de l'ensemble des éléments de métadonnées. répétabilité et modifiabilité. **[20]**

➢ **Propriété intrinsèque :**

Le Dublin Core a été dirigé, dés le départ pour décrire des propriétés intrinsèques de l'objet. Par exemple, l'élément "Sujet" est une donnée intrinsèque, tandis que des informations de transaction telle que coût et droit d'accès sont des données extrinsèques.

➢ **Extensibilité :**

En plus de son emploi en traitant de la propriété intrinsèque des données, le mécanisme d'extension permettra l'inclusion de données intrinsèques pour des objets qui ne peuvent pas être décrits suffisamment par un petit ensemble d'éléments.

➢ **Indépendance de Syntaxe :**

Les constructions syntaxiques sont évitées parce qu'il est trop tôt de proposer des définitions formelles et parce que le Dublin Core est destiné à être employé tôt ou tard dans une large gamme de programmes d'application et de disciplines.

➢ **Optionalité :**

Tous les éléments sont optionnels, pour deux raisons :

- La première est que le Dublin Core peut tôt ou tard être appliqué aux objets pour lesquels certains éléments n'ont pas de signification : Qui est l'auteur d'une image satellitaire?
- La seconde est qu'il semble futile de donner des descriptions complexes lorsque les auteurs du contenu prévoient de fournir la matière descriptive.

➢ **Répétabilité :**

Tous les éléments du Dublin Core sont répétables. Par exemple, plusieurs éléments 'auteur' seraient employés quand une ressource a plusieurs auteurs.

3.1.3. Liste des éléments du Dublin Core [21]

Le *Dublin Core* est un ensemble de 15 éléments de métadonnées ayant trait:

- au *Contenu*: Title, Description, Subject, Source, Coverage, Type, Relation

- à la *Propriété intellectuelle*: Creator, Contributor, Publisher, Rights
- à la *Version*: Date, Format, Identifier, Language

Nom de l'élément	Définition
Title	Le nom donné à la ressource
Creator	L'entité principalement responsable de la création du contenu de la ressource
Subject	Le sujet du contenu de la ressource
Description	Une description du contenu de la ressource
Publisher	L'entité responsable de la diffusion de la ressource, dans sa forme actuelle, tels, un département universitaire, une entreprise.
Contributor	Une entité qui a contribué à la création du contenu de la ressource
Date	Une date associée avec un événement dans le cycle de vie de la ressource
Type	La nature ou le genre du contenu de la ressource
Format	La matérialisation physique ou digitale de la ressource
Identifier	Une référence non ambiguë à la ressource dans un contexte donné
Source	Une référence à une ressource à partir de laquelle la ressource actuelle a été dérivée
Language	La langue du contenu intellectuel de la ressource
Relation	Une référence à une autre ressource qui a un rapport avec cette ressource
Coverage	La portée ou la couverture spatio-temporelle de la ressource
Rights	Information sur les droits sur et au sujet de la ressource

Tableau 2.1: Éléments de base du Dublin Core et leur signification

3.2. Modèle XML

3.2.1. Qu'est-ce que le XML?

XML (eXtensible Markup Language) est une recommandation du W3C. C'est un langage à balise définissant un format universel de représentation des données. Un document XML contient à la fois des données et les indications sur le rôle que jouent ces données. Ces indications permettent de déterminer la structure du document : ce sont des balises.

XML est un format largement répandu et accepté par la communauté informatique. De nombreuses sources de données sont disponibles et de nombreuses applications permettent d'exporter leurs données en XML. XML offre deux techniques pour créer une structure de document XML **[22]**

Syntaxe XML : [23]

- Information crue: Ali Benali, 23, rue Med-V, Rabat. 010 12 34 56
- Structurée en XML

```
<?xml version="1.0"?>
<personne>
<prénom>Ali</prénom>
<nom>Benali</nom>
<adresse>23, rue Med-V, Rabat</adresse>
<tel>090 12 34 56</tel>
</personne>
```

- Un élément racine <personne>
- d'autres éléments (nœuds fils) <prénom> <nom> <adresse> ...
- Des attributs <tel type="personnel">090 12 34 56</tel>

3.2.2. Les DTD

DTD signifie Document Type Définition (ou définition de type de document), c'est la grammaire historique des documents XML. La puissance de description des DTD est faible : une DTD permet uniquement de décrire la structure d'un document XML (liste des balises et organisations des balises), et non la typologie des données contenues (chaîne de caractère, date, entier, etc.)**[22]**.

3.2.3. Les Schémas XML

Un schéma XML **[24,25]** a le même rôle qu'une DTD, c'est-à-dire définir la structure d'un document XML. L'objectif des schémas XML est de proposer une solution aux inconvénients majeurs des DTD **[25,26]**:

1. Une syntaxe différente que celle de XML : on doit donc écrire un document XML dans un langage et la DTD dans un autre;
2. La notion d'espace de noms n'existe pas (pour faire face aux risques de collisions de balises);
3. Pas de notion de type de données;
4. Pas de notion d'héritage: Les systèmes orientés objet s'appuient sur l'idée de décrire de nouveaux objets à partir d'objets existants. Ceci n'est pas possible.

Un schéma XML est composé principalement d'éléments qui sont représentés par les balises. Ces éléments sont associés à un type de données qui peut être défini comme « type simple » ou « type complexe ».

Exemple de document XML Schéma : [23]

- Syntaxe XML, types de données (booléen, nombres entiers décimaux, Date et Time...) avec contraintes associées.
- Constructeurs de types (séquence, group, complexe Type, ...)
- Définit quels éléments (et attributs) doivent figurer dans un document XML, dans quel ordre, lesquels sont optionnels, combien d'occurrences, etc.
- XML schéma est un standard bien défini (recommandation W3C, XML schéma)

Espaces de noms (namespaces) : [23]

- Comment gérer les conflits de noms?
- Deux noms identiques pour désigner deux choses conceptuellement différentes.
- Un espace de nom, namespace, est une recommandation W3C, qui permet de résoudre les conflits en donnant un contexte aux éléments d'un document
- Syntaxe

  <xsd:sequence> <agenda:prénom> <agenda:tel> <agenda:type> ...
- Chaque élément ou attribut du document est préfixé par une étiquette.

- Une étiquette fait référence à un espace de nom, identifié par une URI, par définition unique au monde.
- Les espaces de nom ne sont pas fait uniquement pour des besoins syntaxiques de résolution de noms.
- Ils peuvent aussi servir à une application pour traiter uniquement les données utiles (ex. les noms des personnes et non pas ceux de marque).
- Cette caractéristique est très utilisée dans les langages dérivés de XML.

La famille XML : [23]

- Langage de style XSL (Extensible Stylesheet Language), recommandation W3C
- Permet d'associer un style d'affichage à un document XML
- Langage XSLT (XSL Transformation), recommandation W3C
- Permet de transformer un document XML vers un autre. En fait, vers tout.
- Un processeur XSLT transforme une structure d'arbre vers une autre.
- Utilise des expressions de chemin XPath, pour désigner les nœuds d'un arbre XML.
- XSLT est un véritable langage déclaratif de haut niveau.
- Considérant un document XML comme une base de données, XQuery, est un véritable langage de requête. Basé sur XPath.
- XQuery donne à XML une dimension base de données.

3.3. Modèle RDF (Resource Description Framework)

- Le formalisme RDF est la base technique du semantic web. **[27,28]**
- RDF (Resource Description Framework) **[29,30]** est une recommandation du W3C pour décrire des ressources. C'est un modèle de graphe pour décrire les (méta-) données en permettant leur traitement automatisé. A l'origine, il a été défini pour décrire des ressources du Web telles que les pages Web; cependant une ressource peut être toute chose ayant une identité (objet physique, concept abstrait, etc.).

- Ressemble à un modèle de données orientées objets. Utilise des URI pour identifier des objets, propriétés et relations entre objets
- Consistes en des entités et des relations binaires (*statements*) entre ces entités. **[27,28]**

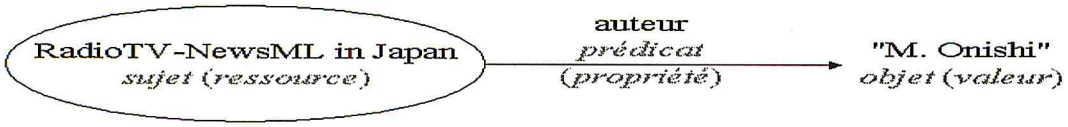
3.3.1. Principe de base

- RDF est un formalisme écrit en XML qui permet de définir à son tour des "langages"
- Modèle de données:
  - ✓ Sujet = La ressource
  - ✓ Verbe = Prédicat = propriété = relation qui va du sujet vers l'objet
  - ✓ Objet = Valeur

Exemple :

Le document *RadioTV-NewsML in Japan*	a pour auteur	*M. Onishi*
Sujet	Prédicat	*objet*

- Ces triplets sont modélisés à l'aide de graphes orientés étiquetés

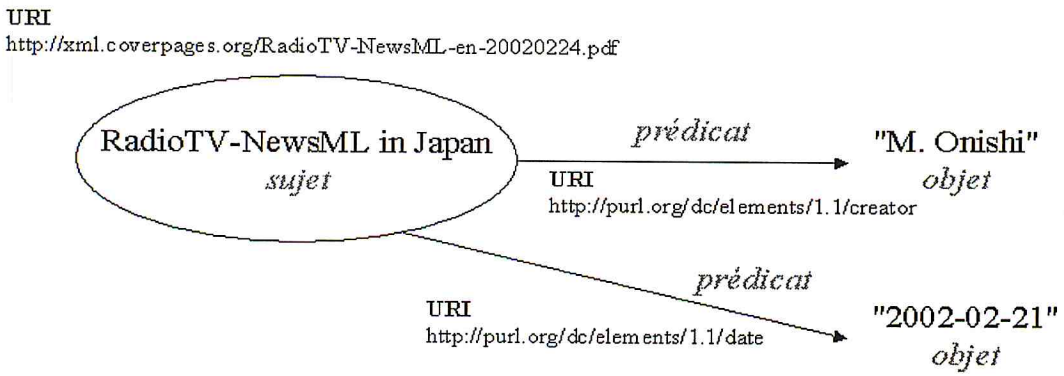


- RDF est extensible
  - ✓ Chaque extension possède son "name space" et sa propre "logique". En connaissant la logique d'une extension, on peut écrire des programmes qui savent traiter une information.
  - ✓ Par exemple: un programme FOAF peut visualiser des réseaux sociaux, une plateforme e-learning peut organiser un apprentissage, ...
  - ✓ RDF est à la base du "semantic web", un web accessible aux machines **[31].**
- Les ressources sont identifiées par des URI (Unified Ressource Identifier)
  - ✓ Les URI sont un "stock de noms" utilisés pour désigner des choses ou des concepts

- ✓ Les *URL* habituelles sont des *URI*. Ainsi, dans notre exemple, le document *RadioTV-NewsML in Japan* peut être identifié naturellement par l'URI *http://xml.coverpages.org/RadioTV-NewsML-en-20020224.pdf*
- ✓ Les prédicats (propriétés) sont également représentés par des URI

L'URI du prédicat "*a pour auteur*" est l'élément *Creator* du schéma *Dublin Core*: *http://purl.org/dc/elements/1.1/creator*

- Un sujet (ressource) peut posséder plusieurs prédicats (propriétés). **[32]**



3.3.2. RDF – syntaxe XML

```
conteneur RDF
<?xml version="1.0"?>
<rdf:RDF   utilisation des espaces de noms rdf et dc
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
    <rdf:Description   about précise la ressource à décrire
rdf:about="http://xml.coverpages.org/RadioTV-NewsML-en-
20020224.pdf">          valeur M. Onishi
propriété creator   <dc:creator>M. Onishi</dc:creator>
            <dc:title>RadioTV-NewsML in Japan</dc:title>
            <dc:date>2002-02-21</dc:date>
            <dc:type>Text</dc:type>
            <dc:format>application/pdf</dc:format>
    </rdf:Description>
</rdf:RDF>
```

Figure 2.2: présentation de graphe RDF dans un fichier XML **[32]**

3.3.3. RDF – Schémas

- RDF(S) fournit un ensemble de primitives simples, mais puissantes, pour la structuration de la connaissance d'un domaine en classes et sous classes, propriétés et sous propriétés avec la possibilité de restreindre leur domaine d'origine (rdf : domain) et leur domaine d'arrivée (rdf : range) **[33]**.
- Un Schéma RDF permet de décrire un vocabulaire et une sémantique des types de propriétés utilisées par une communauté d'utilisateurs **[32]**.

3.3.4. Le vocabulaire utilisé dans RDF [34]

Ci-dessous sont présentés deux tableaux qui donnent une vue générale du vocabulaire de **RDF**, reliant le vocabulaire originellement défini dans la spécification de la syntaxe et du modèle **RDF** avec les classes et les propriétés qui ont leur origine dans le schéma **RDF**.

Nom de la classe	Commentaire
rdfs:Resource	La classe Resource, tout.
rdfs:Literal	La classe des valeurs littérales, c'est à dire des chaînes caractères et les entiers.
rdfs:Class	La classe des classes.
rdf:Property	La classe des propriétés RDF
rdf:Statement	rdf:Statement La classe des déclarations RDF.
rdf:Bag	La classe des containers non ordonnés.
rdf:XMLLiteral	La classe des valeurs littérales permises par XML.
rdfs:Datatype	rdfs:Datatype La classe des types de données RDF.
rdf:Seq	La classe des containers ordonnés.
rdf:Alt	La classe des containers d'alternatives.
rdfs:Container	La super classe des containers.
rdfs:ContainerMembership Property	La classe des propriétés des membres des containers, rdf:_1, rdf:_2, ..., dont tous sont des sous-propriétés de 'member'.
rdf:List	La classe des listes RDF.

Tableau 2.2: Classes RDF/RDFS

Nom propriétés	Commentaire	domaine	range
rdf:type	Le sujet (ressource), une instance d'une classe	rdfs:Resource	rdfs:Class
rdfs:subClassOf	Le sujet est une sous-classe d'une classe.	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	Le sujet, une sous-propriété d'une propriété.	rdf:Property	rdf:Property
rdfs:domain	Le domaine des ressources s'appliquant à une ressource. (A domain of the subject property.)	rdf:Property	rdfs:Class
rdfs:range	Le domaine des ressources s'appliquant à une propriété. (A range of the subject property.)	rdf:Property	rdfs:Class
rdfs:label	Nom lisible par un humain du sujet (ressource).	rdfs:Resource	rdfs:Literal
rdfs:comment	Une description du sujet (ressource).	rdfs:Resource	rdfs:Literal
rdfs:member	Un membre de la ressource. (A member of the subject resource.)	rdfs:Resource	rdfs:Resource
rdf:first	Le premier article d'une liste RDF de sujet. (The first item in the subject RDF list.)	rdf:List	rdfs:Resource
rdf:rest	Le reste des articles d'une liste RDF de sujet après le premier article. (The rest of the subject RDF list after the first item.)	rdf:List	rdf:List
rdfs:seeAlso	Information complémentaire sur le sujet.	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	La définition du sujet.	rdfs:Resource	rdfs:Resource
rdf:value	Propriété idiomatique utilisée pour déclarer des valeurs structurées (i.e. typées).	rdfs:Resource	rdfs:Resource
rdf:subject	Le sujet d'une déclaration (statement) RDF	rdf:Statement	rdfs:Resource
rdf:predicate	Le prédicat d'une déclaration (statement) RDF	rdf:Statement	rdf:Property
rdf:object	L'objet d'une déclaration (statement) RDF	rdf:Statement	rdfs:Resource

Tableau 2.3: Propriétés RDF/RDFS

3.4. Modèle RuleML(Rule Markup Language)

3.4.1. Définition

L'initiative RuleML a vu le jour en Août 2000 lors de la conférence international sur l'intelligence artificielle PRICAI2000 **[35,36]**.

