République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : SI

Sujet :

Intégration de données :
Réalisation d'une plateforme pour l'interrogation de plusieurs sources de données hétérogènes

Présenté par :
Mr. SALHI HOCINE
Mr. HASNAOUI MOHAMMED

Promotrice : M^me^. L OUAHRANI
Encadreur: Mr. M. RAHIM

Organisme d'accueil :

NAFTAL SPA *(Direction Centrale des Systèmes d'Information DCSI)*

Soutenue le: date soutenance, devant le jury composé de :

Nom président du jury, grade, organisme	**Président**
Nom examinateur 1, grade, organisme	**Examinateur**
Nom examinateur 2, grade, organisme	**Examinateur**

- 2006/2007-

MIG-004-153-1

Remerciement

Avant tout nous remercions dieu tout puisant qui nous a donné la force du parvenir au bout de nos peines.

Nous tenons à adresser nos sincères remerciements à notre promotrice Mme : L. OUAHRANI qui a su nous conseiller par ses critiques constructives et nous guider dans notre travail.

Nous remercions aussi les membres du département informatique, surtout les membres de jury.
Nous remercions tous les travailleurs de la société NAFTAL, surtout notre encadreur Mr : M. RAHIM.

Nos sentiments de profonde gratitude vont à nos professeurs qui tout au long des années d'études nous ont transmit leur savoir sans réserve.

Nos remerciements vont aussi à tous ceux et celles qui ont participé de près ou de loin à l'élaboration du présent travail.

Enfin, nous tenons à remercier tous nos amis et collègues pour leur soutien moral tout au long de la préparation de ce mémoire.

Résumé

Le *présent travail a pour objectif la réalisation d'une plate forme pour* l'interrogation des plusieurs sources de données distribuées et hétérogènes, répondant aux besoins du direction DCSI de la société **NAFTAL**, pour accéder directement, simultanément et d'une manière transparente à plusieurs sources de données distribuées et hétérogènes. Ces sources de données peuvent être structurées (base de données relationnelle...), semi structurées (doc XML...) et non structurées (fichier a plat...).

Pour atteindre cet objectif, nous basons sur l'approche médiateur qui consiste à construire un modèle commun des sources de données sous forme de schéma XML, et qui permet d'analyser les sources de données sans se préoccuper de leurs emplacements physiques ou logiques.

Mots clés: sources de données distribuées et hétérogènes, structurée, semi structurée, non structurée, schéma XML, modèle commun, médiateur.

Abstract

The *present work has as an objective the realization of plat form for the* questioning of many data sources distributed and heterogeneous, responding to the needs of the **CDIS** of **NAFTAL** company to accede directly, simultaneously and by transparent manner at many sources of distribution and heterogeneous data sources, these ones can be structured (relational data base), semi structured (XML document) and no structured (flat file).

To attain this objective, we base on the mediatory approach that consists on constructing a common model of data sources under the XML schema that permits to analyze the data sources without coring about them physical or logic positions.

Keywords: data sources of distribution and heterogeneous, structured, semi structured, no structured, XML schema, common model, mediatory.

ملخـــــــــص

الهدف من العمل المنجز هو إنشاء أرضية تسمح باستجواب مجموعة من مصادر البيانات، موزعة و هجينة، وذلك لتحقيق بعض متطلبات المديرية المركزية لنظام المعلوماتية لشركة نافطال ، للمرور مباشرة، في وقت واحد، و بطريقة شفافة إلى مجموعة من مصادر البيانات موزعة و هجينة، هذه المصادر يمكن أن تكون منتظمة (قاعدة بيانات...)، نصف منتظمة (XML...)، و غير منتظمة (نص...)، من اجل تحقيق هذا الهدف، نركز على طريقة الوسيط الذي يرمي إلى انجاز نموذج مشترك لمصادر البيّنات على هيأة مخطط XML، والذي يسمح بتحليل مصادر البيانات دون الانشغال بمواضعها الفيزيائية و المنطقية.

SOMMAIRE

Liste des figures:

Listes des tableaux:

Introduction générale

1 Contexte :

La diversité des sources de données distribuées et leur hétérogénéité sont des principales difficultés rencontrées à l'exploitation des sources de données à la société NAFTAL. Cette hétérogénéité peut provenir du format ou de la structure des sources (sources structurées : bases de données relationnelles, sources semi-structurées : documents XML, ou non structurées : textes), du mode d'accès et de requête ou de l'hétérogénéité sémantique.

Ces problèmes nous ont conduit à réaliser un système pour accéder à des sources variées et en temps réel.

L'aide apportée par les systèmes de médiation peut recouvrir différentes formes : découvrir les sources pertinentes étant donnée une requête posée, puis aider à accéder à ces sources pertinentes, évitant à l'utilisateur d'interroger lui-même chacune d'elles selon leurs propres modalités et leur propre vocabulaire, enfin combiner automatiquement les réponses partielles obtenues de plusieurs sources de façon à délivrer une réponse globale. De tels systèmes de médiation offrent à l'utilisateur une vue uniforme et centralisée des données distribuées et hétérogènes, cette vue pouvant aussi correspondre à une vision plus abstraite, condensée, qualitative des données et donc, plus signifiante pour l'utilisateur. Ces systèmes de médiation sont, par ailleurs, très utiles, en présence de données hétérogènes, car ils donnent l'impression d'utiliser un système homogène.

2 Problématique et objectif :

L'exploitation des sources de données pose d'énormes difficultés, car ces sources de données ont des structures et des formats de stockage différentes (l'hétérogénéité), Cette hétérogénéité peut provenir du format ou de la structure des sources (sources structurées : bases de données relationnelles, sources semi-structurées : documents XML, ou non structurées : textes), du mode d'accès et de requête ou de l'hétérogénéité sémantique : entre les schémas conceptuels.

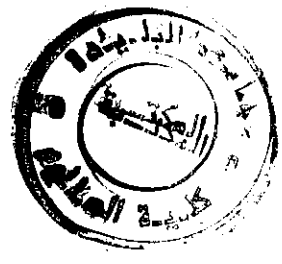
NAFTAL Spa se trouve confrontée à ces problèmes, elle assure une gestion d'une grande masse de données liées aux différents systèmes opérationnels. Ces systèmes assurent le traitement et le stockage des données. Ils sont éparpillés dans l'entreprise et ne constituent pas une source efficace pour le reporting et analyse.

L'objectif de notre travail est de construire un système homogène de données (Un système d'information est homogène si le logiciel qui gère les données est le même sur tous les sites. Les données ont le même format et la même structure),

Ceci donne la possibilité à l'utilisateur de consulter et d'analyser les différentes sources de données sans se préoccuper de leurs emplacements physique ou logique, ce système permet :

- D'interroger les sources de données.
- Uniformiser l'accès aux sources de données.
- D'accéder directement et simultanément à plusieurs sources de données hétérogènes.

L'uniformisation des sources de données est basée sur l'approche médiateur qui consiste à construire un modèle commun des sources de données sous forme de schéma XML. L'extraction des données est automatisée et permet au fur et a mesure d'intégrer des nouvelles sources de données sous une forme unique.



3 Plan de mémoire:

Ce mémoire est structuré de la façon suivante:

Présentation d'organisme d'accueil : présentation de la société NAFTAL, son historique et l'organigramme.

Chapitre I: Les systèmes d'information fédérés:

Dans ce chapitre nous présentons les différents systèmes d'informations, systèmes d'informations fédérés et les techniques et approches d'intégration.

Chapitre II: Modèles communs et sources de données:

Dans ce chapitre nous présentons les différents types des sources de données, les models commun les plus connues et un apport sur le model XML.

Chapitre III: Les technologies distribuées:

Dans ce chapitre nous présentons les technologies distribuées suivantes: CORBA, DCOM, RMI, .NET, SOAP.

Chapitre IV: La conception:

Ce chapitre comporte l'architecture générale de notre système et son fonctionnement, les règles de passage d'un schéma local à un schéma XML et la modélisation avec UML.

Chapitre V: L'implémentation et réalisation:

Ce chapitre présente l'environnement logiciel et matériel de développement, l'implémentation de différent module de système et l'enchaînement de système.

Nous terminons par une conclusion mettent le point sur le travail réalisé et ses perspectives.

L'organisme d'accueil

1 Présentation d'organisme d'accueil (NAFTAL):

NAFTAL, Société par actions filiale à 100% de SONATRACH, a pour mission principale la distribution et la commercialisation des produits pétroliers sur le marché national. NAFTAL a mis en place une nouvelle stratégie de développement dont l'objectif est de conserver sa position de leader de la distribution sur le plan national et de se déployer sur le plan international. Dans ce contexte, NAFTAL a mis en place une nouvelle organisation, composée de six divisions opérationnelles ayant une totale indépendance de gestion, en l'occurrence la division Carburant, la division commerciale, la division lubrifiants et pneumatiques, la division GPL, la division Aviation/Marine et enfin la division bitumes. La société contribue à hauteur de 51% de l'énergie finale algérienne en fournissant plus de 9 millions de tonnes tous produits confondus en l'occurrence, les carburants, les GPL, les bitumes, les lubrifiants, les produits aviation/marine et les produits spéciaux. NAFTAL a, en outre, lancé un large programme de formation à l'endroit de ses employés et ses cadres, une modernisation de ses installations et de ses infrastructures ainsi qu'une mise a niveau de tous ses systèmes de gestion. NAFTAL a inscrit dans sa stratégie de développement une option sérieuse pour l'international et elle est déjà présente sur le marché international puisqu'elle réalise du remplissage à partir de Tébessa vers la Tunisie, elle est également présente en Mauritanie. NAFTAL est en discussion avec des groupes privés Marocain, Egyptien, Libyen et Yéménite pour des alliances stratégiques à travers la création de sociétés, des mixtes joint-venture ou des échanges d'actifs. Elle est aussi en discussion avec plusieurs sociétés de pays africains, européens et latino américains.

2 Historique:

Issue de SONATRACH, l'entreprise ERDP a été créée par le décret N° 80/101 du 06 avril 1981. Entrée en activité le 1er janvier 1982, elle est chargée de l'industrie du raffinage et de la distribution des produits pétroliers sous le sigle NAFTAL. En 1987, l'activité raffinage est séparée de l'activité distribution. La raison sociale de la société change suite à cette séparation des activités et NAFTAL est désormais chargée de la commercialisation et de la distribution des produits pétroliers et dérivés. A partir de 1998, elle change de statut et devient Société par actions filiale à 100% de SONATRACH. Le siège de la Direction Générale : Route des dunes Chéraga BP 73, Alger.

3 Organisation:

Suite à son intégration dans le groupe Sonatrach dont elle est filiale à 100%, NAFTAL s'est réorganisée autour de quatre (04) Branches.

Les divisions ont pour mission de définir avec la direction générale, la stratégie de distribution et de commercialisation des produits pétroliers en veillant à rassembler toutes les conditions de son application dans les centres opérationnels de la société.
La figure suivant représente l'organigramme général de la société NAFTAL.

Légende:

Champ d'étude.

Champ externe.

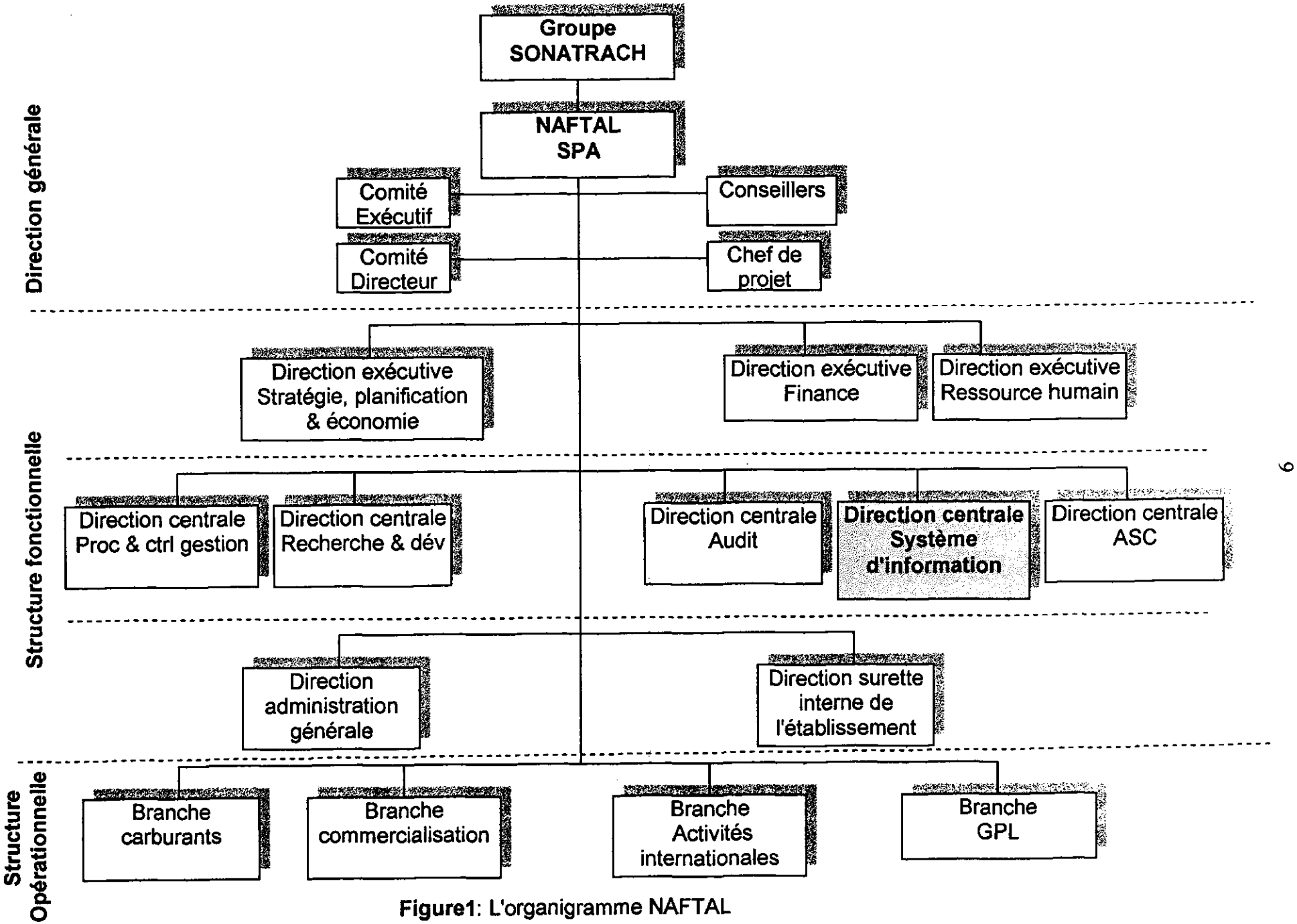


Figure1: L'organigramme NAFTAL

Chapitre I

les systèmes d'informations fédérés

Introduction :

L'évolution constante en matière des réseaux et de bases de données a engendré une demande croissante d'accès rapide à une large quantité d'informations.

Certaines de ces informations sont enregistrées dans les bases de données et sont accessibles par des langages de requêtes, d'autre sont simplement stockées dans des systèmes de fichiers ou de simple tableurs, d'autres encore sont les résultats d'applications plus ou moins complexes. Ces informations qui sont stockées sur des machines distantes sont appelées *sources de données.*

La diversité de ces sources de données conduit à des modes de consultation qui peuvent être très différentes. Ainsi, une base de donnée relationnelle sera interrogée par l'intermédiaire d'une requête SQL, une page Web sera consultée par une adresse web (URL) particulière, et un tableur par une formule spécifique. Une telle variété de sources implique diverses façons de les interroger (c-a-d de formuler une *requête*) mais aussi plusieurs manières pour la source de présenter un *résultat*. Il est donc nécessaire de concevoir des solutions (infrastructure, protocoles d'accès, architectures) permettant de construire des systèmes de traitement de données sur de grandes quantités de données multiformes stockées dans des bases interconnectées pour des applications telles que la génomique, l'ingénierie des langues ou des recherches, les application scientifiques, le traitement de données financières, la recherche documentaire etc.

Les systèmes d'informations fédérés offrent une solution intéressante pour l'interrogation de plusieurs sources de données hétérogènes, distribuées et autonomes, celui offre une vue intégrée (transparente) sur des sources de données, ces dernières étant variées et interconnectées.

Remarque : dans la suite le terme de système d'information n'est pas utilisé au sens large mais plutôt celui d'une sources de données.

1 Systèmes d'information :

Nous pouvons classer un système d'information selon trois aspects **[Buss et Al 1999]** : la distribution et l'hétérogénéité des données, l'autonomie des sources et, entre autres, l'interopérabilité des systèmes qui gèrent les sources de données.

1.1 Types de systèmes d'information :

Cette classification est établie selon la distribution et l'hétérogénéité **[Buss et Al 1999]**:

1.1.1 Un système unique (monolithique, centralisé) s'exécute comme une application monolithique sur une seule machine. Les données gérées par un tel système sont, soit stockées dans un SGBD, soit dans des fichiers du système des fichiers de la machine hôte.

1.1.2 Un système distribué : Dans les systèmes d'information distribués, les données sont reparties dans des sources différentes, elles-mêmes également reparties dans des machines géographiquement distribuées. La distribution est traitée par un certain nombre de techniques, telles que le CORBA, DCOM, SOAP et INTERNET.

En utilisant une de ces techniques, on peut développer des applications qui peuvent ignorer l'endroit physique des composants et l'hétérogénéité syntaxique.

1.1.3 Un système hétérogène : qui est un système distribué ou les systèmes sont différents selon des aspects logiques ou syntaxiques (plate-forme, système d'exploitation, modèle de données, etc....).Différentes classifications de l'hétérogénéité ont été proposées dans la littérature **[Buss et Al 1999],** nous distinguons l'hétérogénéité technologique, l'hétérogénéité des modèles des données, l'hétérogénéité structurelle et l'hétérogénéité sémantique :

- **L'hétérogénéité technologique :**

Cette l'hétérogénéité est due à la différence dans l'aspect technique des sources de données, par exemple les différences entre plates formes matérielles, et les systèmes logiciels (comme le système d'exploitation), les méthodes d'accès aux données (ODBC, SQL, OQL, XQuery...) et les protocoles de communications (http, CORBA, SOAP...).

- **L'hétérogénéité des modèles des données :**

Un modèle des données correspond à deux aspects:

- La description des données.
- Le langage de requête.

La diversité des structures de données et des opérations entre les différentes sources de données mène vers ce type d'hétérogénéité. La grande majorité des chercheurs supposant que les schémas initiaux (locaux) sont tous exprimés dans un même modèle de données, appelé modèle commun, sur lequel le logiciel d'intégration

est construit, les traductions requises sont celles entre les modèles de données locaux et le modèle commun.

Exemple : le modèle relationnel n'a pas le concept d'héritage en le composant avec le model orienté objet.

- **L'hétérogénéité schématique :**

Concerne la représentation des concepts par différents éléments de modèles.
On a dans le model relationnel trois types de conflits ont été recensés **[Buss et Al 1999]**:

- Relation ⇔ nom de l'attribut : elle a lieu quand une relation (table) a le même nom que celui d'un attribut d'une autre relation (dans une autre base de données).

- Relation ⇔ valeur de l'attribut : elle a lieu quand une relation (table) a le même nom que celui de la valeur d'un attribut dans une autre relation (dans une autre base de données).

- Nom de l'attribut ⇔ valeur de l'attribut :

L'exemple suivant illustre le conflit nom attribut/valeur de l'attribut.

Nom prof	THL	BD	IA
Jean Y	X		
Denis T	X	X	
Estelle S			X

Tableau 1

Nom prof	Cours
Jean Y	THL
Denis T	THL
Denis T	BD
Estelle S	IA

Tableau 2

Figure 1 : *Conflit « nom attribut ⇔ valeur de l'attribut »*

Alors que le Tableau1 représente les cours enseigné par un professeur comme des noms attribut, et Tableau2 représente les cours comme des valeurs de l'attribut « cours »

- **L'hétérogénéité sémantique :**

Concerne la sémantique associée aux données et aux modèles, cette hétérogénéité exprime le fait que le nom symbolique d'un concept peut être interprété de façon différente suivant les systèmes. Par analogie, l'interprétation de nom par différentes personnes peut ne pas coïncider (représentations mentales différentes). Ces conflits sémantiques se produisent lorsque :

- un même nom symbolique recouvre différents concepts (on parle d'homonymie).

- plusieurs noms symboliques recouvrent le même concept (synonyme). Par exemple, nous considérons un attribut nommé *article* de la relation *journal* dans une base de données BD1 qui décrit le numéro d'un article dans un journal, et le même attribut d'une relation *produits* dans une base de données BD2 qui décrit le numéro d'un article d'un produit donnée. L'attribut *BD1.journal.article* et *BD2.produit.article* ont les significations différentes, d'où l'hétérogénéité sémantique (homonymie).

♦ **L'hétérogénéité structurelle :**

Elle a lieu lorsque des éléments identiques ayant le même sens (contenu sémantique identique), modélisés avec le même modèle de données sont schématiquement homogènes mais structurés de différentes façons (par Exemple une adresse définie comme une chaîne de caractères dans une base de données locale et comme un n-uplet *<N°, rue, ville, code postal>* dans une autre).

2 systèmes d'information fédérés :

Un système d'information fédéré (SIF) est compose d'un ensemble des systèmes d'informations hétérogénies. distribues et autonomes.

2.1 Définition de l'autonomie :

Quand un système d'information est composé de plusieurs sites (appelé composants ou composants système), l'autonomie indique la capacité de la source de s'exécuter en dehors de la fédération. Elle dépend de plusieurs facteurs tels que l'échange d'informations entre les sources ou la possibilité de modification. Les quatre dimensions de l'autonomie sont **[Buss et Al 1999]** :

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

- **Communication** : les composants locaux décident quand et comment répandre eux requêtes.

- **Exécution** : les composants locaux n'ont pas besoin d'informer d'autres composants de l'ordre d'exécution de la transaction locale ou des opérations externes. Un système local ne distingue pas l'opération locale et globale.

- **Association** : les composants locaux décident quand se connecter et se déconnecter du système global. Ils décident également quelle fonction et donnée partagée, Avec quels types d'utilisateurs. Des informations statistiques (coût,

De plus, *l'adaptateur* met à la disposition de la *couche fédération*, des méta-données décrivant la source.

Les principaux objectifs de système fédéré sont :

- supprimer la nécessité de faire migrer des données d'un système à un autre ;
- respecter l'autonomie des composants de la fédération ;
- masquer l'hétérogénéité des environnements locaux de chaque source de données (ex. machine, réseau) ;
- supporter les transactions contentant des opérations de lecture et de mise à jour sur l'ensemble des systèmes de fédération ;
- offrir des performances qui approchent celle des systèmes d'information distribuées homogènes.

2.3 Méta-données pour les systèmes d'information fédérés :

Les méta-données sont des données permettant de décrire d'autres données ou des composants du système. Ce concept a pour but de résoudre certains problèmes liés à l'hétérogénéité et à la distribution des données. Sachant que les méta-données sont utilisées particulièrement dans la couche fédération d'un SIF. **[Buss et Al 1999]** Propose la classification suivante :

- **Les méta-données techniques :** décrivent les mécanismes techniques d'accès des composants, tels que les protocoles, la vitesse de connexion, le coût des requêtes, etc. Elles sont utilisées pour résoudre l'hétérogénéité technique.
- **Les méta-données logiques :** qui concernent les schémas et leurs relations logiques.

Les méta-données logiques sont disponibles dans le dictionnaire des données dans les

SGBD classiques ou dans les diagrammes de classes pour les SGBDOO ;

- **Les méta modèles :** sont des modèles adaptés à la description des modèles de données utilisés dans les sources de données, ils permettent de résoudre l'interopérabilité de plusieurs schémas issus de modèles de données différents (hétérogénéité des modèles de données). Le méta-modèle connaît tout les concepts de modélisation et sait comment transformer des structures de données entre elles ;
- **Les méta-données sémantiques :** qui permettent de fournir une description

différentes approches de développement des SIFs. Nous les utiliserons plus tard pour caractériser avec précision les classes importantes des SIFs. **[Busse et al. 1999]** propose les critères suivants :

a) Types de composants :

Un système d'information fédéré fait la différence entre les types de composants qu'il peut intégrer, il peut accepter ou rejeter des composants structurés, semi structurés ou non structurés dans la fédération.

