Accès transparent et sécurisé à des données largement distribuées

Béatrice Finance

Université de Versailles Saint-Quentin-en-Yvelines

Accès transparent et sécurisé à des données largement distribuées

Béatrice FINANCE

rapport scientifique pour l'obtention de

habilitation à diriger des recherches en informatique

Soutenue le 11 Juillet 2006

JURY

Anne Doucet	Professeur à l'Université de Paris 6	Rapporteur
Jerry Kiernan	IBM Almaden, senior research	Rapporteur
Michel Scholl	Professeur au CNAM Paris	Rapporteur
Christine Collet	Professeur à l'INPG de Grenoble	Examinateur
Georges Gardarin	Professeur à l'Université de Versailles - St Quentin en Yvelines	Examinateur
Philippe Pucheral	Professeur à l'Université de Versailles – St Quentin en Yvelines	Examinateur

Remerciements

Je remercie tout d'abord Georges Gardarin sans qui rien n'aurait jamais commencé. J'ai, auprès de lui, fait mes premiers pas de chercheur. Il a su me guider, m'encourager et me faire confiance dans la gestion de ma recherche. Son enthousiasme et son énergie dans le montage et la réalisation de projets européens m'ont toujours inspiré. À côté de lui, j'ai beaucoup appris. Qu'il trouve ici toute ma reconnaissance et ma joie d'avoir travaillé avec lui.

Je remercie Philippe Pucheral pour m'avoir accueilli au sein de son équipe, d'abord au PRiSM, puis en tant que collaboratrice extérieure au sein du projet SMIS à l'INRIA Rocquencourt. Je le remercie particulièrement d'avoir accepté d'être le tuteur de cette HDR. Nos collaborations scientifiques sur le domaine de la sécurité ont été riches d'enseignement. Son esprit perspicace attire l'attention de tous.

Je remercie profondément Anne Doucet, Jerry Kiernan et Michel Scholl d'avoir accepté de rapporter cette HDR.

Je remercie tout particulièrement Christine Collet pour son amitié, nos échanges en tout genre et ses encouragements sincères tout au long de ces dernières années. Elle me fait un grand plaisir en participant à ce jury.

Je ne peux finir sans avoir une pensée pour mes collègues et amis qui ont contribué à faire de ces années une expérience humaine chaleureuse. Ils sont nombreux et je ne peux les citer tous, mais je pense particulièrement :

À Elisabeth, Zoubida, Annick, Chantal, Leila et Soraya, pour leur amitié.

À Véronique, Thierry, Saida et Jérôme pour les bons moments passés ensemble et leur travail qui a permis et soutenu la qualité des résultats scientifiques.

Enfin je remercie ma famille et mes amis proches qui dans l'ombre m'ont soutenu et encouragé. Qu'ils trouvent ici le témoignage de toute mon affection.

Table des Matières

Listes des Figures

Préambule

Ce manuscrit présente mes principaux travaux de recherche effectués depuis ma prise de fonctions en qualité de Maître de Conférences à l'Université de Versailles St Quentin en Yvelines en 1992. Ces travaux ont été réalisés au sein du thème "Systèmes de Bases de Données" du Laboratoire PRiSM, tout d'abord sous la direction de Georges Gardarin (de 1988 à 2000), puis sous la direction de Philippe Pucheral à partir de 2000 lors de la création de l'équipe Mobilité, Sécurité et Disponibilité de l'Information (MSDI). En 2002, les membres de l'équipe MSDI ont créé le projet SMIS (Secure & Mobile Information System) à l'INRIA Rocquencourt. J'ai donc rejoint cette équipe tout d'abord en tant que collaboratrice extérieure, puis en délégation en septembre 2005.

Mes orientations scientifiques se sont élargies et diversifiées au cours des années, d'une part pour suivre la stratégie des équipes auxquelles j'ai participé mais aussi pour des raisons d'ouverture et de collaboration scientifique. Mes activités de recherche tournent principalement autour de la définition d'architectures de système pour la gestion de données, de langages d'interrogation, de techniques d'évaluation et d'optimisation de ces langages, et de mécanismes de protection de la confidentialité des données.

Dans ce manuscrit, j'ai choisi de décrire uniquement les travaux qui ont un lien avec l'accès transparent et sécurisé à des données largement distribuées. Je n'ai donc pas décrit les travaux menés juste après ma thèse sur l'extensibilité du processus d'optimisation et les techniques d'optimisation de requêtes dans les systèmes de gestion de base de données relationnelles étendus aux objets et à la déduction. Un petit résumé de ces travaux se trouve dans mon curriculum vitae. Par la suite, j'ai continué tout naturellement à faire évoluer cette thématique de recherche en proposant de nouvelles techniques adaptées à des contextes différents, cela tout en gardant la même culture et les mêmes fondements.

Chapitre I – Introduction

1 Contexte des travaux

Les récents progrès réalisés en matière de communication (réseaux hauts débits, normalisation des protocoles et des architectures à objets répartis, explosion de l'internet) permettent aujourd'hui d'envisager la construction de systèmes d'information de grande envergure au cœur desquels se trouvent de gros volumes d'information multiforme. Les données sont de natures diverses : structurées (n-uplets, ensemble, listes, objets) ou semi-structurées (XML, HTML) et multimédias. Elles sont stockées dans des systèmes relationnels, orientés objets ou XML, des serveurs de fichiers ou d'objets, et sont réparties à l'échelle planétaire.

Pour gérer ces systèmes d'information de grande envergure, il faut des infrastructures permettant d'accéder à de multiples sources de données ou à des programmes préexistants, autonomes et potentiellement hétérogènes. Les domaines d'application couvrent aussi bien les systèmes d'information scientifiques (génétique, astronomie, environnement, etc.), que les systèmes d'informations industriels et tertiaires (systèmes de supervision et de contrôle pour le transport et l'énergie, systèmes décisionnels, bases de données des opérateurs télécoms, etc.).

Offrir un accès transparent et sécurisé à un ensemble de ressources passe par la définition de logiciels de médiation[1] (i.e, *intergiciel ou middleware)* qui rendent la complexité de l'architecture sous-jacente transparente à l'utilisateur en offrant des *facilités* de conception, d'intégration, d'interrogation et d'administration permettant le partage de données et de programmes d'une manière fiable et efficace. Un très gros effort a été mené ces vingt dernières années pour aider à la mise en œuvre de ces logiciels de médiation, que cela soit par la communauté bases de données ou par la communauté systèmes distribués.

La communauté bases de données approche l'accès transparent aux données largement réparties via des logiciels de médiation tournés vers les problèmes d'interopérabilité entre bases de données. Ces systèmes sont apparus dans les années 80 pour suppléer les systèmes de bases de données distribuées [147], jugés trop rigides car ne laissant pas assez d'autonomie à leurs composants locaux et nécessitant l'installation du Système de Gestion de Bases de Données (SGBD) de base sur tous les sites. Ils ont été développés pour répondre à trois besoins : la distribution, l'hétérogénéité et l'autonomie des sources d'informations locales. Ce type de systèmes permet de faire coopérer un ensemble de bases de données locales pour répondre à

1 *Définition* : en informatique, le middleware désigne les logiciels servant d'intermédiaire entre d'autres logiciels.

certains besoins comme l'intégration de données hétérogènes, l'interrogation, la gestion de transactions de mises à jour, etc. Déjà en 1994, Hurson [86] en dénombrait une cinquantaine. Ce nombre n'a cessé de croître notamment avec l'évolution des bases de données objets et semi-structurées.

Une des caractéristiques forte et essentielle de l'approche bases de données est de proposer, le plus souvent, une vision intégrée et cohérente des données qui passe par la définition et l'interrogation de vues (i.e, données virtuelles) à partir de langages de haut niveau de type SQL[46], OQL[32], XPath [155] ou XQuery [156]. Certains de ces systèmes mettent aussi l'accent sur les mises à jour et le partage de données ainsi que sur la navigation. Pour ce qui concerne l'accès sécurisé aux données, cette approche se repose sur les techniques classiques bases de données de contrôle d'accès (i.e, définition de vues).

La communauté systèmes distribués approche l'accès transparent aux données via les problèmes d'interopérabilité entre applications, tout particulièrement dans des environnements ouverts et largement répartis. Cette approche s'appuie sur l'utilisation de catalogues (DNS, CORBA Naming Service [117]) et d'annuaires (LDAP [148], JNDI [140], UDDI [146]). Elle permet de localiser rapidement, simplement et en toute sécurité des ressources par l'intermédiaire de langages d'interrogation simples et déclaratifs accessibles par des non-spécialistes. Elle permet aux applications, après cette phase de localisation, d'accéder et de mettre à jour les ressources au travers de bus distribués comme ceux définis par l'Object Management Group (CORBA[119], CCM[41]), par Microsoft (COM/DCOM[39], .Net[1]), pour l'environnement Java (Entreprise JavaBeans[61]), et plus récemment l'approche par services Web proposée par le W3C[151].

Une des caractéristiques essentielle de cette approche est de pouvoir gérer des ressources dont la nature et les propriétés rendent le catalogage centralisé impossible ou inadapté. Les catalogues ou annuaires sont répliqués et distribués pour permettre le passage à l'échelle, c'est-à-dire la montée en charge du nombre de connexions utilisateurs et la gestion d'un très grand nombre de ressources réparties au niveau planétaire. Une autre des caractéristiques est l'accès sécurisé aux ressources par le biais de services spécialisés tels que CORBA Security [116], Globus Security [139], LDAP security [55], SAML [131], XACML [115]. Ces services comprennent des mécanismes d'authentification, d'autorisation, de chiffrement, d'intégrité des données, de délégation, de non-répudiation et d'audit associés.

Sur cette approche, et dès 1996, se développent des infrastructures pour aider au développement d'applications et au partage de données sur la grille. On peut citer les projets phares dans ce domaine qui sont GLOBUS [73] pour le calcul distribué et DataGrid [138] pour le partage de données à large échelle. Plus récemment l'architecture OGSA (Open Grid Services Architecture) [142] a été proposée par l'alliance GLOBUS fédérant les projets GLOBUS et DataGrid. La caractéristique de OGSA est de revisiter les infrastructures pour la grille afin de s'appuyer sur les concepts et technologies services Web.

Même si les communautés bases de données et systèmes distribués adressent toutes deux le problème de l'accès transparent et sécurisé à des ressources largement distribuées il n'en demeure pas moins que les solutions proposées sont souvent très différentes d'un système à un

autre. Il est important aujourd'hui de comprendre la variabilité des solutions en termes de fonctionnalités, de design, d'algorithmes et d'architectures afin d'identifier les différentes dimensions du problème. Tout au long de ma carrière, je me suis intéressée aux aspects pluridisciplinaires de l'accès transparent aux données. J'ai regardé très tôt les systèmes de médiation de données pour l'interopérabilité de bases de données relationnelles et objets, puis plus récemment de bases XML. J'ai conçu et réalisé des architectures de systèmes de médiation de données aux fonctionnalités variées. Je me suis également intéressée au domaine des annuaires et aux plates-formes à objets distribués. J'ai étudié les limites des approches existantes d'un point de vue modèle de données, langage et modèle d'exécution. J'ai étudié le problème de l'accès sécurisé aux données et plus particulièrement le problème de la gestion de la confidentialité et des modèles de contrôle d'accès pour XML. En effet, la confidentialité des données est devenue un enjeu majeur pour la majorité des applications qu'il s'agisse de gestion de données personnelles, de commerce électronique, de systèmes d'information en ligne, d'intelligence ambiante ou plus classiquement de préservation de secrets industriels ou scientifiques.

Cette approche pluridisciplinaire a été riche d'expérience et d'enseignements, elle m'a permis de créer et de participer à de nombreux projets de recherche, permettant de définir le cadre de plusieurs thèses et menant à de nombreuses publications. Mais avant d'en reprendre les détails et de conclure cette introduction avec le plan de ce manuscrit, j'aimerais souligner que la majorité des recherches relatées dans ce manuscrit sont le fruit de collaborations multiples. J'en profite pour exprimer ici toute ma reconnaissance à tous ceux et celles qui ont apporté leur concours à ces travaux et pour cette raison, j'utiliserai dans la suite du document la première personne du pluriel.

2 Contributions

De 1993 à 1997, nous avons conçu et réalisé le système IRO-DB pour l'accès transparent à des bases de données objets et relationnelles. IRO-DB répond aux besoins d'un nombre croissant d'applications comme les applications coopératives (CAD/CAM), le multimédia, ou les applications géographiques. Ces dernières gèrent des données de plus en plus complexes en provenance d'une très large variété de sources d'informations et pour lesquelles il y a un besoin croissant de faire interopérer des bases de données existantes, autonomes et hétérogènes [127]. Ces bases doivent partager leurs données, autoriser l'interopérabilité avec les autres SGBDs, sans perdre le contrôle sur leurs propres données et sans avoir à réagencer les applications existantes. Ces différentes applications ont également des besoins en performance. Les temps de réponses aux requêtes ne doivent pas êtres trop pénalisants. L'accès aux informations doit être rapide. C'est pourquoi un effort important a été apporté aux performances du système.

La particularité du système IRO-DB [63, 78, 77] est qu'il tire parti des nombreuses recherches et apports des Systèmes de Gestion de Bases de Données Objets (SGBDO). Il s'appuie sur le modèle de l'ODMG (Object Data Management Group) [32,31] et offre un système complet pour l'accès transparent et sécurisé aux données. Il propose des mécanismes d'intégration de schémas et de données semi-automatique pour la définition de vues [26], des

techniques d'évaluation et d'optimisation de requêtes [71], un gestionnaire d'objets répartis et un gestionnaire de transactions [137] et de droits d'accès [60].

Une des originalités du système IRO-DB est qu'il n'offre pas uniquement un accès déclaratif aux objets, via le langage OQL [32], mais il permet également un couplage fort avec le langage de programmation OML/C++ [32]. Il offre donc un mode d'accès transparent par navigation très utile aux applications. Son modèle d'intégration orienté objets permet une intégration d'objets et de méthodes, une intégration forte et élégante des objets applicatifs avec ceux résultants des bases de données distantes et une propagation des mises à jour. Une autre originalité du système IRO-DB repose sur sa capacité à stocker durant une transaction, des objets venant de diverses bases de données locales dans un gérant d'objets. En fait, le gérant d'objets du système IRO-DB joue le rôle de cache d'objets se trouvant généralement dans les SGBDO. La réplication d'objets dans le gérant d'objets améliore les performances d'une application. En effet, après un certain temps, IRO-DB se comporte comme un système centralisé. Cette hypothèse est surtout intéressante si l'on considère des applications (programmes) complexes, ayant un fort besoin d'intégration de données, avec des transactions longues manipulant des objets complexes et nécessitant un mécanisme de navigation efficace parmi les objets.

Au-delà de nos contributions liées à la définition de l'architecture et du modèle d'intégration de données, nous nous sommes particulièrement intéressés à l'étude des interrogations multi-bases qui sont difficiles à traiter du fait de l'hétérogénéité des sources, de la complexité des requêtes, de la multiplicité des sites, du manque de méta-connaissances sur les sources de données, de l'absence de modèle de coût global. Dans IRO-DB, nous avons proposé un optimiseur qui décompose les requêtes multi-bases en un sous ensemble de sous-requêtes optimales. Cet optimiseur tire profit du modèle de données sous-jacent ODMG/OQL, du processus d'intégration de données, avec son mécanisme d'instanciation partielle des vues intégrées et du cache situé au niveau du client [71].

Les travaux publiés sur les aspects architecturaux ont été réalisés avec Georges Gardarin, Peter Fankhauser, Wolfganf Klas et Antonis Ranfos. Les travaux autour des problèmes d'optimisation ont servi de cadre à la thèse de Véronique Smahi [99].

De 2002 à 2004, nous nous sommes intéressés à la définition et à la réalisation d'une infrastructure de médiation MEDIAGRID [37] pour l'accès transparent à des sources de données XML réparties sur la grille. En effet, il est aujourd'hui important de pouvoir gérer des données faiblement structurées produites par des applications et stockées soit dans des bases de données, soit le plus souvent dans des fichiers HTML ou XML. L'objectif de MEDIAGRID consiste d'une part à définir des composants adaptables et réutilisables, et d'autre part à étudier les techniques s'adaptant à la forte dynamicité de l'internet (i.e, un nombre croissant de sources disponibles, l'évaluation de requêtes complexes en cas d'indisponibilité de sources et/ou de trafic réseau inconstant). Ces travaux ont été réalisés avec les membres du projet MEDIAGRID.

Les contributions originales du projet se sont focalisées sur l'automatisation de la construction de requêtes de médiation dans un contexte XML et sur les techniques d'évaluation de requêtes. Le fait d'automatiser le processus d'intégration permet de s'adapter à la forte

dynamicité de l'environnement (i.e, ajout, retrait de sources, mise à jour des schémas des sources). La définition d'un canevas (i.e, ensemble de classes abstraites réutilisables) pour la gestion de requêtes, appelé QBF (Query Broker Framework) permet d'adapter l'évaluation de requêtes XQuery aux contraintes spécifiques de l'environnement d'exécution. Le défi consiste à offrir des algorithmes, des outils, des modèles pour choisir les "meilleurs" plans d'exécution d'une requête, tout en tenant compte des besoins des utilisateurs en offrant un mode interactif, pour autoriser des résultats partiels et pour offrir la possibilité de raffiner le plan d'exécution en cours d'exécution des requêtes.

De 1997 à 2002, nous nous sommes intéressés au problème de l'accès transparent à des ressources largement distribuées via des annuaires. Nos travaux de recherche ont consisté à étudier les limites des annuaires existants. Nous nous sommes particulièrement intéressés aux problèmes des environnements télécoms. L'interrogation dans ces environnements pose des problèmes bien spécifiques [66,68] : (1) des principes de nommage complètement différents de ceux que l'on utilise dans les bases de données, (2) un besoin de gérer des objets pas forcément gérés dans des SGBD, (3) un très grand nombre d'objets de granularité fine et des contraintes temps réel, (4) un besoin de gérer à la fois la distribution et la fédération d'objets. Trois technologies différentes peuvent coexister et coopérer au cœur du système d'information [48] : des SGBD, des serveurs LDAP [148] et des plates-formes à objets distribués. Bien que nos recherches aient été motivées par le domaine des télécoms, les solutions proposées sont génériques et peuvent servir à d'autres domaines d'applications.

Pour répondre aux besoins des applications récentes comme les applications de gestion de terminaux mobiles, il est nécessaire d'offrir des annuaires flexibles s'adaptant à la forte dynamicité et aux contraintes temps-réel des objets qu'ils gèrent, c'est-à-dire fréquemment mis à jour. Nous avons donc proposé un service d'annuaires [53,50] aux fonctionnalités avancées capables : (i) de manipuler des données riches englobant des valeurs mais également des programmes (i.e. objets CORBA), (ii) de conserver la flexibilité inhérente à LDAP et (iii) de gérer de façon efficace et transparente la distribution des objets référencés. Nous avons défini le langage de requêtes DQL (Directory Query Language) permettant de gérer (i.e, créer, modifier et supprimer) et d'interroger des objets complexes sur un réseau [49]. DQL étend le standard d'annuaires LDAP avec des expressions de chemin. Afin de pouvoir faciliter l'accès transparent aux données, nous avons également réalisé une interface permettant de naviguer à travers les objets de l'annuaire. Le côté original de la solution repose sur le modèle d'exécution de DQL et sa capacité à retourner des réponses incomplètes ou à réaliser des évaluations partielles de requêtes. Ces points sont particulièrement importants lorsque l'on interroge un très grand nombre de sites ; ces sites pouvant êtres inaccessibles car saturés ou en panne, voire momentanément déconnectés [52].

Une dernière contribution adresse le problème de l'intégration des technologies LDAP et bases de données, à laquelle l'IETF (Internet Engineering Task Force) s'intéresse [12]. Un aspect de cette intégration concerne le partage de données communes entre ces différents systèmes, comme les données d'identification par exemple. Pour assurer ce partage, nous avons proposé d'offrir un double accès LDAP-SGBD [51] au-dessus d'un SGBD relationnel.

Ces travaux ont servi de cadre à la thèse de Thierry Delot [53]. Certaines de nos

publications ont été réalisées avec des partenaires de France Telecom R&D.

À partir de 2002, nous nous sommes intéressés à la confidentialité des données. La défiance envers la gestion actuelle des données confidentielles est vue, selon une étude IBM-Harris, comme le principal frein au développement de nouvelles applications sur l'Internet, et donc un frein au partage et à l'interrogation de ressources largement distribuées. La préservation de la confidentialité est une tâche gigantesque et touche de nombreuses thématiques : chiffrement des données et des communications, détection d'intrusions, contrôle de droits d'accès, anonymisation des données et des actions, fouille de données préservant l'intimité des données, etc. Nos principales contributions ont porté sur les modèles de contrôles d'accès pour XML.

Beaucoup de travaux de recherche ont été proposés dans la littérature pour répondre aux problèmes de la confidentialité des documents XML[2]. Certains adressent la caractérisation du modèle de contrôle d'accès c'est-à-dire la granularité de protection (i.e, éléments, attribut, DTD, XMLSchema), la modélisation des sujets (i.e, utilisateur, groupe, rôle) et des modes d'accès (i.e, lecture, mis à jour). D'autres ont étudié (1) les algorithmes permettant d'implanter ce contrôle, (2) les canaux de communication et de distribution de l'information (i.e, dissémination sélective en mode Push ou Pull) et (3) la non corruption du modèle de contrôle d'accès par des techniques de chiffrement ou d'environnements d'exécution sécurisés. Cependant, tous ces travaux partagent le même point de vue; ils ont choisi de focaliser le contrôle d'accès sur les nœuds d'un document XML (i.e, attributs et éléments). Les relations de parenté et de fraternité entre les nœuds ne sont pas considérées comme des éléments de première classe dans leur modèle. Cela pose des problèmes majeurs qui vont à l'encontre de deux principes de base édictés par les directives et lois mises en place par les gouvernements pour protéger les informations personnelles. Le premier est le principe de finalité (ou "need-to-know"), il limite l'accès à l'information, seule l'information strictement utile à une tâche doit pouvoir être accédée. Le second est le principe de consentement, il empêche la divulgation d'information sans le consentement explicite du propriétaire de la donnée.

Plus que jamais, il est nécessaire de définir des modèles de contrôles d'accès qui permettent de traduire les lois en pratique. Afin de faire un pas dans cette direction, nous avons proposé un nouveau modèle de contrôle d'accès pour des documents XML [72]. Ce modèle intègre les concepts de finalité et de consentement [69]. Nous avons tout d'abord caractérisé les dimensions du problème et proposé un modèle basé sur des règles permettant de protéger à un nœud vis-à-vis de ses ancêtres et de certains de ses frères. Ce modèle a un fort pouvoir d'expression et garde un fort degré de concision. L'ensemble des règles d'autorisations est sûr, c'est-à-dire qu'il existe un algorithme déterministe et compréhensible par un humain qui permet de calculer la vue autorisée à partir du document initial. Pour finir, il est facile de faire évoluer les droits d'accès. L'intérêt de l'approche est qu'elle est compatible avec les modèles existants qui se focalisent uniquement sur la protection des nœuds, nous avons étendu ces modèles afin d'intégrer la gestion des droits sur les liens entre les nœuds. Ces travaux ont servi de cadre à la thèse de Saida Medjdoub [108] et ont été réalisés avec Philippe Pucheral.

[2] Les références bibliographiques à ces travaux seront données dans le chapitre sur la confidentialité des données.

3 Plan

Dans la suite de ce manuscrit, nous avons choisi de décrire plus en détail certains des travaux et contributions présentés ci-dessus. Dans un premier chapitre, nous adressons la problématique associée à la définition d'un système de médiation pour l'accès transparent à de multiples sources de données qu'elles soient de type objet, relationnel ou XML. Nous présentons dans le détail nos contributions majeures dans ce domaine dont beaucoup ont été menées dans le cadre du système IRO-DB. Dans un second chapitre, nous décrivons la problématique associée à l'accès transparent aux objets dans les plates-formes à objets distribués via des annuaires. Dans un troisième chapitre, nous décrivons la problématique associée à la définition d'un modèle de contrôle d'accès pour XML, les limites des modèles existants et nos contributions.

Dans un dernier chapitre, nous détaillons nos perspectives et directions de recherche, à la lueur de notre expérience pluridisciplinaire, autour du constat de l'adoption croissante de XML comme fondation technologique pour l'accès transparent et sécurisé aux données largement distribuées. En particulier, nous reviendrons sur l'intérêt d'une fondation technologique autour d'XML pour l'accès transparent et sécurisé aux données largement distribuées. Nous pensons que des capacités d'intégration et de partage de documents XML, ainsi qu'un accès transparent aux données soit de façon déclarative, soit par navigation, sont cruciaux pour des solutions communes pour la mise en place d'un système de médiation complet aux fonctionnalités avancées, le développement d'applications réparties sur le Web et la construction d'annuaires aux fonctionnalités avancées. Les capacités d'exécution doivent être flexibles et efficaces, par exemple les techniques d'évaluation de requêtes doivent s'adapter à la nature virtuelle et répartie des documents et offrir des résultats et des évaluations partiels de requêtes, les techniques de cache doivent permettre une gestion hybride et performante des données virtuelles, matérialisées ou non. Par ailleurs, il est important d'intégrer les modèles de contrôle d'accès XML existants qui ne s'appuient pas sur des techniques classiques BD et la définition de vues XQuery, et de protéger l'accès non seulement de façon déclarative mais aussi par navigation. Nous pensons que ces réflexions, fondées sur nos contributions et travaux antérieurs, confirment l'intérêt d'un axe de recherche pluridisciplinaire qui accélérerait la synergie naissante autour de XML dans le domaine de l'accès transparent et sécurisé aux données largement distribuées.

Chapitre II – Systèmes de médiation de données

Ce chapitre présente nos travaux de recherche dans le domaine des systèmes de médiation pour résoudre les problèmes d'interopérabilité entre bases de données. Ces travaux recouvrent deux périodes distinctes de recherche. La première s'est déroulée entre 1993 et 1997. Elle s'est intéressée au problème de l'interopérabilité entre bases de données relationnelles et objet. La seconde s'est déroulée entre 2000 et 2002 et s'est intéressée au problème de l'interopérabilité de bases de données ou de fichiers XML dans le contexte de l'internet.

1 Introduction

La fédération de données permet d'accéder de manière intégrée à des données disséminées sur différents calculateurs et d'interroger en temps réel de vastes bases de données virtuelles intégrant potentiellement tout ou partie des données de l'entreprise. Au-delà de l'accès par des requêtes unifiées, les systèmes de fédération de données cherchent à retrouver rapidement et facilement à l'information disséminée sur un sujet donné. Avec les grands réseaux internationaux tels internet et dans les intranets d'entreprise, le besoin de fédération de données hétérogènes est immense.

Les systèmes de gestion intégrant des sources de données hétérogènes en assurant la transparence à la distribution et à l'hétérogénéité sont de deux types [76] : (i) bases de données virtuelles composées à partir de sources hétérogènes; (ii) plate-forme d'intégration ouverte basée sur un entrepôt de données. La première est connue sous le nom de bases de données hétérogènes intégrées ou encore de bases de données fédérées. La seconde s'articule autour d'un entrepôt de données (*datawarehouse*). Nos recherches ont porté sur les bases de données virtuelles.

Alors que les bases réparties intègrent plusieurs bases souvent homogènes et sont gérées par un SGBD réparti, les bases fédérées sont gérées par un **système de médiation de données** et permettent d'accéder à plusieurs sources de données hétérogènes (e.g, fichiers structurés, bases de données, documents textuels, etc.) comme une seule via une vue commune (un modèle commun) et un langage de requête commun. Le terme **SGBD fédéré** est aussi employé, mais il est plus archaïque et fait plus penser à un système monolithique, alors que les systèmes de médiation sont distribués sur des architectures Web multi-tiers. Un système de médiation de données désigne un ensemble de modules logiciels permettant un accès unifié à des sources de données hétérogènes sur le Web en Internet ou Intranet.

Un système de médiation reçoit des requêtes référençant des objets d'une base de données fédérée. Il assure la décomposition des requêtes distribuées en sous-requêtes locales envoyées à chaque site. La décomposition prend en compte les règles de localisation, notamment l'existence

de méta-données pertinentes à la requête. Lors de l'évaluation globale de la requête, il faut à la fois minimiser les transferts de données et optimiser les temps de calcul, en utilisant au mieux le parallélisme notamment. L'optimisation et l'évaluation des requêtes réparties sont donc des fonctions essentielles. Elles visent à élaborer des plans d'exécution proches de l'optimal pour les requêtes et à les exécuter de manière distribuée performante. Pour ce qui concerne les mises à jour, le moteur du système de médiation doit bien sûr les router vers le ou les sites concernés, mais il doit aussi assurer la gestion des transactions réparties pour les cas de mises à jour. Ceci inclut la vérification des règles d'intégrité, le contrôle des accès concurrents, et surtout la gestion de l'atomicité des transactions distribuées. La coordination des mises à jour est alors difficile et conduira à introduire des transactions longues.

Un très grand nombre de systèmes de médiation existent dans la littérature. En 1994, Hurson [86] en dénombrait déjà une cinquantaine et ce nombre n'a cessé de croître notamment avec l'évolution des bases de données orientées objet et semi-structurées. Tous les systèmes de médiation partagent la même architecture de référence. À partir de cette architecture, différentes options sont possibles. Un choix important est celui du modèle pivot et du langage pivot. Des modèles relationnels, objets et semi-structurés ont été utilisés pour fédérer les données et leurs langages respectifs SQL [46], OQL [32] ou XQuery [156] ont été proposés pour l'interrogation des données intégrées.

Une caractéristique importante des systèmes de médiation est le degré d'hétérogénéité supporté, notamment les possibilités de réconciliations sémantiques offertes entre données hétérogènes. Une autre caractéristique importante concerne l'ensemble des fonctionnalités et outils qu'ils offrent pour concevoir et maintenir de grandes fédérations de base de données hétérogènes et pour aider au développement d'applications. Pour finir, la puissance des mécanismes internes d'exécution (i.e, des techniques d'optimisation et de caches) doivent permettre d'obtenir de bonnes performances.

Au cours de nos recherches, nous avons travaillé à la définition et à la réalisation de systèmes de médiation. Dans un premier temps, nous avons étudié les problèmes liés à la fédération de bases de données objet et relationnelles et défini le système IRO-DB. Dans un second temps, nous avons proposé le système MEDIAGRID pour la fédération de sources de données XML. Afin de mieux comprendre nos contributions vis-à-vis de l'état de l'art, nous décrivons dans une première section l'architecture de référence d'un système de médiation et les composants fonctionnels associés. Ensuite nous caractérisons l'ensemble des objectifs et motivations qui nous ont conduit à la définition de ces nouveaux systèmes. Dans une seconde section, nous détaillons nos principales contributions. Puis, nous concluons.

2 État de l'art & Contributions

Tous les systèmes de médiation partagent la même architecture de référence. Cette architecture est dérivée de la proposition du groupe I3 (*Information Integration Interoperability*) de DARPA[3] qui a été élaborée en 1995. La Figure 1 précise chacun des niveaux et détaille les

[3] organisme de recherche de l'armée américaine.

modules qui la composent.

Tout d'abord, une base de données fédérée est décrite par différents niveaux de schémas. Chaque source locale possède un schéma géré par le système local, appelé **schéma local**, décrivant les données d'une source de données locale gérée par le système local. Lors de la constitution d'une base de données fédérée, chaque base locale rend visible une partie de la base aux sites clients. Cette partie doit être décrite dans le modèle pivot servant de modèle d'échange au niveau du médiateur. Cette description est appelée **schéma exporté**, décrivant les données exportées par un site local pour une base fédérée. Vu d'un serveur de médiation, un schéma exporté par un serveur de données devient un **schéma importé**. Il est en général toujours exprimé dans le modèle pivot utilisé pour les échanges.