RuleML a pour objectif de développer un formalisme standard de règles métier neutre et ouvert basé sur XML/RDF dans le but de permettre à des systèmes hétérogènes de s'échanger des règles. Le groupe de travail sur RuleML est composé d'académiciens et d'industriels. Les principes de RuleML **[48] [49]** ont beaucoup servi pour d'autres systèmes de formalismes de règles. RuleML supporte aussi bien le chaînage avant qu'arrière.

D'un point de vue syntaxique, RuleML nous offre des règles définies par plusieurs balises, très proches du langage XML. D'un point de vue structural, une règle est formée de deux parties principales (qui créent une implication) :

- la partie 'if' : prémisse ou antécédent (<_body> ... </_body>),
- la partie 'then' : conclusion ou conséquence de la règle (<_head> ... </_head>).**[50]**

3.4.2. Les types de règles

RuleML permet de modéliser différents types de règles selon le contexte applicatif :

- des règles réactives, du type " si événement E, alors E' ",
- des règles de transformation, du type " A se transforme en B ",
- des règles d'implication, du type " A implique B ",
- des faits,
- des requêtes,
- des contraintes d'intégrité, du type " A et B et C vrais ".
- des balises XML spécifiques sont introduites pour chaque type de règle. **[DIO, 2007]**

Example RuleML Document: A Rulebase own.ruleml **[48]**

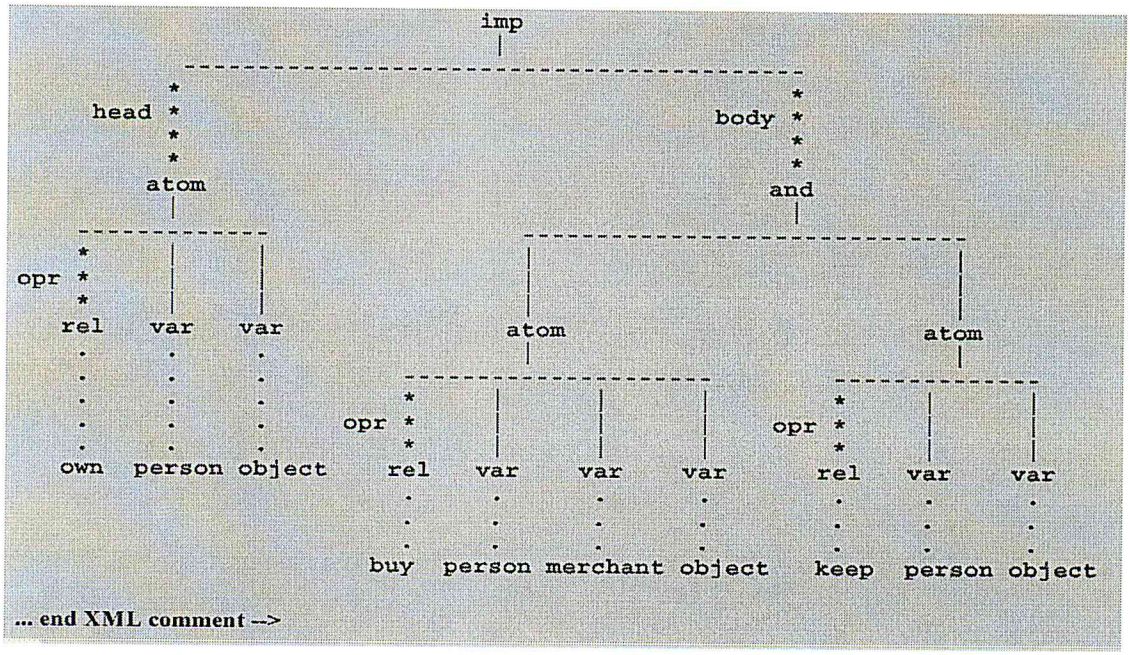
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE rulebase SYSTEM "http://www.dfki.de/ruleml/dtd/0.8/ruleml-datalog-monolith.dtd">
<rulebase>

<!-- start XML comment ...
```

This example rulebase contains four rules. The first and second rules are implications; the third and fourth ones are facts.

The first rule implies that a person owns an object if that person buys the object from a merchant and the person keeps the object.

As an OrdLab Tree:




```
... end XML comment -->
```

```
<imp>
 <_head>
  <atom>
   <_opr><rel>own</rel></_opr>
   <var>person</var>
   <var>object</var>
  </atom>
 </_head>
 <_body>
  <!-- explicit 'and' -->
  <and>
   <atom>
    <_opr><rel>buy</rel></_opr>
    <var>person</var>
    <var>merchant</var>
    <var>object</var>
   </atom>
   <atom>
    <_opr><rel>keep</rel></_opr>
    <var>person</var>
    <var>object</var>
   </atom>
  </and>
 </_body>
</imp>
```

<!-- The second rule implies that a person buys an object from a merchant if the merchant sells the object to the person. -->

```
<imp>
 < head>
  <atom>
   < opr><rel>buy</rel></ opr>
   <var>person</var>
   <var>merchant</var>
   <var>object</var>
  </atom>
 </_head>
 < body>
  <atom>
   < opr><rel>sell</rel></ opr>
   <var>merchant</var>
   <var>person</var>
   <var>object</var>
  </atom>
 </_body>
</imp>
```

<!-- The third rule is a fact that asserts that John sells XMLBible to Mary. -->

```
<fact>
 <_head>
  <atom>
   <_opr><rel>sell</rel></_opr>
   <ind>John</ind>
   <ind>Mary</ind>
   <ind>XMLBible</ind>
  </atom>
 </_head>
</fact>
```

<!-- The fourth rule is a fact that asserts that Mary keeps XMLBible.

Observe that this fact is binary - i.e., there are two arguments for the relation. RDF viewed as a logical knowledge representation is, likewise, binary, although its arguments have type restrictions .-->