• Les sources structurées ont un schéma prédéfini, tout élément de données est défini sous ce schéma, donc le schéma impose un format pour tous les éléments de données de la source (exp. base de données relationnelle).

• Les sources de données semi-structurées ont une structure qui n'est pas prédéfinie sous forme d'un schéma strict. Cependant, chaque élément de données doit porter sa propre définition sémantique sous forme de label (exp. sources XML, image.).

• Les sources de données non structurées n'ont aucune structure (tel que le texte, le langage naturel.),

b) Transparence :

Nous considérons que la transparence est le but principal à offrir à l'utilisateur. Une bonne intégration d'un SIF donne l'impression aux utilisateurs du système qu'ils travaillent sur un système d'information centralisé, homogène et consistant. Nous distinguons les types suivants de transparence :

- **Transparence d'emplacement** : l'utilisateur n'a pas besoin de connaître l'emplacement physique de l'information que ça soit la localisation du serveur (par adresse IP par exemple) ou le nom de la source de donnée (le nom de la BD si un SGBD est utilisé)

- **Transparence du schéma** : l'utilisateur n'a pas besoin de connaître les différentes notations des entités et des attributs appartenant aux différentes sources de données c-a-d tous les conflits sémantiques et schématiques sont masqués. La transparence du schéma peut être réalisée seulement si un schéma fédérateur existe.

- **Transparence du langage** : l'utilisateur n'a pas besoin de faire face aux différents langages et mécanismes de requêtes, les requêtes sont exécutées soit par

un langage de requête tel que SQL soit par des méthodes d'application via RPC.

c) l'approche ascendante et l'approche descendante :

Il existe deux manières fondamentales pour organiser un SIF : l'approche ascendante et l'approche descendante **[Meylan 2003]** :

Stratégie descendante : le processus de développement descendant est utilisé lors de la décomposition d'une base de données en plusieurs bases locales (sites), les schémas locaux sont créés à partir du schéma global.

Stratégie ascendante : le processus de développement ascendant est utilisé pour faire coopérer des bases de données existantes, le schéma global est obtenu à partir des schémas locaux. C'est le cas le plus courant mais le plus compliqué, car il nécessite de gérer l'existant qui est généralement hétérogène, d'ou la mise en place d'un système fédéré est généralement issue de celte approche.

e) Intégration virtuelle et intégration matérialisée :

Il y a des systèmes orientés vers l'interrogation et des systèmes qui permettent la modification des données. Nous proposons une classification des premiers en prenant comme critère l'intégration physique des données **[Solar and Doucet 2001]** :

- Système virtuellement intégré (non matérialisé) : Les données restent dans leurs sources et sont récupérées à la demande lors des interrogations. Selon la stratégie utilisée pour intégrer les données, les systèmes

• Faiblement couples intègrent des données structurées, non structurées ou semi structurées ;

• Fortement couples intègrent, en général, des données structurées (p. ex. fédérations de bases de données) ;

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

- **Système matérialisé :** Les données provenant de sources différentes sont intégrées sur une seule base. Selon le type de données intégrées les systèmes sont des SGBD universels qui intègrent des données structurées ou des entrepôts de données qui intègrent des données structurées et dérivées.

Il est à noter que dans le contexte de la recherche d'information (besoin de lecture seule), les systèmes matérialisés ne sont pas considérés.

2.5 Classification des systèmes d'information fédérés :

En se basant sur les critères cités ci-dessus, nous pouvons classer les SIFs selon trois grandes classes **[Busse et al 1999]** : les Systèmes d'Information Faiblement Couples. Les Bases de Données Fédérés et les Systèmes d'Information basés sur le Médiateur.

2.5.1 Systèmes d'information faiblement couplés :

Les systèmes d'information faiblement couplés n'offrent pas un schéma fédéré (global), mais seulement un langage de requête uniforme pour accéder aux composants de la fédération. L'utilisation d'un langage de requête uniforme permet de masquer l'hétérogénéité technique et l'hétérogénéité des langages. De plus, les composants n'ont pas besoin de renoncer à leur autonomie pour participer à ce type de fédération.

Cependant, ce type de système n'offre pas une transparence par rapport à la localisation et à l'accès au schéma des composants, si l'utilisateur veut obtenir une information d'un composant de la fédération ou d'un élément particulier faisant partie du schéma du composant, il doit le spécifier explicitement dans sa requête, et c'est a l'utilisateur (ou bien aux services de la couche présentation de la fédération) d'intégrer les données et de résoudre tous les conflits logiques qui peuvent survenir.

Dans la littérature les systèmes faiblement couplés sont parfois appelés des systèmes *multi-bases de données* **[Busse et al 1999]**.

2.5.2 Systèmes d'information fortement couplés :

Un système d'information fortement couplé offre un schéma fédérateur mais pas un langage de requête. Ce schéma fédérateur garantie à l'utilisateur de la fédération une transparence d'emplacement et de schéma. **[Busse et al 1999]** considère que les bases de données fédérés et les systèmes basées sur le médiateur sont des systèmes fortement couples. Dans ce qui suit, nous allons voir en détail ces deux approches.

2.5.2.1 Bases de données fédérées :

Les Systèmes de Bases de Données Fédérés (SBDF) sont des Systèmes d'Information (SI) fortement couplés, ils sont construits suivant l'approche ascendante

en appliquant quelques techniques d'intégration de schémas **[Busse et al 1999]**. De nombreux prototypes et systèmes de base de données fédérés proposés sont basés sur l'architecture de référence à cinq niveaux

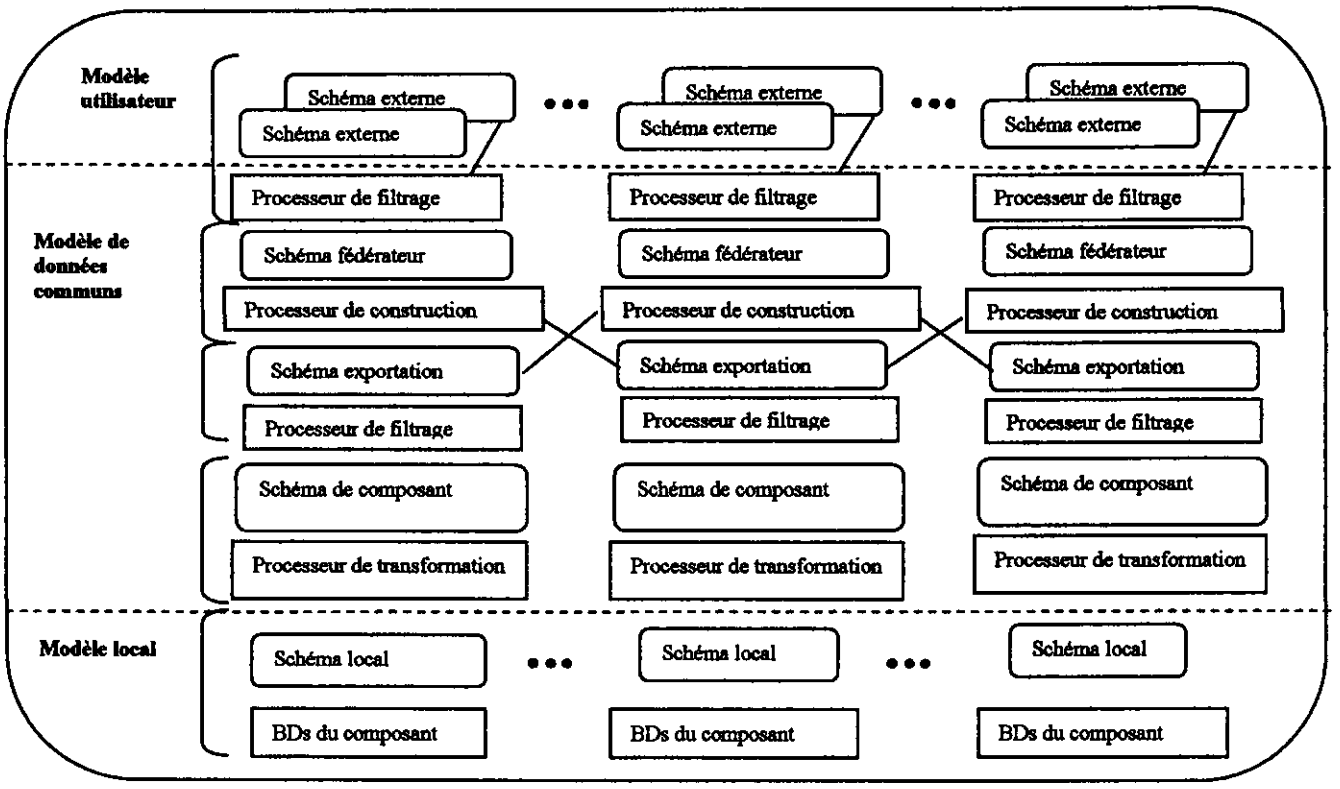


***Figure 4** : Architecture à 5 niveaux d'un système de bases de données fédérées*

Premièrement, un modèle canonique pour l'ensemble de la fédération doit être adopté. Les schémas locaux des composants sont alors transformés vers le modèle commun de données par le processeur de transformation, fournissant le schéma du composant. Ce schéma est ensuite filtré par le processeur de filtrage vers un ou plusieurs schémas d'exportation. A partir de ces derniers, un schéma fédéré est construit N processus est réalisé par le processeur de construction et s'appelle **l'intégration de schéma**; plusieurs schémas fédérés peuvent exister pour une même fédération. Enfin, à partir d'un schéma fédéré, plusieurs schémas externes sont déduits par le processeur de filtrage, pour différents utilisateurs (ou catégories d'utilisateurs) du système de base de données fédéré. Nous avons donc :

• **Le schéma local :** il s'agit du schéma conceptuel d'un composant base de données exprimé dans le même modèle de données interne au composant. Ainsi, plusieurs schémas locaux peuvent être exprimés dans différents modèles de

données.

• **Le schéma de composant :** est le résultat de la transformation d'un schéma local vers un modèle commun de données qui a précédemment été choisi ; cette transformation est effectuée par le processeur de transformation.

• **Le schéma d'exportation :** représente le sous-ensemble d'un schéma de composant rendu disponible à la fédération et aux utilisateurs de cette dernière. Il faut noter que toutes les données d'un composant ne sont pas forcément disponibles, le processeur de filtrage est utilisé pour fournir les contrôles d'accès et les opérations permises.

• **Le schéma fédéré :** il s'agit de l'intégration de plusieurs schémas d'exportation ; il inclut les informations concernant la distribution des données; ces informations sont générées lors de l'intégration des schémas d'exportation. Il est possible d'avoir plusieurs .schémas fédérés à l'intérieur d'un même SBDF.

• **Le schéma externe :** il définit un schéma, pour un utilisateur ou pour une application ou pour une classe d'utilisateurs et d'applications. Un processeur de filtrage est utilisé pour analyser les commandes sur un schéma externe et pour garantir leur conformité avec le contrôle des accès et les contraires d'intégrité du schéma fédéré.

Lorsqu'une fédération se forme ou lorsqu'un SGBD est sur le point de participer à une fédération existante, un processus de négociation est mis en place. Chaque SGBD négocie quelles données parmi celles qu'il possède doivent être rendues accessible (exportées) au reste de la fédération (à quelles catégories d'utilisateurs ou utilisateur de la fédération) et quelles parties de ces données peuvent être lues ou/et mises à jour et par qui. Une fois que la fédération est constituée, une requête utilisant un schéma externe (correspondant à son schéma fédéré N par le processeur de filtrage) sera décomposée en sous-requêtes vers les composants de BD concernés (sollicitant le processeur de construction), qui seront transformées vers leur schéma local et soumises à leurs SGBD correspondants.

La conclusion est qu'il n'est pas possible de considérer l'interopérabilité de différents SGBD sans étudier la similarité et l'équivalence des données concernées. Le problème de l'équivalence sémantique est crucial. Actuellement, la solution consiste à définir un modèle canonique approprié ; cette définition est un processus

nécessaire mais laborieux qui ne peut être fait de manière automatique.

2.5.2.2 Systèmes d'information basée sur le médiateur :

Les Systèmes d'Information Basés sur le Médiateur SIBM (les systèmes de médiation) sont des systèmes d'information fortement couplés, ainsi un schéma fédérateur est utilisé pour fournir l'accès intégré aux données des différents composants. Dans le titre suivant on va expliquer cette approche en détail.

3 Techniques et approches d'intégration :

Les solutions à l'intégration d'informations proposées tireront parti des recherches déjà effectuées dans le domaine. Nous présentons ci-dessous les deux approches d'intégration les plus connues : les approches médiateurs et les approches entrepôts de données.

3.1 L'approche médiateur

3.1.1 Présentation générale

L'approche médiateur « middleware » consiste à définir une interface entre l'agent (humain ou logiciel) qui pose une requête et l'ensemble des sources accessibles. L'objectif est de donner l'impression d'interroger un système centralisé et homogène alors que les sources interrogées sont réparties, autonomes et hétérogènes. La figure 5 est un exemple qui explique clairement comment interroger les requêtes et obtenir les réponses **[Agora 2000]**.

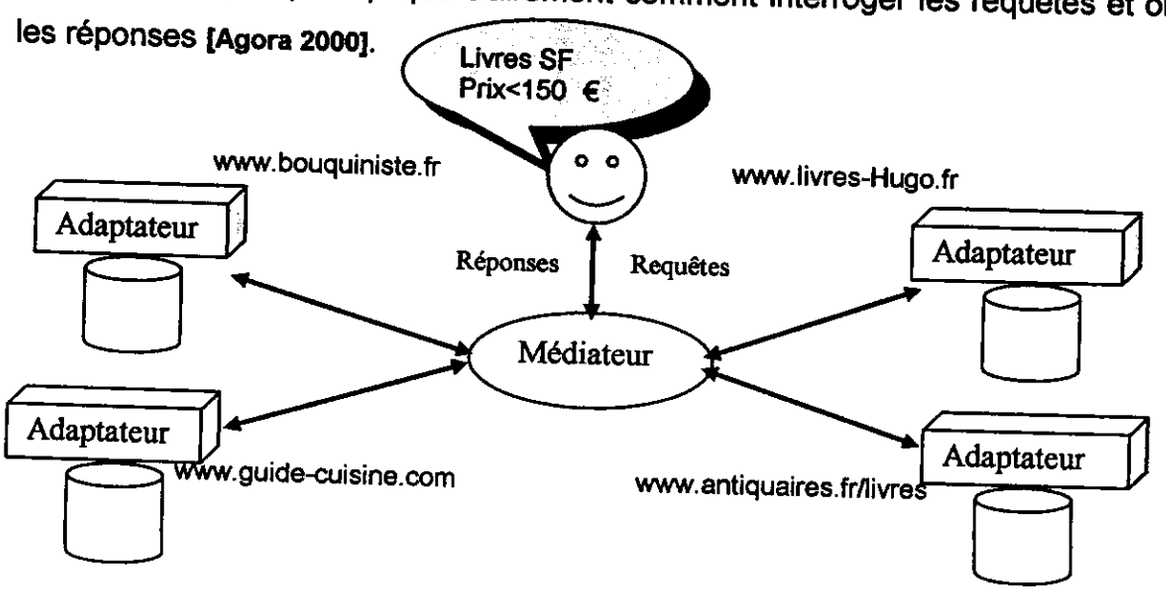


Figure 05 : *Exemple de schéma d'intégration basé sur le médiateur.*

Un médiateur (figure 06) comprend un schéma global **[MS et CR 2003]**, ou ontologie, dont le rôle est central. C'est un modèle du domaine d'application du système. L'ontologie fournit un vocabulaire structuré servant de support à l'expression des requêtes. Par ailleurs, elle établit une connexion entre les différentes sources accessibles. En effet, dans cette approche, l'intégration d'information est fondée sur l'exploitation de vues abstraites décrivant de façon homogène et uniforme le contenu des sources d'information dans les termes de l'ontologie. Les sources d'information pertinentes, pour répondre à une requête, sont calculées par réécriture de la requête en termes de ces vues. Le problème consiste à trouver une requête qui, selon le choix de conception du médiateur, est équivalente ou implique logiquement, la requête de l'utilisateur mais n'utilise que des vues. Les réponses à la requête posée sont ensuite obtenues en évaluant les réécritures de cette requête sur les extensions des vues. L'approche médiateur présente l'intérêt de pouvoir construire un système d'interrogation de sources de données sans toucher aux données qui restent stockées dans leurs sources d'origine. Ainsi, le médiateur ne peut pas évaluer directement les requêtes qui lui sont posées car il ne contient pas de données, ces dernières étant stockées de façon distribuée dans des sources indépendantes. L'interrogation effective des sources se fait via des adaptateurs, appelés des wrappers en anglais, qui traduisent les requêtes réécrites en termes de vues dans le langage de requêtes spécifique accepté par chaque source.

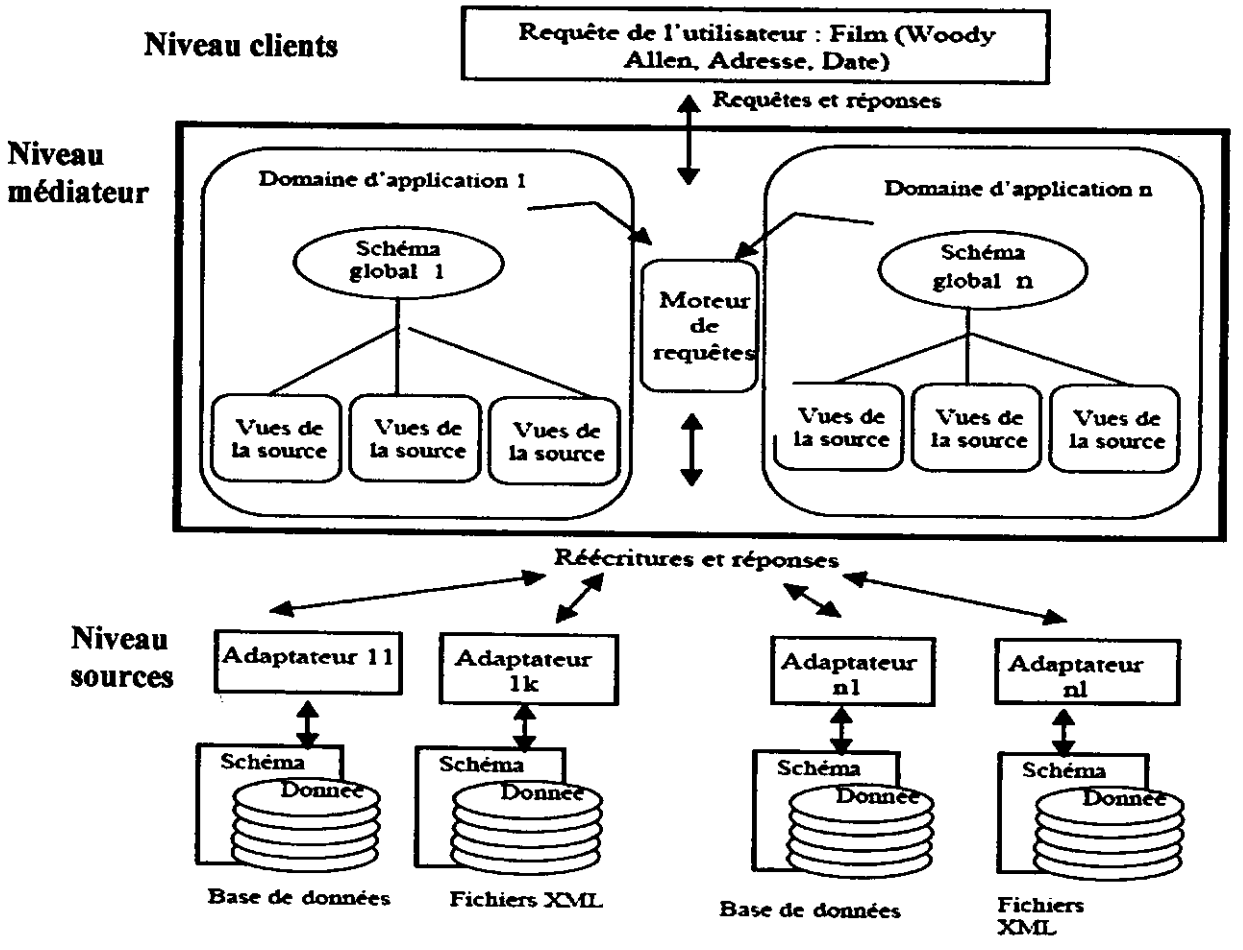


Figure06 : *Architecture d'un système médiateur* **[MS et CR 2003].**

- **Le médiateur :** est une sorte d'administrateur des bases, résout les conflits entre les différents modèles de données et les différents schémas de bases de données, traduit le langage uniforme en sous-langages adaptés aux bases de données.

- **L'adaptateur :** est un module qui transforme les sous-questions dans le langage de la base à laquelle il est rattaché, reformule la réponse pour le médiateur. Ainsi, l'utilisateur d'un tel système soumettra une requête au médiateur, qui se chargera de traduire la requête en un ensemble de sous-requêtes exécutables sur les sources locales. La réponse à la requête est construite à partir des réponses fournies par les différentes sources.

Cette architecture est composée de trois niveaux comme suit :

- **Le niveau source:** comporte les différentes source de données, à l'aide d'un adaptateur, qui est capable de communiquer avec les médiateurs du niveau supérieur en leur fournissant une vue homogène de la source à laquelle il est associé. Un

adaptateur accepte une requête donnée dans le langage commun du médiateur, la transcrit dans le langage natif de la source (p.ex. SQL dans le cas d'une base de données relationnelles, XQuery dans une base XML) et exécute la requête. Le résultat de la requête transmise sous forme native (des tuples dans le cas relationnel, du XML pour une base de données XML, des objets dans le cas d'une base de donnée objet) est alors transformé suivant le modèle de données global du médiateur et renvoyé à celui-ci.

- **Le niveau médiateur** : comporte des médiateurs permettant d'intégrer les données en provenance de différentes sources afin de répondre aux requêtes des utilisateurs. Ce module joue un rôle actif dans la couche entre les applications utilisateurs et les sources de données. Son rôle est de fournir à la couche supérieure une vue unifiée des sources qui sont hétérogènes et distribuées.

Nous trouvons aussi à ce niveau des facilitateurs (moteur de requêtes) permettant d'identifier les sources qui peuvent être des sources de données ou des médiateurs, et de composer les réponses pour les utilisateurs.

- **Le niveau client:** comporte les applications clientes (navigateurs, programme d'application, interface graphique). L'intégration d'une nouvelle source se résume à développer un adaptateur pour cette dernière et le rendre accessible par le médiateur. Un langage de requête commun ainsi qu'un modèle de données commun entre les facilitateurs, le médiateur et les adaptateurs doivent être définis pour toute 1'architecture. Le rôle d'un adaptateur consiste essentiellement à permettre la traduction du langage de requête commun dans langage natif de la source qu'il gère, et la traduction du modèle de données de la source en modèle de données commun.

La disparité des sources de données doit être prise en compte dans le développement de l'adaptateur spécifique, mais cela n'est pas toujours évident. En effet, certaines sources (par exemple un moteur de recherche, ou une page web) offrent des capacités limitées d'interrogation (par rapport à un SGBD relationnel par exemple). Le médiateur doit pouvoir en tenir compte. Ensuite, la répartition des sources sur un réseau sur l'échelle de l'Internet entraîne l'accroissement de situations d'indisponibilités des sources dues à des problèmes de communication ou de serveurs inaccessibles.

3.1.2 Panorama des médiateurs existants :

Les différents systèmes d'intégration d'informations à base de médiateurs se distinguent par : d'une part, la façon dont est établie la correspondance entre le schéma global et les schémas des sources de données à intégrer, d'autre part les langages utilisés pour modéliser le schéma global, les schémas des sources de données à intégrer et les requêtes des utilisateurs.

Concernant le premier point, on distingue l'approche Global As Views (GAV) de l'approche Local As Views (LAV). L'approche GAV, qui provient du monde des bases de données fédérées, consiste à définir le schéma global en fonction des schémas des sources de données à intégrer.

