IXIA (IndeX-based Integration Approach) A Hybrid Approach to Data Integration

Shokoh Kermanshahani

Université Joseph Fourier – Grenoble I **École Doctorale MSTII**

IXIA (IndeX-based Integration Approach)
A Hybrid Approach to Data Integration

THÈSE

présentée et soutenue publiquement le 10 Juillet 2009

pour obtenir le grade de

Docteur de l'Université Joseph Fourier – Grenoble I

(Spécialité : Informatique)

par

Shokoh KERMANSHAHANI

Composition du jury

Président :	Jacques DEMONGEOT	Professeur, Université Joseph Fourier - Grenoble 1
Rapporteurs :	Mohand-Saïd HACID	Professeur, Université de Lyon-Claude Bernard
	Danielle BOULANGER	Professeur, Université de Lyon-Jean Moulin
Examinateurs :	Nicolas LUMINEAU	Maître de Conférences, Université de Lyon-Claude Bernard
	Ana SIMONET	Maître de Conférences, UPMF de Grenoble (Co-directrice)
	Michel SIMONET	Chargé de recherche CNRS (Directeur de thèse)

Laboratoire des Techniques de l'Imagerie, de la Modélisation et de la Cognition
TIMC-IMAG — UMR 5525

Mis en page avec la classe thloria.

Everybody wants to live at the top of the mountain, forgetting that is how we climb is all that matters.

Gabriel Garcia Marquez

Remerciements

Je tiens tout d'abord à exprimer ma reconnaissance à Madame Danielle Boulanger, Professeur de l'Université de Lyon-Joan Moulin, et Monsieur Mohand- Saïd Hacid, Professeur de l'Université de Lyon-Claude Bernard, pour avoir accepté d'être les rapporteurs de ma thèse. Je tiens à remercier Monsieur Jacques Demongeot, Professeur de l'Université Joseph Fourier de Grenoble, qui m'a fait l'honneur de présider le jury de ma thèse. Je remercie également Monsieur Nicolas Lumineau, Maître de Conférences de l'Université de Lyon-Claude Bernard, pour avoir été membre du jury de ma thèse. Je remercie Michel Simonet et Ana Simonet, Directeur et Directrice adjointe de l'équipe OSIRIS du laboratoire TIMC/IMAG pour leur accueil au sein de leur équipe et pour leur encadrement durant ma thèse. Leurs critiques constructives ont été d'un apport déterminant pour l'aboutissement de cette thèse.

Je remercie tout particulièrement mes sympathiques collègues de l'équipe OSIRIS pour leur présence et leur soutien. Un grand merci à Gayo qui m'a guidé lors du début de mon travail. Merci à Radja pour le soutien qu'elle m'a apporté dans les moments difficiles et pour sa main tendue toujours vers ses amies. Je remercie tout particulièrement Houda, Lamis, Samer, Noura et Badia pour leur bonne humeur et pour les discussions intéressantes. Je n'oublierai jamais les très bons moments qu'on a passé ensemble. Je n'oublie surtout pas les aides et les remarques scientifiques de Samer toujours constructives. J'envoie mes remerciements aussi aux anciens membres d'OSIRIS que je n'aurais pas explicitement cités pour leur soutien et leurs encouragements.

Je remercie très chaleureusement mes amies, notamment, Maryam et sa famille qui m'ont soutenu de tous leurs moyens pendant ces années ; Shadi, Nazi, Farideh, Masoumeh, Bahar, Tahereh, Samia, René, Michelle et beaucoup d'autres que je n'ai pas cité ici mais ils sont dans mon cur, pour leur amitié et leur soutien moral.

Je n'oublie pas de citer ma mère, ma sur et mes frères, mon beau-père, mes oncles ainsi que tous mes proches en Iran que leur soutien et leur amitié ont toujours été avec moi, pendant ce travail aussi bien que tout au long de ma vie.

Je voudrais finir par remercier les personnes qui ont le plus contribué à ce travail, mon époux Hamid et mon fils Soroush, mais mes mots ainsi que tous les mots existants dans le monde sont très pauvres pour exprimer ce que j'éprouve dans mon cur pour eux. En conclusion : merci infiniment pour vos sacrifices et pour tout ce que vous m'avez apporté ; et en particulier, je m'adresse à Hamid : je te remercie pour ton amour.

À la mémoire de mon père,
À ma mère,
et à mes chers Hamid et Soroush.

Contents

Part II A Hybrid Approach to Data Integration

List of Figures

Introduction Générale

1 Introduction

L'intégration de l'information pose le problème de combiner et de questionner des données de plusieurs sources autonomes et hétérogènes d'une manière homogène. Un grand nombre d'applications ont été développées et de sources d'informations créées tout au long des décennies de développement de systèmes, que ce soit sur le Web ou dans des entreprises de toutes tailles. On peut répondre à de nouveaux besoins de divers organismes si ces sources existantes partagent leurs fonctionnalités et leurs informations. Cet objectif a une grande importance économique parce qu'en l'absence de ressources partagées, de nouvelles applications devraient être développées pour tous les nouveaux besoins. De plus, dans beaucoup de cas comme le Web, développer une nouvelle application qui contient toutes les tâches et l'information nécessaires apparaît inaccessible ou très complexe.

L'intégration de plusieurs sources locales dans un cadre unique et global est la principale réponse de la communauté informatique pour satisfaire ces nouveaux besoins. Plusieurs technologies ont été développées à cette fin dans le domaine de la recherches aussi bien que dans la pratique. Deux types principaux d'intégration ont été proposés: l'intégration des fonctionnalités et l'intégration des données.

Application Integration Frameworks, Workflow Systems, Digital Libraries, etc., sont des solutions basées sur l'intégration des fonctionnalités. Les entrepôts de données, la fédération de données, et les systèmes d'intégration de données sont des solutions d'intégration des données de différentes sources. Dans cette deuxième catégorie, un système d'intégration répond aux besoins des utilisateurs en intégrant des données de différentes sources sous une sémantique partagée (commune) de sorte que le système d'intégration apparaisse comme une source d'informations indépendante. Pour atteindre cet objectif, beaucoup de défis doivent être surmontés ; l'autonomie, l'hétérogénéité et l'interopérabilité des sources des sonnées doivent être prises en compte et les contradictions des données doivent être résolues.

Un des défis les plus importants pour intégrer différents sources des données autonomes est l'hétérogénéité qui peut apparaître à différents niveaux. Le matériel sur lequel deux sources d'informations sont développées, les protocoles de réseau, le logiciel, les langages de donnée et de requête peuvent être différents. Cependant, l'aspect essentiel et plus complexe de l'hétérogénéité est l'hétérogénéité sémantique. L'hétérogénéité sémantique caractérise les différences dans le sens, l'interprétation ou l'utilisation des mêmes données citent Kashyap96. Plusieurs travaux de recherche ont été effectués pour résoudre les différents aspects de l'hétérogénéité dans un système d'intégration de données.

2 Intégration des données

Dans un système d'intégration de données, les données de différentes sources sont intégrées dans un schéma intégré global qui répond aux besoins des utilisateurs et qui est contrôlé par un système de gestion de données (généralement un SGBD). Tous les hétérogénéités sont cachées pour l'utilisateur, qui questionne le schéma global comme simple schéma de base de données.

2.1 Correspondance entre les schémas

Pour interroger les données de plusieurs sources locales par l'intermédiaire d'un schéma global, ce schéma global doit être relié aux schémas des sources locales. Différentes méthodes ont été développées pour connecter plusieurs schémas locaux à un schéma global : Global-as-View (GAV), Local-as-View (LAV) et Both-as-View (BAV) sont les plus importants. Dans une méthode de correspondance GAV, le schéma global est défini comme vue des schémas des sources sous-jacentes. Dans une méthode LAV, le schéma de chaque source locale est défini comme une vue du schéma global, et dans une méthode BAV, ces deux méthodes sont combinées. Dans la méthode BAV, nous pouvons obtenir un schéma local des relations du schéma global et vice-versa.

2.2 Intégration Matérialisé de Données

Le schéma global d'un système d'intégration de données peut être entièrement matérialisé. Cela signifie qu'une nouvelle base est développée via un système de gestion des données et une copie de toutes les données qui correspondent au schéma global est enregistrée dans cette base. Des données des sources locales sont extraites, transformées et chargées (processus ETL) dans cette base. L'évaluation de questions dans une telle approche est semblable à celle d'une base de données simple et nous avons accès à un langage d'interrogation puissant et à l'optimisation de requêtes. Cependant, on ne peut accéder directement aux données des sources locales et la base de données doit être périodiquement mise à jour. Par conséquent, il y a toujours un délai pour accéder aux dernières données mises à jour. En outre, il est coûteux de construire une nouvelle base et des processus d'ETL.

2.3 Intégration Virtuelle de Données

Le schéma global d'un système d'intégration de données peut être virtuel. Dans ce cas, toutes les données demeurent dans les sources locales et on y accède par l'intermédiaire d'une infrastructure intermédiaire, généralement appelée un médiateur, contenant le schéma global. Ce médiateur décompose ou reformule une requête d'utilisateur sur un ensemble de sources locales. Il recompose les réponses partielles dans une réponse pour l'utilisateur. Les données et les langages d'interrogation des sources locales peuvent être différents de ceux du logiciel personnalisé. Les wrappers contiennent des correspondances entre les schémas, traduisent les données et les questions entre les sources et le médiateur. Avec cette approche, les utilisateurs ont un accès en ligne aux données des sources locales, mais le traitement de requête est complexe et coûteux.

2.4 Approche Hybride pour l'Intégration de données

Une troisième solution possible pour l'intégration de données est de développer un schéma intégré partiellement matérialisé. Une approche d'intégration de données qui emploie un tel schéma global s'appelle une approche hybride. Les approches entièrement matérialisées et entièrement virtuelles d'intégration de données obéissent à des priorités différentes. Dans une approche

entièrement matérialisée, la priorité principale est le temps de réponse aux requêtes, et dans l'intégration entièrement virtuelle de données, la fraîcheur de données est prioritaire.

Cependant, dans beaucoup de scénarios d'intégration de données, différentes priorités peuvent être associées aux différentes données, et un compromis entre le temps de réponse aux requêtes et la fraîcheur de données peut être préféré à la satisfaction d'une seule de ces deux questions. Une approche souple qui permet à quelques données d'être matérialisées et à d'autres données d'être virtuelles satisfait les deux objectifs. Dans les approches hybrides existantes, la vue globale est divisée en une parties matérialisée et une partie virtuelle. Certains objets ou relations sont choisis pour être matérialisés et d'autres restent dans les sources locales et seront extraits au moment de la requête.

2.5 Efficacité du Traitement des Requêtes

Le traitement efficace des requêtes est l'un des défis les plus importants dans l'intégration de données. En raison de la nature dynamique du contexte d'intégration de données, de nouveaux défis surgissent pour optimiser le traitement des requêtes. Au niveau du médiateur, l'optimiseur peut ne pas avoir assez d'information pour décider d'un bon plan et, en outre, au moment de l'exécution, une source peut ne pas répondre exactement comme prévu au moment de l'optimisation [Halevy *et al.*, 2006].

L'optimisation sémantique des requêtes est l'une des manières les plus efficaces de réaliser l'optimisation de requête dans le contexte de l'intégration de données. Dans une base de données simple, l'optimisation de requête est dite sémantique si la transformation de la requête se fonde sur la connaissance sémantique liée au schéma conceptuel de la base de données. Dans un système d'intégration de données, deux niveaux d'optimisation de requête peuvent être considérés : l'optimisation globale pour le plan de requête et l'optimisation locale pour les sous-requêtes qui cherchent les données dans les différentes sources [Hsu and Knoblock, 2000]. La restriction de l'espace de recherche pour une requête, en utilisant la connaissance sémantique au niveau du médiateur, est une solution pour fournir une optimisation globale pour le traitement de la requête. Un des objectifs principaux de cette thèse est de développer une approche d'intégration de données qui se concentre dans son architecture sur la performance du traitement des requêtes. Cette architecture étend un mécanisme sémantique d'optimisation de requête d'un système de bases de données à une plateforme d'intégration de données. -Nous décrivons maintenant notre contribution.

3 Contributions

Nous considérons dans cette thèse considère deux approches différentes de l'intégration de données, toutes deux s'appuyant sur un système de gestion de bases de données orienté-objet.

3.1 Osiview, Une Approche d'Intégration des Données base sur les Vue

Notre premier effort dans cette thèse a été de développer un cadre entièrement virtuel d'intégration de données basé sur une hiérarchie des vues [Kermanshahani *et al.*, 2008]. Nous avons proposé une architecture multi-médiateurs dans laquelle chaque médiateur correspond à une vue. Les sources des données sont classées sous les médiateurs correspondants. Une requête d'utilisateur est envoyée à un médiateur selon la vue sous laquelle elle est classée. Cette classification est faite en employant les contraintes des vues. Classer une requête sous une vue permet de réduire l'espace de recherche pour la requête.

Ce travail est présenté dans l'annexe A. Il est basé sur la plateforme d'Osiris qui est un prototype d'un système de base de base de données et de connaissances.

En étudiant cette solution nous avons constaté que nous pourrions obtenir plus d'avantages et un degré plus élevé d'optimisation sémantique de requête en profitant du mécanisme de classement des objets en Osiris, classement qui est basé sur les contraintes des vues, ce qui a conduit à notre travail sur une approche hybride d'intégration de données, que nous présentons ci-dessous.

3.2 IXIA, Une Approche Hybride pour Intégration des données

L'idée principale de cette thèse est de développer un cadre semi-matérialisé d'intégration de données, qui représente un nouvel aspect d'une méthode hybride [Kermanshahani, 2008].

Motivation

Les approches hybrides existantes fournissent un accès rapide aux données matérialisées. D'autres données demeurent dans les sources locales et sont interrogées directement depuis les sources si nécessaire. Par conséquent, seulement les requêtes portant sur la partie matérialisée du système sont optimisées. Un exemple typique de scénario d'intégration compatible avec une telle approche est l'intégration des informations d'hôtellerie, de tourisme, d'information géographiques et de temps pour une agence de voyage. Dans cet exemple, les données des centres géographiques et de tourisme sont stables et peuvent être matérialisées tandis que d'autres informations telles que des données de temps changent plus fréquemment et sont intégrées d'une façon virtuelle. Généralement dans un tel approche hybride, les données stables et les données qui ne sont pas modifiées fréquemment sont matérialisées et les données qui sont mises à jour fréquemment sont questionnées directement à partir des sources.

Beaucoup d'autres scénarios d'intégration de données peuvent profiter du compromis qu'une approche hybride offre entre le temps de réponse aux requêtes et la fraîcheur de données. Notre motivation principale dans cette thèse est de développer une approche hybride d'intégration de données qui peut étendre l'optimisation de requête à toutes les requêtes du système d'intégration.

En outre, nous proposons une méthode par accroissement flexible de mise à jour incrémentale pour la partie matérialisée du système d'intégration, ce qui nous permet de contrôler le système d'intégration selon les caractéristiques des sources de données qui peuvent changer.

Les seuls critères considérés par des approches hybrides existantes sont la fréquence des requêtes pour certaines données et la fréquence de mise à jour pour d'autres. Il y a également d'autres paramètres intéressants qui peuvent être considérés dans un scénario d'intégration, tels que la puissance d'interrogation des sources, leur disponibilité, et l'importance de leurs données. Notre approche tient compte également de ces différences entre les sources de données.

Nous avons proposé une approche hybride d'intégration de données dans laquelle la partie matérialisée du système dans le médiateur est la structure d'indexation d'objet. Cette structure est basée sur classement des objets de sources qui correspondent au schéma global. L'identificateur d'objet dans la structure d'indexation est matérialisé ainsi que les attributs qui sont nécessaires pour la mise à jour incrémental de cette indexation (attributs classificateurs). D'autres attributs d'objet sont interrogés au moment de la requête.

IXIA, l'approche hybride présentée dans cette thèse, a été développé à partir de la plateforme Osiris. Elle profite de sa structure d'indexation, qui est basée sur un système de classement d'objets. Le classement d'objet en Osiris se fonde sur la hiérarchie des vues et sur une partition de l'espace des objets en utilisant des contraintes des vues. Il offre une optimisation sémantique

de requête en réduisant l'espace de recherche pour une requête à un sous-ensemble d'objets qui peuvent faire certainement ou potentiellement [Simonet, 1996] partie de la réponse à la requête.

Dans le contexte de l'intégration de données, la matérialisation de cette structure au niveau du médiateur peut guider une requête directement vers un sous-ensemble des objets des sources correspondantes. Les objets de ce sous-ensemble font certainement ou potentiellement partie de la réponse à la requête. En d'autres termes, l'utilisation du mécanisme de classement des objets pour les objets du schéma global a l'avantage d'éviter la recherche des sources et des objets qui ne peuvent pas participer à la réponse à la requête.

En conclusion, étant donné la flexibilité de l'architecture d'IXIA, nous pouvons considérer les caractéristiques de différentes sources des données. Ceci est fait en contrôlant les mises à jour incrémentales indépendamment pour chaque source.

Dans la partie I de cette thèse nous avons étudié différents défis et solutions d'intégration de données. Les approches hybrides d'intégration de données sont étudiées dans la partie II. Nous avons également présenté notre architecture hybride, son prototypage et son implémentation dans la partie II.

Introduction

1 Introduction

Information integration is the problem of combining and querying data from several autonomous and heterogeneous sources in a homogeneous fashion. Many applications and information sources have been developed throughout decades of system development, be it on the Web or in enterprises of any size. Many new needs of various organizations can be met, should these existing sources be shared for their functionalities and information. Achieving this goal is important economically because in the absence of shared resources, new applications would have to be developed for all new needs. Additionally, in many cases such as the Web, developing a new application that contains all the necessary tasks and information seems unattainable or very complex.

Integrating several local sources into a single global framework is the main response of the computer community to satisfy these new needs. Several technologies have been developed for this purpose in the research domain as well as in practice. Two main types of integration have been proposed: integration of functionalities and integration of data.

Application Integration Frameworks, Workflow Systems, Digital Libraries, etc., are solutions based on the integration of the functionalities. Data Warehouses, Data Federation, and Data Integration Systems are the solutions for integrating data from different information sources. In this second category, an integration system satisfies the interest of the users by integrating data of different sources under a sharing (common) semantic so that the integration system appears as an independent information source. To achieve this goal many challenges must be overcome; the autonomy, heterogeneity and interoperability of data sources must be considered and the inconsistency of data has to be solved.

One of the most important challenges for integrating different autonomous data sources is the heterogeneity which can appear at different levels. The hardware on which two information sources are developed, the network protocols, the software, the data and the query languages may be different. However the essential and more complicated aspect of heterogeneity is semantic heterogeneity. Semantic heterogeneity characterises the differences in signification, interpretation or utilisation of the same data [Kashyap and Sheth, 1996]. Several research works have been done to resolve the different aspects of heterogeneity in a data integration system.

2 Data Integration

In a data integration system, data from different sources are integrated into a global integrated schema which satisfies the needs of users and which is managed by a management system (generally a DBMS). All heterogeneities are hidden from the user, who queries the global schema as a single database schema.

2.1 Schema Mapping

To query the data of several local sources via a global schema, this global schema has to relate to the schemas of the local sources. Different methods have been developed to map several local schemas to a global schema: Global-As-View (GAV), Local-As-View (LAV) and Both-As-View (BAV) are the most important ones. In a GAV mapping approach, the global schema is defined as a view of the underlying source schemas. In a LAV mapping, the schema of each local source is defined as a view of the global schema, and in a BAV mapping, these two methods are combined. In the BAV method we can obtain a local schema from the relations of the global schema and vice-versa.

2.2 Fully Materialized Data Integration

The global schema of a data integration system can be fully materialized. It means that a new repository is developed by a data management system and a copy of all the data which correspond to the global schema is saved in this repository. Data from local sources are Extracted, Transformed and Loaded (ETL process) to this repository. Query evaluation in such an approach is similar to that of a single database and we have access to a powerful query language and to query optimization. However, in such an approach, the data of local sources cannot be directly accessed and the data repository has to be periodically updated. Therefore, there is always a delay to access the last updated data. In addition, building a new repository and ETL processes are expensive.

2.3 Fully Virtual Data Integration

The global schema of a data integration system may be virtual. In this case, all data remains in the local sources and are queried via an intermediate infrastructure, generally called a mediator, containing the global schema. This mediator, decomposes or reformulates a user query over a set of local sources. It then recomposes the partial responses into a single response for the user. Data and query languages of local sources may be different from those of the middleware. Wrappers which contain schema mappings translate data and queries between the sources and the mediator. With this approach, users have online access to the data of the local sources, but query processing is complicated and time-consuming.

2.4 Hybrid Data Integration

A third possible solution for data integration is to develop a partially materialized integrated schema. A data integration approach which uses such a global schema is called a Hybrid Approach. Fully materialized and fully virtual data integration approaches obey to different priorities. In a fully materialized approach, the main priority is the query response time, and in fully virtual data integration, data freshness is more important.

However, in many data integration scenarios different priorities may be associated with different data, and a trade-off between query response time and data freshness may be preferred to satisfying only one of these two issues. A flexible approach which permits some data to be materialized and other data to be virtual can satisfy both of these goals. In the existing hybrid approaches the global view is partitioned into materialized and virtual parts. Some objects or relations are chosen to be materialized and others reside in the local sources and will be extracted at query time.

2.5 Efficient Query Processing

Efficient query processing is one of the most important challenges in data integration. Because of the dynamic nature of the data integration context, new challenges arise to optimize the processing of queries. At the mediator level, the optimizer may not have enough information to decide on a good plan and in addition, at execution time, a source may not respond exactly as it had been considered at optimization time [Halevy *et al.*, 2006]. Semantic query optimization is one of the most efficient ways to perform query optimization in the context of data integration. In a single database, query optimization is said to be semantic if the transformation of the query relies on semantic knowledge associated with the conceptual schema of the database. In a data integration system, two level of query optimization can be considered: global optimization for query plan and local optimization for subqueries that retrieve data from individual data sources [Hsu and Knoblock, 2000]. The restriction of the search space for a query, using semantic knowledge at the mediator level, is a solution to provide global optimization for query processing. A main objective of this thesis is to develop a data integration approach which focuses in its architecture on the query processing performance. It is done by extending a semantic query optimization mechanism of a single database system to a data integration platform. Below we describe our contributions.

3 Contributions

This thesis considers two different approaches to data integration based on an object-based database system.

3.1 Osiview, View-based Data Integration Approach

Our first effort in this thesis was to develop a fully virtual data integration framework based on a hierarchy of views [Kermanshahani *et al.*, 2008]. We proposed a multi-mediator architecture in which each mediator corresponds to a view. Data sources are classified under the corresponding mediators. A user query is sent to a mediator according to the view under which it is classified. This classification is made by using the constraints of the views. Classifying a query under a view can reduce the search space for the query. This work is presented in annex A. It is based on the Osiris platform which is a prototype of an object-based database and knowledge base system. While studying this solution we found that we could obtain more advantages and a higher degree of semantic query optimization by profiting from the instance classification mechanism of Osiris which is based on view constraints in Osiris, hence our work on a hybrid approach to data integration, which we present now.

3.2 IXIA, A Hybrid Approach to Data Integration

The main idea of this thesis is to develop a semi-materialized data integration framework, which represents a new aspect of a hybrid method [Kermanshahani, 2008].

Motivation

The existing hybrid approaches provide a rapid access to the materialized data. Other data remain in the local sources and are queried directly from the sources when necessary. As a consequence, only the queries to the materialized part of the system are optimized. A typical

example of integration scenarios compatible with such approaches is the integration of geographical data, hotel and tourism information and weather information for a travel agency. In this example, the data of geographical and tourism centers are stable and can be materialized while other information such as weather data change more frequently and are integrated in a virtual manner. Generally in such hybrid approaches stable data and data which are not modified frequently are materialized and the data which are updated frequently are queried directly from the sources.

Many other data integration scenarios can profit from the trade-off that a hybrid approach offers between query response time and data freshness. Our principal motivation in this thesis is to develop a hybrid data integration approach which can extend query optimization to all the queries of the integration system.

In addition, we propose a flexible incremental update method for the materialized part of the integration system. It permits us to manage the integration system according to the characteristics of the data sources which can change.

The only criteria considered by existing hybrid approaches are the frequency of querying for some data and the updating frequency for other data. There are also other interesting parameters which may be considered in an integration scenario, such as the query power of data sources, their availability, and the importance of their data. Our approach also takes into account these differences between data sources.

We have proposed a hybrid data integration approach in which the materialized part of the system in the mediator is the object indexation structure. This structure is based on an instance classification of the sources objects which correspond to the global schema. The object identifier of each object in the indexation structure is materialized together with the attributes which are needed for the incremental updating of this indexation (classifying attributes). Other object attributes are queried at query time.

IXIA, the hybrid approach presented in this thesis, has been developed from the Osiris platform. It profits from its indexing structure which is based on an instance classification system. Instance classification in Osiris relies on the hierarchy of views and on a partition of object space using view constraints. It offers a semantic query optimization by reducing the search space for a query to a subset of objects which can be certainly or potentially [Simonet, 1996] part of the query response.

In the context of data integration, materializing this structure at the mediator level can guide a query directly to a subset of the objects of the corresponding sources. The objects of this subset are certainly or potentially part of the query response. In other words, the use of the instance classification mechanism for the objects of the global schema has the advantage of avoiding the search for both sources and objects which cannot participate in the query response.