```
<fact>
 < head>
  <atom>
   < opr><rel>keep</rel></ opr>
   <ind>Mary</ind>
   <ind>XMLBible</ind>
  </atom>
 </ head>
</fact>
</rulebase>
```

4. Conclusion

Dans ce chapitre, nous avons évoqué d'une manière détaillée les notions liées aux métadonnées. Nous avons traité les différents type et standards de métadonnées existants, leurs formats (syntaxes) d'implémentation.

Chapitre III

Conception du système médiateur

1. Introduction

La conception a pour but de définir de façon très précise les fonctions du logiciel, à partir des besoins exprimés et des contraintes générales définies en phase d'analyse et de spécification des besoins.

2. Le cycle de vie d'un logiciel

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

L'origine de ce découpage provient du constat que les erreurs ont un coût d'autant plus élevé qu'elles sont détectées tardivement dans le processus de réalisation. Le cycle de vie permet de détecter les erreurs au plus tôt et ainsi de maîtriser la qualité du logiciel, les délais de sa réalisation et les coûts associés. **[37]**

2.1 Modèle de cycle de vie en cascade [37]

Le modèle de cycle de vie en cascade a été mis au point dès 1966, puis formalisé aux alentours de 1970. Il définit des phases séquentielles à l'issue de chacune desquelles des documents sont produits pour en vérifier la conformité avant de passer à la suivante .

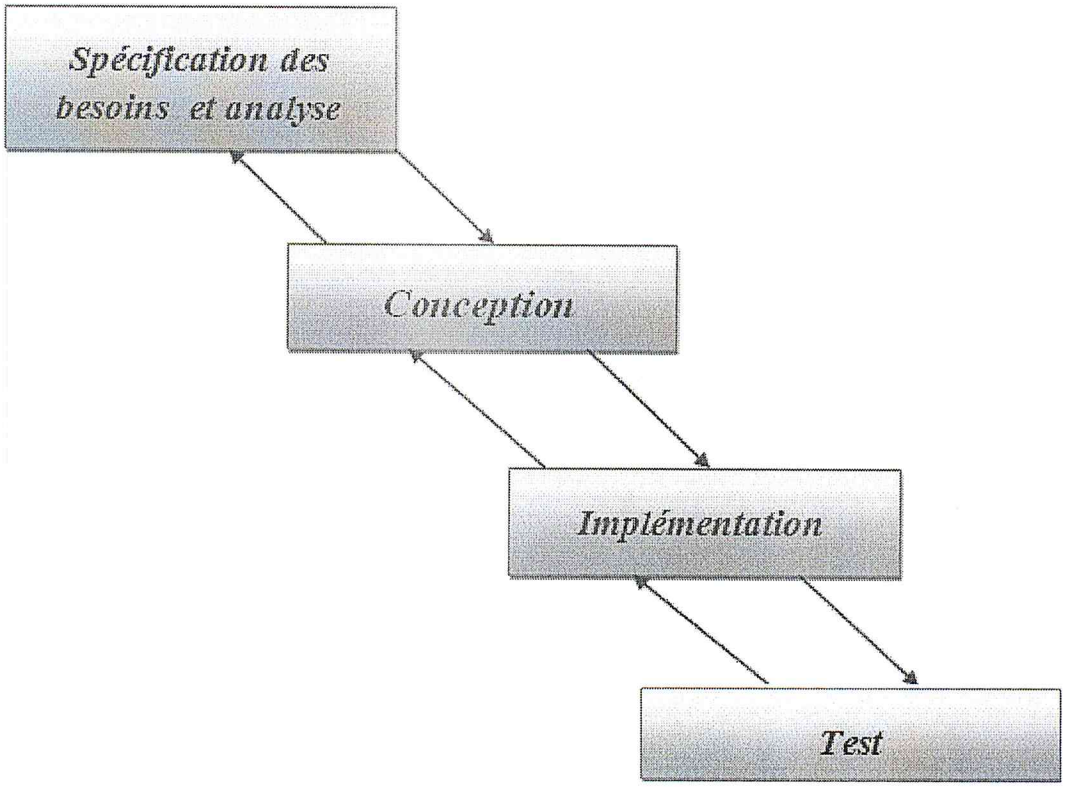


Figure 3.1 : Modèle du cycle de vie en cascade

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

❖ **Spécification des besoins**

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

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

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

❖ **Analyse**

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

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

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

❖ **Conception**

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

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

Elle constitue un point de départ à l'implémentation :

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

❖ **Implémentation**

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

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

❖ **Test**

Les tests permettent de vérifier des résultats de l'implémentation en testant la construction.

Pour mener à bien ces tests, il faut les planifier pour chaque itération, les implémenter en créant des cas de tests, effectuer ces tests et prendre en compte le résultat de chacun.

3. Le langage UML

UML ou Unified Modeling Language, est un langage de modélisation qui est né au milieu des années 90 de la fusion de trois méthodes objets: OMT (Object Modeling Technique), BOOCH1 (GRADY BOOCH le concepteur de la méthode) et OOSE (Object Oriented Software Engineering). L'idée de cette fusion est partie du constat qu'à l'époque ils existaient plusieurs méthodes objets liées par un consensus autour d'idées communes: objets, classes, sous-systèmes etc. **[38]**

UML comporte ainsi treize types de diagrammes représentant autant de *vues* distinctes pour représenter des concepts particuliers du système d'information. Ils se répartissent en deux grands groupes : **[37]**

Diagrammes structurels ou diagrammes statiques (*UML Structure*)

- diagramme de classes (*Class diagram*)
- diagramme d'objets (*Object diagram*)
- diagramme de composants (*Component diagram*)
- diagramme de déploiement (*Deployment diagram*)
- diagramme de paquetages (*Package diagram*)
- diagramme de structures composites (*Composite structure diagram*)

Diagrammes comportementaux ou diagrammes dynamiques (*UML Behavior*)

- diagramme de cas d'utilisation (*Use case diagram*)
- diagramme d'activités (*Activity diagram*)

- diagramme d'états-transitions (*State machine diagram*)
- Diagrammes d'interaction (*Interaction diagram*)

- diagramme de séquence (*Sequence diagram*)
- diagramme de communication (*Communication diagram*)
- diagramme global d'interaction (*Interaction overview diagram*)
- diagramme de temps (*Timing diagram*)

4. Spécification des besoins

4.1. Recueil des Besoins Fonctionnels

4.1.1. Identification des Acteurs

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

Définition

Un **Acteur** représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système), qui interagissent directement avec le système étudié **[37]** .

4.1.2. Identification des cas d'utilisation

4.1.2.1. Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation décrit la succession des opérations réalisées par un acteur (personne qui assure l'exécution d'une activité). C'est le diagramme principal du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre **[39]**

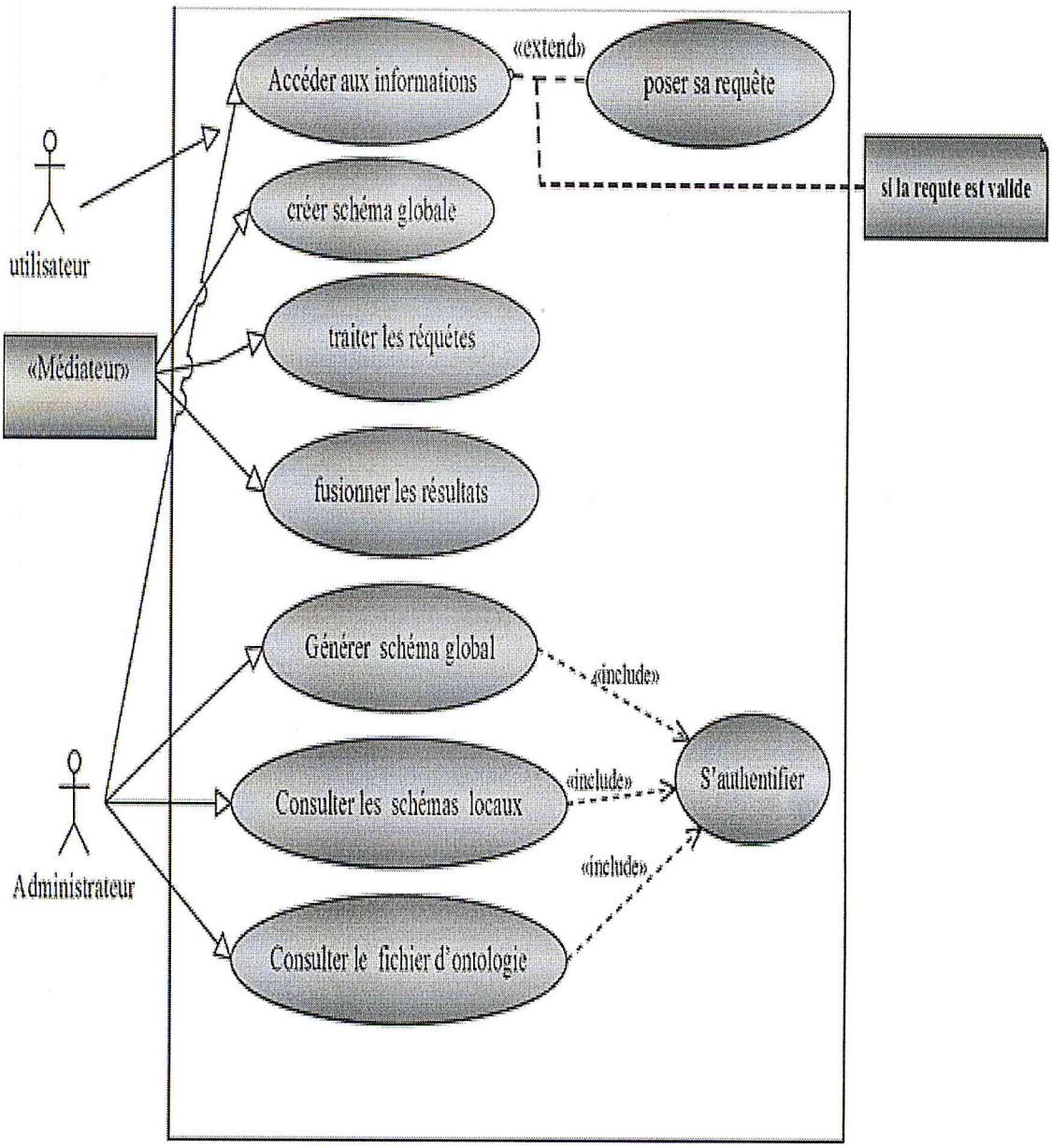


Figure 3.2 : Diagramme de cas d'utilisation global de système.

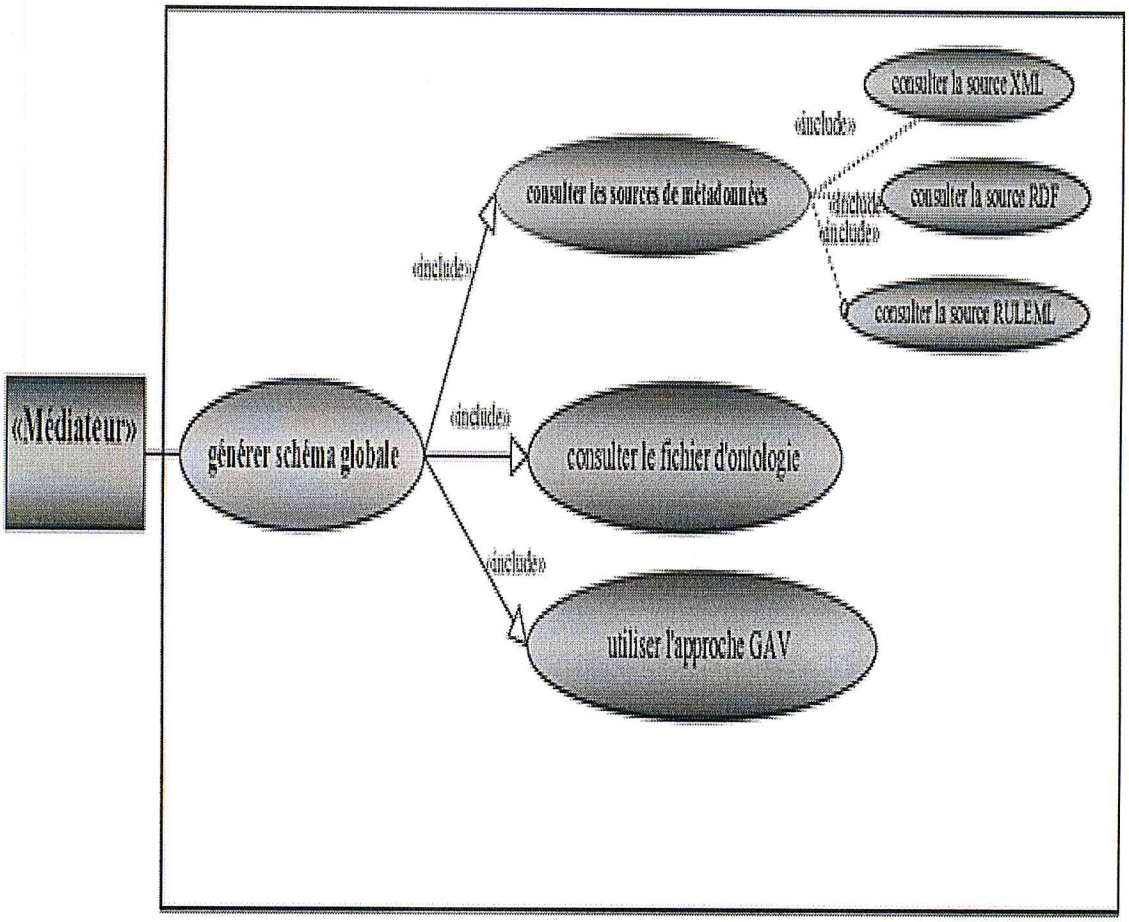


Figure 3.3 : Diagramme de cas d'utilisation de schéma global

Description du diagramme du cas d'utilisation global de système

Cas d'utilisation	Description
Accéder aux informations	L'utilisateur et administrateur choisissent d'accéder aux informations.
Poser sa requête	L'utilisateur pose sa requête en terme global.
Créer schéma global	Le médiateur crée le schéma global.
Traiter les résultats	Le médiateur traite les requête poser par l'utilisateur.
Fusionner les résultats	Le médiateur fusionne les résultats prévenants de différentes sources.
Générer schéma global	L'administrateur consulte le schéma global.
Consulter les schéma locaux	L'administrateur consulte les schéma locaux
Consulter l'ontologie	L'administrateur consulte le fichier ontologie.

Tableau 3.1: Description du diagramme du cas d'utilisation global de système

Description du diagramme du cas d'utilisation de schéma global

Cas d'utilisation	Description
générer schéma global	Le médiateur est chargé de générer le schéma global de système
Consulter les sources de métadonnées	Le médiateur consulte les différentes sources de métadonnées
Consulter le fichier d'ontologie	Le médiateur consulte le fichier d'ontologie affin de construire le schéma.