L'ensemble des schémas exportés par un site de médiation peut être intégré dans un schéma unique exprimé dans le modèle pivot et décrivant globalement la base fédérée. Un tel schéma est appelé **schéma global**. La définition du schéma global, qui offre une vue uniforme des données, peut se faire selon deux approches différentes. La première, appelée global-as-view (GaV) est assez intuitive et consiste à définir le schéma global comme une vue sur les schémas locaux. La seconde, appelée local-as-view (LaV) consiste à procéder à l'inverse, c'est-à-dire à définir les schémas locaux comme des vues sur le schéma global. Dans un cas comme dans l'autre la transformation de questions s'exprime à l'aide de vues.

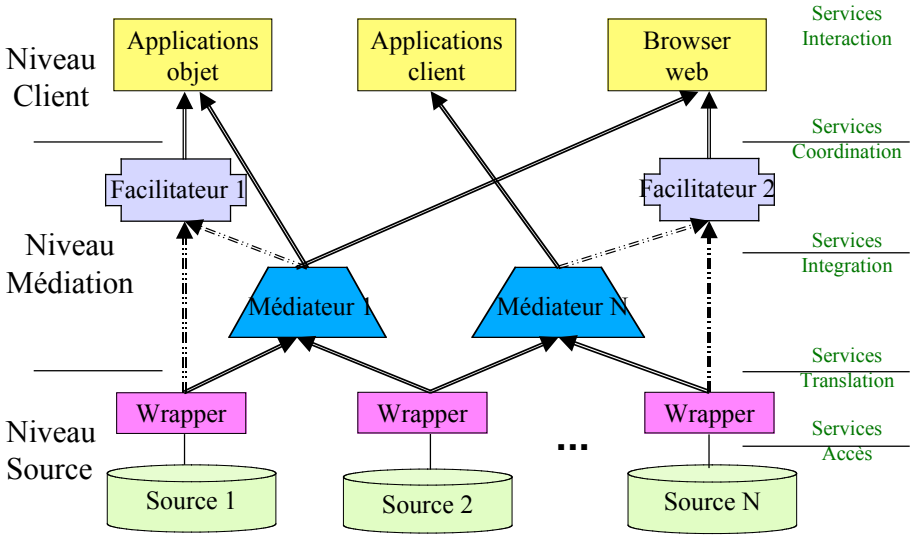


Figure 1 : Architecture d'un système de médiation

Le niveau source intègre, outre la source (fichier ou base), un ***wrapper*** (parfois appelé aussi *adapter* ou **adaptateur** en français) capable de recevoir des requêtes en langage pivot référençant un schéma exporté, de les traduire en requêtes locales, de récupérer les résultats et de les traduire en modèle pivot conformément au schéma exporté.

Le niveau médiation se compose essentiellement d'un **médiateur** responsable de l'intégration des données pour répondre aux requêtes des clients. Il reçoit des requêtes en langage pivot référençant une vue intégrée et les traite en déléguant au maximum aux adaptateurs locaux. En clair, un médiateur décompose une requête globale en sous-requêtes locales référençant chacune un schéma exporté. Il envoie ensuite ces dernières aux adaptateurs

locaux, collecte les résultats et les fusionne afin de générer la réponse transmise aux clients. Le niveau médiation comprend également en option un **facilitateur**. Il permet de transcrire les réponses aux requêtes dans le format souhaité par le client. Typiquement, les documents XML pourront être transformés en objets (cas d'un facilitateur objet) ou en HTML par des feuilles de style XSL (facilitateur HTML).

Le niveau client permet de soumettre des requêtes mettant en jeu des vues intégrées des données; il assure la transmission des requêtes vers le niveau médiation et présente les résultats aux applications, selon le codage désiré; il s'agit essentiellement d'une API de soumission de requêtes.

À partir de l'architecture de référence définie ci-dessus, différentes générations de systèmes ont vu le jour sur la base du choix du modèle et du langage pivot. Les premiers systèmes de médiation, tels que Multibase [100], Mermaid [136] et Le Select [101], ont choisi le modèle relationnel comme modèle commun de données. Ils ont intégré des sources de données multiples et hétérogènes comme des fichiers hiérarchiques, des bases de données réseaux et relationnelles. Ils sont bien adaptés aux applications qui ne manipulent pas de données complexes (i.e, seulement des données plates, bien modélisées en relationnel).

Avec l'émergence des systèmes de gestion de bases de données orientés objets dans les années 90, la nouvelle génération des systèmes de médiation orientés objets prend vie. Parmi eux, Pegasus [10], IRO-DB [77], GARLIC [29] et DISCO [145], ont vu le jour aux Etats Unis et en Europe. Ces systèmes répondent aux mêmes besoins que leurs prédécesseurs, mais au lieu d'utiliser le modèle relationnel, ils utilisent le modèle objet ou objet-relationnel soit comme modèle de données commun, soit en intégrant des SGBDO, soit les deux à la fois. Avec ses concepts de type et sous-type, d'opération, d'attribut multivalué et d'association, le modèle objet permet une meilleure intégration de données hétérogènes autour d'un modèle sémantiquement riche.

Plus récemment, une nouvelle génération de systèmes de médiation basée sur XML a vu le jour. Tsimmis [132] en 1994 a été le précurseur de cette nouvelle génération en permettant l'intégration de données faiblement structurées. Aujourd'hui de nombreux produits voient le jour. Ils sont commercialisés soit par des éditeurs reconnus (BEA avec Liquid Data) ou par des *start-up* telles Nimble.com (www.nimble.com), Enosys Software (www.enosys.com) ou e-XMLMedia (www.e-xmlmedia.com). IBM prépare aussi un produit correspondant à une version distribuée de XPeranto [28] de type SGBD fédéré autour de XML appelé DB2 Information Integrator. Tous ces produits ont en commun une architecture basée sur un modèle pivot XML et offrent un langage de requêtes pour XML, typiquement XPath [155] et XQuery [156].

Au cours de nos travaux de recherche, nous nous sommes particulièrement intéressés à la deuxième et troisième générations de systèmes de médiation de données. En effet, la fédération de bases de données hétérogènes nécessite très souvent un modèle pivot avec une sémantique riche pour la fédération de données.

La fédération de données autour d'un **modèle objet** pose des problèmes difficiles tels que le support d'identifiants globaux pour les objets permettant la migration d'objet sans ré-

identification, l'intégration de SGBD relationnels dans un environnement objet, l'optimisation de requêtes objet distribuées, la gestion d'associations entre objets intégrés, la mise à jour d'objets intégrés, etc. Afin de résoudre ces différents problèmes, nous avons proposé le système IRO-DB. Une particularité intéressante de notre système est qu'il adopte le modèle objet pour toutes les couches de son architecture et pas uniquement au niveau de la définition des vues intégrées. C'est un système de médiation complet. Il propose des outils pour concevoir et maintenir des applications sur de grandes fédérations de bases de données hétérogènes.

IRO-DB est un système complètement orienté objets, il s'appuie sur le modèle défini par l'ODMG [32,31] pour la description des données hétérogènes, ainsi que pour l'accès à la fédération. Par conséquent il hérite des nombreux avantages des SGBDO comme l'identité et le partage d'objets, la capacité de naviguer à travers les objets intégrés et des mécanismes de cache d'objets. Il intègre également les fonctionnalités bases de données de façon transparente dans le langage de programmation ce qui élimine les problèmes d'impedance-mismatch et offre des gains de performance.

Notre système permet donc, ce qui est rare pour un système de médiation, un accès à la fois déclaratif et interactif via le langage OQL, mais aussi par navigation via OML/C++ aux vues intégrées. Grâce aux objets images introduits au niveau du schéma de médiation objet et aux facilités d'accès offertes par l'interface de programmation OML/C++, IRO-DB offre des capacités non seulement d'interrogation, mais aussi de mises à jour des objets intégrés et permet l'invocation de méthodes à distance.

Une autre particularité de notre système est qu'il offre une approche hybride entre bases de données virtuelles et entrepôt de données. Il permet de stocker durant une transaction des objets venant de diverses bases de données locales. La réplication d'objets au niveau du système de médiation améliore les performances des applications. En effet, après un certain temps, IRO-DB se comporte comme un système centralisé. Cette hypothèse est surtout intéressante si l'on considère des applications complexes, ayant un fort besoin d'intégration de données, avec des transactions longues manipulant des objets complexes et nécessitant un mécanisme de navigation efficace parmi les objets.

L'autre particularité de notre système de médiation est d'offrir des techniques d'optimisation originales. En effet, comme le modèle d'intégration de données est riche, il permet la gestion d'associations entre objets intégrés et autorise des parcours de chemin. Au cours de nos travaux nous nous sommes particulièrement intéressés aux techniques d'évaluation et d'optimisation de requêtes adaptées à la gestion des parcours de chemin entre objets intégrés. À notre connaissance, ce problème n'a jamais été étudié auparavant. Par ailleurs, nous avons étudié les techniques d'optimisation de requêtes en relation avec les mécanismes d'instanciation et de réplication d'objets intégrés. Même si dans certains systèmes de médiation de données des caches existent, ils ne sont généralement pas exploités par l'optimiseur de requêtes. Notre approche a consisté à définir des règles de décomposition et d'optimisation de requêtes qui tirent profit de toute la puissance du mécanisme d'instanciation hybride des objets intégrés. Ces règles permettent d'éviter des envois de requêtes distantes et/ou de limiter les transferts de données inutiles.

La fédération de données autour d'un **modèle semi-structuré** pose d'autres types de problèmes. En effet, la plupart des systèmes de première et deuxième génération proposent généralement des approches statiques pour fédérer un nombre réduit et prédéfini de bases de données hétérogènes. Les techniques proposées s'adaptent difficilement à un environnement largement distribué et très changeant comme celui de l'Internet. La volatilité du nombre de sources (i.e, ajout, suppression et mise à jour des sources) ne permet pas une intégration manuelle ou semi-automatique des données. Par ailleurs, il est important d'offrir des techniques d'évaluation efficace même en cas de requêtes complexes et/ou de trafic réseau inconstant, des résultats partiels de requêtes en cas d'indisponibilité de certaines sources et une production continue de données.

Une grande majorité des systèmes choisissent comme modèle pivot un modèle de données semi-structurées comme XML. Dans le domaine du Web sémantique, les systèmes adoptent un modèle de données sémantiques comme les ontologies. Bien que ces deux approches permettent d'intégrer des sources semi-structurées aussi bien que sources structurées, elles ne s'appuient pas sur les mêmes langages d'interrogation, ni sur les mêmes techniques de réécriture. Au cours de nos travaux, nous n'avons pas abordé les problèmes liés au Web sémantique.

Plusieurs approches ont été proposées pour permettre un passage à l'échelle au niveau de la définition des vues intégrées. Certains systèmes comme Information Manifold [102], PICSEL [123] ou Agora [107,106], prônent l'utilisation de l'approche LaV, c'est-à-dire les schémas locaux sont exprimés comme des vues sur le schéma global. Chaque ajout d'une nouvelle source consiste à ajouter une nouvelle vue sur le schéma global. Cette approche offre des avantages mais également quelques inconvénients. Elle nécessite parfois des techniques de réécriture de requêtes plus complexes, et par ailleurs l'intégration de données consiste en une union pure et simple des données des sources et ne permet donc pas ni la fusion, ni la restructuration des données des sources afin d'offrir une vue cohérente et intégrée au niveau de la fédération.

Au cours de nos travaux de recherche, nous avons proposé le système MEDIAGRID [37, 141] dont l'objectif est de spécifier et d'implanter les éléments d'une infrastructure de médiation tout en tenant compte des spécificités de la globalisation des données sur la grille (GRID). Les activités de recherche du projet se sont focalisées sur l'automatisation de la construction de requêtes de médiation dans un contexte XML et sur la définition d'un ensemble d'interfaces et de classes abstraites réutilisables pour l'évaluation de requêtes XML adaptatives et interactives.

Le fait d'automatiser le processus d'intégration permet de s'adapter à la forte dynamicité de l'environnement (i.e, ajout, retrait de sources, mise à jour des schémas des sources). Nous avons proposé un module appelé *Mediation Queries Generator* dont l'objectif est de générer la(les) requête(s) de médiation candidates à partir de méta-informations décrivant les besoins en interrogation (i.e. le schéma global), la description des sources (i.e. les schémas exportés des sources), ainsi que les correspondances sémantiques entre le schéma global et les schémas exportés. Une requête de médiation décrit les liens de calcul entre le schéma global et les schémas exportés, c'est-à-dire la manière dont les instances du schéma global sont calculées à partir des instances des schémas exportés.

Dans MEDIAGRID, les requêtes de médiation sont générées selon l'approche GaV [96] et stockées comme partie de méta-données, les schémas sont décrits en XMLSchemas et les requêtes de médiation sont définies en XQuery. Les requêtes provenant des applications s'expriment sur le schéma global et sont réécrites en utilisant la (les) requête(s) de médiation générée(s) précédemment. L'intérêt de ce type d'approche est de combiner les avantages des approches GaV (i.e, intégration forte et cohérente des données avec fusion et restructuration des sources, et techniques de réécriture efficaces) et LaV (i.e, pour ajouter une nouvelle source il suffit d'ajouter de nouvelles correspondances sémantiques), tout en s'adaptant à la forte volatilité des sources de données.

Nos autres travaux de recherche se sont intéressés à la définition d'un canevas appelé *QBF (Query Broker Framework)* pour l'évaluation de requêtes XQuery. Un canevas consiste à définir des classes abstraites paramétrables et réutilisables. Le défi de QBF [38] est d'offrir des algorithmes, des outils, des modèles pour choisir les "meilleurs" plans d'exécution d'une requête, tout en tenant compte des besoins utilisateurs en offrant un mode interactif, pour autoriser des résultats partiels et pour offrir la possibilité de raffiner le plan d'exécution en cours d'exécution des requêtes. L'objectif de cet axe de recherche n'était pas de définir de nouvelles techniques d'optimisation ou d'évaluation de requêtes, mais plutôt d'intégrer et de modéliser au sein d'un même composant une grande majorité des techniques d'optimisations proposés dans la littérature, tout en gardant une bonne flexibilité et réutilisabilité.

Au sein du projet MEDIAGRID, nous avons confié l' évaluation des requêtes de médiation au module Evaluator qui est une instance particulière de QBF. Le module Evaluator utilise lui aussi un certain nombre de métadonnées comme les capacités des sources, des statistiques sur les données (cardinalités, distribution, etc.). Ce module est susceptible d'envoyer des sous-requêtes aux sources (ou plus précisément aux Wrappers), de récupérer des données et de réaliser les calculs nécessaires pour construire les résultats et les retourner à l'utilisateur. Ce dernier point est très classique. Mais l'intérêt de l'approche est que l'utilisateur peut choisir de paramétrer l'exécution de sa requête comme par exemple choisir des résultats partiels, un temps maximum de calcul, etc. Il peut également interagir avec le module d'exécution comme par exemple raffiner sa requête en ajoutant de nouvelles restrictions, suspendre l'exécution de certaines parties du plan d'exécution, etc.

Pour conclure cet état de l'art, nous mentionnerons une autre problématique apparue ces dernières années qui est celles des systèmes Pair-à-Pair. Bien que nous n'ayons pas travaillé sur cette problématique, il nous semble pertinent de la mentionner car elle constitue une suite logique et naturelle des travaux de recherche que nous avons menés jusqu'à présent et, sans elle, l'état de l'art ne serait pas complet.

Les systèmes Pair à Pair sont apparus pour répondre au besoin des systèmes ouverts où l'auto-administration[4], la tolérance aux fautes, la zéro-centralisation et la consultation sont la règle. Les nœuds du réseau arrivent et partent très rapidement, mais malgré tout on souhaite partager et interroger un très grand nombre de ressources. Les premiers systèmes P2P tels que Napster (www.napster.com) et Gnutella (www.gnutella.com), ont été conçus pour le partage de

[4] Self-organizing : auto-géré, égalitaire.

fichiers. Plus récemment Skype [133] adopte une architecture P2P pour la téléphonie sur IP. Actuellement le Pair à Pair intéresse non seulement la communauté GRID, mais également la communauté base de données. De nombreux projets voient le jour [81,4,101,130,121,58,8], l'objectif est de faire reculer les limites associées aux systèmes Pair à Pair actuels comme la granularité forte de l'indexation (le fichier), les capacités de recherche et de localisation qui sont basées sur une recherche exacte. On cherche donc à avoir une indexation à grain plus fin (n-uplets, couples attribut-valeur, expression de chemins), à autoriser des requêtes par intervalles ou complexes, et à autoriser non seulement la consultation mais également l'écriture.

3 Le système IRO-DB

Dans cette section, nous détaillons nos principales contributions. Comme il n'était guère facile de tout présenter, nous avons choisi de nous concentrer sur le système de médiation de données IRO-DB. D'une part, ce système offre un très grand nombre de particularités et de fonctionnalités intéressantes par rapport à l'ensemble des systèmes de médiation présentés dans la littérature. D'autre part, bien que ces solutions aient été développées dans un contexte particulier qui est celui de l'orienté objet, elles nous semblent intéressantes pour aller plus loin dans l'utilisation actuelle de XML. En effet, les solutions existantes pour l'accès transparent à des sources de données XML se focalisent beaucoup sur l'aspect valeur de XML et sur l'interrogation du Web. Ces solutions sont très intéressantes, mais elles ne mettent pas l'accent sur le développement d'applications sur le Web ou la grille. C'est pourquoi, on ne trouve pas de mode d'accès transparent par navigation, de partage de fragments de données XML, de mises à jour, d'intégration forte et performante de documents XML. Tous ces aspects sont présents dans le système IRO-DB. Nous discuterons de cette synergie dans la conclusion de ce manuscrit.

Dans une première section, nous présentons l'architecture fortement couplée et orientée objets du système IRO-DB. Puis nous détaillons les techniques d'intégration de schémas et données associées. Dans une dernière partie, nous décrivons les problèmes liés à la gestion des requêtes dans IRO-DB et les techniques d'optimisation associées.

3.1 Une architecture orientée objets et fortement couplée

Le système IRO-DB a été réalisé dans le cadre d'un projet européen (ESPRIT), conjointement par des équipes de recherche et des industriels européens. Il a pour but de permettre la coopération entre des SGBD relationnels et objets. IRO-DB a adopté le modèle objet pour toutes les couches de son architecture et utilise le modèle défini par l'ODMG [32,31] pour la description des données hétérogènes, ainsi que pour l'accès à la fédération. L'architecture du système IRO-DB [63, 78, 77] est organisée en trois niveaux de composants, comme le montre la Figure 2.

La *couche interopérable* comporte de nombreux modules, chargés de réaliser la coopération à tous les niveaux entre les SGBD intégrés. Le composant *Integrator's Workbench* permet de spécifier les schémas intégrés, qui sont des vues intégrées des bases de données fédérées. Les vues et les schémas exportés sont stockés dans un dictionnaire, sur le client. La

manipulation d'objets comprend le langage OQL imbriqué dans le langage utilisateur OML/C++, l'interface de programmation des applications conforme à l'ODMG. Afin de gérer des programmes OML/C++, la couche interopérable fournit un certain nombre de services comme : (1) le gestionnaire de requêtes responsable de l'évaluation des requêtes OQL sur le schéma intégré, (2) le gérant d'objets responsable de gérer les appels de méthodes et la création d'objets intégrés. Ces deux services sont très importants pour déléguer les accès aux objets intégrés vers les bases de données distantes associées. Un gestionnaire de transactions globales [137] est offert. Ce module se base sur le modèle de transactions défini par l'ODMG qui supporte les instructions *begin, commit, abort, abort_top_level* et *checkpoint.* Cependant, la sémantique de ces primitives n'étant pas très claire, une sémantique plus précise a été définie. Enfin, il existe aussi un système de sécurité [60], permettant de vérifier les accès au niveau global. Son rôle est d'assurer (1) le "secret" des données, c'est-à-dire leur protection contre des interrogations non autorisées, (2) leur intégrité, c'est-à-dire leur protection contre des modifications non autorisées, et (3) leur disponibilité, pour les usagers autorisés.

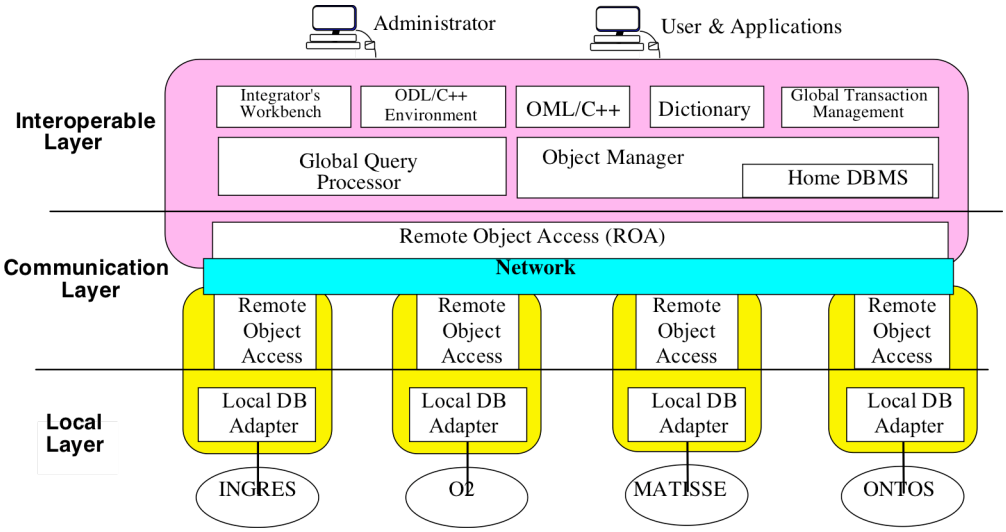


Figure 2 : Architecture de IRO-DB

La *couche communication* implante les services RDA (Remote Data Access) étendus aux objets, grâce aux modules ROA (*Remote Object Access*) qui permettent l'accès aux objets distants à la fois sur les clients et sur les serveurs. Le système respecte les standards pour l'interconnexion de systèmes ouverts. L'interface externe suit les recommandations du groupe *SQL Access Group* Call Level Interface [45], en l'étendant pour pouvoir traiter les objets.

La *couche locale* est composée d'adaptateurs locaux, propres à chaque SGBD. Elle permet à un système local de répondre à des requêtes OQL, exprimées sur une abstraction du schéma local, le schéma exporté, défini en utilisant les spécifications de l'ODMG. Comme un schéma exporté décrit uniquement les types implantés dans le SGBD local, seules les fonctions disponibles localement sont possibles dans les requêtes OQL globales.

3.2 Un schéma de médiation objet

La gestion des schémas est réalisée par le module d'intégration, appelé l'*Integrator Workbench*.

Ce module comporte, en plus des aspects de gestion de schémas, une interface utilisateur destinée à l'administrateur du système. Le module d'intégration s'appuie sur trois niveaux de schémas et un processus classique d'intégration constitué des phases de transformation, importation et intégration. Trois niveaux de schémas sont définis dans IRO-DB et quatre types différents de classes sont identifiés dans ces schémas, comme décrits en Figure 3[26] :

• Les *classes externes* correspondent aux classes préexistantes sur les SGBD locaux. Les classes préexistantes, également appelées classes locales, sont définies dans le modèle de données propre à leur SGBD, alors que les classes externes sont décrites en ODL, afin de les rendre accessibles au système fédéré. Les objets des classes externes sont des objets locaux.

• *Les classes importées* sont définies comme une copie 1-1 des classes externes. Elles sont les "images" dans IRO-DB, au niveau global, des classes externes. Les objets des classes importées sont des objets globaux. Chaque objet est aussi l'"image" d'un objet local. Une classe importée fournit la visibilité de la classe externe correspondante au niveau de la couche interopérable.

• *Les classes dérivées* sont ajoutées au-dessus des classes importées. Elles permettent la fusion de plusieurs objets importés. Un objet d'une classe dérivée peut être issu d'objets de plusieurs classes importées. La dérivation des propriétés et des *associations* d'un objet dérivé peut inclure des correspondances arbitraires 1-N entre les deux types de classes.

• *Les classes standard* sont définies directement au niveau interopérable. Elles ne sont pas dérivées de classes externes. Elles permettent aux développeurs d'applications de gérer leurs propres classes d'objets persistantes localement comme dans tout SGBDO.

Les classes importées, dérivées et standards composent le schéma intégré d'IRO-DB. Les classes externes sont uniquement vues par l'*integrator Workbench*, qui est le composant réalisant l'intégration. Les classes importées et dérivées sont aussi appelées classes virtuelles, ou vues. Une classe virtuelle dérive ses objets des objets des autres classes.

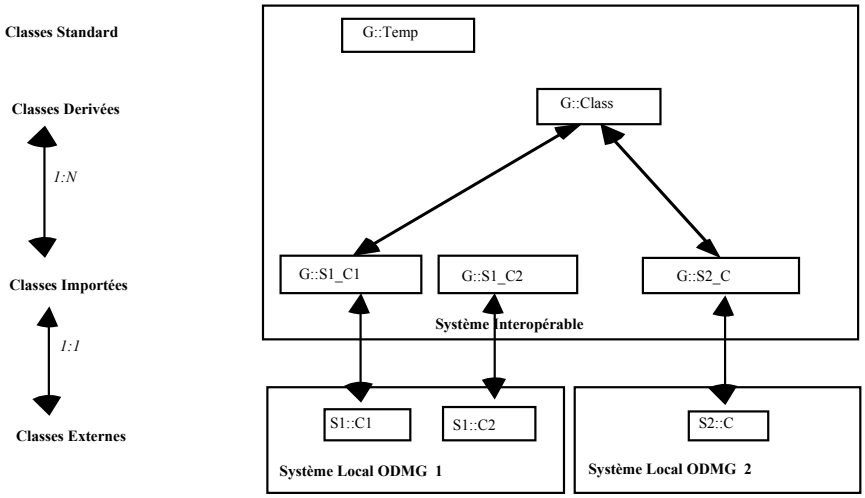


Figure 3 : Les niveaux de schémas

La méthode d'intégration proposée dans IRO-DB est indépendante d'un modèle de données particulier. Cependant, la volonté étant de suivre le plus possible les recommandations de l'ODMG, les schémas et les correspondances entre classes ont été traduits en ODL.

La traduction comprend deux parties: (1) la spécification du schéma, composé d'un ensemble d'interfaces et (2) la spécification des règles de correspondance appelées *mappings*. Les interfaces sont spécifiées en ODL standard. Rien ne les distingue de l'interface d'une classe standard. Les règles de dérivation sont données dans un bloc de définition séparé. La syntaxe utilisée est proche de ODL. Les règles elles-mêmes sont décrites en OQL. Trois types de requêtes sont nécessaires pour : (1) peupler l'extension de la classe, (2) définir les règles de correspondance entre les attributs, (3) définir les associations. La spécification générique des règles de correspondance est présentée dans la Figure 4.

Un aspect original du modèle d'intégration est de proposer le concept d'objets 'images' (ou origin), qui sont des liens spécifiques au niveau des classes virtuelles, et contiennent des références vers les littéraux ou objets originaux. Toutes les règles de correspondances s'appuient sur ce concept. Il facilite l'identification et la création des objets intégrés au niveau du système de médiation, le support des mises à jour et l'invocation de méthodes distantes. Tous ces points sont expliqués dans la suite de cette section. Ces associations sont cachées aux utilisateurs des classes virtuelles. Notons qu'en raison de la cardinalité 1-N du lien entre classes importées et dérivées, il peut exister plusieurs définitions d'origine dans les classes dérivées, mais uniquement une seule, en raison de la cardinalité 1-1 du lien entre classes externes et importées, dans les classes importées.

```
mapping [imported] clsname {
        origin          typename        orig1;
        [origin …]
        [def_extent extentname  as      select clsname(orig1 : i1,…)
                                        from i1 in , …
                                        where …;
        def_att attributename   as      OQL_query;
        def-rel pathname        as      select p
                                        from p in otherclsname
                                        where   (p.orig1 = this.orig1.pathname1)
                                        or  (p.orig2 = this.orig2.pathname2) };
```

Figure 4 : Spécifications génériques des règles de correspondances

La définition de l'extension de la classe est optionnelle. Elle n'est pas utile pour les classes importées, dont l'extension est toujours définie comme celle des classes externes dont elles dépendent. Pour les classes dérivées par contre, cette définition est très importante. Les trois clauses, *select*, *from* et *where*, ont un rôle différent et tout aussi important. La clause *select* comporte le constructeur de la classe dérivée. Comme de nouveaux objets "images" sont générés lors de l'exécution, il faut s'assurer que des accès futurs à la même classe virtuelle rendent toujours le même objet image. Il faut donc que l'objet "image" persiste durant toute la transaction, c'est-à-dire qu'aucun nouvel objet 'image' ne soit généré pour le même objet de base.

Ce problème de duplication est évité grâce au constructeur "clsname ($orig_1$: i_1, …)". En effet, avant de créer une nouvelle image, le constructeur vérifie le contenu de l'extension de la classe : si l'objet existe déjà, le constructeur rend sa référence, sinon, il crée une nouvelle image. La clause *from* permet de connaître toutes les classes importées impliquées dans la définition de la classe dérivée. La clause *where* spécifie comment les instances des classes importées seront

combinées entre elles ; elle traduit les similitudes détectées entre les instances des classes de base. Elle peut donc exprimer une jointure, une jointure externe, ou une union.

Les propriétés des classes virtuelles sont définies par rapport aux propriétés des classes de bases qui la composent. Pour les classes importées, la définition est implicite, toujours en raison de la cardinalité 1-1 du lien entre classes externes et importées. Pour les classes dérivées, la définition consiste en une requête OQL, plus ou moins complexe. La requête doit préciser, pour chaque propriété, de quelle(s) source(s) elle provient, et, s'il y en a plusieurs, comment ces sources sont combinées : par préférence à l'une des sources, par composition (un agrégat tel que min, max, somme, moyenne...), etc. De plus, la requête doit pouvoir expliciter les conversions éventuellement nécessaires entre les différentes propriétés de base.

Quelques propriétés simples sont décrites ci-dessous. Le *this* correspond à l'instance courante, sur laquelle la propriété est définie. La source de base est également spécifiée; elle est représentée par l'origine. S'il n'y en a qu'une, ou si l'on suppose qu'il n'y a pas de conflit, on peut choisir une seule origine, comme cela est illustré pour la définition de l'attribut *nom* d'une classe. Le "ou", utilisé pour la définition de l'attribut *date*, est l'équivalent du *chooseany*, c'est-à-dire qu'il prend la valeur de la première instance de base pour laquelle la propriété est définie. Enfin, la règle de correspondance définissant l'attribut *prix* applique la méthode de conversion *francs* à l'attribut *prix*. Ces requêtes ne sont que des exemples. Toute requête OQL peut, en principe, servir à définir une propriété.

```
nom as this.orig1.name
date as this.orig1.date or this.orig2.date
prix as this.orgi1.prix.francs()
```

Un exemple complet de schéma de médiation est donné en ANNEXE A. Il décrit l'intégration de pièces et de fabricants provenant de deux sites différents. Dans IRO-DB, il est possible d'ajouter de nouvelles propriétés aux classes dérivées. En effet, puisqu'un OID "global" est attribué à chaque instance d'une classe dérivée, on peut définir sur cette instance de nouvelles propriétés, qui ne sont pas calculées à partir des classes importées, et qui ont un sens uniquement au niveau global, ainsi les utilisateurs peuvent personnaliser leur vue dérivée.

3.3 Évaluation et optimisation de requêtes

Dans cette section, nous présentons nos contributions concernant les mécanismes d'évaluation et d'optimisation de requêtes développés dans IRO-DB. Le but d'un optimiseur est de réduire le nombre de messages à envoyer, les transferts de données, ainsi que leur taille. Il serait trop long de retracer ici toutes les techniques d'évaluation et d'optimisation de requêtes, certaines d'entre elles proviennent des travaux de recherche menés dès les années 80 sur les systèmes de bases de données distribués et parallèles. Pour trouver un état de l'art détaillé et récent sur la gestion de requêtes réparties, le lecteur se référera à l'article de synthèse de Donald Kossman [97].