Les systèmes suivant cette approche sont **[MS et CR 2003]:** HERMES, TSIMMIS, MOMIS. L'approche LAV est l'approche duale. Elle est adoptée dans les systèmes suivants : Razor, Internet Softbot, Infomaster, Information Manifold, OBSERVER, ICSEL. Les avantages et inconvénients de ces deux approches sont inverses. Selon l'approche LAV, il est très facile d'ajouter une source d'information, cela n'a aucun effet sur le schéma global. En revanche, la construction des réponses à des requêtes est complexe, contrairement à la construction de réponses dans un système adoptant une approche GAV qui consiste simplement à remplacer les prédicats du schéma global de la requête par leur définition.

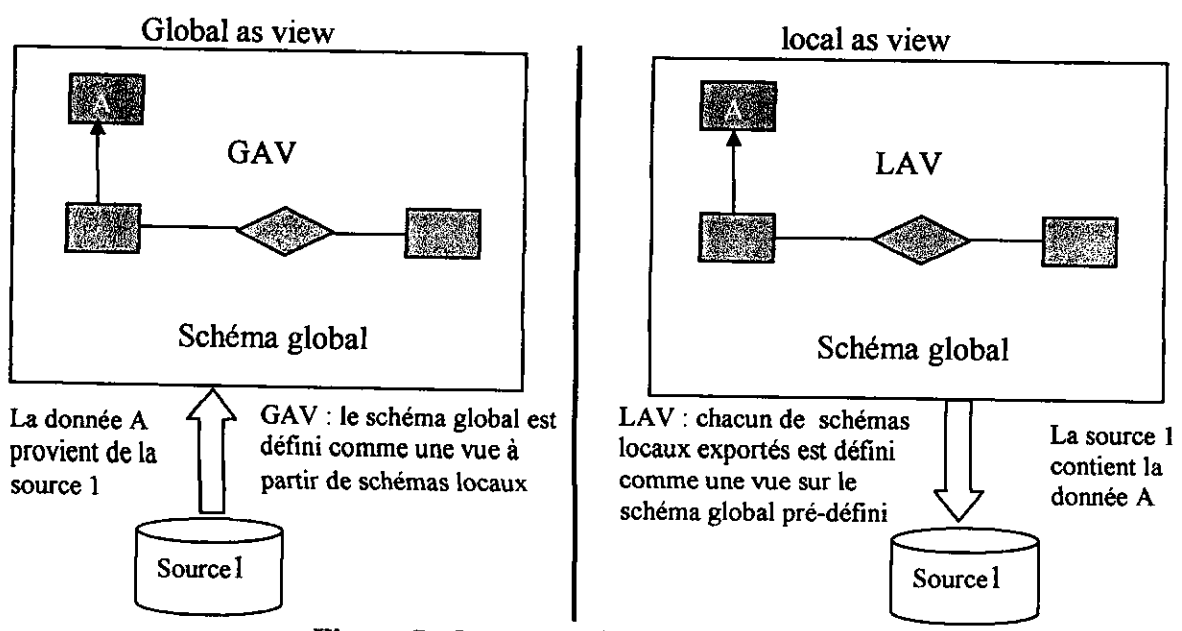


***Figure 7** : Les approche GAV et LAV*

Les systèmes existants se différencient également par le langage qu'ils utilisent pour exprimer le schéma global. On distingue les systèmes fondés sur un schéma global à base de règles **[MS et CR 2003]** (Razor, Internet Softbot, Infomaster, Information Manifold, HERMES), des systèmes fondés sur un schéma à base de classes (langage orienté objet (TSIMMIS)), logique de description (SIM S,OBSERVER, MOMIS), ou encore des systèmes combinant le pouvoir d'expression d'un formalisme à base de règles et d'un formalisme à base de classes (PICSEL). Enfin, plus récemment, sont Réécritures et réponses Requête de l'utilisateur : Film (Woody Allen, Adresse, Date) apparus des médiateurs au dessus de données semi-structurées ayant le format de documents XML (C-Web, Xyleme). Ces systèmes sont fondés sur un schéma global à base d'arbres. Ils relèvent à la fois de l'approche GAV et LAV, la correspondance entre le vocabulaire du médiateur et celui des sources étant exprimé par de simples mappings de chemins

3.1.3 Processus d'intégration :

- **pré intégration :** (traduction des schémas sources dans un modèle des données commun), le but de cette phase est d'extraire un maximum d'information sur les éléments (attribut, classe, méthode,...) composent un schéma, ainsi que les relations et les règles existant entre ces éléments.
- **Comparaison :** recherche les correspondances inter schémas,lorsque les schémas initiaux ont atteint le niveau de conformité souhaité, l'étape suivante consiste a identifier les éléments de base existant , rappelons que deux bases de données ont un concept commun si le portion de monde réel qu'elle représentent ont des éléments commun soit une intersection non vide , ou ont des éléments en relation entre eux par un lien sémantique pour des applications futures , parmi les travaux dédiés à cette phase de comparaison citons les approches suivantes :
  - calcul d'une fonction de similitude entre paires d'éléments
  - calcul d'une pertinence sémantique entre éléments a laide de la logique floue
  - méthode manuelle : les paires de noms sont proposés à l'expert de la base de données qui doit déterminer lui-même les correspondances à retenir.
- **fusion (intégration) :** création d'un schéma global à partir des schéma sources, les correspondances inter-schémas.

Une fois effectuer le travail de description sémantique des schémas (pré-intégration) et de recherche de correspondances entre leurs éléments (comparaison),

le processus d'intégration proprement dit peut être effectué, il consiste de manière semi-automatique en générale à construire un schéma intégré à partir :

- des schémas sources (éventuellement traduits dans un modèle canonique)
- des correspondances inter-schémas (déterminé lors de la comparaison)
- des règles d'intégrations (propre à la méthode et aux modèles de données utilisés).

- **restructuration :** modification des schémas sources, cette phase se trouve plutôt dans les cas d'intégration de vues ou lorsque la modification ultérieure des schéma sources est admise, les approches récentes, de type fédérés, n'admettent pas de telles modification, ce ci afin de préserver l'environnement local (données, applications) des bases composent la fédération et leur garantir leur autonomie.

3.2 L'approche entrepôt de données :

Un Data Warehouse répond aux problèmes de données surabondantes et localisées sur de multiples systèmes hétérogènes, c'est une architecture capable de servir de fondation aux applications décisionnelles. Pour être exploitables, toutes les données provenant des systèmes distribués doivent être organisées, coordonnées, *intégrées* et enfin stockées pour donner à l'utilisateur une vue globale des informations **[Agora 2000]**.

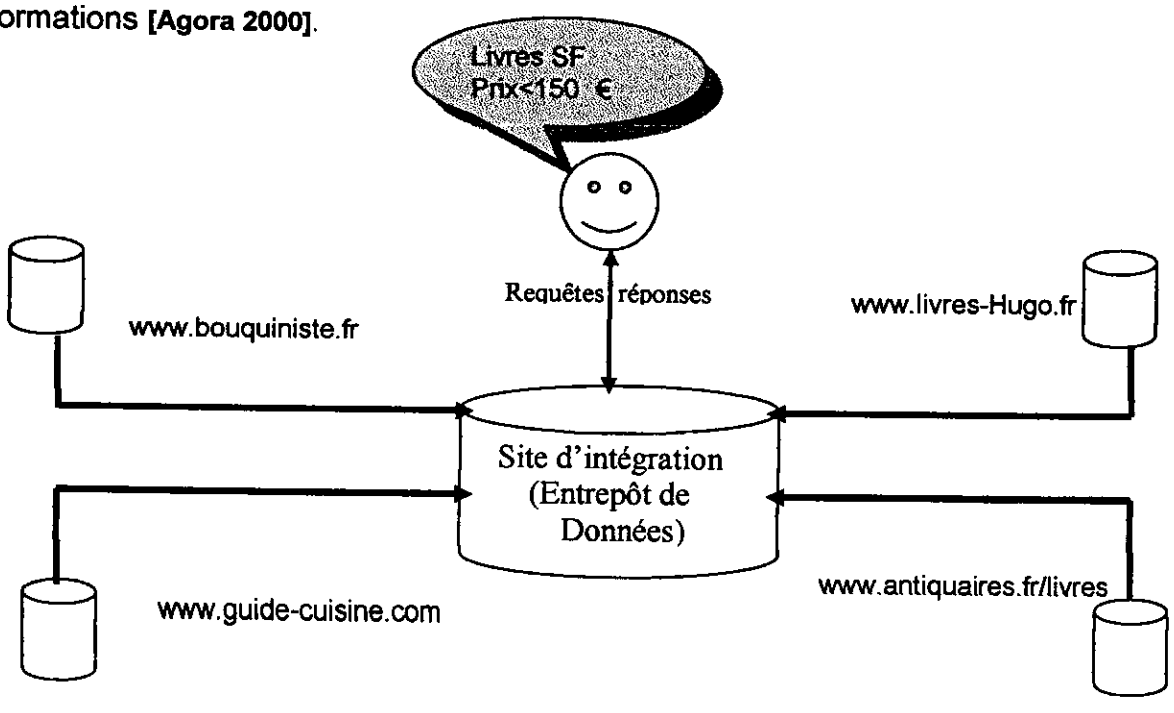


Figure 08 : *Exemple de schéma d'intégration basé sue l'entrepôt de données.*

opérationnelle typique de ces composants est donnée par la figure 3. Les composants logiciels sont représentés par des rectangles. Les ellipses désignent des stockages intermédiaires des résultats de l'étape d'extraction/transformation. Toutes les données qui sont en entrée du composant intégration utilisent le même modèle de représentation de données. Finalement, un «wrapper » est associé à chaque source fournissant ainsi une interface API à la source.

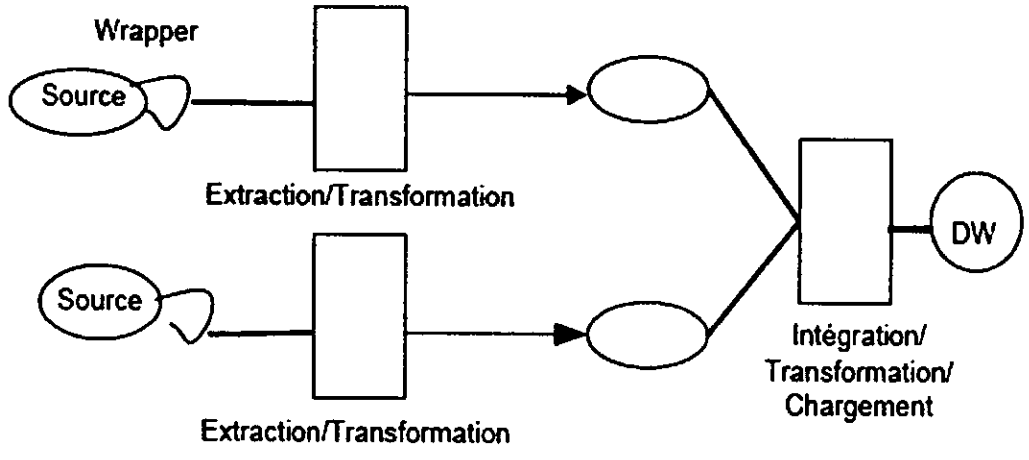


Figure10 : *Vue opérationnelle des composants utilisés pour la construction d'entrepôts de données* **[MS et CR 2003].**

Au second niveau, le processus de construction comporte trois étapes distinctes, qui sont :

1- l'extraction de données à partir d'une base de données (entrepôt de données local ou global)

2- le calcul des données dérivées pour l'entrepôt de données local cible,

3- le stockage des résultats dans l'entrepôt de données local. L'étape d'extraction est un cas particulier de celle du premier niveau car les données de l'entrepôt sont stockées dans une base de données. A l'opposé, dans le premier niveau, l'extraction peut concerner des sources de données arbitraires, comme des fichiers par exemple. Le calcul des données dérivées est assez spécifique car il peut impliquer des requêtes complexes avec agrégats.

3.2.2 Les types d'intégration :

Le type d'intégration réalisé dans la conception d'un entrepôt de données est celui que l'on réalise dans le domaine de l'intégration d'information, qui a été exploré dans différents domaines comme :

- Les bases de données,

• Les systèmes d'information coopératifs,

• Les systèmes d'information globaux,

• La représentation des connaissances.

Une première classification des différentes approches repose sur le contexte d'intégration, et par conséquent, le type des entrées/sorties du processus d'intégration, et le but du processus lui-même. Nous distinguons l'intégration de schémas, l'intégration de données virtuelle, et l'intégration de données matérialisée.

• Intégration de schémas : Dans ce cas, l'entrée de l'intégration est un ensemble de schémas sources, et la sortie est un schéma de données correspondant à la représentation intensionnelle

réconciliée de tous les schémas en entrée. L'entrée comporte également la spécification de la façon d'associer les schémas des données sources à des parties du schéma résultant (cible).

• Intégration de données virtuelle (médiateurs) : L'entrée est un ensemble de données sources, et la sortie est une spécification décrivant la façon de fournir un accès global et unifié aux sources dans le but de satisfaire certains besoins en information, sans interférer avec l'autonomie des sources.

• Intégration de données matérialisée : Comme dans le cas précédent, l'entrée est un ensemble de données sources, mais ici la sortie est un ensemble de données représentant une vue réconciliée des sources, à la fois au niveau intensionnel et au niveau extensionnel.

3.3 Comparaison entre l'approche médiateur et entrepôt de données :

	médiateur	**entrepôt de données**
Avantages	– évite la réplication des données – répartition de la charge – sécurité des données	-disponibilité -comportement prédictible – possibilité de mise en forme des données
Inconvénient	– définir le schéma intégré – relier schémas locaux et schéma intégré – exécution distribuée	– besoins de stockage – difficultés légales – données inaccessibles, confidentielles – évolution (rafraîchissement) – montée en charge (traitement centralisé)

***Table1**: Comparaison des techniques d'intégrations*

4 Synthèse :

Les déférentes approches présentées dans ce chapitre a savoir : «les systèmes d'information faiblement couplés », « les bases de données fédérés » et « les systèmes d'information basés sur le médiateur», ont des soucis de prendre en compte les différents types d'hétérogénéité. Les systèmes faiblement couplés intègrent des base de données qui présentent une hétérogénéité technologie, contrairement aux autres approches ou l'on peut intégrer des composants qui différent dans tous les aspects (modèle de données, sémantique). Concernant l'autonomie des composants de la fédération, les trois approches citées dans le tableau ci-dessous ne peuvent pas garder leurs autonomies d'exécution, c'est-à-dire que les sources locales doivent différencier entre les transactions locales et globales (ordre d'exécution et priorité).

Selon l'approche adoptée et le domaine d'application, les sources de données à intégrer peuvent ; être de nature différentes. Les systèmes basés sur le médiateur peuvent faire face a n'importe quel type de données (structurées, semi-structurées et non structurées), contrairement aux systèmes faiblement couplés et aux bases de données fédérées qui ne supportent que des sources de données structurées (des bases de données relationnelles et bases de données objet) , Les systèmes faiblement couplés et les bases de données fédérées supportent des traductions qui contiennent des opérations de lecture et des mise a jour sur l'ensemble des base de fédération ce qui est un avantage, mais cela pose quelques problèmes tels la gestion de la concurrence d'accès aux données, en plus de la complexité de la mise ouvre de tels systèmes. en effet, l'accès aux données est fait en lecture seulement dans des systèmes basés sur le médiateur, ce qui privilégie cette approche dans le développement d'applications de recherche d'information.

	Système d'information faiblement couplé	*Bases de données fédérés*	*Système d'information Basé sue le médiateur*
Type d'hétérogénéité concernée	Hétérogénéité technologique	Toutes	Toutes
Perte d'autonomie ?	Autonomie d'exécution	Autonomie d'exécution	Autonomie d'exécution
Type de transparence	Langage	Emplacement, schéma, et partiellement du langage	Emplacement, schéma, Et du langage
Type de composants	Structuré	Structuré	N'importe
Méthode d'accès	Langage de requête	Langage de requête	N'importe
Méta-données nécessaires	Tcchniquc, infrastructure	Logique, technique, sémantique	Logique, technique, sémantique
Virtuelle ou matérialisée	Virtuelle	Virtuelle	Virtuelle
Ascendant ou descendant	-	Ascendant	Descendant

Table 2: Comparaison entre les types des SIF.

Conclusion :

Nous avons vu quelques critères sur les quels des approches de SIF peuvent être classifiées. En effet, nous avons essayé de différencier clairement entre les concepts base de données fédérée, systèmes basés sur le médiateur et systèmes faiblement ou fortement couplés.

L'objectif principal de ce chapitre était de mettre en évidence que les systèmes d'information fédérés sont la solution appropriée pour notre problème, à savoir l'intégration de sources des données. L'approche que nous avons finalement trouvée adéquate pour l'interrogation de sources d'information distribuées, hétérogènes et autonomes est l'approche médiateur et grâce de cette technique on peut accéder directement et simultanément a plusieurs sources de données hétérogènes et de transformer ces données en informations d'entreprise accessibles en temps réel, et cela se justifie par le fait qu'un système d'information basé sur le médiateur est un système qui offre un mécanisme d'accès homogène, virtuel et en lecture seule à un ensemble variable de sources d'information hétérogènes, autonomes et distribués. Les SIBM sont des systèmes en général légers adressant toutes sortes d'hétérogénéité. Les composants accèdes ne sont pas forcement des systèmes de gestion de base de données et n'offrent pas souvent des langages d'accès, tel est le cas dans l'intégration de données. Les principaux composants d'un SIBM sont des

adaptateurs, qui encapsulent des sources et enlèvent l'hétérogénéité technique et celle du modèle de données, et des médiateurs, qui résolvent l'hétérogénéité logique.

Les technologies à objets distribués peuvent apporter des solutions à la réalisation du système, tels que CORBA, DCOM, etc. qui ont fait leurs preuves dans l'industrie. Ainsi que la norme XML pourra être une solution inévitable pour le choix du modèle de données commun.

Chapitre II

Modèle commun et sources de données

Introduction:

Nous présentons dans cette partie les différentes sources de données à intégrer, les modèles communs les plus connues et à la fin une conclusion.

On distingue 03 types de sources de données :

- Structurées : les base de données.
- Semi-structurées : XML.
- Non structurées : texte...

1 Source de donnée structurée :

1.1 Les bases de données [CCM1 2004]:

Une base de données (son abréviation est BD, en anglais DB, database) est une entité dans laquelle il est possible de stocker des données de façon structurée et avec le moins de redondance possible. Ces données doivent pouvoir être utilisées par des programmes, et par des utilisateurs différents. Ainsi, la notion de base de données est généralement couplée à celle de réseau, afin de pouvoir mettre en commun ces informations, d'où le nom de base. On parle généralement de système d'information pour désigner toute la structure regroupant les moyens mis en place pour pouvoir partager des données.

Une base de données peut être locale, c'est-à-dire utilisable sur une machine par un utilisateur, ou bien répartie, c'est-à-dire que les informations sont stockées sur des machines distantes et accessibles par réseau.

L'avantage majeur de l'utilisation de bases de données est la possibilité de pouvoir être accédées par plusieurs utilisateurs simultanément.

Afin de pouvoir contrôler les données ainsi que les utilisateurs, le besoin d'un système de gestion s'est vite fait ressentir. La gestion de la base de données se fait grâce à un système appelé SGBD (Système de Gestion de Bases de Données) ou en anglais DBMS (Database management system). Le SGBD est un ensemble de services (applications logicielles) permettant de gérer les bases de données, c'est-à-dire:

- permettre l'accès aux données de façon simple ;

- autoriser un accès aux informations à de multiples utilisateurs ;
- manipuler les données présentes dans la base de données (insertion, suppression, modification).

Les principaux systèmes de gestion de bases de données relationnelles sur le marché sont les suivants : Oracle, Microsoft SQL Server, Microsoft Access, Sybase, DB2 et MySQL.

L'interrogation est une opération permettant d'extraire des données ou une chaîne de données d'une base de données à l'aide d'un langage spécial.

Cependant la diversité de ces sources de données conduit à des modes de consultation qui peuvent être très différents. Ainsi, une base de données relationnelle sera interrogée par l'intermédiaire d'une requête SQL, une page Web sera consultée par une adresse web (URL) particulière, et un tableur par une formule spécifique. Une telle variété de sources implique diverses façons de les interroger c'est-à-dire de formuler une requête mais aussi plusieurs manières pour la source de présenter résultat.

1.1.1 SQL (Structured Query Language, traduisons Langage de requêtes structuré) : est un langage de requête pour les bases de données relationnelles.

Le SQL est à la fois un langage de manipulation de données et un langage de définition de données. Toutefois, la définition de données est l'oeuvre de l'administrateur de la base de données, c'est pourquoi la plupart des personnes qui utilisent le langage SQL ne se servent que du langage de manipulation de données, permettant de sélectionner les données qui les intéressée.

2 source de donnée semi-structurée :

2.1 L'historique de XML :

XML (*eXtensible Markup Language*) a été mis au point par le XML Working Group sous l'égide du World Wide Web Consortium (W3C) dès 1996. Depuis le 10 février 1998, les spécifications *XML 1.0* ont été reconnues comme recommandations par le W3C, ce qui en fait un langage reconnu. [w3]. XML est un sous ensemble de SGML (*Standard Generalized Markup Language*), défini par le standard ISO8879 en

1986, utilisé dans le milieu de la Gestion Electronique Documentaire (GED). XML reprend la majeure partie des fonctionnalités de SGML, il s'agit donc d'une simplification de SGML afin de le rendre utilisable sur le web.

2.2 Présentation :

XML (eXtended Markup Language) **[DANG NGOC 2003]** est un format textuel extensible de description de document défini par le W3C. De la famille des langages de marquage SGML, il permet de s'adapter à quasiment tous les domaines ou l'on a besoin de structurer de l'information de façon portable.

XML est conçu de façon à faciliter l'intégration et l'échange de données entre applications. Il isole le formatage et le rendu des documents par rapport à sa structure.

XML est un langage à base d'éléments, d'étiquettes, d'attributs et de valeurs. Les balises (tags) ouvrantes (resp. fermantes) sont constituées d'étiquettes (label) représentés entre le symbole < (resp. </) et le symbole >. Le composant logique compris entre une balise ouvrante et une balise fermante est appelé valeur. Le composant logique constitué de la balise ouvrante, de la valeur et de La balise fermante est appelé élément (élément). La valeur peut être vide, contenir du texte, d'autres éléments ou contenir un mélange des deux (mixed élément content). Les balises définissent la structure du document. L'élément de plus haut niveau englobant tous les autre et n'ayant pas de parent est appelé élément racine. Un élément peut contenir des informations additionnelles appelées attributs (attributes). Un attribut est un couple forme d'un nom et d'une valeur et est représenté à l'intérieur de la balise ouvrante sous la forme *nom* = *"valeur"*. Un document XML est un ensemble d'éléments ainsi imbriqués.

2.3 Document XML :

L'exemple suivant décrit les comportements d'une facture, qui sont le client (nom, prénom, adresse et N°téléphone), ligne de commande (nom de produit, quantité et prix).

```
<?xml version='1.0' encoding='ISO8859-1' ?>
<facture>
    <Client>
        <nom> rahim </nom>
        <prenom> hassen </prenom>
        <adresse> 13 rue amor mihoub alger </adresse>
        <Ntelephone> 021251456 </Ntelephone>
  </Client>
  <lignedeCommande>
    <nomProduit> machine (5.08 cm) </nomProduit>
    <quantite> 17 </quantite>
    <prix> 1250.00 </prix>
  </lignedeCommande>
</facture>
```

***Figure1** : Exemple de document XML*

2.3.1 Composants de document XML :

Tout document XML se compose :

- D'un prologue, dont la présence est facultative mais conseillée. Il contiendra un certain nombre de déclaration ;
- D'un arbre d'éléments. Il forme le contenu proprement dit du document ;
- De commentaires et d'instructions de traitement, dont la présence est facultative, et qui pourront, moyennant certaines restriction, apparaître aussi bien dans le prologue que dans l'arbre d'éléments.