Finally, given the flexibility of the IXIA architecture, we can consider the characteristics of different data sources. This is done by managing the incremental updates independently for each source.

In Part I of this thesis we have studied data integration challenges and solutions. The hybrid data integration approaches are studied in part II. We have also presented our proposed hybrid architecture, its prototype and implementation in part II.

Part I

Preliminary Concepts

Introduction

There is a large and growing body of information in the form of autonomous and heterogeneous sources such as documents, data sources and database management systems, which may speak of the same realities.

Many organizations and individuals need to integrate these sources to create a unified system and handle the unified data. Enterprises owning multiple sources of data benefit from integrating their data sources. According to [Haas, 2007], 79% of companies have more than two documents stores and 25% have more than fifteen. Large scale scientific projects need to profit from the data sets being produced independently, for instance, in biological data analysis. Decision making systems, life science and medical information management, among others, use information from several sources. On the World Wide Web, people need to access good quality data and use semantic search across the numerous structured, semi-structured and non-structured data sources [Halevy *et al.*, 2006]. Data integration is the proposed solution to meet this requirement. The issue of sharing and combining heterogeneous data sources has been a problem since the 1960's; however, the new generation of data integration systems has now about two decades of experience.

The goal of a data integration system is to develop a homogeneous interface for end users to query several heterogeneous and autonomous sources. Building such a homogeneous interface raises many challenges among which the heterogeneity of data sources, the fragmentation of data, the processing and optimization of queries appear to be the most important. The heterogeneity of the sources can be present at different levels such as hardware, system, DBMS, data language or query language. It can also be present at a semantic level; the schemas of sources can be heterogeneous. At data level, different names or references can be used to label the same reality. The current literature offers various approaches, each of them advising a solution to all of these problems. There has been a significant progress in individual challenges, but data integration remains an important and difficult task, both theoretically and in practice.

In chapter 1, we briefly discuss the integration of information sources as well as different levels of integration and problems faced by the delivery of a DIS (Data Integration System). We also make a chronological survey of information integration. Chapter 2 discusses the existing approaches to data integration and data integration requirements, in particular schema mapping and query evaluation.

Résumé de partie I

Il existe un important et croissant corps d'information sous forme de sources autonomes et hétérogènes telles que des documents, des sources des données et des systèmes de gestion de bases de données, qui peuvent modéliser les mêmes univers de discours.

Beaucoup d'organismes et d'individus doivent intégrer ces sources pour créer un système unifié et pour manipuler les données unifiées. Les entreprises possédant des sources multiples de données profitent d'intégrer leurs sources de données. Selon [Haas, 2007], 79 % des entreprises ont plus de deux sources et 25 % plus de quinze. Les projets scientifiques à grande échelle doivent profiter des ensembles de données produits indépendamment, par exemple, dans l'analyse de données biologique, les systèmes de prise de décision, les sciences de la vie et la gestion de l'information médicale, entre autres, utilisent les informations provenant de plusieurs sources. Sur le World Wide Web, les gens doivent accéder à des données de bonne qualité et employer la recherche sémantique à travers de nombreuses sources structurées, semi-structurées et non-structurées [Halevy *et al.*, 2006]. L'intégration de données est la solution proposée pour répondre à ce besoin.

L'objectif d'un système d'intégration de données est de développer une interface homogène pour que les utilisateurs interrogent plusieurs sources hétérogènes et autonomes. Établir une interface homogène soulève beaucoup de défis parmi lesquels l'hétérogénéité des sources, la fragmentation des données, le traitement et l'optimisation des requêtes semblent être les plus importants.

L'hétérogénéité des sources peut être présente à différents niveaux tels que le niveau matériel, le niveau système, le système de gestion de bases de données, le langage de programmation ou le langage d'interrogation. Elle peut également être présente à un niveau sémantique : les schémas des sources peuvent être hétérogènes. Au niveau hétérogénéité de données, différents noms ou références peuvent être employés pour désigner la même réalité. La littérature courante offre diverses approches, chacune d'elles proposant une solution à tous ces problèmes. Il y a eu des progrès significatifs dans différents domaines, mais l'intégration de données demeure une importante et difficile, à la fois en théorie et en pratique.

Dans le chapitre 1, nous discutons brièvement l'intégration des émetteurs d'informations ainsi que les différents niveaux de l'intégration, et les problèmes soulevés par la livraison d'un système d'intégration de données. Nous donnons également un aperçu chronologique de l'intégration de l'information. Le chapitre 2 discute les approches existantes d'intégration de données et les principes d'un système d'intégration de données, en particulier la correspondance entre les schémas et l'évaluation des requêtes.

Chapter 1

Data Integration, Information Integration

1.1 Introduction

The necessity of integrating several information sources under a single query interface has been felt little after the adoption of databases in the sixties but it is particullary an active domain from 1985 [Heimbigner and Mcleod, 1985]. The first generation of databases and information systems was developed for single usage but with the growing importance of computer information systems, sharing and combining these sources has become inevitable. Most enterprises or organizations need to share and combine their data. This necessity has still increased with the World Wide Web. One can find a huge amount of data sources which contain information about the same reality and that can be highly different in their structures and semantics. There are structured, semi-structured and non-structured data sources on the Web, and integrating them has led to several new challenges.

"The goal of information integration is to enable the rapid development of new applications requiring information from multiple sources" [Haas, 2007]. This integration can be done at several levels such as data, middleware, or applications; consequently, several technologies are proposed to integrate these sources. The new generation of data integration systems proposes solutions to integrate data at the semantic level. It provides users with a high-level query language and it hides all of the challenges from the end users.

The rest of this chapter is organized as follows: in section 2 we briefly discuss the problems that appear when developing an information integration system. In section 3 we discuss the existing solutions for integrating several data sources. Section 4 makes a survey of information integration in practice (enterprise and commerce). A brief review of the Semantic Web is made in section 5 and the chapter will end with a conclusion.

1.2 Information Integration Problems

Integrating autonomous information sources raises several challenges. Creating an appropriate interface to access the data sources, mapping this integrated view with the source schemas, querying different sources with different performances and different querying power are some of the most important ones. Below we describe three fundamental problems raised when facing the integration of information sources, which themselves cause other important challenges for data integration.

1.2.1 Autonomy

The information sources which participate in an integration scenario are autonomous. It means that the organizational parts that manage and control different information sources are independent. Administrators of information sources often let other systems share data only if control is retained. As a consequence, to build an information integration system the aspect of autonomy has to be considered [Elmagarmid and Sheth, 1999].

The different aspects of autonomy are [Sheth and Larson, 1990]:

- *Design autonomy:* each local information source has its own data model, query language, conceptual schema, semantic interpretation of data, etc.
- *Communication autonomy:* each local information source decides when it communicates with the integration system, i.e. when it responds to the requests.
- *Execution autonomy:* it is not the integration system management that controls the transactions and local or external operations of local sources. In other words, an information source in a data integration scenario does not need to inform other sources of its execution order and its operations.
- *Association autonomy:* local sources have the possibility of associating or disassociating from the integration system, i.e. they decide whether and how their functionalities and data participate in the integration system.

1.2.2 Interdependency

Interdependency implies that data in different information sources are dependent to each other. Different data sources may model overlapping universes of discourse and then contain the same real world objects that may be in different formats. There can also be dependencies between the functionalities of different data sources. The management of interdependent data enforces the use of consistency constraints in an integration system.

1.2.3 Heterogeneity

One of the more complex problems when integrating several autonomous data sources is the heterogeneity of these sources. The information systems which have to be integrated may be developed in different environments and with different conceptual schemas and data definitions. This leads to heterogeneity at different levels of the system. The heterogeneity is independent from the physical distribution of data.

According to [Elmagarmid and Sheth, 1999] an information system is homogeneous if the same software manages data on all sites, data have the same format and structure (same data model) and belong to the same universe of discourse. On the contrary, an information system is heterogeneous if it does not have access to all the characteristics of a homogeneous system. That means it uses different data models, different query languages, different DBMS or languages [Sheth and Larson, 1990].

Richard Hull [Hull, 1997] categorizes heterogeneity into two perspectives: platform (system) and logical (semantic). The platform perspective contains hardware, data model, DBMS and APIs that are supported. Network communication protocols and the standards such as ODBC, JDBC and CORBA can help to manage these kinds of heterogeneity.

Semantic differences mean the different ways similar real-world entities are modelled. Semantic heterogeneity is focused on heterogeneity in the logical representation of data, including different data models, different schemas, and different kinds of data.

We review semantic heterogeneity, which is the most complicated one that must be overcome to achieve a homogeneous view of the sources.

Semantic Heterogeneity

Semantic heterogeneity refers to differences in the meaning and the use of data [Elmagarmid and Sheth, 1999]. It characterises the differences in signification, interpretation or utilisation of the same data [Sheth and Larson, 1990]. To develop a data integration system, semantic heterogeneity must be managed. Generally, semantic heterogeneity is classified into heterogeneity in schemas (or schematic heterogeneity) and heterogeneity in data.

The heterogeneity between two different schemas is significant when there are semantically related data. The schematic conflicts between data sources appear when equivalent concepts are represented differently in the sources. These conflicts are associated to the names, data types, attributes, etc.

Several taxonomies of conflicts have been proposed. Below we briefly review these incompatibilities [Sheth and Kashyap, 1992]:

Domain Incompatibility Problem. This type of conflict arises when two objects have differing definitions of semantically similar attribute domains. Naming conflicts, data scaling conflicts and data precision conflicts are classified in this category.

Entity Definition Incompatibility problem. This incompatibility arises when the entity descriptors used by two semantically similar objects are only partially compatible. Database identifier conflicts, union compatibility conflicts, schema isomorphism conflicts and missing data item conflicts are the scenarios in which the entity definitions of similar entities might conflict.

Data Value Incompatibility Problem. This type of conflict arises due to the values of the data present in different data sources. Known, temporary or acceptable inconsistencies are the cases of this category.

Abstraction Level Incompatibility Problem. This class of incompatibility arises when two semantically similar entities are represented at differing levels of abstraction. Generalization and aggregation conflicts are two cases of this type of incompatibility.

Schematic Discrepancies Problem. This incompatibility arises when data in one source correspond to metadata in another source. Data value attribute conflict, attribute entity conflict and data value entity conflict are considered in this category.

A more simple version consisting of six types of conflicts is considered by Parent and Spaccapietra [Christine and Stefano, 1996]. Below we describe these conflicts:

Classification conflict: A classification conflict appears when two corresponding types describe two different but semantically related sets of a same reality. For example two enterprise sources can contain personal entities, first only for their official staffs and second for all staffs (official, temporary and part-time). The standard solution for this type of conflict is to consider the appropriate hierarchy of generalization / specialization.

Data / Meta-data conflict: This conflict arises when data in one information source correspond to meta-data in another. For example, data in a relational table corresponds to the name of an attribute of a table in another source. The value of the attribute "speciality" in the table *Specialist (id, name, speciality, ...)* in hospital A may be the attribute name of the table *sector-personnel* table *(sector, nurse, generalist, anaesthetist, neurologist, ..)* in hospital B.

Structural Conflict: The structural conflict appears when two corresponding elements are described in different representation levels, with different constraints or with different numbers of attributes. For example when "address" is an attribute (*String* type) of a table in source A and is defined as a table in source B. Another familiar example is represented when the name of a person is defined with an attribute *name* in source A, but in source B is defined by *firstname* and *lastname*. In some references this last conflict is known as *"Schema Isomorphism"* [Elmagarmid and Sheth, 1999]

Descriptive conflict: A descriptive conflict arises when there is a difference between the properties of the corresponding types. Two objects for example can differ by their name, their attributes or their identifiers. A typical example of this conflict is synonymy. It occurs when a set of semantically equivalent objects have different names in different schemas.

Data Conflicts There are also conflicts at the data level. Data conflicts arise when there are differences between the values and different representations in value level for the same data in two data sources. One important kind of different representations is "expression conflicts". We explain this below, through an example. Differences in unit and precision for a same data, and wrong or obsolete data can also cause data conflicts [Kim and Seo, 1991].
Expression Conflict arises when:

- Different words are used to designate the same data: e.g. "Persian-Gulf" and "Arabian-Gulf"
- Different strings are used for the same data: e.g. "N° 21, Gabriel Peri st Martin d Heres" and " Numéro 21, rue Gabriel Péri, Saint Martin d'Hères"
- Different codes are used for the same data: e.g. for the grade in a university: A, B, C, D, E and 1, 2, 3, 4, 5.

Other heterogeneity conflicts are described in several studies and different classifications of the conflicts are proposed. In addition, in the literature, different terminologies are used to explain a same kind of conflict.

[Kim and Seo, 1991] makes a clear and practical classification of schema and data conflicts for structured data that can be extended to all kinds of information integration, see figure 1.1.

Schema Mapping

Schema mapping is currently the main solution to solve semantic heterogeneity in a data integration system. For each source, a mapping is defined between its schema and the global (integrated) schema. Several kinds of heterogeneity conflicts must be overcome in the mapping process. The technology that prepares the mapping between source and target schemas is known as schema matching and the operation is called match. It takes two schemas as input and produces a mapping between the elements of the two schemas that are semantically corresponding. A first solution to schema matching is to do it manually, but it is time consuming and has significant limitations. Several research projects try to develop an automated schema matching approach. Erhard Rahm and Philipe A. Bernstein [Rahm and Bernstein, 2001] give a complete survey of existing approaches in 2001. For a more recent overview, see [Algergawy *et al.*, 2007].

Object Matching

Another heterogeneity problem when developing a data integration system corresponds to a same real-world object that is referred to differently in two data sources. For example, a patient in

- Schema Conflicts
  1. Table-versus-table conflicts
     - One-to-one table conflicts
       (a) Table name conflicts
           * Different names for equivalent tables
           * Same name for different tables
       (b) Table structure conflicts
           * Missing attributes
           * Missing but implicit attributes
       (c) Table constraint conflicts
     - Many-to-many table conflicts
  2. Attribute-versus-attribute conflicts
     - One-to-one attribute conflicts
       (a) Attribute name conflicts
           * Different names for equivalent attributes
           * Same name for different attributes
       (b) Default value conflicts
       (c) Attribute constraint conflicts
           * Data type conflicts
           * Attribute integrity-constraint conflicts
     - Many-to-many table conflicts
  3. Table-versus-attribute conflicts
- Data Conflicts
  1. Wrong data
     - Incorrect-entry data
     - Obsolete data
  2. Different representations for the same data
     (Same representation for different data)
     - Different expressions
     - Different units
     - Different precisions

Figure 1.1: Heterogeneity between data sources comprises different conflicts at schema and data level. This is a classification of schema and data conflicts for relational databases [Kim and Seo, 1991].

Figure 1.2: [Diallo, 2006] Different levels of integration lead to different integration approaches.

source A that is a relational database is defined by a key and he is represented by an Object Identifier in source B. This problem can also arise for two data sources which have the same data model. For example, in two relational databases a same data instance of a person may be referred to by two different keys. "Object Matching" aims at overcoming this problem and different methodologies are proposed for this purpose in data integration systems [Zhou *et al.*, 1995].

1.3 Information Integration Solutions

Historically and depending on the application needs, different solutions have been proposed to integrate several data sources. We can classify these solutions in different manners. We believe that a primary classification can be made depending on the level of integration. The integration of data sources can be made at hardware, software, system, application or semantic levels. Figure 1.2 shows a schema of different levels of integration. In this section we briefly review the most important existing solutions for data integration from this point of view.

1.3.1 Application-specific Solution

In this case, a special purpose application is developed and it has direct access to the corresponding sources and retrieves data from them. It is the application that combines the retrieved data. The integration of data sources with this approach is fast but it is expensive and single-used and the modification in the data sources or the addition of a new source requires a new code to be written.

1.3.2 Application Integration Frameworks

An Application Integration Framework (AIF) provides an extensible framework to integrate a variety of enterprise information systems (EIS) and exchange data between them. In this approach a well-defined interface is provided to the applications to access data of other applications. To add a new data source an adaptor must be written and other applications can use already existing adaptors . The changes in data sources are hidden from the users' applications. When a data source changes, the adaptor may also have to change. Such systems usually use a standard data or programming model such as CORBA or J2EE. Service Oriented Architecture (SOA) is also a class of AIF architectures. In SOA, a group of services can communicate with each other and can be joined to make a new service.

The interface of such systems can be the views that provide a level of abstraction between users' application and the programming level which enables a technical analyst to define its needs without having to know programming.

However, in this technology, to make combination, comparison or analysis of data, the corresponding code must be provided by an application developer. It is still far from the data integration paradigm [Haas *et al.*, 2002].

1.3.3 Workflow Systems for Information Integration

This approach, like AIF (Application Integration Frameworks), is based on the integration of functionalities of different sources. Workflows systems use the flow technology to relate data in the different sources and then to integrate data sources. The result of a function in one application can be the entered data in another application. In other words, a function performed in one source implies other functions to perform in other sources [Leymann and Roller, 2002] and a workflow is the representation of this sequence of operations[1].

Like AIF systems, this approach uses adaptors (connectors) to solve the heterogeneity of data format in different sources. An adaptor accepts data in one particular format of one source and transforms it to a target data format that can be the data format of another data source. Workflows, however, usually use the hub technology. A hub is a space that prepares a common format for all data interchanged between adaptors.

Workflow systems use a more process-oriented model and provide a higher level of abstraction for application developers [Haas *et al.*, 2002]. This offers a higher level of flexibility to EAI (Enterprise Application Integration) systems and it provides some protection to user applications against changes in the sources and environment. However the help it offers to the integration of several external data sources is limited and it is still not a data integration solution.

1.3.4 Digital Libraries and Portals

Digital libraries and portals are the other technologies that prepare a collection or conjunction of results from different sources in response to a user request. Although this is a kind of information integration, we would rather consider these methods in the field of information retrieval categories. Because they collect information from quite different data sources, for any more sophisticated analysis or comparison the application provider must write some code. As examples, see Stanford's InfoBus [Roscheisen *et al.*, 2000] for the architecture of a digital library approach and IBM's Enterprise Information Portal [2] for a portal technology.

[1] http://msdn.microsoft.com/en-us/library/bb245667.aspx
[2] http://www.e-asl.com/downloads/IBM_Enterprise_Info_Portal_v8_Brochure.pdf

In the above approaches, the integration of different information sources is made through the integration of their functionalities. We can consider two main aspects for information integration: data integration and function integration. Whereas data integration means querying data in heterogeneous and autonomous data sources via a homogeneous interface, "Function integration deals with the problem of making local functions from disparate applications available in a uniform manner" [Leymann and Roller, 2002].

Data warehousing and data federation were the first propositions of the database community to realize all the aspects of data integration. A user sends a query to these systems and retrieves the response in a unified database. All heterogeneities are hidden from the user and the system performs schema and data reconciliation.

Below, we briefly review these two technologies and their advantages and drawbacks.

1.3.5 Data Warehouse

The concept of data warehouse [Inmon, 1996, Liu, 2004] was invented in the late 1980s. In such an approach, a new database called warehouse is defined which satisfies an integrated schema of data sources' local schemas. Data is extracted, transformed and loaded (ETL process) to this new database. Users send their queries to this database in a high-level query language. Such a system provides users with the possibility of combining, comparing and analyzing their data [Blakeley *et al.*, 1986].

The query response time in this approach is that of the new database and is then optimized in comparison with all other approaches. The modifications in the sources, however, are not immediately accessible to the users. ETL processing is repeated periodically to refresh the data of the warehouse. The changes in the sources schemas may cause changes in the ETL process, but the DW schema and the user applications are protected from such modifications.

A data warehouse solution is expensive in maintenance and implementation and ETL processing is time consuming. In addition, it is not always feasible to migrate data from their original location to a warehouse repository [Haas *et al.*, 2002].

1.3.6 Database Federation

Federating databases is another solution in the domain of data sources integration. McLeod and Heimbigner [Heimbigner and Mcleod, 1985] publish one of the first papers that define a Federated Database. In database federation architecture, a relational database provides a uniform access to several heterogeneous and autonomous databases in order to allow partial and controlled sharing of their data [Haas *et al.*, 2002].

[Sheth and Larson, 1990] considers a federated database as a collection of cooperating but autonomous component database systems in which the component databases can continue their local operations and be managed by the autonomous DBMSs, while at the same time they participate in a federation. This federated database can be virtual or materialized.

Later, some research papers [Haas *et al.*, 2002] consider that the underlying sources in a federated system are not necessarily databases and it is possible to federate various kinds of sources such as structured, semi-structured or flat files. In addition, according to this last reference, a federated database is a virtual relational database and the sources are queried by the full power of the SQL language. IBM DB2 Integrator (DB2II) [Bruni *et al.*, 2003] is a commercial tool of data federation technology.

The Database federation approach has limitations. The middleware is often supported by a relational database [Haas *et al.*, 2002] but other models have also been used, e.g., the object

model [Hammer *et al.*, 1994] Several rules must be respected to make the correspondence between the schemas of the source databases and the federated schema. Thus the expressiveness of the correspondence and the flexibility of system to integrate different data structure is limited [Poggi and Ruzzi, 2004].

1.3.7 Data Integration

Data federation, as we discussed above, has some limitations. In addition, Web development has led to the integration of a wide variety of data sources for different purposes. This has led the database community to propose another approach derived from the data federation architecture. The mediation approach has a more general definition and a more flexible architecture than data federation. In such an approach a global view is defined and schemas from different sources are combined into this global view. A mapping is defined between the schemas of the local sources and the schema of the global view called "global schema". As in the database federation approach, wrappers are in charge of these mappings.

We briefly recall here that compared with database federation, the virtual data integration approach supplies a higher level of abstraction. In addition, in this approach, the domain of interest can be defined independently from data sources in the global schema and it is also possible to define integrity constraints in the global schema. There are also some data integration projects in which the integrated view is materialized or partially materialized.

To complete our overview, we add that Peer-to-Peer data integration is another virtual approach based on a network of mediators that uses the client-server technology to communicate between data sources and the mediators (see page 27). We discuss data integration with a virtual view as well as the materialized data integration approach in chapter 2. We will also explain how to define multiple partial mediation views instead of one single global mediated view.

Any of the approaches described above cannot alone answer all needs of information integration. Each of them is adapted to some situations and often several technologies are used in parallel or sometimes in combination.

1.4 Information Integration in Practice

There are several technologies to integrate different data sources in enterprise or commercial environments. Broadly speaking, Enterprise Integration (EI) systems can be categorized into Data Warehousing, Enterprise Application Integration (EAI), data federation and Enterprise Information Integration (EII). Application Integration Framework (AIF) and Workflow systems, which we have described in section 3, are EAI approaches. We have also briefly explained data warehousing. The most recent technology is EII. The objective of Enterprise Information Integration is to provide data integration tools for multiple sources without having to first load all data into a central repository.

It was in the late 1990's that the first EII products appeared in commercial products. The need for organization to create external coherent web sites and the desire of people to share data using XML were some reasons to develop such products. Data warehousing solutions were deemed inappropriate for supporting these needs and research projects have been estimated as ready for commercialization [Halevy *et al.*, 2005]. In addition, the cost of data warehousing in many cases was estimated prohibitive.

Although there have been much progress in individual challenges of data integration, information integration in practice lacks a clear view that positions these various technologies relative

to each other. Research projects focus on schema mapping, replication and query processing individually. But practically, information integration is a process consisting of four major tasks: *understanding, standardization, specification* and *execution* [Haas, 2007].

The *understanding* phase can be considered as an analysis process to discover the relative information and statistical properties such as inconsistent values, frequent values or data distributions. Metadata is used to discover and understand the data to be integrated.

In the *standardization* phase, the integrated schema is designed and the terminology and the abbreviations are defined. The rules for data cleansing or repairing may also be provided in this step.

The mapping tools which specify the relationship between the data sources and the integrated schema, as well as the processes to produce data in the desired target form, are generated in the *specification* step.

It is in the *execution* step that integration actually takes place. It can be performed with materialization, federation, mediation or a hybrid approach.

IBM Information Server (IIS) is an example and one of the most complete commercial products for information integration. It consists of a suite of products that together provide a platform which cover all needed tasks for the integration of information sources. The Garlic project [Carey *et al.*, 1995] also developed a federation approach that fits easily to enterprise environment because of extending a relational query processor [Haas, 2007]. Another commercial product of data integration is Oracle Integration [3].

1.4.1 EII and Data Warehousing

The increasing demand for a real-time and less expensive data integration technology led to developing EII approaches. However, this technology will not replace data warehousing. In many cases, we need data to be persistent in a warehouse or a very rapid query response time is necessary.

Another feature to take into account in this comparison is the cost of technology. The cost of an EII is optimal because there is no need for a repository nor for ETL processing, which is expensive. It is necessary to consider that the cost of a warehousing system and its loading process is predictable, but an EII system is not well understood in its performance and its load cost. In addition, an important part of the cost of an ETL process is the human cost for administration tasks in understanding and standardization steps. An EII has the same administration cost for such tasks [Halevy *et al.*, 2005].

1.4.2 EII and EAI (Enterprise Application Integration)

EII and EAI are two different but close categories of integration middleware. Whereas EII focuses on data and the technology of query-response, EAI products try to prepare an environment for applications to talk to each other.