Dans cette section, nous nous focalisons essentiellement sur l'ensemble des règles d'optimisation définies pour prendre en compte les problèmes associés aux parcours de chemin et intégrer les mécanismes de réplication et de cache proposés par le gérant d'objets. Ces deux aspects constituent des contributions intéressantes par rapport à l'état de l'art comme nous

l'avons expliqué dans la section précédente. À chaque fois, pour illustrer le problème, nous donnons un exemple de requête et les étapes clés du processus d'optimisation par réécriture que nous proposons. Pour ne pas alourdir le manuscrit, nous avons omis la liste complète des règles qui serait trop longue. Pour de plus amples détails, le lecteur se réfèrera à la thèse de Véronique Smahi [99]. Mais avant de commencer, il est important de rappeler les composants et leurs interactions du gestionnaire de requêtes.

3.3.1 Le gestionnaire de requêtes

Le gestionnaire de requêtes comporte six modules principaux définis ci-dessous.

L'*analyseur syntaxique.* Il prend une requête exprimée sur le schéma intégré. Il effectue l'analyse syntaxique et sémantique et produit un OET (Object Expression Tree) qui correspond à la représentation interne d'une requête [67]. Nous avons ainsi proposé une représentation formelle des requêtes recouvrant une bonne part d'OQL. L'analyseur syntaxique vérifie également que la requête référence des points d'entrée corrects, c'est-à-dire des objets nommés ou des extensions de classes.

Le *Gestionnaire de Requêtes Global.* Il réalise trois fonctions qui sont (1) la transformation réalisée par le *Translateur de requêtes*, (2) la décomposition et (3) l'optimisation globale de la requête. *Le Translateur de requêtes* transforme les noeuds correspondants aux classes dérivées. Une classe dérivée peut être comparée à une vue. Translater une requête revient à faire un calcul de vue, c'est-à-dire à remplacer la vue par sa définition sur les classes de base. Dans IRO-DB, les classes de base correspondent aux classes importées. Le *Décomposeur de requêtes* décompose la requête transformée en une requête de recomposition et un ensemble de sous-requêtes mono-sites. *L'Optimiseur* a pour mission d'améliorer les performances du traitement des requêtes. Il génère un plan d'exécution comprenant deux parties: (i) l'ensemble des requêtes distantes à exécuter sur les SGBD locaux, (ii) les requêtes correspondant à la partie de la requête globale devant être évaluée sur le Home DBMS (cf Figure 2). L'optimiseur réduit les transferts d'objets et les coûts de communication entre IRO-DB et les sites locaux. Un des intérêts de notre approche est de proposer un optimiseur extensible basé sur un ensemble de règles et différentes stratégies d'optimisation.

L*'évaluateur de Requêtes Global* (GQE). Il reçoit le plan d'exécution généré par le *Gestionnaire de requêtes*. Il envoie les requêtes distantes au *Délégateur de requêtes* qui en retour renvoie les résultats des évaluations locales. Ensuite, le GQE recompose les résultats en commençant l'évaluation de la partie globale de la requête. Le résultat est alors renvoyé à l'application.

Le *Délégateur de requêtes* se charge d'envoyer et de recevoir les requêtes OQL aux bases de données distantes à travers les primitives OQL/CLI. Il reçoit les résultats et génère les objets images en appelant le gestionnaire d'objets. Les objets images correspondent aux références d'objets contenues dans le résultat.

Le *Gestionnaire d'Objets Virtuels* assure la gestion des objets virtuels des classes dérivées et importées. Il permet la création des objets et leur accès ultérieur, en garantissant leur unicité.

Il n'est pas nécessaire d'instancier une classe importée pour l'évaluation d'une requête OQL. Par contre, un objet d'une classe dérivée doit toujours être créé puisqu'il n'existe dans aucun SGBD distant. Le *Virtual Object Manager* est appelé par le *Délégateur de requêtes*, pour la création des objets des classes importées et par l'évaluateur pour la création des objets des classes dérivées.

3.3.2 Les parcours de chemin

Le parcours de chemin est un principe important des modèles à objets et il remplace la jointure par valeur. Dans un système tel qu'IRO-DB, il n'existe pas de références inter-sites. Par conséquent il n'existe pas de chemin multi-sites entre classes importées. Un parcours de chemin entre classes importées est donc forcément mono-site. Un parcours de chemin entre classes dérivées n'est donc jamais un parcours de pointeurs, binaire ou n-aire, comme cela est réalisé dans les systèmes centralisés. Ce n'est jamais non plus un parcours d'index. En effet, chaque propriété d'une classe dérivée étant définie par rapport aux classes importées, les associations sont elles aussi définies par rapport aux associations existant dans les classes importées.

Supposons qu'un utilisateur souhaite retrouver la description des pièces qui sont fabriquées à Paris. Il écrit donc la requête OQL ci-dessous sur le schéma de médiation donné en Annexe A.

```
Requête initiale :
                    select p.description
                    from p in PIECES, fa in p.fabriquant
                    where fa.ville = "Paris "
```

La première étape du processus de réécriture consiste à transformer la requête utilisateur et à remplacer le nom de la classe dérivée PIECES par sa définition. Le processus de transformation utilise les spécifications de dérivations définies pour chaque classe et décrites en ANNEXE A. S'il y a plusieurs niveaux de classes dérivées, le processus s'applique récursivement, jusqu'à ce que la requête ne référence plus que des classes importées. Le résultat de la transformation est donc une requête OQL, à plusieurs niveaux d'imbrication telle que définie ci-dessous.

```
Requête transformée :
    select p.description
    from p in (     select PIECE  (orig1 : p1, orig2 : p2)
                    from p1 in pièce1s, p2 in pièce2s
                    where p1.id = p2.id),
         fa in (    select f
                    from f in (     select FABRIQUANT  (orig1 : f1, orig2 : f2)
                                    from f1 in fabriquant1s, f2 in fabriquant2s
                                    where f1.id = f2.id  )
                    where (p.orig1.fabriquant1 = f.orig1)
                    or  (p.orig2.fabriquant2 = f.orig2))
    where fa.ville = "Paris "
```

Après l'étape de transformation, toute requête contenant une association devient donc une requête imbriquée au *from* et corrélée. Si l'optimiseur ne fait rien, l'évaluation de ce type de requêtes s'effectuera instance par instance et nécessitera la matérialisation complète de la classe dérivée FABRICANT, ce qui est très coûteux. Nous avons donc proposé un ensemble de règles d'optimisation permettant d'aplatir ces requêtes dans l'objectif d'obtenir un traitement

ensembliste. Nous avons défini une règle de fusion qui analyse la requête OQL, recherche les constructeurs de classes dérivées et réalise les substitutions de variables appropriées en consultant à la fois les origines et les règles de correspondances. Il est à noter que les optimiseurs OQL traditionnels ne peuvent réaliser ce type d'optimisation puisqu'ils ne connaissent pas à priori la sémantique associée à l'appel de méthode dans le SELECT.

```
Requête après applications successives de fusion :
select p1.description
from p1 in pièce1s, p2 in pièce2s, fa in (select FABRIQUANT (orig1 : f1, orig2 : f2)
                                          from f1 in fabriquant1s, f2 in fabriquant2s
                                          where f1.id = f2.id
                                          and (p1.fabriquant1 = f1) or (p2.fabriquant2 = f2))
where fa.ville = "Paris "
and p1.id = p2.id

Requête après application de la fusion des associations :
    select p1.description
    from p1 in pièce1s, p2 in pièce2s, f1 in fabriquant1s, f2 in fabriquant2s
    where f1.ville = "Paris "
    and f1.id = f2.id
    and p1.id = p2.id
    and ((p1.fabriquant1 = f1) or (p2.fabriquant2 = f2))
```

Afin de décrire l'ensemble du processus, nous donnons ci-dessous le résultat du processus de décomposition de la requête globale qui est composée d'une requête de recomposition et d'un ensemble de sous-requêtes mono-site. Les sous-requêtes sont envoyées aux sites locaux, par l'intermédiaire de la couche de communication. Les résultats des sous-requêtes sont stockés dans des mémoires tampons. La requête de recomposition est ensuite évaluée.

```
Sous-requêtes mono-site :
    Define Temp1 as
        remote_query (S1 ,
            "select struct (id : x.id, description : x.description, fabricant : x.fabricant1)
             from x in external_pièce
             where x.ville = 'Paris'")
    Define Temp1b as
        remote_query (S1 ,
            "select struct (orig : x, id : x.id)
             from x in external_fabricant")
    Define Temp2 as
        remote_query (S2 ,
            "select struct (id : x.id, fabricant : x.fabricant2)
             from x in external_pièce")
    Define Temp2b as
        remote_query (S2 ,
                "select select struct (orig : x, id : x.id)
                 from x in external_fabricant")

Requête de recomposition :
        select p1.description
        from p1 in Temp1, p2 in Temp2, f1 in Temp1b, f2 in Temp2b
        where f1.id = f2.id
        and p1.id = p2.id
        and ((p1.fabriquant1 = f1.orig) or (p2.fabriquant2 = f2.orig))
```

3.3.3 Optimisation et Mécanismes d'instanciation

Le gérant d'objets d'IRO-DB joue le rôle de cache d'objets se trouvant généralement dans les

SGBDO. La réplication (ou création) d'objets est optionnelle. Cette décision est contrôlée par le gestionnaire de requêtes ou le gérant d'objets qui décident de créer des objets images pour améliorer l'évaluation d'une requête ou d'un ensemble de requêtes, ou pour pouvoir exécuter des appels de méthodes (i.e, accès à une valeur d'attribut d'un objet intégré ou mise à jour des objets distants). Le pourcentage d'objets gérés dans le cache constitue une information utile pour l'optimiseur de requêtes car si les objets sont déjà stockés sur le client cela évite certaines transformations inutiles de la requête et donc un calcul inutile des classes dérivées. Cela évite ou diminue les accès distants et réduit la quantité d'information qui doit être transférée. Dans IRO-DB, trois états peuvent être définis pour une classe virtuelle qu'elle soit importée ou dérivée [5]:

- *L'état Minimal*. Il correspond à l'instanciation minimale (i.e., aucun objet image n'a été créé dans le cache pour une classe virtuelle),
- *L'état Total*. Il correspond à une instanciation totale quand l'extension d'une classe virtuelle est complètement définie (i.e., tous les objets images appartenant à une classe virtuelle ont été créés dans le cache),
- *L'état Partiel*. Il correspond à une instanciation partielle quand seulement une partie de l'extension virtuelle d'une classe est connue.

Nous ne donnons ici que quelques règles relativement simples correspondant au traitement des classes virtuelles importées . Nous nous focalisons uniquement sur les deux extrêmes, c'est-à-dire l'état minimal et l'état total. Pour une étude complète du problème, le lecteur se réfèrera à la thèse de Véronique Smahi [99] ou à notre l'article [71].

L'état minimal correspond à une base vide dans le gérant d'objets. Ici, l'optimiseur de requêtes doit définir l'évaluation de base qui consiste à évaluer complètement la requête sur un site local. La règle donnée ci-dessous correspond à la règle par défaut qui peut toujours être appliquée pour des requêtes "ad-hoc", c'est-à-dire ne nécessitant pas une instanciation du cache comme par exemple lors de l'utilisation de l'interface déclarative interactive. L'optimiseur de requêtes génère une requête distante envoyée au site gérant la classe importé et stockée dans les règles de correspondances. Le nom de la classe est remplacé par le nom de la classe externe correspondante.

```
select c, c.att1, c.att2,..., c.atti, c.m1(),...c.mk()
from c in clsname
where pred (c.attj, .., c.attn, c.ml(),...c.mt())            ==>

  remote_query (Site,
                "select c, c.att1, c.att2,..., c.atti , c.m1(),...c.mk()
                from c in external_clsname
                where pred (c.attj, .., c.attn,  c.ml(),...c.mt() ) " )
```

L'état total est le cas le plus intéressant parce que, pour une classe importée, toutes les données sont déjà dans le cache; ainsi, aucun accès distant n'est requis. Cet état peut être comparé, du point de vue des performances, à celles qui sont réalisées dans un système centralisé. Deux règles doivent être définies en fonction de l'ensemble des opérations exécutées

[5] Rappelons que, lorsqu'une classe importée est instanciée, tous ses attributs sont évalués (i.e., l'origine et tous ses attributs). Au contraire, quand une classe dérivée est instanciée, seules les origines sont connues.

sur la classe importée. Quand l'ensemble des opérations ne contient pas d'appel de méthode, le noeud peut être transformé de deux manières différentes : soit la question entière est évaluée globalement, soit une partie est évaluée à distance. Les alternatives sont présentées ci-dessous. La décision dépend de la présence de méthodes d'accès sur les sites locaux. En effet, sur le SGBD-hôte, aucune méthode spécifique d'accès n'est définie. En conséquence, la recherche d'un objet peut être coûteuse si la cardinalité de la classe importée est élevée. Au contraire, des méthodes spécifiques d'accès, comme des index sont fréquemment disponibles sur les sites distants. L'idée consiste donc à tirer profit de ces techniques pour évaluer les prédicats sur le site distant, mais en récupérant toujours les valeurs des attributs dans le SGBD-hôte pour les objets qui satisfont les prédicats.

```
select c, c.att1, c.att2,..., c.atti
from c in clsname
where pred (c.attj, .., c.attn)                    ==>

        Première alternative :
                select c, c.att1, c.att2,..., c.atti
                from c in clsname*
                where pred (c.attj, .., c.attn)

        Deuxième alternative :
                define Temp1 as
                        remote_query (Site,  "select c
                                              from c in external_clsname
                                              where pred (c.attj, .., c.attn)" )

                select c, c.att1, c.att2,..., c.atti
                from c in clsname*
                where c.orig in Temp1
```

La première alternative correspond à une transformation simple. Elle consiste seulement à annoter le nom de la classe, pour indiquer à l'évaluateur global que toute l'information requise est disponible dans le cache d'objets. La deuxième alternative engendre deux requêtes. La première récupère les références des objets qui satisfont le prédicat, sur le site local. La deuxième requête évaluée par l'évaluateur global donne les valeurs des attributs pour les références récupérées dans la requête distante. Notons qu'on procédera de la même façon que la deuxième alternative si nous sommes en présence de méthodes. En effet, les méthodes doivent être évaluées localement.

Dans les systèmes de médiation de données, même si des caches existent ils ne sont pas exploités par l'optimiseur de requêtes. Ils sont uniquement utilisés pour l'évaluation des requêtes. Si l'on regarde l'ensemble des règles que nous avons proposées, hors d'un contexte particulier, on constate que l'instanciation peut engendrer un surcoût, qui n'est pas rentabilisé immédiatement. Nous allons maintenant indiquer quelques axes permettant de déterminer quand l'instanciation peut être bénéfique.

Le mapping n-m entre classes dérivées et classes importées a pour conséquence de faire apparaître plusieurs fois la même classe importée dans une requête transformée. Lorsque cela est possible, nous avons proposé au niveau de la phase de réécriture de regrouper les classes identiques dans une seule sous-requête mono-site. Cependant, le regroupement n'est pas

toujours possible. Il est alors intéressant d'instancier la classe importée évitant ainsi les accès distants ultérieurs. Ainsi, le coût de l'instanciation est factorisé.

Les requêtes OQL qui sont optimisées proviennent en majorité d'applications écrites en OML/C++. Les requêtes renvoient dans l'application soit une valeur ou un ensemble de valeurs, soit des références d'objets. Dans la seconde hypothèse, les références peuvent être utilisées dans l'application comme des références d'objets classiques. En conséquence, les références peuvent être déréférencées, et la valeur d'un attribut non "remontée" la première fois, peut être demandée par la suite. Si la classe est instanciée, les déréférencements ultérieurs seront moins coûteux, puisque aucun accès distant ne sera nécessaire. Par contre, s'il n'y a pas eu instanciation, la recherche ultérieure d'une propriété se fera instance par instance et générera un nombre important d'accès distants.

Dans les deux cas précédents, nous avons considéré le cadre d'une seule requête. Or, nous avons déjà évoqué que notre cadre incluait des transactions longues. Une transaction longue comporte un ensemble de requêtes. Si l'on suppose que l'ensemble de requêtes associée à une même transaction concerne le même domaine et donc touche les mêmes classes, il est intéressant d'instancier les classes dans les premières requêtes, afin que par la suite, plus aucun accès distant ne soit nécessaire. Malheureusement, comme dans tous les cas de prédiction, l'optimiseur « envisage » la suite des traitements, mais n'a aucune certitude. Il se peut très bien que les classes instanciées dans la première requête ne servent plus jamais ensuite. Dans ce cas, l'instanciation aura été effectuée inutilement. Une bonne connaissance du domaine d'application serait un plus.

4 Conclusion

Dans ce chapitre, nous avons décrit nos travaux et contributions dans le domaine des systèmes de médiation de données. Les deux systèmes que nous avons proposés s'appuient tous deux sur l'architecture de référence définie dans la section 2. Ils adoptent également tous deux une approche GaV concernant l'intégration et l'interrogation des données. L'approche proposée par IRO-DB est semi-automatique, elle ne dépend pas d'un modèle de données particulier. L'approche proposée dans MEDIAGRID est automatique.

Dans les années passées, on a souvent critiqué les approches GaV comme ne passant pas à l'échelle. Certes ces approches ne sont pas véritablement adaptées à une interrogation massive sur le Web et des approches développées dans le cadre du Web sémantique ou des systèmes Pair-à-Pair sont plus appropriées. Cependant elles sont nécessaires si l'on désire un minimum d'intégration de données même dans un contexte largement ouvert comme celui sur la grille. Si beaucoup de travaux ont été réalisés dans le domaine relationnel, il reste encore beaucoup à faire dans le cadre de XML.

Nos travaux dans MEDIAGRID ont permis de démontrer qu'il est possible d'intégrer plus fortement un ensemble de documents XML et d'offrir une vision intégrée et homogène d'un ensemble de sources aux utilisateurs. Cette approche tout d'abord définie dans un contexte relationnel [95] a été adaptée à un contexte XML. Les requêtes de médiation XQuery [156]

générées sont complexes et laissent envisager un processus d'intégration coûteux. En effet, des restructurations importantes de documents sont nécessaires. Afin de pouvoir s'adapter à la forte dynamicité des sources de données (i.e, ajout, retrait et mise à jour des schémas des sources), nous avons proposé une automatisation du processus d'intégration qui permet de tirer profit des approches LaV et GaV. Cependant, l'approche n'est pas encore incrémentale et il n'est pas encore possible de gérer ses propres données au milieu des données intégrées.

Actuellement, le gestionnaire de méta-données et les modules d'intégration et d'évaluation de requêtes ont été prototypés dans MEDIAGRID. Cependant nous ne disposons pas encore d'un processus de réécriture, ni d'optimiseur de requêtes XQuery et il n'est pas facile de prédire aujourd'hui à quel point nous serons capables d'optimiser. Des approches similaires à celles qui sont proposées dans MEDIAGRID voient le jour dans OGSA [142] pour la gestion de données. Parmi ces composants, on distingue OGSA-DAI (OGSA-Data Access & Integration) pour l'accès et l'intégration de sources de données relationnelles et XML sur la grille, OGSA-DQP (OGSA- Distributed Query Processing) pour l'évaluation de requêtes orienté service sur des collections de données et OGSA-MCS (OGSA-Metadata Catalog Service) pour la gestion des informations relatives à des domaines d'application spécifiques.

Actuellement, la plupart des systèmes de médiation n'offrent pas de mécanismes de navigation sur les données intégrées comme nous l'avons proposé dans IRO-DB. Bien que cela s'explique dans un contexte relationnel, cela n'est pas justifié dans un contexte objet ou semi-structuré. Même si il est important d'interroger de manière déclarative un document XML avec des langages comme XPath ou XQuery, il n'en demeure pas moins que la plupart des applications travaillent à partir d'API DOM [57]. Si certains travaux ont été menés dans le cadre des bases de données XML natives [114] autour de Persistant DOM [7], il reste encore beaucoup à faire pour offrir, sur des documents virtuels résultant de l'intégration de plusieurs sources de données XML, une interface équivalente à celle qu'offre l'interface OML/C++ avec ces mécanismes de navigation et d'évaluation de requêtes OQL intégrés. Nous pensons que nos travaux peuvent aider à la résolution du problème. En effet, le modèle XML permet d'identifier des documents sur le Web et intègre le concept de référence.

L'atout d'IRO-DB est d'offrir un système aux fonctionnalités avancées. Il adopte une approche orientée objets à tous les niveaux et bénéficie donc des atouts des SGBDO. Le modèle d'intégration de données proposé est intéressant car il permet la création, l'interrogation et la mise à jour des objets intégrés, mais également l'extension et l'ajout de nouvelles classes d'objets pour les besoins des applications et cela de façon totale transparente pour les développeurs. Nous avons vu que la gestion des liens inverses (i.e, les objets images) permet une bonne gestion du cache d'objets, la propagation des mises à jour et l'exécution de méthodes à distance. Actuellement, peu de systèmes proposent une approche hybride, c'est-à-dire une approche entre systèmes de médiation et entrepôt. Pourtant cela nous paraît particulièrement important pour offrir des systèmes de médiation performants. Les problématiques que nous avons adressées au niveau du processus d'optimisation ont été peu abordées dans la littérature. En résumé, nous pensons que notre approche bien que fortement couplée est intéressante pour le développement d'applications à base de services Web pour lesquelles le partage et la cohérence des données sont forts, le besoin d'intégration relativement fin et le besoin de

performances important. Le couplage fort entre tous les composants du système (intégration, gestion des requêtes, gestion de cache et de transactions) offre de nombreux avantages en termes de performance, mais aussi de fonctionnalités.

Dans le cadre du projet européen, nous avons prototypé le système IRO-DB. Trois systèmes locaux objet ont été considérés (O2, MATISSE et ONTOS), ainsi qu'un système relationnel. Lors d'une première expérience, nous avons choisi ONTOS comme support de stockage (Home DBMS), mais n'importe lequel des SGBD objet compatible ODMG peut le remplacer.

Chapitre III – Services d'annuaires

Ce chapitre présente nos travaux de recherche qui se sont déroulés entre 1997 et 2001 dans le domaine des intergiciels systèmes distribués. Les services qui nous ont particulièrement intéressés sont les services d'annuaires car ils permettent de localiser rapidement, simplement et en toute sécurité des ressources largement réparties à l'échelle planétaire. La transparence d'accès à un ensemble de ressources est offerte par l'intermédiaire de langages d'interrogation simples et déclaratifs accessibles par des non-spécialistes. Une des caractéristiques essentielle de ces systèmes est d'offrir une architecture permettant de gérer des ressources pour lesquelles la nature des ressources ou des propriétés rend le catalogage centralisé impossible ou inadapté. Au cours de nos travaux, nous nous sommes particulièrement intéressés aux problèmes des environnements télécoms.

1 Introduction

Dans l'environnement des télécoms, trois technologies différentes peuvent coexister dans le système d'information : des SGBD, des serveurs LDAP et des plates-formes à objets distribués. En effet durant les années 90, la communauté télécom a été fortement influencée par les technologies proposées par l'OMG (Object Management Group) [119], puis par celle des EJB (Enterprise JavaBeans [61]). Cet intérêt repose sur le fait qu'il est beaucoup plus facile de développer et d'intégrer des systèmes de gestion en utilisant des technologies interopérables au niveau API (e.g. CORBA) plutôt qu'au niveau protocole (e.g, CMIP pour les systèmes de gestion OSI, SNMP pour les systèmes de gestion du monde Internet).

Puis vers le milieu des années 90, le W3C (World Wide Web Consortium) et l'IETF (Internet Engineering Task Force) ont connu un essor important. L'impact de la vague IP sur le monde des télécom a été très fort. De plus en plus de services de communication sont transférés des serveurs HTTP et regroupés dans des portails Web. Ainsi, pour développer et offrir des services autour du Web (e.g, portail, téléphonie sur Internet, plate-forme d'accès à Internet, etc.), de nombreux services sont gérés dans des annuaires LDAP (Lightweight Directory Access Protocol) [148].

Les opérateurs télécoms doivent faire face à une compétition sévère dans le domaine des services Web. Afin de réduire le temps de mise sur le marché des services et pour différencier leurs offres, ils combinent sur un même portail, des services disponibles sur le marché et des services développés par leur soin. Différents services cohabitent sur une même plate-forme et doivent très fréquemment partager des données communes. Les services d'annuaires disponibles sur le marché sont des "boîtes noires" qui imposent leur propre modèle de données pour gérer leurs données persistantes. Par exemple, le service Net Meeting de Microsoft utilise un serveur LDAP. D'autres services nécessitent l'utilisation de grosses bases de données avec des

transactions et une haute disponibilité des données qui ne peut être assurée que par des SGBDs traditionnels.

L'intégration des technologies LDAP, base de données et systèmes distribués pose un certain nombre de problèmes à la communauté télécoms. Dans [48], nous avons étudié différentes approches possibles pour cette intégration. Au cours de nos travaux, nous nous sommes particulièrement intéressés à l'intégration des technologies LDAP et systèmes distribués, d'une part et des technologies LDAP et base de données d'autre part. Un élément commun de cette intégration est LDAP qui joue un rôle central. Le succès de LDAP est principalement dû à sa simplicité. LDAP fournit un modèle de données flexible et un ensemble de services nécessaires pour développer des services autour du Web (authentification, sécurité, couplage avec TCP/IP, distribution, etc.). Bien adapté aux accès distants à couplage faible, il propose des mécanismes d'authentification très souples.

Un annuaire LDAP est un arbre qui rassemble les relations de contenance entre les objets et qui reflète les limites politiques, géographiques et organisationnelles des données sur le réseau. Tous les organismes de normalisation (ISO, OSI, NMF/Forum) ont choisi de nommer les objets gérés en fonction de leurs relations de contenance avec d'autres objets. Les liens de contenance sont utilisés pour le nommage hiérarchique des objets. Les objets (i.e, les ressources) sont caractérisés par un ensemble de propriétés. Les annuaires sont répliqués et distribués pour permettre le passage à l'échelle, c'est-à-dire la montée en charge du nombre de connexions utilisateurs et la gestion d'un très grand nombre de ressources réparties au niveau planétaire. Ils offrent d'excellentes performances en consultation pour la localisation des ressources.

Même si les annuaires LDAP offrent de nombreux avantages, ils ne répondent pas à tous les problèmes. Dans la suite de cette introduction, nous présentons les différentes problématiques associées à l'utilisation des technologies LDAP existantes. Ces problématiques nous ont motivé à définir un service d'annuaires étendu pour le monde CORBA, à proposer un nouveau langage appelé DQL (Directory Query Language) pour l'interrogation d'annuaires et pour finir à offrir un double accès LDAP/SGBD. Pour conclure, nous présentons le plan de ce chapitre.

1.1 Vers un nouveau service d'annuaires pour CORBA

Les annuaires LDAP existants ne répondent pas aux besoins de certaines applications comme par exemple, les applications de gestion de terminaux mobiles qui nécessitent beaucoup plus de flexibilité et de dynamicité. Pour les besoins de telles applications, un service d'annuaires doit offrir des facilités de localisation et d'interrogation à la fois pour des objets statiques, mais également pour des objets ayant un comportement beaucoup plus dynamique, c'est-à-dire fréquemment mis à jour, ce qui est généralement le cas sur des plates-formes à objets répartis (e.g, CORBA).

Traditionnellement pour gérer des objets CORBA, les applications ont généralement besoin de fonctionnalités offertes à la fois par les services de nommage [117] et de courtage [120]. Ces deux services, proposés par l'OMG, assurent respectivement la transparence à la localisation et

la possibilité d'identifier les objets relativement à leurs valeurs d'attributs. Le premier service permet de trouver des objets en fonction d'un nom symbolique. Le second utilise les propriétés de l'objet et notamment son type. Actuellement, ces deux services sont indépendants et il n'est pas toujours aisé de les utiliser conjointement de manière efficace. Ainsi, il n'est pas possible d'interroger les objets enregistrés dans ces deux services comme le propose LDAP (i.e , interrogation sur le nom et/ou les propriétés).

Une approche possible consiste à utiliser un serveur LDAP pour assurer les accès aux objets CORBA . Il suffit d'enregistrer leur nom et leurs attributs, et de conserver leur proxy comme un attribut spécifique. Toutefois, cette approche pose des problèmes d'inconsistances [135]. En effet, toute modification d'une propriété effectuée sur un objet géré dans un serveur distant doit être répercutée de façon explicite vers le service dans lequel l'objet est enregistré.

Une autre approche consiste à ne pas gérer les attributs dans l'annuaire, mais à référencer l'objet CORBA depuis l'annuaire. Cette solution permet aux utilisateurs de n'accéder aux objets externes que lorsqu'ils en ont besoin et n'entraîne aucune mise à jour inutile dans l'annuaire. Cette solution repose sur l'utilisation du représentant (i.e. proxy) de l'objet pour accéder systématiquement à la version la plus récente. Le coût d'accès aux attributs des objets dans ce cas est évidemment beaucoup plus important.

Du point de vue des performances, la première solution assure de meilleurs temps de réponse lors de l'interrogation des attributs de l'objet car les valeurs sont répliquées et donc stockées localement dans le serveur. Aucun accès au réseau n'est requis lors de l'interrogation du service d'annuaire avec cette solution. La deuxième approche, qui permet de toujours accéder aux dernières mises à jour des objets, est très pénalisante car à chaque fois qu'un utilisateur souhaite accéder à l'attribut d'un objet géré par un proxy, un accès distant doit être effectué.

La réplication introduit quant à elle des risques d'inconsistances et par conséquent engendre une mauvaise qualité de service si cela n'est pas bien géré au niveau de la plate-forme. Ces risques sont accrus lorsque les problèmes d'inconsistances doivent être gérés de façon explicite par les programmeurs. Ils sont très pénalisants pour les applications nécessitant beaucoup de dynamicité. Par exemple, la propriété caractérisant la localisation d'un objet représentant un téléphone mobile peut être modifiée très fréquemment lorsque le mobile se déplace. Maintenir à jour cette information dans l'annuaire nécessite beaucoup de mises à jour, qui d'une part n'intéressent pas toujours les utilisateurs et qui de surcroît ont un impact conséquent sur les performances. En effet, les accès en consultation sur l'annuaire sont fortement pénalisés par les accès en écriture effectués pour enregistrer les modifications des propriétés des objets CORBA ce qui peut être néfaste pour certaines applications, comme les applications temps réel par exemple.

S'il est clair que la première approche est bien mieux adaptée aux objets qui n'ont pas un comportement dynamique, la seconde est mieux adaptée aux objets ayant un comportement dynamique. Actuellement, si le modèle LDAP répond bien à la première approche, il n'est pas adapté à la seconde. Le principal reproche adressé à la norme actuelle est le manque d'interopérabilité, son administration en silo qui rend impossible le partage de données entre domaines hétérogènes.

L'approche que nous avons retenue consiste à fournir un service d'annuaire qui offre à la fois la flexibilité et l'efficacité pour interroger des objets CORBA ayant des caractéristiques différentes, certains objets ayant un comportement dynamique et d'autres très peu mis à jour [49, 50]. L'idée consiste à introduire différents types d'entrées dans l'annuaire afin de capturer la sémantique particulière des ressources gérées. On distingue les entrées locales, externes et références. Le service d'annuaire permet un compromis entre des impératifs de performance et de maintien de cohérence pour les informations qu'il contient. Un des intérêts de l'approche consiste à étendre la norme LDAP existante. Ainsi nous pouvons garder toutes les propriétés inhérentes au modèle d'annuaires. Cette première contribution apporte des solutions au problème du couplage de LDAP avec les plates-formes à objets distribués. Aucun service d'annuaires ayant été proposé pour CORBA, nous pensons que notre service pourrait compléter la panoplie des services proposés par l'OMG.