2.3.1.1 Le prologue

Le prologue peut contenir une déclaration XML, des instructions de traitement et une déclaration de type de document.

a) Déclaration XML

Cette déclaration a la forme suivante :

<?xml version="1.0" encoding='ISO-8859-1' standalone='yes' ?>

Elle indique au processeur qui va traiter le document :

• La version du langage XML utilisée dans le document, ici la version 1.0.

• Le codage de caractères utilisé dans le document, si la valeur par défaut ne convient pas.

XML utilise les jeux de caractères définis par la norme ISO 10646, les processeurs XML devant être capables de traiter au moins les codages UTF-8 et UTF-16. Les codages de caractères sont présentés à la section 6.2. L'indication de codage dans le prologue d'un document est facultative. Nous verrons à la section 6.3 dans quels cas elle est réellement utile.

• L'existence de déclarations extérieures au document qui doivent être prises en compte pour le traitement de celui-ci. Si l'attribut standalone a la valeur « yes », le processeur considère que toutes les déclarations nécessaires au traitement du document y sont incluses. Si cet attribut a la valeur « no », le processeur doit rechercher des déclarations dans d'autres fichiers que celui qui contient cette déclaration XML pour pouvoir traiter convenablement le document.

b) Instructions de traitement

Les instructions de traitement sont facultatives. Elles ne font pas, à strictement parler, partie du document. Leur contenu est simplement transmis à une application en vue de déclencher certains traitements. Elles peuvent aussi apparaître dans le corps du document. Leur syntaxe est la suivante :

< ? instruction de traitement ?>

c) Déclaration de type de document

Celle-ci indique dans le cas où le document se conforme à une structure type particulière quel est ce type. Elle permet aussi de spécifier certaines déclarations propres au document.

2.3.1.2 L'arbre d'éléments

L'exemple précédant peut représenté sous forme d'un arbre comme ci-dessus :

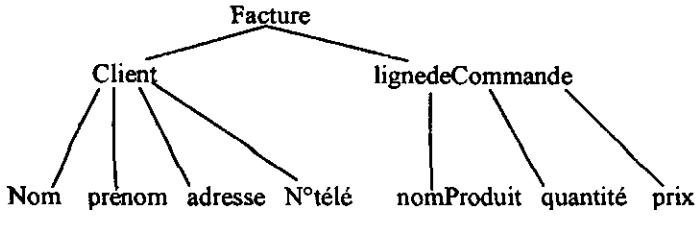


***Figure2** : L'arbre d'élément*

2.3.1.3 Commentaires

Des commentaires peuvent être inclus dans un document. Ils sont encadrés par les marques de début et de fin de commentaire :

<!– Ceci est un commentaire –>

2.4 Document Type Définition DTD

Une DTD (*Document Type Définition*)[w3Ca] permet aux utilisateurs de définir leurs propres balisages, attributs et entités pour des documents de types SGMl. ou XML.
Un document DTD est signalé par une déclaration de type de document (Document Type Déclaration) par 1'intermédiaire de la chaîne **<!DOCTYPE**. Il peut être défini explicitement ou en référence vers une entité externe. Une DTD définit une structure de document par l'intermédiaire d'un ensemble de déclarations de balisage.
Une déclaration de balisage peut être une déclaration de type d'élément, une déclaration de liste d'attributs, une déclaration d'entité ou une déclaration de notation.

2.4.1 Déclaration de type d'élément : Elle se signale par la chaîne **<!ELEMENT**. On déclare ensuit le nom d'élément suivi des spécifications de son contenu. Si l'élément est de type chaîne de caractères, on utilisera la chaîne **#PCDATA**. Dans le cas ou l'élément comporte de sous éléments, ceux-ci sont spécifiés. Il est ensuite possible de définir le nombre d'occurrences des sous-éléments avec les symbole : « + », le sous-éléments est represenié une fois au moins ; « * », le sous-éléments est représenté zéro ou plusieurs fois , « ? », le sous-éléments est optionnel ; sans symbole, le sous-éléments est représenté exactement une fois.

2.4.2 Déclaration de type d'attributs : on introduit une déclaration d'attributs par la chaîne **<!ATTLIST** et le nom de l'élément concerné. Vient ensuit l'ensemble des attributs de cet élément, chacun suivi d'une définition de type et d'une critère existentiel (**#REQUIRED** si l'attribut est obligatoire, **#FIXED** si celui-ci est fixé, et **#IMPLIED** si l'attribut n'a pas de valeur par défaut et peut être déclaré ou non).

2.4.3 Déclaration d'entité : la chaîne **<!ENTITY** permet de définir une constante. La valeur de l'entité peut ensuite être appelée en précédant le nom de l'entité par le symbole « & ».

Dans l'exemple suivant, le type de document de *bibliothèque*, se défini comme suit :

La racine de l'arbre est l'élément *bibliothèque* qui compte un ou plusieurs *livres*. Un *livre* est constitué d'au moins une date, d'exactement un *titre* et de un on plusieurs auteurs. Un livre comporte également les attributs obligatoires *référence* et *catégorie* ainsi que l'attribut optionnel *pages*. Les éléments *date*, *titre*, *nom* et *prénom* sont de type simples de type chaîne de données. Un auteur a un *nom* et un *prénom* obligatoire et une *adresse* optionnelle. L'élément *auteur* a aussi un attribut optionnel *pays*. Enfin, une entité *nom_bibliotheque* est définie et sa valeur est "Bibliothèque Centrale ".

```
<!DOCTYPE bibliotheque [
<!ENTITY nom_bibliotheque "Bibliothèque Centrale">
<!ELEMENT bibliographie (livre+) >
<!ELEMENT livre (date+, titre, auteur +) >
<!ATTLIST livre refrence CDATA #REQUIRED
                Categorie CDATA #REQUIRED
                Page CADATA #IMPLIED>
<!ELEMENT date ( #PCDATA ) >
<!ELEMENT titre ( #PCDATA ) >
<!ELEMENT auteur (nom, prenom, adresse ?) >
<!ATTLIST auteur pays CDATA #IMPLIED >
<!ELEMENT nom ( #PCDATA ) >
<!ELEMENT prenom ( #PCDATA ) >
<!ELEMENT adresse ( #PCDATA ) >
]>
```

Figure3 : *Exemple de DTD*

Remarque : Donc après **[DANG NGOC 2003]**, un document XML peut être :

- bien formé : quand il respecte la syntaxe du langage XML défini par W3C ;
- validé : quand il est associé à une définition de type de document (DTD) et qu'il la respecte (nom des éléments, type, répétition et ordre d'apparition dans le document).

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

1- il existe un et un seul élément racine qui contient tous les autres éléments.

2- les balises sont correctement imbriquées : chaque balise ouvrante à une balise fermente associée et il n'y a pas de chevauchement.

3- le nom des balises est libre mais il contient au moins une lettre.

4- les attributs des balises, lorsqu'ils existent, doivent comporter obligatoirement une valeur qui doit toujours apparaître entre doubles apostrophes.

5- quand un élément est vide, les balises peuvent être simplifiées.

2.5 XML-Schema :

XML Schéma [ZERDAZI 2003] est un standard constituant l'ensemble des règles devant être respectées par tout document XML. Le point faible des DTD est sa faible capacité à spécifier la sémantique des données et sa syntaxe non XML (ce qui entraîné ici naissance de XML-Schema).

Les éléments et les attributs globaux sont créés par des déclarations qui apparaissent directement à l'intérieur de l'élément **schéma**. Une fois déclaré, un élément ou un attribut global peut être référencé dans une ou plusieurs déclarations en utilisant l'attribut **réf**. Les éléments sont déclarés en utilisant le mot-clef **élément** et les attributs sont déclarés en utilisant le mot-clef **attribute**. Des types simples, tels que *string. token, int, décimal, time, date, ID, IDREF,* etc. sont prédéfinis dans XML-Schema. De nombreux types simples peuvent être définis par dérivation de type simple déjà existant.

Les nouveaux types complexes sont créés en utilisant l'élément **ComplexeType** composé, dans la plupart des cas, d'une série de déclarations d'éléments et d'attributs et de références d'éléments. Ces déclarations sont rassemblées dans des groupes modèles permettant de définir, un modèle de contenu constitué de d'une séquence **sequence** (ensemble ordonné), d'un choix exclusif **choice** ou d'un ensemble non-ordonné **all**.

Le nombre minimum (resp. maximum) de fois qu'un élément peut apparaître est déterminé dans sa déclaration par la valeur de l'attribut **minOccurs** (resp. **maxOccurs**). Cette valeur peut être un entier positif comme par exemple 42, ou encore le mot **unbounded** qui signifie qu'il n'y a pas de valeur limite. La valeur par défaut dans les deux cas (**minOccurs** et **maxOccurs**) est 1.

Le document suivant est le document XML-Schema du document XML présenté dans la section1.

```
<?xml version='1.0' encoding='ISO8859-1' ?>
<schema xmlns=" http://www.w3.org/2001/XMLSchema">
<element name="facture" type= "factype"/>
<complexeType name="facture">
<sequence>
      <element name="client" type="clientype"/>
      <complexeType name="client">
      <sequence>
            <element name="nom" type=" string" minOccurs="1" maxOccurs="1"/>
            <element name="prenom" type=" string" minOccurs="1" maxOccurs="1"/>
            <element name="adresse" type=" string" minOccurs="1" maxOccurs="1"/>
      </sequence>
      </complexeType>
      <element name="lignecommande" type="lignecommandetype"/>
            <complexeType name=" lignecommande ">
            <sequence>
                  <element name="nomproduit" type=" string" minOccurs="1"
maxOccurs="1"/>
                  <element name="quantite" type=" integer" minOccurs="1"
maxOccurs="unbounded"/>
                  <element name="prix" type="float" minOccurs="1"
maxOccurs="unbounded"/>
            </sequence>
            </complexeType>
```

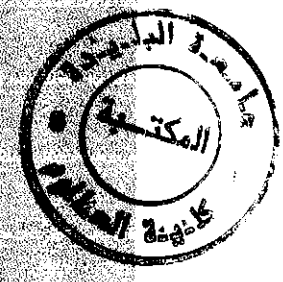


Figure 4 : Exemple de schéma XML (.xsd)

2.6 xquery :

XQuery ou XML Query (XML requête) est un langage fonctionnel d'interrogation de documents XML. C'est un langage qui manipule une collection de documents XML comme étant une base de données. Autrement dit, il prend comme données d'entrée et comme résultats des structures XML

3 Source de donnée non structurée :

3.1 Fichier plat [kevin w 2001]:

Les fichiers plats stockent des données d'une façon le plus souvent spécifique à l'application qui les utilise et on les rencontre fréquemment lors du travail avec des systèmes hérités. Les fichiers de configuration sont souvent des fichiers plats. Les systèmes hérités stockent leurs données de façon native selon un format de fichier plat ou possèdent déjà un mécanisme permettant d'importer ou d'exporter des données selon ce format. Un des avantages inhérents aux fichiers plat par comparaison avec un tableau de correspondance est qu'un développeur peut modifier le paramétrage ou d'une façon quelconque lire et traiter des données qui se

trouveraient autrement dans un format propriétaire. Le fait d'apprendre à récupérer des données se présentant ainsi et à les transformer en une forme plus utile pour votre nouveau système peut vous épargner de coûteuses mises à jour du code hérité.

3.2 Types de fichier plat :

Il existe trois grands types de fichiers plats classiquement utilisés pour des données. En apprenant à manipuler, créer et utiliser des fichiers sous ces trois formes, l'utilisateur devient être capable de traiter pratiquement n'importe quel fichier plat ou de créer tout type de fichier plat exigé par un système client.

3.2.1 Délimité

Dans un fichier *délimité*, les champs de données sont séparés, ou délimités, par un caractère prédéfini (un caractère supposé ne figurer dans aucun des champs de données). Les délimiteurs les plus fréquents sont les caractères virgule, deux-points, tabulation et barre verticale. Chaque enregistrement se termine classiquement par un saut de ligne consolidé (la paire saut de ligne-retour chariot). Si un saut consolidé est le caractère de fin utilisé, cela pose des problèmes lors de la conversion entre UNIX et DOS/Windows, par exemple. Vous trouverez ci- dessous un exemple de fichier délimité utilisant une virgule comme séparateur. Si un champ est vide, vous trouvez deux séparateurs adjacents, comme dans le second exemple.

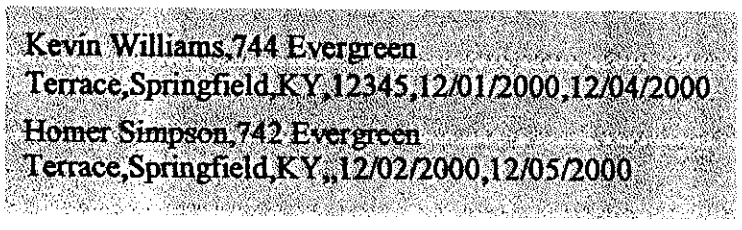


***Figure 5** : Fichier délimité.*

3.2.2 Largeur fixe

Dans un fichier à *largeur fixe*, chaque champ de données se voit allouer un nombre d'octets particulier. Il s'agit du type que vous êtes susceptible de rencontrer en travaillant avec des systèmes hérités COBOL. Il y a souvent (mais pas toujours) un saut de ligne consolidé à la fin de chaque enregistrement. Voici un exemple de fichier à largeur fixe. C'est un exemple qui manque un peu de naturel. En principe, la

longueur de chaque champ exige que la ligne excède la largeur de l'écran, mais, tant qu'il n'y a pas de saut de ligne consolidé, cela ne pose pas de problème.

```
Kevin Williams 744 Evergreen Terrace Springfield
KY12345
12/01/200012/04/2000
Homer Simpson 742 Evergreen Terrace Springfield
KY12345
12/02/200012/05/2000
```

Figure 6 : Fichier à *largeur fixe.*

3.2.3 Enregistrements balisés

Les fichiers à *enregistrements balisés* apportent une structure rudimentaire aux fichiers à plat. Des balises peuvent être utilisées pour des fichiers délimités ou à largeur fixe, servant à indiquer ce que représente le reste de l'enregistrement dans lequel elles apparaissent. Vous pourriez, par exemple, faire en sorte que le premier caractère d'un enregistrement indique s'il s'agit d'une facture (F), d'un client (C), ou d'une ligne de facture (L). Le reste de la signification de l'enregistrement dépend alors du type d'enregistrement. Voici un exemple de fichier à enregistrements balisés :

```
F,12/01/2000,12/04/2000
C,Kevin Williams,744 Evergreen
Terrace,Springfield,KY,12345
L,bleu,9 cm. Machins,0001700000.10
L,argent,14 cm. Trucs,0002200000.20
F,12/02/2000,12/05/2000
C,Homer Simpson,742 Evergreen
Terrace,Springfield,KY,12345
L,rouge,4 cm. Chose,0001300000.30
L,bleu,9 cm. Machins,0001100000.10
```

Figure 7 : Fichiers à *enregistrements balisés*

4 les modèles communs [X Baril 2003] :

La construction du schéma médiateur (approche GAV ou LAV) s'effectue en définissant des vues sur des sources de données hétérogènes. Certains problèmes lies à l'hétérogénéité du format des sources (bases de données relationnelles, objets, documents xml, . . .) sont résolus grâce à l'utilisation d'un modèle commun (également appelé modèle pivot ou global). Ce modèle doit permettre de représenter des données provenant de sources hétérogènes pour les intégrer d'une façon uniforme dans un schéma médiateur. Les premiers systèmes d'intégration de données ont utilisé les modèles relationnels ou objets comme modèles communs. Cependant, l'intégration de sources de données peu ou pas structurées est difficile avec ces modèles permettant de représenter des bases de données. Les modèles de données semi-structurées ont été conçus pour représenter facilement des données irrégulières provenant de sources hétérogènes, structurées ou non. Ce sont des modèles flexibles qui permettent de représenter des données irrégulières en mélangeant la structure et les données.

OEM (Object Exchange Model) [X Baril 2003] est le modèle de données semistructurées qui s'est impose, devenant un standard pour les systèmes d'intégration de données apparus avant la naissance de XML. Il avait été proposé à l'origine dans le projet TSIMMIS, comme modèle commun pour intégrer des données provenant de sources hétérogènes. Nous allons en faire une brève présentation dans cette section. Les données OEM sont représentées par un graphe, dont les noeuds sont des objets permettant de stocker les données et leur schéma Un objet OEM est structure de la manière suivante :

Oid	étiquette	Type	valeur

Figure 8 : Schéma d'un objet OEM

oid : un identifiant,

étiquette : une chaîne de caractères qui décrit ce que l'objet représente,

type : le type de la valeur de l'objet,

valeur : la valeur de l'objet.

Les objets OEM peuvent être simples ou complexes. Les objets simples ont une valeur atomique et sont de type *integer, real, string,* etc. Les objets complexes sont composes à partir d'autres objets et peuvent être de type *set* ou *list.*

Exemple:

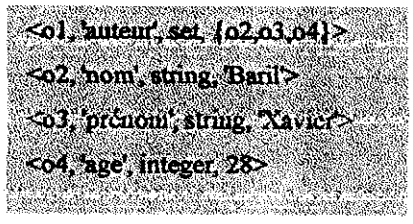


***Figure 9** : Les données OEM*

Les données OEM décrivent un auteur. L'objet o1 a pour valeur un ensemble contenant les trois autres objets o2, o3 et o4. Plus simplement, on peut dire que l'objet auteur est compose d'un nom, d'un prénom et d'un âge.

Bien que le modèle OEM contienne le terme "objet", un objet OEM ne possède pas de comportement. En effet, il n'est pas possible de définir de méthodes sur un objet OEM. Le seul concept objet présent dans OEM est la notion d'identifiant (oid) qui permet de définir un graphe d'objets.

4.1 Apport de XML pour l'intégration de données:

La plupart des systèmes d'intégration de données actuels utilisent XML comme modèle commun[X Baril2003] pour les raisons suivantes. Tout d'abord, XML permet de représenter des données provenant de sources hétérogènes, dont les modèles de données sont différents. Ensuite, XML est un format largement répandu et accepte par la communauté informatique. De nombreuses sources de données sont disponibles et de nombreuses applications permettent d'exporter leurs données en XML. Le langage XML permet de représenter des données structurées,semi structurées et non structurées, en intégrant la plupart des concepts des modelés connus. Il permet de composer des hiérarchies de données liées entre elles par des hyperliens. Les relations peuvent être vues comme des instances d'éléments avec des attributs. Les hyperliens correspondent aux associations des modèles entité/association. Avec les schémas qui sont apparus récemment, le typage des données élémentaire est possible.

Pour cela, de nombreux types sont proposes permettant de décrire de manière précise les données. De plus, il est possible de créer des types complexes avec les constructeurs séquence, choix et tas. Enfin, la structure est contenue dans les données sous la forme de balise et de nom d'attribut.

Aujourd'hui, de nombreux logiciels permettent d'exporter leurs données au format XML. En effet, comme c'est un format d'échange, de nombreux éditeurs de logiciel ont développé des modules qui permettent de transformer des données en XML. Parmi les plus connus, le SGBD Oracle propose des extensions pour exporter les données contenues dans des tables relationnelles en XML. Certains logiciels des suites bureautiques actuelles de Microsoft*TM* permettent aussi d'exporter des données XML.

Enfin, la standardisation de XML par le w3c et son succès dans l'industrie informatique en font un bon candidat comme langage commun pour un système d'intégration. De plus,[X Baril2003] souligne qu'un certain effet de mode tend à associer XML à l'intégration de données.

Conclusion :

Nous avons choisi le langage XML comme formalisme de représentation pour sa capacité à représenter tout type de données contrairement eux autres modèles. Notamment, le formalisme XSD est beaucoup plus riche que celui des DTD pour représenter la structure permise pour un document, et contrôler de façon plus stricte le contenu pouvant apparaître dans un document XML. Autrement dit, un schéma XSD ou une DTD constituent l'ensemble des règles devant être respectées par tout document XML. Le point faible des DTD est sa faible capacité à spécifier la sémantique des données (ce qui entraîné la naissance du langage XSD). Les DTD sont donc insuffisantes dans le cadre de ce projet car les règles de structure ne sont pas assez précises sémantiquement (pas de typage possible des données, pas de cardinalités précises d'occurrences d'un élément, etc.). Si nous pensons maintenant en terme d'arbre, les DTD ne permettent pas de définir des règles de construction d'arbre XML par héritage et ne sont pas homogènes au niveau du langage avec XML.

Chapitre III

Les technologies distribuées

Introduction :

L'une des qualités recherchées dans un système distribué est la transparence. Il s'agit de qualifier de manière impropre que ce dernier doit masquer à l'utilisateur ou aux programmes d'application la distribution. Il offre par conséquent l'illusion d'un système unique. Ce masquage concerné aussi bien l'emplacement des ressources, que la diversité des plates formes d'exécution,

Les sources de données distribuées sont des sources de données qui se trouves sur plusieurs machines reliées en réseau. Elle nécessite donc un protocole de communication pour les accéder.

Plusieurs technologies permettant de développer des applications distribuées, **DCOM** (Distributed Comportent Object Model) de Microsoft, **CORBA** (Common Object Request Broker) de l'OMG (Object Management Croup), **RMI** (Remote Method Invocation), la solution de Sun et enfin le protocole **SOAP** qui allié les technologie Internet avec XML et le protocole Web HTTP.

L'objectif des technologies distribuées est de décomposer une application complexe en un ensemble de processus coopérant les uns avec les autres, ces processus pouvant être situés sur des machines différentes.

1 La technologie CORBA :

Depuis 1989, une association internationale appelée l'OMG (Object Management Croup) définit la spécification de l'architecture d'un système à objets répartis, appelée **CORBA** (Common Object Request Broker Architecture).

L'OMG est une organisation mondiale qui regroupe Près de 850 acteurs du monde informatique, aussi bien des sociétés informatiques comme HP, Digital, Sun, Microsoft et IBM, que des utilisateurs comme Boeing ou Alcatel. Toutes ces organisations dont le domaine varie entre la recherche, le développement, les réseaux et même les bases de données, sont toutes intéressées par des technologies d'objets distribués telles que CORBA. CORBA est né donc du besoin de faire communiquer un ensemble des applications en environnement hétérogène (plusieurs systèmes et plusieurs langages), il associe les concepts d'objet et de répartition dans une approche à la fois intégrée et appliquée. L'intégration se fait à travers la notion d'invocation à distance d'objet (l'objet est l'unité de répartition), et l'application se fait à travers les services de répartition qui

sont organisés sous forme de bibliothèques de classes.

Bien que CORBA soit une technologie performante **[J.FRANC01S 2001]**, son utilisation dans notre projet ne parait pas très pratique, vu sa complexité de mise en oeuvre. L'enjeu dans notre travail est de mettre à la disposition des utilisateurs une vue transparente des sources de données situées dans des sites distribué et hétérogène, pour l'intégration de ces sources nous allons utiliser XML comme modèle de données commun. Le choix d'une autre technologie véhiculant le XML comme format de transport paraît plus judicieux pour notre travail. Par contre CORBA transporte ces paquets au format binaire puisqu'il est compilé, ce qui entraîne aussi des difficultés pour le franchissement des équipements de filtrage.

2 Les technologies COM/DCOM :

Bien que membre de l'OMG, Microsoft a développé son propre modèle à composants objets appelé **COM** (Comportent Object Model). **COM** est une architecture logicielle qui permet aux applications et aux systèmes d'être conçus par différents développeurs. Il offre certains mécanismes qui permettent aux composants de communiquer et de coopérer de manière transparente aux utilisateurs. Les principaux objectifs de COM sont :

- Définition d'un standard binaire pour l'interopérabilité entre les composants.
- Indépendance des langages de programmation.
- Permettre une évolution et une extensibilité du système et des applications

DCOM (Distributed Comportent Object Model) est un standard d'objets répartis. Le terme standard revêt dans ce contexte un caractère particulier, car il signifie standard pour, et seulement pour, les produits Microsoft.

COM offre des mécanismes permettant la communication entre les composants se trouvant dans le même site ou répartis sur le réseau. C'est une proposition de Microsoft pour le problème de conception de composants. Elle permet à plusieurs sociétés de développement de créer leurs propres composants tout en leur assurant que leurs composants vont communiquer avec les autres entités du système, pouvoir être mis à niveau sans créer de conflits, être indépendants du langage et être distribués.