EAI is "integration complete". It means that there is basically nothing that it fundamentally cannot do, but in many cases constructing the EAI business process is like hand-writing a distributed query plan. On the other hand, EII faces several challenges like differences between the performance of query processing and execution powers of different sources.

The big commercial projects use these technologies in parallel or in combination. Nasa Ames

[3] http://www.oracle.com/products/middleware/odi/index.html

[4] and BEA integration projects [5] for example, have a product offering in each of the EAI and EII categories[Halevy *et al.*, 2005].

1.5 Semantic Web

> "I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialize."
>
> *Tim Berners-Lee, 1999*

The current Web can be understood only by people. According to [Berners-Lee et al.,2000] [6] the objective of the Semantic Web is to provide machines with a simple access to information.

In other words, the Semantic Web aims at developing software agents that are capable of reasoning about a subject by accessing various resources. In this context, the Semantic Web must provide an infrastructure in which the information of a variety of sources can be integrated.

The Semantic Web takes advantage of current and past research in data integration [Hacid and Reynaud, 2006]. However, the context of the Web is more dynamic and the integration approach has to adapt to this dynamicity.

There are numerous large distributed information sources on the Web and there is still more heterogeneity in their formats: structural sources as relational data bases, semi-structured data sources such as XML documents or non-structured sources like text documents. There is also heterogeneity in their semantics, e.g., between the conceptual schemas or the ontology that they use. Data integration projects for the Semantic Web generally profit from the mediation architecture.

Ontology-based data integration has been frequently used in the Semantic Web. In this approach, to obtain an automated access to the information contained in the sources, the information items are described by means of metadata provided by an ontology.

Ontology is a term which is widely used in the literature and many different definitions are proposed for it. According to [Gruber and Gruber, 1993] "an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents". In the context of information integration, [Seyhmus, 2009] [7] defines an ontology as follows: "ontologies are knowledge bodies that provide a formal representation of a shared conceptualization of a particular domain". Indeed, it is illusory to believe that a universal ontology can be widely used. Mapping between different ontologies is therefore needed to achieve a homogeneous integrated view. Ontology matching is the main technology to overcome this problem. Data integration approaches based on ontologies or metadata are also known as semantic data integration approaches [Noy, 2004]. We describe in more detail the ontology-based data integration approach in chapter 2.

[4] http://www.nasa.gov/centers/ames/research/lifeonearth/lifeonearth-adis.html
[5] http://e-docs.bea.com/wli/docs81/index.html
[6] http://www.jeckle.de/files/tblSW.pdf
[7] http://iwayan.info/Research/Ontology/Papers_Research/OntoMapping/Sinir_OntologyMappingSurvey.pdf

1.6 Conclusion

The integration of different sources of information is a crucial necessity today. Enterprises, E-commerce, bioinformatics, the Semantic Web and many other sectors are environments that cannot continue to live without integrating their information.

Different challenges exist and several technologies are proposed to integrate and manage heterogeneous sources in a homogeneous way.

In this chapter we have described the fundamental challenges that appear when developing an integration system and we have briefly reviewed the main existing technologies. There are other approaches, among which hybrid data integration approaches which we will discuss in part II of this thesis. Our goal is to highlight that a data integration approach based on a hybrid of materialized and virtual factors is an optimal solution to many integration scenarios.

Chapter 2

Approaches to Data Integration

2.1 Introduction

Information integration of several data sources is performed based on the needs of one or more end users. These requirements are represented by a view called integrated view. There are several research and commercial projects in data integration whereby each proposes an integration approach. According to the integrated view, these different approaches can be categorized into two main areas: materialized and virtual. There are also some hybrid approaches when there is a combination of materialized and virtual factors. We will discuss this third category in the second part of this thesis.

2.2 Materialized Approaches

In materialized data integration, a new physical database is created to integrate and import data from several data sources, usually in one direction only. In some studies, data integration using materialized views is known as "data warehouse", but we believe that these approaches do not comprise all the paradigms of traditional data warehousing.

In this section, we first recall materialized views and we briefly discuss "data warehouse" and "materialized data integration" in the two subsections.

2.2.1 Materialized view

"A view is a derived relation defined in terms of based (stored) relations". A view is materialized if the tuples of the view are stored in a database. Thus a materialized view, like a cache, is a copy of data that can be accessed quickly [Gupta and Mumick, 1995]

View Maintenance

Materialized views must be maintained when the underlying data is updated. This maintenance is made either by re-computing the view or by the use of incremental view maintenance. This second method is often cheaper, because only a part of view is changed in response to changes in the base relations. In this method, data in a materialized view is updated using the periodic logs of underlying data updates.

In some cases underlying sources are not accessible at the update time and materialized views must be updated without accessing the underlying sources. The views which can be maintained

without accessing to the underlying data sources are called Self-maintainable materialized views [Gupta *et al.*, 1996].

2.2.2 Data Warehouse

A Data Warehouse (DW) is a repository which contains a collection of data derived from operational databases (ODBs), legacy systems, and other external data sources. Generally a DW architecture integrates data from several data sources and transforms it in a materialized integrated schema to provide data for reporting, analytical processing, and decision making [Inmon, 1996], (see figure 2.1).

ETL process: The data is loaded in the data warehouse usually in bulk and by the ETL (Extraction, Transformation, Loading) process. Contrary to an operational database, in which data is updated in an on-line manner, in a warehouse approach, the data is not updated in a general sense. Instead, the ETL process is executed periodically to refresh data.

Below we consider the main issues of a DW which distinguish it from a materialized data integration system:

Historical aspect: In each ETL process, the fresh data is loaded into a new snapshot and a historical record of old data is kept in the data warehouse [Inmon, 2005]. This historical aspect is an important characteristic of warehousing technology.

OLAP(On-Line Analytical Processing) and multidimensionality: Analytical processing is another important feature of a DW system. On the logical level, a DW can have a relational or a multidimensional data model, but usually in the conceptual level it uses the multidimensional model. This model is favorable to analytical purposes. Data in a data warehouse and data marts are analysed with OLAP servers. Nigel Pendse [8]" gives a simple definition of OLAP, "*Fast Analysis of Shared Multidimensional Information.* If the DW uses a multidimensional data model, the OLAP tools are directly built on the DW. For a relational DW, a relational OLAP is used that is an OLAP engine with multidimensionality [Wu and Buchmann, 1997].

Metadata management: The last important issue of a DW that we will consider here is metadata management. In a DW, data is integrated from different data sources. Schema mapping and semantic heterogeneity solving are necessary for such a system. Since the historical data is kept, version control information is required for the metadata.

In a DW architecture, materialized views are defined to access DW data more rapidly. Data can be retrieved from an integrated materialized view, or from several data marts which are designed for several specialized purposes. In a bottom-up design approach, data marts are first created to provide specific reporting and analytical capabilities. These data marts will be joined together to create an eventual data warehouse [Kimball and Ross, 2002].

2.2.3 Materialized Data Integration

Materialized data integration means data integration with a materialized integrated view, generally in a database. This field is the one closest to data warehousing [Jarke, 2003].

Like in a DW, in such methods [Zhou *et al.*, 1996, Gupta *et al.*, 1996], data are extracted from data sources and transformed in order to be stored in the materialized view of the integration system. Data refreshing typically is made by n ETL process which must be repeated periodically. Some materialized integration projects use incremental approaches for updating the data of materialized views [Gupta, 1999].

[8] ref: WWW.olapreport.com ,2008

Figure 2.1: Data Warehouse architecture

There are some important differences between a DW and a materialized data integration system. Compared to a DW, multidimensionality is absent in such approaches. Materialized data integration generally is not intended to provide analytical processing, but to provide a homogeneous user view from different heterogeneous and autonomous data sources. In addition, contrary to a DW, the historical data usually are not kept and the new version of data replaces the last one.

The integration of XML sources over the Web is a new research area in data integration. Several research papers use materialized views to integrate XML information sources (see VMIX [Baril, 2006] for an example). The flexibility of XML increases the interest of using XML as a middleware model [Liu, 2004]. More recently, several research projects [Manolescu *et al.*, 2001, Xyleme, 2001] focus on using XML materialized views to integrate different data sources over the Web.

Using materialized views for data integration has appeared in the literature later than virtual data integration. There are several situations in which the materialized approach is preferable to virtual data integration. Some of them are as follows [Hull and Zhou, 1996]:

- When query response time is crucial and high availability is important,
- Where network connectivity is unreliable,
- When for a particular query answer, re-computing queries each time they are needed is more expensive than incremental updating.

The support that materialized integration can provide to "object matching" is another advantage. "Object matching is determined when two object representations (e.g. keys in the relational model) from two different sources refer to the same real-world object"[Hull and Zhou, 1996]. In other words, object matching corresponds to objects in different data sources which are semantically related but are referred to differently. In a materialized approach, object matching can be provided when extracting data into materialized view rather than at the query time.

Figure 2.2: Virtual Data Integration System

Virtual solutions to data integration typically use universal keys to overcome this challenge [Gupta *et al.*, 1996].

2.3 Virtual Data Integration

In the virtual data integration approach, the sources contain the real data and one or several virtual view(s) contain reconciled integrated schemas over these sources [Lenzerini, 2002]. The autonomous and heterogeneous data sources are queried by these homogeneous views. For this purpose, the system needs mappings that describe the relationships and semantic dependencies between integrated and source schemas. Different methods are proposed to make this mapping; we explain these methods in section 2.4.

The query evaluation in virtual data integration makes use of mapping information by decomposition and reformulation of the user query into the queries over information sources and then re-composition of partial answers for a unified response. The query evaluation method depends on the mapping approach that the virtual data integration system chooses. In this section we review the existing virtual approaches for data integration.

2.3.1 Mediation

The mediation approach introduced in [Wiederhold, 1992] can be characterized as an approach that provides an intermediate infrastructure (a middleware) which simplifies query evaluation of several autonomous and heterogeneous local sources [Solar and Doucet, 2002]. This middleware has to overcome the problems of heterogeneity and concurrency, and it must provide query optimization.

Figure 2.2 shows the typical architecture of a mediation system. In this architecture a central mediator is defined as a virtual view and contains a global integrated schema of underlying sources. For each source, a wrapper (translator) is developed which contains the information of mapping between the global schema and the schemas of the corresponding source.

A mediation data integration system uses a common data model and a common query language in the mediation level which may be different from that of some or all data sources [Halevy, 2001, Hull, 1997, Lattes and Rousset, 2000].

Figure 2.3: Main components of a mediator based integration system

The architecture of a mediation approach, however, can be quite different from those of the figure 2.2. Some projects such as TSIMMIS [Chawathe *et al.*, 1994] use several mediators in the mediation level. Applications can request data sources through one or several of them. Figure 2.3 shows a more general architecture of the mediation system. Below we explain the components of the mediation approach in more detail.

Global or Mediated Schema: A data integration system supplies the user with a unified schema to query the various schemas. This unified schema is typically named global schema [Halevy *et al.*, 2006]. Such a schema is a view whose design is based on the user's needs, that presents and exports the data from the mediator integration system. Integrity constraints can be added to a global schema to provide a best description of the domain's needs [Calì *et al.*, 2004].

The DB instance corresponding to the global schema is virtual. The user makes queries in terms of the global schema. The relationship between this global schema and the local schemas in the data sources is specified at the mediation level.

Mediator: [Wiederhold, 1992] defines a mediator as follows: "A mediator is a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications." In other words, a mediator is a software system for an architecture that offers a common query interface to a set of heterogeneous information sources. Sources may (but not necessarily) be cooperating with each other. The set and the number of participating

sources may be flexible and open. The mediator combines and arranges the query results. In addition, the semantic dependencies and the inconsistency between data in different sources must be managed by the mediator.

Wrapper: The wrapper or translator is a module that is responsible for wrapping a data source and providing access to its data, so that the source can interact with the rest of the mediator-based integration system. Wrappers in a mediation system translate data of local sources into the common data language (CMD) of the integration system. They may have to perform some preliminary transformations such as cleaning before sending data to the mediation system. They also translate the queries in query language of the integration system into queries that are adapted to the local sources in meaning and language.

We can summarize the main features of a mediation approach as follows:

- The interaction with the system is realized through queries and the answers to these queries.
- Update operations of any kind are not allowed. Update is made only on the individual database source and it is not possible nor allowed through the MIS.
- Data sources are mutually independent and may participate in different mediated systems at the same time.
- Data is kept in the local, individual sources, and extracted at the mediator's request.
- The system allows sources to get in and out.

An example of a scenario in which a mediation approach is preferable is when searching jobs or apartments or searching a train or airplane ticket. Instead of going to several sources that may have relevant information, the user can query necessary information using a web application that implements a mediation system.

[Lenzerini, 2002] states that the main components of a data integration system are the global schema, the sources and the mapping. Thus it formulates a data integration system $\boldsymbol{I}$ in terms of a triple $\langle \mathbf{G}, \mathbf{S}, \mathbf{M} \rangle$, where

- $\mathbf{G}$ is the global schema, expressed in a language L_G over an alphabet A_G. The alphabet comprises a symbol for each element of $\mathbf{G}$ (i.e., relation if $\mathbf{G}$ is relational, class if $\mathbf{G}$ is object-oriented, etc.).
- $\mathbf{S}$ is the source schema, expressed in a language L_S over an alphabet A_S. The alphabet A_S includes a symbol for each element of the sources.
- $\mathbf{M}$ is the mapping between $\mathbf{G}$ and $\mathbf{S}$, constituted by a set of assertions of the forms

$$q_S \rightsquigarrow q_G;$$
$$q_G \rightsquigarrow q_S;$$

  where q_S and q_G are two queries of the same arity, respectively over the source schema $\mathbf{S}$, and over the global schema $\mathbf{G}$. Queries q_S are expressed in a query language $L_{M,S}$ over the alphabet A_S,and queries q_G are expressed in a query language $L_{M,G}$ over the alphabet A_G.

2.3.2 Peer-to-Peer Data Integration

In basic P2P systems, the files are exchanged. Recently, several research works [Halevy *et al.*, 2003, Bernstein *et al.*, 2002, Calvanese *et al.*, 2004, Chatalic *et al.*, 2006] investigate peer data management so as to evolve the basic P2P system to more complex systems supporting the integration and exchange of structured contents [Giacomo *et al.*, 2007]. Each peer (data source) can select a set of other peers called neighbours and it only provides the semantic mappings to these neighbour peers. It is an advantage in cases where an organization would like to share data but does not wish to make a centralized management.

The P2P data integration system can be helpful in cases where developing a single mediated schema is not obvious or is very expensive. With a P2P architecture for data integration the data in one peer does not require to be replicated in other peers.

Query processing in the P2P data integration system is different. When a query is sent to a peer, query processing is performed by integrating the relevant data in a local peer, neighbour peers and in other necessary peers, according to the P2P mapping.

2.3.3 Ontology-based Data Integration

Using ontology for the integration of information sources is an active area in several disciplines such as Enterprise Application Integration, Data Integration and in particular in the Semantic Web. While there are many definitions for ontology in different categories, in the context of data integration, we can consider an ontology as some formal description of a domain of interest intended for sharing among different applications [Poggi *et al.*, 2008, Noy, 2004]. In other words, an ontology describes the semantics of the data in order to provide a uniform way to make different parts understand each other. One of the first projet in which a mediation system has been developed use of an ontologie is InfoSleuth [Bayardo *et al.*, 1997].

An ontology-based data integration system is a system that provides a conceptual view on top of pre-existing information sources. Here, ontologies are used for the identification and association of semantically corresponding information concepts of data sources. However in several projects, they take on additional tasks. [Wache *et al.*, 2001] consider three directions to data integration approaches using ontology: single ontology approaches, multiple ontology approaches, and hybrid approaches.

In *single ontology architecture*, one global ontology is defined and all information sources are related to this global ontology. SIMS [Arens *et al.*, 1996] and PICSEL [Rousset and Reynaud, 2003] are example approaches that use this kind of ontology-based integration. In such an approach, mapping the data at the sources to the elements of the global ontology is needed [Poggi *et al.*, 2008].

In *multiple ontology architecture*, each data source is described by its own ontology that is independent of others. Because of this independence, such an approach simplifies adding or removing of sources or modifying information sources. But with the lack of a common vocabulary, comparing different sources is extremely difficult. [Mena *et al.*, 1996] is an example of this architecture.

Hybrid architecture has been developed to overcome the drawbacks of the last two architectures. In this approach, each source is described with its own ontology and a shared vocabulary is defined. The sources ontologies are connected and compared using this shared vocabulary. Sometimes this shared vocabulary is a global ontology [Wache *et al.*, 2001]. In this case, ontology mapping [Kalfoglou and Schorlemmer, 2003] is necessary to relate the source ontologies and

the global ontology [9]. Ontology is also used as query model in some data integration approaches.

2.4 Description of the Sources: Schema Mapping

An integrated schema is designed to describe the logic of the interface layer of a data integration system. The local schemas describe the logic of data in the local data sources. Without a mapping between the integrated and the local schemas, the interrogation of local sources via the integrated schema is impossible. Schema mapping refers to transformations between objects in local sources and objects in the integrated schema. This is a necessary process for all schema-based information integration approaches. There are several approaches to mapping between a global schema and the schemas of the local sources: Global-as-View, Local-as-View, Global-Local-as-View and Both-as-View. Below we briefly describe these approaches.

Figure 2.4: Global-as-View method for schema mapping

2.4.1 Global-as-View (GAV)

In the GAV mapping method, the predicates in the global schema are described as views over the predicates in the local schemas. In other words, each entity in the mediated schema is defined as a query over the local sources. The main advantage of this method is the facility of translating a query on the global schema into queries on local schemas. It is done by a simple process of view unfolding. For a user query over the mediated schema, query decomposition is made using mapping views. However, in this method, when adding a new source to the system or changing a local source schema, the mapping predicates must be recreated (See figure 2.4).

[9] http://www.authorstream.com/presentation/aSGuest757-94524-ontology-mapping-survey-science-technology-ppt-powerpoint/

2.4.2 Local-as-View (LAV)

In the LAV [Levy *et al.*, 1996] mapping method, the predicates in the local schemas are described as views over the global schema. In other words, in this method, the entities in each local source are defined as the queries over the mediated schema. Each modification in the mediated schema needs to change all of the source correspondences. The main advantage of LAV is the simplicity of adding a new source to the system. Independently of the rest of the system, the data of a new source are accessible by defining the mapping between it and the mediated schema. Query evaluation with the LAV mapping method is made by reformulating the query in terms of the schemas of the local data sources and is more complex than with GAV. Figure 2.5 shows a schematic presentation of this method.

Figure 2.5: Local-as-View method for schema mapping

2.4.3 Global-Local-as-View (GLAV)

GLAV [Friedman *et al.*, 1999] is a language for source descriptions. Regarding the growing number of sources on the web that can no longer be modelled as a set of relations, but as a set of web data with a set of entry points, modelling websites may need the expressive power of LAV and GAV combined. GLAV was designed mainly for the web data sources but it could also be interesting for data integration independent of web data, because of the flexibility it provides to integrate the diverse sources.

2.4.4 Both-as-View (BAV)

BAV, presented in [McBrien and Poulovassilis, 2003], is another schema mapping method that is based on a transformation procedure. BAV can capture the semantic information that is present in LAV and GAV derivation rules, and it is possible to derive exact or complete LAV and GAV rules from BAV.

In this method the transformation processing is made in two directions: local schemas to global schema and global schema to each local schema. Schema transformation is made incrementally by applying a sequence of primitive transformations consisting of adding, deleting or renaming just one schema construct. The new or deleted element is defined with a query based on the rest of the construct of the schema (See figure 2.6).

Figure 2.6: Both-as View method for schema mapping

2.5 Query Evaluation

Query answering is the principal objective of a data integration system. In such systems, query processing is usually done by means of conjunctive queries. Queries are requested in terms of the mediated schema and they must be translated and decomposed into a set of queries on the local data sources. This translating process is known as the problem of query reformulation and it depends on the method used for schema mapping: GAV, LAV or BAV. For example, "with LAV, the problem of reformulating a query is the problem of answering queries using views" [Chekuri and Rajaraman, 2000, Halevy *et al.*, 2006].

From a modular aspect, query processing in a data integration system can be categorized as follows [Gardarin *et al.*, 2006]:

- Syntactical and semantical analysis of the query including query reformulation
- Decomposition of the query
- Query processing in the local sources, that is the transformation of the query to the local source query language and then the transformation of the result to the integration system common data language
- Re-composition of the partial results considering data consistency

When the reformulation of a query over a global or mediated schema is made over the local sources, it needs to be executed efficiently. But because of the dynamic nature of the data

integration system, especially when working on web data sources, there are several new challenges for making an execution plan. An execution plan describes the method of execution of a query. It is usually represented by an algebraic tree which is a tree whose nodes are the algebraic operations and whose leaves are data sources. For each query, several execution plans can exist which correspond to different possible ways to answer the query. The collection of these execution plans is called *research space*.

Unlike a traditional database setting, a data integration system cannot neatly divide its processing into a query optimization step followed by a query execution step. All sources do not have the same possibilities to answer queries. In addition, the context in which a data integration system operates is very dynamic and the optimizer has much less information than the traditional setting. Thus the optimizer may not have enough information to decide on a good plan, and a plan that looks good at optimization time may be arbitrarily bad if the sources do not respond exactly as expected [Halevy *et al.*, 2006]. [Ives *et al.*, 2004] discuss query processing and optimization in a data integration system.

Query processing and query optimization in a data integration system generally differ with different classifications of integration approaches: materialized or virtual, LAV, GAV, GLAV or BAV; under or without integrity constraints; etc. [Ives, 2002] makes a state of the art of different solutions for query processing for data integration.

2.6 Data Integration Prototype Examples

In this section, we briefly review some main data integration projects. We try to present one or several examples in each category. TSIMMIS and IBIS are mediation integration systems based on GAV mapping. TSIMMIS has a multi-mediator architecture while IBIS uses a global mediator approach including integrity constraints in the global schema. IM is the first global mediation project that uses the LAV mapping approach. AutoMed is a mediation approach based on BAV mapping. Amos II proposes a peer mediator architecture. SIMS, PICSEL and OBSERVER are different ontology-based data integration projects.

Finally, because of the growing use of XML and its importance on the Web, we describe two sample projects in this domain. Today there are many data integration projects that aim to integrate XML documents or to integrate different data sources using an XML integrated view. We briefly review Xyleme, which is a warehousing approach to integrate XML documents, and Agora which integrates relational databases and tree-structured data sources under a global XML schema, using the LAV mapping approach.

2.6.1 TSIMMIS

TSIMMIS[10] [Chawathe *et al.*, 1994] is a joint project of the Stanford University and the Almaden IBM database research group. It is a mediator data integration approach that supports a hierarchical architecture (a network) of mediators and wrappers. Each mediator has its own logical integration schema independent of other mediators and there are several partial integration schemas rather than one single global schema. Each mediator also supports the queries related to the view that it provides and performs the integration independently [Garcia-Molina *et al.*, 1997].

TSIMMIS uses a "lightweight" object model, OEM[11] [Papakonstantinou *et al.*, 1995] as Common Data Model (CDM) for its integration system. A Mediator Specification Language (MSL)

[10] The Stanford-IBM Manager of Multiple Information Sources
[11] Object Exchange Model

and a Wrapper Specification Language (WSL) are defined respectively to specify the mediators and the wrappers. The output data format of the mediators as well as the wrappers is OEM. It provides the possibility of querying the integration system via either mediators or wrappers and when defining a wrapper, one can add a source to the system. Mediators combine and integrate data exported by wrappers as well as by other mediators. The simplicity of OEM makes it potentially capable to support the integration of a large variety of sources in number and structure.

TSIMMIS uses the GAV method for schema mapping. User queries are expressed in MSL or in LOREL (Lightweight Object REpository Language), a specific query language that is an object-oriented extension of SQL. Wrapper and Mediator Generators in this project automatically generate the classes of wrappers and mediators from simple descriptions of their functions. Because of the lack of a global schema, the schema notion is lost and no user has a global vision of the information system.

2.6.2 IBIS

IBIS[12] [Calì *et al.*, 2003] is a semantic data integration approach that fully exploits all available information, including integrity constraints, for query answering. It has a single mediator architecture and uses the Global-As-View approach to mapping. It uses a relational global schema to query the data at the sources. The key issue of IBIS is that it allows the specification of integrity constraints in the global schema.

Typically in GAV data integration systems, the queries over the global schema are evaluated by unfolding each atom of the query using the corresponding view. In IBIS, the presence of integrity constraints over the global schema makes possible to have several potential databases adapted to the data in the sources. To optimize query answering it uses appropriate techniques to limit the number of source access.

Query answering in IBIS is separated into three phases:

- Expanding the query to take into account the integrity constraints in the global schema.
- Unfolding the atoms in the expanded query according to their definition in terms of mapping to obtain a query over the sources.
- Executing the expanded and unfolded query over the retrieved source databases.

The last two steps are the standard phases of query processing in GAV data integration systems. The expansion phase uses the algorithm presented in [Calì *et al.*, 2002].

2.6.3 IM

IM (Information Manifold) [Levy *et al.*, 1995, Levy, 1996] is a data integration system that uses a Local-As-View approach for its schema transformation. In such an approach, query processing is done with query rewriting over the local sources.