Tableau 3.2: Description du diagramme du cas d'utilisation de schéma global

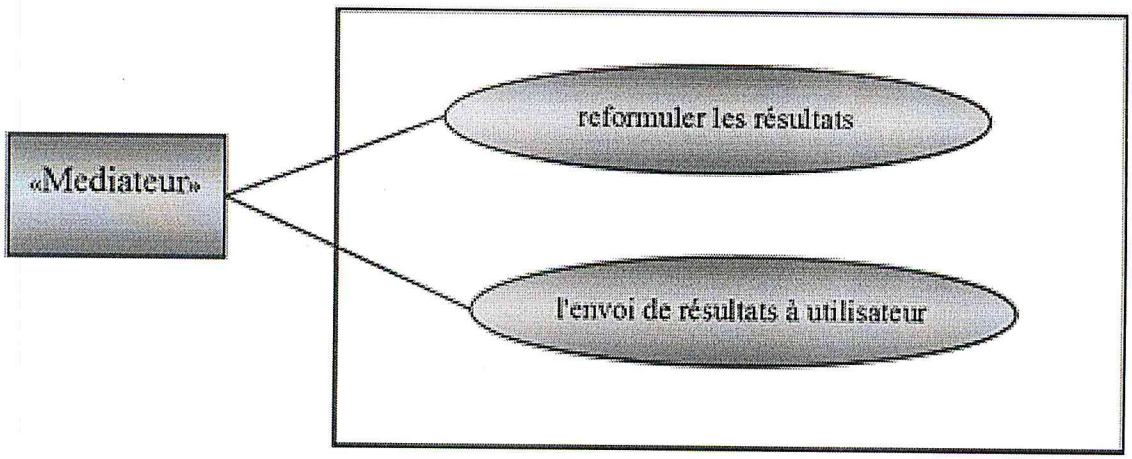


Figure 3.4 : Diagramme de cas d'utilisation fusion des résultats

Description du diagramme du cas d'utilisation fusion des résultats

Cas d'utilisation	Description
reformuler les résultats.	Le médiateur reformule les résultats provenant de différentes sources locales.
l'envoi de résultats à l'utilisateur	Le médiateur envoi les résultats finaux.

Tableau 3.3: Description du diagramme du cas d'utilisation fusion des résultats

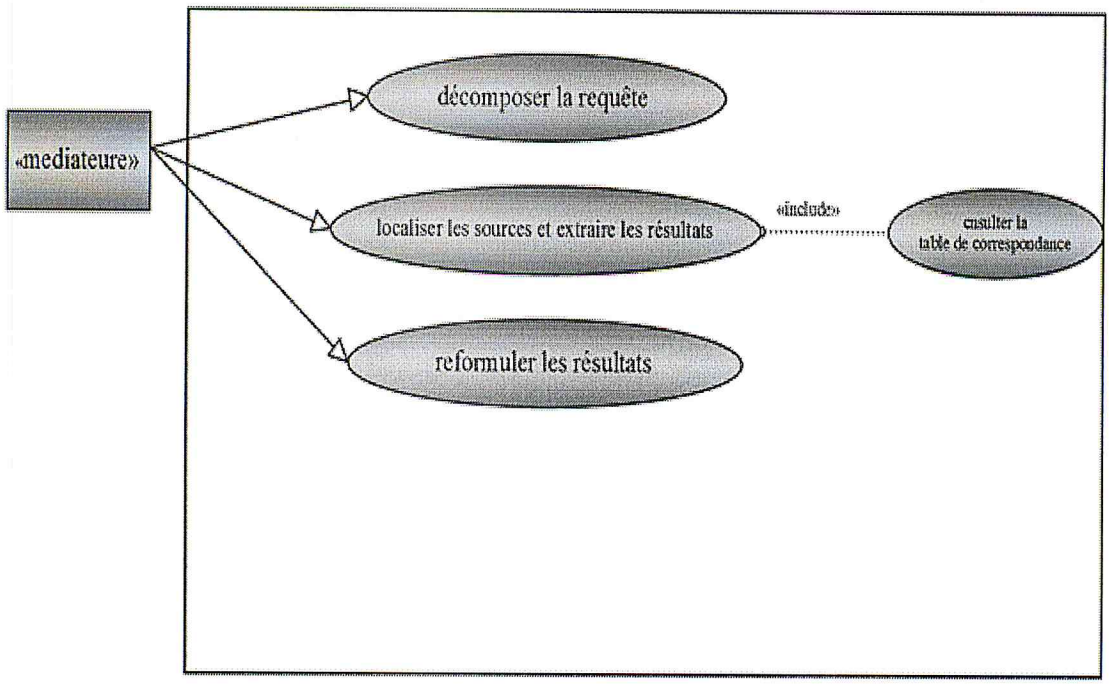


Figure 3.5 : Diagramme de cas d'utilisation traitement d'une requête

Description du diagramme de cas d'utilisation: traitement d'une requête

Cas d'utilisation	Description
décomposer la requête	Le médiateur décompose la requête globale en sous requêtes
localiser les sources et extraire les résultats	Le médiateur localise les sources pertinentes.
reformuler les résultats	Le médiateur reformule les résultats provenant de différentes sources locales et envoi les résultats a l'utilisateur.

Tableau 3.4: Description du diagramme de cas d'utilisation: traitement d'une requête

5. Analyse de système

Dans la phase d'analyse nous présenterons les diagrammes de séquences :

5.1. Diagramme de séquence

Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur : saisir une donnée, consulter une donnée, lancer un traitement ; il indique les objets que l'acteur va manipuler, et les opérations qui font passer d'un objet à l'autre, graphe dont les nœuds sont des objets et les arcs (numérotés selon la chronologie) les échanges entre objets. **[volle,02]**

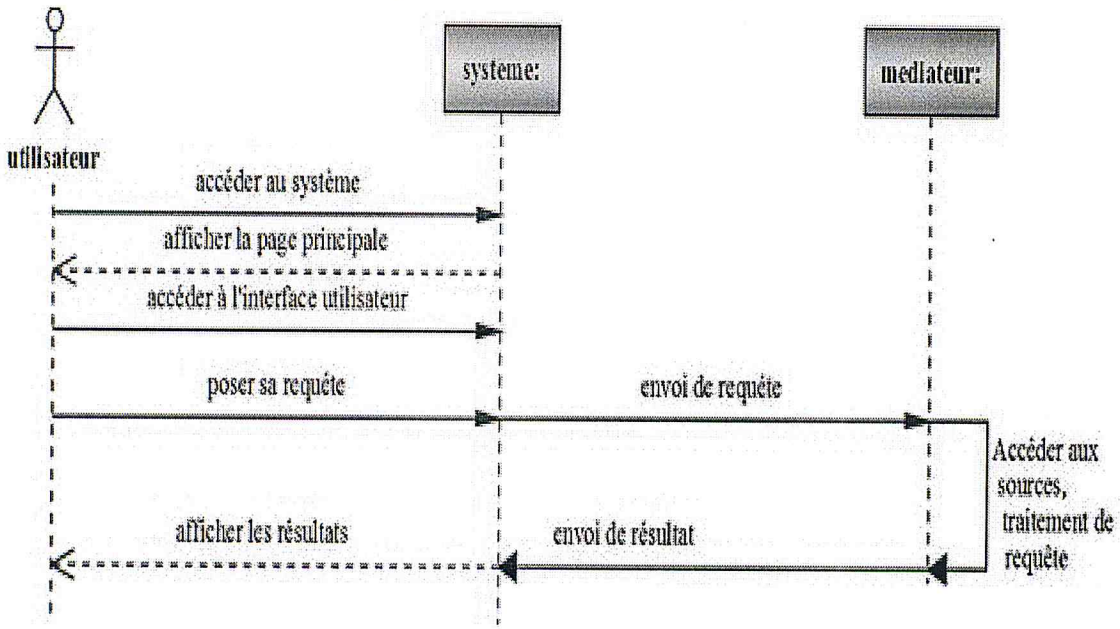


Figure 3.6 : Diagramme de séquence de lancement requête

1. Accédé aux informations

- **Scenario de diagramme de séquence de lancement d'une requête**
  1. L'utilisateur accéder au système
  2. Le système afficher la page principale de l'application.
  3. L'utilisateur choisir d'accéder à l'interface utilisateur.

4. Le système affiche a l'utilisateur l'interface pour qu'il puisse poser sa requête.
5. Après avoir reçu la requête pose par l'utilisateur le système envoi les paramètres de requête au médiateur.
6. Le médiateur envoi le résultat de requête au système après leur traitement.
7. Le système affiche le résultat.

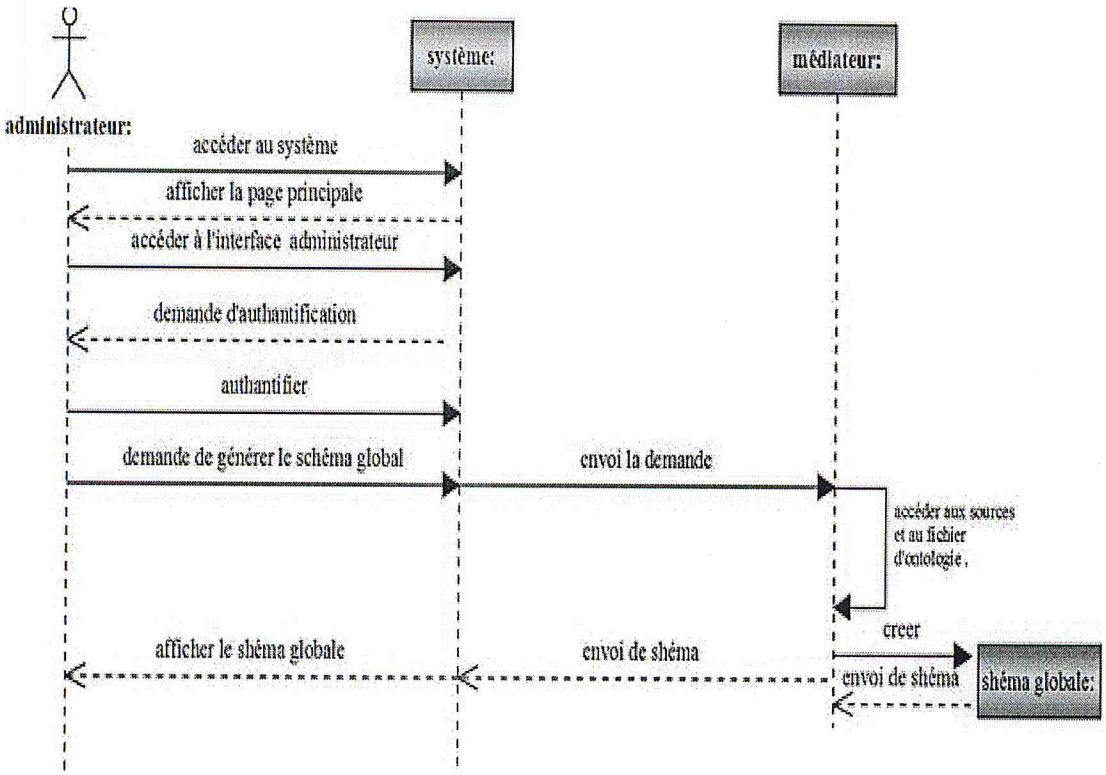


Figure 3.7 : Diagramme de séquence de consultation schéma global

2. Affichage de schéma globale

- **Scenario de diagramme de séquence de consultation de schéma globale**
  1. L'administrateur accéder au système.
  2. Le système lui affiche la page principale de l'application.
  3. L'administrateur accéder à l'interface administrateur

4. L'administrateur demande de générer le schéma global.
5. Le système envoi la demande au médiateur.
6. Le médiateur accéder aux sources et au fichier d'ontologie.
7. Le médiateur créer le schéma globale.
8. Le médiateur envoi le schéma au système.
9. Le système affiche le résultat à l'administrateur.

6. Conception

6.1. Identification des classes candidates

Cette phase va préparer la modélisation orientée objet en aidant à trouver les classes principales du futur modèle statique d'analyse.

6.2. Diagramme de classes

Le diagramme de classe représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens, que ceux-ci représentent un emboîtage conceptuel (héritage, marqué par une flèche terminée par un triangle) ou une relation organique (agrégation, marquée par une flèche terminée par un diamant). **[volle,02]**

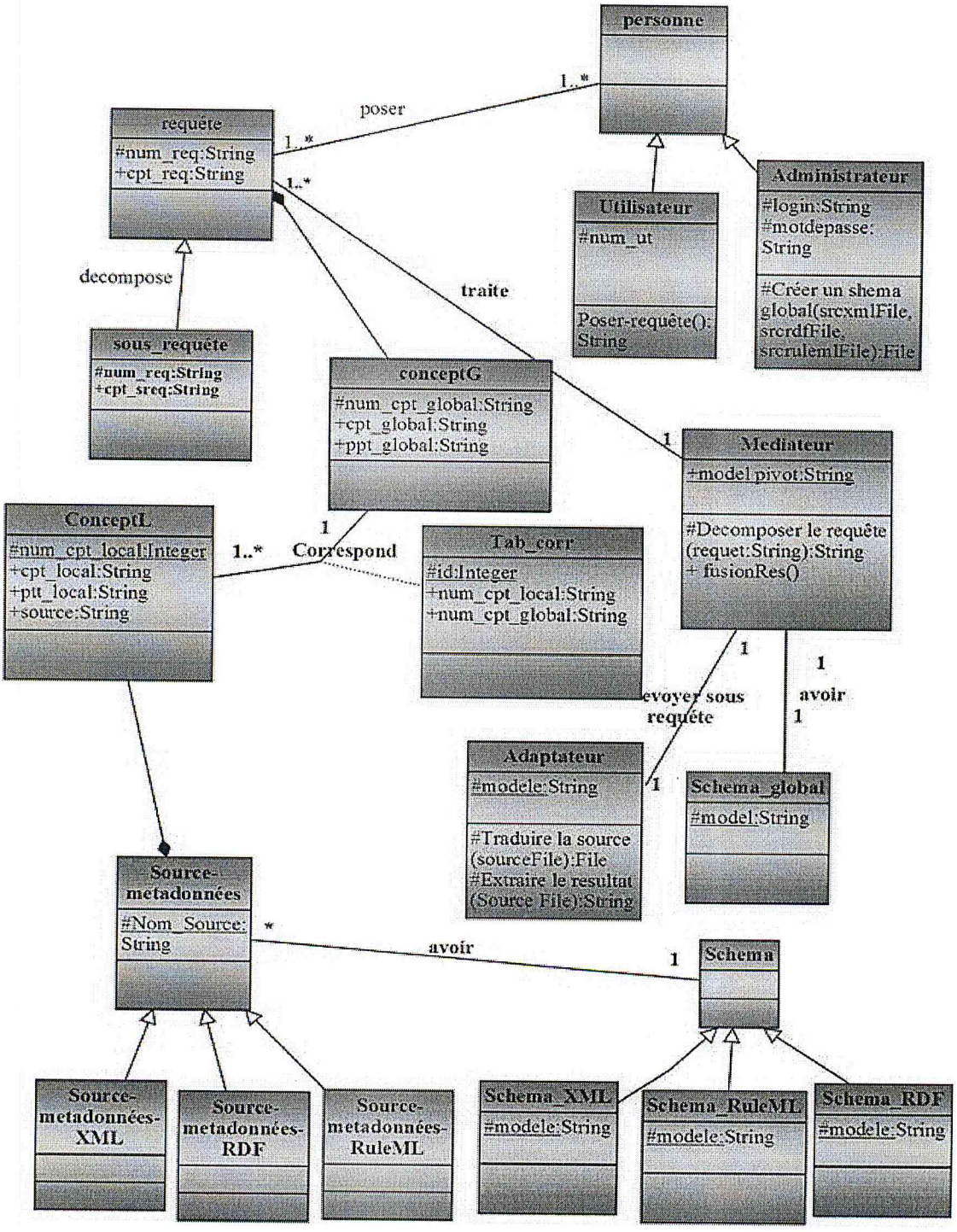


Figure 3.8 : Diagramme de classe de notre système.

Description des classes existantes :

Classe médiateur	
opération	signification
#Decomposer la requête (requet:String) :String	Décompose la requête en sous requête.
Classe Tab_corr	
#recomposer les resultats (resxml:String,resrdf:String, resRuleml:String):String	Fusionne les resultats des sous requetes

Tableau 3.5: description de la classe mediateur

Classe conceptG		
Une classe qui contient les concepts du schéma global		
attribut	**type**	**signification**
#num_cpt_global	integer	Identifiant de concept globale
#cpt_global	String	Le concept globale
#ppt_global	String	Propriété de concept globale

Tableau 3.6: description de la classe conceptG

Classe Tab_corr :Une classe qui contient les concepts la correspondance entre les concepts locaux et globaux		
attribut	**type**	**signification**
#id	integer	Identifiant de la correspondane
#num_cpt_local	String	Identifiant de concept locale
#num_cpt_global	String	Identifiant de concept globale

Tableau 3.7: description de la classe Tab_corr

Classe conceptL		
Une classe qui contient les concepts des schémas locaux		
attribut	**type**	**signification**
#num_cpt_local	integer	Identifiant de concept locale
#cpt_local	String	Le concept local
#ppt_local	String	Propriété de concept locale
source	String	La source de concept

Tableau 3.8: Description de la classe conceptL

7. Architecture générale de notre système de médiation

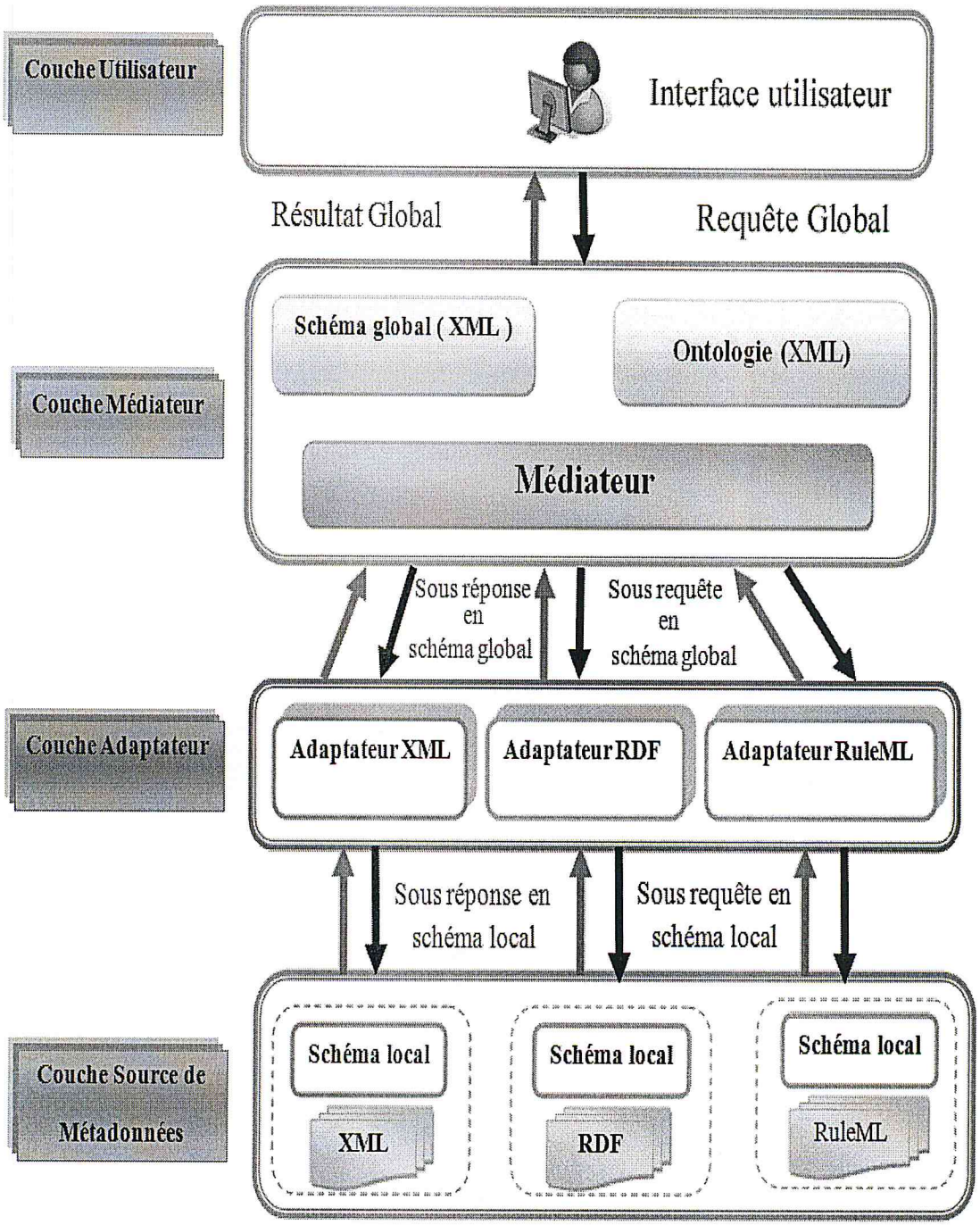


Figure 3.9 : Architecture de notre système de médiation.

8 . Description des couches de système

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

8.1 *Couche Source de Données*

les sources de métadonnées utilisées peuvent être de nature structurée, semi structurée ou bien non structurée. Dans notre approche les données sont semi structurées du fait qu'on s'intéresse à des documents XML, RDF, RuleML.

- **Une source de métadonnées XML** : c'est une source de métadonnée semi structurée qui porte le nom **«Médicament»**

Le schéma suivant présente le contenu de cette source :

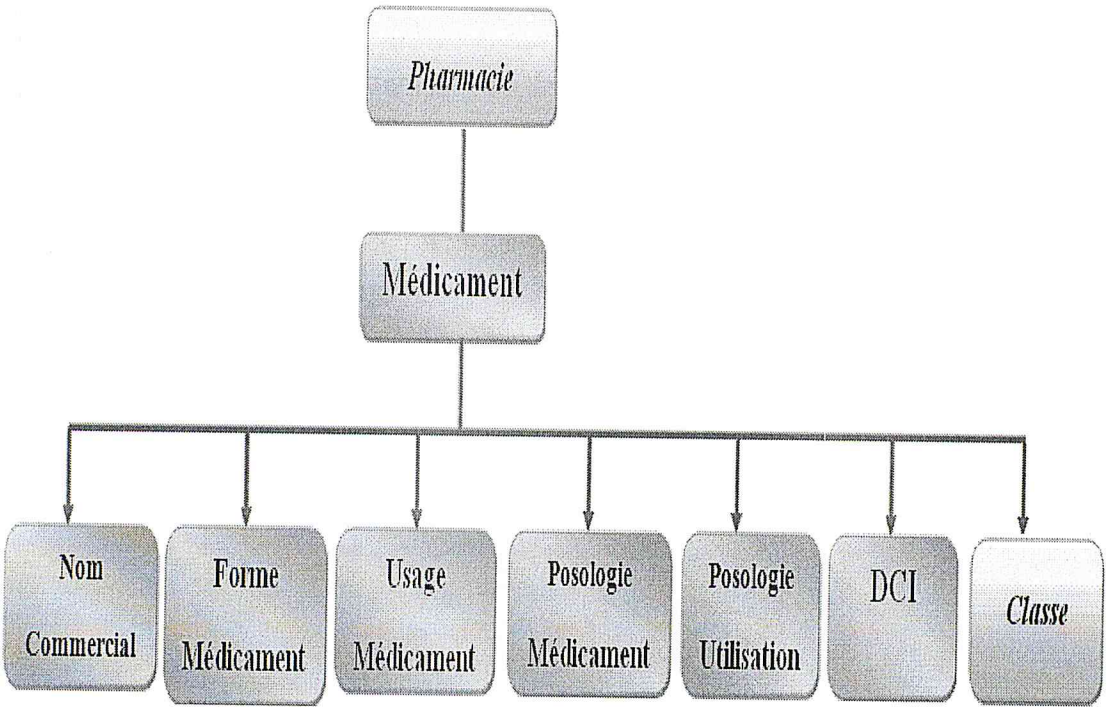


Figure 3.10 : Architecture de source XML

- **Une deuxième source de métadonnées RDF**: c'est une source de métadonnées semi structurée qui porte le nom **«Générique»**
- Le schéma suivant présente le contenu de cette source :

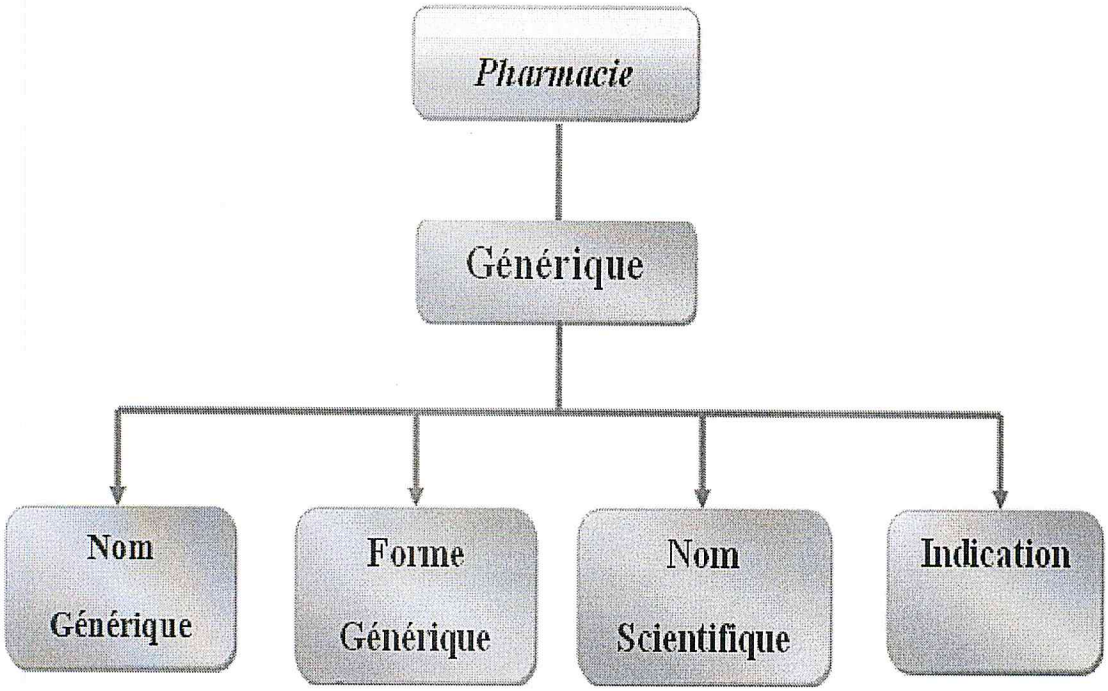


Figure 3.11 : Architecture de source RDF

- **Une source données RuleML** : c'est un document RuleML portant le nom « Médicament-possède-Générique », voici un fragment de ce document :