COM permet la communication entre les composants, entre les clients et les serveurs via des interfaces. Les composants peuvent accéder à des interfaces se

trouvant sur le même site ou sur un site distant via le mécanisme DCOM. Ainsi, une application COM est constituée de:

• **Interface COM** : Le moyen par lequel un objet expose ses services aux clients. Un objet COM fournit une interface pour chaque ensemble de méthodes.

• **Serveur COM** : Un module qui contenant le code d'un objet COM. Les implémentations d'objets résident sur les serveurs. Un objet COM implémente une ou plusieurs interfaces.

• **Client COM** : Le code appelant les interfaces afin d'obtenir du serveur les services demandés. Les clients savent ce qu'ils veulent obtenir du serveur (via l'interface) ; les clients ne savent pas comment en interne le serveur fournit les services.

Le choix de DCOM pour réaliser notre architecture paraît un choix difficile puisque les sources à intégrer sont hétérogènes (des systèmes d'exploitation différents Windows, UNIX, Linux...).

DCOM est le modèle d'architecture distribuée incontournable sur plate-forme Windows 32 bits il apporte des facilités importantes au niveau de La sécurité et de la distribution des composants mais aussi des gains notables de performances Par contre cette technologie est difficile à mettre en oeuvre. Autre point faible cette architecture c'est qu'elle s'exécute sous plates-formes win32 uniquement, ce qu'il n'est pas avantageux dans notre projet puisque les bases de données sont multi plats formes.

3 la technologie Java RMI:

RMI (Remote Method Invocation) **[Ray et Varin 2001]**: est un mécanisme Java réservé aux objet Java. Plus précisément, RMI permet d'invocation d'objet résidant dans des machines virtuelles différentes. Il est donc tout a fait conforme au modèle objet Java. De plus, RMI a d'autres objectifs, et notamment celui de minimiser la complexité de l'implémentation des objets serveur et clients. Il ce doit aussi de préserver la sécurité déjà inhérente à l'environnement Java.

Un programme utilisant RMI se compose de deux parties. La première est un objet distant, le serveur, qui propose plusieurs de ses méthodes à l'invocation à distance. La seconde partie est un client, qui récupère une référence du serveur et

peut dés lors invoquer ses méthodes, comme s'il s'agissait d'un objet local.

Comme nous venons de le voir, RMI permet de développer d'une manière facile une application répartie. Une bonne transparence est assurée au programmeur. La manipulation de XML via Java est très pratique, puisque XML est le modèle de donnée commun dans notre projet d'intégration des données hétérogènes et distribué, en plus RMI est une technologie multi plates-formes. Donc, l'utilisation de cette architecture est bien adaptée pour notre système. La faiblesse de RMI est qu'il peut être programmé en utilisant que le Java, il n'est pas multi langage.

4 SOAP et service web:

SOAP (Simple Object Access Protocol) est un protocole d'invocation de méthodes sur des services distants. Basé sur XML, SOAP assure la communication de machine à machine. Le protocole permet d'appeler une méthode RPC (remote procedure call) et d'envoyer des messages aux machines distantes via HTTP.

SOAP permet donc l'échange d'information dans un environnement décentralisé et distribué, comme l'Internet par exemple. Il permet l'invocation de méthodes, de services, de composants et d'objets sur des serveurs distants et peut fonctionner sur de nombreux protocoles (des systèmes de messagerie à l'utilisation de RPC).cependant, il fonctionne particulièrement bien avec le protocole http.

SOAP se sert le plus souvent du protocole HTTP. Un message SOAP est écrit en XML. HTTP est utilisé comme protocole de transport. Les messages SOAP vont donc être encapsulés dans HTTP, ce qui permet une utilisation et une compatibilité très importante avec les réseaux et équipements existants. HTTP est le protocole de transport le plus utilisé mais il n'est pas impossible de trouver des implémentations de SOAP sur d'autres protocoles (avec SMTP par exemple).

Un message SOAP est un document XML qui possède une enveloppe SOAP et éventuellement une déclaration XML. L'enveloppe SOAP est composée d'un corps SOAP et éventuellement d'un en-tête

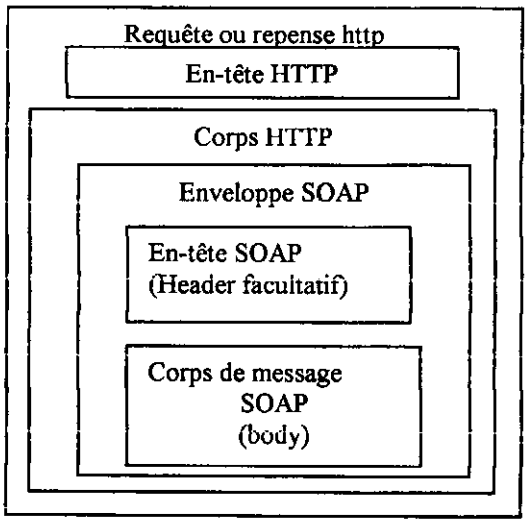


Figure1: *Structure d'un message SOAP encapsulé dans une requête http*

SOAP est muitiplates-formes, multilangages. Dans notre travail où les sources à intégrer sont distribués et hétérogènes et gardant un certain degré d'autonomie, l'utilisation de SOAP paraît plus adéquate, puisque SOAP ne nécessite pas l'installation d'un ORB[1] ce qui le cas avec CORBA et DCOM.

5 La technologie .NET :

5.1 L'environnement d'exécution .NET

L'environnement .NET **[Bienvenu 2003]** peut être défini de la façon suivante : Il s'agit d'un ensemble de services communs, utilisables depuis plusieurs langages. .NET définit une architecture proche de CORBA. Plus précisément, il utilise le langage intermédiaire MSIL[2] (MicroSoft Intermediate Language) pour uniformiser la représentation des applications et le protocole de communication SOAP pour permettre l'invocation de méthodes distantes à travers le réseau Internet. Toutes les applications écrites dans le langage intermédiaire ne sont pas disponibles pour une méthode distante.

[1] ORB (Object Request Broker) est le noyau de transport des requêtes aux objets dans la technologie CORBA.

[2] MSIL: c'est un langage intermédiaire constitué d'instructions élémentaires de type "assembleur évolué". Tous les langages supportés par la plate forme .NET peuvent être compilés en code intermédiaire MSIL. Ce code intermédiaire peut ensuite être exécuté par le CLR, l'environnement de l'exécution de la plate forme .NET. SMIL permet l'indépendance au langage de la plate forme .NET.

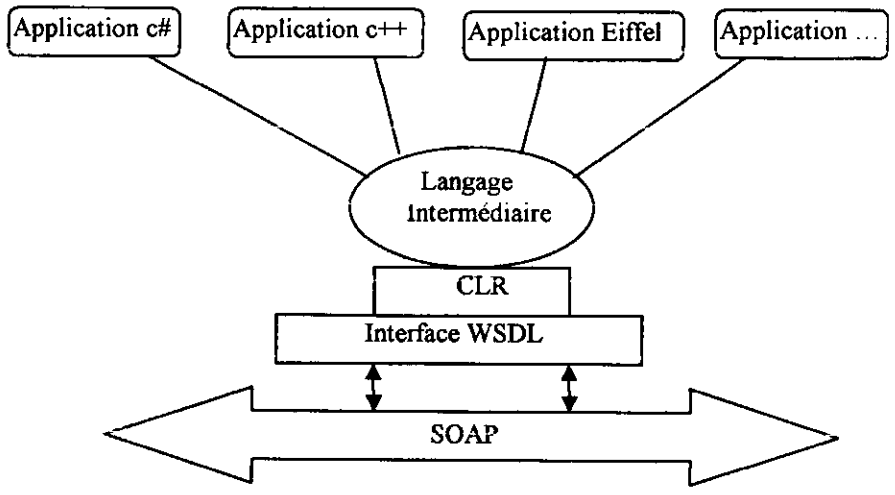


Figure2: *Architecture de l'environnement .NET*

Les applications dans .NET s'exécutent en code intermédiaire MSIL (MicroSoft Intermediate Language) dans une sorte de machine virtuelle dénommée CLR (Common Language Run-time). MSIL est un langage pivot, il permet de représenter dans un même formalisme des applications programmées dans différents langages tels que C#, C++, Eiffel, VB, etc. Les applications compilées sous la forme de code intermédiaire se présentent sous la forme de binaires exécutables portables CLR assure la compilation en code natif des programmes (les logiciels .NET sont compilés, contrairement à Java), la gestion mémoire, la gestion des erreurs et les contrôles de sécurité. Il fournit l'accès au système d'exploitation et aux ressources système. Il est au coeur de l'unification des langages. Le fait que SOAP soit basé sur XML est fondamental, car il rend SOAP beaucoup plus attractif que d'autres protocoles tels que IIOP(CORBA), JRMP(RMI) ou bien DCOM sur IP, tout les trois protocoles reposants sur des flux binaires, et donc plus difficile à gérer .

5.2 Présentation de .NET Remoting Framework :

.Net Remoting Framework est la nouvelle réponse de Microsoft à J2EE. En effet comme DCOM, cette nouvelle technologie basée sur les principes de .NET permet de créer des applications distribuées sur des réseaux locaux (Intranet) ou via Internet. Ce n'est donc pas une révolution chez Microsoft mais une évolution. DCOM devenant sous .NET.

Microsoft .NET Remoting Framework offre une structure riche et extensible permettant à des objets existant dans différents domaines d'applications, différents processus et sur différentes machines de communiquer sans problème. .NET

Remoting propose un modèle de programmation à la fois très simple et très puissant. Cette structure intègre un certain nombre de services, parmi lesquels la prise en charge de l'activation et de la durée de vie des objets, des canaux de communication responsables du transport des messages de et vers les applications distantes.

6 Synthèse:

Il serait intéressant de terminer ce chapitre en comparant les différentes technologies présentées ci-dessus.

Devant cette multitude de normes, il n'est pas évident de déterminer quelle architecture est la plus efficace pour intégrer des sources de données hétérogènes et distribuées.

Java RMI est très simple et efficace, mais ne fonctionne que dans un environnement Java, en plus, RMI est un interpréteur de Bytecode (code intermédiaire) et il est relativement lent comparativement à CORBA et à DCOM par exemple. CORBA et DCOM sont compilés et donc, plus efficaces que RMI. Coté développement d'applications.

CORBA permet de s'affranchir des architectures et des langages utilisés, objectif repris par .NET, à l'aide d'un langage de définition d'interface IDL (Interface Définition Language) et d'un protocole commun de transport, il est indépendant vis-à-vis des systèmes d'exploitation et peut être programmé avec un grand nombre de langages. CORBA est compilé (il génère un binaire), donc il est efficace et rapide mais dans le cas des services Web.

COM et DCOM ont été écrits pour faciliter la communication entre composants logiciels de Windows. Ce protocole reste largement restreint au monde Windows et dans le contexte des Intranets, ce qui n'est pas avantageux dans notre projet, puisque les données à intégrer se trouvent sur des plates-formes hétérogènes. Tout comme CORBA, DCOM est difficile à déployer.

L'utilisation de CORBA ou de DCOM nécessite de compiler et distribuer des souches clients pour chaque type de clients avec qui l'on communique. Ce n'est évidemment pas très pratique, en particulier quand le nombre de plateformes différentes ou de langages différents est important, ce qui est le cas dans notre

application ou il y a un grand nombre de bases hétérogènes qui adhèrent à la fédération.

SOAP est un protocole simple et extensible, il est indépendant des systèmes d'exploitation et des langages. Il offre un déploiement assez léger et rapidement évolutif, à la différence de CORBA, de DCOM ou de RMI ou l'installation de binaires et de descripteurs de déploiement lourds est nécessaire pour interopérer. SOAP étant développe au-dessus du protocole http.

Un autre avantage de SOAP : il fonctionne avec l'infrastructure Internet existante (http), ce qui est avantageux des données dans notre projet puisque la communication se fait à travers un réseau. En effet, l'échange de données dans notre application se fait entièrement au format XML. Il était donc naturel continuer à utiliser un protocole utilisant du XML, en plus, ce protocole n'impose pas l'utilisation d'un ORB (CORBA DCOM) ou d'un serveur Web particulier (IIS, Apache..).

SOAP est donc un très bon protocole si l'on cherche à dialoguer avec des plateformes hétérogènes, utilisant un réseau pour s'échanger des données au format XML ce qui est particulièrement le cas dans les applications de type invocation de services de recherche d'informations sur des base de données distribuées et hétérogènes utilisant XML comme modèle de données commun.

	CORBA	**DCOM**	**RMI**	**NET remoting framework**	**SOAP**
Fournisseurs	OMG	Microsoft	SUN	Microsoft	W3C
Plates-formes	Multiples	Windows	Multiple	Windows cl Linux	Multiple
Langages de programmation	Multiples	Moins multiple	Java	Multiples	Multiples
Langage de définition d'interface	IDL Corba	IDL Microsoft	Java	XML	XML
Protocole de communication	HOP	IP/IPX	JRMP ou HOP	TCP, HTTP SNMP	RPC,HTTP SNMP
Intégration de XML	Non	Non	Non	Oui	Oui

Table 1: *Comparaison entre les protocoles de technologie distribuée.*

Conclusion :

Nous avons tenté dans cette partie de donner un aperçu sur les Principales solutions permettant de développer des applications distribuées.

La première solution est les systèmes à objets distribués. Nous avons pris le cas de quatre architectures, DCOM, CORBA, Java RMI et .Net Remoting. Ces solutions restent des merveilles sur le plan théorique mais elles restent trop complexes à réaliser ou à gérer. En même temps, chaque architecture a ses limites, par exemple, DCOM est limité par sa dépendance des. Plates-formes Microsoft, RMI du langage Java et CORBA par sa complexité de développement et de déploiement des applications.

En parallèle, pour pallier aux difficultés rencontrées avec les systèmes à objets distribués, les développeurs tournent de plus en plus vers des solutions plus simples et plus souples basés sur le standard d'échange de documents XML. Les principaux constructeurs informatiques sont allés même plus loin en proposant des protocoles basés sur XML tels que SOAP.

Cette solution est en effets très simple à réaliser et très peu coûteuse, mais elle n'atteigne quand même pas la structuration en objets des différents acteurs du système. Cependant, elle reste suffisante pour des systèmes moins complexes tels que le système que nous proposant pour l'interrogation des sources de données hétérogènes et distribuées.

Chapitre IV

La conception

Introduction:

L'objectif de notre projet est d'offrir un système qui permet à son utilisateur d'interroger d'une façon uniforme un ensemble des sources de données hétérogènes et distribuées. Les sources de données peuvent être structurées (bases de données paradoxe, SQLserver..), semi structurées (document xml), ou non structurées (fichiers plats).

Dans cette partie nous représentons le schéma général de système et son fonctionnement, les règles de passages et nous modélisons le système en utilisant le langage de modélisation UML (Unified Modeling Language).

1. **Architecture de système :**

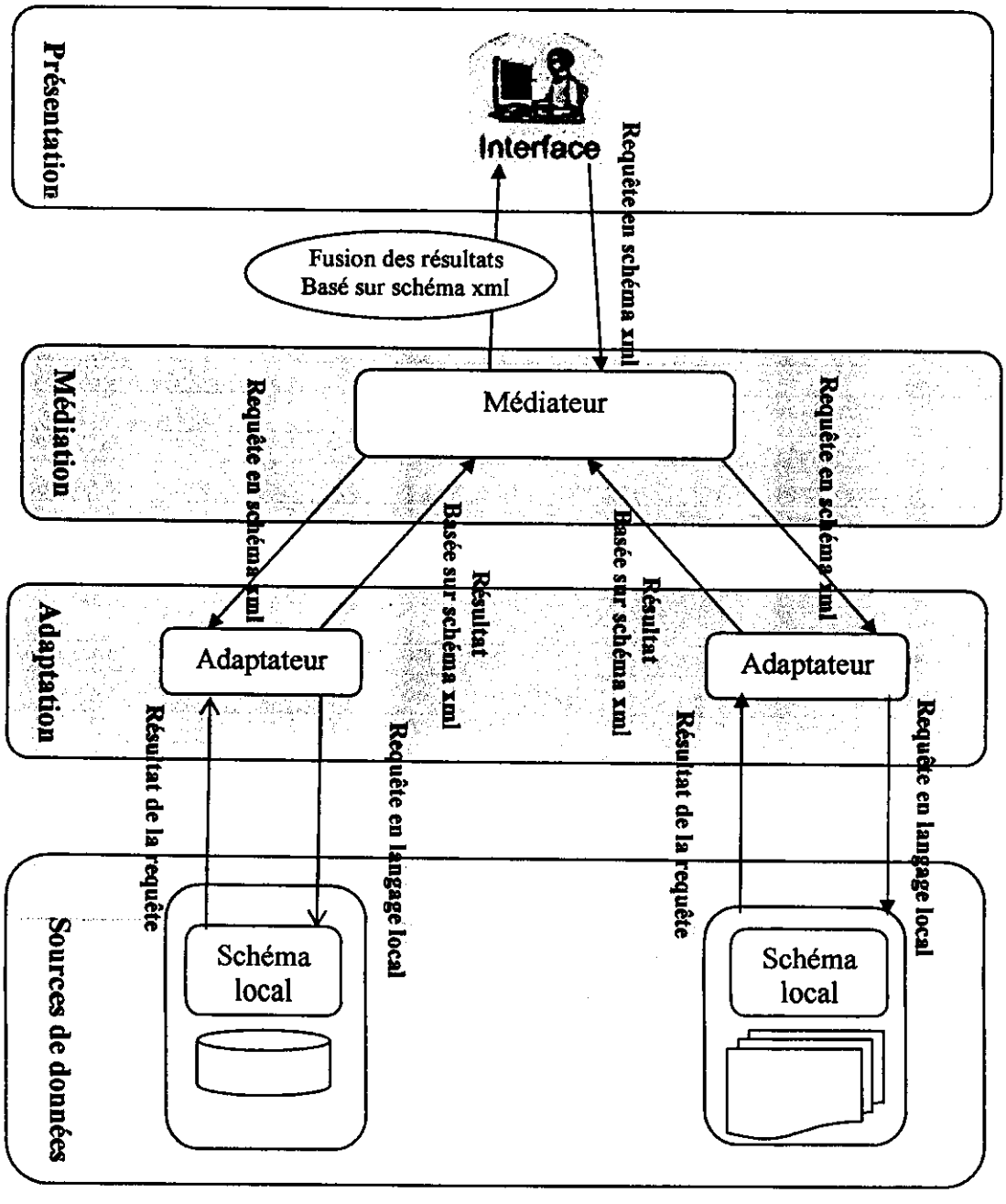


Figure01 : Architecture de système

Nous ne nous intéressons pas dans notre système par l'application cliente mais par l'uniformisation d'accès aux sources de données c-a-d la médiation et l'adaptation.

La figure précédente décrit l'architecture générale du système réalisé. Dans cette architecture, l'utilisateur envoie sa requête vers le module médiateur. Le module médiateur reçoit la requête en format XML compréhensible par les différents adaptateurs du système.

La formulation de la requête en XML sert à structurer cette dernière à l'aide d'une norme de balisage, ce qui facilite énormément sa traduction en langage de requête natif des sources de données.

Le médiateur envoie pour chaque adaptateur la requête en schéma XML. Ce dernier s'occupe de la transformer en une requête compréhensible par la source locale.

Après l'exécution de la requête au niveau de la source locale, l'adaptateur convertit les résultats obtenus après l'exécution de la requête en un format respectant le schéma XML.

Les résultats sont envoyés par les différents adaptateurs au médiateur, ce dernier s'occupe de fusionner les résultats en se basant sur le schéma XML.

Le médiateur renvoie le résultat de la requête à l'utilisateur, La communication entre les sources de données et notre système s'effectue en utilisant l'adresse IP de serveur base de données.

2 Les règles de passage:

Les règles de passage permettent de passer d'un schéma local à un schéma XML, quelque soit le type de source de données (BD, document XML, fichier plat), grâce a ce travail nous pouvons avoir des sources uniformisées.

Donc nous devons définir les règles de transformation de schéma source en un schéma XML pour chaque type de source de données,

2.1 Source structurée:

Pour traduire le schéma local d'une base de données en un schéma XML nous appliquons les règles suivantes:

Règle 1 : chaque nom de la base de données devient la racine du schéma XML.

Règle 2 : Chaque table d'une BD apparaissant dans le catalogue système d'un SGBD est traduite par un noeud de la racine du schéma.

Règle 3 : Chaque attribut d'une table est traduit en un noeud fils de nœud corresponde a sa table.

Exemple: soit le modèle logique de données de la base de données "bdpaie":

Personne (matricule personne, nom, prénom)

Paie (numéro paie, date, matricule personne)*

Le schéma XML correspondant:

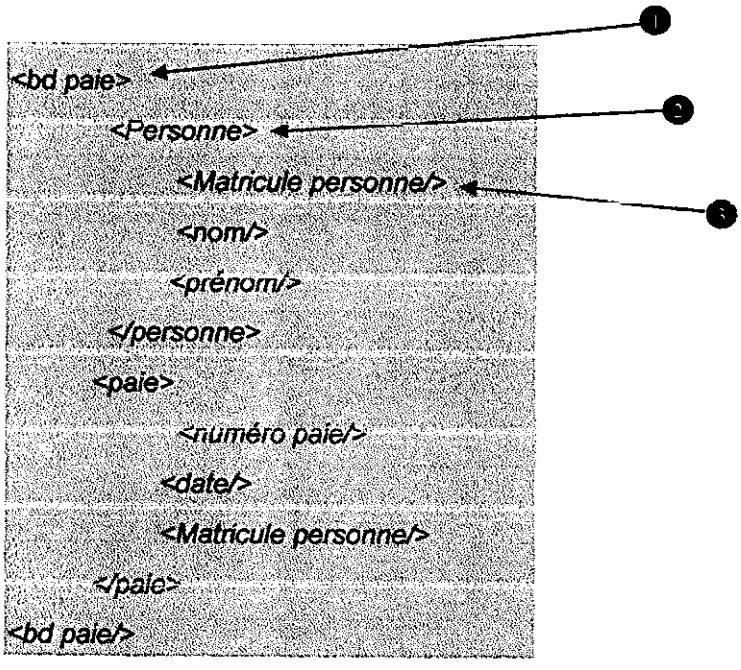


Figure02: *Schéma XML de la base de données bd paie*

2.2 Source semi structurée:

Dans ce type de source, nous appliquons les règles suivantes:

Règle 1 : chaque racine de document XML reste la racine de schéma XML.

Règle 2 : chaque nœud de document pris comme un nœud dans le schéma XML.

Règle 3 : chaque attribut d'un noeud de document est pris comme un fils de son noeud dans le schéma XML.