Local-As-View (LAV) was introduced in Information Manifold to express that an information source is described as a view expression over the mediated schema. Describing information sources is easier because it does not involve knowledge about other information sources and their relationships. As a consequence, adding a source to the system is simpler. In addition, profiting from the expressive power of the view definition language, it simplifies the description of the

[12] Internet Based Information System

precise constraints on the content of data sources. It is a crucial characteristic of IM and other LAV integration approaches; it offers query optimization which enables the system to select a minimal number of data sources relevant to a query [Halevy *et al.*, 2006].

However, in the LAV approach to IM, the problem of query reformulation boils down to the problem of answering queries using views [Ullman, 1997] which has been studied earlier in the context of query optimization for databases, but it still remains a time-consuming and complex process.

2.6.4 AutoMed

The AutoMed project [Boyd *et al.*, 2004] developed the first implementation of a data integration technique called both-as-view(BAV) [McBrien and Poulovassilis, 2002, McBrien and Poulovassilis, 2003]. In the BAV approach, the mapping between source schemas and a global schema is described as a pathway of primitive reversible transformation steps applied in sequence. One can go from a source schema to the global schema and vice-versa.

In this approach, the schema can be modified incrementally rather than be regenerated. The most essential component in the AutoMed architecture is its metadata repository. It forms a platform for other components of the software architecture. After wrapping a data source, a definition of its schema is saved in this repository. Schema matching tools, template transformation tools and query processor are the other components of the AutoMed architecture that respectively identify related objects in various data sources, generate transformation between the data sources, and retrieve data.

[Boyd *et al.*, 2004] remarks that the unique property of AutoMed is that it does not insist that an entire data integration system be conducted into a single data modelling language. In one single integration system, different domains can be integrated in a modelling language suited to each domain. AutoMed then uses inter-model transformations to connect between the domains.

Standard query optimization techniques which are used in LAV and GAV systems can be applied to the view definitions derived from BAV transformation pathways. Thus query processing in BAV is not significantly slower than other systems [McBrien and Poulovassilis, 2003]. Another particular characteristic of AutoMed is that it can implement a virtual mediator as well as a materialized view for the global schema.

2.6.5 Amos II

Amos II (Active Mediator Object System) is an object-oriented peer mediator system. It has a distributed mediator architecture in which each mediator peer provides a number of transparent functional views of data reconciled from other mediator peers, wrapped data sources, and other data stored in Amos II itself [Risch *et al.*, 2003].

There is no central schema and the mediator peers are autonomous. Several mediator peers communicate over the internet. A mediator peer can consider other mediators as data sources and thus the mediators can be composed into a new mediator peer. It uses a functional data model and a functional query language (AmosQL) where query plans are distributed over several communicating mediator peers. Users can interrogate the system using one or several mediator peers. Different groups of mediators are defined and a special mediator, called "name server", keeps track of what mediator peers are members of each group and other needed information of mediator peers. The Amos II system is fully implemented[13].

[13]Amos II can be downloaded from http://user.it.uu.se/ udbl/amos

2.6.6 SIMS

SIMS [Arens *et al.*, 1996], Service and Information Management of Decision Systems, is a semantic integration system that allows to integrate databases, knowledge bases and Web pages. Instead of defining a global schema and querying it, SIMS creates a domain model that is a global ontology of the application domain, described in LOOM [MacGregor, 1991]. It has a single mediator architecture and uses the LAV method for the mapping process.

Queries are posed in the LOOM language and a subset of SQL. The domain-level query is transformed into a source-level query and during this process the corresponding sources to this query are selected and the system generates a query plan using database abstractions. The system performs global optimization to reformulate the query plan [Arens *et al.*, 1994].

2.6.7 PICSEL

PICSEL [Lattes and Rousset, 2000] is a semantic data integration approach that uses a logical formalism to represent both the domain of application and the contents of information sources. It uses CARIN [Levy and Rousset, 1995] to mix the LAV and GAV approaches in order to avoid the query reformulation problem [Lattes and Rousset, 2000]. CARIN is a family of representation languages that combine the expressive power of Description Logic and Datalog rules. PICSEL integrates relational databases and XML documents.

2.6.8 OBSERVER

OBSERVER [Mena *et al.*, 1996] has a multiple ontology architecture for a global information system. It uses metadata descriptions to model and query the information of several underlying sources. It uses several pre-existing ontologies to describe the source contents using a Description Logic based system. Rather than integrating all pre-existing ontologies into a global ontology, interoperation across these several ontologies is achieved via terminological relationships represented between terms across the ontologies. This architecture provides a "loosely coupled" approach that makes it scalable, extensible and easier to support.

2.6.9 Xyleme

Xyleme [Xyleme, 2001], a Learning Content Management System (LCMS), provides a fully integrated environment for authoring, managing and publishing XML-based documents.

It develops a dynamic warehouse to integrate the massive volume of XML data on the Web. XML is a widely adopted standard to exchange semi-structured data over the Web. Different companies have their own DTDs or XML schemas, which are designed to describe the concept of XML documents in a standard manner for specific areas. Consequently, a simple query may correspond to many different types of DTDs.

As a solution, Xyleme provides a view mechanism to support different domain of interest. The XML documents are partitioned into several clusters, each of them corresponding to a domain of interest. Instead of querying many heterogeneous DTDs, the user queries a single structure which summarizes a cluster by using these views.

Xyleme also provides some form of semantic data integration [Xyleme, 2001]. Xyleme runs on a network of Linux PCs. At the physical level, it uses Ntix [Kanne and Moerkotte, 2000] to save XML data in its repository.

2.6.10 Agora

Agora [Manolescu *et al.*, 2000] presents an architecture based on the LAV mapping to integrate relational databases and tree-structured documents, notably HTML and XML under an XML global schema. In other words, relational and tree-structured data sources are defined as views over an XML global schema [Manolescu *et al.*, 2001]. The internal data model in Agora which is used for the data flowing inside the execution engine remains relational. Therefore, mappings are defined in terms of a generic relational schema which models the generic structure of the XML global schema. Thus tree-structured data sources are modelled by this generic relational schema.

For the query evaluation process, Agora uses XML as the user interface format. Queries are posed in Xquery, which is a standard XML query language developed by the W3C. Queries are then translated to SQL. In Agora, in order to be efficient, query optimization and most query executions are compatible with the relational model and algebra. A GUI has been designed for novice users to pose queries, while expert users can write queries [Manolescu *et al.*, 2000].

2.7 Conclusion

In this chapter we have reviewed the existing approaches for data integration under two main categories: materialized and virtual. We also have discussed schema mapping and query processing, which are two fundamental tasks when developing a data integration system. We have briefly presented the research prototype examples for all the discussed approaches.

In the second part of this thesis we will give a state of the art of hybrid approaches and we will propose an indexed-based hybrid approach for data integration.

Part II

A Hybrid Approach to Data Integration

Introduction

Broadly speaking, data integration approaches are classified into materialized and virtual approaches, which we have reviewed in the first part of this thesis. In the first approach, a materialized integrated view is developed and maintained, a physical repository is created and data are extracted from local sources into this repository. In the mediation (virtual) approach [Wiederhold, 1992], the real data stays in the local sources and one or several virtual schemas are defined which are the interface between users and the integration system. The users pose their queries through this virtual schema.

"The virtual and materialized approaches represent two ends of a vast spectrum of possibilities" [Hull and Zhou, 1996].
In certain applications, we need to profit from the capabilities of these two approaches in a flexible manner. The hybrid data integration approaches [Wiederhold, 1992] have been designed to satisfy such applications.

In the first chapter of this part, we will discuss the history of hybrid approaches in data integration, we will briefly review some studies in this domain and we will also make a summary presentation of IXIA, Indexed-based data Integration Approach, which we have developed and which is presented in detail in chapter 6. IXIA proposes a hybrid architecture to data integration. It develops a mediator which materializes the indexation data of the objects of the underlying data sources. IXIA has been developed based on the indexation system of Osiris, an object-based knowledge base platform. We present Osiris in chapter 4.

In chapter 5 we will give more details about the IXIA architecture and its prototype. An implementation will be presented in chapter 6. An implementation of IXIA has been done for a telecommunication data integration application. In this application we integrate data of public switched telephone network (PSTN) switches, which may be heterogeneous, and a subscriber database.

Résumé de Partie II

En général, les approches d'intégration de données sont classées en approches matérialisées et virtuelles. Dans la première approche, une vue intégrée matérialisée est développée et maintenue, un base physique est créée et des données sont extraites dans cette base à partir des sources locales. Dans l'approche de médiation (virtuelle) [Wiederhold, 1992], les données résident dans les sources locales et un ou plusieurs schémas virtuels sont définis qui constituent l'interface entre les utilisateurs et le système d'intégration. Les utilisateurs posent leurs requêtes au travers de ce schéma virtuel.

Les approches matérialisées et virtuelles sont deux extrêmes d'un large spectre possible pour un scénario d'intégration de données [Hull and Zhou, 1996].
Dans certaines applications, nous avons besoin de profiter des possibilités de ces deux approches d'une façon flexible. Les approches hybrides d'intégration de données [Wiederhold, 1992] ont été conçues pour satisfaire de telles applications.

Dans le premier chapitre de cette partie, nous discutons l'histoire des approches hybrides dans l'intégration de données, nous présentons brièvement quelques études dans ce domaine et nous faisons une présentation sommaire d'IXIA, l'approche d'intégration de données basée sur une méthode d'indexation, que nous avons développée et qui est présentée en détail au chapitre 6. IXIA propose une architecture hybride pour l'intégration de données, en développant un médiateur qui matérialise les données d'indexation des objets des sources locales. IXIA est développé sur la base du système d'indexation d'Osiris, une plateforme objet-basée de bases de données et de connaissances. Nous présentons Osiris au chapitre 4.

Au chapitre 5 nous donnons plus de détails sur l'architecture d'IXIA et son prototype. Une implémentation est présentée au chapitre 6. Cette implémentation d'IXIA a été faite pour une application d'intégration de données de télécommunication. Dans cette application nous intégrons des données des switch publics du réseau téléphonique (PSTN), qui peuvent être hétérogènes, et d'une base de données de clients (subscriber).

Chapter 3

IXIA: A hybrid framework for data integration

3.1 introduction

In this chapter, we present a partially materialized (hybrid) framework for a data integration system. It provides a query optimization to the integration system as well as a flexibility of data refreshing for different data sources, according to the needs of the integration application.

In the first section we try to explain why and in which cases a hybrid data integration approach is preferable. We give a state of the art for such approaches and we review some examples. In the second section, we give a rapid presentation of the architecture of IXIA, the hybrid data integration approach we have developed. We will also discuss the efficiency of its query answering compared to a fully virtual integration approach and its flexibility of data refreshing in comparison with a fully materialized one.

3.2 Hybrid Approaches: State of The Art

In data integration systems, there is a trade-off between the query response time and data freshness. Fully materialized and fully virtual approaches favor one of these objectives. From another point of view, in enterprise applications for example, the trade-off is between the cost of building a warehouse and the cost of a live query. The needs of different integration applications are very different and a one-size-for-all approach is often unsuitable [Halevy *et al.*, 2005]. Hybrid approaches have been developed to create optimal and more flexible integration systems. They try to give the possibility of materializing some underlying data and querying others in a virtual manner.

In many cases in an integration process, for some data of underlying sources, modifications are frequent and users need to know immediately if there are updates. In these cases data have to remain in the sources and be extracted at query time, which leads to a time-consuming query processing. In the same integration process, some data may change less frequently or users can support a certain delay of updating for these data. Materializing this second category of data optimizes the query processing of the integration system. Such an application was the main motivation to develop hybrid integration approaches. The hybrid approach to data integration was introduced in [Hull and Zhou, 1996] with the Squirrel project. It develops a general and flexible mediator approach which allows a relation in the integrated view to be fully materialized, fully virtual or partially materialized.

Figure 3.1: Squirrel data integration architecture.

Other efforts for the hybrid approach are made in [Zhu, 1999], [McHugh, Widom 1997], [Alasoud *et al.*, 2005]. Here we briefly review these approaches.

3.2.1 A hybrid Approach based on Squirrel Integration Mediators

Squirrel integration mediators were developed initially to support only fully materialized integrated views [Zhou *et al.*, 1996]. Later [Hull and Zhou, 1996] they were extended and generalized by developing a framework to support virtual as well as hybrid integrated views for relational and object-oriented models.

In a Squirrel hybrid integration mediator, some relations can be materialized, some can be virtual and others are hybrid. A hybrid relation is a relation in which some attributes are materialized and others are virtual; they are queried at query time. Materialized data are maintained by incremental update. Figure 3.1 shows the architecture of the Squirrel hybrid integration mediator.

A View Decomposition Plan (VDP) is a central construct of a Squirrel-generated integration mediator which specifies materialized, virtual and hybrid relations that the integration mediator will maintain. It also provides the basic structure to support incremental maintenance of materialized data and querying of virtual data. In other words, like a query execution plan, VDP presents the decomposition of queries. However contrary to query execution plans which are typically developed on a query by query basis, VDPs are relatively static and "one VDP can be migrated to another VDP as a result of optimization". Different VDPs may be appropriate for the same integrated view.

Materialized data updates which are received from the source databases are hold in the

update-queue to be used by the Incremental Update Processor. Incremental updates are represented as deltas.

The query response time for queries that request only the materialized attributes is the query response time of a fully materialized approach. For queries which request some materialized and some virtual attributes the query response time is a function of those of each attributes but it is still optimal compared to a fully virtual approach. The hybrid squirrel mediator integrates relational and object-oriented data sources. One of the challenges for such an approach is assuring the consistency of data extracting from materialized and virtual parts of the system. [Hull and Zhou, 1996] discuss how the developers have overcome this challenge.

3.2.2 Semi-structured Data Integration with the Lore database system

Another hybrid approach has been designed to integrate the semi-structured databases with the Lore system. The Lore (Lightweight Object Repository) [McHugh *et al.*, 1997] database management system has been designed specifically for managing semi-structured data with OEM (Object Exchange Model) data model.

The Lore hybrid integration project uses the Lore platform to develop a data warehouse and it adds a virtual part to the system to query the external data required for the system. A DW has some data that never or infrequently update, for example geographical data or hotel data for a travel agency. Other data, like weather data in contrast, change very frequently and may be important for the decision making system based on DW as well as for the client of travel agency. Lore materializes the more stable data in a Lore database and it provides the Lore's *external data manager* to query external data such as weather information. It makes the distinction of local and external data invisible to the user.

Figure 3.2 shows the architecture of the Lore data integration system. When a query arrives to the system, a query execution engine sends the requests for objects to either *Object Manager* or *External Data Manager* to be handled. The Object Manager requests data in the Lore database. The external data manager integrates the data of external sources and also functions as the integrator between data in an external data source and data of local databases.

3.2.3 A Hybrid Framework to Warehousing the Web Contents

[Zhu, 1999] proposes a hybrid approach to warehousing the Web contents. Data on the Web are usually updated frequently. The objective of this project is to overcome the challenge of DW maintenance when developing a DW on the Web. It selects a set of stable or frequently queried data to materialize in a DW containing historical data. External data, which are the complementary Web data for DW and are frequently modified, are queried by a virtual method. A query can be applied on the DW or by using the virtual approach as well as by using a combination of warehouse data and Web data.

It uses the MIX model (Metadata based Integration model for data X-change) to represent data together with a description of their underlying interpretation context. With this model, data and metadata are interpreted using domain-specific ontologies. The mapping rules are defined by a system-specific ontology in order to semi-automatically generate transformation codes. An ontology engine makes the mapping rules between Web data and attributes in the data integration system. These mapping rules are used for the virtual approach as well as for DW maintenance.

Figure 3.2: Architecture of the Lore data integration system.

3.2.4 A Hybrid Approach for Ontology Integration

A hybrid approach for ontology integration is also discussed in [Alasoud *et al.*, 2005]. Its main difference with [Zhu, 1999] is that the external data sources can be ontologies, web data, or any form of structural or semi-structured data sources. It supports fully materialized, fully virtual, and hybrid integrated views. Ontologies are used as formalism for the integrated view. For mapping between the concepts of the source ontologies and the integrated schema, this approach uses a LAV method in which the concept of the source ontology corresponds to a query of the integrated view ontology.

3.3 IXIA, IndeX-based data Integration Approach

Like other hybrid approaches, the main objective of IXIA [Kermanshahani, 2008] is to make a trade-off between query response time and data freshness in a data integration application. However, there are some important differences between the hybrid methodology that we use in IXIA and the ones which we reviewed in the last section. Below we make a global presentation of the IXIA architecture. We begin with explaining our motivation to choose this architecture. The details of the prototype and the implementation of IXIA are explained in chapters 5 and 6.

3.3.1 Motivations

In all the hybrid approaches described in the last section some objects, relations or attributes of the integrated schema are implemented virtually and others by a materialized method. All data

manipulation for data in the virtual parts is done in a virtual manner; data remain completely in the data sources and are extracted at query time. Similarly, for data of the materialized view, all data manipulation is done by materialized paradigms; data are extracted to a repository and the user queries this repository. Generally, in these hybrid approaches, a user query can be decomposed into virtual and materialized parts.

This kind of partition of the integration system into virtual and materialized parts has advantages when some data change frequently, some queries are repetitive, or the query response time for some queries is crucial. An example of applications compatible to such approaches is a travel data integration system in which some data are rarely changed and others, like the airline data, change frequently, and accessing on-line data is very important.

In such approaches, however, the query processing for virtually implemented parts of the system remains time-consuming and materialized data refreshing is similar to that of a fully materialized approach.

Our effort in this thesis was to develop a hybrid approach which optimizes the query response time for all the queries of the integration system. The middleware is an object database. The idea is to materialize some semantic knowledge of the data of the underlying sources in the mediator. The materialized part of the mediator in this approach is the object indexation structure based on an instance classification of the sources objects which correspond to the global schema. This materialization permits us to be guided directly to a subset of the sources and objects which can be part of the response to the query (certainly or probably). The Objects attributes remain in the local sources and are extracted at query time.

Reducing the query response time by some materialization implies data refreshing for these materialized parts. The data refreshing process is done to refresh the indexation data. Some data that are needed for this refreshing process are also materialized.

Our other important motivation when developing this approach is to consider the differences between sources' characteristics and to provide a flexibility of data refreshing depending on the sources data and capabilities. We focus on the applications in which data sources do not have the same level of data importance; some data sources have data that change frequently or the freshness of their data is much more important than others. In addition all the sources may not have the same querying power or the same availability. To achieve this goal and also to achieve the best trade-off between data freshness and query response time, depending on each application, we make the data freshness process of each data source independent of others; it can also be done with a different frequency.

3.3.2 Global Architecture

Like a mediator approach, IXIA has a mediator-wrapper architecture, although with some materialization. IXIA has been developed based on the Osiris system in order to take advantage of its object indexation system. Osiris is an object-based database and knowledge base system based on a hierarchy of views where views are similar to concepts defined by logical properties, like in a Description Logic approach [Roger *et al.*, 2002]. We will present the main features of Osiris, notably its object classification method, in chapter 4.

As mentioned above, the main materialized part of the IXIA is the indexation structure which is based on the instance classification of Osiris. A direct advantage of this materialization is query optimization for the integration system.

Figure 3.3.2 shows a global presentation of the IXIA architecture. Only the relevant modules of Osiris are shown here. We briefly describe this architecture in the following two subsections:

Figure 3.3: IXIA data integration architecture.

Indexation and Indexation Maintenance

After defining the integrated schema (an Osiris schema), the classification server makes a first object indexation for all the sources objects which correspond to the global schema and sends the indexation data to be saved in the Osiris indexation module.

The indexation data are then incrementally updated by the classification server. The "Modification Detector" modules detect if there is some updating in the sources which results in updating the indexation data from the last indexation maintenance. The Modification Detector of each source functions independently and can be executed with different frequencies.

Updating information obtained from the modification detectors is sent to the "Source Modification Manager" module of the IXIA classification server. This module adds the source information and prepares the "indexation repairing message" for the "Osiris Classification Server", which does the indexation maintenance just as in a single Osiris database.

We note that mappings between the object indexation and data in local sources are made in the wrappers. We save the (oid, primary-key) correspondence between the Osiris objects of the global schema and the data in sources. Wrappers also do the mapping between the local sources' schemas and the Osiris Global Schema.

Query Processing

Query processing in Osiris and consequently in IXIA is done using the indexation information and provides a query optimization.

Depending on the method of schema transformation (LAV [Levy, 1996], GAV [Chawathe *et al.*, 1994], BAV [Jasper *et al.*, 2003]), a query decomposition / reformulation is made in a virtual approach; query processing is done at the source level, then a mediator composes the partial responses with respect to data consistency and sends a unified response to the user. These processes are time-consuming. In most mediation approaches, the procedures involved in the query evaluation process are executed at query time.

In IXIA, some procedures associated with query evaluation are executed in an off-line manner, thus reducing the query response time. We call such procedures pre-procedures, and they consist of the indexation process and maintenance. In other words, in IXIA the problem of analyzing and processing a query is transformed into the problem of object indexing, refreshing this indexation, and searching the response objects' attributes in the sources. The query decomposition and the generation of the execution plan are done by the "Query Evaluator" module of the IXIA query processor. The partial queries are sent to the Osiris Query Processor to find the satisfied objects using the object indexation system. Re-composition of partial responses into a final response is also done by IXIA Query Evaluator.

The object indexation system of Osiris also takes advantage of the hierarchy of views of Osiris in its structure (see chapter 4). This implies reducing of the search space at the mediator level in our integration approach.

3.4 Conclusion

In this chapter, we have briefly introduced a hybrid approach to data integration. Like a mediation approach it has mediator-wrapper architecture but some information is materialized at the mediator level.

Contrary to the other hybrid methods [Hull and Zhou, 1996, Zhu, 1999, McHugh *et al.*, 1997, Alasoud *et al.*, 2005], we do not choose some relations or view/queries to be materialized and

leave the others virtual. Instead, we use a materialized object indexation at the mediator level which reduces the response time for all the queries of the Data Integration System. The materialized indexing information is refreshed using a Modification Detector over each source. Data refreshing at the indexation level can be done with different frequencies for different sources. In many cases the refreshing delay after information modification can be shortened so that it can be considered as an on-line access.

Chapter 4

OSIRIS

4.1 Introduction

The Osiris system is a database and knowledge base system which implements the P-type data model. The concept of P-type has been created in 1984 by Ana Simonet [Simonet, 1984] with two principal objectives: data sharing through the views and automatic verification of integrity constraints. In this chapter we describe the principal characteristics of the Osiris system and we focus on object indexation and query evaluation, which are two main issues used in IXIA.

4.1.1 The P-type concept

The P-types data model, where **p** stands for the French word ***p**artagé*, which means *shared*, had been first designed in a database management perspective but it later proved to be adapted as well to knowledge base needs. Compared with other data and knowledge representation models, the novel characteristic of P-types is that views, which define a *point of view* on a family of objects, constitute its central concept: a p-type is not defined first, and then its views, but a P-type is defined *through* its views.

An object is an instance of one and only one p-type, but it can belong to several of its views and change the views it belongs to during its lifetime. Classifying an object into the views of its p-type is a characteristic inherent to this model. This is why the Osiris system, which implements the p-type model, can offer functionalities for decision support, alert management, semantic query optimization, etc. In our work, we went deeper into the use of the p-type concept with the purpose of profiting from its object indexation system to develop an indexed-based data integration system.

A p-type makes possible the definition of a collection of objects of the real world that have similar semantics and that can be perceived according to several viewpoints, called views, by different categories of users. The views are organized in a hierarchical manner (they form a *lattice*) and the top view plays a particular role. It is named the *minimal view* and it defines the minimal properties (attributes and constraints) an object must satisfy to belong to the p-type. It defines the necessary and conditions for an object to belong to a p-type.

The views are built in a top-down manner: first the minimal view is defined, and then its other views through a strict enrichment of the minimal view or of views that have already been defined.

4.1.2 The object concept

The *very notion* of an object (of the real world) shared and possibly perceived through various viewpoints by different categories of users implies that the *identity* of an instance (a computer object) can be defined. Identity is one of the concepts that can be defined in the minimal view of a p-type. It must be defined when the computer application deals with objects shared by different categories of users.

An object[14] belongs to one and only one p-type and satisfies some of its views, among which the minimal view. When the object is updated it has to be classified again. Its views in the new state may be different from those in the previous state but in any case the object continues to belong to the minimal view of its p-type.

In the following, we present the main notions of the p-type model and their formalization. Then we present the main functionalities of the Osiris system, which implement the p-type data model.

4.1.3 General presentation

The Osiris system implements the p-type data model. This model was initially designed in the perspective of a Database Management System favoring the sharing of information through viewpoints (views). The objects of a given domain are grouped into semantic families, namely P-types, and within a p-type an object can be perceived according to different views. A view is defined within a p-type and it is characterized by logical properties expressed through attributes[15] and constraints (also called assertions or axioms). Except the minimal view, which is the root of the view lattice, a view specializes one or several other views, i.e., specialization can be single or multiple. These logical properties constitute the necessary (for the minimal view) or necessary and sufficient (for other views) conditions for an object to belong to a view.

When an object is (imperatively) assigned to a view, the properties of the view play the role of integrity constraints to be verified. The methodology used for the checking of these constraints makes also possible to determine all the views that are *valid* for an object, which corresponds to *instance classification* in knowledge based systems. This methodology determines as well the views whose properties are not satisfied by the object to be classified, thus leading to *invalid* views. The views that are neither valid nor invalid are *potential*, which may happen when some attributes (classifying attributes) have unknown values. A view that is *potential* for an incomplete object at a given time will evolve towards the *valid* or *invalid* status as unknown attributes become known.