```
<ruleml:rulebase>

<Atom>
<Rel>possede</Rel>
<Var>Generique</Var>
<Var>Paralgan</Var>
</Atom>

<Atom>
<Rel>possede</Rel>
<Var>Paralgan</Var>
<Var>Doliprane</Var>
</Atom>

.....
</ruleml:rulebase>
```

Figure 3.12 : Fragment de la source RuleML

Chaque source de données est associée a un schéma local ;

➢ **schéma local** : c'est le schéma d'une source de métadonnées, il représente les structures dans laquelle les métadonnées sont stockées dans cette source. Ces schémas sont représentés par le modèle XML.

Les figure (3.13 , 3.14 , 3.15) présente des fragments des schéma locaux XML,RDF,Ruleml.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<pharmacie>
<Medicament>
<Nom_commercial></Nom_commercial>
<Forme_med> </Forme_med>
<Usage_med></Usage_med>

.....

</Medicament>
</pharmacie>
```

Figure 3.13 : Un fragment Schéma local de la source XML

```
<?xml version="1.0 " encoding="iso-8859-1"?>
<pharmacie>
<Generique>
        <Nom_generique></Nom_generique>
        <Forme_gene>
        </Forme_gene>
        <Nom_Scientifique></Nom_Scientifique>
        <Indication></Indication>
</Generique>
</pharmacie>
```

Figure 3.14 : Un fragment Schéma local de la source RDF

```
<? xml version="1.0" encoding="iso-8859-1"?>
<possede>
<Medicament>
<possede>
<Generique></Generique>
</possede>
</Medicament>
</possede>
```

Figure 3.15 : Un fragment Schéma local de la source RuleML

8.2 *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 » qui fait :

- La Traduction de la source de langage natif propre à la source en langage pivot.
- L'extraction des résultats de la sous requête envoyé par le médiateur.
- L'Envoie le résultat au médiateur.

- **Adaptateur RDF** : cet adaptateur est chargé de traduire la source **RDF** en une source **XML** et d'exécuter la requête sur la source traduite et de fournir le résultat au médiateur.
- **Adaptateur RuleML** : cet adaptateur est chargé de traduire la source **RuleML** en une source **XML** et d'exécuter la requête sur la source traduite et de fournir le résultat au médiateur.
- **Adaptateur XML :** est chargé d'interroger la source XML pour extraire une réponse.

8.3 *Couche Médiateur*

Il est décomposé en modules reliées entre eux. un médiateur doit :

1. Accepter les requêtes des utilisateurs.
2. Localiser les sources de données pertinentes.
3. Décomposer la requête en sous requête.
4. Envoyer chaque sous requête au adaptateur correspondant.
5. Recomposer les résultats des adaptateurs et envoie le résultat à l'utilisateur.

Afin de bien réaliser ses taches, 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 recomposition des résultats.

- **Schéma global :**

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

Nous allons construire notre 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.

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 sources de métadonnées a travers une table de correspondance.

Correspondances entre schémas Global / Local (Global as View) :

L'approche Global as View est celle utilisée pour décrire les correspondances entre schémas (schéma global /schémas des sources). C'est une approche ascendante depuis les sources vers le médiateur.

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SchemaGlobal>
<Concept>

<NomCptG> Medicament </NomCptG>
<NomAttG> Nom_medicament </NomAttG>
<NomAttG> Forme_medicament </NomAttG>
<NomAttG> Usage_medicament </NomAttG>

.....

</Concept>
</SchemaGlobal>
```

Figure 3.16 : Un fragment de Schéma Global

- **Ontologie :** qui sont définies comme « Une spécification explicite et formelle d'une conceptualisation commune» par Thomas Gruber **[40]** peuvent contribuer à la résolution du problème de l'hétérogénéité sémantique. En effet, elles offrent une description formelle des concepts.

  L'ontologie est un outil utilisé pour la représentation de la sémantique des informations.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Ontologie>
<Equivalents>
    <  CptG  >MedG</  CptG  >
    <Equivalent>Medicament</Equivalent>
    <Equivalent>Generique</Equivalent>
</Equivalents>
<Equivalents>
    <  AttG  >Nom_medG</ AttG  >
    <Equivalent>Nom_commercial</Equivalent>
    <Equivalent>Nom_generique</Equivalent>
</Equivalents>
.....
</Ontologie>
```

Figure 3.17 : Un fragment de fichier Ontologie en modèle XML

8.4 *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.

9. Description des modules

❖ Algorithme de Création du schéma Global

Algorithme : Création schéma global

Entrée : Schémas locaux, Ontologie

Sortie : Schéma global

Début

Connecter aux sources de métadonnées

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 Equivalent SC_i dans le fichier ontologie

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**

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

Chercher son Equivalent SA_i dans le fichier ontologie

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

a. Le module Traitement de requête

Le processus traitement de requête passe par trois phases principales :

- **Décomposition de requête et Localisation des sources**

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 :

Exemple

```
for $c in doc ("nom-fichier-xml")/racine
  for $a in $c/cpt-globale
return
  { $a
  }
```

dans notre domaine pharmaceutique un médicament donné peut avoir plusieurs génériques et ce dernier possèdes plusieurs propriétés.

Donc on s'intéresse a extraire le résultat d'un médicament ainsi que ses génériques.

Tel que :

Le médicament : se trouve dans la source de métadonnées XML.

Le générique : se trouve dans la source de métadonnées RDF.

La relation médicament possède des générique :se trouve dans la source de métadonnées RULEML.

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

```
for $c in doc ("schemaglobale.xml")/pharmacie
where $a/nom_medG="paralgan"
return
  { $a
  }
```

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

```
Sous Requête 1: for $c in doc ("sourceXML.xml")/pharmacie
                  for $a1 in $c/nom_commercial="Paralgan"
```

return
{$a1}

Sous Requête 2: for $c in doc ("source-RuleML.xml")/ pharmacie
for $a2 in $c/nom_generique
return
{$a2}

Sous Requête 3: for $c in doc ("sourceRDF.xml")/ pharmacie
for $a3 in $c/ DCI
return {$a3}

b. Le module Recomposition 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 : Recomposition 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

10. Conclusion

Dans ce chapitre nous avons présenté une conception détaillée de notre système médiation en utilisant le langage UML, Apres avoir définie l'architecture du système avec ses composants nous allons adopter l'approche pour porté de la conception en

utilisant une ontologie pour traiter l'hétérogénéité . La prochaine étape est de la réalisation notre médiateur et les outils d'implémentation.

Chapitre IV

Implémentation et Validation du système

1. Introduction

Après l'expression des besoins et la modélisation du système à développer, nous allons dans cette partie,définir les outils de développement choisis pour l'implémentation de notre système . Ensuite, nous présentons notre application.

2. Implémentation

2.1. Présentation des outils de développement

2.1.1. Le langage de programmation JAVA

Pour la réalisation de notre projet nous avons utilisé le langage JAVA qui est un langage de programmation orienté objet développer par SUN, le choix de ce langage du aux avantages suivant : **[41]**

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

2 .1.2. Implémentation des sources

2.1.2.1. Altova SemanticWorks 2012

Altova SemanticWorks 2012 est un éditeur de documents RDF et IDE développement d'une ontologie.il vous permet de :

- ✓ Graphiquement créer et éditer des documents RDF, les documents RDF Schéma et OWL ontologies.
- ✓ Vérifiez la syntaxe et la sémantique des ontologies que vous les modifiez, et la syntaxe de RDF documents.
- ✓ Convertir ontologies graphique crées dans le RDF/XML et les N-Triples formats.

Avec Altova SemanticWorks 2012, donc, en plus d'être en mesure d'éditer des documents RDF en une interface graphique et de vérifier sa syntaxe, vous pouvez concevoir le schéma RDF et des ontologies OWL en utilisant un vue de conception graphique , vérifier la syntaxe de tout RDF Schéma ou une ontologie OWL et la sémantique de OWL Lite et OWL DL ontologies et les ontologies à l'exportation dans le RDF/XML et N-Triples formats**[42]**

2.1.2.2. Stylus Studio 2011 XML Entreprise

Offre un ensemble complet d'outils XML et des fonctionnalités pour travailler avec XML, XQuery, les services Web XML édition, et de nombreuses autres technologies XML.il inclut un XML Validateur, il offre aussi une gamme complète de soutenir le développement XSLT, y compris le débogage XSLT, cartographie XSLT, le profilage XSLT **[43]**

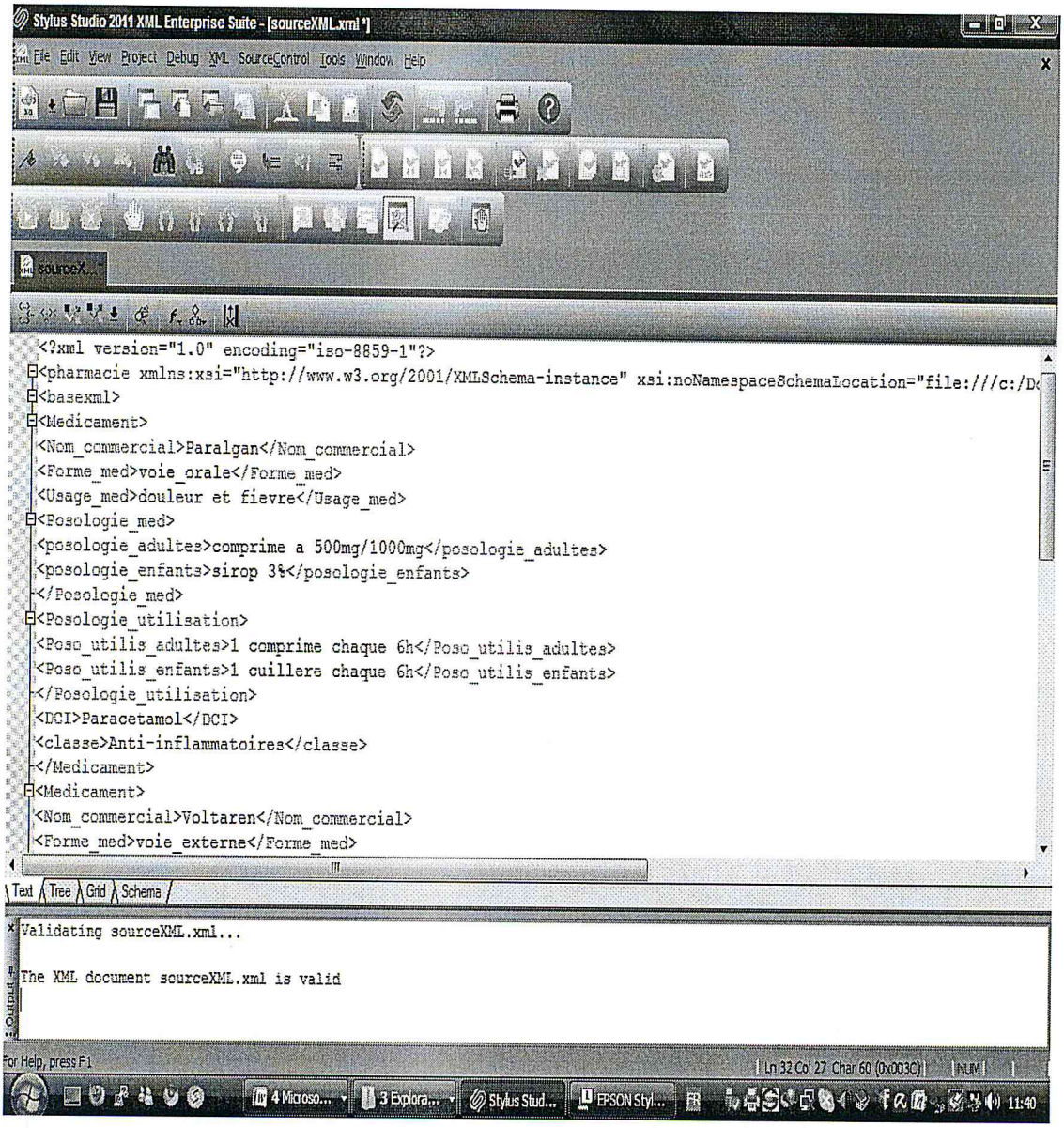


Figure 4.1: Le contenu de notre Source de métadonnées XML sur Stylus Studio 2011.