1.2 Vers un nouveau langage d'interrogation d'annuaires DQL

Une autre limitation des annuaires concerne l'interrogation des données. En effet, la majorité des services d'annuaires et de gestion de ressources (CMIS [88], X500 [89], LDAP [148]) proposent traditionnellement aux utilisateurs un protocole pour l'interrogation. Cela limite l'interrogation par des non-spécialistes. Nous pensons que des langages sont nécessaires pour ces protocoles, comme cela a déjà été fait dans le domaine des bases de données avec le langage de requêtes SQL et le protocole RDA (Remote Data Access [90]). LDAP URL constitue une première tentative dans cette direction. Il permet l'interrogation à travers un navigateur Web. Cependant les capacités d'interrogation restent limitées à un sous-ensemble de LDAP. Actuellement, aucune facilité n'est offerte dans LDAP URL pour la création, la suppression et la mise à jour des données de l'annuaire.

Par ailleurs, les capacités d'interrogation de ces annuaires sont limitées. Dans la littérature, des extensions ont été proposées par la communauté bases de données pour étendre les fonctionnalités d'interrogation offertes par le protocole LDAP pour les besoins de certaines applications comme les applications DEN (Directory Enabled Applications) [25, 91]. Les chercheurs de AT&T ont proposé plusieurs opérateurs et une propriété de fermeture qui permet de composer un ensemble de requêtes. Le premier opérateur appelé sélection hiérarchique exploite les liens de contenances entre les entrées de l'annuaire [91]. Le second permet le calcul d'agrégats [91].

De notre côté, nous avons proposé le langage DQL (Directory Query Language). DQL étend les capacités de LDAP avec la recherche d'expressions régulières [53,52]. Notons que la recherche d'expressions régulières est une fonctionnalité déjà offerte par les langages XPath [155] et XQuery [156] pour l'interrogation de documents XML, qui eux-mêmes se sont appuyés sur les travaux réalisés autour des extensions proposées autour de OQL[32] pour le support de documents semi-structurés [5,33,34]. Même si LDAP partage la flexibilité des modèles semi-structurés et qu'il est possible de simuler un arbre de nommage, il n'est pas aisé de prendre en compte la notion de portée généralement définie dans le langage LDAP, ni de nommage hiérarchique. Par ailleurs, les langages XPath et XQuery n'adressent pas le problème de la distribution des données semi-structurées et ne permettent pas de déréférencer un lien de façon

transparente. Tous ces aspects sont étudiés dans DQL.

DQL permet d'exprimer des requêtes simples faciles à évaluer et qui ne peuvent être exprimées simplement avec une requête XPath comme nous le montrerons dans la section 3.2. XQuery est un langage trop complexe à utiliser par des non-spécialistes. Par ailleurs, DQL est un langage complet. Il permet la création, la suppression, la mise à jour et l'interrogation d'objets CORBA via l'annuaire de façon simple. DQL comprend non seulement la sémantique des données qu'ils gèrent (i.e, les entrées locales ou externes), mais également la création dynamique d'objets CORBA. Nous avons également proposé une interface par navigation qui permet aux non-spécialistes une interrogation plus conviviale des données de l'annuaire. En effet, il n'est pas toujours facile de se rappeler la racine du sous-arbre de nommage que l'on souhaite interroger.

Au-delà des aspects langages, nous nous sommes particulièrement intéressés au modèle d'exécution de DQL. En effet, la gestion de la distribution et en particulier la distribution à granularité fine induite par la gestion d'objets distants de notre annuaire, impose un certain nombre de contraintes au niveau du modèle d'exécution des requêtes. Les entrées externes et références de notre annuaire pointent respectivement vers des objets distants gérés dans des serveurs CORBA et vers des services d'annuaires. Le nombre d'objets distants étant potentiellement très important, il est nécessaire d'assurer la transparence à la distribution pour les utilisateurs, ce qui n'est généralement pas assuré dans les serveurs LDAP traditionnels, ni par les évaluateurs de requêtes XPath et XQuery qui travaillent sur un document XML à la fois et ne traversent pas les liens vers les autres documents. La distribution a également un impact important sur la probabilité de pannes qui augmente avec le nombre de données distantes. Nous avons donc défini un modèle d'exécution qui prend en compte ces contraintes et propose des résultats partiels afin de gérer les pannes éventuelles d'un ou plusieurs serveurs [52].

1.3 Vers un double accès LDAP/SGBD

Nos derniers travaux de recherche concernent le partage de données communes entre des serveurs LDAP et des systèmes de gestion de bases de données. Pour assurer ce partage, nous avons proposé un wrapper (i.e, adaptateur) dont l'objectif est d'offrir un accès à des données relationnelles via le protocole LDAP, et ce sans modifier la structure des relations utilisées par les clients du SGBD. En effet, la mise en place d'un tel wrapper au-dessus d'un SGBD relationnel doit limiter au maximum l'impact du double accès LDAP-SGBD sur les applications qui utilisent les données relationnelles. Dans ce cas, les utilisateurs ne souhaitent généralement ni modifier, ni réécrire ces applications à cause du coût prohibitif de cette opération. Un des intérêts de l'approche repose sur les mécanismes de génération automatique de wrapper LDAP [51]. Les données relationnelles ne sont pas exportées dans un modèle pivot, elles sont directement accédées via le protocole LDAP. Actuellement, il n'existe pas à notre connaissance de travaux similaires dans la littérature.

Dans ce chapitre, nous avons choisi de décrire le service d'annuaires étendu que nous avons défini pour le monde CORBA, ainsi que son langage d'interrogation DQL et son modèle d'exécution associé. Nous comparons nos travaux avec des approches similaires, puis nous

concluons. Afin de mieux comprendre les extensions que nous avons apportées à la gestion d'annuaires LDAP, nous rappelons succinctement dans une première section les concepts de base du modèle LDAP. Pour une description plus détaillée du modèle LAP se référer au chapitre 2 de la thèse de Thierry Delot [53] ou alors à la norme LDAP [148].

2 Le modèle LDAP

LDAP a rencontré un réel succès ces dernières années, notamment dû à l'avènement de l'Internet. Comparativement aux Systèmes de Gestion de Bases de Données (SGBD) traditionnels, qu'ils soient relationnels ou objets, LDAP propose beaucoup plus de flexibilité et de simplicité, qu'il s'agisse de la gestion du schéma, de celle des données ou de l'interrogation de celles-ci.

- **Le modèle LDAP**

LDAP est composé d'un modèle d'information et d'une API pour manipuler, et notamment interroger, les objets (i.e. ressources) qui sont gérés dans un système. Le modèle d'information de LDAP repose sur le concept d'entrée qui regroupe un ensemble d'associations attribut – valeur(s). Ces entrées sont organisées de manière hiérarchique dans le Directory Information Tree (DIT). Chaque entrée[6] possède un nom relatif, ou RDN (Relative Distinguished Name), permettant d'identifier un objet dans un contexte de nommage donné, et un nom global, ou DN (Distinguished Name), identifiant l'objet de manière unique dans tout l'annuaire. Ce nom global est obtenu en réalisant la concaténation des noms relatifs depuis l'objet considéré jusqu'à la racine. Un exemple d'annuaire est donné dans la Figure 5. Les noms d'attributs C, L, O, OU, utilisés pour définir les noms relatifs, représentent respectivement des pays (*country*), des localisations, des organisations et des départements (*organization unit*). Les noms relatifs commençant par CN (*Common name*) représentent soit des objets de type "Personne", soit des objets de type "Périphérique". Dans cet exemple, la personne "John Jones" a pour nom relatif : {CN = John Jones} et {C = GB, L = Winslow, CN = John Jones} comme nom absolu.

[6] Une entrée représente un objet géré du système (i.e. une ressource).

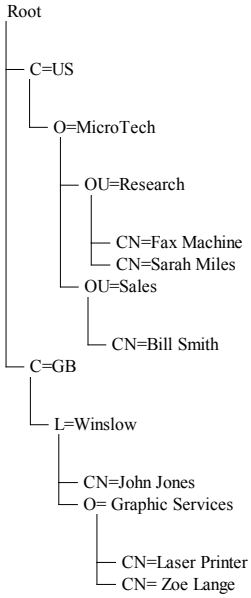


Figure 5 : Exemple d'annuaire LDAP

La notion de schéma dans LDAP est très souple. Le schéma d'une entrée est défini par sa classe d'objets (*object class*), terme utilisé dans la littérature LDAP pour désigner les types de données gérés dans l'annuaire. Les classes d'objets sont standardisées pour faciliter l'interopérabilité des applications manipulant des annuaires. Une classe d'objets définit pour chaque entrée l'ensemble des attributs obligatoires et des attributs optionnels. Chaque attribut possède un nom et/ou un nom alternatif comme par exemple CN, pour "*Common Name*", ainsi qu'un type qui est, soit une chaîne de caractères, soit des champs binaires pour le stockage de données complexes comme des images. Pour définir un schéma LDAP, on commence par donner la liste des attributs pouvant apparaître dans une classe d'objets, puis la liste des classes d'objets. Un héritage simple est possible entre les classes d'objets. Le schéma de données est un peu plus riche qu'un schéma de structure. Aucune règle ne définit en effet les relations de contenance autorisées entre les objets ou les classes d'objets.

La gestion d'un annuaire LDAP peut être répartie entre plusieurs sites. Pour cela, LDAP propose le concept de *referral*, un type d'entrée particulier permettant de référencer, dans un annuaire donné, un sous-arbre géré par un serveur LDAP distant. Ces entrées possèdent un attribut particulier dans lequel est stocké l'adresse du serveur distant, ainsi que le nom global de l'objet auquel on fait référence dans ce serveur distant. Cette notion est utilisée pour gérer une distribution à forte granularité, elle correspond à la fédération de plusieurs espaces de nommage gérés par des serveurs de nommage distincts. Le support de la distribution dans LDAP est très important car la gestion centralisée d'un annuaire de grande taille peut poser des problèmes de performance d'accès aux données. De plus, un annuaire centralisé risque de ne pas résister à la montée en charge du nombre d'utilisateurs. Une solution à ce problème consiste à partitionner les annuaires par pays, par région, ou par entreprise d'une même filiale afin de gérer ces différentes sous-parties dans des serveurs séparés.

- **Interrogation d'un annuaire LDAP**

Dans LDAP, les entrées sont interrogées à l'aide d'une API de bas niveau ou par l'intermédiaire

d'un navigateur WEB. L'API offre des primitives pour créer, supprimer ou modifier une entrée de l'annuaire, ainsi qu'une primitive *"ldap_search"* pour interroger dans son ensemble un annuaire. Le langage d'interrogation LDAP repose sur les notions de portée et de filtre décrites dans la Figure 6.

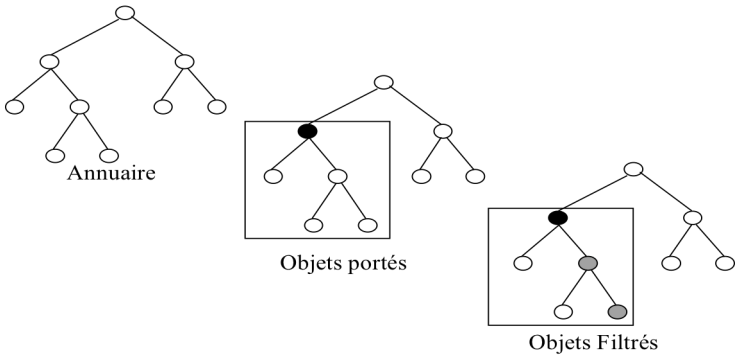


Figure 6 : Définition de la portée et du filtre

L'opération de portée (scope) identifie un sous-arbre sur lequel porte la requête. L'utilisateur spécifie pour cela le DN de l'objet qui sera considéré comme la racine du sous-arbre et le (ou les) niveau(x) à considérer. Les différents niveaux proposés dans LDAP sont *"base"*, *"one"* et *"sub"* et permettent respectivement de désigner uniquement la racine de l'arbre, la racine et ses fils à un seul niveau, ou tout le sous-arbre défini par la racine. Le "scope" définit l'ensemble des objets portés par la requête. Sur cet ensemble d'entrées, l'utilisateur peut appliquer un filtre (i.e. formule booléenne) et récupérer ainsi toutes les entrées de l'annuaire vérifiant le filtre (i.e. objets filtrés). Un filtre est composé d'un ensemble de critères de sélection reliés par des connecteurs logiques; chaque critère de sélection teste la valeur d'un attribut ou l'existence d'un attribut sur une entrée.

3 Le service d'annuaires pour CORBA

Dans cette section, nous présentons les caractéristiques de notre service d'annuaire, c'est-à-dire son modèle étendu d'information et de distribution. Ensuite nous présentons son langage d'interrogation DQL et son modèle d'exécution associé. Pour conclure, nous comparons notre approche avec les approches existantes.

3.1 Le modèle étendu de l'annuaire

Notre annuaire étend LDAP [148], le standard proposé par l'IETF (Internet Engineering Task Force) pour modéliser et interroger des annuaires. Le nommage des données de l'annuaire reste identique à celui proposé dans LDAP. Cependant, afin d'assurer un maximum de flexibilité dans la gestion d'objets CORBA, en fonction de leurs caractéristiques, nous introduisons dans le service d'annuaire trois types d'entrées différents qui sont décrits ci-dessous et représentés dans la Figure 7:

- les *entrées locales*, représentées en gris dans la Figure 7, sont définies pour enregistrer

dans un annuaire des objets CORBA dont les attributs ne sont pas fréquemment modifiés. Pour ce type d'entrées, les attributs de l'objet CORBA sont répliqués dans le service d'annuaire. Ce mécanisme correspond à celui utilisé par le service de courtage. Dans le service d'annuaire, les entrées locales peuvent être utilisées pour représenter de l'information statique, qui sera peu modifiée et éviter ainsi le surcoût induit par la gestion d'objets distants.

- les *entrées externes*, en blanc dans la Figure 7, sont définies pour la gestion des objets CORBA avec un comportement dynamique, c'est-à-dire que ces objets sont fréquemment modifiés, ou du moins susceptibles de l'être. Pour de telles entrées, l'objet CORBA sous-jacent est référencé et aucune propriété de l'objet n'est répliquée dans le service d'annuaire afin de ne pas avoir à répercuter sur celui-ci les fréquentes mises à jour potentielles de l'objet. Pour accéder aux propriétés des objets enregistrés dans le service d'annuaire à l'aide d'entrées externes, le protocole IIOP (Internet Inter-Orb Protocol) est utilisé.
- les *entrées références*, en noir dans la Figure 7, sont définies pour le support du modèle de distribution. Ces entrées correspondent aux références utilisées dans les services CORBA de nommage ou de courtage, afin d'assurer la fédération entre plusieurs services homogènes, ou dans LDAP pour référencer des annuaires distants. Ces entrées ne sont pas typées avec une interface IDL mais avec le mot clé REFERRAL. Nous ne détaillerons pas l'utilisation de ce type de références dans la suite car elles sont gérées comme dans LDAP.

Chaque entrée de type externe ou locale possède un attribut spécifique, le proxy, qui contient la référence de l'objet CORBA représenté dans l'annuaire par cette entrée. Même si cet attribut n'est pas indispensable pour les entrées locales, il peut être utile pour programmer pour ce type d'entrée des politiques de rafraîchissement (e.g, tous les jours, tous les mois, etc.).

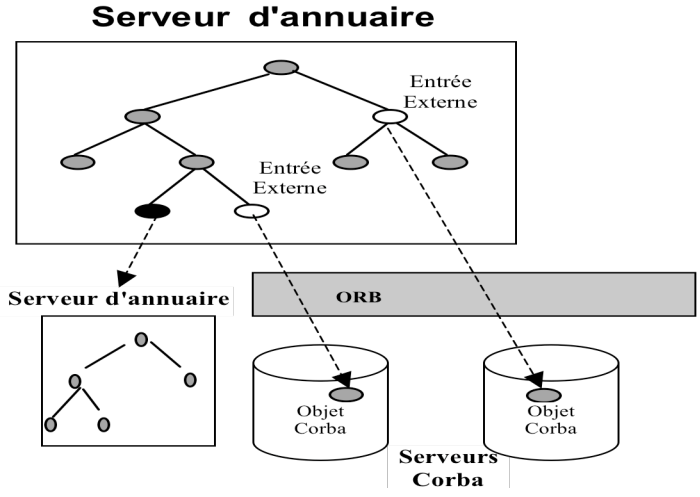


Figure 7 : Le service d'annuaires

Bien que les entrées externes et les entrées locales de notre service d'annuaire partagent le même modèle d'information, c'est-à-dire qu'elles possèdent toutes deux un ensemble de couples attribut-valeur plus ou moins important suivant le type d'entrée, la manière dont sont gérés ces deux types d'entrées par le gestionnaire de requêtes proposé dans le service d'annuaire est

complètement différente. Il appartient en effet à ce composant de l'architecture d'assurer aux utilisateurs une transparence totale lors des accès aux propriétés de l'objet. Dans la suite de cette section, nous décrivons le langage d'interrogation du service d'annuaire et discutons les contraintes spécifiques qui doivent être gérées par le gestionnaire de requêtes du fait de l'introduction d'objets CORBA dans un annuaire.

Dans la philosophie CORBA, chaque composant, ou objet, est décrit par une interface contenant un ensemble d'attributs et/ou d'opérations, définie en IDL (Interface Definition Language). Ces interfaces assurent la transparence avec la représentation physique des objets (e.g, bases de données, fichiers, objets Java ou C++, etc.), ainsi que leur distribution. Pour typer les entrées d'un service d'annuaire pour objets CORBA, nous n'utilisons pas la notion de classe d'objets proposée dans LDAP mais directement les interfaces IDL qui définissent le schéma des objets CORBA enregistrés dans l'annuaire. Chaque entrée possède donc un attribut caractérisant l'interface IDL de l'objet correspondant.

De plus, afin d'assurer un minimum de flexibilité pour les objets gérés, l'utilisateur peut souhaiter ajouter de nouveaux attributs aux objets au cours du temps. Ceci est particulièrement difficile à gérer sur des plates-formes CORBA car l'évolution de schéma est problématique. Elle entraîne la modification de l'interface IDL associée à l'objet. Généralement, il est impensable d'arrêter la plate-forme chaque fois qu'une modification survient afin de recompiler l'application. Afin d'éviter de telles contraintes, des attributs supplémentaires peuvent donc être ajoutés aux entrées de l'annuaire. Ces attributs optionnels ne sont pas gérés par le serveur dans lequel a été créé l'objet CORBA mais en local, dans le serveur d'annuaire, en complément des attributs de l'objet. Comme cela est fait dans LDAP, ces attributs sont stockés sous la forme de chaînes de caractères et peuvent être multi-valués. Ce mécanisme permet à l'utilisateur d'ajouter des propriétés aux objets CORBA s'il le désire, contrairement à ce qui est défini dans le service de nommage de CORBA, où seule la correspondance entre un nom et un objet CORBA est stockée. Par exemple, de nouvelles adresses électroniques peuvent être ajoutées à un utilisateur représenté à l'aide d'une entrée de l'annuaire, et ce même si aucun attribut n'avait été défini dans l'interface IDL pour stocker ce type de données.

3.2 Le langage d'interrogation DQL

Dans cette section, nous présentons par quelques exemples le langage d'interrogation de notre annuaire DQL[7] [53,52]. Les exemples de requêtes sont exprimés sur le DIT de la Figure 5 et illustrent la syntaxe et la sémantique de DQL, qui est proche de celle des langages pour les bases de données comme SQL ou encore OQL, et plus particulièrement des extensions réalisées autour de OQL pour le support de documents semi-structurés tels que Lorel [5], OQL-Doc [33] et son extension [34]. DQL ne gère pas uniquement l'interrogation d'objets mais également la création et la destruction d'objets distants.

Le premier exemple de requête présenté permet de créer un objet de type *Peripheric*. Comme plusieurs DIT peuvent être définis dans un même service d'annuaire, il est nécessaire lors de la création d'un objet de préciser le nom du DIT auquel il doit être attaché. L'objet créé

est identifié par son nom relatif (ou *rdn*), "*cn=LaserPrinter*" dans notre exemple, et attaché à l'objet dont le nom global (ou *dn*) est précisé, ici "*c=GB, l=Winslow, o=Graphic Services*". Les contraintes à vérifier lors de la création d'un objet sont : (i) l'existence de l'objet auquel est attaché l'objet créé, et (ii) l'unicité du nom global attribué à l'objet créé dans l'annuaire.

Création *:*

```
CREATE IN DIT_1 LocalEntry
AT DN ={c=GB,l=Winslow,o=GraphicServices}
RDN = {cn=LaserPrinter} OF TYPE Peripheric
WITH VALUES ( id : 1, name : 'Diva');
```

Pour créer dynamiquement un objet, l'évaluateur doit : (1) vérifier l'existence de l'interface correspondante, (2) trouver le serveur, ou plus précisément sa 'factory', qui va gérer l'objet à créer. Pour ce faire, plusieurs approches sont possibles : gérer chaque interface IDL dans un serveur spécifique ou spécifier explicitement le serveur cible. Deux options sont également possibles lors de la création d'un objet distant afin de l'enregistrer soit dans une entrée locale, soit dans une entrée externe. Dans l'exemple précédent, nous avons précisé à l'aide du mot clé *LocalEntry* que les attributs de l'objet créé seraient répliqués dans le service d'annuaire.

DQL permet également d'interroger les objets gérés dans un annuaire. Pour cela, il s'appuie comme le protocole LDAP sur les notions de portée et de filtre. La clause GET du langage permet à l'utilisateur de sélectionner les valeurs des attributs qui l'intéressent, les clauses SCOPE et FILTER permettent respectivement de spécifier la portée de la requête et le filtre à appliquer aux objets concernés par l'opération de portée. Dans la Figure 8, nous proposons deux exemples de requêtes dont la sémantique est compatible avec celle définie par le protocole LDAP. La première requête retourne tous les attributs des objets appartenant à l'organisation MicroTech située aux Etats-Unis qui possèdent un attribut *name* dont la valeur est "Bill*". La requête 2 illustre la possibilité d'interroger les objets de l'annuaire et de leur appliquer un prédicat. Ainsi, le prédicat *PresentAttribute("name")* de notre deuxième exemple est appliqué à chaque entrée de l'annuaire, puisque aucun *dn* n'est spécifié (i.e. DN={}), et sélectionne toutes celles qui possèdent un attribut *"name"*.

Requête1 *:*

```
GET *
SCOPE DN = { C=US, o= Microtech } LEVEL = All
FILTER name = "Bill*"
```

Requête2 *:*

```
GET name
SCOPE DN = {} LEVEL = All
FILTER PresentAttribute("name")
```

Figure 8 : Exemples de requêtes DQL simples

Dans LDAP, le résultat d'une opération de recherche est composé de la liste des entrées qui vérifient l'opération de portée et satisfont le filtre spécifié. Cette liste hétérogène peut contenir des entrées appartenant à plusieurs classes d'objets différentes, ou plusieurs interfaces IDL dans le cas du service d'annuaire présenté dans cet article. Cependant, tous les liens de contenance entre les objets du DIT sont perdus dans le résultat de la requête. Par ailleurs, le langage de

[7] CMIS-L était l'ancêtre de DQL. CMIS-L est le pendant de DQL dans le monde ISO.

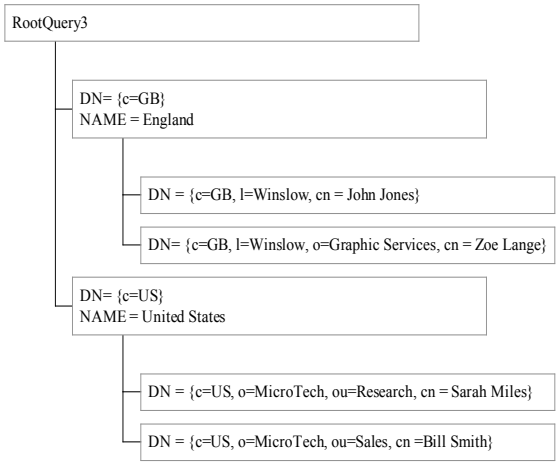
requêtes LDAP ne fournit pas les moyens d'interroger les objets en fonction des liens de contenance définis dans le DIT. Il n'est pas possible de retrouver, pour chaque entreprise, les employés qui gagnent plus de 10000 euros. Dans LDAP, plusieurs requêtes doivent donc être exécutées. Dans un premier temps, on recherche le *dn* de chaque entreprise. Puis pour chaque entreprise, on enverra une autre requête pour chercher les employés. D'autre part, comme les liens de contenance entre les objets sont perdus dans le résultat d'une requête, l'utilisateur doit a posteriori exploiter le *dn* pour retrouver les liens entre les entrées de l'annuaire. Toutefois, cette opération est à la fois fastidieuse et coûteuse.

Pour interroger le DIT en fonction des liens de contenance qui existent entre les objets, DQL étend LDAP en autorisant la recherche d'expressions régulières. Pour définir des expressions de chemins avec définition de variables, une clause FROM a été ajoutée dans DQL. L'utilisation de cette clause FROM est illustrée dans la Figure 9. La requête 3 renvoie l'attribut nom et le nom global de chaque pays, et pour chacun de ces pays le nom de leurs employés. La requête 4 retourne quant à elle le nom global, les valeurs des attributs *name* et *email* de toutes les personnes appartenant à la Grande-Bretagne.

Requête3 :

```
GET c.getDn(), c.name, p.getDn()
SCOPE DN = {} LEVEL = All
FROM Country(c).*.Person(p)
```

Résultat :



Requête4 *:*

```
GET p.getDn(), p.name, p.email
SCOPE DN = {c=GB } LEVEL = 2
FROM *.Person(p)
```

Résultat :

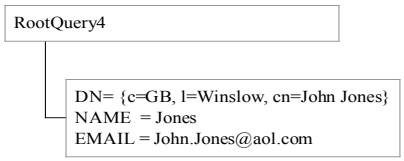


Figure 9 : Requêtes DQL présentant des fonctionnalités avancées

En autorisant la définition de variables de chemin, des résultats complexes peuvent être

calculés à l'aide d'une requête DQL ce qui n'est pas possible en XPath. Même pour des requêtes simples, comme la requête 2 définie la Figure 8, XPath ne retourne pas le résultat escompté. Par exemple, la requête XPath //*[@name] retournera bien l'ensemble des entrées de l'annuaire disposant d'un attribut *name*, mais pour chaque entrée retournera également l'ensemble de ses attributs et nœuds-fils, c'est-à-dire les entrées situées sous-elle dans le DIT, même si ces entrées ne disposent pas d'un attribut *name*. La sémantique de LDAP est ici particulière, on ne sélectionne pas des éléments dans un arbre avec leur sous-arbre associé, mais on filtre un ensemble d'entrées, identifié par la portée. Les extensions que nous avons proposées avec la définition de variables de chemin et de sélection permettent de filtrer un ensemble d'entrées sur leur lien de contenance et de garder dans le résultat la sélection les liens de contenance qui les unissaient. Ces extensions sont similaires aux extensions proposées dans OQL pour le support des documents semi-structurés [5,33,34]. DQL reste un langage simple à utiliser et propose un compromis pragmatique entre XPath et XQuery adapté au contexte des annuaires.

D'autre part, nous avons étendu la notion de niveaux associée à l'opération de portée. Seules trois valeurs (i.e., *base*, *one* ou *sub*) peuvent être utilisées dans LDAP. Dans une requête DQL, l'utilisateur peut choisir précisément la profondeur qu'il souhaite. D'ailleurs, l'objet correspondant à l'entrée *DN = {C=GB, L=Winslow, O=Graphic Services, CN=Zoe Lange}* définie dans l'annuaire de la Figure 5 ne fait pas partie du résultat de la requête 4 à cause de la restriction LEVEL = 2. DQL est donc un langage basé sur des concepts simples et qui permet d'interroger des annuaires.

Afin de faciliter l'interrogation de notre service d'annuaire étendu, nous avons défini une interface interactive. Elle permet à la fois de naviguer à travers l'annuaire et de l'interroger de façon conviviale (voir Annexe B). Cette fonctionnalité est très importante pour permettre l'accès à des non-spécialistes. Les résultats des requêtes DQL sont soit au format standard LDIF (LDAP Information Format), soit au format XML/DSML (Directory Service Markup Language) [59] étendu.

3.3 Le modèle d'exécution de DQL

La particularité de notre annuaire repose sur sa capacité à gérer des entrées externes référençant des objets CORBA dont les propriétés sont dynamiques. L'introduction d'une distribution à granularité très fine pose effectivement de nombreux problèmes en ce qui concerne l'évaluation des requêtes DQL. Par opposition aux langages XPath et XQuery qui s'évaluent en local sur un seul document, l'évaluation du langage DQL doit s'effectuer de façon complètement distribuée sur un très grand nombre d'objets distants.

Les annuaires que nous considérons présentent une différence fondamentale avec ceux manipulés dans le monde LDAP : de nombreuses entrées sont des entrées externes, c'est-à-dire que les attributs de cette entrée sont stockées dans un serveur d'objets distant. La distribution est, dans ce cas, gérée avec une granularité très fine, contrairement à ce qui est réalisé dans LDAP où seuls des annuaires peuvent être référencés. Pour exploiter largement cette distribution des données et des programmes, et éviter à l'utilisateur de devoir émettre une requête par entrée externe, ce dernier doit pouvoir récupérer toute l'information qu'il souhaite

en une seule requête, et ce même si cette requête met en jeu des données réparties sur plusieurs sites distincts. La transparence à la distribution doit donc être assurée. Ainsi, lorsqu'une requête concerne des entrées externes, les serveurs distants gérant les données de ces entrées doivent être interrogés par le serveur d'annuaire ayant reçu la requête de l'utilisateur. Les résultats sont ensuite centralisés au niveau de ce serveur et filtrés avant d'être transmis à l'utilisateur. Le temps d'accès aux objets distants étant naturellement plus important que le temps d'accès aux objets stockés dans l'annuaire, un nouveau problème se pose : comment rester performant malgré la transparence à la distribution ?

Le modèle de stockage de l'annuaire adopté par LDAP est basé sur un partitionnement vertical des entrées : pour chaque attribut qui apparaît dans l'annuaire un index est défini. L'évaluation classique proposée par LDAP semble peu adaptée pour notre service d'annuaire. En effet, elle induit le viol de l'autonomie des serveurs CORBA et introduit de nouveaux problèmes de mise à jour des index. Un modèle d'évaluation de requêtes 'ad-hoc' est donc nécessaire pour interroger le service d'annuaire. Dans [52], nous avons proposé un modèle d'exécution pour évaluer de manière performante les requêtes DQL. Ce modèle est bien adapté au modèle d'annuaire que nous proposons. Il est basé sur la génération de messages asynchrones du serveur d'annuaire vers les serveurs CORBA gérant les objets des entrées distantes. Ces messages asynchrones permettent de rapatrier les données gérées dans l'annuaire, sous la forme d'entrée externe dans le service, l'évaluateur de requêtes peut ensuite appliquer sur ces données le filtre spécifié par l'utilisateur.