The objects of a p-type can be modified, which may happen as a consequence of the modification of some attribute values or when an unknown attribute is assigned a value. *At each object update it is automatically reclassified*, which causes the determination of its valid, invalid and potential views in its new state.

The methodology used for instance classification relies on a partitioning of the object space, called the *Classification Space*. This partitioning and the resulting Classification Space are obtained through a static analysis of all the logical properties defined in the whole set of views of a p-type. This partitioning also supports an indexing method that indexes all the objects of the database and that naturally leads to a semantic optimization of queries. Note that contrary to DL-based optimization processes[16] it is not necessary to classify views through the subsomp-

[14] In this chapter, the term object will denote a computer object. The terms real object or object of the real world will be used explicitly to designate objects of the real world.

[15] An attribute can be seen as an existential constraint.

[16] Optimisation processes based on subsumption in DLs (Description Logics).

tion relationship, which avoids a process with complexity NP while offering similar advantages [Simonet, 1996].

4.2 Specification of a P-type

A p-type is specified by providing its functions and its constraints. In the following we make an informal presentation of the p-type notion and we continue with a formalization of this concept. The language in the examples is that of the Osiris system, which implements the p-type model.

Attributes In Osiris, an attribute is a unary fonction whose domain is the p-type and whose codomain is either a usual type (predefined, structure, collection) or a p-type.
Examples of attribtes are given below :

```
age : INT
adress : struct(num : INT; street : STRING, postCode : INT; town : STRING)
adress : ADRESS
lastName : setOf(STRING)
address : setOf(struct(num: INT; rue: STRING, cPostal: INT; ville: STRING))
friends : setOf(struct (friend : PERSON ; tel : STRING))
owner : setOf(PERSON)
isOwner : setOf(CAR)
```

Classifying Attributes A classifying attribute is an attribute whose value is used for the classification of an object of a p-type. By default, in a p-type, an attribute appearing in at least a domain constraint or an Inter Attribute Dependency is a classifying attribute. Classifying attributes are at the very center of classification, indexing and query semantic optimization processes.

Constraints The semantic of a p-type is expressed through constraints, which are called assertions in the Osiris language. In the design of the p-type data model a particular attention has been paid to constraints dealing with restrictions to the domain of an attribute. They are divided into three main categories

1. Elementary Domain Constraints (EDC),
2. Inter-Attribute Dependencies (IAD),
3. View constraints (VC).

Domain Constraints Elementary Domain Constraints, which we call Domain Constraints in the following, express domain restrictions on the domain of an attribute.

Inter-Attribute Dependencies (IAD) are Horn clauses whose literals are elementary domain constraints (such as defined above) and logical connectors are the logical operators *and*, or and *not*.

Some examples of IADs are given below :
$age \in [25, 65]$ – An elementary domain constraint is a particular case of an IAD
$age \leq 16 \rightarrow MilitaryService \in \{notdone\}$
$age \in [5, 14] \rightarrow ownsVehicle \in$(BICYCLE or MOTORBIKE)

View Constraints View Constraints (VC) are propositional formulas over views (of the same p-type), built using the logical connectors and, or and not.

Examples of view constraints are given below :

```
SALARIED
SALARIED or SCHOLAR
STUDENT and not SENIOR
```

Views in OSIRIS The views of a p-type are provided in an incremental and modular way : one first specifies the root view, called the *minimal view*, then its other views through a strict enrichment of views that have already been specified. Enrichment can be simple or multiple.; along with strict restrictions of the codomain of functions inherited from the parent views, new functions and new constraints can be defined.

Minimal view The specification of the minimal view of a p-type consists in providing its name, its attributes and its constraints. Besides domain constraints and IADs (Inter-Attribute Dependencies), the unicity constraint can also be provided. It enables the system to determine whether two instances represent the same real object. The default name of a p-type is that of its minimal view (see example below).

Non minimal view A non-minimal view is obtained by enriching one or several other views of a p-type. Enrichment (also called specialization) can be single (one parent view) or multiple (several parent views). In the example below, figure 4.1, TEACHER is a view obtained through single specialization whereas TEACHING-ASSISTANT is obtained through multiple specialization.

Figure 4.1: Views specialization hierarchy

4.3 An Example of a P-type

The default name of a p-type is that of its minimal view. In figure 4.3 we define the p-type PERSON and some views.

Figure 4.2: Views domain inclusion

We can also define the next views:

```
view ADULT : PERSON  assertions  age ≥ 18   end ;
view MINOR : PERSON  assertions  age < 18   end ;
view SENIOR : PERSON  assertions  age ≤ 65   end ;
view FEMALE  : PERSON  assertions  sex = "f"   end ;
```

This example will be used to illustrate the instance classification function : according to his age or his sex, a person will be automatically placed in these views. The views are not exclusive and an object can belong simultaneously to several views. Figure 4.1 presents the hierarchy of the p-type PERSON and figure 4.2 presents the set inclusion of the sets of interest of this p-type.

4.4 Classification Space

Elementary Domain Constraints (EDC) and Inter-Attribute Dependencies (IAD) are the basis of the *Classification Space* of a p-type. It is an n-dimensional space, where n is the number of classifying attributes of the p-type. The classification space of a p-type is built through a partitioning of the object space that is defined by the EDCs and the IADs of the p-type.

In the annex B we have presented a simple and complete example of a P-type specification and all steps of object classification and query processing in Osiris.

Stable SubDomains In a p-type T, for each attribute $Attr_i$ let P($Attr_i$) be the set of elementary predicates on $Attr_i$ that appear in all the assertions of all the views of T. Each elementary predicate has the form $Attr_i \in D_{ik}$ where D_{ik} is a subset of the domain of definition Δ_i of $Attr_i$.

An elementary predicate $Attr_i \in D_{ik}$ determines a *partition* of Δ_i into two elements: D_{ik} and (Δ_i - D_{ik}). The product of all the partitions [Stanat and McAllister, 1997] defined by the predicates of P($Attr_i$) constitutes a partition of Δ_i whose blocks d_{ij} are called Stable SubDomains (SSD). A SDS verifies the *stability* property.

Stability property of an attribute : When the value of the attribute $Attr_i$ of an object o_k varies within a SSD d_{ij}, the object o_k continues to satisfy exactly the same predicates of P($Attr_i$).

```
view PERSON // minimal view of P-type PERSON
  attr name :  STRING;
    father :  PERSON;
    sex :  CHAR in { f, m }; // assertion a1
    birthdate :  DATE;
    age :  INT in ]0,140] calc ; // assertion a2
    militaryService :  STRING in
        {no, ongoing, done, deferred, exempted}; // assertion a3
    own-car :  setOf (CAR); // CAR is a view of p-type VEHICULE reverse property;
    key name // Extenal key, base of the unicity constraint
  assertions
    a4:  age < 18 ==> MilitaryService = "no";
    a5:  age ≥ 18 and
        sex = m ==>MilitaryService in {ongoing, done, efferred, exempted};
    a6:  sex = f ==> MilitaryService = "no";
end ;
// An OID attribute (toid) is created automatically in the minimal view
view TEACHER : PERSON // TEACHER specializes PERSON
  attr salary :  REAL;
    affectation :  UFR;
    grade :  STRING ;
    indice :  INT;
  assertions
    a7:  grade in {maitreConf, professor, monitor, Teaching-Assistant, secondDegree};
    a8:  indice ≤ 1200 ;
end TEACHER ;
view STUDENT : PERSON
  attr diplomes :  setof STRING in { high school, A.A., A.S.,
            B.A.,B.S., M.S., M.A., degree, Doctoral Program, Adult Studies}
    studies :  STRING in { graduate, postgraduate, thesis};
  ...
end STUDENT;
view MONITOR : STUDENT, TEACHER // specialize STUDENT and TEACHER
  assertions // and then it defines a subset of them
    a9 :  age ≤ 27
    a10 :  state = monitor;
    a11 :  studies = thesis;
    a12 :  diplomes contain "master2";
end ;
view SPORTSMAN : PERSON // specialize PERSON
  attr isSportsman:  BOOL;
  ...
end SPORTSMAN;
```

Figure 4.3: P-type PERSON.

Consider the set of IADs defined in all the views of the p-type PERSON:

```
a1   sex : char in  f, m ;
a2   age : int in ]0,140];
a3   MilitaryService : string in no,ongoing,done,deferred,exempt;
a4   age < 18 ==> MilitaryService = no;
a5   age ≥ 18 and sex = m ==> MilitaryService in ongoing , done,
                                                    deferred, exempt;
a6   sex = f ==> MilitaryService = no;
a7   age ≤ 27;
a8   status = teaching-assistant;
a9   studies = thesis;
a10  diploms contain "DEA"
a11  age ≥ 18 ;
a12  age < 18 ;
a13  age ≥ 65 ;
a14  sex = f;
```

The first three assertions define the domain of the attributes *sex*, *age* and *MilitaryService* in the P-type PERSON, because they are defined in its minimal view. The assertion a9 defines only the domain of the attribute *studies* in the view TEACHING-ASSISTANT. Hence the domain of definition of *studies* in the p-type PERSON is " thesis " ∪ OTHER ∪ UNDEFINED ∪ UNKNOWN, where OTHER is a generic set of strings whose exact value is meaningless. OTHER potentially contains all the strings different from " thesis ".

Below are the products of the partitions for the attributes *age*, *militaryService*, *sex* and *studies*, determined from their elementary predicates[17]:

```
age : d11 = [0, 18[, d12 = [18, 27], d13 = ]27, 65[, d14 = [65,140]
sex: d21 = {f}, d22 = {m}
militaryService : d31= {no}, d32 = {ongoing,done,deferred,exempt}
studies: d41= {thesis}, d42 = other, d43 = undefined
```

An element of the partition of an attribute constitutes a stable subdomain (SSD) of the attribute, written d. We note SSD_{ATTR} the set of all the stable subdomains of the attribute *Attr*. In the considered example, the SSDs of the attributes are:

$SSD_{age} = \{d_{11}, d_{12}, d_{13}, d_{14}\}$
$SSD_{sex} = \{d_{21}, d_{22}\}$
$SSD_{militaryService} = \{d_{31}, d_{32}\}$
$SSD_{studies} = \{d_{41}, d_{42}, d_{43}\}$

The partition of the domain of each attribute of a p-type is prolonged into the partition of its object space and it constitutes the Classification Space of the p-type. Considering that the n classifying attributes of a p-type have been partitioned into stable subdomains, the Classification Space of a p-type is a subset of the Cartesian product of its SSD:

$SSD_1 * SSD_2 * ... * SSD_i * ... * SSD_n = \{< d_{1i}, d_{2j}, ...d_{nk} > | d_{1i} \in SSD_1 ... d_{nk} \in SSD_n\}$
where SSD_j represents the set of stable subdomains of the attributes $Attr_j$, for $j \in [1..n]$.

[17] For these attributes, the stable subdomain corresponding the " unknown " value has been voluntarily omitted.

Figure 4.4: Eq-Classes of the p-type PERSON

The classification space is a n-dimensional space where each element, called Eq-Class (for **Eq**uivalence **Class**) is a hypercube represented by a n-tuple of SSDs. For graphical representations we limit ourselves to the 3D space. Thus, considering the three attributes *sex*, *age* and *MilitaryService*, the Classification Space of the p-type PERSON is represented by :

The valid Eq-classes of a p-type are those that satisfy the assertions of the minimal view. They are represented in bold in figure 4.5. For example, the Eq-class (d_{14}, d_{22}, d_{32}), that contains among others the object (age=70, sex="m" et militaryService = "done") is valid, whereas any object of the Eq-class (d_{11}, d_{22}, d_{32}) is invalid, because any person aged less than 18 ($age \in d_{11}$) can only satisfy d_{31}.
The stability property of an attribute can be extended to the whole Classification Space:

Stability property of an Eq-class : all the objects of the same Eq-class have the same validity for all the views of a p-type.

Corollary 1 : when an attribute of an object is modified while remaining in the same Stable SubDomain, the object continues to satisfy the same predicates, hence the same assertions and consequently the same views.

Corollary 2 : when several attribute of an object are modified while remaining in their initial Stable SubDomains, the object continues to satisfy the same predicates, hence the same assertions and consequently the same views. As two objects of the same Eq-class satisfy the same assertions, and consequently validate (or invalidate) the same views, it is possible to determine *a priori* the views which the objects of an Eq-class satisfy. As a consequence, it is possible to associate with each view the set of Eq-classes that validate it.

Semi-domain or Generalized Eq-class Object sharing underlies the manipulation of objects with missing values: a user who creates an object knows at most the data specific to the views

Figure 4.5: Classification Space of the p-type PERSON

through which he perceives this object. Therefore it is quite normal that a shared object has unknown attribute values during its lifetime. In such situations, the Stable SubDomains of an attribute with an unknown value (in short: an unknown attribute) are themselves unknown. In order to manipulate these objects, we have created a SSD whose domain is the domain of the attribute itself.

A semi-domain or *Generalized Eq-class* of a p-type T characterized by n classifying attributes is a tuple $<d_{1f}, d_{2l}, d_{ik}, . \ . \ ., d_{np}>$ such that $\ni i | d_{iq} = \Delta_i$, noted "*", for $i \in [1, n]$, where Δ_i represents the domain of the i^{th} attribute of T.

An *explicit component* of a semi-domain is an element different from "*". A Semi-domain all the elements of which are explicit components is an Eq-class.

4.5 Object Classification

Classifying an object into the views of its p-type is done through a network structure called Classification Network. Broadly speaking objects are classified into the n-dimensional cubes whose axes are the stable sub-domains.

4.5.1 Principle Network

The classification network of a p-type is a three-layer connectionist network without learning. The cells of the first layer represent the Stable SubDomains of the p-type: the cells of the second

layer represent the Eq-classes and the cells of the third layer represent the views.

The static architecture of the classification network is the following :

1. each SSD cell is connected to all Eq-class cells of which this SSD is a component.
2. each Eq-class cell :
   - has input arcs representing the SSDs of the classifying attributes of the p-type (one SSD per classifying attribute),
   - implements a logical AND of its input values,
   - has ouput arcs connected to all the View cells it validates.,
3. each View cell :
   - has input arcs connected to all the Eq-classes that validate it,
   - performs a logical OR of its input values,
   - delivers a Boolean result.

During object classification :

1. The cells representing a valid SSD have a *true* ouput; those representing an invalid SSD have a *false* output,
2. an Eq-class cell whose all input values are *true* has a true output value ; other Eq-class cells have a *false* output value,
3. a View cell whose at least an input value is *true* has a *true* output value ; other View cells have a *false* output value.

4.5.2 Classification methodology

In Osiris, classifying an object in a p-type is equivalent to classifying its Eq-class. Hence :

1. for each known attribute, the valid SSD (corresponding to the attribute value) is determined,
2. the corresponding Eq-class[18] is deduced,
3. the corresponding tuple is classified by using the network

Figure 4.6 shows a simply presentation of object classification with our simple example.

Classifying complete objects : VI classification

A complete object is an object for which the value of all the classifying attributes are known. The result of the classification process is a set of **V**alid views and a (complementary) set of **I**nvalid views, hence the name of VI classification.

Let **o** be an object of which the values of the attributes $ATTR_1, ATTR_2, ..., ATTR_n$ respectively belong to the SSDs $d_{12}, d_{22}, ..., d_{n2}$ (cf. figure 4.6). Its Eq_class is $< d_{12}, d_{22}, ..., d_{n2} >$. The network of figure 4.6 shows that this object is valid for the views V1 and V2, and Invalid for the views V3 and V4. The same conclusion stands for all the objects in the same Eq-class.

[18] Or the Eq-classes of the object is not completely valued.

Figure 4.6: Principle Classification Network

Classifying incomplete objects : VIP classification

An object **o** is said to be incomplete when the value of at least a classifying attribute is unknown.

An incomplete object **o** does not belong to one Eq-class, but potentially to several ones. The set of these potential Eq-classes constitute its *semi-domain*.
In order to take into account incomplete objects, an adaptation of the classification methodology and the corresponding network has been made [Bassolet, 1998]. The result is **VIP** classification, where P means Potential. Potential views are views that are neither Invalid (none of the known attributes contradicts the constraints of the view) nor Valid (the known attributes are not sufficient to demonstrate that the set of constraints of the view is logically valid). A Potential view may become Valid or Invalid during the lifetime of the object, when some unknown values become known.

4.6 Object Indexing

An indexing structure called ISD (Indexing Structure Descriptor) is defined for each p-type [Simonet et al., 94]. Its main fields are:

1. A vector of Stable SubDomains, which represent an Eq-class or a semi-domain,
2. A vector of views providing the status (Valid, Invalid, Potential) of each view for the objects indexed by this ISD,
3. A reference to the set of objects of the ISD,

4. the total number of objects indexed by the ISD.

When an object is created, it is classified. Classifying an object needs that the SSDs of the classifying attributes of the p-type have been previously determined. When the value of a classifying attribute $ATTR_i$ is unknown, the system assigns it a *generalized* SSD that spans the whole domain of $ATTR_i$. When the SSDs of the object are determined, it is classified and the Osiris system determines the validity of each view of the p-type for this object. A view is:

- **V**alid, if all its assertions are verified,
- **I**nvalid, if at least one of its assertions is contradicted,
- **P**otential, if the available data is not sufficient to determine that it is valid or invalid..

We recall that potential views can be found only in the case of incomplete objects. This potential notion is used in the query processing of Osiris. Once the object has been classified, the Osiris system can determine its ISD. If the ISD does not exist yet (i.e., the current object is the first one with this ISD) it is created and the object is associated with it. When an object is modified, two situations can occur: it does not change any SSD and therefore remains in the same ISD, or at least one SSD changes and the object changes its ISD (a new ISD may then be created).

It may happen that two distinct ISD share some Eq-classes. For example, if d_i* represents the generalized subdomain of $ATTR_i$, the ISDs

$$ISD_{12} :< d_{11}, d_{23}, d_{35}, V, V, I, V >$$

and

$$ISD_{20} :< d_{11}, d_{2*}, d_{35}, V, P, I, P >$$

have in common the Eq-class $< d_{11}, d_{23}, d_{35} >$. When an attribute becomes known, the object changes its ISD and the new ISD is necessarily *subsumed* by the initial one.

4.7 Query Evaluation

4.7.1 General form of queries

In this paragraph we consider only questioning queries. In Osiris questioning queries are distinguished from queries that can create new P-types from existing objects. In [Scholl *et al.*, 1991] questioning queries are called *object-preserving*, as opposed to *object-generating* queries. The general form of an Osiris questioning query is:

$$(Context|Condition)[Attr_i]^{0,*}$$

Where:

- *Context* is a Constraint on the views of a p-type,
- *Condition* is a logical condition on the attributes of the views of the context part. Conditions have the same form as Elementary Domain Constraints and Inter-Attribute Dependencies.

- The result of a query is either a collection of oids, or a collection of tuples

  $< Oid, ATTR_i, ATTR_j, ... >$, where $ATTR_i, ATTR_j, ...$ are the named attributes of $[Attr_i]^{0,*}$.

Below are examples of queires :

```
(STUDENT and SPORTSMAN | Age > 23)
(TEACHER or SPORTSMAN | Age > 40 and Income > 10 000) [name, age]
(TEACHING-ASSISTANT and not SPORTSMAN)
(STUDENT and not SPORTSMAN)
```

When a query is labeled, it defines a **dynamic view**. Contrary to the views that define a p-type, the constraints defined in the condition part of a dynamic view do not contribute to the partitioning of the Object Space of a p-type.

4.7.2 Evaluation

Queries are evaluated in four steps:

1. Determination of the ISDs corresponding to the query when rewritten in terms of the SSDs of its attributes. In other words; determination of the ISDs proper to the query,
2. Determination of the actual ISDs that are subsumed by the ISDs of the query,
3. For each actual ISD, automatic (if it is subsumed by a valid ISD) or semi-automatic (if it is subsumed by a potential ISD) extraction of the objects of the ISD.
4. Projection of the resulting objects onto the attributes of interest of the query.

ISDs proper to a query

The determination of the ISDs proper to a query is made in four steps :

Step 1 : the query is transformed into a disjunction of conjunctive subqueries. A conjunctive query is a query iff in its *condition* part the only logical operator is AND, and the operands are either elementary predicates on attributes or views, or negated elementary predicates.
For each subquery in conjunctive form, steps 2 to 4 are applied.

Step 2 : determination of the Eq-classes of the ISDs of interest.
The Eq-classes belonging to the sets **V**alid, **I**nvalid et **P**otential are determined. These Eq-classes are often des semi-domains, obtained in the following manner:

1. For each attribute that is explicitly named in the query, determine the set of its valid, invalid and potential SSDs.
2. Build the Eq-classes (and more generally semi-domains) of interest for the query. Their corresponding tuples are the result of the Cartesian product of the valid and potential SSDs obtained in 1),

3. Assign a truth value (SV or PV) to each Eq-class obtained in step 2. The truth value of an Eq-class is obtained as a conjunction of the truth values of the SSDs that compose it:
   - an Eq-class is *valid* iff all its SSDs are valid for the query. This Eq-class belongs to the **SV** (Sure Validity) set.
   - an Eq-class is **potential** iff at least one of its SSD sis potential. This Eq-class belongs to the **PV** (Potetial Validity) set.

Step 3 : determination of the ISDs of interest.
The vector of views of a conjunction of views is a vector, named VECT-VIEW, where a truth value (T, F) has been assigned to the views that are named explicitly (the truth values of the other views are without interest). The truth value of a view that is explicitly named in a query is

1. **V**alid if the view is not negated,
2. **I**nvalid, if the view is negated.

Step 4 : building of the ISDs proper : These ISDs are obtained as the Cartesian product of the elements of $(SV \cup PV)$ (Step 2) and of the view vector (Step 3). They are called ISDs proper. Each *ISD proper* is assigned a truth value:

1. **V**alid, if its Eq-class belongs to **SV**,
2. **P**otential, if its Eq-class belongs to **PV**.

4.7.3 Examples of questioning queries

Object-building queries are not considered here.

Example 1
Consider the query :

```
Q5 : (PERSON | 12<Age<40)
```

The SSDs of the attribute Age are:

```
Age : d11 = [0, 18[, d12 = [18, 27], d13 = ]27, 65[, d14 = [65,140]
```

Let **E** be the set of valid Eq-classes of the p-type PERSON. The query Q5 divides **E** into three collections of Eq-classes :

$SV = \{(d_{12}, *, *, *, *)\}$: persons aged between 18 and 27 satisfy the query.
$PV = \{(d_{11}, *, *, *, *), (d_{13}, *, *, *, *)\}$: persons under 18 or older than 27 must be tested.

Note that the generalized Eq-class $\{(d14, *, *, *, *)\}$ constituted of persons older than 65 does not satisfy the query. The view vector of Q5 is $VECT - VIEW_{Q5} = (V, *, *, *, *, *, *, *, *)$. As the view of interest of Q5 is the minimal view, this view vector is without interest because all the objects of the p-type belong to the minimal view.
The *ISDs proper* of Q5 are:

1. ISDs proper with Sure Validity :
   $\{((V, *, *, *, *, *, *, *, *), (d12, *, *, *, *))\}$,

2. ISDs proper with Potential Validity:
$\{((V,*,*,*,*,*,*,*,*),(d11,*,*,*,*)); ((V,*,*,*,*,*,*,*,*),(d13,*,*,*,*))\}$.

The objects of the actual ISDs subsumed by the Valid ISDs proper belong to the answer without any individual checking, whereas those subsumed by Potential ISDs proper have to be tested individually.

Example 2 :
Consider the query :

```
Q5.1 : (MINOR | 12<Age<40)
```

The sets of Eq-classes **SV** and **PV** are identical to those of Q5. However, $VECT-VIEW_{Q5.1}$ is $(*,*,*,*,*,*,V,*,*)$, considering that MINOR est la 7^{th} view[19] among the 9 views of the p-type PERSON (figure 4.3). Consequently:

1. **V**alid ISDs proper:
$\{((*,*,*,*,*,*,V,*,*),(d_{14},*,*,*,*))\}$,

2. **P**otential ISDs proper:
$\{((*,*,*,*,*,*,V,*,*),(d_{11},*,*,*,*)); ((V,*,*,*,*,*,*,*,*),(d_{13},*,*,*,*))\}$.

No actual ISD can be subsumed by the potential ISD $((d_{13},*,*,*,*),(*,*,*,*,*,*,V,*,*))$, because MINOR is incompatible with d_{13}. Osiris does not need to make this information explicit as this ISD proper and all the actual ISDs which it subsumes are automatically excluded because they cannot be a valid ISD for any object of the P-type.

4.8 Query Optimization in Osiris

Query optimization consists in transforming a query into another equivalent query based on a narrower search space. Two queries are equivalent iff their evaluation leads to the same result, for any state of the database.

Semantic Query Optimization Query optimization is said to be semantic if the transformation of the query relies on semantic knowledge associated with the classes of the conceptual schema of the database. In a database, this knowledge is mainly expressed through the integrity constraints of the conceptual schema.

Semantic Query Optimization in Osiris In Osiris, a query is evaluated within a p-type. It can be assimilated to a dynamic view of the p-type and it is not classified explicitly with the other views of the p-type. However, it is rewritten in terms of the SSDs of the attributes it contains; this leads to situating in within the classification space in terms of valid, invalid and potential (generic) Eq-classes.