2.1.2.3. DR-Device

Un environnement de développement visuel intégré pour le développement et l'utilisation de bases annulable règle logique sur le dessus d'ontologies RDF. un éditeur de règles visuelle RuleML-conforme qui contraint le vocabulaire autorisé par l'analyse des ontologies RDF entrée et un système de raisonnement révisable que les processus de

données RDF et des ontologies RDF Schéma. **[44]**

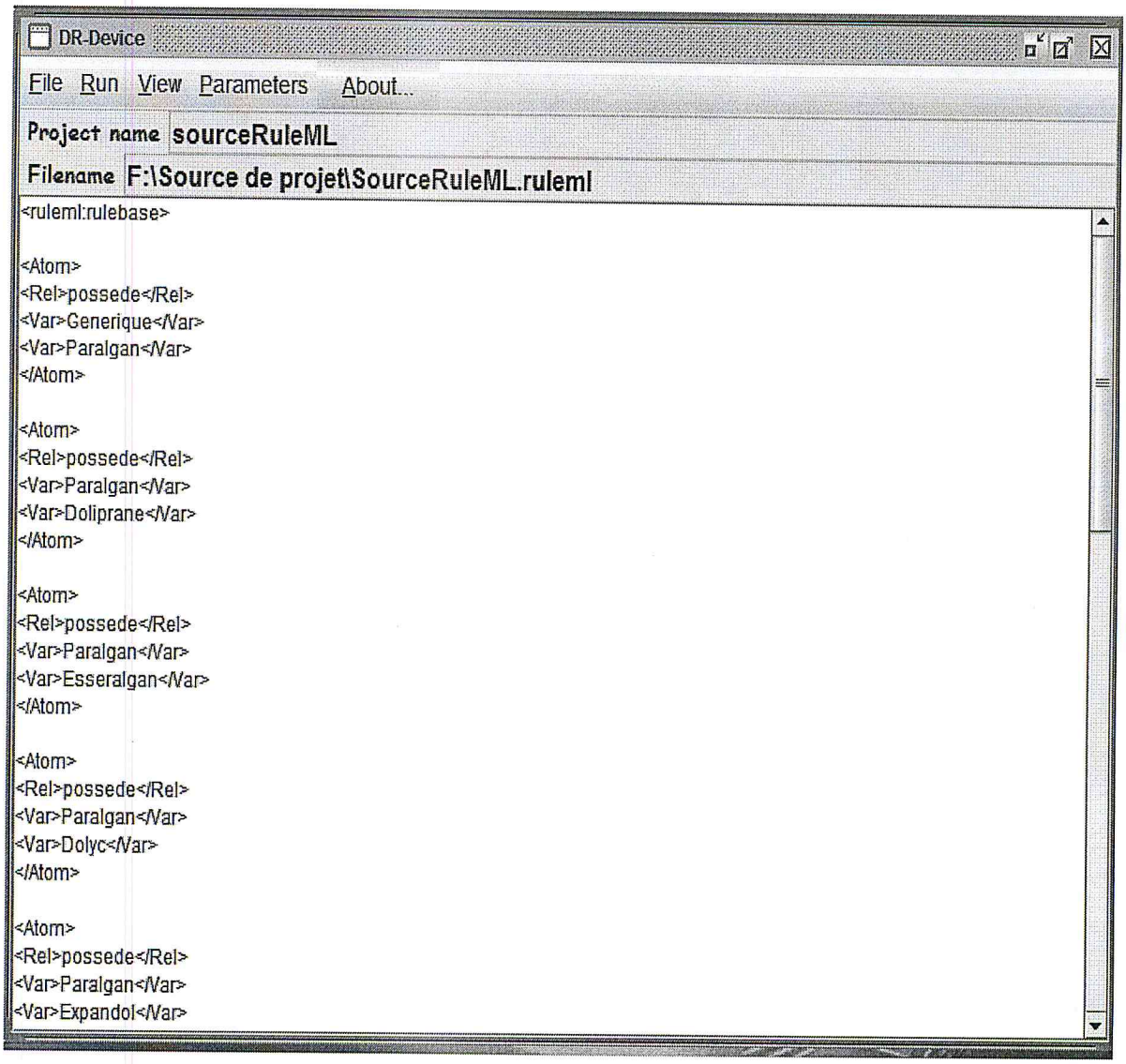


Figure 4.2 : Le contenu de notre Source de métadonnées RuleML sur DR-Device

2.1.2.4. L'API JDOM

JDOM est une API open source Java dont le but est de représenter et manipuler un document XML de manière intuitive pour un développeur Java sans requérir une connaissance pointue de XML. Par exemple, JDOM utilise des classes

plutôt que des interfaces. Ainsi pour créer un nouvel élément, il faut simplement instancier une classe.

Malgré la similitude de nom entre JDOM et DOM, ces deux API sont très différentes. JDOM est une API uniquement Java car elle s'appuie sur un ensemble de classes de l'API Java notamment celles de l'API Collection.

- **La présentation de JDOM**

Le but de JDOM n'est pas de définir un nouveau type de parseur mais de faciliter la manipulation au sens large de document XML : lecture d'un document, représentation sous forme d'arborescence, manipulation de cet arbre, définition d'un nouveau document, exportation vers plusieurs formats cibles ...

JDOM est donc un modèle de documents objets open source dédié à Java pour encapsuler un document XML. JDOM propose aussi une intégration de SAX, DOM, XSLT et XPath.

JDOM n'est pas un parseur : il a d'ailleurs besoin d'un parseur externe de type SAX ou DOM pour analyser un document et créer la hiérarchie d'objets relative à un document XML. L'utilisation d'un parseur de type SAX est recommandée car elle consomme moins de ressources que DOM pour cette opération. Par défaut, JDOM utilise le parseur défini via JAXP.

JDOM interagit donc avec SAX et DOM pour créer un document en utilisant ces parseurs ou pour exporter un document vers ces API, ce qui permet de facilement intégrer JDOM dans des traitements existants. JDOM propose cependant sa propre API.

- **Les fonctionnalités et les caractéristiques**

JDOM propose plusieurs fonctionnalités :

- Création de documents XML
- Encapsulation d'un document XML sous la forme d'objets Java de l'API
- Exportation d'un document dans un fichier, un flux SAX ou un arbre DOM
- Support de XSLT
- Support de XPath

Les points caractéristiques de l'API JDOM sont :

- elle est développée spécifiquement en et pour Java en utilisant les fonctionnalités de Java au niveau syntaxique et sémantique (utilisation des collections de Java 2, de l'opérateur new pour instancier des éléments, redéfinition des méthodes equals(), hashCode(), toString(), implémentation des interfaces Cloneable et Serializable, ...)
- elle se veut intuitive et productive notamment grâce à des classes dédiées à chaque élément instancié via leur constructeur et l'utilisation de getter/setter
  Exemple pour obtenir le texte d'un élément
  DOM : String content = element.getFirstChild().getValue();
  JDOM : String text = element.getText();
- elle se veut rapide et légère
- elle veut masquer la complexité de certains aspects de XML tout en respectant ses spécifications
- elle doit permettre les interactions entre SAX et DOM. JDOM peut encapsuler un document XML dans un hiérarchie d'objets à partir d'un flux, d'un arbre DOM ou d'événements SAX. Il est aussi capable d'exporter un document dans ces différents formats. **[45]**

2.1.2.5. Le SGBD MYSQL

Un serveur de bases de données stocke les données dans des tables séparées plutôt que de tout rassembler dans une seule table. Cela améliore la rapidité et la souplesse de l'ensemble. Les tables sont reliées par des relations définies, qui rendent

possible la combinaison de données entre plusieurs tables durant une requête. Le SQL dans "MySQL" signifie "Structured Query Language" : le langage standard pour les traitements de bases de données **[46]**

2 .1.2.3. Langage d'interrogation pour XML : XQuery

- **Présentation**

XQuery est le langage de manipulation des documents XML servant à la consultation de l'information contenue dans un document XML. XQuery est basé sur le **XPath** (un langage pour localiser une portion d'un document XML)

XQuery est conçu pour interroger des données XML, pas seulement sous forme de document XML, mais tout ce qui peut apparaît comme XML dans le sens large de bases de données. XQuery est au XML ce que SQL est aux tables de la base de données.

- **Spécification et Syntaxe**

Le langage minimal de XQuery: se base sur la norme XPath 2 (qui spécifie le langage de requête XML proprement dit), ayant comme cœur l'expression FLWOR (For , Let , Where ,Order by ,Return) :

L'expression FLWOR est une instruction de boucle, avec de nombreuses fonctionnalités, qui est assez similaire au SELECT de SQL. Grâce au **Where,** il est possible d'écrire des jointures internes ou externes.

- **for :** Fourni un mécanisme d'itération.
- **let :** Permet l'assignation de variable.
- **where :** Les clauses for et let génèrent un ensemble de noeuds qui peuvent être filtré par un ou plusieurs prédicats dans une clause where.
- **return :** génère le résultat de l'expression FLWR. Elle contient généralement un ou plusieurs éléments constructeurs et/ou des références à des variables, et est exécutée une fois pour chaque noeud renvoyé par les clauses For/Let/Where.

Voici un exemple de requête XQuery basée sur l'expression FLWOR :

```
for $x in doc("sourceXML.xml")/pharmacie/Medicament
where $x/DCI= "Paracetamol"
return $x/Nom_commercial
```

Ici la clause « for » choisit tous les éléments « Medicament » sous l'élément de « pharmacie » dans une variable appelée $x. La clause « where » permet de choisir les « Medicament » dont le DCI est égale à Paracetamol et la clause « return » spécifie ce qui devrait être rendu. Ici il rend Le résultat de cette requête contient les éléments de Nom_commercial. **[47]**

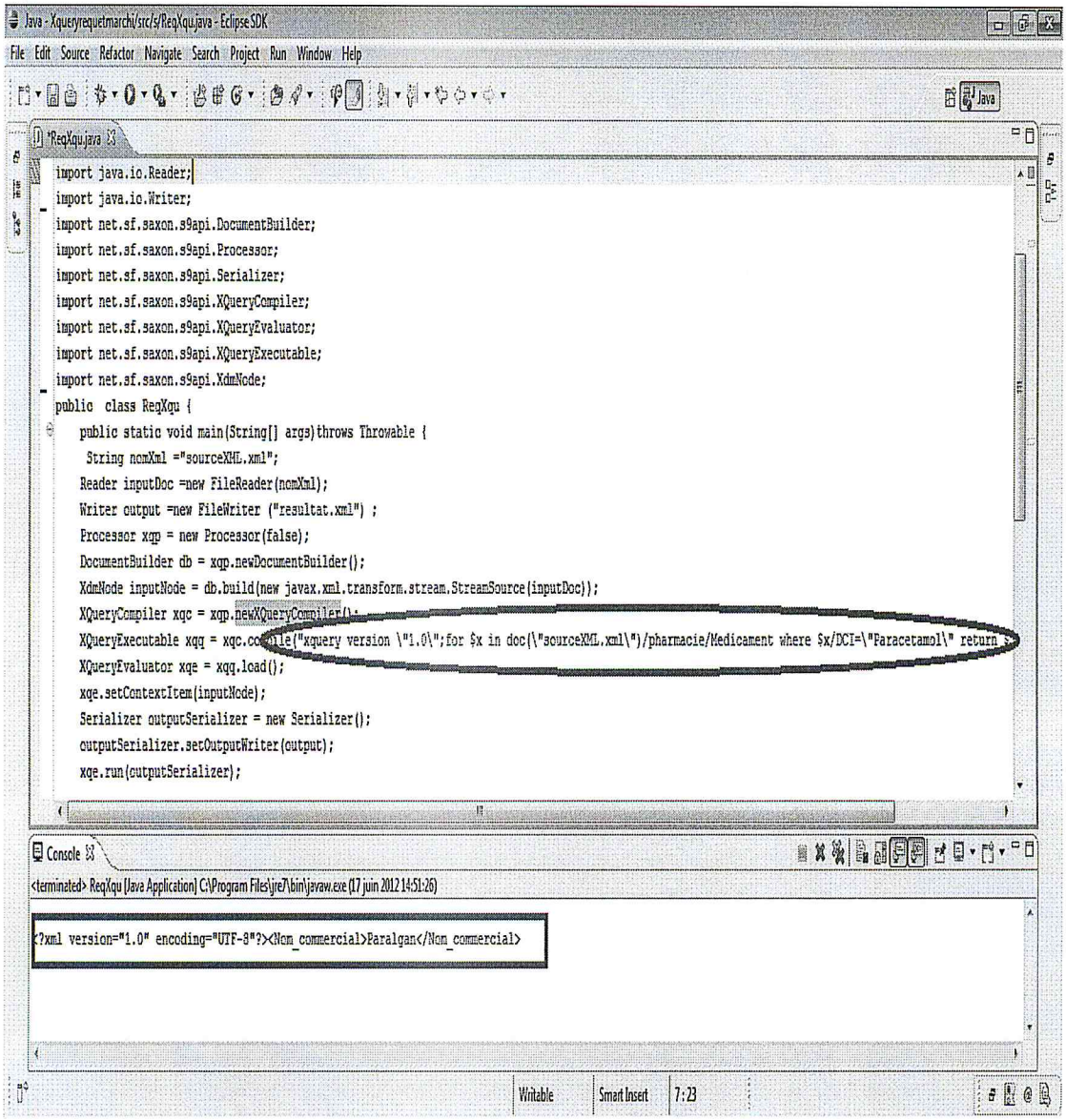


Figure 4.4 : Exemple requête Xquery par code java .