D'un point de vue modélisation, l'évaluateur de requêtes gère en fait, lors de l'évaluation d'une requête DQL, un arbre composé de nœuds lents, qui correspondent aux entrées externes, et de nœuds rapides qui correspondent quant à eux aux entrées locales. En effet, les temps d'accès aux données gérées dans les entrées locales et externes varient de manière significative et par conséquent différentes stratégies de parcours d'arbre (e.g, en largeur, en profondeur) peuvent être utilisées. Afin de déterminer le meilleur compromis entre les performances et l'utilisation du réseau, nous avons mis en place un algorithme de simulation qui intègre les facteurs de sélectivité des filtres, les temps éventuels de démarrage des serveurs distants. Cet algorithme permet ainsi de déterminer pour les requêtes avec des parcours de chemins s'il vaut mieux lancer des traitements éventuellement inutiles puisque le filtre n'a pas encore pu être évalué sur un nœud du chemin ou poursuivre l'évaluation d'une autre branche de l'arbre. Par exemple, nous avons étudié pour différents contextes d'évaluation soit de type intranet ou internet quelle était la meilleure stratégie d'évaluation de requêtes. Nous avons étudié plus particulièrement les stratégies en profondeur d'abord car elles permettent d'obtenir très rapidement une réponse valide à la requête, par opposition aux stratégies en largeur d'abord qui retarde l'obtention d'un chemin complet. Nous avons étudié trois stratégies. La première est la stratégie en profondeur simple, à chaque fois que l'on rencontre un nœud distant on envoie le message asynchrone, puis on continue le parcours en profondeur. On attend ensuite le résultat de tous les appels asynchrones pour conclure. La seconde est la stratégie en profondeur avec nettoyage, ici on s'appuie sur la stratégie simple, mais on n'attend pas tous les résultats des appels asynchrones. Par exemple, si on a pu détecter qu'aucun fils ne satisfaisait les conditions, on peut élaguer l'arbre plus vite. La dernière est la stratégie en profondeur avec blocage, ici on ne lance pas d'appels asynchrones tant qu'il existe des nœuds locaux, on repart en largeur, les

nœuds distants sont traités à la fin, cette dernière stratégie permet d'éviter l'encombrement de la bande passante avec des messages inutiles. Les résultats des tests sont décrits dans la thèse de Thierry Delot [53].

La localisation des serveurs d'objets est assurée par les mécanismes proposés par CORBA. Les messages asynchrones reposent quant à eux sur les mécanismes d'invocation dynamique qui reposent sur les services InterFace Repository (IFR) et Dynamic Invocation Interface (DII) fournis par CORBA. En effet, toutes les interfaces IDL qui doivent être connues par le serveur, ne sont pas obligatoirement toutes définies lors de la phase de compilation du système. Ces mécanismes permettent donc à l'utilisateur d'introduire de nouvelles interfaces IDL à l'exécution, sans avoir besoin de recompiler tout le système.

Un autre problème induit par la gestion d'entrées externes dans un annuaire, et donc de la distribution à une granularité fine, est celui des pannes. En effet, la probabilité d'interroger un serveur en panne ou inaccessible augmente proportionnellement au nombre d'entrées externes. Ceci est d'autant plus gênant que la transparence à la distribution est assurée. Un utilisateur n'est donc pas assuré de trouver la totalité de son résultat pour une requête donnée. Notre approche pour traiter ce problème a consisté à introduire le concept de réponses partielles dans le modèle d'exécution de DQL. L'idée sous-jacente est de s'appuyer sur le schéma de structure que l'on possède toujours au niveau de l'annuaire. Même si le serveur gérant l'entrée externe correspondante est en panne, il est toujours possible de fournir un résultat minimal à l'utilisateur, et ce dans un délai fixé par l'utilisateur.

Dans le cas d'une requête DQL cherchant un chemin dans l'annuaire, on peut présenter à l'utilisateur un chemin potentiellement valide composé de toute l'information collectée, y compris les raisons pour lesquelles le calcul du résultat de la requête n'a pu être effectué en totalité (e.g, timeout expiré, serveur inaccessible, etc.). Un chemin potentiellement valide est un chemin de données pour lequel le filtre sur un des nœuds du chemin n'a pu être vérifié. Pour les nœuds correspondant à des serveurs en panne, seule l'interface IDL du nœud sera alors vérifiée, c'est-à-dire le filtre imposé sur le schéma. Le filtre sur les valeurs d'attributs ne peut quant à lui pas être vérifié. Ce mécanisme est intéressant pour certains types d'applications afin de déterminer par exemple l'ensemble des tâches à relancer en cas d'exécution incomplète d'une tâche et éviter ainsi de relancer systématiquement la totalité du traitement.

Pour des raisons d'espace, nous ne présentons pas en détail le modèle d'exécution de DQL, qui permet également de retrouver des chemins dans un annuaire, ce que ne permet pas LDAP. Le lecteur intéressé peut se référer à la thèse de Thierry Delot [53] et à l'article [52] pour plus d'informations.

3.4 Comparaison avec d'autres approches

Dans la littérature, d'autres approches ont été proposées pour gérer et interroger des objets CORBA. Dans cette section, nous nous intéressons particulièrement à l'Object Query Service (OQS)[118], le service de requêtes proposé par l'OMG, ainsi qu'à l'environnement CORBAScript/CORBAWeb [109].

- **Object Query Service (OQS)**

L'OQS propose un gestionnaire de requêtes distribuées sur les plates-formes de l'OMG. L'OQS est conforme au modèle objet défini par l'OMG et permet à un utilisateur ou un objet quelconques d'invoquer des requêtes sur des collections d'objets. L'OQS est indépendant de tout langage de requêtes. Dans la spécification de ce service de requêtes, les mécanismes d'évaluation, d'indexation ou d'optimisation ne sont pas spécifiés et peuvent varier de manière significative en fonction des environnements. L'OQS définit simplement des mécanismes pour envoyer des requêtes à des serveurs de données et permet ainsi son optimisation. Toutefois, l'OQS présente quelques limites pour gérer efficacement des objets distribués dans des environnements CORBA. En effet, l'approche de l'OQS ne fournit aucun support pour la gestion des pannes des serveurs sous-jacents. De plus, à notre connaissance, aucune implémentation de la spécification définie par l'OMG pour ce service CORBA n'est disponible.

- **CORBAScript - CORBAWeb**

Traditionnellement, les applications à objets distribués sont basées sur une approche statique où chaque application connaît à la compilation tous les types d'objets qui peuvent être manipulés. Une approche très intéressante a été proposée dans [109]. Le but de cette approche est de construire un nouveau type d'applications objets, appelées applications génériques. Ces applications ont comme caractéristique de gérer des objets dynamiquement, sans requérir aucune connaissance lors de la phase de compilation de l'application quant aux types d'objets qui seront gérés.

Deux outils ont été développés à l'université de Lille. Le premier, appelé CORBAScript, est un langage de scripts qui permet de manipuler des objets CORBA distribués dans plusieurs serveurs. Chaque script est composé d'une suite d'instructions telles que l'affichage d'une valeur, l'appel d'une opération, l'affectation d'une variable, le contrôle de flots et la gestion d'exceptions. Ces scripts permettent de manipuler dynamiquement des objets CORBA ainsi que de définir des procédures applicables sur ces objets. Le deuxième outil, l'environnement CORBAWeb, vise à offrir une passerelle générique entre d'un côté un serveur Web et de l'autre les objets CORBA. En utilisant CORBAWeb, un client muni d'un navigateur Web peut naviguer à travers des liens entre les objets CORBA grâce à des URLs générées dynamiquement pour chaque objet. Dans l'environnement CORBAScript-CORBAWeb, les objets CORBA sont gérés de manière complètement dynamique et sont accessibles via un simple navigateur web. Toutefois, même si beaucoup d'efforts ont été portés aux aspects "systèmes distribués", la gestion des données peut être améliorée. En effet, aucun langage de requêtes n'est proposé pour retrouver des collections d'objets.

L'environnement CORBAScript-CORBAWeb fournit donc des outils faciles à manipuler pour gérer des objets (i.e. retrouver, créer, modifier ou détruire des objets) en accédant aux objets distants et en invoquant des méthodes sur ces objets. Néanmoins, les utilisateurs doivent écrire des scripts qui deviennent vite complexes pour les interroger.

4 Conclusion

Dans ce chapitre, nous avons décrit un service d'annuaire adapté à la gestion d'objets CORBA. Le modèle de ce service d'annuaire est très similaire à celui des annuaires LDAP, il conserve notamment sa flexibilité. Ce service d'annuaire permet de gérer à la fois des objets avec un comportement dynamique et d'autres avec un comportement plus statique en faisant un compromis entre efficacité et dynamicité. L'architecture du service est ouverte et dynamique. Elle garantit la prise en compte à l'exécution de nouvelles interfaces IDL et de nouvelles propriétés pour les objets.

Nous avons proposé des fonctionnalités d'interrogation et de navigation, qui permettent une gestion à la fois conviviale et efficace des objets CORBA. DQL offre en effet à l'utilisateur un langage puissant pour interroger les objets gérés dans un annuaire, avec notamment des expressions de chemins et avec la possibilité de définir des variables pour exploiter au mieux l'organisation hiérarchique des données traditionnellement utilisée par les applications de gestion des ressources d'un réseau. DQL permet également la création, la mise à jour et la destruction d'objets sur la plate-forme.

Les nombreuses contraintes induites par la gestion d'entrées externes dans un annuaire ont été prises en compte, notamment en ce qui concerne l'évaluation des requêtes. Un modèle d'exécution adapté à la distribution avec une granularité fine inhérente à la gestion d'objets CORBA dans un annuaire a été proposé. Il assure la transparence à la distribution pour les entrées externes et peut également fournir à l'utilisateur des résultats partiels en cas de panne ou d'inaccessibilité des serveurs d'objets CORBA sous-jacents. Ce prototype a été démontré lors de la conférence BDA en 1998.

Un prototype du serveur d'annuaire a été développé dans le cadre d'un projet entre France Télécom R&D et le laboratoire PRiSM. Une première version de l'annuaire ne gérant que des objets CORBA (et pas d'attributs optionnels) a permis de valider l'approche et les aspects liés au langage DQL comme la transparence à la localisation et les résultats partiels. Le prototype est réalisé en Java avec OrbixWeb3.2/Windows NT Professional Edition, l'ORB de Iona Technologies. OrbixWeb propose une implémentation pour les services d'invocation dynamique et du référentiel d'interfaces utilisés pour l'interrogation des objets CORBA référencés dans l'annuaire sous forme d'entrées externes. L'espace de nommage hiérarchique défini par l'annuaire est construit avec l'API JNDI (Java Naming Directory Interface) qui permet de rester compatible avec LDAP tout en conservant la possibilité d'étendre ses fonctionnalités. La persistance de l'annuaire est assurée par le SGBDOO Versant. Versant est un SGBDOO indépendant des langages et des plates-formes. Son architecture Client/Serveur équilibrée, multi-sessions et multi-threads offre une haute disponibilité des données ainsi que des mécanismes de tolérance aux pannes (FTS). Versant permet également un couplage fort avec Java qui facilite le stockage efficace de l'annuaire construit avec l'API JNDI. En ce qui concerne les objets CORBA correspondant aux entrées externes de l'annuaire, ils peuvent persister soit dans une base de données relationnelle (Oracle), soit dans une base de données objets (Versant), soit dans des fichiers (objets java sérialisés).

Chapitre IV – Protection de la confidentialité des données

Ce chapitre décrit nos activités de recherche récentes entamées à partir de 2002. Ces travaux de recherche sont complémentaires aux travaux de recherche, présentés dans les deux chapitres précédents et menés autour des architectures, des langages de requêtes, des techniques d'évaluation et d'optimisation associées pour l'accès transparent aux données largement distribuées. Il adresse le problème de la confidentialité des données à caractère personnel dans un contexte XML.

1 Introduction

Pendant plusieurs années, nous nous sommes intéressés à offrir un accès transparent aux données largement distribuées et il nous est apparu naturel d'étudier les problèmes de confidentialité de données afin de garantir un accès sécurisé à ces données. En effet, la préservation de la confidentialité est devenue un enjeu majeur pour la majorité des applications, qu'il s'agisse de gestion de données personnelles, de commerce électronique, de systèmes d'information en ligne, d'intelligence ambiante ou plus classiquement de préservation de secrets industriels ou scientifiques. La défiance envers la gestion actuelle des données confidentielles est vue, selon une étude IBM-Harris [87], comme le principal frein au développement de nouvelles applications sur l'Internet. La préservation de la confidentialité est une tâche gigantesque et touche de nombreuses thématiques : chiffrement des données et des communications, détection d'intrusions, contrôle de droits d'accès, anonymisation des données et des actions, fouille de données préservant l'intimité des données, etc.

Nos travaux de recherche se sont centré sur les modèles de contrôles d'accès pour XML. XML est aujourd'hui le standard de facto pour décrire, échanger et disséminer toutes sortes d'informations entre différents acteurs et pour des objectifs très variés. De nombreux chercheurs se sont intéressés à différentes facettes du problème. Tous ces travaux partagent le même point de vue; ils ont choisi de focaliser le contrôle d'accès sur les nœuds d'un document XML (attributs et éléments). Les relations de parenté et de fraternité entre les nœuds ne sont pas considérées comme des éléments de première classe dans leur modèle, et d'autre part, la vue autorisée d'un document est toujours un sous-ensemble du document initial. Cela pose des problèmes majeurs qui vont à l'encontre de deux principes de base, le principe de finalité ou "need-to-know" et le consentement, deux principes édictés par les directives et lois mises en place par les gouvernements pour protéger les informations personnelles. Plus que jamais, il est nécessaire de définir des modèles de contrôle d'accès qui permettent de traduire les lois en pratique.

Au cours de nos recherches, nous avons proposé un nouveau modèle de contrôle d'accès

XML. Notre approche défend l'intégration des associations d'ancêtres et de fraternité entre les nœuds comme des éléments de première classe [72]. Un des intérêts de l'approche est qu'elle est compatible avec les modèles existants qui se focalisent uniquement sur la protection des nœuds, nous avons étendu ces modèles afin d'intégrer la gestion des droits sur associations. Afin de comprendre les motivations qui nous ont conduit à définir ce nouveau modèle de contrôle d'accès, nous présentons dans une première partie un état de l'art des modèles de contrôle d'accès, et plus particulièrement les principes associés au contrôle d'accès XML. Puis dans une seconde partie, nous montrons comment et pourquoi les modèles existants ne permettent pas de traduire certains principes de lois en pratique, sur la base de l'exemple du dossier médical personnel [69]. L'étude des textes de lois nous a permis de caractériser deux classes d'autorisation sur les associations (i.e, la dépersonnalisation des ancêtres et la réduction de chemin) et de prendre en compte la dimension des associations de fraternité (i.e, la décorrélation sélective). Une fois les besoins analysés, nous présentons, dans une troisième partie, notre modèle. Nous décrivons le formalisme à base de règles pour exprimer les classes d'autorisation sur les associations et les deux mécanismes de clonage et de shuffling introduits pour traduire exactement ces associations en une vue autorisée. Puis dans une quatrième partie, nous concluons.

2 État de l'art

L'une des exigences majeures du partage des données entre plusieurs utilisateurs est la protection de ces données contre des atteintes à la confidentialité (divulgations d'information non autorisées), contre des atteintes à l'intégrité (modifications non autorisées) et contre des atteintes à la disponibilité (déni de service). Afin d'assurer cette protection, chaque accès aux données doit être contrôlé et bien évidemment tous les accès non autorisés doivent être impérativement bloqués. Cela est appelé le ***contrôle d'accès***. Le développement d'un modèle de contrôle d'accès repose sur la définition de politiques de contrôle d'accès qui déterminent "***qui a le droit d'effectuer quelle action sur quelle donnée"***. Le modèle veille à ce que les données ne soient accessibles que par des utilisateurs ayant le droit d'y accéder.

2.1 Modèles de contrôle d'accès

Dans la littérature, un très grand nombre de modèles de contrôle d'accès ont été proposés afin de garantir la confidentialité des données. Il existe deux grandes classes de modèles de contrôle d'accès :

- Les *modèles de contrôle d'accès discrétionnaires* DAC (Discretionary Access Control)[113], le créateur d'un objet se voit affecter tous les droits sur cet objet et peut transmettre tout ou partie de ses droits à d'autres utilisateurs, et ce de façon récursive,
- Les *modèles de contrôle d'accès obligatoires* MAC (Mandatory Access Control) [13,21], le plus connu, fixe des niveaux hiérarchiques de sécurité aux données (public, confidentiel, secret…), et des niveaux d'habilitation (public, confidentiel, secret…) aux utilisateurs.

Afin de mieux s'adapter à des organisations particulières, d'autres modèles ont été définis, en particulier, le modèle de contrôle d'accès basé sur la notion de rôle RBAC (Role Based Access Control) [65, 64 ,129] qui affecte des droits à des rôles; un utilisateur peut être autorisé à jouer des rôles différents au cours de sessions différentes. Le modèle basé sur la notion d'équipe TMAC (TeaM Access Control) [144], qui regroupe un ensemble de rôles différents pour former une équipe de travail afin d'atteindre le même objectif. Le modèle de contrôle d'accès basé sur la notion d'organisation (Or-Bac) [5] qui permet d'introduire un niveau d'abstraction permettant d'exprimer la politique de contrôle d'accès indépendamment de son implémentation.

Ces modèles de contrôle d'accès ont été conçus de façon complètement indépendante des modèles de données. Ils fixent la sémantique du contrôle au niveau abstrait, étudient les facilités d'administration d'un ensemble de politiques de contrôle d'accès. Dans la littérature, ces modèles de contrôle d'accès ont été mis en œuvre dans des contextes bien différents que cela soit celui des bases de données relationnelles, des bases de données orientées objets et la protection des documents XML. Quel que soit le modèle considéré, ce qui les distingue véritablement c'est la mise en œuvre des mécanismes de base indispensables. Ces mécanismes sont parfois bien différents d'un modèle de données à un autre. Par exemple, on s'appuie dans les bases de données relationnelles sur le principe de vue SQL [80], par contre dans les bases de données semi-structurées (XML) la presque totalité des modèles de contrôle d'accès reposent sur un ensemble de règles d'autorisation définissant quels nœuds du document XML sont accessibles ou non.

2.2 Modèles de contrôle d'accès XML

Au cours de nos travaux de recherche, nous nous sommes plus particulièrement intéressés aux modèles de contrôle d'accès pour XML. L'apparition de XML comme nouveau standard d'échange d'information à travers le Web a relancé les travaux de recherche sur les modèles de contrôle d'accès. En effet, la structure arborescente des documents XML et la nature semi-structurée des données imposent d'autres exigences au niveau du contrôle d'accès. Les travaux proposés dans la littérature s'appuient principalement sur les modèles DAC [19,18,17,15,42,43,75,103], RBAC [82,84,122,150,158] et MAC [35,134]. Des extensions au modèle DAC ont été proposées dans [98], l'idée consiste à intégrer des autorisations provisionnelles dans le modèle afin que le système puisse contrôler si l'utilisateur est autorisé à accéder à certaines informations seulement et a effectué certaines actions de sécurité.

Bien plus que le modèle de contrôle d'accès lui-même (DAC, MAC, RBAC), l'attention des auteurs s'est plutôt portée sur : (1) la caractérisation du modèle de contrôle d'accès c'est-à-dire la granularités de protection (éléments, attribut, DTD, XMLSchema…), la modélisation des sujets (utilisateur, groupe, rôle…), les modes d'accès (lecture, mis à jour) [18,15,42,43,75,98,103], (2) sur les algorithmes permettant d'implanter ce contrôle [30, 35, 112,125,126,149], (3) sur les canaux de communication et de distribution de l'information (dissémination sélective en mode Push ou Pull) [20,16,111,115] et (4) sur la non corruption du modèle de contrôle d'accès par des techniques de chiffrement ou d'environnements d'exécution sécurisés [22].

Parmi tous ces travaux, nous nous sommes particulièrement intéressés à la caractérisation

du modèle de contrôle d'accès lui-même, c'est-à-dire aux principes de base pour définir la politique de contrôle d'accès et ce quel que soit le contexte mis en œuvre, avec ou sans optimisation. Nous avons constaté que malgré tous les efforts consacrés à la résolution de ce problème, la définition d'une sémantique claire et non ambiguë reste à faire. Ce problème de sémantique n'est pas lié au type de modèle adopté (DAC, MAC et RBAC) mais plutôt lié à la manière dont est défini le contrôle d'accès en XML. Normalement, pour une politique de contrôle d'accès donnée, nous devons avoir une et une seule sémantique qui amène toujours à un résultat cohérent. Même si différentes interprétations sont possibles pour une même situation (i.e, les modèles de contrôle d'accès existants ne rendent pas toujours la même vue autorisée), le modèle ne doit pas générer des résultats imprévisibles ou incohérents vis-à-vis de sa propre sémantique ce qui est malheureusement parfois le cas dans les modèles existants.

Par exemple, supposons le document XML décrit dans la Figure 10 et dont les nœuds sont marqués par le signe positif si le nœud est autorisé et négatif sinon. Pour un même marquage c'est-à-dire pour une même politique de contrôle d'accès, différentes vues autorisées sont générées selon les sémantiques des modèles existants. Cela est décrit dans la Figure 11.

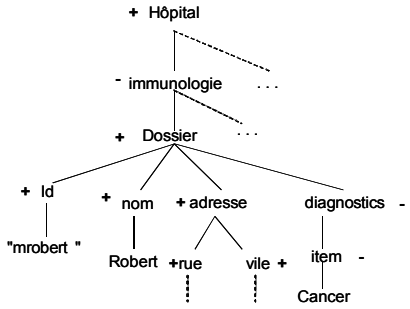


Figure 10 : Le document marqué

La vue autorisée (décrite dans la Fig.(a)), générée par les modèles de contrôle d'accès de Bertino [19], Gabillon [75], Murata [112], Wang [150] et Zhang [158], interdit l'accès à l'élément autorisé dans un sous-arbre interdit. Ces modèles imposent une sémantique qui dit : "si l'accès à un élément est ***non autorisé*** alors l'accès à tous ses descendants est également ***non autorisé"***. Par conséquent, ces modèles de contrôle d'accès contredisent la base de règles d'autorisation définie en interdisant l'accès aux éléments qui étaient autorisés au départ.

D'autres modèles de contrôle d'accès comme ceux de Damiani [42, 43], Kudo [98], Fan [62] et Gabillon [74] permettent l'accès à un élément autorisé dans un sous-arbre non autorisé (c'est-à-dire le sous-arbre est enraciné par un élément non autorisé). Ces modèles peuvent être décomposés en deux sous-catégories. La première sous catégorie est composée des modèles de Damiani [42, 43] et Kudo [98] qui révèlent tous les ancêtres (uniquement leurs tags) non autorisés d'un élément accessible (autorisé), cela afin de préserver la structure du document XML initial. Ces modèles contredisent la base de règles d'autorisation prédéfinie car ils rendent visible ce qui a été interdit au départ. Par conséquent, l'information à protéger est divulguée. La vue autorisée du document XML pour cette catégorie de modèles est présentée dans la Fig.(b).

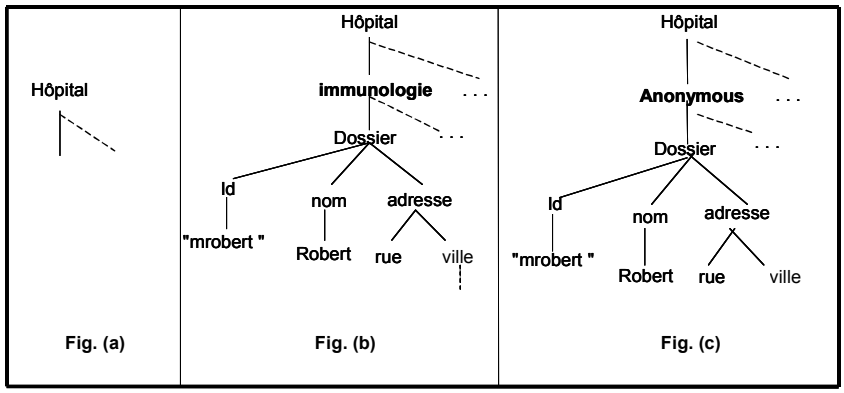


Figure 11 : Les différentes vues retournées selon la sémantique des modèles existants

La deuxième sous-catégorie des modèles de contrôle d'accès de Fan [62] et Gabillon [74] propose de renommer l'élément inaccessible ayant des descendants accessibles (autorisés) afin de résoudre le problème posé par les modèles précédents et de minimiser la divulgation d'information non autorisée. Cela est exprimé dans les modèles de Gabillon [74] par un privilège spécifique (*position*) qui garde l'existence de l'élément sans connaître sa valeur (par le biais du renommage). La vue autorisée du document XML est présentée dans la Fig.(c). Concernant l'élément inaccessible ayant un descendant accessible le modèle de Fan [62] donne une autre alternative qui consiste à supprimer cet élément si cela ne viole pas la structure de la DTD initiale. La sémantique de l'élément inaccessible, dans ce modèle, dépend de la DTD.

3 Problématique & Motivations

Nous constatons que les modèles existants focalisent le contrôle d'accès sur les ***éléments*** et les ***attributs*** et supposent que ***la vue autorisée est un sous-ensemble du document original***, que leur sémantique de contrôle d'accès sur les éléments diffère d'un modèle à un autre particulièrement lorsque qu'il existe un élément intermédiaire non autorisé entre deux éléments autorisés d'un même chemin dans le document XML. Cette différence entre les modèles est fortement liée à la difficulté de ***définir la vue exacte d'un chemin*** qui conduit à un élément autorisé dans les modèles de contrôle d'accès existants. Ces trois aspects (en gras dans le texte) posent de nombreux problèmes car ils restreignent les possibilités quant à l'expression de certaines politiques de contrôle d'accès notamment dans le contexte de la protection des données personnelles.

Aucun des modèles de contrôle d'accès ne s'est intéressé à la protection des associations existant entre les nœuds. Ces associations dans le document XML ont été exploitées, uniquement, pour définir les politiques de propagation pour les règles d'autorisation. Pourtant, le problème est fondamental du point de vue de la préservation de la confidentialité et de l'intimité des données. En effet, les associations ancêtre-descendant et les associations de fraternité dans un document sont, également, porteuses d'informations potentiellement sensibles. Par conséquent, ne pas tenir compte des associations dans la définition d'un modèle du contrôle d'accès conduit à deux problèmes fondamentaux :

- *Divulgation de la classification* : la structure d'un document XML révèle certaines

classifications[8] (e.g. les différents services d'un hôpital dans lesquels les patients sont traités, les types d'activités ou départements de vente d'une compagnie, une répartition socio-économique). C'est pourquoi l'appartenance d'un nœud à un sous-arbre véhicule par défaut sa classification. Bien que l'on puisse cacher l'information portée par la classe, elle n'en demeure pas moins sensible aux attaques statistiques. En effet, souvent la cardinalité d'une classe peut en révéler sa nature. De plus, cacher l'appartenance d'un élément à une classe entraîne de le faire pour chaque élément de cette classe.

- *Filiation uniforme* : les règles d'autorisation exprimées sur les nœuds ancêtres donnent une seule vue autorisée de chemin qui conduit à tous ses descendants. En d'autres termes, il n'y a aucun moyen de délivrer deux vues autorisées différentes du même ancêtre pour ses deux descendants. Par exemple, un patient souhaite cacher le service médical où il était traité, alors qu'un autre patient donne son consentement pour divulguer cette information.

Ces deux problèmes violent deux principes fondateurs imposés par les législations relatives à la protection des données à caractère personnel qui sont le ***need-to-know*** et le ***consentement***. Le principe de finalité ou "need-to-know" limite l'accès à l'information, seule l'information strictement utile à une tâche doit pouvoir être accédée. Le principe de consentement empêche la divulgation d'information sans le consentement explicite du propriétaire de la donnée.

Afin de faciliter la compréhension de l'ensemble de ces problèmes vis-à-vis des modèles de contrôle d'accès XML existants, nous avons choisi de les illustrer sur la base d'une problématique réelle et de grande importance. En effet, une étape significative a été franchie en France avec l'adoption, le 13 août 2004, de la loi relative à l'assurance maladie [105,104]. Cette loi a pour objectif principal la réorganisation de l'assurance maladie et la gestion des dépenses de santé. Parmi les mesures, on envisage la création du ***dossier médical personnel,*** appelé ***DMP***. D'après le bulletin de l'ordre des médecins [24], le DMP "*est une application informatique qui permet de collecter des données individuelles de santé auprès des professionnels, de conserver ces données dans un lieu* ***sécurisé****, de* ***protéger ces données*** *et de* ***gérer des droits d'accès****, enfin de mettre ces données à la disposition des personnes habilitées, simplement et rapidement*".

L'exemple du dossier médical informatisé est intéressant car il pose des questions pertinentes sur la confidentialité des informations hautement sensibles contenues dans ce dossier. En effet, les données sensibles sont des données à caractère personnel encadrées juridiquement par un ensemble de législation[9]. Ces données doivent respecter les principes de finalité et de consentement.

Afin d'illustrer les problèmes, nous avons choisi de raisonner sur une abstraction des dossiers médicaux présentée dans la Figure 12 et une politique de contrôle d'accès dont les règles ont été motivées par la lecture des textes législatifs notamment de HIPAA [83]. Sur la base de cet exemple, nous démontrons les limites et carences des modèles existants et le besoin

[8] Synonyme des catégorisations

[9] La protection des données à caractère personnel aux États-Unis est assurée par la loi Privacy Act de 1974 [124] et en Europe par la directive européenne[9] du 24 octobre 1995 [54].

de protéger les associations dans un modèle de contrôle d'accès XML.

Les dossiers médicaux des patients sont présentés au format XML[10]. Ils sont organisés en service (e.g Immunologie, Psychothérapie). Chaque dossier est composé d'un ensemble de données administratives (numéro de sécurité social, nom, adresse) et de données médicales (actes médicaux, analyses). Les actes médicaux (ActesMed) sont décomposés, à leur tour, en deux types d'actes médicaux : i) les actes médicaux normaux et ii) les actes médicaux sous protocole. D'après l'article R. 1112-2 du code de la santé publique, le consentement du patient est un élément de base du dossier médical. Nous l'intégrons donc comme un élément à part entière du dossier XML.

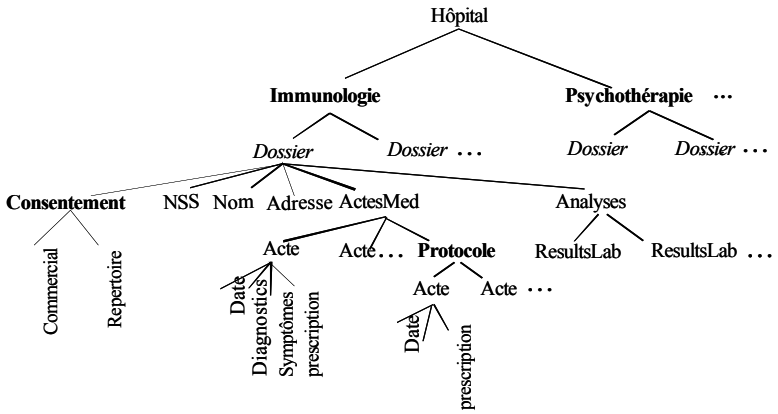


Figure 12 : Dossiers Médicaux

La politique inspirée de HIPPA comprend trois règles d'autorisation importantes de l'hôpital et est représentative des principes de ***consentement*** et de ***need-to-know***. Pour chacune des règles d'autorisation nous montrons comment les modèles de contrôle d'accès traitent la règle.

Les dossiers médicaux sont partagés entre plusieurs utilisateurs (patients, professionnels de santé, laboratoire médical et compagnies d'assurance) ayant des objectifs et des tâches de travail différents.

1. *Règle 1* : *"cacher au groupe d'annuaire[11] le nom du service où les patients sont traités pour ceux qui n'ont pas donné leur consentement".*

La première règle d'autorisation de l'hôpital stipule que le patient (e.g. victime d'une violence domestique, célébrités…) a le droit de cacher le service dans lequel il est traité [83]. Cette règle vise, donc, à respecter la vie privée et la liberté individuelle des patients, en l'occurrence, respecter le principe de consentement imposé par les législations. L'effet de cette règle d'autorisation sur le document XML présenté dans la Figure 12 devrait être de rattacher l'élément dossier du patient en question à un service dépersonnalisé (e.g élément avec un tag anonyme) tout en gardant les autres dossiers médicaux non affectés. Nous appelons cette opération la "*dépersonnalisation des ancêtres*". La restructuration devrait se faire de sorte à

[10] XML a été choisi comme standard d'échange d'information médicale (HL7) [85].
[11] Les utilisateurs responsables de la gestion d'annuaires de l'hôpital.

empêcher l'inférence de la classification initiale. En principe, la vue retournée devrait être comme illustrée Figure 13:

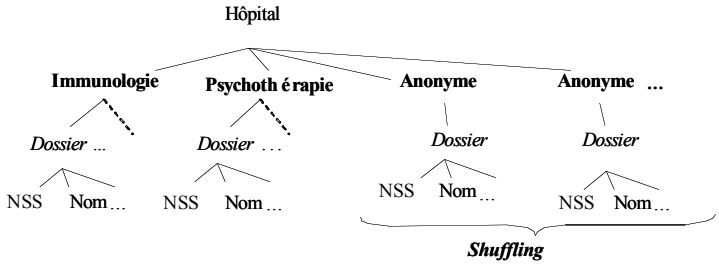


Figure 13 : Dépersonnalisation des ancêtres

Dans ce qui suit nous montrons comment les modèles proposés dans la littérature réagissent à la règle d'autorisation Règle 1 :

1. Les modèles de Bertino [19], Gabillon [75], Murata [112], Wang [150] et Zhang [158] ne permettent pas d'autoriser l'accès à un nœud dans un sous-arbre inaccessible. Ce qui veut dire que, si un nœud donné est inaccessible, tout le sous-arbre porté par ce nœud sera inaccessible. La seule possibilité donnée par ces modèles est de cacher complètement le dossier médical du patient concerné en définissant une règle d'autorisation négative sur l'élément dossier.