For example, let us consider the p-type PERSON and the query $(PERSON|age < 15)$. A classical DL-based query optimization approach would classify the query in the hierarchy of views, which would restrain the search of potential objects to the MINOR view.

In Osiris, the restriction of the search space is based on ISDs instead of views, which is more precise. As a consequence, when evaluating a query, our objective is to determine:

[19] The ordering of views is without interest in itself.

1. ISDs whose objects are certainly part of the answer
2. ISDs whose objects have to be examined individually
3. ISDs whose objects are certainly not part of the answer

4.9 Osiris and Data Integration

Some characteristics of the Osiris system had led the Osiris research group to tackle the integration of relational database in two one-year projects led by students in engineering [Gay, 1999] and [G.Laperrousaz, 2000]. These projects had a limited scope but they established some fundamental aspects of an integration system based on views and instance classification. We will briefly describe this work in the next subsection. The possibility to define several points of view over a given family of objects proved to be convenient to support the integration of data sources which themselves define different points of view over the same reality. The main characteristics of Osiris that were considered useful in an integration perspective are:

- The possibility to define a family of objects by the points of view,
- The possibility that an object satisfy a limited number of views at a given moment and can change these views
- Instance classification, which enables the system to identify the current views that an object satisfies and to reclassify it only when necessary,
- Semantic indexing of objects through their SSDs and their validity with respect to views,
- Semantic optimization of queries, through the semantic indexing of objects.

4.9.1 Mapping a Relational Source Schema into an Osiris target Schema

One of the fundamental needs of a data integration system is the mapping between source schemas and the global schema. In [Gay, 1999] a mapping technology was developed to translate a relational source schema into an Osiris target schema, to be used later in a data integration framework. The particularity of this approach was to do the mapping in two levels: mapping between the Object Identifier of the Osiris schema and the primary-key of the relational schema, and the mapping between the attributes of the two schemas. Below we describe the main modules of this work.

Correspondence Rules

Correspondence rules were designed to describe the relationships between the attributes of the schema of the target system and the corresponding attributes in the source bases. A correspondence between two attributes may be simple (synonymy, for example, PERSON.name = Base1.TEMPLOYEE.id) or complex, defined by a procedure. The main types of procedures that were considered are:

- Numerical operations (e.g., PERSON.income = SUM (Base1.TEMPLOYEE.salary x 12 + Base1.TEMPLOYE.bonus)
- Join between two tables

- An aggregative query (count, max, ...)
- The construction of a new object

The synonymy relationships establish correspondences between the terms of the global schema and the terms of the (local) schemas of the sources. These correspondences enable the system to build Osiris objects from the entities of the source databases.

Osiris Base Module

In the first experiments, the data items were maintained in the source databases. However, in order to take advantage of the instance classification mechanism offered by Osiris, i.e., consistency checking and the semantic optimization of queries, some pieces of information were materialized:

- The correspondence between the OID assigned by the Osiris system to a reconstructed object and the primary key of the corresponding entity in the source base,
- The ISDs that index the reconstructed objects,

Materializing these pieces of information implied the maintenance of these data. This was the perspective of this first experiment. A local ontology manager to manage the relationship between the terms of local bases and a query manager to querying data sources trough ISDs and using OID/Primary key correspondence was planned.

4.10 Conclusion

In this chapter we have made a global presentation of the P-type model and the Osiris system that implements it. We have described the p-type model, the data model of the system, which is an object model based on a hierarchy of views. We have focused on the object space classification and the object indexation of Osiris which is the main issue of Osiris used by IXIA and we have biefly reviewed its advantages, semantic query optimization. In the next section we will explain the details of the IXIA architecture and we will present the Osiris modules which we have used or extended in this architecture.

Chapter 5

IXIA Architecture

5.1 Introduction

We have presented the global architecture of our hybrid data integration approach in chapter 3. Here we describe it in more detail. We review its modules and then we explain the functionalities of the system. We give a comparison with other hybrid approaches and we end this chapter with a conclusion and perspectives.

5.2 Description of the Modules

Figure 5.1 shows a detailed architecture of IXIA. This prototype is an extension of the Osiris database system in order to develop a platform for data integration. Consequently, the architecture of IXIA consists of some modules of Osiris, some extended modules and some new modules. The following subsections describe these modules.

5.2.1 Osiris Global Schema

The integrated schema of IXIA is an Osiris schema. An Osiris schema as we discussed in chapter 4, is a collection of P-types in which each P-type is defined by a hierarchy of views.

5.2.2 Indexation Structure Module (ISD Space)

The Indexation Structure in IXIA is much similar to that of Osiris. The only difference is that we added the sources information to the indexation structure at the mediator level in order to make them unique. Adding sources information to the indexation structure can be done in different ways. In our implementation we add a source field to the object identifier (OID) in the indexation (ISD) module. Object identifiers at the wrappers level are identified through a simple Oid, which is unique in the wrapper of each source. We call it a Local Oid (Loid). The object identifier in the ISD Space is: *Goid = Loid + Source-ID* (Goid: Global Oid).

5.2.3 Wrappers

A wrapper in this architecture, like in all integration architectures, contains the mapping between a source schema and the integrated schema. However, in this approach wrappers contain the assignments between object identifiers (OID) in the global schema and the primary-keys in the data sources. In other words, the mapping in this approach, is based on the (Oid, primary-key)

Figure 5.1: IXIA Architecture

correspondence. A study for the mapping of a relational database schema to an Osiris target schema had been done in [Gay, 1999]. We have described this work briefly in chapter 4. In this study, after defining the identifiers assignments, each attribute of the target schema is mapped to one attribute or a function of several attributes of the source schema. In this work, the maintenance of the wrappers when adding, deleting or modifying an object was not considered. The Modification Detector of IXIA deals with this task (see section 5.3.2).

Since the mapping between the global schema and source schemas is made by a mapping between the identifiers, adding a new source is independent from the other sources and a novel source can be integrated provided that a new wrapper be developped. However some dependencies may exist between data of different local sources, the wrappers develop independently and the dependencies are managed by IXIA Query Evaluator when query processing.

IXIA can integrate the sources with other data models provided a wrapper for each source data model. Another study is ongoing to translate the XML source schema into an Osiris target schema [Ahmad *et al.*, 2009].

5.2.4 Modification Detector

The object indexation information must be refreshed periodically. In order to maintain the indexation data, the modifications of data sources has to be detected. A module called Modification Detector (MD) has been developed for this purpose. A MD module is developed for each source independently from other MDs and the detection process can be executed at different frequencies for different sources.

If the classifying attributes of a P-type of the global schema are in two different sources,

then to reclassify the objects of this P-type, the corresponding attributes of both sources must be considered. In such case, the MDs remain independent and this question will be dealt with in the "Source Modification Manager". To simplify and according to our motivation application (chapter 6), in the current implementation we consider the situation where all classifying attributes of an object are in a same source. In such case when a real world object is in two different sources, we will create two objects at the mediator level, each for one source, and by joint the source codes to Oids they will be distinguished (see chapter 6). Such implementation permits us to verify the consistency of data in different sources. Studying the data consistency verification can be cosidered as a future work.

In the current IXIA prototype, the MD module has been implemented on top of each wrapper and together they make a single component with two parts. Thus it can be used to add or delete the Oid/Primary-key mapping in the wrapper. We explain the details of the Modification Detectorand its implementation through the description of the indexation maintenance functionality, in section 5.3.2 where we also give the pseudo-code of this module.

5.2.5 Classification Server

The Classification Server of IXIA consists of "Osiris Classification Server" (OCS) and "Source Modification Manager" (SMM). It is responsible for object classification in the IXIA mediator. The SMM receives the modification of each source from the MD and prepares a message containing object insertions, object deletions or updated objects to the Osiris classification server. It also adds the source information to the message (it creates a Goid from the Loid received from the MD). The Osiris classification server receives the modification message and makes the classification updates. Figure 5.2 presents a pseudo-code for the Source Modification Manager.

5.2.6 Query Processor

Like all query processors in data integration systems, the IXIA query processor receives the user query, performs query decomposition / reformulation and then re-composes partial responses in order to send a response to the user. The IXIA query processor contains two main modules which we describe below: "Osiris Query Processor" and "Query Evaluator".

Osiris Query Processor

This module works as in a single Osiris database. For each partial query, received from the query evaluator, it searches in the ISD Space the objects' Oids which satisfy or potentially satisfy that query. However, in an Osiris database, after defining these Oids, the query processor does the verification of the complementary conditions and extracts the requested attributes. In IXIA, since data remain in the sources, this second part is not done by the Osiris Query Processor.

Query Evaluator

All the tasks which correspond to the data integration query processing are done by this module. It generates a query plan for the user query. Following this logical query plan, the query evaluator sends partial queries to the Osiris query processor and retrieves the partial response which are the Global Oids (Goids). It is also responsible for preparing the partial queries to the sources and combining the partial responses in order to provide a final response for the user. We describe the functionality of this module in section 5.3.3 where we review the query processing in IXIA.

```
01.  class SourceModificationManager
02.  {
03.    OSIRIScServer OSIRIS_cServer     // OSIRIS classification server identifier
04.    SourceModificationManager (OSIRIScServer t){     //constructor
05.        OSIRIS_cServer:=t
06.    }
07.    inserts (ModificationDetector MD, Wrapper wrapper)
08.    { //-- classification of last inserted
09.      Buffer tmp_inserts=new MD.last_inserts     //dump last inserts
10.      MD.I_dumped:=true
11.      while (tmp_inserts has more object) {
12.          extract a local object from tmp_inserts
13.          create a global object based on local object and wrapper
14.          OSIRIS_cServer.insert(global object)
15.      }
16.    }
17.    // ...  same implementation for "deletions" and "updates" methods considering:
18.    // ...  in line 13.  find a global object instead of create a global object
19.  } //SourceModificationManager
```

Figure 5.2: Java-like pseudocode of the source modification manager.

5.3 Functionalities

We continue to present the prototype of our approach by describing its three main functionalities.

5.3.1 Initialization Phase

When developing an integration application by using IXIA, the integrated schema is described by an Osiris conceptual schema. In the first step all the Eq-classes (i.e., collections of SSDs) will be extracted from the existing assertions in the P-types. The structure of indexation is defined by these Eq-classes together with the views of the P-type.

In the initialization phase, the classifying attributes for all the objects in the sources which correspond to the integrated schema are extracted and the objects are classified in the indexation table. The extraction of these data is made by the Modification Detector using the polling process (see section 5.3.2). In this case all extracted data will be considered as new insertions by the MD and will be sent to the "Source Modification Manager" (SMM), which in turn sends the insert messages to the "Osiris Classification Server".

5.3.2 Indexation Maintenance

The Indexation maintenance processing is done by an incremental update and when some data in one or several sources is modified, the object indexation update becomes necessary. Generally in materialized approaches, incremental update (compared with static update) is best applied in conditions where changes on the data are visibly smaller than the size of the data set in a certain period of time. Here, the data which we are interested in updating (classifying attributes) are significantly smaller than the size of data which participate in integration scenario.
The modification at the object level can be:

1. Delete an existing object

2. Add a new object

3. Modify the value of an attribute

Deleting an object from a source consists in deleting its corresponding Oid from the ISD space in the mediator. If this object is the only object in an ISD, this ISD (one line of ISD table) is also deleted.

When adding a new object or modifying an attribute value in the sources, a classification process may be necessary. It is important to note that all the new entries in a local source do not correspond to the creation of a novel object in the Osiris schema. For example in a relational data source, only the insertion or deletion of the tables whose primary-keys are mapped to an Oid of the global schema, imply creating a novel object or deleting an existing object. Thus only the classifying attributes modifications which cause a SSD change are important for the indexation system. On the other hand, all of the attribute modifications do not require an indexation modification (see chapter 4).

The most complex problem is to find an approach to distinguish the important modifications in local sources in an optimal way in time and feasibility. The most important challenge is that the sources are autonomous and that often there are no direct accesses to the structure of the sources to distinguish if there have been modifications (for example by a trigger). The Modification Detector has been developed for this purpose.

The Modification Detector (MD) over each source detects new objects, deleted objects and modifications of classifying attributes. This detection is made independently and in predefined periods for each source. A MD is linked to the classification server and is joined to the wrapper of its corresponding source. The data extracting process makes use of the mapping information in the wrappers. When detecting modifications, it informs the Source Modification Manager (SMM), which adds the source information and sends the corresponding message (insert, delete or change) to the Osiris Classification Server. If the modification is an "insert", a Local Oid (Loid) is assigned to the novel object at the wrapper level and is joint to the ID of the corresponding source, in order to create the Global Oid (Goid) and classify it in the ISD Space of the mediator. This last task is done by Osiris Classification Server.

We propose three detecting technologies for the Modification Detector. All or some of these technologies can be used in an integration application depending on the sources' characteristics and the application needs. Figures 5.3, 5.4 show the MD pseudo-code with these technologies.

Log System

Some data updating may be signalled in a log file. We can use this log information to update the indexation data if the log data is accessible for classifying attributes. In such cases the Modification Detector reads the important updates from this log system and sends it to Classification Server in predefined periods of time. As a consequence the rating of data refreshing is at most the rating of the log system updating.

Another log system which can be used for detecting important updates in the local sources is the transaction logs of transactional functionalities. Today, most DBMS provide the possibility of developing a database with transactional functionalities. In such a database, we can profit from the logging and recovery capabilities of the DBMS to monitor the changes in which we are interested. In this case the MD does this monitoring. This technique cannot be used for file

systems or semi-structured sources. It cannot be used either if the database is not implemented by transactional functionalities.

One of the drawbacks of this technology is that the transaction logs must remain available until the capture of the needed data has been done. If between two monitorings (two MD executions) the database administrator decides to trim the transaction logs, the data updating of this period will be lost.

Database Trigger

A database trigger is a stored procedure that is automatically executed in response to certain conditions or events on a particular table in a database. Triggers can restrict the access to specific data, perform logging, or audit data modifications. We can use triggers if we have access to the source databases. We write the trigger on the classifying attributes of such sources. Triggers immediately signal the modifications in the sources. Thus we can almost have access to an on-line indexation refreshing.

Polling

Log systems and triggers are almost inaccessible in the context of data integration. Having access to transaction logs or log files which signal all information which we need for indexation maintenance is not likely. In most data integration scenarios, the data sources already exist and we do not have access to them at the administrating level for adding the desirable triggers. In addition, triggers and log files (in some cases) are resource consuming while in certain applications we do not need to know the modifications immediately. In such cases, the all updates signalled between two MD executions are rejected.

For these reasons, polling the necessary attributes from data sources is our main strategy to develop the "modification detector" module. In this method, the images of classifying attributes are kept to distinguish the differences. In the predefined period of time, which can be different for each modification detector, the polling process is executed to extract the classifying attributes for all the objects which correspond to the global schema. The extracted data are compared with the last version saved in the modification detector and the differences are reported to the classification server.

5.3.3 Query Processing

Like in Osiris, the query processing in IXIA is made by using the indexation system. However, there are some important differences.

We recall that in Osiris, queries are translated into a set of SSDs and views to find the corresponding ISDs in the indexation system. Complementary verifications are needed to validate the potential objects response. We have described this process in detail in chapter 4. There are two important differences between query processing in a single Osiris database and in IXIA. The first difference is that in IXIA there is not any access to object attributes at the mediator level. Thus for all complementary verifications and to extract the requested attributes, data sources must be queried.

The second difference arises when the *condition* part of a query corresponds to more than one P-type. To process such query, it must be decomposed into single Osiris queries, each corresponding to a P-type. In an Osiris database, to answer a query, when attributes of several P-types are needed, Osiris searches the valid and potential response Oids for a first P-type. For potential Oids, it makes the necessary verifications and then extracts the requested attributes.

This is because often the value of some attributes of the first P-type may be needed for querying the second part of the query (which corresponds to the second P-type). In IXIA one does not have access to the value of attributes at the mediator level. Figure 5.5 presents our proposed pseudo-code[20] for this module. Below we present an informal description.

A user query in IXIA is an Osiris query. The general form of an Osiris query is:

```
(Context | conditions) [attributes]
```

Two scenarios are possible for this query:

1. Both *context* and *conditions* correspond to a single P-type of the global schema. The query is sent to the Osiris Query Processor in order to extract the response Oids.

2. The *conditions* of the user query corresponds to more than one P-type of the global schema (two P-types, for example). In this case, the user query must be decomposed into several Osiris queries (each corresponding to a P-type). This decomposition is necessary because when a query corresponds to more than one P-type, we may need the value of one or several attributes in one P-type in order to query other attributes in another P-type. In the context of data integration, attributes are in different sources. Thus, often a part of the query may need the result of another part.

For each Osiris partial query, the Osiris Query Processor (OQP) searches the valid and potential Oids, which are Global Oids (Goid). The Query Evaluator prepares the source partial queries and sends them to the wrappers to verify all the complementary conditions and extract attributes. In this process, before sending partial queries to the sources, Goids are decoded in order to obtain the Loids (Local Oids) and the corresponding sources. We note that some attributes found in the materialized part of the Modification Detector may be used in the query processing.

The partial responses will be re-composed by the IXIA query evaluator into an Osiris Response (a partial Osiris query). The final response for the user is prepared after receiving all the partial Osiris responses. In IXIA we then use the Osiris Query Processor only in single P-type query option.

The algorithm which must be followed by the IXIA query processor in order to make the decomposition, evaluation and re-composition of a user query is generated by the query evaluator using the sources information. It is memorized in the Query Plans module until the end of the processing of a user query.

In the next chapter we will give an example of this process. For more sophisticated queries which need calculations with response values (an average for example), a query interface must be provided.

5.4 Comparison with other approaches

In this section we give a comparison of our approach and other existing solutions for data integration, particularly other hybrid approaches.

[20] This module is not implemented in this work and to be able to make a test in our application, we have made a manual simulation.

5.4.1 Horizontal Hybrid Approach / Vertical Hybrid Approach

In all the hybrid data integration approaches that we have studied in this chapter, a horizontal hybrid method is used. This means that in the integrated schema some objects or relations are implemented virtually and others by a materialized method. All data manipulation for data in the virtual parts is made in a virtual manner and data of the materialized views are manipulated by materialized paradigms. A user query in such approaches may correspond to materialized or virtual parts or it can be decomposed between virtual and materialized parts. Consequently, the performance of the query processing as well as the data accessing delay are similar to those of fully materialized or fully virtual approaches respectively for materialized and virtual parts of the data integration system.

IXIA, the hybrid approach that we propose in this thesis, implements a vertical hybrid approach. It means that at the mediator level, some data of each object are materialized and others are virtual. The attributes of the objects remain in the local sources and generally data are extracted from the sources at query time. The object identifier of each object in an indexation structure is materialized together with the attributes which are needed for the refreshing of this indexation. We call this kind of partition of the integration system to the materialized and virtual parts a vertical partition, and we name our approach a vertical hybrid approach.

5.4.2 Advantages and Drawbacks

Query Optimization

The first advantage of IXIA is query optimization. IXIA takes advantage of the Osiris indexation system which is based on the instance classification mechanism using the views and the partition of the object space at the conceptual level. This partition makes use of constraints of the conceptual schema in the integration system.

Using views in the indexation system can reduce the search space when views are used in a query. Using the object space partition in the indexation system is beneficial when a query contains conditions corresponding to the global schema constraints.

This query processing is optimized at two levels : Mediator and Wrapper. At the mediator level, through the indexation system we are guided directly to the object Oids which satisfy (or potentially satisfy) a query, and their sources. There may be some complementary conditions which have to be verified in the sources. In other words, by using the indexation data in the mediator, the search space for a query reduces to a subset of both sources and objects. This query processing is also optimized at the wrapper level, because the wrappers contain the mapping between each object of the global schema and its attributes in the sources.
To extract the requested attributes from data sources, for each group of Oids which correspond to a local source, the IXIA query processor sends a query to the corresponding source.

Contrary to other hybrid approaches in IXIA, query optimization is not privileged for querying some materialized data but for all the queries of the integration system. In addition, if a query only uses classifying attributes, it obtains a higher level of optimization, thanks to the materialization of classifying attributes in the Modification Detector.

Tolerance wrt Source Characteristics and Flexibility in Materialization

The main motivation of hybrid data integration approaches is that in many integration scenarios, some queries are frequent and some data updates more frequent than others. A hybrid approach has to find an optimal solution for such situations by choosing the most adaptable data to be

materialized. There are two main restrictions for the hybrid approaches which we reviewed in this chapter:

1. Generally in such approaches, all materialized data have the same conditions. Data refreshing for all materialized data is done in the same way and with the same delay. However, this disadvantage can be partially overcome by using different incremental update technologies.

2. One cannot change the decision which is made for materialized and virtual data when developing the integration system. For materialized data, the refreshing process is done as defined at the system development phase and one cannot change this condition for some of these data. In the same way, virtual data always remain in the sources and querying it is time-consuming. Changing these conditions results in changes of the global schema.

IXIA proposes a data refreshing solution which overcomes these two disadvantages. The modification Detector is the core of this solution. It offers two advantages:

1. Data refreshing for different sources, hence for different data, can be done in different periods of time [21]. This flexibility in IXIA permits us to consider the characteristics of different sources in each data integration application. Four important issues must be considered for each source to make decisions for MD arrangement:

   - The rate of data updating
   - The importance of the data
   - The querying frequency
   - The query power
   - The availability of the source

2. The arrangement of refreshing for different data sources can be changed by changing the frequencies of different MDs, which is a decision of the system administrator.

Another advantage of IXIA is that adding a novel source to the system is independent from other sources. Because of its special mapping system (object-to-object), when defining a wrapper for a novel source, it can be integrated to the system. The novel source information, however, must be added to the rule base of the query evaluator and the Source Modification Manager.

There are some restrictions and difficulties in the IXIA architecture. In the wrappers we need two levels of mapping: schema mapping and (Oid, primary-key) mapping. This second kind needs to be updated each time an object is deleted or inserted in a source. When an object is inserted an Oid must be created and assigned to the primary-key. However, this process can also be done by the Modification Detector in order to be optimized.

Finally, it must be considered that like all the approaches which use some materialization in their architecture, we often do not have access to on-line data and generally there is a slight delay in receiving updated data. Therefore, this approach cannot be applied to applications in which on-line querying is crucial. IXIA cannot either be used if there are sources which do not give access to the primary-key of data.

We review below the problems which arise when a modification occurs in a source but because of the updating delay the IXIA indexation system does not know it at query time. Three cases must be considered:

[21] We note that in most hybrid scenarios for data integration, data of some sources have conditions adapted to be materialized and others to be virtual. Considering the travel agency example; data of geography sources never change, hotel data sources do not change frequently (an exception is the reservation system), and the weather data change frequently (see [McHugh *et al.*, 1997]).

1. An object is deleted in the source but it participates in the response Oids. Query evaluation sends a query to retrieve attributes but the response is *unknown*.

2. An object which is a response to the query is inserted in a source but the indexation system has not yet been informed. In this case we will lose this response.

3. An object is updated in the source so that it must participate in the response of a query but the indexation system does not know it yet . In this case we will also lose this response.

4. An object is changed in the source so that it is no longer a response object for a query but the indexation system considers it in the responses. In this case, we have a wrong response. However, in certain cases this response can be omitted from the final response by complementary verifications. It is the most important challenge that exists in this approach, as well as in all materialized or hybrid approaches.

In order to reduce these potential problems (particularly the last one), we believe that in many cases, we can profit from the tolerance of MD modules for querying the updates of critical attributes in an acceptable delay.

5.5 Conclusion

In this chapter we have presented the IXIA prototype and we have made a comparison with other data integration approaches, notably other hybrid approaches.

As shown in the last sections, the IXIA data integration system offers a significant degree of query optimization thanks to its indexation system which profits from the indexation system of Osiris. Query evaluation in our data integration approach relies on the native object indexation system. As the indexing process and its refreshing are not made at query time, query processing benefits from the indexing structures previously built.

The modification detector is used for indexation update processing, but its particular implementation leads to data refreshing for each source autonomously and at different frequencies. This characteristic offers a tolerance that makes possible the management of data refreshing for each source separately. This is done according to the update frequency of each data source, the importance of the freshness of its data, as well as its other characteristics such as query power. By increasing the execution frequency of the MD or by using the trigger or transaction log system, we are able to achieve on-line data integration in certain cases.

In comparison with a fully materialized data integration approach, we do not need to perform a full data migration and the real data remain in the local sources. However, the query response time in a fully materialized approach remains faster. In our approach the query response time is faster than that of a fully virtual approach, but in certain situations, access to freshly updated data can be delayed.