3. test

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

3.1. Fenêtre d'Accueil

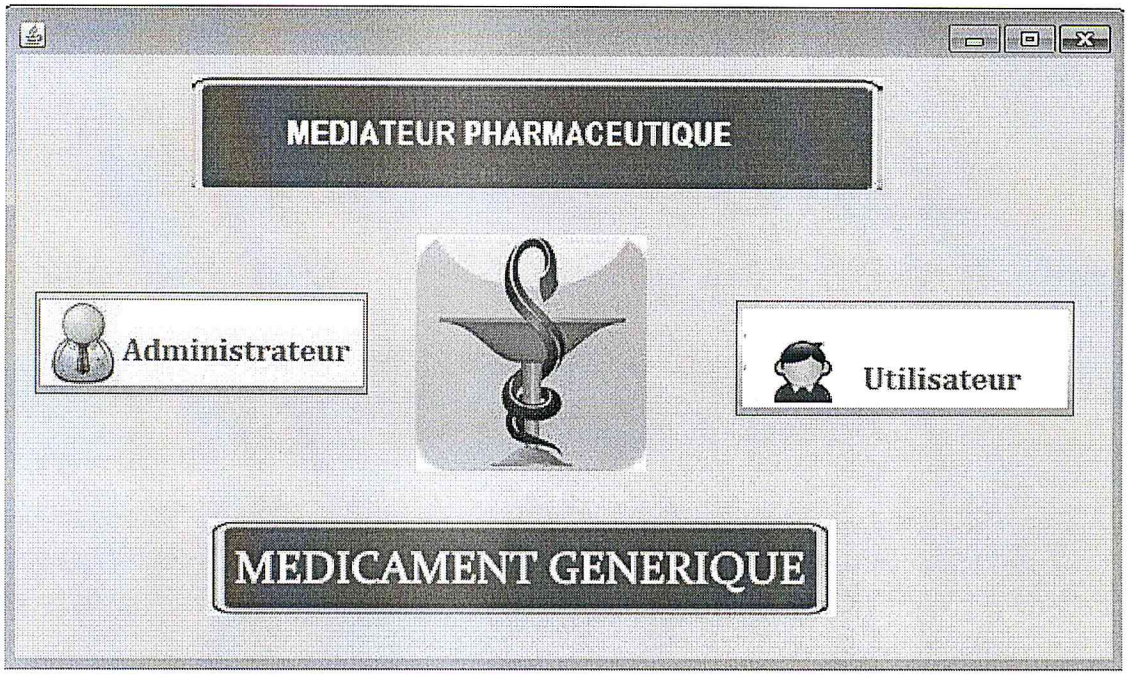


Figure 4.5 : Fenêtre d'Accueil.

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

- L'administrateur s'authentifier,
- Pour les utilisateurs qui ne disposent pas des droits d'accès administrateur, ils peuvent accéder à notre système de manière restreinte, ils peuvent juste poser les requête et voir le résultat final.

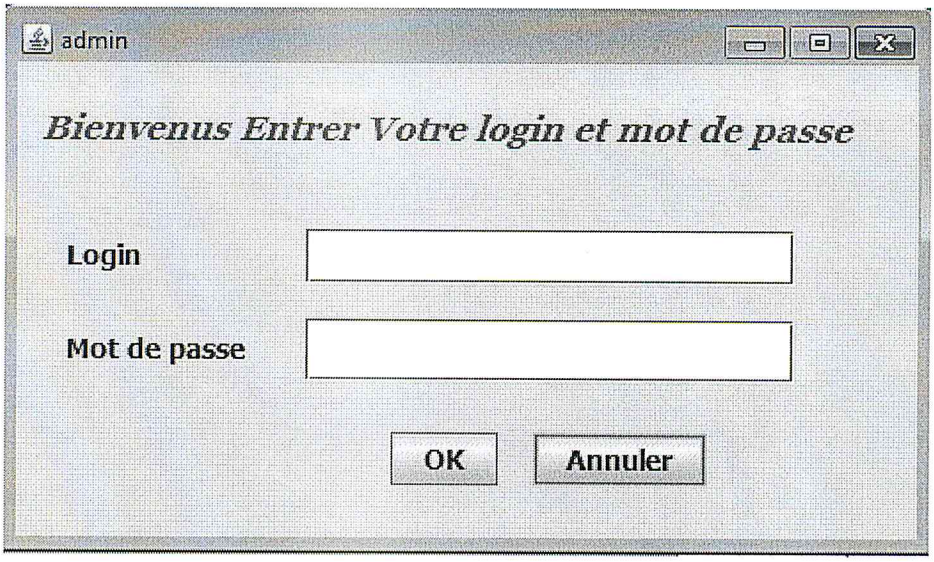


Figure 4.6 : Fenêtre d'authentification.

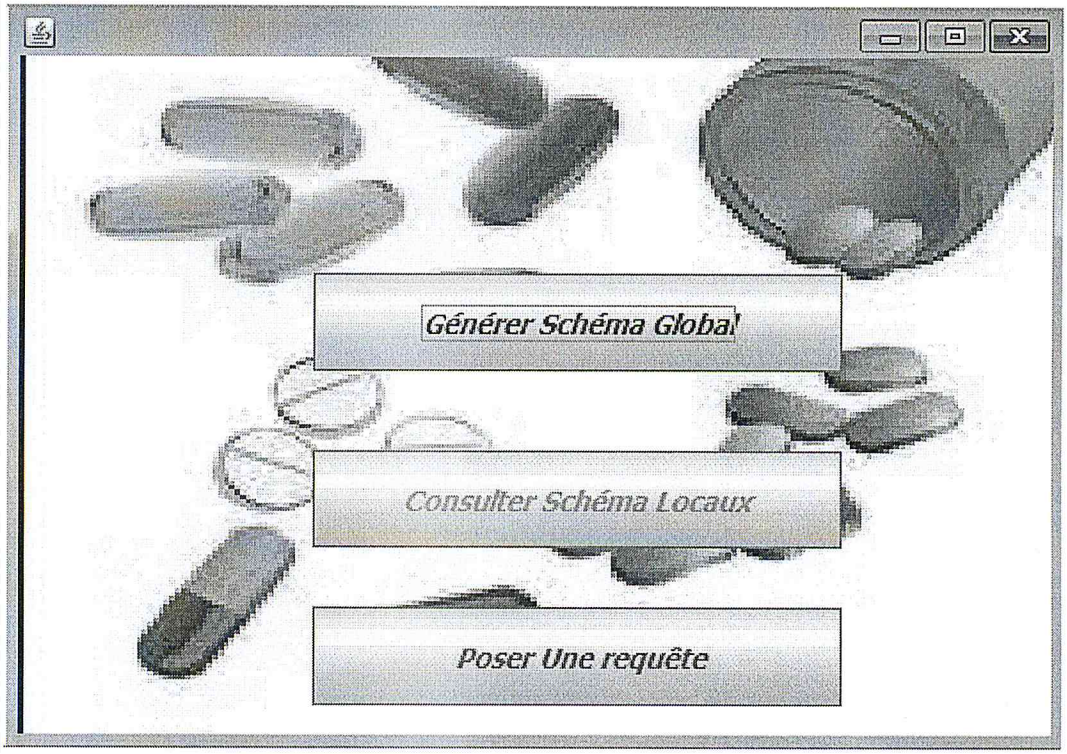


Figure 4.7: Fenêtre Menu Administrateur.

Cette interface donne le choix a administrateur de créer le schéma global , consulter les schéma locaux ou de poser requête.

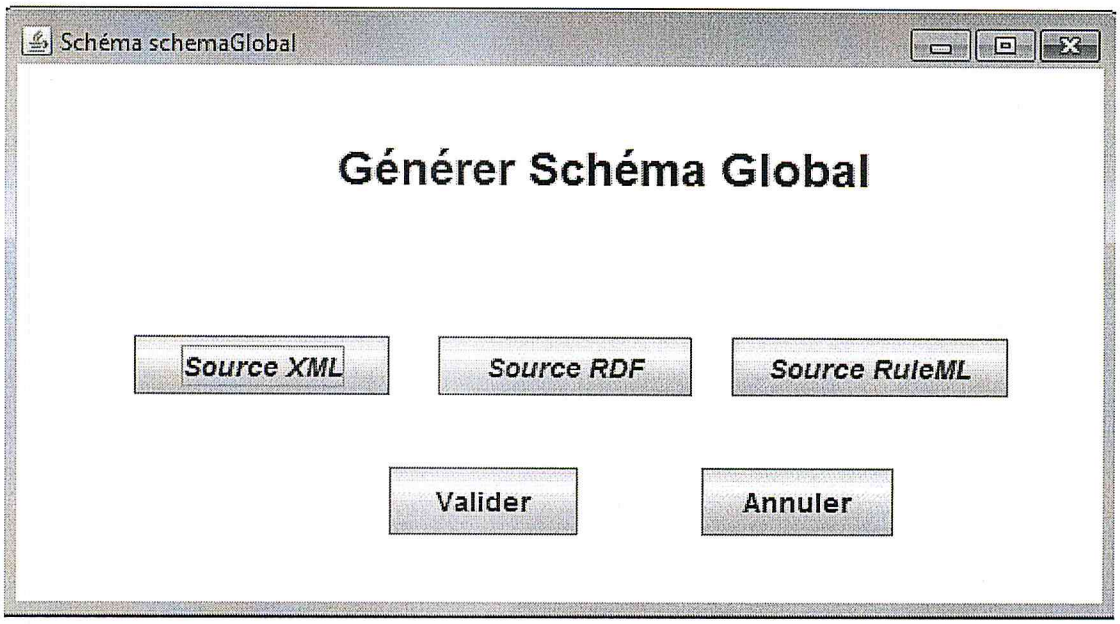


Figure 4.8: Fenêtre Générer Schéma Global.

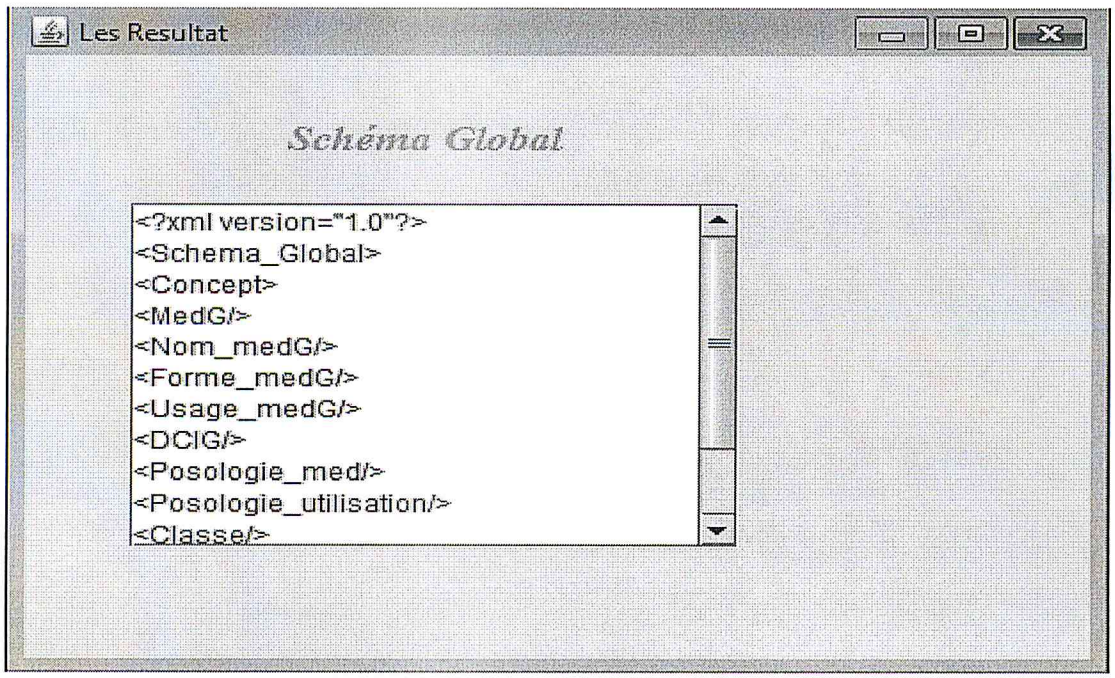


Figure 4.9: Fenêtre Schéma Global.

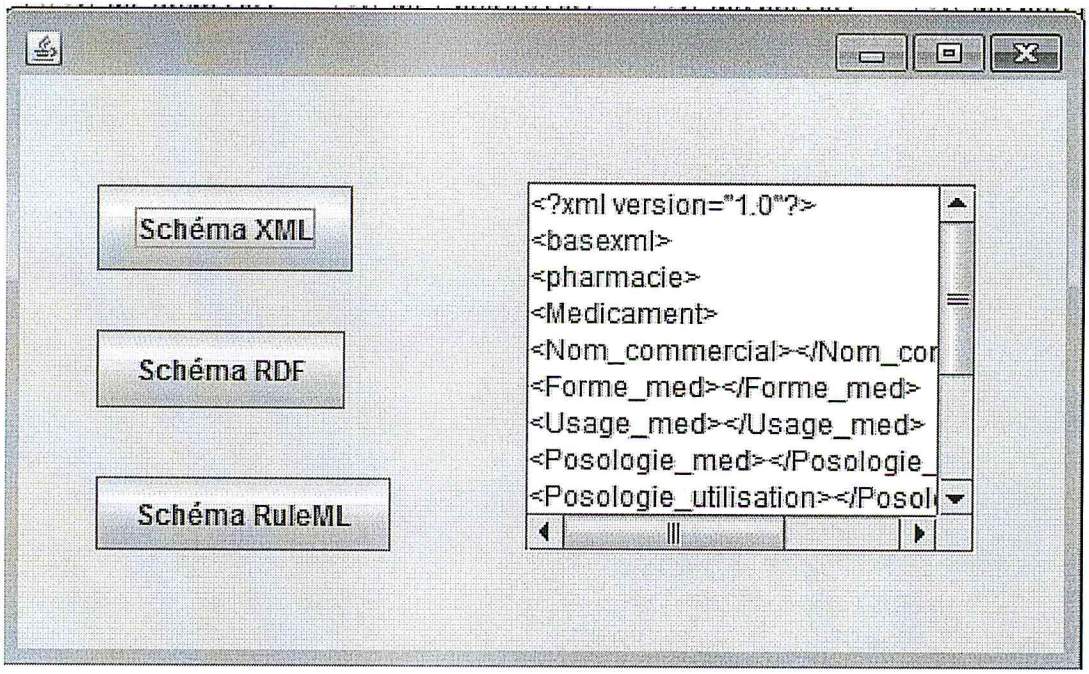


Figure 4.10: Fenêtre Consultation de schéma locaux.