2. Les modèles de Damiani [42,43] et de Kudo [98] autorisent l'accès à un nœud dans un sous-arbre inaccessible. Ils permettent donc de définir une règle d'autorisation négative sur l'élément service (e.g, *immunologie*) et une autre règle positive sur l'élément dossier. Par contre, afin de garder la structure initiale du document XML, ils révèlent tous les éléments ancêtres[12] du nœud (descendant) autorisé.

3. Les modèles de contrôle d'accès de Wei Fan [62] et Gabillon [75,74] résolvent le problème de révélation des tags des ancêtres du nœud autorisé en les renommant par un tag *anonyme.* Ces modèles donnent donc la possibilité de définir une règle d'autorisation négative sur l'élément service (*e.g, immunologie*) et une autre règle positive sur l'élément *dossier*. Bien que les modèles cachent le nom du service en question (*e.g, immunologie*), ils ne répondent pas à la règle d'autorisation pour les raisons suivantes :

    - Le renommage du service s'applique à tous les dossiers médicaux des patients de ce service (sans exception) même pour les patients qui ont donné leur consentement. Cela viole le principe du respect de la *liberté individuelle,* imposé par la convention européenne des droits de l'Homme.

    - Si tous les dossiers restent groupés ensemble, même si le nom du service est caché, une personne connaissant un des patients de ce service anonymisé peut inférer le nom du service de tous les patients appartenant à cette classe.

2. *Règle 2* : "*cacher pour les pharmaciens le fait que certaines prescriptions sont*

[12] Ils révèlent uniquement le tag des éléments ancêtres. Les attributs sont supprimés s'ils existent.

administrées dans le cadre d'un protocole".

La deuxième règle d'autorisation indique que le pharmacien n'a pas le droit de savoir si certaines prescriptions sont administrées via un protocole. Cela veut dire qu'il ne faut pas révéler plus d'information que nécessaire pour accomplir la tâche du pharmacien; en d'autres termes, il faut respecter le principe de **need-to-know** imposé par les législations. L'effet de cette règle d'autorisation consiste à supprimer le nœud *protocole* du dossier du patient, ensuite à rattacher tous les éléments *Acte* au-dessous du protocole comme des descendants directs de l'élément *ActesMed* pour qu'ils rejoignent la classe des Actes médicaux ordinaires. Nous appelons cette opération la "*réduction de chemin*". La vue retournée devrait être comme dans la Figure 14.

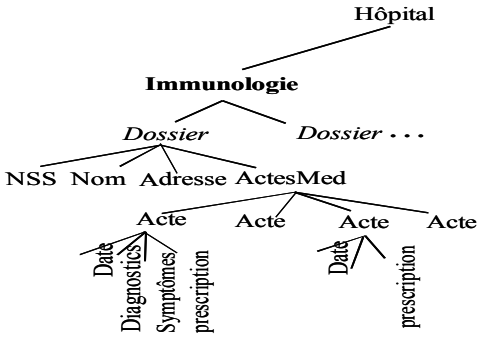


Figure 14 : Réduction de chemin

Tout comme dans le cas de la première règle d'autorisation, les modèles de contrôle d'accès existants échouent à répondre à cette deuxième règle d'autorisation. Ici le renommage de l'élément *protocole* proposé par Wei Fan [62] et Gabillon [74] est inutile car la présence du nœud lui-même révèle une information qui amène à faire la différence entre les types des actes médicaux. Cette information, le pharmacien ne doit pas la connaître pour accomplir sa tâche. Cela viole donc le principe de ***need-to-know***.

3. *Règle 3* : "*cacher aux laboratoires médicaux la corrélation entre les informations médicales et les informations administratives pour chaque dossier*".

Que le patient le veuille ou pas, il n'est pas la seule personne à décider de la gestion de ses données médicales. En effet, pour des raisons d'intérêt public, les législations autorisent la transmission des données médicales des patients à des fins de recherches scientifiques. Par contre, pour tout traitement avec un objectif commercial, la loi HIPAA exige le consentement du patient. Dans cette situation, les données médicales doivent être accessibles pour répondre au principe de ***need-to-know*** concernant les recherches scientifiques et les données administratives (nom, adresse) doivent être accessibles pour répondre au principe de consentement. Le patient souhaite donc donner son consentement pour recevoir des offres commerciales cependant il veut cacher exactement quelles sont les données médicales qui le concernent. Le résultat de cette règle d'autorisation est schématisé dans la Figure 15.

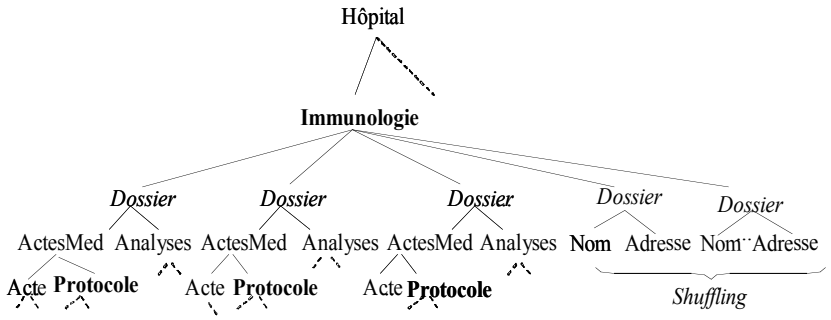


Figure 15 : Décorrélation

Les modèles de contrôle d'accès ne peuvent répondre à cette règle d'autorisation car cela imposerait de définir deux bases de règles d'autorisation pour le même document et pour le même utilisateur (e.g, laboratoire médical). Par conséquent, l'utilisateur obtiendra deux vues séparées du même document XML. Par contre, la conjonction de ces deux vues peut permettre d'inférer la structure initiale du document en comparant l'ordre des éléments dans les deux vues.

En résumé, les modèles de contrôles d'accès ne permettent pas de traduire certains principes fondateurs des textes de lois relatifs à la protection des données personnelles. En effet, tous les modèles proposés imposent une contrainte forte concernant la structure du document XML. La structure de la vue retournée doit être compatible avec la structure du document initial. Cela amène à ce que tous les descendants d'un nœud se comportent de la même façon par rapport à leurs ancêtres. Ce constat nous a encouragé à définir un nouveau modèle de contrôle d'accès pour XML qui intègre la protection des associations comme des éléments de première classe [72] et qui permet de mieux traduire les principes de législation en pratique [69].

4 Modèle de contrôle d'accès XML avec associations

L'objectif de cette section est de présenter succinctement notre modèle de contrôle d'accès qui vise, plus particulièrement, à protéger les associations dans un document XML. Plutôt que de proposer un tout nouveau modèle, l'approche proposée vise à étendre les modèles de contrôle d'accès existants avec de nouvelles règles sur les associations. Puisque chaque modèle de contrôle d'accès adopte de petites variantes concernant les règles de propagation et de résolution de conflit, nous introduisons, dans un premier temps, le modèle de référence pour définir les règles d'autorisation sur les nœuds qui capture les bases communes des modèles de contrôle d'accès existants. Ensuite, nous proposons une extension de ce modèle de référence pour le support des autorisations sur les associations.

4.1 Modèle de référence pour les autorisations sur les nœuds

Les modèles de contrôle d'accès existants partagent des similarités fortes dans leurs conceptions. Leurs différences se situent dans les types de propagation utilisés, les politiques de résolution de conflits adoptés, etc. Généralement, la règle d'autorisation est définie sous forme d'un quadruplet <*Sujet, Objet, Action, Signe*>. Suivant les modèles, le *Sujet* peut prendre plusieurs formes (utilisateur, groupe, rôle, adresse IP, etc.). L'*objet* caractérise la portion du

document ciblée par la règle d'autorisation. L'*action* représente les opérations d'accès (lecture, mise à jour, suppression et ajout) que le sujet peut effectuer sur l'objet. Finalement, le *Signe* prend soit la valeur positive (+), ce que signifie que la permission est accordée, ou soit la valeur négative (-) pour une interdiction. Dans ce qui suit, nous ne nous intéressons pas à la façon de gérer les sujets, et nous traitons uniquement l'opération de lecture puisque nous nous intéressons, dans notre travail, à la confidentialité des données. Par conséquent, la règle d'autorisation sur les nœuds prend une forme simplifiée. Cela nous permet de nous focaliser sur les aspects fondamentaux de notre contribution qui consiste à caractériser les objets à protéger. Ainsi, la règle d'autorisation sur les nœuds *AN* est définie comme suit : <*Sujet*, *Objets*, *Signe*>.

- Sujet : est une entité abstraite,
- Objet $\subseteq N_{Source}$,
- Signe $\in \{+, -\}$

Objet correspond aux attributs et aux éléments d'un document source identifiés par une expression XPath. La puissance d'expression d'un modèle de contrôle d'accès, et donc la granularité de partage, est liée directement au sous-ensemble de XPath supporté. Nous considérons, par la suite, un sous-ensemble significatif de XPath désigné par $XP^{\{[],*,//\}}$ [110]. Ce sous-ensemble, largement utilisé en pratique, englobe l'utilisation de tests sur les nœuds, l'axe parent-enfant (/), l'axe de descendance (//), de jokers (*) et de prédicats ([]).

La combinaison des règles d'autorisation positives et des règles d'autorisation négatives représente un bon moyen pour gérer les exceptions [93]. Afin de respecter le principe de *moindre privilège* nous adoptons la politique fermée. Nous rappelons que la politique fermée signifie qu'une règle d'autorisation négative s'applique, par défaut, à tout le document. En d'autres termes, l'accès à l'objet qui n'est pas explicitement autorisé est interdit. Nous supposons que la propagation des règles d'autorisation positives et des règles d'autorisation négatives est implicite, dans notre modèle, ce qui signifie qu'une règle se propage d'un noeud à tous ses descendants dans la hiérarchie XML. Ce mode de propagation correspond à l'option *cascade* présente dans les modèles les plus connus [19,42,75]. Étant donnée cette politique de propagation et le fait que plusieurs règles (positives et négatives) peuvent être définies pour un même utilisateur sur un même document, l'utilisation des politiques de résolution de conflit est indispensable. Ces politiques sont "*l'objet le plus spécifique est plus prioritaire*" et "l*'interdiction est plus prioritaire*", c'est-à-dire si deux règles s'appliquent sur le même nœud, la règle définie le plus bas dans l'arbre s'applique prioritairement, si le conflit persiste, la règle négative l'emporte.

4.2 Les règles d'autorisation sur les associations

Afin de répondre aux deux problèmes cités précédemment, notre approche défend l'intégration des associations d'ancêtres et de fraternité entre les nœuds comme des éléments de première classe dans les modèles de contrôle d'accès. Deux mécanismes (i.e, ***le clonage et le shuffling***) sont introduits pour traduire exactement ces associations en une vue autorisée. Un formalisme à base de règles a été défini pour exprimer ces classes d'autorisation. Il permet leur intégration d'une façon très simple dans les modèles de contrôle d'accès existants. Nous allons illustrer ces

trois aspects dans la suite de cette section. Nous ne présentons pas dans ce manuscrit la caractérisation formelle de notre modèle, nous en donnons une intuition (pour plus de détails lire l'article [72]).

4.2.1 Les mécanismes de base

Prendre en compte le consentement de l'utilisateur dans les modèles de contrôle d'accès impose de cloner des nœuds et des chemins du document `Source` dans la `Vue` retournée. Fondamentalement, le clonage de nœud `Source` n_1 (voir la Figure 16) est obligatoire à chaque fois que deux de ses descendants autorisés n_3 et n_4 ont deux règles d'autorisation conflictuelles correspondant par exemple à deux consentements différents. Cela traduit le fait que deux descendants (n_3 et n_4) ont deux visions différentes par rapport à leur ancêtre (n_1). Par exemple, le nœud n_1 (e.g, un élément service immunologie ou psychothérapie de notre exemple) doit se dépersonnaliser pour le noeud n_3 (e.g, un *patient VIP)*. Par contre, ce nœud n_1 ne se dépersonnalise pas pour le noeud n_4 (e.g, un patient qui a donné son consentement pour révéler le nom du service où il est traité).

Cela représente exactement le comportement de notre première règle d'autorisation R_1 qui dit "*cacher au groupe du répertoire*[13] *le nom du service où les patients sont traités pour ceux qui n'ont pas donné leur consentement*". Puisqu'un document XML est un arbre, chaque noeud participant dans le sous-chemin commun Path(*n1*,Parent(*n3*)) ∩ Path(*n1*,Parent (*n4*)) va être, à son tour, cloné.

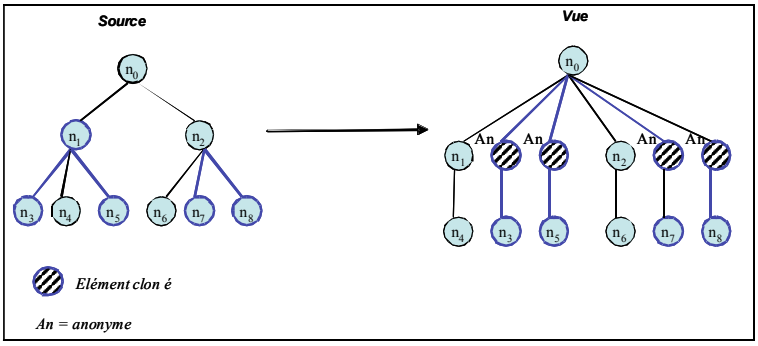


Figure 16 : Mécanisme de clonage

Le *clonage* est un mécanisme qui permet de dupliquer les chemins et les nœuds du document `Source` dans la `Vue`. Il est à noter que les nœuds feuilles (les éléments et les attributs terminaux) d'un document source ne font jamais l'objet d'un clonage. Le clonage est une condition préalable et indispensable pour cacher les associations ancêtre-descendant ainsi que les associations de fraternité entre les nœuds. Par contre, l'ordre des nœuds clonés, dans la `Vue`, doit être géré avec prudence pour éviter des problèmes d'inférence de base. Par exemple, d'après la Figure 16, le noeud n_1 s'est dépersonnalisé, d'une façon indépendante, pour les noeuds n_3 et n_5, le noeud n_2 s'est dépersonnalisé, également, pour les nœuds n_7 et n_8. Mais, le fait que les nœuds clonés prennent, à chaque fois, une position très proche par rapport à leur nœud original, cela conduit à avoir une structure régulière du document XML. Par conséquent, un utilisateur donné peut, inévitablement, deviner la structure initiale du document.

[13] Les utilisateurs responsables de la gestion d'annuaire de hôpital.

Pour illustrer ce problème, prenons la règle d'autorisation R_3 de notre politique de contrôle d'accès. Supposons que la `vue` est ordonnée de telle sorte que les instances des deux groupes (`ActesMed`, `Analyses`) et (`Nom, Adresse`) garde le même ordre relatif défini initialement dans le document `Source`. Dans ce cas, leur association de fraternité initiale, qui devrait être cachée par le clonage, est révélée par l'ordre des éléments (i.e., le $i^{ième}$ instance de (`ActesMed`, `Analyses`) correspond à la $i^{ième}$ instance de (`Nom, Adresse`)). Un problème similaire existe, également, avec la règle R_1. Donc, l'opération de clonage n'a aucun sens sans mélanger les nœuds clonés entre eux pour éviter l'inférence basée sur l'ordre. Nous appelons cette opération de mélange le "*shuffling des nœuds*". Cela est illustré dans la Figure 17.

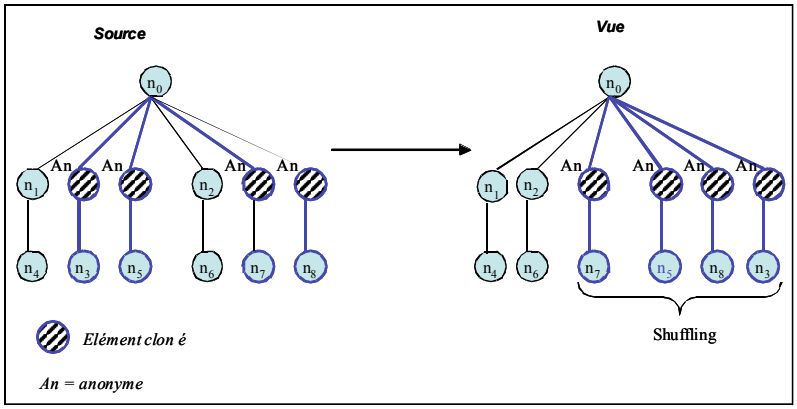


Figure 17 : Mécanisme de shuffling

Le shuffling des nœuds est un processus récursif qui s'applique à chaque nœud de la `Vue` contenant un clone comme un nœud fils. Tous les clones, fils d'un nœud donné, sont mélangés ensemble pour éviter l'inférence. Pour un nœud donné, les fils clonés sont groupés après les fils originaux (par convention) et ensuite ils sont mélangés. L'ordre relatif des fils originaux doit être préservé dans la `Vue,` puisque l'ordre des nœuds dans XML est important.

4.2.2 Les règles d'autorisation sur les associations

Une règle d'autorisation sur les associations *AA* est définie sous la forme d'un tuple <*Sujet, objet*>, où l'*objet*, à son tour, est défini par un quadruple : <*Anc, Desc, Visibilité, fraternité*> :

- *Anc* et *Desc* caractérisent l'association à protéger entre un ou ensemble de descendant(s) et un de leur ancêtre. *Anc* et *Desc* constituent le dénominateur commun de toutes les autorisations sur les associations. Ils sont identifiés par des expressions XPath.
- *Visibilité* caractérise la visibilité du chemin *u* reliant chaque nœud descendant à son ancêtre. Pour chaque nœud *n* participant dans *u*, *Visibilité* dit si le nœud est préservé ou non dans le chemin cloné. Dans le cas où le nœud est préservé, la *Visibilité* dit également si le label du nœud *n* sera préservé ou non.
- *Fraternité* : implicitement, cacher l'association avec l'ancêtre revient à cacher, également, l'association entre le nœud descendant et ses frères. Pour permettre une décorrélation sélective entre les frères, le paramètre de *fraternité* caractérise la liste

des frères que le descendant doit garder au moment de sa dissociation.

La définition de la règle d'autorisation sur les associations *AA* soulève deux remarques importantes. La première, concerne la concision et la maniabilité, *AA* capture, d'une façon très simple, toutes les formes d'autorisation sur les associations. La deuxième remarque, contrairement à AN, AA ne comporte pas le paramètre *signe*. La raison pour cela est qu'AA caractérise uniquement les règles négatives qui visent à chaque fois à dissocier le nœud en question. En effet, les règles d'autorisation sur les associations sont elles définies selon une politique ouverte qui retourne en résultat la vue finale autorisée `Vue`. Par conséquent, si aucune règle d'autorisation sur les associations est définie, la sémantique de notre modèle est conforme avec les modèles de contrôle d'accès existants dans la littérature. Par conséquent, l'intégration des autorisations sur les associations dans ces modèles existants peut être effectuée facilement.

La sémantique globale de notre modèle est comme suit : les règles d'autorisation sur les nœuds (AN) sont définies selon une politique fermée et retournent en résultat une vue autorisée `Vue`$\subseteq$`Source`. D'une façon générale, les arcs ayant des noeuds extrémités supprimés par une règle AN sont à leurs tours supprimés de la `Vue`.

4.2.3 Les ancêtres et les descendants

Pour qu'une règle d'autorisation sur les associations soit cohérente, elle doit satisfaire la condition *Desc* $\subseteq$ *Anc*, où $\subseteq$ désigne la relation d'inclusion entre les expressions XPath. Malheureusement, le problème d'inclusion a été montré co-NP complet pour la classe d'expression $XP^{\{[],*,//\}}$ [110]. Afin d'éviter les vérifications d'incohérence, *Desc* est défini comme une expression relative (chemin relatif) par rapport à Anc. Donc, Anc détermine un ensemble d'origines du chemin et Anc/Desc détermine un ensemble d'extrémités de chemin.

4.2.4 La visibilité du chemin

Pour chaque nœud *n* participant au chemin sélectionné par *Anc* et *Desc*, *Visibilité* doit spécifier si *n* participe ou non au chemin cloné dans `Vue.` Si cela est vrai, alors *Visibilité* doit spécifier, également, si le label de *n* est hérité ou non par *ñ*, clone de *n*. Nous définissons quatre possibilités différentes pour le paramètre *Visibilité* : i) sélectionner les nœuds à supprimer du chemin cloné ou dépersonnaliser leur label original ; ii) préserver tous les nœuds du chemin cloné avec leur label original. Cette option représente l'option par défaut. iii) préserver tous les nœuds de chemin cloné tout en anonymisant leur label original ; iv) enfin la suppression de tous les nœuds du chemin cloné est envisagée. Le Tableau 1 récapitule les choix possibles pour ce paramètre et leurs sémantiques. La Figure 18 illustre graphiquement leurs effets sur un chemin caractérisé par un ancêtre /A et des descendants //C. La première ligne du tableau donne une syntaxe étendue pour ce paramètre tandis que les autres lignes proposent une abréviation d'expression de la politique.

Visibilité du chemin	Sémantique de visibilité du chemin
[$label_1$?,…, $label_n$?]	Donne la liste des nœuds à supprimer (?= †) ou à dépersonnaliser (?= Φ).
[ ]	Tous les nœuds sont préservés sur le chemin (i.e, tous les nœuds clonés) et leur label original est hérité. Cette option est l'option par défaut.
[Φ]	Tous les nœuds sont préservés sur le chemin mais ils sont dépersonnalisés (i.e, le label de leur clone est égal à " anonyme").
[†]	Tous les nœuds sont supprimés du chemin.

Tableau 1 : La sémantique de Visibilité du chemin

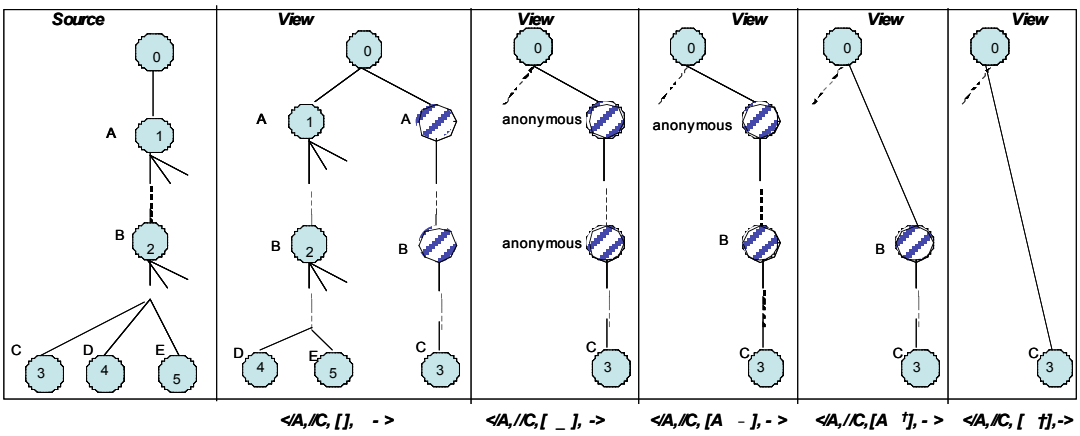


Figure 18 : Exemple de Visibilité du chemin

4.2.5 Les associations de fraternité

Le paramètre *fraternité* est utilisé pour exprimer la décorrélation sélective des frères. Ce paramètre donne la liste des frères qu'un descendant doit préserver lors de sa dissociation. Le nœud descendant peut préserver les associations de fraternité avec certains nœuds ou avec des nœuds ayant le même label. Il peut également se déconnecter de tous ses frères ou préserver toutes ses associations de fraternité. Le Tableau 2 récapitule les choix possibles pour ce paramètre avec leurs sémantiques et la Figure 19 illustre graphiquement leurs effets. Encore une fois, la première ligne de tableau donne une syntaxe étendue pour ce paramètre tandis que les autres lignes proposent une abréviation d'expression de la politique.

Fraternité	Sémantique du paramètre de *fraternité*
[$label_1$,… $label_n$]	Tous les nœuds ayant un label appartenant à cette liste doivent préserver leurs associations avec le descendant en question.
[⊥]	Le nœud descendant est déconnecté de tous ses frères. C'est l'option par défaut.
[ψ]	Le nœud descendant préserve ses associations de fraternité avec tous les frères ciblés par la même règle d'autorisation.
[≡]	Le nœud descendant préserve ses associations de fraternité avec tous ses frères.

Tableau 2 : La sémantique du paramètre de fraternité

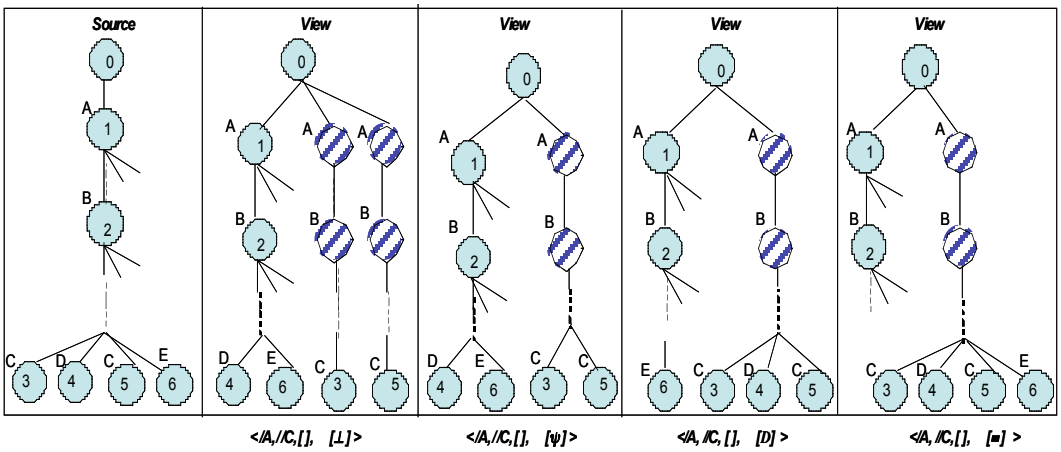


Figure 19 : Exemples de décorrélation sélective

4.2.6 Résolution des conflits

Dans le modèle de contrôle d'accès étendu aux associations, trois classes de conflits doivent être gérées [72]: (i) les conflits entre les règles d'autorisation sur les nœuds ; (ii) les conflits entre les règles d'autorisation sur les nœuds et les règles d'autorisation sur les associations ;(iii) les conflits entre les règles d'autorisation sur les associations. La résolution des conflits entre les règles d'autorisation sur les nœuds est héritée directement des modèles de contrôle d'accès existants que nous avons déjà expliquée dans la section 3.2. Les conflits entre les règles AN et AA sont évités puisque les règles sur les associations sont définies sur la vue produite après l'évaluation des règles d'autorisation sur les nœuds, selon une politique ouverte par défaut. En d'autres termes, les règles AN sont, toujours, plus prioritaires que les règles AA. Les décisions prises pour résoudre les conflits respectent toujours le principe du moindre privilège. La politique de résolution de conflits n'est pas détaillée ici, pour plus d'informations le lecteur peut se référer à la thèse de Saida Medjdoub [108].

4.3 Application

Nous montrons dans cette section comment notre modèle est utilisé pour exprimer la politique de contrôle d'accès présentée en section 2.2. Chacune d'elles comprend un mélange entre règles d'autorisation sur les nœuds et les associations.

Pour exprimer la règle R1, trois règles AN sont nécessaires. AN1 et AN2 et AN3 capturent les informations strictement nécessaires pour le groupe responsable de la gestion d'annuaires afin d'accomplir leur tâche. Par conséquent, les éléments `ActesMed` et `Analyses` sont supprimés. La règle d'autorisation sur les associations AA1 dépersonnalise ([Φ]) l'ancêtre service médical (/* identifie tous les services médicaux) de chaque dossier pour les patients qui n'ont pas consenti à divulguer leur service médical et, par conséquent, elle dissocie le dossier médical de tous ses frères ([⊥]).

Règle d'autorisation R1 :

AN1: < Groupe-annuaire , /Hôpital, + >

AN2: < Groupe-annuaire, //ActesMed, - >

AN3: < Groupe-annuaire, //Analyses, - >

AA1: < Groupe-annuaire, /*, /Dossier[./Consentement/annuaire/Service= 'no visible'], [Φ], [⊥] >

L'expression de la règle R2 se fait également par un mélange de règles AN et AA. La règle AA2 exprime une réduction du chemin supprimant le parent des éléments `Acte ("Protocole")`. Les règles AN5, AN6 et AN7 suppriment respectivement les éléments `Analyses`, `diagnostics et symptômes.`

Règle d'autorisation R2

AN4 : <Pharmacien, //Hôpital, + >

AN5 : <Pharmacien, //Analyses, - >

AN6 : <Pharmacien, //diagnostics, ->

AN7 : <Pharmacien, //symptômes, ->

AA2 : <Pharmacien, //ActesMed/Protocole, /Acte, [†], [⊥] >

Enfin, pour la règle R3, deux règles sur les nœuds (AN9 et AN10) interdisent au laboratoire médical d'accéder aux noms, aux adresses des patients qui n'ont pas donné leur consentement pour des raisons de commercialisation. La règle AN11 supprime tous les nœuds NSS. Pour les patients donnant leur consentement, RA3 empêche l'inférence entre les informations d'identification (`nom`, `adresse`) et le reste du dossier médical.

Règle d'autorisation R3

AN8 : < lab Médical, //Hôpital, + >

AN9 :<lab Médical,//Dossier[./Consentement/Commercial/InfoPersonnel='no-visible']/nom ->

AN10 :<lab Médical,//Dossier[./Consentement/Commercial/InfoPersonnel='no-visible']/Adresse, ->

AN11 : :< lab Médical, // NSS, ->

RA3 : < lab Médical, //Dossier, /nom, [ ], [Adresse]>

5 Conclusion

Dans ce chapitre, nous avons caractérisé les carences des modèles existants. Nous avons tout d'abord montré qu'il n'existe pas une sémantique unique et non ambiguë des modèles de contrôle d'accès XML. En effet, pour une même politique de contrôle d'accès, plusieurs interprétations sont parfois possibles, certaines ne respectant pas les règles d'autorisation définies. D'autre part, les modèles existants ne permettent pas, dans un nombre significatif de situations, de traduire fidèlement les deux principes fondateurs (finalité et consentement) édictés dans les textes de loi pour la protection des données personnelles. À ce titre, notre première contribution est une étude confrontant les textes de loi avec les modèles de contrôle d'accès existant, étude durant laquelle nous avons identifié leurs carences [69].

Puis, nous avons proposé un nouveau modèle de contrôle d'accès pour documents XML [72] qui répond de façon crédible aux problèmes mentionnés ci-dessus. Ce modèle intègre les concepts de finalité et de consentement. Ce modèle repose sur des règles permettant de protéger un nœud vis-à-vis de ses ancêtres et de certains de ses frères.