Exemple:soit le document XML suivant:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <facture>
  - <Client>
      <nom>rahim</nom>
      <prenom>hassen</prenom>
      <adresse>13 rue amor mihoub alger</adresse>
      <Ntelephone>021251456</Ntelephone>
    </Client>
  - <lignedeCommande>
      <nomProduit>machine (5.08 cm)</nomProduit>
      <quantite>17</quantite>
      <prix>1250.00</prix>
    </lignedeCommande>
  </facture>
```

Figure03: *Document XML "facture".*

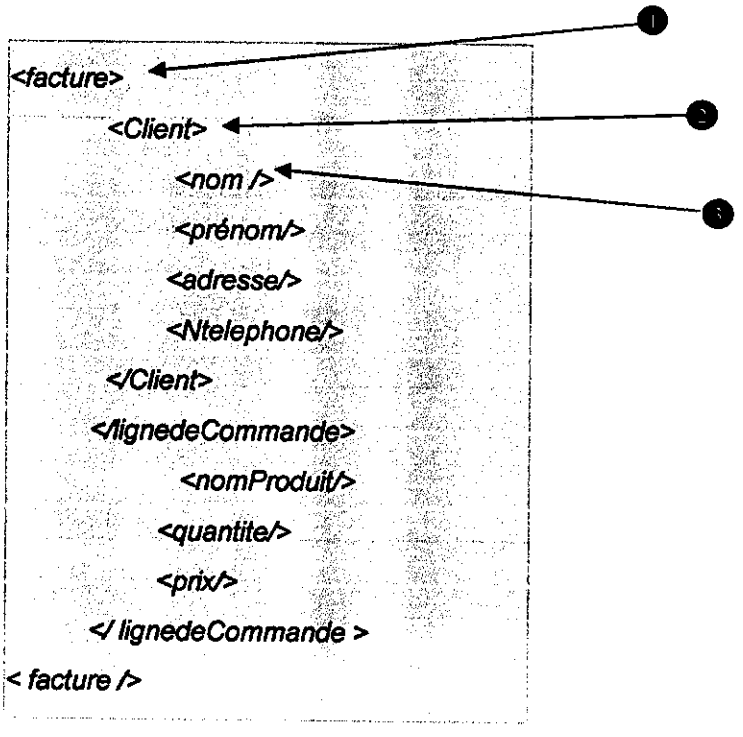


Figure04: *Schéma XML de document "facture".*

2.3 Source non structurée:

Puisque ce type de source n'a pas une structure bien déterminée, le fichier doit être transformé en un document XML, donc nous appliquons les règles suivantes:

Règle 1: le nom de fichier est pris comme la racine du document XML.

***Règle* 2:** Chaque ligne i du fichier devient un noeud porte le mon personne i.

***Règle* 3:** pour chaque mot de la ligne i devient un nœud dans le nœud personne i,

Exemple: soit le fichier texte suivant et sa transformation en document XML.

001 Salhi Socine.
002 Hasnaoui Mohamed.

Figure05: *Fichier texte"étudiant".*

<étudiant>
<personne1>
<mat> 001 </mat>
<nom> Salhi </nom>
<prenom> Hocine </prenom>
</personne1>
<personne2>
<mat> 002 </mat>
<nom> Hasnaoui </nom>
<prenom> Mohamed </prenom>
</personne2>
</étudiant>

Figure06 : *Document XML de fichier texte "étudiant".*

3 Modélisation:

3.1 Le langage de modélisation unifié (UML):

UML (Unified Modeling language) est un langage graphique de modélisation objet résultant des tentatives d'unification des méthodes OOD (Object Oriented Design) de Grady Booch, OMT (Object Modeling Technics) de James Rumbaugh et OOSE (Object Oriented Software Engineering) de Ivar Jacobson. Les tentatives d'unification de ces méthodes ont permis la construction de plusieurs versions de la méthode unifiée (version 0.8 en 1995, 0.9 en Juin 1996 et 0.91 en octobre de la même année).

C'est sous le nom d'UML version 1.0 que ce langage sera remis à l'OMG (Object Management Group) en Janvier 1997 en vue de sa standardisation.

UML est avant tout un support de communication performant qui facilite la représentation et la compréhension des modèles objet :

- Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation des modèles objets.
- L'aspect formel de sa notation limite les ambiguïtés et les incompréhensions.
- Son indépendance par rapport aux langages de programmation et aux domaines d'application en fait un langage universel.

Ainsi donc, UML a été défini pour modéliser des systèmes. Pour ce faire, il dispose d'outils qu'on appelle des « diagrammes ». Ces diagrammes sont au nombre de neuf et sont répartis en deux classes : les diagrammes offrant une vue statique du système et les diagrammes offrant une vue dynamique du système. Ces diagrammes sont les suivants :

1. **Diagrammes offrant une vue statique du système :**
   - Diagramme des cas d'utilisation
   - Diagrammes d'objets
   - Diagrammes de classes
   - Diagramme de déploiement
   - Diagramme de composants
2. **Diagrammes offrant une vue dynamique du système :**
   - Diagrammes de collaboration
   - Diagrammes de séquences
   - Diagrammes d'états/transition
   - Diagrammes d'activités

- **Diagramme de cas d'utilisation :** ce diagramme décrit le comportement général (actions et réactions) du système vis-à-vis des utilisateurs (que l'on appelle « acteurs ») du système. En d'autres termes, il s'agit de la description des fonctionnalités du système.
- **Diagramme de classes :** il décrit la structure des classes : identité, relations entre les classes (héritage et composition), attributs et opérations.
- **Diagrammes d'objets :** un tel diagramme représente simplement une instance du diagramme de classes pour des cas concrets à un instant donné.
- **Diagramme de déploiement :** ce diagramme illustre la disposition spatiale et physique des différents matériels composant le système.

- **Diagramme de composants :** ce diagramme décrit le choix des composants : bibliothèques, fichiers, code,...
- **Diagramme de collaboration :** ce diagramme décrit les interactions entre groupes spécifiques d'objets par représentation des liens et des envois de messages.
- **Diagramme de séquences :** il décrit les interactions entre les acteurs et les objets d'un point de vue temporel sous la forme d'un scénario qui constitue une instance particulière du cas d'utilisation.
- **Diagramme d'états/transition :** ce diagramme décrit les aspects et le séquencement des opérations en relation avec le temps. Il s'agit ici de mettre en évidence les évènements qui marquent un changement, des états particuliers, etc. on ne s'intéresse cependant qu'au séquencement et non pas à ce qu'effectuent les opérations.
- **Diagramme d'activités :** il s'agit là d'une variante du diagramme précédent (diagramme d'états/transition). Ce diagramme met en évidence les activités et modélise le comportement interne des opérations.

3.2 Diagramme de cas d'utilisation :

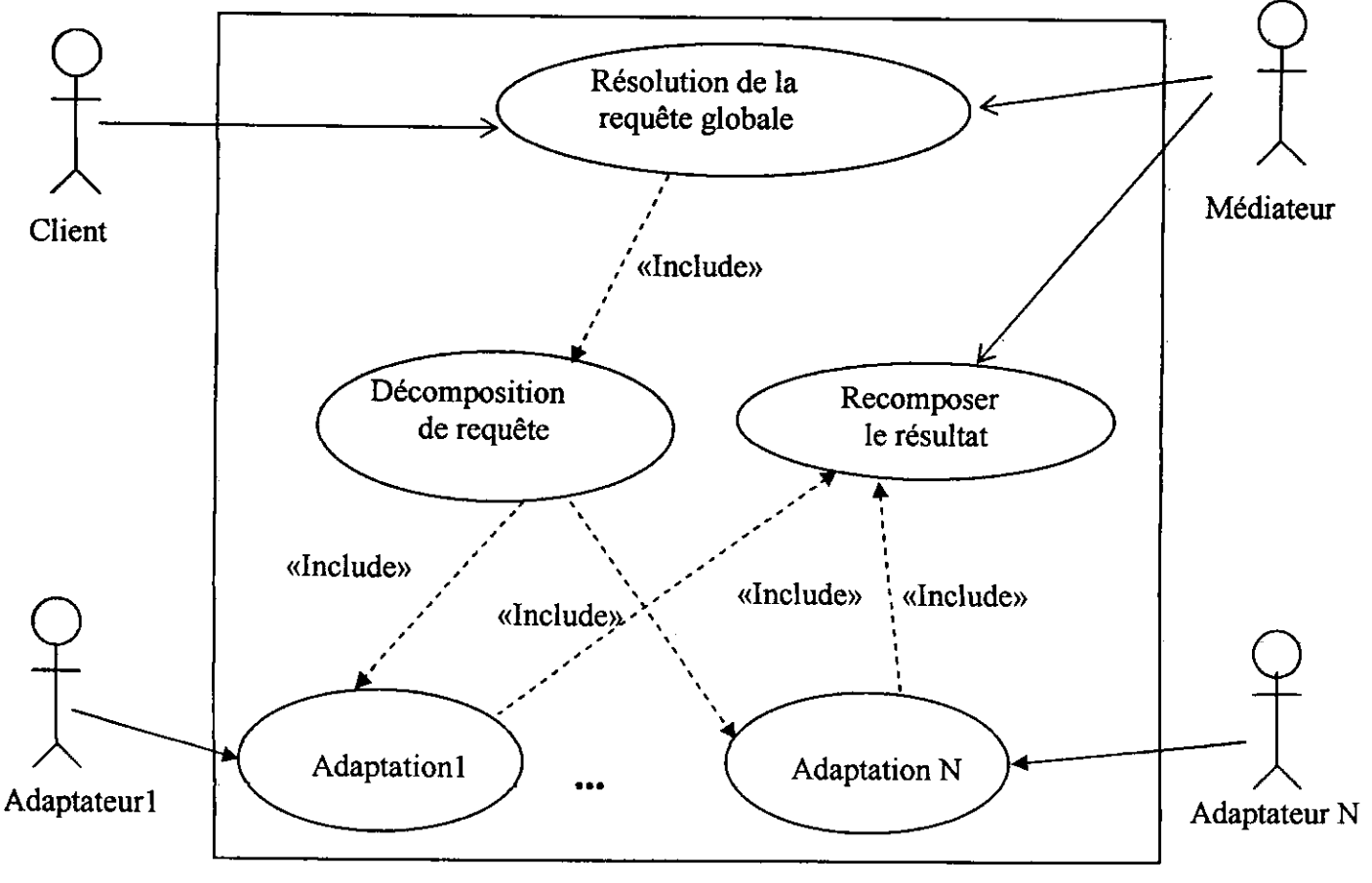


Figure07 : Diagramme de cas d'utilisation de système.

3.3 Scénario de cas d'utilisation:

Résolution de la requête globale

1- l'utilisateur formule une requête globale.
2- Envoie la requête en XML au médiateur.

Décomposition de requête

1- Le médiateur reçoit la requête en XML.
2- le médiateur connecter les adaptateurs concernés.
3- Le médiateur Consulter le schéma fédéré.
4- Le médiateur décomposer la requête.
5- Le médiateur dispatcher la requête vers les adaptateurs concernés.

Adaptation

Après que l'adaptateur reçoit la requête.
- si la source de données est structurée.
1-l'adaptateur traduire la requête en langage local, pour ce la il doit:
- consulter la table des méta-données.
- consulter la table de correspondance.
2- envoie la requête vers le middleware ODBC pour l'exécution.
- sinon
1- l'adaptateur utilise un mécanisme de recherche pour extraire les données, en consultant
- la table des méta-données.
- le table de correspondance.

Après l'exécution da la requête:
1- l'adaptateur traduire le résultat de langage local en XML pour ce la il consulte :
- la table des méta-données .
- la table de correspondance.
2- envoie le résultat au médiateur.

Recomposer le résultat

Une fois les résultats arrivant eu médiateur, ce dernier doit:
1- fusionner les résultats.
2- Afficher le résultat.

3.4 Diagramme de séquence:

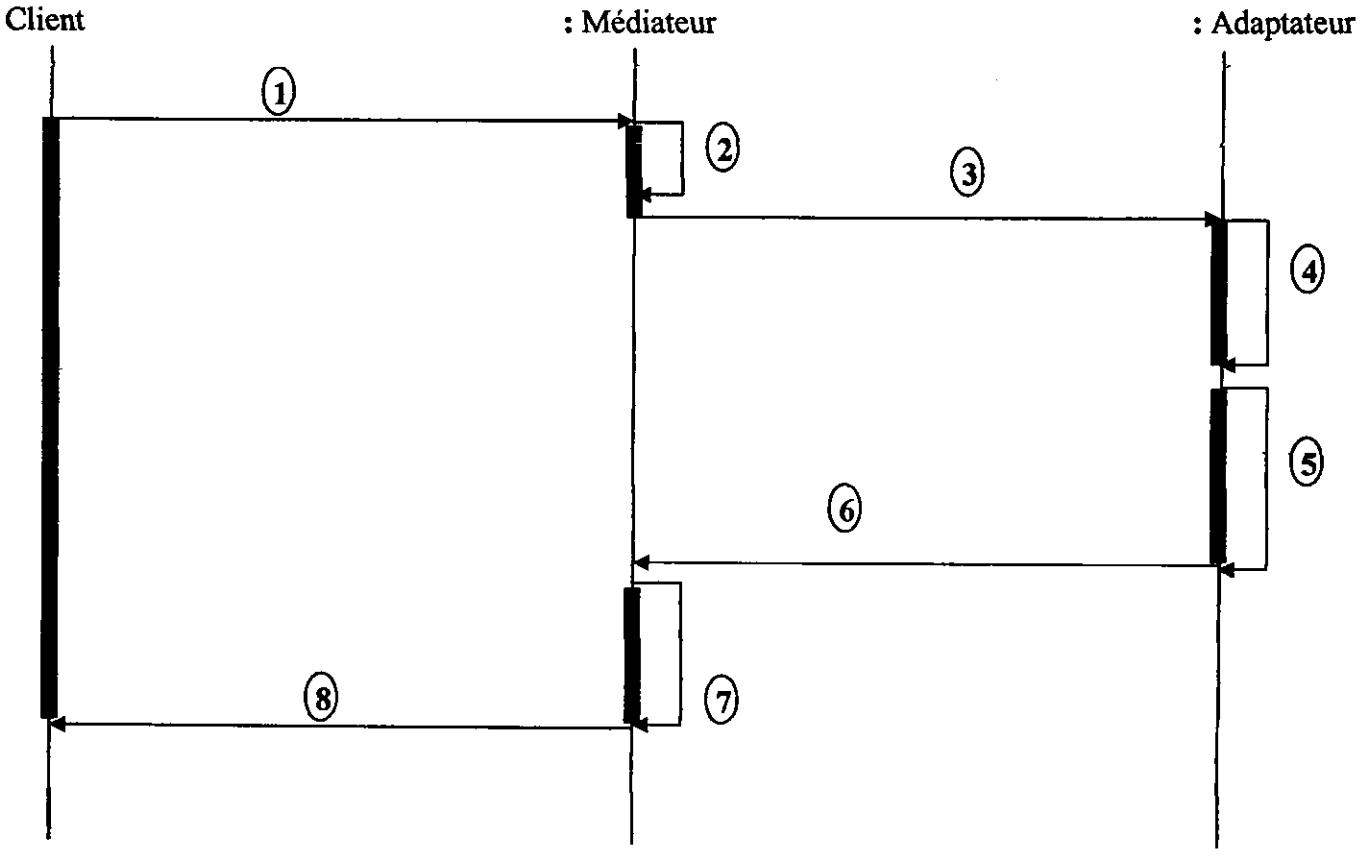


Figure 08 : Diagramme de séquence.

numéro	Description
1	Requête en XML
2	Requête décomposer
3	Requête dispatcher vers les adaptateurs concernés
4	Requête en langage local
5	Résultat obtenu
6	Résultat en XML
7	Résultats fusionner
8	Résultat Afficher

Figure 09 **:** Description des interactions.

3.5 Diagramme de classes:

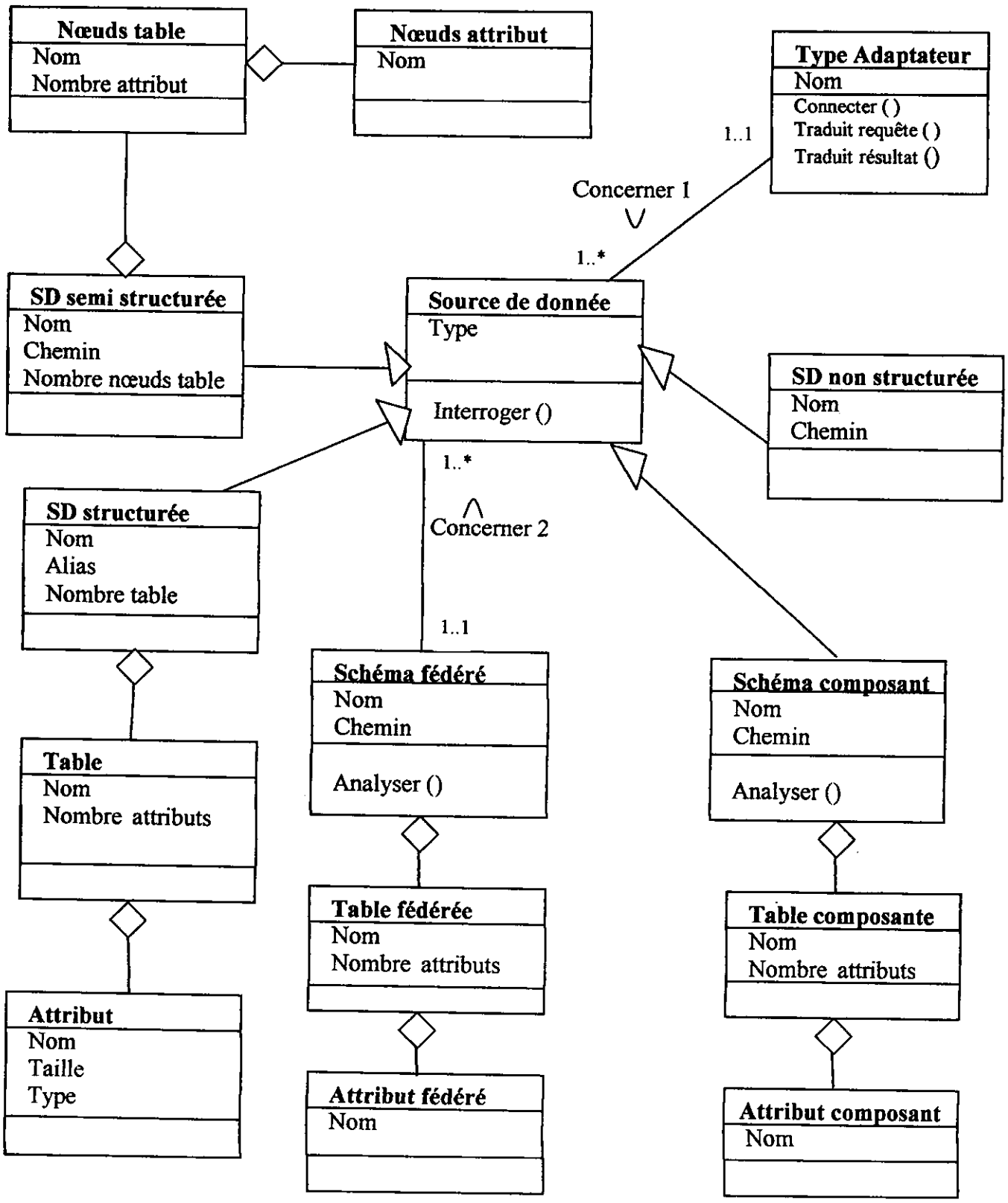


Figure 10 : Diagramme de classes.

3.6 Diagramme d'état transition :

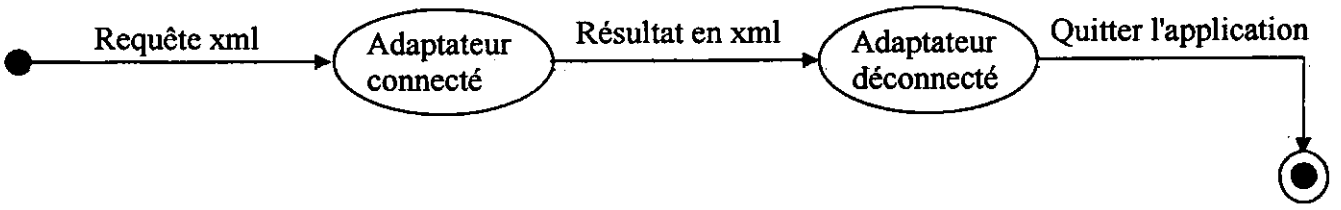


Diagramme d'état-transition pour la classe adaptateur

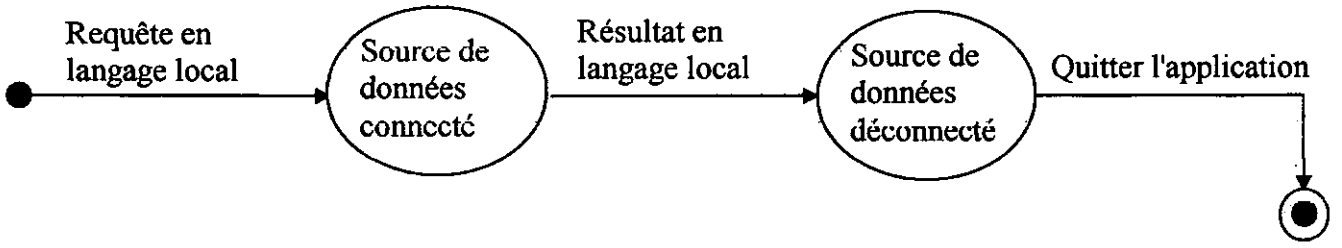


Diagramme d'état transition pour la classe source de données

Figure 11 : Diagramme d'état transition.

3.7 Diagramme de déploiement :

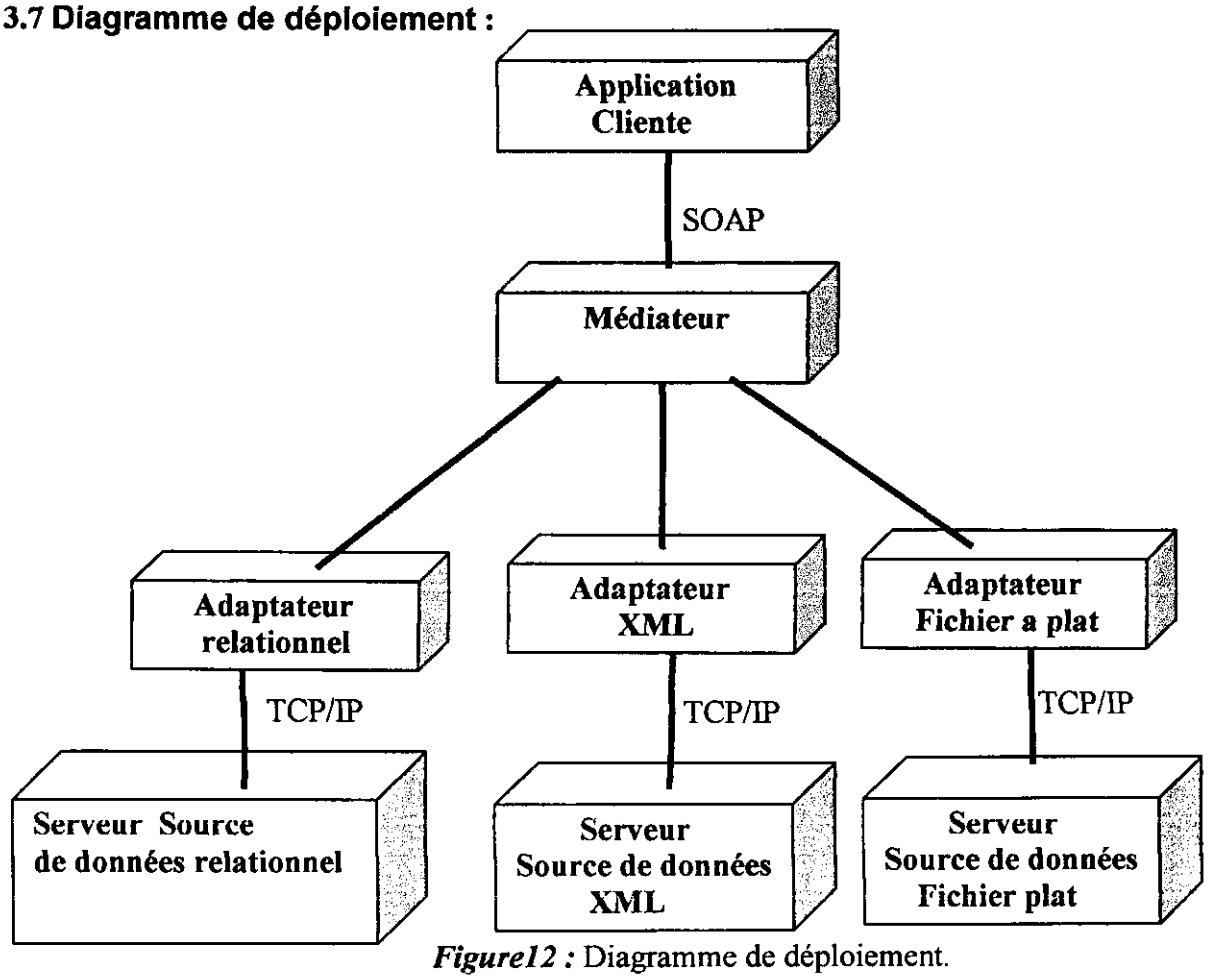


Figure12 : Diagramme de déploiement.

Conclusion:

La conception nous a permet de comprendre le système en détail, de Connaître les rôles de chaque module et la communication entre ces modules.

Les règles de passage permettent d'uniformiser les schémas locaux des sources de données pour simplifier l'analyse et l'interrogation des sources de données.

La modélisation UML nous a permet d'incarner les solutions des problèmes d'hétérogénéité et la distribution.

Au prochain chapitre, nous présentons en détail l'implémentation de ce prototype.

Chapitre V

Implémentation et réalisation

Introduction

Dans ce chapitre, nous décrivons la structuration interne des différents modules constituant notre système (module médiateur, module adaptateur, module application cliente), ainsi que leurs modes de communication. nous essayerons d'implémenter un prototype permettant d'interroger trois types de source de données, à savoir : celles qui sont basées sur le modèle structuré (bases de données relationnelles comme SQL Server, Access et Paradoxe), et les autres sont sur le modèle semi structuré (document XML) et non structuré (fichiers plats).

1 L'environnement de développement :

L'implémentation de l'architecture décrite dans la partie conception a été réalisée dans un réseau local (Intranet) comprenant des PC interconnectés fonctionnant sous Windows XP et un serveur fonctionne sous WindowsXP qui est configuré comme un serveur de l'application médiateur. Pour réaliser une telle architecture, nous avons choisi le langage de programmation Pascal Objet, en utilisant Borland Delphi 7.0 pour implémenter le module médiateur, les adaptateurs et l'application cliente.

Le choix de Delphi est basé essentiellement sur :

- Borland Delphi est un environnement de programmation visuel orienté objet permettant le développement rapide d'applications 32 bits en vue de leur déploiement sous Windows et sous Linux. En utilisant Delphi, nous pouvons créer des puissantes applications avec un minimum de programmation. Delphi propose une bibliothèque des classes complète appelée la VCL (*Visual Component Library*), la bibliothèque des composants Borland multi plates formes appelées CLX et une suite d'outils de conception rapide d'applications (*RAD*) comprenant des modèles d'application, des modèles de fiche et des experts de programmation.

- La bibliothèque des classes VCL comprend des objets qui encapsulent l'API Windows ainsi que d'autres techniques de programmation utiles (Windows) ;
- La bibliothèque des classes CLX comprend des objets qui encapsulent la bibliothèque Qt (Windows ou Linux).

- Delphi comprend de nombreux composants permettant d'accéder aux bases de données et de représenter les informations qu'elles contiennent. Ils sont regroupés en fonction du mécanisme d'accès aux données, dans notre réalisation nous avons opté pour l'utilisation de BDE (Borland Database Engine), ce dernier contient les

composants qui utilisent le moteur de bases de données Borland. Le BDE définit une API importante pour l'interaction avec les bases de données.

- Delphi nous permet de travailler avec des documents XML, il définit diverses classes et interfaces pour manipuler les documents XML. Elles simplifient le processus de création, chargement, modification et d'enregistrement des documents XML. Dans notre réalisation nous avons opté pour l'utilisation de la classe TXMLDocument. Cette classe sert à représenter un document XML, elle peut lire un document XML existant à partir d'un fichier, peut être associé à une chaîne en mémoire contenue dans un document XML ou peut créer un document XML vide.
- Delphi avec ses techniques de programmation permet d'accéder, analyser les fichiers textes et les transformer en format XML.
- Nous pouvons utiliser Delphi pour concevoir aussi bien le serveur d'application, que le client appelant le serveur en utilisant la technologie SOAP.

2 Implémentation des sources de données:

Pour tester le bon fonctionnement de notre système nous avons utilisé les gestionnaires des bases de données suivants : Microsoft Access 2003, Microsoft SQL Server 2000 version 8.0 et Paradoxe 7, car ces sont des SGBDs relationnels utilisées dans les services de ressources humains et paie de la société NAFTAL, et que le service **DCSI** souhaite les interroger.

Puisque l'architecture proposée prévoit l'utilisation de différents types des bases de données, c'est-à-dire des SGBDs différents, le passage par un Middleware s'avère nécessaire. Le Middleware utilisé est l'ODBC de Microsoft (*Open Data Base Connectivity*). Pour le cas d'une source de données semi structurée nous avons pris les documents XML qui seront analysés en basant sur ses structures arborescentes, et pour non structurée nous avons pris le cas de fichier plat qui va transformer en document XML pour simplifier l'analyse.

3 Implémentation de l'application médiation :

3.1 Implémentation de médiateur:

Le médiateur est un script CGI signifie *Common Gateway Interface*, autrement dit, une passerelle interface standard entre l'application cliente et les adaptateurs, permettant de faire exécuter une requête venant de l'application cliente. Ce script est implémenté en langage Pascal Objet en utilisant Delphi.

Le médiateur offre 04 services:

3.1.1. Le service d'inscription :

Ce service est invoqué par les adaptateurs lors de la configuration, ce service permet à un adaptateur de s'inscrire dans la table des adaptateurs connectés, l'adaptateur concerné sera ajouté à la table et sera ensuite pris en compte lors de la prochaine requête. Ce service permet de mettre à jour les informations de la table des adaptateurs, il a comme paramètre d'entrée le nom du adaptateur qu'il invoque. Et reçue trois fichiers XML, un comporte tout les noms des source de données concernes cet adaptateur (figure V.08), un autre comporte le schéma composant de la source configurée (figure V.09), et l'autre fichier comporte le schéma fédéré correspondant (figure V.11).

3.1.2. Le service de recherche :

Une fois le médiateur reçoit la requête formulée en XML, il consulte la table des adaptateurs (figure V.03) et les schémas fédérés pour la décomposé en sous requêtes, et ensuite les sous requêtes seront dispatchées c_a_d chaque requête destiner pour un adaptateur spécifie, en invoquant les services de recherche de tous les adaptateurs inscrits dans la table des adaptateurs au moment de l'arrivée de la requête,