Finally, compared with other hybrid approaches, it offers a query optimization for all the queries of the system and a flexible and manageable refreshing method for materialized parts.

```
01.  class ModificationDetector
02.  {
03.    Buffer last_inserts     //last inserts after last classification
04.    Buffer last_deletions   //like last_inserts for deletions
05.    Buffer last_updates     //like last_inserts for updates
06.    Buffer new_inserts     //last inserts after last detection
07.    Buffer new_deletions   //like new_inserts for deletions
08.    Buffer new_updates     //like new_inserts for updates
09.    Buffer data_source     //data on the related source
10.    Buffer last_updated     //replica of data_source updated in last detection
11.    Wrapper wrapper     //wrapper identifier
12.    SourceModificationManager SM_manager     //modification manager identifier
13.    boolean I_dumped,D_dumped,U_dumped   //dumped flags
14.    Type D_type     //detection technique = polling or trigger or logging
15.    Timer D_timer= new Timer()     //activation timer for detection
16.    {
17.      OnTime()         //-- kernel of ModificationDetector
18.      {
19.          if (I_dumped)
20.              purge last_inserts
21.          // same operation on last_deletions, last_updates respectively
22.          if (D_type=polling)     //-- polling case
23.          {
24.              // do based on comparison of data_source and last_updated
25.              insert new objects into new_inserts
26.              insert deleted objects into new_deletions
27.              insert modified objects into new_updates
28.          }
29.          insert objects of new_inserts into last_inserts
30.          delete objects of new_deletions from last_inserts where exist
31.          insert objects of new_deletions into last_deletions
32.          insert objects of new_updates into last_updates where not exist
33.          update objects of last_updates to new_updates where exist
34.          delete objects of new_deletions from new_updates where exist
35.          if (I_dumped) and (last_inserts is not empty){
36.              SM_manager.inserts(this ModificationDetector, wrapper)
37.              I_dumped=false
38.          }
39.          // same operation on last_deletions, last_updates respectively
40.          insert objects of new_inserts into last_updated
41.          delete objects of new_deletions from last_updated
42.          update objects of last_updated to new_updates
43.          purge new_inserts, new_deletions, new_updates
44.      }//kernel of ModificationDetector
45.    } //timer
```

Figure 5.3: Java-like pseudocode of modification detector, part-I.

```
46.   ModificationDetector (Type t, int dur, SourceModificationManager mm,
47.                          Wrapper wr) //constructor
48.   { D_type:=t , SM_manager:=mm, wrapper:=wr
49.     I_dumped:=D_dumped:=U_dumped:=true
50.     if (D_type=trigger)     //-- database trigger case
51.         create trigger MD on data_source
52.             after insert, delete, update
53.         {
54.             if (insert){
55.                 insert new objects into new_inserts
56.             }
57.             if (delete){
58.                 insert deleted objects into new_deletions
59.                 delete deleted objects from new_inserts where exist
60.                 delete deleted objects from new_updates where exist
61.             }
62.             if (update is on classifing attributes){
63.                 insert modified objects into new_updates
64.                 update objects of new_updates to modified objects where exist
65.             }
66.         }//trigger
67.     if (D_type=logging)     //-- "transaction log" or "log file" case
68.         create EventHandler MD on log of data_source
69.         {
70.             if (commited log signal is insert){
71.                 insert new objects into new_inserts
72.             }
73.             if (commited log signal is delete){
74.                 insert deleted objects into new_deletions
75.                 delete deleted objects from new_inserts where exist
76.                 delete deleted objects from new_updates where exist
77.             }
78.             if (commited log signal is update on classifing attributes){
79.                 insert modified objects into new_updates
80.                 update objects of new_updates to modified objects where exist
81.             }
82.         }//logging
83.     D_timer.interval:=dur , D_timer.enable:=true
84.   }//constructor
85. }//ModificationDetector
```

Figure 5.4: Java-like pseudocode of modification detector, part-II.

```
01.  class IXIAqueryEvaluator
02.  { //undescribed data types in details
03.    OSIRISquery   // OSIRIS query requesting Global Oids
04.    UserQuery   // OSIRIS-like IXIA user query
05.    UserResponse   // OSIRIS-like IXIA user response
06.    SourceQuery   // query to the wrapper of a source requesting Local Oids
07.    SourceResponse   // response of the wrapper of a source
08.    ...
09.    OSIRISqProcessor OSIRIS_qProcessor     // OSIRIS query processor
10.    Wrapper[] wrapper  // array of wrappers' identifier
11.    int SourceCount  // number of sources
13.    //----- IXIA query processing for a single OSIRIS query
12.    SourceResponse[] execQuery(OSIRISquery O_q)
13.    {
14.          SourceQuery S_q= new generateSourceQuery(O_q)
15.          Buffer G_Oids= new OSIRIS_qProcessor.process(O_q)
16.          SourceResponse[SourceCount] S_r
17.          for j:= 1 to SourceCount{
18.            Buffer L_Oids= new extractSourceOid(G_Oids, j)
19.            S_r[j]= new wrapper[j].process(S_q, L_Oids)
20.          }
21.          return S_r
22.    }
23.    IXIAqueryEvaluator()     //-- kernel of IXIA query evaluator
24.    {
25.      while (1) {
26.        UserQuery U_query= new getUserQuery()
27.        if (U_query is on single P-type){
28.          SourceResponse[] S_response= new execQuery(U_query)
29.          putUserResponse(U_query.generateUserResponses(S_response))
30.        }
31.        else {
32.          QueryPlan Q_plan= new generateQueryPlan(U_query)
33.          while (Q_plan.hasMoreQuery()) {
34.            SourceResponse[] S_response= new execQuery(Q_plan.nextQuery())
35.            Q_plan.re_evaluate(S_response)
36.          }
37.          putUserResponse(Q_plan.generateUserResponses())
38.        }
39.      }
40.    }
41.  } //...
```

Figure 5.5: Java-like pseudocode of IXIA query evaluator.

Chapter 6

Data Integration Application: Telecommunication Network

6.1 Introduction to Telecommunication Networks

PSTN, the public switched telephone network, is a circuit-switched network that is used primarily for voice communications world wide, with more than 800 million subscribers.

Originally PSTN was a network of analogous fixed-line telephone systems. Today however, PSTN is almost entirely digital and includes mobiles as well as fixed telephones.

The first networks were created using analog voice connections through manual switchboards. Later, automated telephone exchanges replaced most switchboards and today with digital switch technology, most switches use digital circuits for exchanges between them but analog two-wire circuits are still used for connections between telephones and switches.

The subscribers use the services by connecting to one predefined PSTN switch. Each switch is directly connected to a limited number of subscribers. In order to organize automated operator dialing, and later direct distance dialing, the telecommunication companies organize the various switches in their networks in a hierarchical structure (see Figure 6.1).

6.1.1 The PSTN switch

The PSTN basic services refer to the basic call connection functions provided by the switch. These call functions include intra-office call, local call, national toll call, international toll call, transit call and others. These services have to be reliable and real-time. It is the reason why the PSTN switches are modular or more precisely, distributed systems. A central system can just support the real-time operations needed for a limited number of subscribers (up to 10k). The related hardware system is a distributed control system and its software system is also a distributed software system. The modular structure of a switch makes it flexible enough for networking for all kinds of telecommunication companies demands.

Database Structure A switch has a modular structure. The whole switching system comprises one administration module and multiple switching modules. The database of switches is a distributed database. The "distributed" characteristic indicates that each switch module stores its local data and the partial data corresponding to the rest of the switch. The data in one module is managed by its subsystem. For all switches of a specific switch vendor, the subsystems of modules are coordinated and consistent with each other, but the switches of different vendors

Figure 6.1: Public Switched Telephone Network

can be heterogeneous in format and software system. Physically, the subsystems of the database in the respective modules are separated, constituting one part of the whole database system. Logically, these subsystems constitute a single entity which forms the management system of the distributed database. The coordination among multiple local databases realizes the database function of a switch. Figure 6.2 shows the database structure of respective modules.

As the modular distribution control mode is applied in the switches, each module is assigned to the corresponding data, depending on its function requirements. Therefore, the database of a switch is the data collection of all the modules.

The database management system is responsible for managing the entire data of the switching system (including the configuration data, subscriber data, office data and charging data) and implementing tasks such as data access organization, data maintenance, data updating, data backup and data recovery. The Application Program Interface (API) is the interface between the switch database and switch applications such as Operation and Maintenance, Monitoring and Charging.

The PSTN switches produced by different vendors are heterogeneous in hardware and software. It has become urgent to integrate PSTN switches in order to reduce costs and improve transparency and manageability of all whose objective is to integrate the switch database systems, and we review the most important challenges that we aim to overcome.

Figure 6.2: Typical database structure of a PSTN switch

6.2 Operation and Maintenance Center

Qazvin Telecommunications Company (QTC) is an Iranian regional telecommunications company covering ten cities. Each city contains several (up to 6) PSTN switches depending on the number of subscribers. The total capacity of all switches installed in Qazvin province is about 430.000 subscribers.

Along with the formation of competitive pattern, in the telecom field, the problem of how to reduce operation and maintenance costs arises for telecom carriers. Besides, the network architecture has become more and more complicated due to the gradual expansion of the network scale. To reduce the costs, an effective way at present is to rebuild the network, reduce the network layers and use a unified network management system. For this reason, carriers require the large capacity, fewer end offices and wide coverage, in order to simplify the network management structure.

Qazvin OMC project[22] aims to create an operation and maintenance center in the Qazvin province so that QTC employees can operate and maintain all the PSTN switches. A web-based platform is expected to result from this project. The platform has to connect remotely to the switches, manage them and access their data with the best possible delay. The user interfaces are expected to be uniform for all switches.

This platform needs to integrate the autonomous switch systems containing three functional subsystems: Switch Monitoring, Subscribers Operating, and Charging. Switch Monitoring is the responsible of signaling and maintenance of the system. It does not participate in the integration application of this thesis. The rest of this section describes the basic functions of two last switches subsystems exploring the expected features in the final platform.

[22]Qazvin OMC project was founded by Qazvin Telecommunication Company and IK International University, 2007-2009.

6.2.1 Subscribers Operating Subsystem

The subscribers operating subsystem manages and maintains data needed for the switch functioning which is related to its subscribers. As shown in figure 6.3, this subsystem is responsible for the authentication and authorization of subscribers. It also processes the query request of complementary switch functions. "switch complementary functions" are the supplementary favor that a switch can do for telecom carriers e.g. prepayment service.

Figure 6.3: Typical operating subsystem interaction with switch DBMS

6.2.2 Subscribers Charging Subsystem

The telecom service users should pay the telecom service operator for the use of the service, such as making telephone calls. The fee per call (or the details of bill information) is related to the call distance (toll call and local call are charged differently), call duration and conversation date and time (the charges are different for normal workdays, for holidays, and in different time segments of the same day). Different charging standards apply to different telecom services and one telecom service may be charged differently in different areas. Usually, the telecom carrier stipulates the charging tariffs according to local conditions.

The charging subsystem consists of modules charging subsystems and switch charging subsystems, as illustrated in figure 6.4. The module charging subsystem collects and generates

billing information and sends it to the switch charging subsystem. The switch charging subsystem receives the bill information, generates the background charging bill and dumps the bill information.

Figure 6.4: Typical charging subsystem interaction with switch DBMS

Dumping bill information is to take bills out of the modules and save them on a permanent magnetic medium for use in the offline charging system. Since the modules bill pool is of limited capacity, the bills need to be dumped in due course. This function is performed by the switch charging subsystem. It sends 'fetch bill' commands to the modules at specified intervals. The modules with bills respond to the command and send bills to it.

The volume of bill information is too big to permanently remain in the switch database. So the received bills are saved into 'bill files' and then they are used for offline billing. Usual billing service is offline and limited to specified periods such as one or two months.

Along with the problem of unpaid bills and the request of subscribers to view the details of their bill information easily, the telecom operators need online billing service. The prepaid service, highly demanded by subscribers and telecom operators, is based on hot billing service.

The main motivation of the Qazvin OMC project was to provide a unique hot billing center in Qazvin province where heterogeneity is hidden from the users.

6.2.3 Operation and Maintenance Center Platform

Figure 6.5 demonstrates the designed platform in the Qazvin OMC project. It aims at reducing communications and remote operations on the switches. For this purpose the OMC database keeps out the extracted information which is valid and does not need to be refreshed. This

historical data will be integrated into a materialized system (data warehouse).

In this thesis we are interested in integrating non-historical data which will be queried directly from switches. In the next section, we explore the data integration challenges and present more details about the final architectures.

Figure 6.5: Designed Platform of Qazvin OMC Project

6.2.4 Integration Challenges and Solutions

The autonomy and the heterogeneity of switches software systems are the main challenge facing the need to provide a data integration system. In one region, the PSTN switches of different switch producers may be used. Each switch producer has its own information technology for the switches of the same family. Only in Qazvin city, for example, the six switches in use are from five different producers: Huawei, ZTE, EWSD, Neax and Kiatel. For autonomy reasons we cannot modify software modules. The Qazvin OMC Project considers three different integration levels :

- **Data Integration:** uniform access to information needs to integrate the data from all switches. A subscriber database which contains all information about subscribers such as their personal information, their telephone line information, their payment conditions,

etc, also participates in the integration system. Such a data integration system provides the possibility of answering many queries useful for telecommunication companies. We believe that with our implementation, telecom carrier can also benefits from an important degree of integrity between administrating data and data in the switches. In addition, data integration enables us to support new applications such as decision support system. We develop an example of this level of integration in this thesis.

- **Shared interface:** A unique interface explores the desired common services of switches. However there are some critical services such as monitoring (for system errors), in which we require all detailed data. For example, the alarms contain common information (such as AlarmType, AlarmLevel, Date, Time, RecoveryState) and vendor specific details, which are indispensable for maintenance and must be considered in the shared interface.

- **Application integration:** In order to develop a complete integration system, we also need to integrate the switch functionalities. The services must be integrated to create new services. Two main objectives are considered for this part:

  - Monitoring of the common channels of a telecommunication company network is based on the alarm services of all switches.
  - Subscribers' operating commands on all dedicated local loops of a client (such as a distributed company containing several agencies) need to be transformed to the commands on the related switches. To realize this application integration for subscribers data, we use the data integration system.

6.3 Data Integration Application

Figure 6.6 makes a global presentation of the telecom data integration system. It consists of a Data Warehouse for the historical data and a mediator system. Three groups of the data are saved in the DW. These are the data that change never:

1. In the billing system, the call details data are extracted from the switch databases and saved in the DW. These data are extracted at the end of each billing period. Switch inserts a ligne in the log file when it dumps these data from its modules to its database.

2. At the end of each billing period (e.g. at the end of each month) the bill data are saved in the DW.

3. Once an alarm is recovered it is saved in the DW.

Other queries which we want to respond by the integration system, are requested using a mediator system. In this integration scenario, different switches have different delays for sending data from their modules to their databases. In addition we have not access to their database source and querying them via specific APIs is the only way to access their data. Subscriber database, however, is developed in the telecom company and access to its database is possible. In the other hand, the query response time to extract switch data is crucial for telecom carrier as well as for subscriber that use telecom Web site. We believe that IXIA integration approach is compatible with the needs and the characteristics of this system. In the next subsections we review the objects and the functionalities needed in the integration system at the mediator level.

Figure 6.6: Telecommunication integration system.

6.3.1 Data Structure

We define two objects at the mediator level: Subscriber and Call Number.

Subscriber Each subscriber of the telecomunication carrier is defined by an unrepeatable code. We define three types of subscribers. Each type is handled differently by the telecom integration system. The conditions for each subscriber type are defined by the telecom administration. They can be changed by the telecom administrator and it is possible that some changes imply a schema change at the integration level. Below we describe the different subscriber types:

- **Personal:** A personal subscriber has one single line. One real person can correspond to several personal subscriber objects at the same time.
- **Public:** A public subscriber is a public center such as a university, a hospital, a police station, etc. Several call numbers can be assigned to a public subscriber. A public subscriber can have several physical subsidiaries with different addresses and it can have call numbers on different switches.
- **Commercial:** A commercial subscriber is a private or public commercial center. Like a public subscriber, it can have several call numbers on different switches.

Call number Consequently we have personal, public and commercial call numbers and each of them is handled differently by the system. We have also defined another call number named

"critical call number". In a police or fire station, for example, some call numbers are for emergency situations. A hospital or a university must have a minimum number of active lines even if the bill has not been paid.

- **Critical call number:** Some of the call numbers in a public or commercial center can be defined as "critical call numbers" which have special conditions. For such call numbers, the national and mobile call-in is always authorized. Other conditions are explained in the assertions part.

We note that for confidentiality reasons we do not have access to the subscriber data of Qazvin telecommunication. In order to make a test for our application in this thesis, we develop a test database with a relational schema and we enter some test values in this database. Figure 6.7 shows the relational schema of this database and figure 6.8 shows the relational schema of a switch database.

Figure 6.7: Subscriber Database.

6.3.2 The constraint procedures of telecommunication system

In this section we explain the main functionalities of telecommunication subscriber system in order to describe the needs of the telecom integration system. Through these functionalities, we define the integrity constraints which can be applied at the mediator level. The assertions of the views in the global schema will be extracted from these constraints. We recall that the functionalities which we define here must be applied to the switches via an application integration system.

Changing the authorizations for a call number

When a telephone line (call number) is allocated, it has all the authorizations. The telecom carrier may wish to limit or lock some authorizations after a predefined charging. The authorizations of a call-number consist of several fields. In this implementation we use only three call-in and three call-out authorizations. The limiting or locking conditions for an authorization must tolerate different types of subscribers and call numbers.

Charge and authorization attributes are defined in order to imply the conditional procedures on the call numbers:

- Charge attributes: they correspond to different charges of a call number from the last payment.

```
National-Charges, International-Charges and Mobile-Charges
```

- Authorization attributes:

```
Authorization-National-Call-out = {"yes", "no"}
Authorization-International-Call-out = {"yes", "no"}
Authorization-Mobile-Call-out = {"yes", "no"}
Authorization-National-Call-in = {"yes", "no"}
Authorization-International-Call-in = {"yes", "no"}
Authorization-Mobile-Call-in = {"yes", "no"}
```

Depending on the kind of call number kind(personal, public, commercial, critical number) we define different functions to limit its authorizations if the corresponding subscriber goes past his authorized charges. We consider the followed conditional values:

```
x1 and x2 for National-Charge
y1 and y2 for International-Charge
z1 and z2 for Mobile-Charge
```

- For a personal call number:

```
If National-Charge > x1 then Send a warning to subscriber;
If National-Charge > x2 then Authorization-National-Call-out := "no";
If International-Charge > y1 then
            Authorization-International-Call-out := "no";
If Mobile-Charge > z1 then Authorization-Mobile-Call-out := "no";
```

- For a public call number:

```
If National-Charge > x2 then Send a warning  to subscriber;
If International-Charge > y1 then
            Authorization-International-Call-out := "no";
If Mobile-Charge > z1 then Send warning to subscriber;
If Mobile-Charge > z2 then Authorization-Mobile-Call-out := "no";
```

- For a commercial call number:

```
If National-Charge > x2 then Send a warning to subscriber;
If International-Charge > y1 then Send a warning  to subscriber;
If International-Charge > y2 then
            Authorization-International-Call-out := "no";
If Mobile-Charge > z1 then Send a warning to subscriber;
If Mobile-Charge > z2 then Authorization-Mobile-Call-out := "no";
```

Inactivating a call-number

Another administrating function that a telecommunication system has to support is to inactivate one or several call-numbers of a subscriber if he goes past the paying delay of a bill. Inactivating a call number is a function of the bill paying delay and the payment history of its subscriber which indicates if he is a faithful client or not. As in the last section, this decision also depends on the type of subscriber and call number. We define the "*Unpaid-Warning*" and "*Faithfulness*" attributes for this purpose. For each subscriber:

- *Unpaid-Warning = {0, 1, 2}*
  For each bill, the telecom carrier defines two delay dates. If a subscriber does not pay the total amount of the bill after the first delay date, "*Unpaid-Warning*" will be set to 1 and if he goes past the second delay date, "*Unpaid-Warning*" will be set to 2.

- *Unfaithfulness = {0, 1}*
  This attribute evaluates the faithfulness of the subscriber that depends on his payment history.

We also define for each call-number the "*Inactivate-Alert*= {0, 1, 2}" attribute. This attribute takes a value depending on the kind and the faithfulness of its owner when he has a first or second unpaid-warning. By default, when there is no paying delay, it is set to "0", other conditions are as follows:

```
If  Unpaid-Warning = 1 and Unfaithfulness = 0 then
    For all lines (call-numbers) of the subscriber
                  Call-Number.Inactivate-Alert := 1;
If  Unpaid-Warning = 1 and Unfaithfulness = 1 then
    For all lines (call-numbers) of the subscriber
                  Own-Call-Number.Inactivate-Alert := 2;
```

For a personal call number:

```
If  Inactivate-Alert = 1 then
    Authorization-National Call-out := "no"
    Authorization-international call-out := "no"
    Authorization-mobile call-out := "no"
    Authorization-International Call-in := "no"
    Authorization-Mobile Call-in := "no"
If  Inactivate-alert = 2 then
    All call-in and call-out authorizations are set to "no";
```

For a public and commercial call number: The same procedure is done for their normal call numbers. However their critical call numbers are processed differently. We discuss this last option below.

For a critical number:

```
All call-in authorizations = "yes".
Authorization-National-Call-Out = "yes"
If Inactivate-Alert = 2 then
    Send a warning to subscriber
    Authorization-International Call-Out := "no"
If Inactivate-Alert = 3 then
    Send a warning to subscriber
    Authorization-Mobile Call-Out := "no"
```

Figure 6.8: A PSTN Switch Database

6.3.3 Requested Queries through the Integrated Schema

In this section we explain different categories of queries that we wish to respond to through the integrated view. For each category we give some examples.

1. Querying the number or other information of some subscribers.
   - Querying the address of all unfaithful personal subscribers.
   - Querying the subscribers that do not pay their bills after the expiration date.
2. Querying the bill value for one or several subscribers from the switches:
   - A simple query to ask the bill value of subscriber A.
   - Querying for a group of subscribers. For example:
     - The bill value of subscribers whose "*international-Charges*" overtakes X.
   - Querying the sum of bills for a public or business subscriber that has several subsidiaries which can be linked to different switches.
   - Querying if the sum of bills for a public or business subscriber or its "*mobile-charges*" overtakes Y.
3. Querying the authorization fields or other attributes of the subscribers' call-numbers which are found in the subscriber database as well as in the switches.
   - Querying the authorization fields for a call number.
   - Querying all the subscribers non-authorized for international calls.

There are many other queries which can be requested by combining the above queries. For example, one can request the bill data for some personal subscribers where the call-out authorization of its call-number is "*no*".

6.3.4 Data Integration Application; Implementation

In this section we present the global schema of the mediator, we define the P-types of the global schema. Using the assertions of the P-type views, we define the structure of the indexation system. We then present the structure of the Modification Detector for this application and finally we detail the the query processing for two sample queries. [23]

The Global Schema

To satisfy the needs of the integration application that we described in the previous section, we define a global schema with two P-types. Figures 6.9 and 6.10 show the views of the global schema and figures 6.11, 6.12 and 6.13 shows the definitions of the SUBSCRIBER and CALL-NUMBER P-types of the global schema. This schema contains the constraints which we described in section 6.3.2. Therefore, these constraints participate in object classification and in the indexation system of the mediator. So through the global schema, we can respond to the three categories of queries which have been described in section 6.3.3.

[23] We note that certain parts of Osiris are not still applicable and we have made a simulation by the relational model for these parts. A study to develop Osiris Classification Server is in the final phase in the Osiris equip [Ana and Samer 2009].

Figure 6.9: Hierarchy of Views for SUBSCRIBER P-type.

Stable Sub-Domain (SSD) of the global schema

Through the use of all the constraints of the P-types, we extract the Stable Sub Domains of the P-Types. We list here the SSDs of all the attributes.

- For SUBSCRIBER P-type:

```
Subscriber-Type: d11 = {"Personal"}, d12 = {"Public"}
                        , d13 = {"Commercial"}
Unfaithfulness :  d21 = {"yes"}, d22 = {"no"}
Current-Bill-Paid: d31 = {"yes"}, d32 = {"no"}
Unpaid-Warning: d31 = {0}, d32 = {1}, d33 = {2}
```

- For the CALL-NUMBER P-type:

```
Active-Line: d11 = {"yes"}, d12 = {"no"}
National-Charges:    d21 = [0, A1], d22 = ]A1, A2]
                        , d23 = ]A2 (or > A2)
International-Charges:   d31 = ]0, B1], d32 = ]B1, B2]
                        , d33 = ]B2 (or >B2 )
Mobile-Charge: d41 = ]0, C1], d42 = ]C1, C2], d43 = ]C2 (or >C2)
National-Warning: d51 = {"yes"}, d52 = {"no"}
International-Warning: d61 = {"yes"}, d62 = {"no"}
Mobile-Warning: d71 = {"yes"}, d72 = {"no"}
Authorization-National-Call-out: d81 = {"yes"}, d82 = {"no"}
Authorization-International-Call-out:  d91 = {"yes"}, d92 = {"no"}
Authorization-Mobile-Call-out: d101 = {"yes"}, d102 = {"no"}
Authorization-National-Call-in: d111 = {"yes"}, d112 = {"no"}
Authorization-International-Call-in: d121 = {"yes"}, d122 = {"no"}
Authorization-Mobile-Call-in: d131 = {"yes"}, d132 = {"no"}
Inactivate-Alert: d141 = {0}, d142 = {1}, d143 = {2}
```

Figure 6.10: Hierarchy of Views for Call-Number P-type.

Eq-Class

Now we can define the Eq-classes corresponding to the Stable SubDomains. As discussed in chapter 4, each Eq-Class consists of a set of SSDs of a P-type.