Un des intérêts de l'approche est de rester compatible avec les modèles existants se focalisant sur la protection des nœuds. En effet, nous avons étendu les modèles existants afin d'intégrer la gestion des droits sur les associations entre les nœuds. Cette approche préserve les propriétés de concision et de validité indispensables à une bonne politique de contrôle d'accès[14]. La politique peut s'exprimer par un ensemble minimum possible de règles d'autorisation. En effet, notre modèle a un fort pouvoir d'expression tout en gardant un fort degré de concision. La vue autorisée traduit exactement la sémantique de la politique de contrôle d'accès. Les politiques de contrôle d'accès exprimables dans ce modèle sont dites sûres dans le sens où il existe un algorithme déterministe et une sémantique claire qui permet de calculer la vue autorisée à partir du document initial. Cette approche ascendante a été validée par un prototype écrit en Java. Il a été construit à partir d'un prototype existant dans le domaine public [44] qui a été étendu par la gestion des règles d'autorisation sur les associations. Nous avons réalisé des tests de performance qui montrent que le coût d'intégration des règles d'autorisation sur les associations est tout à fait comparable au coût d'intégration de règles d'autorisation sur les nœuds.

Par ailleurs, la plupart des travaux font aujourd'hui l'hypothèse que le document XML tient en mémoire pour évaluer la politique de contrôle d'accès. C'est l'hypothèse que nous avons retenue pour notre travail préliminaire de validation. Cependant, nous sommes conscients qu'il faut se tourner vers d'autres techniques d'évaluation pour pouvoir gérer des documents de grande taille. C'est pourquoi l'idée serait d'évaluer en parallèle et à la volée l'ensemble des requêtes XPath définies dans les règles d'autorisations sur les nœuds et les associations afin de rendre en flux la vue autorisée du document. Une étude préliminaire a été menée afin de voir comment notre modèle de contrôle d'accès étendu avec des règles d'autorisation sur les associations entre les nœuds peut faire l'objet d'un traitement en flux. Les challenges sont liés au problème de la restructuration parfois complexe liée à notre modèle (e.g. le clonage et le shuffling). Un premier prototype est aujourd'hui opérationnel, mais il reste encore beaucoup d'améliorations et d'extensions à apporter. La difficulté repose sur le fait de devoir gérer en même temps les règles sur les nœuds et les associations, de savoir quand délivrer l'information ou la mémoriser.

Pour conclure, il serait intéressant d'expérimenter notre modèle de contrôle d'accès dans un cadre réel et de valider si le modèle proposé répond bien à une attente applicative. Dans le cadre du projet SMIS, nous sommes en discussion avec le département des Yvelines, pour proposer avec l'aide de UNIMEDECINE et du laboratoire PRiSM une plate-forme expérimentale de dossier médical personnel dans un contexte de réseau de soins de portée départementale. Si ce projet aboutit, il pourra servir de cadre à une telle expérimentation.

[14] Notons que l'ensemble des règles n'est pas forcément minimal, il peut y avoir de la redondance et des inconsistances qui sont toutes deux gérées par le mécanisme de résolution de conflits.

Chapitre VII – Conclusion et perspectives de recherche

Dans ce manuscrit, nous avons présenté différentes contributions pour l'accès transparent et sécurisé à des données largement distribuées. Notre approche pluridisciplinaire a été riche d'expérience et d'enseignements, elle nous a permis de créer et de participer à de nombreux projets de recherche, permettant de définir le cadre de plusieurs thèses et menant à de nombreuses publications. Dans cette section, nous avons choisi de présenter nos conclusions et perspectives de recherche selon deux axes, celui de l'accès transparent d'une part, et celui de l'accès sécurisé d'autre part.

1 Accès transparent aux données

Les communautés bases de données et systèmes distribués définissent des logiciels de médiation pour l'accès transparent aux ressources largement distribuées. Dans ce manuscrit, nous avons décrit nos contributions dans ces deux communautés.

Nous nous sommes intéressés aux systèmes de médiation de sources de données relationnelles, objets ou XML. Nous avons défini et réalisé plusieurs architectures de système pour l'accès transparent à des bases de données objets et relationnelles (IRO-DB) d'une part, et à des bases de données XML (MEDIAGRID) d'autre part. Au-delà des modèles de données, nous avons contribué à résoudre différentes dimensions du problème comme les mécanismes d'intégration de données, le partage, la mise à jour, l'accès de façon déclarative et par navigation, la gestion hybride des données entre entrepôts (i.e, vues matérialisées) et techniques de réécriture de requêtes à travers les vues. Nous avons également étudié les problèmes liés à la forte dynamicité de l'internet et regardé les mécanismes d'évaluation et d'optimisation de requêtes multi sources avec indisponibilité de sources, trafic réseau inconstant et offrant des possibilités d'interaction avec l'utilisateur.

Nous nous sommes également intéressés aux annuaires. Ils permettent d'interroger et de localiser des ressources largement réparties à l'échelle planétaire et sont très bien adaptés au passage à l'échelle. Dans ce cadre, nous avons contribué à résoudre différentes dimensions du problème comme la gestion et la localisation de ressources aux propriétés dynamiques, la capacité limitée des langages d'interrogation existants. Nous avons proposé et réalisé un nouveau service d'annuaires aux fonctionnalités avancées permettant de gérer (i.e, créer, modifier et supprimer) et d'interroger des ressources de façon déclarative via le langage DQL et par navigation (cf. Annexe B). Notre annuaire gère des ressources aux propriétés statiques et dynamiques dont les caractéristiques ne sont pas sans rappeler la gestion hybride de données (i.e, matérialisées ou non). Nous avons gardé la flexibilité et la compatibilité avec les annuaires

LDAP et proposé des mécanismes d'évaluation de requêtes offrant des résultats et des évaluations partiels de requêtes car les nombreux sites gérant les ressources dynamiques peuvent êtres inaccessibles car saturés ou en panne, voire momentanément déconnectés.

À la lueur de nos travaux, nous avons constaté que les architectures fonctionnelles sont très similaires. Il y a un besoin d'intégrer des ressources hétérogènes, de les décrire, de les nommer et de les mettre à jour. Il y a aussi un besoin de gérer des documents (ou objets) virtuels qui sont le résultat de l'agrégation d'un certain nombre d'objets (ou documents), de les interroger de façon efficace et simple, de façon déclarative ou par navigation et d'autoriser des résultats partiels. Il est important de gérer la cohérence des données vis-à-vis des mises à jour et d'avoir de bonnes performances, ce qui nécessite parfois de matérialiser plus ou moins d'informations au niveau du logiciel de médiation.

Un très grand nombre de travaux ont été faits autour des systèmes de médiation de données relationnelles et objets, ainsi que dans le domaine des bases de données distribuées, ils peuvent être réutilisés et adaptés dans un contexte XML. Actuellement, les solutions existantes pour l'accès transparent à des sources de données XML se focalisent beaucoup sur l'aspect valeur de XML et sur l'interrogation du Web (e.g, les systèmes Pair-à-Pair, le Web sémantique). Les solutions s'appuient sur le langage XQuery pour la définition de vues au niveau du système de médiation et l'utilisation de techniques classiques de réécriture et de décomposition de requêtes à travers les vues. Ces solutions sont très intéressantes, mais elles ne mettent pas l'accent sur le développement d'applications sur le Web ou la grille et la protection de la confidentialité des données. En effet, plusieurs problèmes ne sont pas résolus.

Le premier problème concerne l'intégration forte et cohérente de données. En effet si l'on souhaite réaliser une intégration forte de documents XML il est important de pouvoir agréger des données, de les annoter, d'en générer de nouvelles qu'il faut pouvoir référencer et utiliser. On ne s'intéresse donc pas uniquement à de simples consultations. On désire partager des ressources et par conséquent on aimerait aussi pouvoir les modifier de façon transparente. On souhaite également une gestion cohérente des données. Actuellement la gestion des IDREFs où références locales à un document n'est pas gérée au niveau global (i.e, du système de médiation). Lors de l'interrogation, les liens sémantiques entre les données locales sont perdus.

Le deuxième problème concerne l'interrogation et l'accès transparents à ces données. Actuellement, seul un accès déclaratif de type XQuery est proposé. Cependant la plupart des programmeurs qui développent des applications réparties accèdent le plus souvent à leurs données par navigation, soit par l'intermédiaire d'API DOM (Document Object Model) [57], soit par des navigateurs Web. D'une part, afin de naviguer sur le document virtuel via une API DOM il est nécessaire de concrétiser complètement la vue. Bien qu'il existe des interfaces spécifiques dans les bases de données XML natives (PDOM [7]), ces solutions ne sont pas adaptées à la gestion de documents virtuels. Par ailleurs, les solutions actuelles privilégient les accès par valeur et ne permettent pas aux utilisateurs de connaître la provenance, la localisation de leurs données et par conséquent une fois leur requête XQuery évaluée, ils ne peuvent naviguer sur le résultat, ni consulter d'autres données en liaison directe avec les données retournées. Actuellement, même dans les systèmes Pair à Pair, on mentionne le besoin de maintenir la provenance des données interrogées [121].

Le troisième problème concerne l'utilisation de XQuery pour offrir un accès transparent, efficace et sécurisé à des données largement distribuées. Il y a deux dimensions à ce problème. La première dimension est que XQuery est un langage puissant, mais aussi très complexe à optimiser. Dans le cadre du projet MEDIAGRID, nous avons constaté, même pour une intégration de documents XML relativement simples, que les requêtes de médiation générées ressemblent plus à des programmes itératifs et laissent présager d'un processus de réécriture et d'optimisation complexe. Ce processus doit se répéter pour chaque requête utilisateur. La seconde dimension du problème concerne l'accès sécurisé aux vues définies avec XQuery. En effet, les modèles de contrôle d'accès XML existants ne s'appuient pas sur des techniques classiques BD, la définition de vues autorisées et la réécriture de requêtes. Bien qu'à première vue, un langage tel que XQuery puisse être considéré comme un moyen approprié pour définir un tel processus[15], il a été montré [23] que même pour des restructurations simples du document XML, tel qu'une suppression d'un sous arbre ou d'un élément, cela nécessite des expressions XQuery relativement complexes. Là également, les expressions ressemblent plutôt à des programmes itératifs qu'à des expressions déclaratives. Par conséquent, les politiques générées via ce modèle ne sont pas très lisibles et sont donc difficiles à gérer. C'est pourquoi les modèles de contrôle d'accès existants proposent d'exprimer la politique de contrôle d'accès par un ensemble de règles d'autorisation sur le document XML à partager. Les techniques de réécriture de requêtes ne fonctionnent plus et il reste encore beaucoup de recherche pour définir et implémenter des techniques performantes pour la vérification des droits d'accès lors de la consultation d'un document que cela soit de façon déclarative ou par navigation.

Le quatrième problème concerne les mécanismes internes nécessaires à la gestion de données dans le système de médiation. Actuellement, il existe une dichotomie entre l'approche de type entrepôt et l'approche totalement virtuelle adoptée dans les systèmes de médiation de données XML. Cependant, nous l'avons vu au travers des approches développées dans IRO-DB et les annuaires, il est souvent nécessaire d'adopter une approche hybride, c'est-à-dire de conserver dans le cache les données pour une période relativement longue. Les performances d'accès aux données peuvent ainsi être améliorées. Des politiques différentes de rafraîchissement de données peuvent être définies en fonction des besoins des utilisateurs et/ou de la sémantique des données gérées. Actuellement chaque requête utilisateur est réécrite, l'optimisation est donc locale à chaque requête. Il serait intéressant d'optimiser globalement la charge du système si des milliers d'utilisateurs interrogent la même vue et d'étudier dans quelle mesure il est possible de partager les accès, particulièrement en consultation. Certaines données sont peut-être accédées bien plus souvent que d'autres. Bien sûr, il est toujours possible de répliquer le système d'intégration pour chaque utilisateur, mais cela maintient une charge constante au niveau des sites distants qui peuvent à leur tour s'écrouler.

Comme nous venons de le voir, il reste encore beaucoup de problèmes à résoudre pour offrir un système de médiation de sources de données XML aux fonctionnalités avancées et aux performances acceptables. Récemment, une approche combinant document XML et appels de services Web a été proposée [2,2] dans la littérature. L'idée consiste à définir un document XML actif (AXML) dans lequel les données sont à la fois décrites en intension et en extension,

[15] XQuery est un langage Turing complet.

autrement dit, le document XML devient un document où les données et les programmes se côtoient. Cette approche n'est pas sans rappeler les appels Javascript introduit dans les pages HTML pour l'écriture de pages Web dynamiques, et les langages fonctionnels pour lesquels un programme égal une donnée et dans lesquels on trouve de la réécriture de code (i.e, de données), de l'évaluation partielle, etc.

Dans cette approche, le document lui-même est vu comme un moyen d'exprimer l'intégration de données et la composition de services. Cette approche a été utilisée pour développer des systèmes P2P (KadoP[4]) et GRID (DBGlobe [47]). Les appels de services Web étant eux-mêmes décrits en XML, on peut les évaluer ou non, c'est-à-dire remplacer l'appel de programmes par le document AXML résultat de l'évaluation. Les auteurs de l'approche préconisent une approche hybride entre entrepôt et système de médiation afin d'optimiser l'évaluation des requêtes. Les requêtes XQuery sont évaluées de façon transparente sur les données intensionnelles et extensionnelles. Le système AXML réalise un couplage faible avec la base de données XML native, Xylème [157]. Les données intensionnelles sont gérées au-dessus du SGBD-hôte.

Notre expérience et expertise acquises dans le monde orienté objets des bases de données (ODMG) aux plates-formes à objets distribués (CORBA) nous suggèrent d'aller encore plus loin et d'utiliser toute la puissance du modèle XML qui intègre la notion de valeur et de référence et qui permet de représenter à la fois des données et des programmes. XML est un modèle hypermédia intégrant des références (XPointer [154], XLink[153]) et des appels de programmes (i.e, services Web). L'approche proposée dans [2,2] constitue une première étape intéressante pour la composition de services. Cependant elle ne permet qu'un accès déclaratif via XQuery. Le partage de données passe obligatoirement par l'écriture et l'appel à un service Web, la gestion n'est pas transparente, ne permet pas la mise à jour de ressources et n'offre pas d'accès par navigation. Par ailleurs, l'approche couplage faible a certaines limitations et ne nous semble donc pas la mieux adaptée si l'on souhaite obtenir de bonnes performances.

Une première perspective de recherche est l'intégration et la gestion transparentes des références (XPointer, XLink) au sein d'un document XML et la caractérisation de celles utiles pour le partage de données. Les références XLink sont décrites en XML et peuvent donc représenter des données intensionnelles. Pour reprendre une terminologie objet, nous pourrions dire qu'il serait intéressant de considérer des documents XML répartis en analogie aux objets répartis gérés dans les plates-formes CORBA ou les SGBDOO. Nos travaux antérieurs constituent un bon point de départ pour cette réflexion. Nous pensons que cette réflexion est cruciale pour fournir des solutions communes pour la mise en place d'un système de médiation complet aux fonctionnalités avancées, le développement d'applications réparties sur le Web et la construction d'annuaires aux fonctionnalités avancées.

En effet, cette nouvelle approche offre plusieurs avantages très importants. Au niveau du modèle de données, elle permet l'intégration de données et le partage de documents. Par conséquent, il n'est pas nécessaire d'écrire un programme pour partager et intégrer un fragment de données. Elle permet une gestion plus cohérente des données car il est possible de remplacer l'IDREF par sa référence. Ainsi, lors de l'interrogation, les liens sémantiques entre les données locales ne sont plus perdus. Elle permet le maintien de la provenance (i .e, la localisation) des

données interrogées sur le Web. Au niveau de l'interrogation, cette approche permet une plus grande flexibilité car l'utilisateur ou le programmeur peut choisir à la demande de déréférencer un lien, en naviguant à travers un navigateur Web ou une interface DOM.

Une seconde perspective de recherche concerne l'étude des mécanismes d'exécution associés. Notre expérience acquise dans le système IRO-DB et les annuaires étendus nous suggère plutôt un couplage fort pour un accès efficace et sécurisé. En effet, un couplage fort permet des optimisations intéressantes dans un contexte XML au niveau de la machine d'exécution. Actuellement, de nombreux efforts sont menés pour une gestion efficace d'un très grand nombre de gros documents XML dans un environnement centralisé. Nous pensons que des efforts similaires doivent être réalisés pour une gestion d'un grand nombre de documents XML répartis. Il serait intéressant d'évoluer vers le concept de base de données XML répartie et d'intégrer de façon similaire et transparente les références aux documents répartis et les appels de programmes (i.e, services Web).

En particulier, les techniques d'évaluation de requêtes doivent s'adapter à la nature virtuelle (i.e, donnée intensionnelle) et répartie des données et offrir des résultats et des évaluations partiels de requêtes. Actuellement, les gestionnaires de requêtes (XPath, XQuery, XUpdate) ne gèrent que des données locales et centralisées et il faut donc étendre les modèles d'exécution existants, adapter et étendre les techniques d'optimisation, autoriser des résultats et des évaluations partiels de requêtes. L'utilisation des références à la place des valeurs complexes permet d'éviter des transferts de données inutiles et permet la gestion des résultats partiels. Par exemple si des sources de données ne sont pas accessibles, il est possible de remplacer la partie manquante par un lien. Par la suite, l'utilisateur pourra réinterroger le document et récupérer la partie manquante[16].

Par ailleurs, il est nécessaire de proposer de nouvelles API DOM et de modifier les navigateurs existants afin qu'ils s'adaptent à la sémantique particulière des données intentionnelles, que cela soit des références ou des programmes. La gestion de cache doit être flexible et permettre une gestion hybride et performante des données intensionnelles, matérialisées ou non. Les données pourront être conservées dans le cache pour une période relativement longue, les performances d'accès aux données seront ainsi être améliorées et des politiques différentes de rafraîchissement de données pourront être définies en fonction des besoins des utilisateurs et/ou de la sémantique des données gérées. Tous ces avantages ne sont pas sans rappeler les avantages offerts par les SGBDO.

Sur le plan de l'accès sécurisé, il reste encore beaucoup à faire pour intégrer et évaluer de façon efficace une politique de contrôle d'accès au sein d'un système de médiation. Nous pensons que la notion de documents XML répartis est plus appropriée à la mise en œuvre des modèles de contrôle d'accès existants, plutôt qu'une approche XQuery. Par exemple dans les approches à base de règles, il est possible de retrouver facilement les règles de contrôle d'accès associés à un fragment de données XML, la vérification des droits d'accès peut s'effectuer localement même au moment d'une navigation.

[16] C'est déjà le cas dans les navigateurs Web et dans les plates-formes à objets répartis, l'utilisateur est informé du fait que le site qui gère sa donnée peut être inaccessible, déconnecté ou en panne et doit lui-même gérer les erreurs.

2 Accès sécurisé aux données

Le principal frein au développement de nouvelles applications sur l'Internet, et donc un frein au partage et à l'interrogation de ressources largement distribuées, est la gestion actuelle des données confidentielles. La préservation de la confidentialité est une tâche gigantesque et est un thème de recherche complémentaire tant aux systèmes de médiation de données qu'aux plates-formes à objets répartis et à leurs annuaires. Dans les deux environnements, il est en effet vital d'authentifier les utilisateurs et de vérifier leurs droits d'accès. Tous nos travaux et perspectives de recherche sont donc utiles aux deux communautés.

Dans ce manuscrit, nous avons décrit et présenté nos contributions concernant la sémantique des modèles de contrôle d'accès pour XML. Dans ce cadre nous avons étudié les limites et carences des modèles existants et proposé un nouveau modèle de contrôle d'accès qui protège non seulement les nœuds mais également les liens entre les nœuds. Ces travaux sont des travaux préliminaires et il reste beaucoup de perspectives de recherche à explorer.

Les modèles actuels de contrôle d'accès définissent des règles d'autorisation en sélectionnant des fragments d'un document par des requêtes XPath. Ces règles sont très dépendantes de la structuration des documents à protéger et n'intègrent pas de sémantique. Les règles sont donc difficiles à définir et ne s'appliquent pas si le concept interrogé ne correspond pas exactement au concept défini dans la règle. Par exemple, la règle permettant de protéger les malades atteints du sida est difficile à exprimer car le concept sida peut correspondre à la présence de plusieurs types d'informations de nature différente (i.e, la prise d'un médicament spécifique, un résultat sanguin, etc). Il est donc impératif de fournir des modèles et outils permettant de définir avec plus de précision « qui est autorisé à faire quelles actions sur quelles données » tout en offrant une bonne flexibilité en cas de mises à jour des règles ou des sources de données interrogées. Nous avons déjà réalisé un premier pas dans cette direction avec notre modèle qui protège des associations entre deux éléments XML. Actuellement, nous travaillons à une généralisation du problème avec Frédéric Cuppens de l'ENST Bretagne. L'idée consiste à définir des politiques de contrôles d'accès à un niveau conceptuel (UML) et de ne pas se préoccuper du format de stockage ou d'échanges des données (i.e, relationnel ou XML).

Aujourd'hui il est important d'adresser la résolution de problèmes plus larges qui touchent à la fois à la vie privée et à la sécurité sur le Web. En effet, les modèles actuels protégeant la confidentialité des données sont limités et il faut évoluer vers de nouveaux modèles protégeant les données personnelles, c'est-à-dire la vie privée et l'intimité des personnes. Nos travaux préliminaires ont montré les carences des modèles existants vis-à-vis des législations protégeant les données à caractère personnel, mais il faut aller encore plus loin. La proposition P3P (Platform for Privacy Preferences)[143] du W3C va dans ce sens et doit fournir un moyen standard, simple et automatique aux utilisateurs, de façon à ce qu'ils prennent davantage contrôle de l'utilisation de leurs données personnelles lorsqu'ils visitent des sites Web.

Il reste encore un long chemin à parcourir avant de satisfaire tous les principes imposés par les législations, notamment celles relatives à la protection des données à caractère personnel. Jusqu'à présent, nous nous sommes intéressés à deux principes fondateurs des législations : le

principe de finalité et de consentement. C'est un point de départ. Aujourd'hui de nouveaux modèles apparaissent et cherchent à répondre à une question fondamentale dans tout traitement des données personnelles: « quel est l'objectif d'un traitement ». Ces modèles vont beaucoup plus loin que les modèles protégeant la confidentialité des données car ils visent à en protéger l'intimité. Par exemple, dans le domaine médical, les législations déclarent que "le médecin peut utiliser les données médicales d'un patient pour un objectif de traitement et de diagnostic". Par ailleurs, certains traitements autorisés par les législations sont soumis à des obligations, comme anonymiser les données avant de les divulguer à un tiers pour des analyses statistiques.

La traçabilité des accès (historique) représente également un principe fondateur de la protection des données personnelles qui consiste à garder trace de l'heure d'accès, des informations divulguées, du nom et de l'adresse du tiers, de l'objectif, etc. Par exemple, dans le domaine médical, l'accès au dossier médical peut être forcé en cas d'urgence pour sauver la vie de patients. Par contre, les circonstances des accès doivent être sauvegardées pour justifier par la suite des raisons de ces accès. Il reste beaucoup de travail pour définir des modèles protégeant l'intimité des données, que cela soit pour XML ou pour les données relationnelles. Avec l'informatisation du dossier médical personnel, la définition d'un tel modèle devient pourtant une priorité. Actuellement, nous nous intéressons au problème de traçabilité (i.e, le droit de chaque utilisateur de consulter la liste des accès à ses données personnelles), ces travaux viennent de démarrer et servent de cadre à la thèse de Medhi Benzine.

Pour aller encore plus loin dans cette direction, le concept de SGBD Hippocratique, c'est-à-dire de SGBD donnant l'assurance du respect du serment d'Hippocrate des médecins pour la gestion de données personnelles, a été introduit dans [9]. Un tel SGBD se doit de respecter un ensemble de principes fondateurs parmi lesquels : préciser l'objectif d'utilisation de chaque donnée collectée sur un utilisateur et recueillir l'assentiment de l'utilisateur sur cet objectif, ne stocker que l'information strictement nécessaire à l'atteinte de cet objectif et uniquement pendant le laps de temps strictement nécessaire, ne pas divulguer cette information à des tiers sans autorisation préalable de l'utilisateur, donner à l'utilisateur la possibilité de consulter les informations qui le concernent et enfin offrir des outils permettant à un tiers de contrôler que ces principes sont bien respectés. Ces principes sont tous très séduisants et gagneraient à être intégrés dans les dispositifs informatiques. Les contributions possibles dans ce cadre sont d'ordre sémantique (e.g, comment définir l'objectif, l'assentiment de l'utilisateur, etc.), algorithmique (e.g, comment vérifier efficacement les principes hippocratiques, quelles données doivent être conservées, comment, etc.) ou au niveau des protocoles (e.g, qui garantit quoi et comment, etc.).

La dernière perspective touche aux mécanismes mis en œuvre pour le partage de données. Traditionnellement le partage des fichiers de données se fait grâce à un serveur de confiance à qui nous déléguons la gestion des droits d'accès. Cependant, ces serveurs sont de plus en plus souvent attaqués par des pirates ou, d'une manière plus grave, par les administrateurs de bases de données [40]. Il est donc difficile de faire confiance à un système de gestion de données dans ces conditions. Si nous n'avons pas confiance dans le serveur de données, nous ne pouvons pas stocker en clair les données, et encore moins lui faire confiance pour qu'il réalise la gestion des droits d'accès. La seule façon de protéger ces données contre toute attaque malveillante "interne

et externe" est de chiffrer les données avant de les déposer sur le serveur de non-confiance et d'effectuer la gestion des droits d'accès côté client. La question qui se pose alors est « *comment traduire une politique de contrôle d'accès par des règles de chiffrement et de distribution de clés* » de telle sorte que chaque utilisateur déchiffre uniquement les parties qui lui sont autorisées par la base de règles d'autorisation.

Dans la littérature plusieurs techniques ont été proposées [16,15,111,128]. Notre analyse approfondie de ces modèles pour XML a permis de mettre en évidence certaines carences et contre-exemples qui montrent que ces modèles peuvent être mis en échec dans certains cas. Afin de résoudre ces problèmes, nous avons proposé des améliorations de l'algorithme de chiffrement [108]. Cependant toutes ces approches permettent d'exprimer une situation de partage à l'instant *t* et sont donc très statiques. Une première étude a montré que la traduction d'une base de règles en chiffrement n'est pas compatible avec un degré élevé de mises à jour. Comme le document chiffré est généré à partir de la base de règles, tout changement conduit à déstabiliser, bien évidemment, le document chiffré et cela coûte très cher. Il est donc important de proposer des techniques adaptées aux environnements dynamiques.

Pour résoudre ce problème plusieurs approches sont possibles. La première proposée par [22] allie techniques cryptographiques et composants sécurisés. L'idée consiste à chiffrer le document par un ensemble de clés et à réaliser le contrôle d'accès au niveau du client via un composant sécurisé de type carte à puce. Nous pensons qu'une autre approche possible consisterait à définir une méthodologie permettant de protéger un document XML indépendamment des règles d'autorisation des utilisateurs. En d'autres termes, l'identification des groupes d'éléments à chiffrer avec une même clé doit se faire indépendamment de la base de règles. Nous avons commencé à réfléchir à ce problème. L'idée consiste à découper le document en un ensemble *d'unités de protection***.** Une unité de protection peut contenir un seul élément, un attribut ou un sous-arbre. Ce découpage repose, par exemple, sur une pré-connaissance des éléments sensibles dans le document formant ainsi une unité de protection. Contrairement aux approches présentées précédemment, où la génération du document chiffré repose sur la base de règles, notre idée inverse la méthodologie. C'est la base de règles d'autorisation qui est générée à partir du document chiffré (ou document découpé).

Pour conclure ce manuscrit, nous pensons que ces réflexions, fondées sur nos contributions et travaux antérieurs, confirment l'intérêt d'un axe de recherche pluridisciplinaire qui accélérerait la synergie naissante autour de XML. Nos recherches pluridisciplinaires nous ont permis d'aborder différentes dimensions du problème, de comprendre la variabilité des solutions et de mieux pressentir l'importance et le rôle que XML peut jouer pour la mise en place d'une plate-forme pour l'interopérabilité des applications et des données via les services Web et la définition de solutions communes pour l'accès transparent à des ressources largement distribuées.

Bibliographie

1. .Net, Microsoft, http://www.microsoft.com/net .
2. Abiteboul S., Benjelloun O., Manolescu I., Milo T., Weber R.. Active XML : A data-centric perspective on web services. In M. Levene and A. Poulovassilis, editors, Web Dynamics, pages 275–300. Springer, 2004.
3. Abiteboul S., Benjelloun O., Milo T., The Active XML project : an overview, 17 novembre 2005, ftp://ftp.inria.fr/INRIA/Projects/gemo/gemo/GemoReport-331.pdf .
4. Abiteboul S., Manolescu I., Preda N., Sharing Content in Structured P2P Networks, Journées Bases de données Avancées (BDA 2005), St Malo.
5. Abiteboul S, Quass D., Mcugh J., Widom J., Wierner J., The Lorel Query Language for semi-structured data, International Journal on Digital Librairies, 1 (1997), 68-88.
6. Abou El Kalam. A., El Baida, R., Balbiani. P., Benferhat. S., Cuppens. F., Deswarte. Y., Miège. A., Saurel. C., Trouessin, G., "Or-BAC : un modèle de controle d'accès basé sur les organisations", IEEE 4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003), 2003.
7. ActiveX-Native XML Database Engine / Persistent DOM, http://www.xml.com/pub/p/626 .
8. Adjiman P., Chatalic P., Goasdoue F., Rousset M.-C., & Simon L., "SomeWhere in the Semantic Web" Proceedings of PPSWR 2005 (International Workshop on Principles and Practice of Semantic Web Reasoning). Dagstuhl Castle, Germany.
9. Agrawal R., Kiernan J., Srikant R., Xu Y., "Hippocratic Databases". 28th International Conference on Very Large Data Bases (VLDB), September 2002.
10. Ahmed R., Albert J., Du W., Kent W., Litwin W., Shan M. : "An Overview of Pegasus ", in proceeding of the RIDE-IMS conference, Vienna, Apr. 1993.
11. Ashley. P, Powers. C , and Schunter. M, "Privacy promises, access control, and privacy management". In the third international sympsium on Electronic Commerce, 2002.
12. B. Aiken, J. Strassner, B. Carpenter, I. Foster, C. Lynch, J. Mambretti, R. Moore, B. Teitelbaum, "Network Policy and Services: A report of a workshop on Middleware", Internet RFC-2768, 2000.
13. Bell.E, Lapadula, "secure computer systems : Unified Exposition and Multics interpretation", the MITRE Corporation, Technical report, 1976.
14. Benson E., Wasson G, Humphrey M, Evaluation of UDDI as a Provider of Resource Discovery Services for OGSA-based Grids, International Parallel and Distributed Processing Symposium (IPDPS 2006), Rhodes Island, Greece, April 25-29, 2006.
15. Bertino, E., Castano, S. Ferrari, E. " On specifiying Security Policies for Web Documents with an XML-Based Language", SACMAT'01, 2001, Virginia, USA.
16. Bertino. E., and Ferrari, E. "Securing XML Documents with Author-X", IEEE Internet Computing (2001).
17. Bertino. E., Castano. S., Ferrari. E. and Mesiti. M. "Controlled Acess and dissemination for XML" Workshop on Web Information and Data Management 1999.
18. Bertino. E., Castano. S., Ferrari. E. and Mesiti. M. : "Author-X : A java-Based System for XML Data Protection". In Proc. Of the 14th Annual IFIP WG 11.3 Working Conference on Database Security 2000.
19. Bertino. E., Castano. S., Ferrari. E., Mesiti. M. "Specifying and Enforcing Access Control Policies for XML Document Sources" World Wide Web Journal 3(3), 2000.
20. Bertino. E., Ferrari, E. "Secure and Selective Dissemination of XML Documents", ACM Transactions on Information and System Security, Vol, 5, No.3 August 2002, Pages 290-331.