```
Début
 Tq ∃ requête i dans la requête globale faire
   Debut
     Tq ∃ nom adaptateur i dans la table des adaptateurs faire
      Debut
        Tq ∃ nom de schéma fédéré dans la table d'adaptateur i faire
          Debut
            Si ∃ attribut de la requête i dans le schéma fédéré
                Alors
                Mettre requête i dans la requête d'adaptateur i
                Fsi
          Fin
      Fin
 Fin
Fin.
```

Figure01 : Algorithme de décomposition de requête globale.

La *figure02* représente un exemple d'une requête en XML.

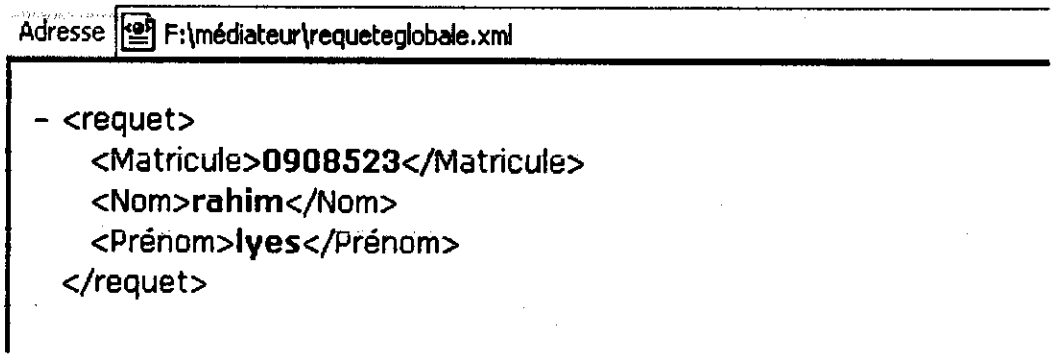


Figure02: *Requête globale en XML.*

3.1.3. Le service de désinscription :

Ce service est invoqué par les adaptateurs lors de la configuration, ce dernier permet à un adaptateur d'informer le médiateur qu'il ne fait pas partie des adaptateurs connectés, l'adaptateur concerné sera supprimé de la table et ne sera pas pris en compte lors de la prochaine requête.

Ce service permet de mettre à jour les informations de la table des adaptateurs, il a comme paramètre d'entrée le nom d'adaptateur qu'il invoque.

La table des adaptateurs sera sauvegardée dans un fichier XML et cela à cause de la simplicité de la manipulation de ce genre de fichier. *La figure 03* montre un exemple d'une table des adaptateurs.

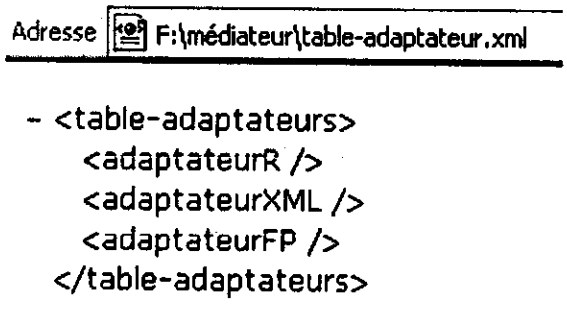


Figure03: *Table des adaptateurs en XML.*

3.1.4. Fusion des résultats:

Ce service est invoqué par le médiateur lors de la réception des résultats de différents adaptateurs. Les résultats seront fusionnés et sauvegardés dans un fichier XML puis envoyé à l'application cliente pour le afficher.

```
Début
  Tq ∃ résultat i dans résultat relationnel faire
    Début
        Mettre résultat i dans résultat fusionner
    Fin
  Tq ∃ résultat i dans résultat XML faire
    Début
        Mettre résultat i dans résultat fusionner
    Fin
    Tq ∃ résultat i dans résultat fichier plat faire
    Début
        Mettre résultat i dans résultat fusionner
    Fin
Fin.
```

Figure04 : *Algorithme de fusion des résultats.*

La figure suivante représente un fichier XML qui porte les résultats.

Adresse f:\mediateur\resultatg.xml

```
- <resultatG>
    <matricule>0908523</matricule>
    <nom>rahim</nom>
    <prenom>lyes</prenom>
    <codefonction>chefprojet</codefonction>
    <codedeplome>magi</codedeplome>
    <codedirection>dircsi</codedirection>
    <Nccp>65667894</Nccp>
    <salaire>26000,00</salaire>
    <adresse>bni messous alger</adresse>
    <numerotéléphone>063022561</numerotéléphone>
  </resultatG>
```

Figure05: *Résultats en XML.*

4. Implémentation des adaptateurs:

Nous avons décomposé l'adaptateur en deux instances : un pour la configuration des sources de données et l'autre pour la traduction et l'exécution de la requête. La figure suivante représente la fenêtre principale de la configuration :

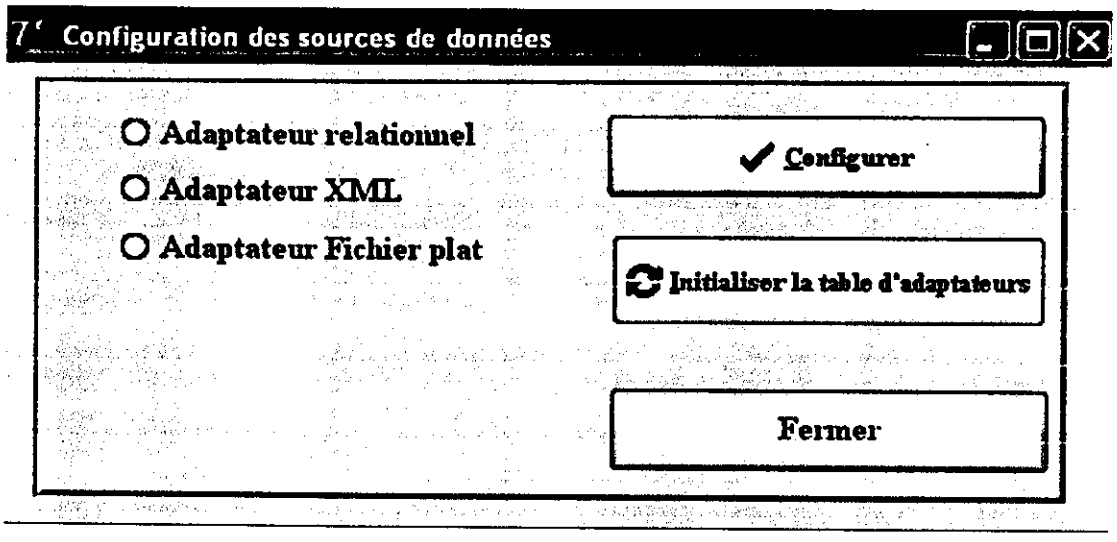


Figure06: Fenêtre principale de la configuration.

Comme la montre la figure précédente Il y a 03 adaptateurs:

4.1 Adaptateur relationnel:

4.1.1 La configuration :

Puisque cet adaptateur est désigné pour les sources de données structurées (base de données), l'adaptateur relationnel peut accéder aux bases de données relationnelles via le Middleware ODBC. Après Le choix de cet adaptateur le bouton *Configurer* permet d'accéder à la fenêtre suivante.

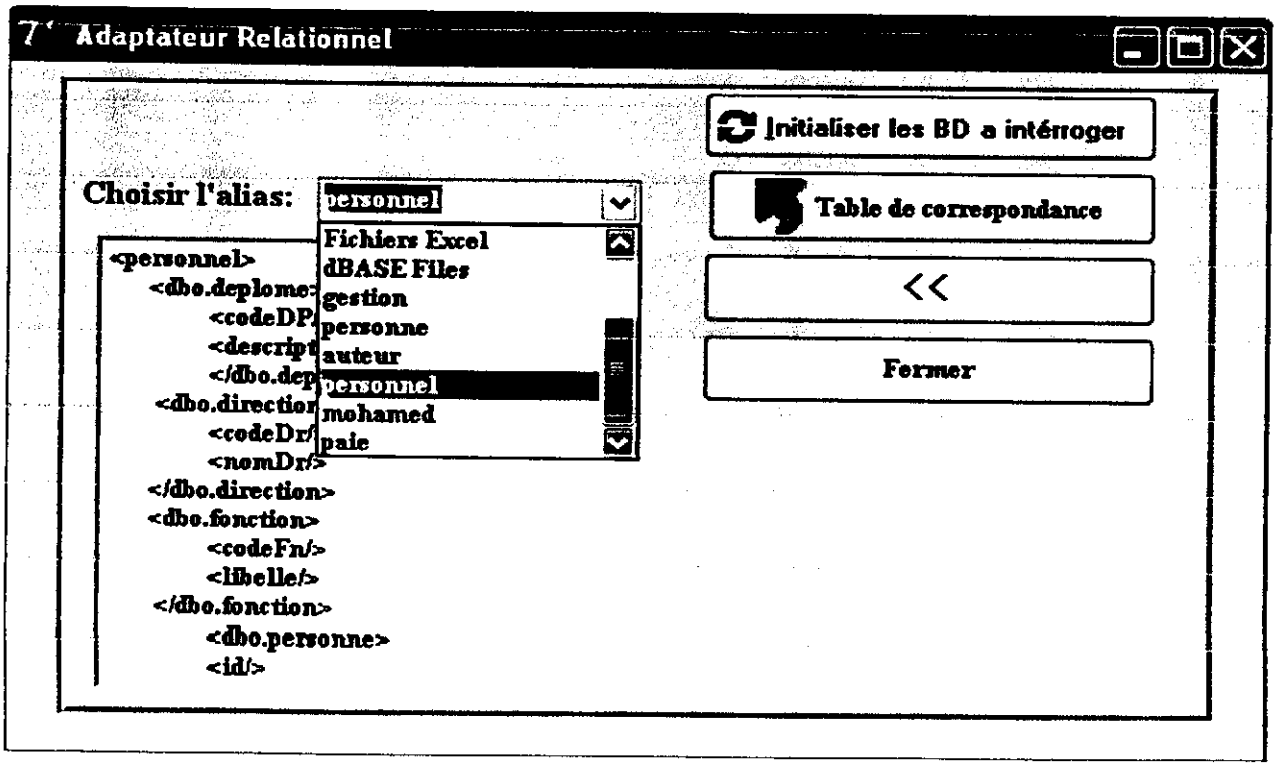


Figure 07: La fenêtre de configuration des bases de données.

Lors de choix de l'alias base de données, l'adaptateur accéder au base de données, ajouter l'alias dans la table *AdaptateurR* et créer le schéma composant de cette base de données en se basant sur les règles de passage, et le sauvegarder dans un fichier XML, ce dernier porte les informations qui concerne cette source tel que l'alias, les noms des tables et les attributs de chaque table.

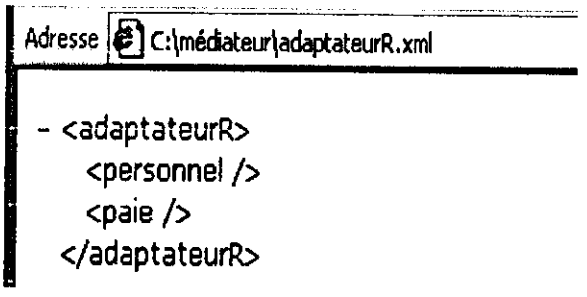


Figure 08 : *Table AdaptateurR en XML.*

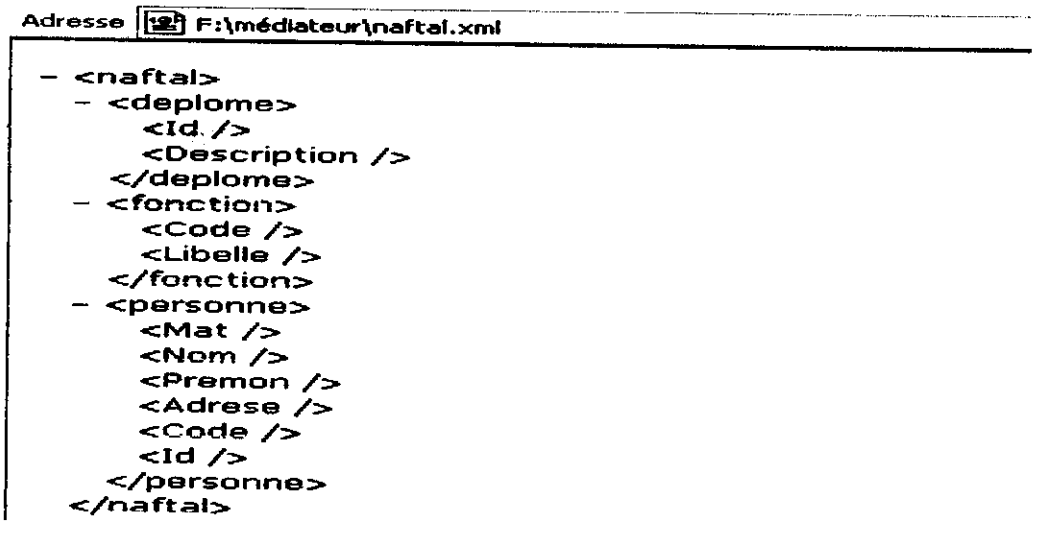


Figure 09 : *Schéma composant en XML.*

Pour éliminer l'hétérogénéité sémantique, le configurateur passe par le bouton *table de correspondance* à l'étape suivante

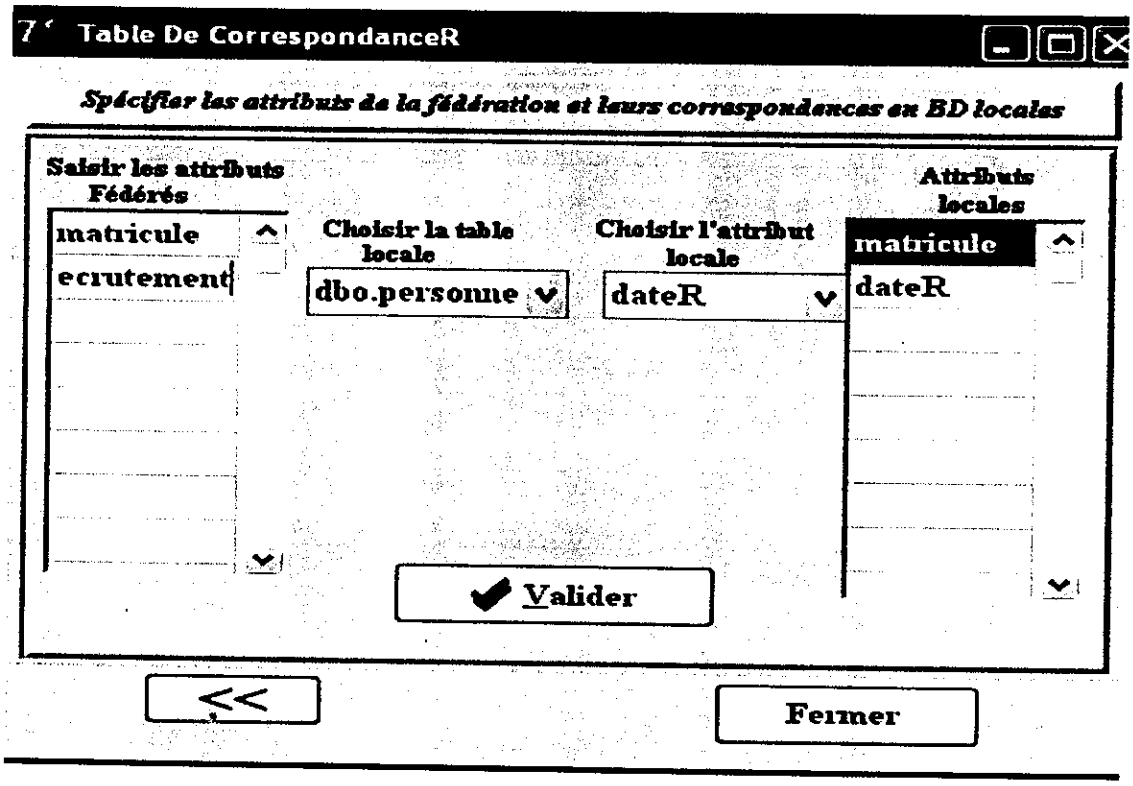


Figure10: *Le fichier de schéma fédéré de l'exemple précédant.*

Cette étape consiste de déterminer les attributs fédérés et ses correspondances dans la source locale, et sauvegarder son schéma fédéré dans un fichier XML. La figure suivante représente le schéma fédéré de schéma composant précédent (figure V.09):

Adresse F:\médiateur\naftal.xml

```
- <naftal>
  - <deplome>
      <Identifientdeplome />
      <Description />
    </deplome>
  - <fonction>
      <Codefonction />
      <Libelle />
    </fonction>
  - <personne>
      <Matricule />
      <Nom />
      <Premon />
      <Adrese />
      <Codefonction />
      <Identifientdeplome />
    </personne>
  </naftal>
```

Figure11 : *Schéma fédéré en XML.*

4.1.2 La traduction et l'exécution de la requête:

Après la configuration, l'adaptateur reçoit la requête, une fois l'adaptateur reçoit une requête il va la traduire en requête SQL pour l'exécuter. La génération de cette requête base sur les schémas composants, schéma fédéré et la requête reçue. Le résultat sera sauvegardé dans un fichier XML et qui porte le nom *resultatR* et envoyé au médiateur.

4.2 Adaptateur XML: cet adaptateur est implémenté comme l'adaptateur relationnel sauf que l'adaptateur relationnel utilise l'ODBC et la requête SQL pour extraire les données, et l'adaptateur XML utilise un mécanisme de recherche dans le document XML.