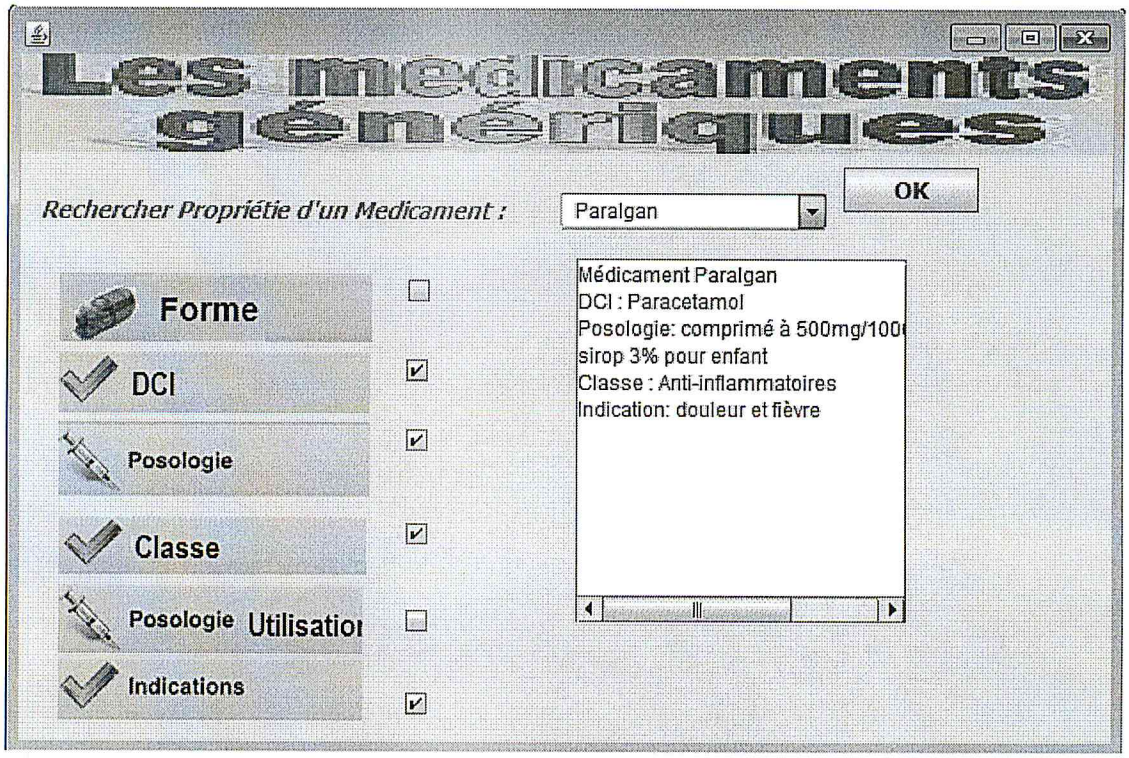


Figure 4.11: Fenêtre pour poser une requête par propriété d'un médicament.

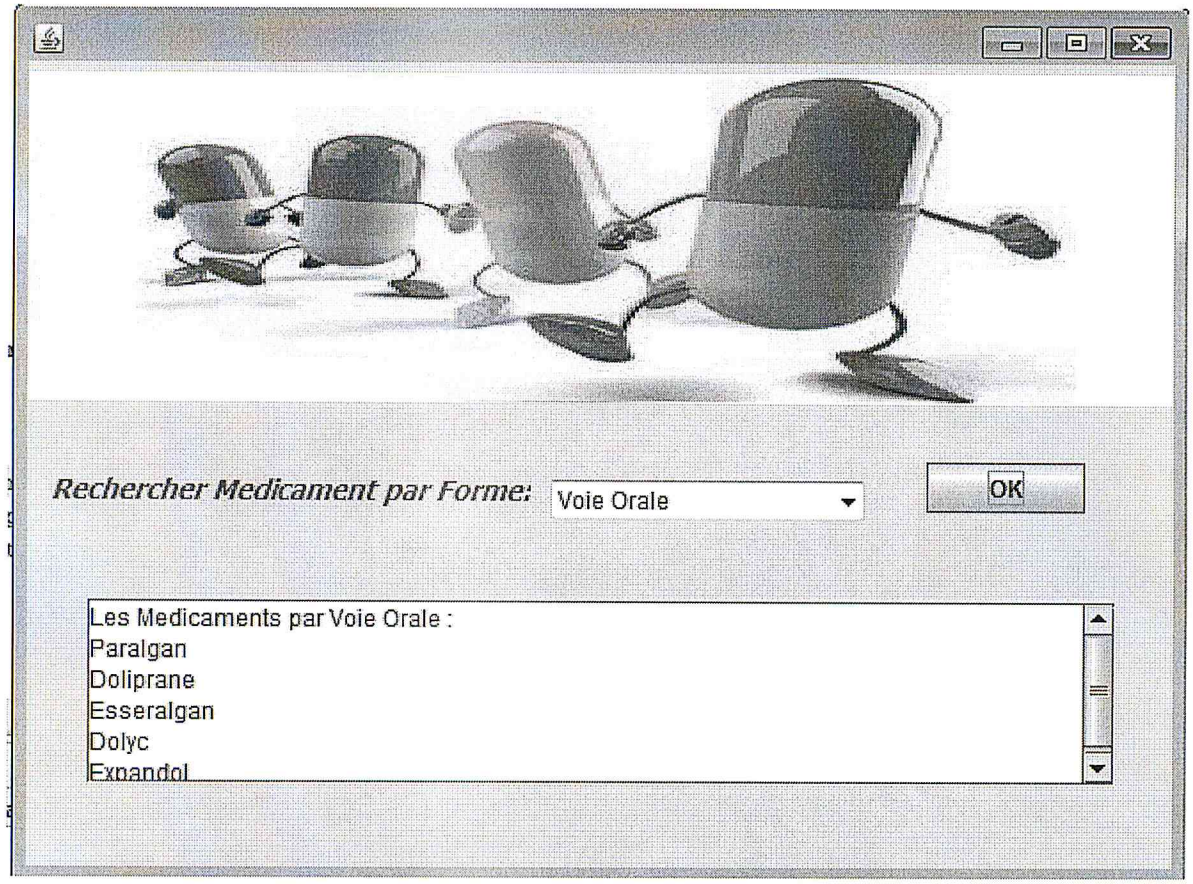


Figure 4.12: Fenêtre pour poser une requête par forme d'un médicament.

4. Conclusion

Dans cette partie nous avons présenté l'environnement de déroulement pour notre système ,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

Conclusion générale

Notre travail consiste principalement à proposer une solution au problème d'intégration de sources de données hétérogènes il supporte principalement les objectifs suivants: la transparence d'accès a la donnée, la gestion des conflits sémantiques.

On utilisant l'approche de type médiation ceci nous a permis d'adopter l'approche GAV qui est une approche d'intégration des sources hétérogènes basée sur la notion de schéma global crées à partir des schémas sources (construites d'après notre trois sources des métadonnées RDF, XML, RULEML) ce schéma global est représenté par le modèle pivot XML que nous avons choisit.

Notre solution est principalement caractérisée par :

1- Elle offre une transparence d'accès à la localisation, aux schémas sources et aux langages de requêtes des données sources.

2- Offre une intégration sémantique des données en utilisant de nouvelles technologies pour la représentation de la sémantique des informations et l'utilisation des ontologies;

4- Les données hétérogènes supportées sont des données RDF, RULEML et XML.

5- Utilise le langage d'interrogation expressive commun XQuery comme modèle de données commun;

6- Approche d'intégration descendante GAV on définissons les règles de mapping comme des correspondances entre des requêtes XQuery écrites en terme du schéma global et des requêtes XQuery écrites en terme des schémas sources;

7- Traitement des sous requêtes destinées aux sources de données se fait sémantiquement, en prend en compte les conflits sémantiques;

Sans doute, ce travail est susceptible d'amélioration. Nous en proposons les suivants :

1- associer d'autres privilèges a l'administrateur du système tel que la possibilité de mise à jour de l'ontologie.

2- Exploiter l'historique des requêtes qui ont été exécuté pour améliorer la reformulation des requêtes.

3- l'ajout de nouveaux types de sources tel que les sources semi structurées (fichiers textes).

En fin sur le plan technique ce projet nous a apprend une certaine expérience dans le développement par les technologies suivantes : Java, DOM et XQuery.

Bibliographie

[1] Xavier BARIL. Un modèle de vues pour l'intégration de sources de données XML: VIMIX. Thèse de Doctorat en Informatique, de l'Université des sciences et techniques du Languedoc. Décembre 2003.

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

[3] Susanne Busse, Ralf-Detlef Kutsche, Ulf Leser, Herbert Weber, "Federated Information Systems: Concepts, Terminology and Architectures", Technische Université Berlin,1999.

[4] Evaggelia Pitoura, Omran Bukhres, and Ahmed Elmagarmid. Object Orientation in Multidatabase Systems. ACM Computing Surveys, 27(2) :141-195, june 1995.

[5] Genoveva Vargas Solar_, Anne Doucet , « Médiation de données : solutions et problèmes ouverts », PARIS, 2002.

[6] Fabrice Jouanot, « DILEMMA : vers une coopération de systèmes d'informations basée sur la médiation sémantique et la fusion d'objets », université de bourgogne, novembre 2001.

[7] Djamel Benslimane et al, « Interoperabilité de SIG : un état de l'art », université de Bourgogne, Lyon1,1999.

[8] G. Wiederhold. Mediators in the architecture of future information systems. Computer, 25(3) :38–49, 1992.

[9] G. Wiederhold and M. Genesereth. The conceptual basis for mediation services. 1997.

[10] Sophie Cluet, Claude Delobel, Jérôme Siméon, et Katarzyna Smaga. Your Mediators Need Data Conversion ! In SIGMOD Conference, pages 177-188, 1998

[11] Georges GARDARIN ,Tuyêt Trâm DANG NGOC. Intégration de données hétérogènes distribuées. Laboratoire PRiSM, Université de Versailles-Saint-Quentin.

[12] C. MOREL-PAIR, « Panorama : des métadonnées pour les ressources électroniques» Disponible sur : http://hal.ccsd.cnrs.fr/docs/00/04/04/73/PDF/Metas_panorama_CMO.pdf, 2005.

[13] C. MOREL-PAIR, « Métadonnées, pour quoi faire ? », Disponible sur http://artist.inist.fr/article.php3?id_article=384 ; mars 2007.

[14] A. J. GILLILAND SWETLAND ; « Introduction to metadata : Setting the stage » ; Los Angeles; Getty Information Institute; Disponible sur:http://www.getty.edu/research/institute/standards/intrometadata/, 2000.

[15] E. COLINET, Inge ALBERTS ; « Les métadonnées nécessaires à la préservation de l'information numérique » ; École de bibliothéconomie et des sciences de l'information, Université de Montréal ; Disponible sur : http://www.esi.umontreal.ca/~albertsi/INU1030/Cours/cours10.html, mars 2003.

[16] A. J. GILLILAND SWETLAND ; « Difining metadata » dans Introduction to metadata : Pathways to Digital information ; Edité par Murtha Baca ; Los Angeles ; Getty Information Institute ; 2000.

[17] Jaouaf Mohamed Amine et Mehenni Noureddine, Génération automatique des requêtes de médiation dans un contexte relationnel, thèse d'ingénieur d'état à université ibn khaldoun-Tiaret ,2009-2010.

[18] Samir Yahiaoui, Structuration de métadonnées en vue d'un filtrage d'information sur le web, thèse Magister à Institut National de formation en Informatique (INI) Oued Smar – Alger,2008

[19] Y. AMEROUALI. « Métadonnées basées sur l'association d'éléments de description de ressources et d'éléments de profil d'utilisateur », Disponible sur : http://www.enssib.fr/bibliotheque/documents/theses/amerouali/amerouali.pdf, 2001.

[20] Patrick Peccatte, Métadonnées: une initiation Dublin Core, IPTC, Exif, RDF, XMP, etc. août 2002

[21] Didier Girard, Tanguy Crusson, XML pour l'entreprise, Version provisoire 0.91,2001, http://www.application-servers.com/livresblancs/xml/.

[22] Najib Tounsi, Le Langage XML: Fondations pour les Plateformes eLearning XML et les Technologies Associées, Ecole Mohammadia d'Ingénieurs Bureau W3C Maroc, Rabat 28 Nov. 2005

[23] Jean-Jacques Thomasson ,XML Schema tome 0 : Introduction Recommandation du W3C du 2 Mai 2001, http://xmlfr.org/w3c /TR/ xmlschema-0/

[24] Grégory CHAZALON Joséphine LEMOINE, les schémas XML, Publié le 21 janvier 2001 sur http://site.voila.fr/xmlschema/

[25] Cécilia COSTES, Ronan HERVE, LES TECHNOLOGIES EMERGENTES DU WEB SEMANTIQUE, , école polytechnique de l'Université de Nantes ,PROJET SILR 3, Le 24 / 01 / 2002

[26] Houssein Ben-Ameur et Youssef Bououlid-Idrissi , Agents, the Semantic Web and Semantic Web Services 26 Mars 2003 .

[27] Lassila, O. et Swick, R. R. (1999). Resource description framework (rdf) model and syntax specification. World Wide Web Consortium W3C Recommendation 22 February 1999 / W3C /.

[28] Daniel K. Schneider , Resource Description Framework Code: xml-rdf , Version: 0.3 (modifié le 30/11/06) .

[29] Patrick Peccatte, Les métadonnées un élément clé de la gestion de contenu , IPA Systems S.A.

[30] McBride, B. (2004). The resource description framework (rdf) and its vocabulary description language rdfs. Handbook on Ontologies, pages 51_66.

[31] G. Chagnon, Cours de XML - Initiation aux Schéma XML , Disponible sur : http://www.gchagnon.fr/cours/xml/schema.html, 2004

[32] Riichiro Mizoguchi and John K. Slaney, editors. PRICAI 2000, Topics in Artificial Intelligence, 6th Pacific Rim International Conference on Artificial Intelligence,Melbourne, Australia, August 28 - September 1, 2000, Proceedings, volume 1886 of Lecture Notes in Computer Science. Springer, 2000.

[33] Harold Boley, Said Tabet, and Gerd Wagner. Design Rationale for RuleML : A Markup Language for Semantic Web Rules. In SWWS, pages 381_401, 2001.

[34] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL : A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, May 2004.

[35] Wagner G., Tabet S., et Boley H., MOF-RuleML: The Abstract Syntax of RuleML as a MOF Model. In Proceedings of Integrate, 2003.

[36] Diane Hillmann, Guide d'utilisation du Dublin Core, Disponible sur : http://www.bibl.ulaval.ca/DublinCore/usageguide-20000716fr.htm, 2001.

[37] Laurent Audibert, UML 2 - de l'apprentissage à la pratique , FNAC, amazon.fr,2006

[38] ZENDAOUI FAIROUZ et DRICI CHERIFA, Conception et réalisation d'une application web pour la gestion du prêt entre bibliothèques (mémoire d'Ingénieur d'Etat), 2009-2010

[39] Michel VOLLE, Langage de modélisation UML, 2002

[40] Thomas R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, Formal Ontology in Conceptual Analysis and Knowledge Representation, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[41] JEROM BOUGEAUT,java la maitrise edition 2008

[42] http://www.litefile.com/altova-semanticworks.html

[43] http://www.componentsource.com/products/stylus-studio-xml-enterprise/index-fr.html

[44] http://www.springerlink.com/content/pq37651733847848

[45] http://jmdoudoux.developpez.com/cours/developpons/java/chap-jdom.php

[46] http://www.futura-sciences.com/fr/definition/t/internet-/2/d/mysql_4640, 2011

[47] http://www.w3schools.com/xquery/default.asp

[48] Harold Boley, Said Tabet, and Gerd Wagner. Design Rationale for RuleML : A Markup Language for Semantic Web Rules. In SWWS, pages 381_401, 2001.

[49] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL : A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, May 2004.

[50] Wagner G., Tabet S., et Boley H., MOF-RuleML: The Abstract Syntax of RuleML as a MOF Model. In Proceedings of Integrate, 2003.