21. Biba. K.J, "Integrity Consideration for Secure Computer Systems", The MITRE Corporation, Technical Report ESD-TR-76-372 & MTR-3153, 1977.
22. Bouganim, L., Dang-Ngoc, F., Pucheral, P. "Client-Based Access Control Management for XML Documents", Proceeding of the 30th Int. Conf. On Very Large Data Bases (VLDB), Toronto, Canada, august 31-September 3, 2004.
23. Bruno, E. , Le Maitre, J. , Murisasco, E. "Extending XQuery with transformation operators". ACM Symposium on Document Engineering 2003: 1-8.
24. Bulletin de l'ordre des médecins : http://bulletin.conseil-national.medecin.fr/CNOM/bulletin.nsf/(html)/503BOMN503P01?OpenDocument#enjeux.
25. Bumpus W., Strassner J., Weis W. : "The DEN-CIM connection : A roadmap to Directory-Enabled Networks", disponible à l'adresse http://www.dmtf.org, 1999.
26. Busse R., Fankhauser P., Huck G., Klas W., "Federated Schemata with ODMG", In Extending Information Systems Technology, Proceedings of the Second International East-West Database Workshop, Klagenfurt,Austria, September, 1994.
27. Byun. J, Bertino. E, and Li. N "Purpose based access control of complex data for privacy protection". In SACMAT, 2005.
28. Carey M., Florescu D., Ives Z., Lu Y., Shanmugasundaram J., Shekita E., Subramanian S.. "XPERANTO: Publishing object-relational data as XML". In Proc. of the Int. Workshop on Web and Databases (WebDB), 2000.
29. Carey M., Haas L., Scwartz P. et al, "Towards Heterogeneous Multimedia Information Systems : The Garlic Approach", in RIDE-DOM, pages 124-131, 1995.
30. Carminati, B., Ferrari, E., "AC-XML Documents : Improving the Performance of a Web Access Contrôle Module", SACMAT 2005.
31. Cattell R. G. G., Barry D., Bartels D., Berler M., Eastman J., Gamerman S., Jordan D., Springer A., Strickland H., Wade D., The Object Database Standard: ODMG 2.0, Morgan Kaufmann, San Francisco, California, 1997, p. 270.
32. Cattell R.G.G.Ed., "Object Databases : The ODMG-93 Standard", Book, Morgan & Kaufman, 1993.
33. Christophides V., Abiteboul S., Cluet S., Scholl M., From structured documents to novel query facilities. In proceedings of the ACM SIGMOD Intl. Conf on Management of Data, 1996, p. 313-324.
34. Christophides V., Cluet S, Moerkotte G., Evaluating queries with generalized path expressions. In ACM SIGMOD Conf, 1996, p. 413-423.
35. Cho, S., Amer-Yahia, S. , Lakshmanan, L., and Srivastava, D. "Optimizing the secure evaluation of twig queries", VLDB, 2002.
36. Collet C., Finance B., Kedad Z., Tahi F., Laurent D., Bernot G., Bruno G., Vu T-T, Xue X.Vargas-Solar G.. État de l'art des infrastructures de médiation. Technical report, IMAG-LSR, PRISM et LAMI, novembre 2003, 150 pages.
37. Collet C., Finance B., Kedad Z., Xue X., Vu T-T, Bruno G., Belhajjame K., Bobineau C., Jouanot F., Vargas-Solar G., Bernot G., Laurent D., Tahi F., "Towards a Mediation System Framework for Transparent Access to Largely Distributed Sources : The MediaGrid Project", International Conference on Semantics of a Networked World (ICSNW), Springer-Verlag Heidelberg , Paris, 1st Int. IFIP conf., Vol. 3226/2004 , 65, June, 2004.
38. Collet, C, Vu,T-T.: "QBF, a Query Broker Framework for Adaptable Query Evaluation". In Proc. of the Sixth International Conference on Flexible Query Answering Systems (FQAS), June 24-26, Lyon, France.(2004).
39. Component Object Model (COM), Microsoft, http://www.microsoft.com/com .
40. Computer Security Institute, "CSI/FBI Computer Crime and Security Survey", www.gocsi.com/forms/fbi/pdf.html.
41. CORBA Component Model (CCM), 2002, http://www.omg.org/technology/documents/formal/components.htm.
42. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. , Samarati, P. "A Fine-Grained Access Control System for XML Documents", ACM TISSEC 5(2), 2002.

43. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. , Samarati, P, " Securing XML Documents". In proc. of the 2000 International conference on Extending Data Technology (EDBT 2000), Konstanz, Germany, March 27-31, 2000.
44. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P. "Fine-Grained XML Access Control Prototype", http://seclab.dti.unimi.it/~xml-sec/.
45. Data Management : SQL Call Level Interface (CLI), Snapshot, X/Open with SQL Access Group, X/Open Company Ltd.
46. Date C.J., Darwen H., "A Guide to the SQL Standard : A user's guide to the standard database languaue SQL", Fourth Edition, Addison-Wesley, ISBN 0-201-96426-0, 1997.
47. DBGlobe, A data-centric Approach to Global Computing, http://softsys.cs.uoi.gr/dbglobe/ .
48. Delot T., Dechamboux P., Finance B., Lepetit Y., Le Brun G. : "LDAP, Databases and Distributed Objects : Towards a better integration", Int. VLDB Workshop on Databases in Telecommunications, Rome, 2001, p. 140-154.
49. Delot T., Finance B. : "Managing CORBA Objects with Dynamic Behaviour in a Directory", International Symposium on Distributed Objects and Applications (DOA), Rome, 2001.
50. Delot T., Finance B. : "Un service d'annuaire pour la localisation et l'interrogation d'objets CORBA", Les intergiciels, coordinatrice Isabelle Demeure, Hermès Informatique, 2002.
51. Delot T., Finance B., "Génération de Wrappers LDAP pour sources de données relationnelles", Actes des 17 journées des bases de données avancées (BDA), Agadir, octobre, 2001.
52. Delot T., Finance B.: "CMIS-L : A Query Language for Telecommunication Management Systems", Actes des 15èmes Journées Bases de Données Avancées (BDA), p.119-137, 1999.
53. Delot T.: "Interrogation d'annuaires dans les systèmes à objets distribués : langages, évaluation, optimisation", Thèse de doctorat de l'université de Versailles, 18 décembre 2001.
54. Directive 95/46/CE du Parlement européen et du conseil, de 24 octobre 1995, relative à la protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données. Journal officiel n° L 281 du 23/11/1995 p. 0031 - 0050 http://www.privacy.fgov.be/nieuw%2029-8-2002/directive_95_46_fr.pdf.
55. Directory Server Security, www.sun.com/blueprints/1200/ldap-security.pdf .
56. Distributed Management Task Force (DMTF), http://www.dmtf.org/.
57. Document Object Model (DOM), W3C, http://www.w3.org/DOM/ .
58. Dragan F., Gardarin G., Yeh L., "MediaPeer: a Safe, Scalable P2P Architecture for XML Query Processing". Globe'05 workshop Copenhagen. Denmark.
59. DSML (Directory Service Markup Language), OASIS, 2001, http://www.oasis-open.org/committees/dsml
60. Emayr W., "Evaluation Report on Database Security Policies", Technical report, IRO-DB Esprit Project (EP8629), IRO/SPEC/FAW/D8-1.1/WE950418, FAW, Autriche, Avril 1995.
61. Entreprise JavaBeans, Sun, Décembre 2000, http://java.sun.com/products/ejb/.
62. Fan, W. , Chan, C.Y. , Garofalakis, M. "Secure XML Querying with Security Views", SIGMOD, 2004.
63. Fankhauser P., Finance B., Klas W., "IRO-DB : Making relational and object-oriented databases interoperable", International Conference On Extending Database Technology (EDBT), Avignon, France, 1996, March. Pp. 485-489.
64. Ferraiolo.D., Cugini.J., and Kuhn. R., "Role-Based Access Control (RBAC) : Features and Motivations", 11th Annual Computer Security Applications Proceeding, 1995.
65. Ferraiolo.D., Sandhu.R., and Gavrila. S,. "Proposed NIST Standard for role-Based Access Control", ACM Transactions on Information and System Security, Vol. 4, No.3, 2001.
66. Fessy J., Finance B., Lepetit Y., Pucheral P., "Data Management Framework & Telecom Query Service for TINA", 5th International Conference on Telecommunication Systems Modeling and Analysis, Nashville, TN, 1997, March.
67. Finance B., "Basic Query Translator Specification", Technical report, IRO-DB Esprit Project (EP8629), IRO/SPEC/EDS/BF941008, EDS, La Défense, France, October 1994.

68. Finance B., Delot B., Ridaoui A. : “Querying Future Telecommunications Networks”, "Querying Future Telecommunication Networks", 7th International Conference on Information and Knowledge Management (CIKM), Washington D.C., USA, 1998, November, p 226-233.
69. Finance B., Medjdoub S., Pucheral P., "Privacy of Medical records: from law principles to practice", 18th IEEE International Symposium on Computer-Based Medical Systems (CBMS), June 22-24, 2005 in Trinity College Dublin, Ireland, p. 220-225.
70. Finance B., Smahi V., "Object Interoperable Systems : Overview, Comparison, Applications", Object-Oriented DBMS Symposium of the ESDA conference (ESDA), Montpellier, 1996, July.
71. Finance B., Smahi V., Fessy J., "Query Processing in IRO-DB", in the Proceedings of the 4th International Conference on Deductive and Object-Oriented Databases (DOOD), Singapore, 1995, December.
72. Finance. B, Medjdoub. S, Pucheral. P, "The Case for Access Control on XML Relationalships", 14th ACM International Conference on Information and Knowledge Management (CIKM), October 31-November 5, 2005 in Bremen, Germany, p. 107-114.
73. Foster I., Kesselman C., Globus : A Metacomputing Infrastructure Toolkit, Intl J. Supercomputer Applications, 11(2) :115-128,1997.
74. Gabillon, A. "An Authorization Model for XML DataBases". ACM Workshop on Secure Web Services. 2004.
75. Gabillon, A., Bruno, E. "Regulating access to XML documents". IFIP Conf. on Database and Application Security, 2001.
76. Gardarin G, "XML des bases de données aux services Web", Editions Dunod 2003.
77. Gardarin G, Fankhauser P., Finance B., Klas W., Ramfos A. Gannouni S., Pastre D., Legoff R. ,:"IRO-DB: A Distributed System Federating Object and Relational Databases", O.Bukhres and E. Elmagarmid Editors, Object-oriented Multibase Systems, Prentice-Hall, September, 1995.
78. Gardarin G., Finance B., Fankhauser P. : "Federating Object-Oriented and Relational Databases : The IRO-DB experience", in the Proceedings of the 4th International Conference on Cooperative Information System (CoopIS), 1997, June.
79. Gokhale A., Kumar B., Sahuget A., "Reinventing the wheel ? CORBA vs. Web Services", In proceedings Intl. WWW Conference (11), Honolulu, Hawai, USA. http://www2002.org/CDROM/alternate/395/ .
80. Griffiths P., Wade B., "An authorization mechanism for a relational database system", ACM Transactions on Database Systems (TODS), 1976.
81. Halevy A. Y., Ives Z. G., Mork P., Tatarinov I.. "Piazza: data management infrastructure for semantic web applications". In Proceedings of the Twelfth Intl. WWW Conference, Budapest, Hungary, 20-24 May 2003, pages 556–567. ACM, 2003.
82. He, H,. Wong, R,. "A Role-Based Access Control Model For XML Repositories", Proceedings of the First International Conference on Web Information Systems Engineering (WISE'00), 2000.
83. HIPAA Basics: Medical Privacy in the Electronic Age, 2003. http://www.privacyrights.org/fs/fs8a-hipaa.htm.
84. Hitchens, M., Varadharajan, V. "RBAC for XML Document Stores", ICICS, 2001.
85. HL7 : http://www.hl7.org/ .
86. Hurson A., Bright M., "Object-Oriented Multidatabase Systems, Introduction chapter", In O. Bukhres and A. Elmagarmid Editors, Object-oriented Multibase Systems, Prentice Hall, September, 1994.
87. IBM survey of consumer attitudes toward privacy in the United States, the United Kingdom and Germany,1999, http://www.securitymanagement.com/library/ibm_priv.html.
88. ISO/IEC 9595 and CCITT/X710, “Common Management Information Service: CMIS”, 1992.
89. ISO/IEC n°9594 and CCITT/X501, “The Directory models”, 1990.
90. ISO/IEC/SC21/7689, “RDA: Remote Database Access”, 1993.
91. Jagadish H. V., Jones M.A., Srivastava D., Vista D. :“Flexible List Management in a Directory”, Int. Conf. On Information and Knowledge Management (CIKM), 1998.
92. Jagadish H. V., Lakshmanan L. V., Milo T., Srivastava D., Vista D. : “Querying Network Directories”, ACM SIGMOD Int. Conf., 1999.

93. Jajodia, S., Samarati, P., Sapino, M., Subrahmanian, V., "Flexible support for multiple access control policies", ACM TODS, 26(2), 2001.

94. Karjoth. G and Schunter. "A Privacy Policy Model for Entreprises", In the 15th IEEE Computer Security Foundations Workshop (CSFW'02), 2002.

95. Kedad Z., Bouzeghoub M., "Discovering View Expressions from a Multi-Source Information System". Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems (COOPIS), Edinburgh, Scotland, September 2-4, 1999, p.57-68.

96. Kedad, Z., Xue, X. : "Mapping Discovery for XML Data Integration", Int. Conf. Cooperative Information Systems (CoopIS),in cunjunction with OTM 2005, 166-182.

97. Kosman D. : The state of the art in distributed query processing, ACM Computing Surveys (2000).

98. Kudo, M., Hada, S. "XML Document Security based on Provisional Authorization", ACM CCS, 2000.

99. L'helguen-Smahi V., "Optimisation de requêtes dans les systèmes interopérables", Thèse de doctorat de Paris VI, 24 janvier 1997.

100. Landers T., Rosenberg R.L., "An overview of MULTIBASE ", in distributed databases, H.J Shneider editor, North-holland, 1982.

101. LeSelect. http://www.le-select.com.

102. Levy A., Mendelzon A., Sagiv Y., Srivastava D., "The information manifold", AAAI Spring Symposium on Information Gathering, 1995.

103. Lim,C., Park, S., Son, S, "Access Control of XML Documents Considering Update Operations", ACM Workshop on XML Security, 2003.

104. LOI n° 2004-810 du 13 août 2004 relative à l'assurance maladie (1): http://www.legifrance.gouv.fr/WAspad/UnTexteDeJorf?numjo=SANX0400122L.

105. Loi n°2002-303 du 4 mars 2002 relative aux droits des malades et à la qualité du système de santé. http://www.legifrance.gouv.fr/WAspad/UnTexteDeJorf?numjo=MESX0100092L.

106. Manolescu I., Florescu D., Kossmann D., "Answering XML Queries over Heterogeneous Data Sources", VLDB 2001, Roma, Italy.

107. Manolescu I., Florescu D., Kossmann D., Xhumari F., Olteanu D.: "Agora: Living with XML and Relational", VLDB 2000, Cairo, Egypt.

108. Medjdoub S., " Modèles de contrôle d'accès pour XML : application à la protection des données personnelles ", Thèse de Doctorat de l'université de Versailles, Décembre 2005.

109. Merle P., Gransart C., Geib J.-M., "CORBAScript et CORBAWeb – Concevoir, déployer et utiliser des services distribués", 1er Colloque International sur les NOuvelles TEchnologies de la REpartition (NOTERE), p. 53-73, 1997.

110. Miklau, G., Suciu, D. "Containment and equivalence for an XPath fragement", ACM PODS, 2002.

111. Miklau.G, and Suciu.D " Cryptographically Enforced Conditional Access for XML", Fifth International Workshop on the Web and Databases (WebDB 2002).

112. Murata M., Tozawa A., Kudo M., "XML Access Control Using Static Analysis", ACM CCS, 2003.

113. National Computer Security Center, "A Guide to Understanding discretionary Access Control in Trusted systems", 1987.

114. Native XML Databases, http://www.xml.com/pub/a/2001/10/31/nativexmldb.html

115. XACML (eXtensible Access Control Markup Language), OASIS standard, http://www.oasis-open.org/committees/xacml, 2003.

116. Object Management Group, "Security Service", 2003, http://www.omg.org/technology/documents/formal/security_service.htm .

117. Object Management Group, "Naming Service Specification", http://www.omg.org, 1995.

118. Object Management Group, "Object Query Service Specifications", Itasca, Objectivity, Ontos, O2, Servio, Sunsoft, Sybase, Taligent, 1995.

119. Object Management Group, "The Common Object Request Broker : Architecture and Specification", report n° 9305089, disponible à l'adresse http://www.omg.org/, 1993.

120. Object Management Group, "Trading Object Service Specification", disponible à l'adresse

http://www.omg.org, 1997.

121. Papadimos V., Maier D., Tufte K.. Distributed Query Processing and Catalogs for Peer-to-Peer Systems. In CIDR 2003, First Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 5-8, 2003.
122. Parmar, V., Shi, H., "XML Access Control for Semantically Related XML Documents", HICSS, 2003.
123. PICSEL, http ://www.lri.fr/ picsel.
124. Privacy Act of 1974. http://www.house.gov/matheson/the_privacy_act_of_1974.html.
125. Qi, N., Kudo, M,. "Access-Condition-Table-driven Access control for XML Databases", 2004.
126. Qi, N., Kudo, M,. "Tree-based Access Control Mechnism for XML Databases", Digital Engineering Workshop, 2005.
127. Ramfos A., Fessy J., Finance B., Smahi V., "IRO-DB: A Solution for Computed Integrated Manufacturing Applications", In the proceedings of the 3rd International Conference on Cooperative Information System (CoopIS), Vienna (Austria), 1995, May.
128. Ray.I, Ray.In and Narasimhamurthi "A cryptographic Solution to Implement Access Control in a hierarchy and More". SACMAT 2002.
129. Sandhu. R., Coyen. E., Feinstein, H., and Youman., C., "Role-Based Access Control Models", IEEE Computer, 1996.
130. Sartiani C., Manghi P., Ghelli G., Conforti G., "Xpeer : A self-organizing XML P2P Database System", 2004, EDBT Worshops on P2P and Databases (P2P&DB), 2004, p 456-465.
131. Security Assertion Markup Language (SAML), Oasis, Nov. 2002, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security .
132. Shawathe S., Garcia-Molina H., Hammer J., Ireland K., Papakonstantinou Y.,Ullman J., Widom J., "The TSIMMIS Project : Integration of Heterogeneous Information Sources", in proceedings of the IPJS conference, Tokyo, 1994.
133. Skype, http://www.skype.com/products/explained.html.
134. Stoica.A and Farkas.C. "Secure XML views". In Proc. 16th IFIP WG11.3 Working Conference on Database and Application Security, 2002.
135. Tari Z., CraskeG., “A Query Propagation Approach to Improve CORBA Trading Service Scalability”, Int. Conf. on Distributed Computing Systems (ICDCS), 2000.
136. Templeton M., Brill D., Dao S.K, Lund E., Ward P., Chen A.L., Macgregor R., "Mermaid :a Front-end to Distributed Heterogeneous Databases", in proceedings of the IEEE conference on Data Engineering, 1987.
137. Tesch T., Wasch Y., Wenjing L. , "Design Specification for the Global Transaction Manager", Technical report, IRO-DB Esprit Project (EP8629), IRO/SPEC/GMD/D4-6.1/TT951019, GMD-IPSI, Darmstadt, Germany, Oct. 1995.
138. The DataGrid European Project, http://eu-datagrid.web.cern.ch/eu-datagrid/ .
139. The Globus Toolkit Security Service, Alliance GLOBUS, http://www.globus.org/toolkit/docs/4.0/security/ .
140. The Java Naming Directory Interface (JNDI), http://java.sun.com/products/jndi.
141. The MEDIAGRID project, http://www-lsr.imag.fr/mediagrid/
142. The Open Grid Services Architecture, http://www.globus.org/ogsa/ .
143. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification, W3C recommendation 16 April 2002, http://www.w3.org/TR/P3P/.
144. Thomas. R., "Team-based Access Control (TMAC) : A primitives for Applying Role-based Access Controls in collaborative Environnements", RBAC’97, 1997.
145. Tomasic A., Raschid L., Valduriez P., "Scaling Access to Heterogeneous Data Sources with DISCO". IEEE Data and knowledge Engeneering, 10(5):808—823, 1998.
146. Universal Description Discovery & Integration (UDDI), OASIS, http://www.uddi.org/ .
147. Valduriez P., Özsu T., "Principles of distributed databases systems", Second Edition, Prentice Hall, ISBN 0-13-659707-6,1999.

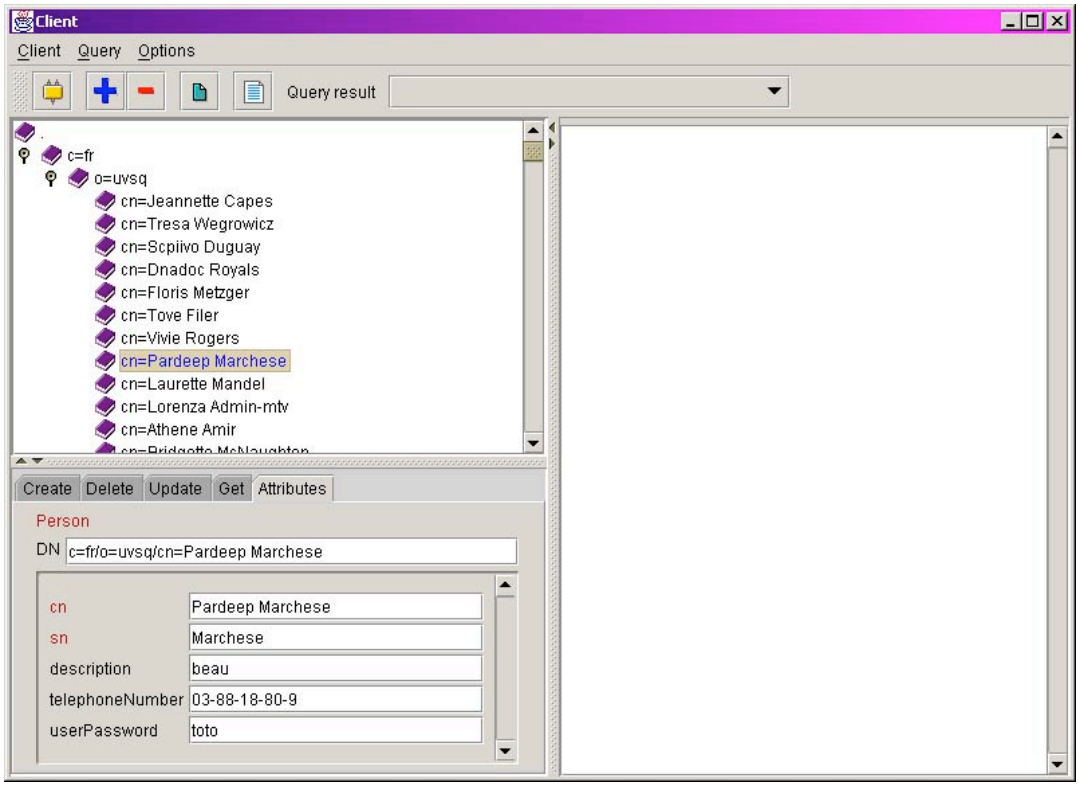
148. Wahl M., Howes T., Kille S. : “Lightweight Directory Access Protocol (v3)”. Internet RFC-2251, 1997.
149. Wang Y., Tan K.L., "A Scalable XML Access Control System", WWW Conference (poster), 2001
150. Wang, J., Osborn, S.L. "A Role-Based Approach to Access Control for XML Databases", ACM SACMAT, 2004.
151. Web Services, W3C, http://www.w3.org/2002/ws/ .
152. XML : http://www.w3.org/XML/.
153. XML Linking Language (XLink) v1.0, W3C Recommendation 27 June 2001, http://www.w3.org/TR/xlink/.
154. XML Pointer Language (Xpointer) v1.0, W3C Working Draft 8 january 2001, http://www.w3.org/TR/WD-xptr.
155. XPath : http://www.w3.org/TR/xpath.
156. XQuery: XML Query Language, W3C, http://www.w3.org/XML/Query/ .
157. Xylème, http://www.xyleme.com/ .
158. Zhang, X., Park, J., Sandhu, R,. "Schema Based XML Security : RBAC Approach ", Seventeenth IFIP 11.3 Working Conference on Data and Application Security, 2003.

Annexe A : Exemple d'intégration de schéma dans IRO-DB

```
                                        // Classes dérivées
 Interface PIECE {
        extent          pièces;
        keys            id;
        attribute       String     id;
        attribute       String  description;
        attribute       Float      prix;
        attribute       Float      poids;
        attribute       String   nom;
        attribute       Float      diamètre;
        relationship    Set <FABRIQUANT> fabriquant inverse FABRIQUANT::pièces_fab;
 }

 Interface FABRIQUANT {
        extent          fabriquants;
        keys            id;
        attribute       String     id;
        attribute       String   nom;
        attribute       String     ville;
        attribute       Float ChiffreAffaire;
        relationship     Set <PIECE> pièces_fab inverse PIECE::fabriquant;
}

                                        // Classes importées depuis le Site 1
 Interface PIECE1 {
        extent          pièce1s;
        keys            id;
        attribute       String     id;
        attribute       String  description;
        attribute       Float      prix;
        attribute       String   nom;
        relationship    FABRIQUANT1 fabriquant1 inverse FABRIQUANT1::pièce1s_fab;
 }

 Interface FABRIQUANT1 {
        extent          fabriquant1s;
        keys            id;
        attribute       String     id;
        attribute       String   nom;
        attribute       String     ville;
        attribute       Float ChiffreAffaire;
        relationship    Set <PIECE1> pièce1s_fab inverse PIECE1::fabriquant1 ;
 }

                                        // Classes importées depuis le Site 2
 Interface PIECE2 {
        extent          pièce2s;
        keys            id;
        attribute       String     id;
        attribute       Float      prix;
        attribute       Float      poids;
        attribute       Float    diamètre;
        relationship    FABRIQUANT2 fabriquant2 inverse FABRIQUANT2::pièce2s_fab;
 }
```

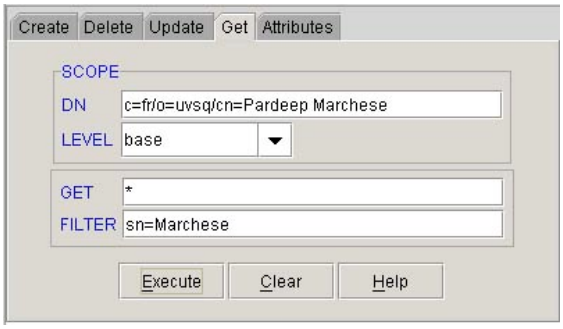
```
Interface FABRIQUANT2 {
        extent      fabriquant2s;
        keys                id;
        attribute           String     id;
        attribute           String   nom;
        attribute           String     ville;
        attribute           Float ChiffreAffaire;
        relationship        Set <PIECE2> pièce2s_fab inverse PIECE2::fabriquant2 ;
}

                                        // Mapping des classes dérivées
    mapping PIECE {
        origin              IMPORT_PIECE1 orig1;
        origin              IMPORT_PIECE2 orig2;

        def_extent          pièces              as      select PIECE (orig1 : p1, orig2 : p2)
                                                        from p1 in pièce1s, p2 in pièce2s
                                                        where p1.id = p2.id;
        def_att   id                    as      this.orig1.id;
        def_att   description           as      this.orig1.description;
        def_att   prix                  as      this.orig1.prix.conversion();
        def_att   poids                 as      this.orig2.poids;
        def_att   nom                   as      this.orig1.nom;
        def_att   diamètre              as      this.orig2.diamètre;

        def-rel   fabriquant            as      select f
                                                from f in fabriquants
                                                where (f.orig1 in this.orig1.fabriquant1)
                                                or (f.orig2 in this.orig2.fabriquant2) ;
}

mapping FABRIQUANT {
        extent              fabriquants;
        origin              IMPORT_FABRIQUANT1  orig1;
        origin              IMPORT_FABRIQUANT2  orig2;

        def_extent          fabriquants         as  select FABRIQUANT  (orig1 : f1, orig2 : f2)
                                                    from f1 in fabriquant1s, f2 in fabriquant2s
                                                    where f1.id = f2.id;
        def_att   id                    as      this.orig1.id;
        def_att   nom                   as      this.orig1.nom;
        def_att   ville                 as      this.orig2.adresse;
        def_att   ChiffreAffaire        as      this.orig2.ChiffreAffaire;

        def-rel   pièces_fab            as      select p
                                                from p in pieces
                                                where ( this.orig1.pièces1_fab = p.orig1)
                                                or ( this.orig2.pièces2_fab = p.orig2) ;
}

                                        // Mapping des classes importées
mappping import PIECE1 {
        imports  external _Piece from Site1 extent piece1s
        origin  orig;
};
mapping import FABRIQUANT1 {
        imports  external_Fabriquant from Site1 extent fabriquant1s
        origin  orig;
};
mapping import PIECE2 {
        imports  external_Piece from Site2 extent piece2s
        origin  orig;
}
mapping import FABRIQUANT2 {
        imports  external_Fabriquant from Site2 extent fabriquant2s
        origin  orig;
}
```

Annexe B : Interface du service d'annuaires étendu



La figure ci-contre illustre les capacités d'interrogation de DQL. La requête recherche la valeur de tous les attributs associés à l'entrée spécifiée par son DN.

Puisqu'il n'est pas facile de se rappeler le DN, l'interface interactive permet à l'utilisateur de sélectionner l'entrée directement dans le DIT(Directory Information Tree). Cette fonctionnalité est offerte pour l'insertion, la suppression et la mise à jour des entrées de l'annuaire que les entrées soient locales ou distantes.



Le résultat de la requête est décrit ci-dessous. Deux formats de données sont possibles. Le premier correspond au format standard d'échange de LDAP (LDIF) et le second correspond au format XML proposé par DSML(Directory Service Markup Language) [59]. Si les requêtes contiennent des expressions de chemin alors on retourne le résultat en XML, car LDIF ne permet pas les résultats imbriqués.

```
dn: schema
require: c
objectclass: Country

dn: schema
require: o
objectclass: Organization

dn: schema
require: cn
require: sn
optionnal: description
optionnal: telephoneNumber
optionnal: userPassword
objectclass: Person
```

```
<dsml>
 <directory-entries>
  <entry dn="schema">
   <objectclass>
                <oc-value>Country</oc-value>
   </objectclass>
   <attr name="Require">
                <value>c</value>
   </attr>
   <attr name="objectclass">
                <value>Country</value>
   </attr>
  </entry>
  <entry dn="schema">
   <objectclass>
                <oc-value>Organization</oc-value>
   </objectclass>
   <attr name="Require">
                <value>o</value>
   </attr>
   <attr name="objectclass">
                <value>Organization</value>
   </attr>
  </entry>
  <entry dn="schema">
   <objectclass>
                <oc-value>Person</oc-value>
   </objectclass>
```

Annexe C - Lexique

CCM : CORBA Component Model

CMIP : Common Management Information Protocol

CMIS : Common Management Information Service

CMIS-L : Common Management Information Service – Language

CORBA : Common Object Request Broker Architecture

COM : Component Object Model

DCOM : Distributed Component Object Model

DNS : Domain Name Service

DQL : Directory Query Language

IDL : Interface Definition Language

IETF: Internet Engineering Task Force

JNDI : java naming Direcctory Interface

LDAP : Lightweight Directory Access Protocol

ODL : Object Definition Language

OMG : Object Management Group

ODMG : Object Data Management Group

OGSA : Open Grid System Architecture

SAML : Security Assertion Markup Language

SNMP : Simple Network Management Protocol

UDDI : Universal Description, Discovery and Integration

WSDL : Web Service Description Language

XML : eXtensible Markup Language

XACML : eXtensible Access Control Markup Language