```
Début
 Tq ∃ requête i dans la table de la requête XML faire
  Début
    Tq ∃ nom de document XML dans la table d'adaptateur XML faire
     Début
      Tq ∃ nœud dans le document XML faire
       Début
        Si le nom de nœud = attribut de la requête i alors
          Si la valeur de nœud = la valeur d'attribut de la requête i alors
            Extraire tous les nœuds frères
          Fsi
        Fsi
      Fin
   Fin
Fin.
```

Figure12 : Algorithme de recherche dans le document XML.

La figure suivante représente un document XML.

Adresse F:\médiateur\facture.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
- <facture>
  - <Client>
      <nom>rahim</nom>
      <prenom>hassen</prenom>
      <adresse>13 rue amor mihoub alger</adresse>
      <Ntelephone>021251456</Ntelephone>
    </Client>
  - <lignedeCommande>
      <nomProduit>machine (5.08 cm)</nomProduit>
      <quantite>17</quantite>
      <prix>1250.00</prix>
    </lignedeCommande>
  </facture>
```

Figure 13: Un document XML

4.2.1 La configuration :

La configuration du document précédent

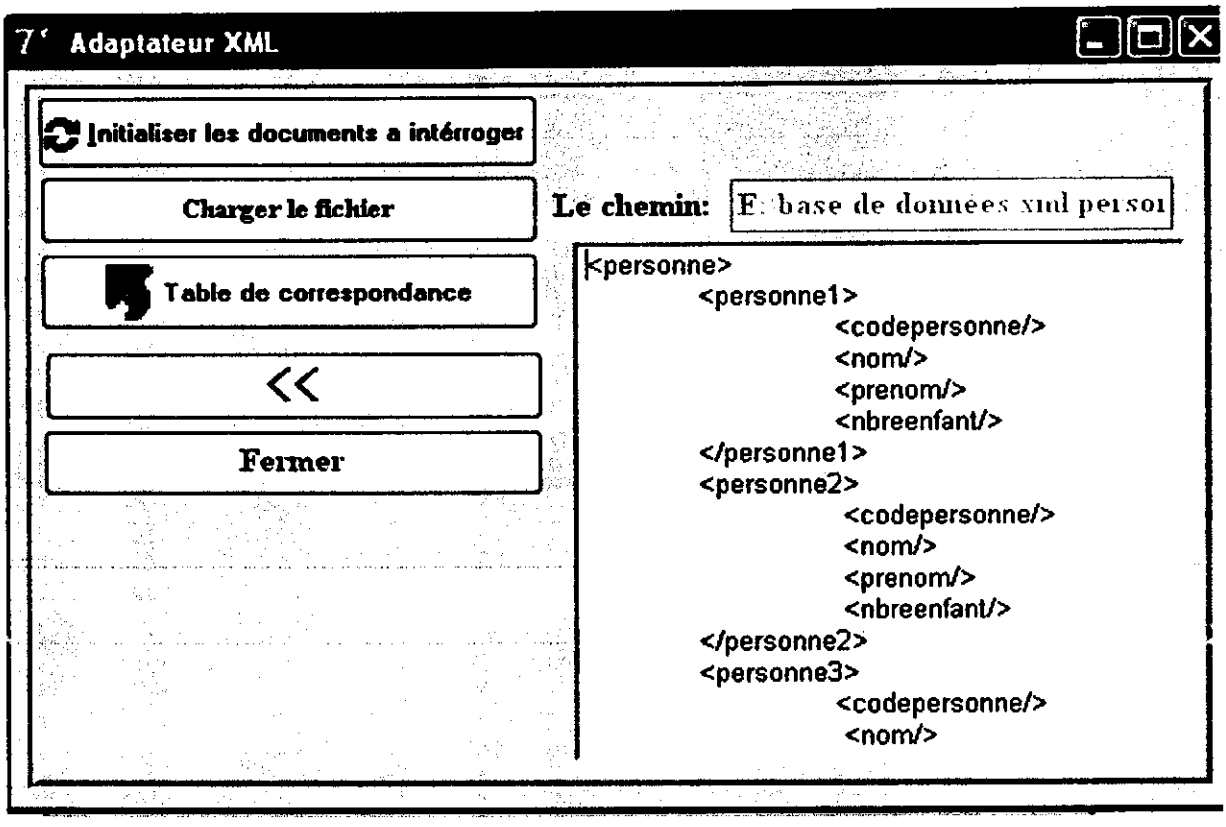


Figure14 : La fenêtre de configuration d'un document XML.

Le bouton *initialiser les document a interroger* supprimer tous les noms des documents XML de la table *AdaptateurXML.*

Le bouton *Charger le fichier* permet d'accéder au document XML local ou distant, garder son adresse et créer son schéma composent, ainsi qu'ajouter le nom de document dans la table *AdaptateurXML.* Pour éliminer l'hétérogénéité sémantique nous accédons à la table de correspondance et déterminé les attributs fédérés de la même façon dans l'adaptateur relationnel.

4.2.2 Traduction et l'exécution de la requête:

Une fois l'adaptateur XML reçoit la requête de médiateur, ce dernier utilise le mécanisme de recherche pour interroger les documents qui sont été configurés par cet adaptateur. Le résultat sera sauvegardé dans un fichier XML puis envoyé au médiateur pour le fusionner avec les autres résultats.

4.3 Adaptateur Fichier Plat :

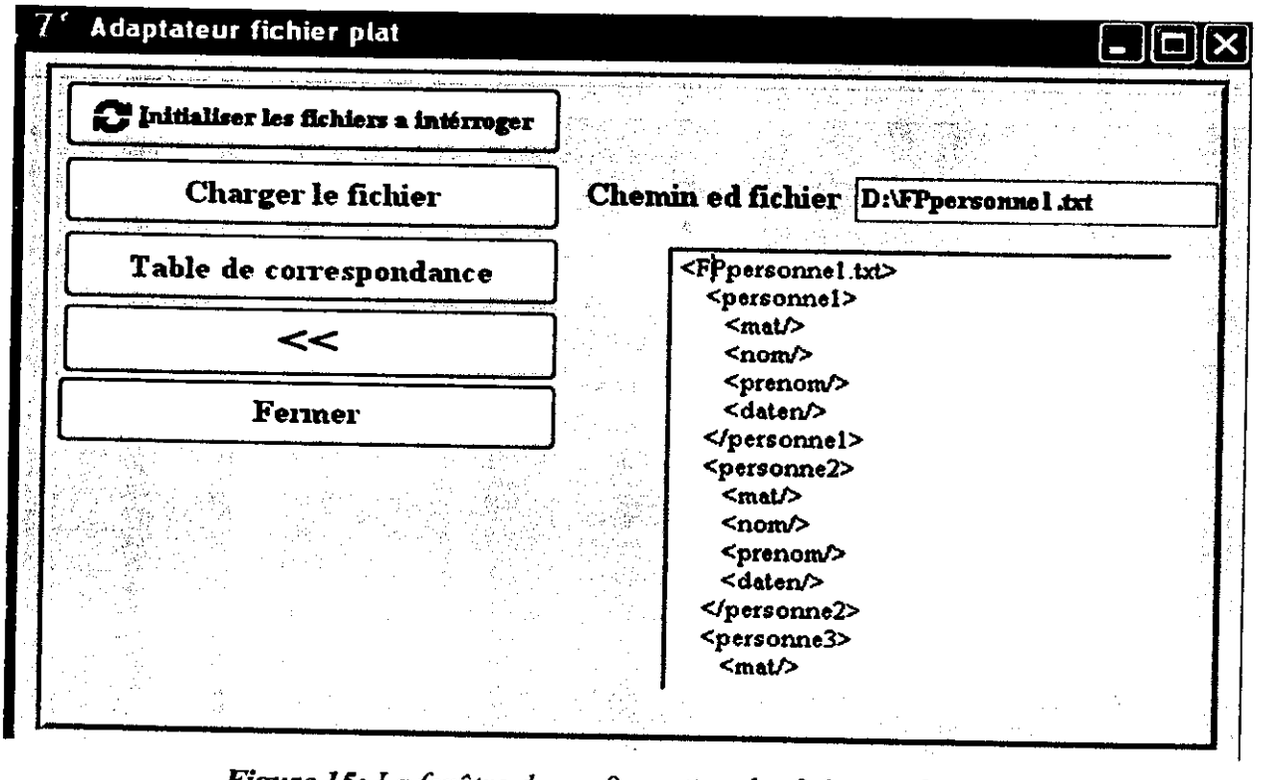


Figure 15: La fenêtre de configuration des fichiers plats.

4.3.1 Configuration:

Comme dans les deux autres adaptateurs le bouton *initialiser les fichiers a interroger* capable de supprimer les nom des fichiers que la table *Adaptateur* comporte, le bouton *Charger le fichier* donne la main au configurateur de sélectionner le fichier qui va configurer, lors de la sélection, le fichier plat va transformer en un document XML selon les règles de passage, et ajouter son nom dans la table *adaptateurFP* qui signifier adaptateur fichier plat. le bouton *table de correspondance* Pour éliminer l'hétérogénéité sémantique nous accédons à la table de correspondance et déterminé les attributs fédérés de la même façon dans les autres adaptateurs.

4.3.2 L'exécution de la requête:

Nous avons utilisé le même mécanisme de recherche utilisé dans l'adaptateur XML, qui permet d'extraire les données de document XML correspondre au fichier plat configuré, après l'exécution, le résultat va sauvegarder dans un fichier XML et envoyer au médiateur qui va le fusionner avec les autres résultats.

5 l'implémentation de l'application cliente:

Nous avons dit que nous n'intéressons pas par l'application cliente, cette phase est seulement pour tester le fonctionnement de notre système. Cette application est implémentée par le langage de programmation Pascal Objet, en utilisant Borland Delphi 7. Après la phase de la configuration des sources de données, la fenêtre suivante permet au client d'interroger les sources de données suivantes:

- Base de données sql server .
- Base de données acces.
- Document XML.
- Fichier plat.

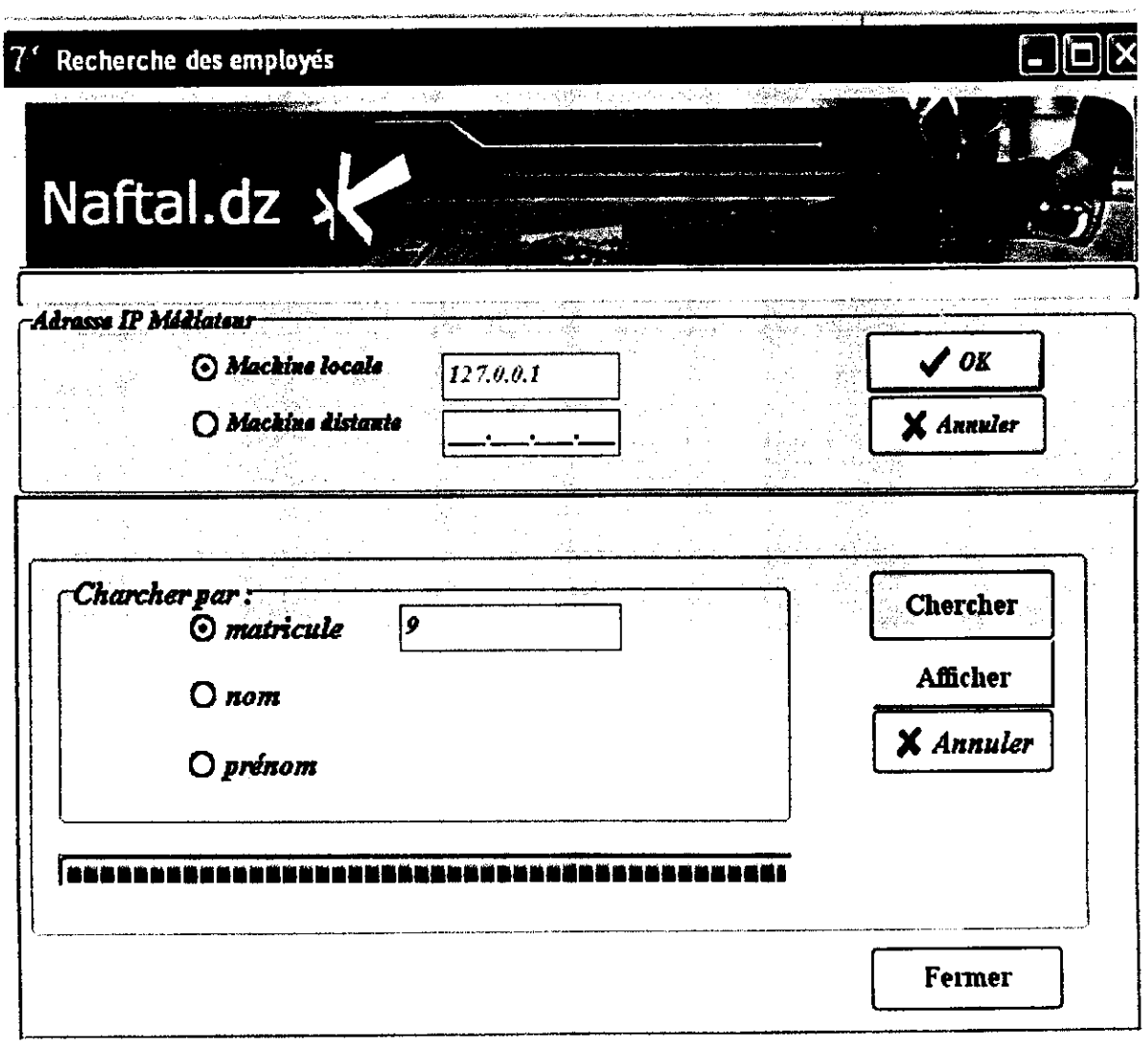


Figure 16 : La fenêtre de l'application cliente.

Le client pour qu'il peut connecter l'application médiateur il doit spécifier son adresse IP, pour ce la nous avons deux choix:

- *machine locale:* si l'application cliente se trouve dans la même machine avec le médiateur.
- *Machine distante:* si non.

Le bouton *ok* donne la main au client d'utiliser le service de recherche information sur les employés et le service d'affichage de résultat.

La recherche d'employé se fait de trois manières:

- par matricule
- par nom
- par prénom

Le bouton "Chercher" permet de formuler la requête en XML et l'envoyer au médiateur.

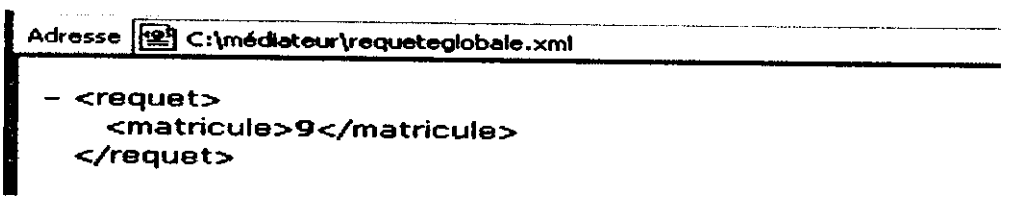


Figure 17: requête XML formulée par le client.

Le bouton "Afficher" permet d'afficher le résultat de la requête en XML.

Si l'employé n'existe pas alors le résultat suivant sera affiché.

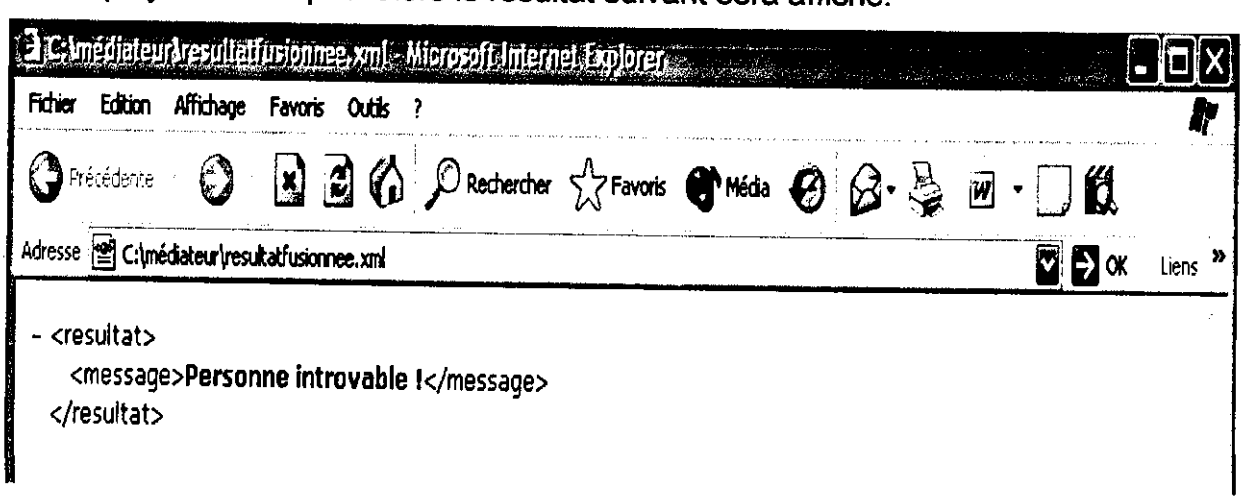


Figure 18: le résultat en XML si l'employé n'existe pas.

Si l'employé existe le résultat suivant est affiché.

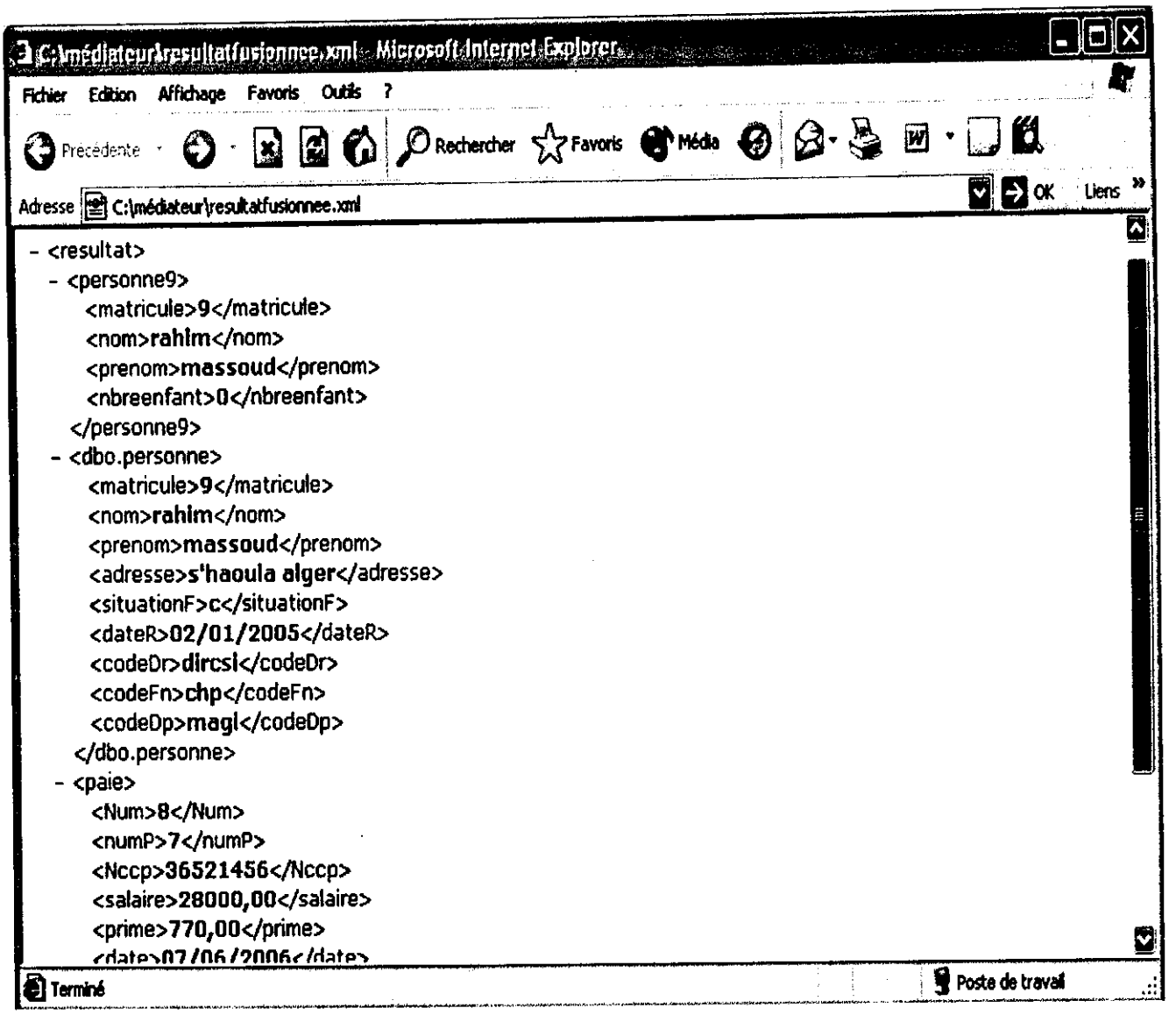


Figure 19: le résultat en XML si l'employé existe.

Conclusion:

Dans ce chapitre nous avons décrit les fonctionnalités réalisées par le prototype d'interrogation, le détail de chaque module de système et la communication entre les différents modules de notre système.

Conclusion et perspectives

Conclusion et perspectives :

Notre travail consiste à concevoir un système permettant l'intégration simultanée et transparente de plusieurs sources de données distribuées et hétérogènes. Il consiste à la fois à concevoir une architecture de ce système et de réaliser un prototype de cette architecture.

Nous avons vu que les SIFs représentent une bonne solution pour ce genre de problèmes dans la mesure où ils permettent l'intégration des sources distribués et hétérogènes. Plus particulièrement, les bases de données fédérées sont les mieux adaptées au problème de l'intégration des données.

La fusion et l'échange de données nécessitent un format pivot et XML est évidemment le meilleur choix pour ce format à cause de son ouverture, son indépendance et son extensibilité.

N'étant qu'une application distribuée, il nous fallait une bonne technologie de développement de ce genre d'applications pour la réalisation de notre système. Le choix s'est fait entre DCOM, CORBA, Java RMI, SOAP et .NET Remoting, et nous avons opté pour SOAP, qui est un protocole plus simple, plus souple, basé sur le standard d'échange XML.

La mise en oeuvre de notre prototype de système a été faite sur plusieurs sources de données, structurée les bases de données relationnelle : SQL, paradoxe et accès, semi structurée fichier XML et non structurée fichier plat.

Nous proposons comme perspective:

- Améliorer les performances du système comme par exemple le temps de réponse et d'implémenter notre prototype dans le système Linux en utilisant kylix qui est un environnement de développement à la fois Delphi et Borland C++Builder pour Linux.

- Enrichi les algorithmes de décomposition et de fusion de manière à prendre en compte n'importe quel type de requête, l'intérêt qu'il faut porter au développement dé l'application cliente.

Finalement nous espérons que notre travail sera pris en considération dans la société NAFTAL pour résoudre les problèmes d'hétérogénéités et la distribution ainsi que il donne un pas d'avance à l'étudiant spécialement et l'enseignement supérieur en générale.

Références bibliographiques

[Agora 2000] http: www.agora.com.

[AUSDERAU 2004] PATRIK AUSDERAU (*Web services et Java*) 2004.

[Bienvenu 2003] Bienvenu Vincent (*les objets distribués: comparaison CORBA et .NET*) 2003.

[Busse et al 1999] S Busse, R.D. Kutsche, U. Leser, H. Weber (*Federated information systems: Concepts, Terminology and Architectures*) université de technologie Berlin,1999.

[DANG NGOC 2003] thèse de doctorat (fédération des données semi structurées en XML) université de Versailles, 2003.

[Gardarin 96] George GARDARIN (le client /serveur) Eyrolles 1996.

[kevin w 2001] kevin w (XML et les base de données) Eyrolles 2001.

[Lomme et Martin 2004] S. Lomme et O. Martin. (*Les services web*) 2001.

[Meylan 2003] Eddy Meylan (*base de données réparties*) University of applied sciences of western Switzerland, 2003.

[MS et CR 2003] Mohand-Saïd Hacid et Chantal Reynaud (*L'intégration de sources de données*) Université Claude Bernard Lyon 1, 2003.

www. mshacid@bat710.univ-lyon.fr.

[Rey et Varin2001] Rey Brice, Varin Valery (*les technologies distribués:CORBA, Java RIM, SOAP*) 2001.

[Solar et Doucet 2001] Genoveva Vargas Solar, Anne Doucet (*médiation de données: solution et problèmes ouverts*) 2001.

[w3] http://www.w3.org/XML/.

[Xavier Baril2003] thèse de doctorat 2003. (Un *modèle de vues pour l'intégration de sources de données XML : VIMIX*) université de Montpellier II.

[X Baril2003] thèse de doctorat (Un modèle de vues pour l'intégration de sources de données XML : VIMIX) Université Montpellier II, 2003.

[ZERDAZI 2003] mémoire DEA (représentation de schémas de bases de données hétérogènes sous forme de métaschémas XML) université de marne la vallée, 2003.