Indexation Structure(ISD Space)

The structure of ISD space is shown in figure 6.14 by a relational schema. In this application, we have an ISD table for each P-type. Each ISD consists of a vector for an Eq-class, a vector for the views of the P-type and a pointer to the list of the Goids which are classified in this ISD. To add the source information in this implementation, we have joined to each Oid (Loid) a field containing its corresponding source. In this application there is no redundancy for the subscriber objects. All subscriber objects are only in the subscriber database. However, each call-number is in two different sources: the subscriber database and one of the switches. Each of these two sources has some of the attributes of a Call-Number object of the global schema, but all the classifying attributes which we use in our implementation are in both of them. We note that we have implemented a light version of this integration application in which charges attributes do not participate in the object classification. [24] With our implementation, at the mediator level (in the ISD space) there are two Goid for each Call-Number, each corresponding to one source. If these two objects are consistent, these two Goid must be classified in one single ISD.

Wrapper

Wrappers are developed as discussed in chapter 6. One important point is that in the switches wrapper we extract the values of warning attributes as a function of the charge values. In other words, National-Warning, International-Warning and Mobile-Warning for the Call-Number objects of the switches databases are extracted from National-Charge, International-Charge and Mobile-Charge in theses databases.

[24] As a perspective of this work we will propose the solutions for the cases in which the classifying attributes are distributed between two local sources.

Modification Detector

In this application, to detect updates in the switches, we use the polling technology. We do not have any access to the switch databases to define a trigger. In addition, although some switch databases are implemented by the SQL-Server DBMS, we do not use its transaction possibility. They have log files which inform the user when the bill information is fetched or when an error alert has occurred, but such information is not the one which we need for indexation maintenance. For the subscriber database, we have developed triggers. The MDs have been implemented as discussed in chapter 5.

The switches that we integrate in this application are a Huawei and a ZTE switch. A Huawei switch guarantees that the data is refreshed every 5 minutes and the ZTE every 30 minutes. Therefore we set the timer of MD for Huawei to 5 minutes and the timer of ZTE MD to 30 minutes. We can also set both of them to 30 minutes if we want to have the same delay for the data of all the switches. Because we use the trigger technology for its MD, the detection of the modification of the subscriber database is on-line.

6.3.5 Query Processing

Two factors play an important role in the query response time of an IXIA integration application.

The first factor is the Modification Detector delay. In the present application, we have developed triggers for the subscriber database and we have used the polling technologies for two PSTN switches. For the queries which are posed directly from Call-Number in one switch, the response are not on-line. In other words, the integration system can guarantee the validity of the response with a delay D, which is the frequency of the corresponding MD (in this example 5 minute for Huawei and 30 minutes for ZTE).

The detection of the modification of subscriber database is on-line. Consequently query processing for queries in which *context* and *conditions* are over the subscriber database is similar to that of a fully virtual approach. For example in this query,

```
(PERSONAL-SUBSCRIBER | Unpaid-Warning = 2) [Own-Call-Number.bill]
```

we search the response objects through the classifying attributes of the subscriber database. Therefore the responses Oids are up-to-date. The bill attribute will be requested directly from the corresponding switch and then the response of the query is an on-line response (there is no update delay in this case).

The Second factor is the type of OID. If we use a semantic OID in this application we can achieve a same Loid for the same real world object which exists in two different sources. For example, in this application, for two Call-Number in the subscriber database and a switch which corresponds to the same phone line, Loid could be a function of its call number. At the mediator level we will have two Goid by joining the source codes to Loid. This semantic definition of Oid permits us to obtain the Call-Number Oids in a switch directly from the subscriber database by adding the corresponding switch code to the Loid of the Call-Number in the subscriber wrapper. A significant query optimization will be obtained by this method. Semantic OIDs have been studied in the Osiris research group but has not yet been implemented. We note that a semantic Oid is also useful for data consistency verification. Using semantic Oids for the verification of data consistency could be one perspective of this work.

Tracing of Two Example Queries

In this section we trace two sample queries in order to show the query processing in IXIA and its challenges. [25]

First Example

We want to request the current bill values for all the personal subscribers that have had a first paying delay. The user query (an Osiris query) which arrives to IXIA Query Evaluator is:

```
U-Q: (PERSONAL-SUBSCRIBER | Unpaid-Warning = 1) [Own-Call-Number.bill]
```

1. To process a query the first step corresponds to the evaluation of the context and conditions (inside the parentheses). The context in this example is a view of the Subscriber P-type and the condition also bears on one of the attributes of this P-type. In such cases we do not have to decompose the query before sending it to the Osiris Query Processor (OQP). Then the first querying step is made at the OQP.

   ```
   O-Q: (PERSONAL-SUBSCRIBER | Unpaid-Warning = 1)
   ```

   The response of this step, extracted from ISD, is a set of Goid (Oid, Source-ID) which are the Oids of the P-type SUBSCRIBER and correspond to the Subscriber database:

   ```
   O-R1 = List of (Subscriber-Oid, source)
   ```

2. The second step corresponds to the requested attributes. We find the names of two attributes in this example: Own-Call-Number is an attribute of the Subscriber P-type. It is in the Subscriber database.

   The bill attribute of the Call-Number P-type is the main requested attribute. It will be found in the Switch databases.

   Now, a query decomposition is necessary in order to prepare the Source queries and send them to the corresponding sources. The main challenge is that in order to query the bill value for each call number which is a response of the first step, we need to know the switch it corresponds to. We propose two solutions:

   - At the wrapper level: In this solution, in the wrapper of the Subscriber database, when extracting the Call-Number attribute of a subscriber from the Subscriber database, we also request the switch attribute:

     ```
     Own-Call-Number = oid (Subscriber-Number.Number)
                                        & Line.Switch(Number)
     ```

     In this solution, the response is compatible with the Goid format of the ISD table. This solution is only possible if we use a semantic Oid.

   - At the Mediator level: In this solution the number and switch attributes are extracted from the subscriber database and then a query is sent to each switch in order to extract the bill values of the requested Call-Numbers.

[25] We did not implement the Query Processor of the IXIA, but these two examples together with the algorithm which we have proposed in chapter 5 prepare the development of the query evaluator as a practical perspective.

Because of the absence of a semantic OID in this implementation, we choose the second solution. Therefore a query is sent to the wrapper of the subscriber database and the list of (number, switch) pairs of the response subscribers will be extracted. The query is as follows:

```
S-q1: (Personal-Call-Number | Own-Subscriber.Oid ∈ O-R1)
                                                [Number, Switch]
```

The response is a list of numbers and their corresponding switches (sources):

```
S-r1 = List of (Number, Switch)
```

Now the partial source queries must be sent to the sources (switch databases) in order to extract the bill values. In this implementation we have two switch databases. Two source queries may be prepared by the Query Evaluator:

```
S-q2 (to switch1) = (Personal-Call-Number |
                          number = subset1 of S-r1) [bill]
S-q3 (to switch2) = (Personal-Call-Number |
                          number = subset2 of S-r1) [bill]
```

Second Example

We request the bill value for all the call numbers which are authorized for international call out when their subscribers are not faithful subscribers. The user query which arrives to the IXIA Query Evaluator in the Osiris query format is as follows:

```
U-Q: (CALLl-NUMBER | (Authorization-International-Call-Out = "yes") &
                        (Own-Subscriber.Unfaithfulness = "yes")) [bill]
```

1. As discussed in the last example the first step is to process the context and the conditions of the query. Here the conditions correspond to two P-types of the global schema. In such cases a query decomposition step is made before Osiris query processing. The user query is decomposed into two Osiris queries as follows:

   ```
   O-Q1: (SUBSCRIBER | Unfaithfulness = "yes") [Own.Call-Number]
   O-Q2: (CALL-NUMBER | Authorization-International-Call-Out = "yes")
   ```

   The responses of these two queries are the two sets of Call-Number Oids: O-R1 and O-R2. However, in order to extract the O-R1, as in the last example, O-Q1 is decomposed into:

   ```
   O-Q11: (SUBSCRIBER | Unfaithfulness = "yes")
   ```

   the response is a list of pairs (Subscriber-Oid, Source): O-R11, and a query must then be sent to the Subscriber database in order to extract

   ```
   S-q1: (CALL-NUMBER | Own-Subscriber.oid ∈ O-R11)
   ```

   The response to this step is

   ```
   O-R1 ∩ O-R2.
   ```

2. This step is like the second step of the first example.

6.4 Conclusion

In this chapter we have presented our data integration application. We have integrated two PSTN switches and a Subscriber database into an IXIA mediator. The conditions of this application are compatible with our hybrid integration approach. There are a number of perspectives for the approach developed in this thesis and consequently for this application. We will present these perspectives in the conclusion of this thesis.

```
view SUBSCRIBER // minimal view of p-type SUBSCRIBER
  attr
    Subscriber-Code :  INT;
    First-Name :  STRING;
    Last-Name :  STRING;
    National-Identity :  STRING; // Unique for each real person
    Adhering-Date :  DATE;
    Address :  STRING;
    Code-Postal :  INT;
    Subscriber-Type :  STRING in {"Personal", "Public", "Commercial"};
    Own-Call-Number :  List of CALL-NUMBER;
    Current-Bill-paid :  BOOLEAN; // {"yes", "no"}
    Unpaid-Warning :  INT in {0, 1, 2};
    Unfaithfulness :  BOOLEAN;
    Line-Numbers :  INT;
    Critical-Line-Numbers :  INT
    Key :  Subscriber-Code; // External key (unique code)
end;
view PERSONAL-SUBSCRIBER : SUBSCRIBER.
  assertion
    Subscriber-Type = "Personal";
    Line-Numbers = 1;
    Critical-Line-Numbers = 0;
end PERSONAL-SUBSCRIBER;
view PUBLIC-SUBSCRIBER : SUBSCRIBER.
  assertion
    Subscriber-Type = "Public";
end PUBLIC-SUBSCRIBER;
view COMMERCIAL-SUBSCRIBER : SUBSCRIBER
  attr
    Private-Subscriber :  INT in {"yes", "no"}
  assertion
    Subscriber-Type = "Commercial";
end COMMERCIAL-SUBSCRIBER;
view PRIVARTE-COMMERCIAL-SUBSCRIBER : COMMERCIAL-SUBSCRIBER
  assertion
    Private-Subscriber = "yes";
    Critical-Line-Numbers ≤ 3;
end PRIVARTE-COMMERCIAL-SUBSCRIBER;
view FAITHFUL-SUBSCRIBER : SUBSCRIBER
  assertion
    Unfaithfulness = "no";
end FAITHFULL-SUBSCRIBER;
view UNFAITHFUL-SUBSCRIBER : SUBSCRIBER
  assertion
    Unfaithfulness = "yes";
end UNFAITHFUL-SUBSCRIBER;
```

Figure 6.11: Definition of the SUBSCRIBER P-type.

```
view CALL-NUMBER // minimal view of p-type CALL-NUMBER
  attr
    Number :  INT;
    For-Subscriber :  SUBSCRIBER; (or:  Subscriber-Code :  INT)
    Address :  STRING;
    Switch-Code :  INT;
    Call-Number-Kind :  STRING in {"personal", "public", "commercial", "critical"}
    National-Charges :  REAL;
    International-Charges :  REAL;
    Mobile-Charges :  REAL;
    National-Warning :  BOOLEAN;
    International-Warning :  BOOLEAN;
    Mobile-Warning :  BOOLEAN;
    Active-Line :  BOOLEAN;
    Authorization-National-Call-out :  BOOLEAN;
    Authorization-International-Call-out :  BOOLEAN;
    Authorization-Mobile-Call-out :  BOOLEAN;
    Authorization-National-Call-in :  BOOLEAN;
    Authorization-International-Call-in :  BOOLEAN;
    Authorization-Mobile-Call-in :  BOOLEAN;
    Inactivate-Alert :  INT in {0, 1, 2};
    Key :  Number
  assertion
    Active-Line = "no" => All Authorizations = "no";
end;
view PERSONAL-CALL-NUMBER : CALL-NUMBER
  assertion
    Call-Number-Kind = "personal;
    National-Charges ≥ A1 => National-Warning = "yes";
    National-Charges ≥ A2 => Authorization-National-Call-out = "no";
    International-Charges ≥ B1 => Authorization-International-Call-out = "no";
    Mobile-Charges ≥ C1 => Authorization-Mobile-Call-out = "no";
    Inactivate-Alert = 1 => Authorization-National Call-out = "no"
        Authorization-international call-out = "no"
        Authorization-mobile call-out = "no"
        Authorization-International Call-in = "no"
        Authorization-Mobile Call-in = "no"
    Inactivate-alert = 2 => Active-Line = "no";
end PERSONAL-CALL-NUMBER;
```

Figure 6.12: Definition of the CALL-NUMBER P-type, part I.

```
view PUBLIC-CALL-NUMBER : CALL-NUMBER
  assertion
    Call-Number-Kind = "public";
    National-Charges ≥ A2 => National-Warning = "yes";
    International-Charges ≥ B1 => Authorization-International-Call-out = "no";
    Mobile-Charges ≥ C1 => Mobile-Warning = "yes";
    Mobile-Charges ≥ C2 => Authorization-Mobile-Call-out = "no";
    Inactivate-Alert = 1 => Authorization-National Call-out = "no"
        Authorization-international call-out = "no"
        Authorization-mobile call-out = "no"
        Authorization-International Call-in = "no"
        Authorization-Mobile Call-in = "no"
    Inactivate-alert = 2 => All call-in and call-out authorization are "no";
end PUBLIC-CALL-NUMBER;
view COMMERCIAL-CALL-NUMBER : CALL-NUMBER
  assertion
    Call-Number-Kind = "commercial";
    National-Charges ≥ A2 => National-Warning = "yes";
    International-Charges ≥ B1 => International-Warning = "yes";
    International-Charges ≥ B2 => Authorization-International-Call-out = "no";
    Mobile-Charges ≥ C1 => Mobile-Warning = "yes";
    Mobile-Charges ≥ C2 => Authorization-Mobile-Call-out = "no";
    Inactivate-Alert = 1 => Authorization-National Call-out = "no"
        Authorization-international call-out = "no"
        Authorization-mobile call-out = "no"
        Authorization-International Call-in = "no"
        Authorization-Mobile Call-in = "no"
    Inactivate-alert = 2 => All call-in and call-out authorization are "no";
end COMMERCIAL-CALL-NUMBER;
view CRITICAL-CALL-NUMBER : CALL-NUMBER
  assertion
    Call-Number-Kind = "critical";
    Inactivate-Alert = 1 => National-Warning = "yes";
        Authorization-International Call-Out = "no"
    Inactivate-Alert = 2 => National-Warning = "yes";
        Authorization-International Call-Out = "no"
    International-Warning = "yes";
        Authorization-Mobile Call-Out = "no"
end CRITICAL-CALL-NUMBER;
```

Figure 6.13: Definition of the CALL-NUMBER P-type, part II.

Figure 6.14: Indexation System in the Mediator

Conclusion

1 Conclusion

Data integration is one of the solutions to bring together several information sources which are autonomous, distributed and heterogeneous. It aims to provide a homogeneous view to users and to hide all the aspects of heterogeneity from them. In this thesis we have studied different existing solutions for integrating information sources. We have reviewed existing approaches for data integration: fully virtual, fully materialized, and hybrid approaches. We have been especially concerned by hybrid approaches. A hybrid approach can provide a trade-off between query response time and data freshness in an integration application. It can also be an optimal solution in certain applications where data sources have important differences in the frequency of data updates and need on-line access.

Most of the existing hybrid approaches propose the partitioning of the global view into virtual and materialized integration. Some data reside in the sources and are extracted at query time. We call such approaches horizontal hybrid approaches. In such approaches the query processing remains time-consuming for accessing the virtual data, and the materialized data refreshing is similar to that of fully materialized approaches.

We have proposed a hybrid approach which is based on object indexating at the mediator level. The object indexation structure and the data needed to refresh it are materialized, while other data remain in the data sources and are queried at query time. We have called such an approach a vertical hybrid data integration approach.

Our proposed approach, IXIA, is based on the Osiris platform which is an object-based database and knowledge base system. It implements the P-type data model, a model based on a hierarchy of views.

IXIA has two main objectives:

1. Extend semantic query optimization to the queries of the integration system
2. Provide a flexible data refreshing mechanism which makes this approach capable to tolerate data sources with different characteristics and to choose an optimal trade-off between query response time and data freshness

The first objective is achieved by materializing the object indexation structure of the objects of the underlying sources, which correspond to the global schema, at the mediator level. IXIA uses an extension of the indexation system of Osiris. The Osiris indexation system relies on the partitioning of the object space at the conceptual level and based on the constraints in the views. The queries of the system are posed as a function of views and conditions in which certain conditions are on the classifying attributes, the attributes that participate in the indexation system. Thus, taking advantage of such an indexation system, the search space for a query in IXIA is reduced to a subset of both sources and objects.

The second objective is achieved by developing a module named Modification Detector, which provides a tolerant system of indexation refreshing. Each data source has its Modification Detector, which is independent of others and which can be executed at different frequencies. This second characteristic, as we have shown in the telecommunication application in chapter 6, lets us manage the integration application according to the possibilities and characteristics of different sources. We have shown that the query response time in this approach is better than that of fully virtual data integration approach and the data freshness is much better that of a fully materialized data integration approach.

2 Perspectives

Several perspectives can be considered for this thesis in both the research and the practice aspects. We begin with the practice perspectives and we present research perspectives in the second subsection.

2.1 Practical Perspectives

We propose two practice perspectives for this thesis.

IXIA Query Evaluator In this thesis we have provided a general algorithm for the query evaluator of IXIA and we have described its functionality and its relation with the Osiris Query Processor through two examples in chapter 6. However, the query reformulation and plan generation has been done manually for the tests. Relying on this algorithm and the tests, a more detailed algorithm can be provided and a query reformulator and query plan generator can be developed.

Horizontal and Vertical Hybrid Integration We have discussed in chapter 5 the IXIA vertical hybrid approach. As in the telecommunication application (chapter 6), there are cases in which we have some historical data integrated in a Data Warehouse.

A new version of IXIA can cover these data by integrating the DW as a source into the IXIA mediator. Figure 1 shows our proposition. In this new version, a wrapper can be developed for the DW and its data will participate in the IXIA indexation system. Through this tolerant architecture we will have the vertical and horizontal hybrid paradigms simultaneously.

Query Interface Developing an interface to request more sophisticated queries may be another perspective of this work. The queries that IXIA processes in this implementation can not do the complementary operations such as adding subtracting or averaging. For such queries, a new interface must be developed.

2.2 Research Perspective

We consider two main research perspectives for this thesis.

Consistency Verification through the IXIA Mediator We believe that by using the indexation data in the mediator, we can verify the consistency of some important data. Since all the constraints of the views of the P-types are implemented in the indexation system of the mediator, we can verify if semantically similar objects from different data sources are coherent.

Figure 1: Data integration query processing.

In the telecommunication application, for example, all the call numbers are in two data sources: the subscriber database, which contains all the call numbers of the region, and one of the switch databases. One solution to achieve this goal is to develop a semantic Oid. We have used telephone numbers, which are strictly unique, to create semantic Oids. In the indexation system an Oid is joined with the source code into a pair (Oid, Source). If the Oids which correspond to the same objects are not in the same Eq-Class, IXIA can detect an inconsistency.

Local Query Optimization Traditional query processing uses statistic data which are computed offline to perform query optimization. This is generally effective in a standard database in which data and computing environment are under the strict control of the database [Ives, 2002]. The context of data integration is very dynamic and the techniques that are efficient in a single database often cannot be useful in this context. In IXIA we have used semantic knowledge (i.e., the constraints of views) at the mediator level to achieve global optimization for logical query plans. This semantic optimization has been guaranteed by the materialization of the indexation structure and the data needed for the incremental updating of this indexation.

The local query optimization, which is the optimization for partial queries over data sources, is still a subject of study.

Dependency between two Data Sources in the Classifying Attributes In the telecom integration application, if we use all the constraint procedures in the global schema, some classifying attributes of the CALL-NUMBER P-type (National-Charge, International-Charge and Mobile-Charge) are in the switches and are not available for the subscriber database. In this case, a Call-Number object of the global schema must be classified in the indexation system by using the attributes of two autonomous sources independently. Several solutions can be proposed. One of these solutions is to define only one object at the mediator level and to classify it using the corresponding attributes from two sources. In this case, the Source Modification Manager has to joint all classifying attributes received from two Modification Detectors. Using a semantic Oid can simplify the task of the Source Modification Manager. In the absence of a semantic Oid, another solution may be the materialization of a unique attribute of the P-type, "number" in this example.

Other Indexation Approaches In this thesis we have focused on providing a hybrid architecture which can offer an optimal trade-off between query response time and data freshness and we have made a comparison of this architecture with other data integration approaches, notably hybrid approaches. A further comparative study of interest could be the comparison of the Osiris indexation system with other existing object indexation systems which provide indexation use of the semantic knowledge at the conceptual level.

Osiview Integration Approach Another perspective of this thesis is to give a more in-depth view of the Osiview approach (Annex A). In its architecture, queries are classified in a multi-mediator system in which the mediators correspond to a hierarchy of Osiris views. We can thus achieve efficient query processing in a fully virtual data integration framework.

1 Conclusion Générale

L'intégration de données est l'une des solutions permettant de rassembler plusieurs sources d'informations qui sont autonomes, distribuées et hétérogènes. Elle vise à fournir une vue homogène aux utilisateurs et à cacher tous les aspects d'hétérogénéité. Dans cette thèse nous avons étudié différentes solutions existantes pour l'intégration des sources d'informations.

Nous avons examiné les approches existantes pour l'intégration de données : approches entièrement virtuelles, entièrement matérialisées, et d'hybrides. Nous avons été particulièrement concernés par des approches hybrides. Une approche hybride peut fournir un compromis entre le temps de réponse aux requêtes et la fraîcheur de données dans une application d'intégration. Ce peut également être une solution optimale dans certaines applications où les sources des données présentent des différences importantes dans la fréquence des mises à jour de données et le besoin d'accès en ligne aux données.

La plupart des approches hybrides existantes proposent de diviser la vue globale entre intégration virtuelle et matérialisée. Certaines données restent t dans les sources et sont extraites au moment de la requête. Nous appelons de telles approches des approches hybrides horizontales. Dans ces approches le traitement des requêtes restent long pour l'accès aux données virtuelles, et le rafraichissement des données matérialisées est semblable à celui des approches entièrement matérialisées.

Nous avons proposé une approche hybride qui est basée sur l'indexation des objets au niveau du médiateur. La structure d'indexation d'objet et les données requises pour la mettre à jour sont matérialisées, alors que d'autres données demeurent dans les sources et sont interrogées au moment de la requête. Nous avons appelé une telle approche une approche hybride verticale d'intégration de données.

L'approche proposée, IXIA, est basée sur la plateforme Osiris qui est un système de base de base de données et de connaissances basé sur un modèle d'objet. Il met en application le modèle de données des P-types, qui est un modèle basé sur une hiérarchie de vues. IXIA a deux objectifs principaux :

1. Etendre l'optimisation sémantique des requêtes à toutes les requêtes du système d'intégration
2. Fournir un mécanisme flexibles de mise à jour des données qui rend cette approche capable de tolérer des sources de données avec différentes caractéristiques, et choisir un compromis optimal entre le temps de réponse de requête et la fraîcheur de données.

Le premier objectif est atteint en matérialisant la structure d'indexation des objets des sources locales, qui correspondent au schéma global, au niveau du médiateur. IXIA emploie une extension du système d'indexation d'Osiris. Le système d'indexation d'Osiris se fonde sur la division de l'espace des objets au niveau conceptuel et basée sur les contraintes des vues. Les requêtes du système sont posées en fonction des vues et des conditions dont certaines portent sur les attributs classificateurs, i.e., les attributs qui participent à l'indexation. Ainsi, profitant d'un tel système d'indexation, l'espace de recherche pour une requête en IXIA est réduite à un sous-ensemble des sources et des objets. Le deuxième objectif est atteint en développant un module appelé Modification Detector (MD), qui fournit un système tolérant de la mise à jour de l'indexation. Chaque source de données a son détecteur de modification (MD), qui est indépendant des autres et qui peut être exécuté à différentes fréquences. Cette deuxième caractéristique, comme nous l'avons montré dans l'application de télécommunication au chapitre 6, nous laisse contrôler l'application d'intégration selon les possibilités et les caractéristiques des différentes sources. Nous avons montré que le temps de réponse aux questions dans cette approche est meilleur que celui de l'approche

d'intégration de données entièrement virtuelle, et la fraîcheur de données est meilleure que celle d'une approche entièrement matérialisée.

2 Perspectives

Plusieurs perspectives peuvent être considérées pour cette thèse, dans le domaine dela recherche et aussi pour les aspects pratiques. Nous commençons par les perspectives pratiques et nous présentons des perspectives de recherche dans la deuxième sous-section.

2.1 Perspectives Pratiques

Nous proposons deux perspectives pratiques pour cette thèse.
Query Evaluator d'IXIA dans cette thèse nous avons fourni un algorithme général pour le "Query Evaluator" d'IXIA et nous avons décrit sa fonctionnalité et sa relation avec le "Query Processor" d'Osiris par deux exemples au chapitre 6. Cependant, la reformulation de requête et la génération de plan a été faite manuellement pour les essais. En s'appuyant sur cet algorithme et les essais réalisés, un algorithme plus détaillé peut être fourni et un reformulateur de requête et un générateur de plan de requête peuvent être développés.

Intégration Hybride Horizontale et Verticale nous avons discuté au chapitre 5 l'approche hybride verticale d'IXIA. Comme dans l'application de télécommunication (chapitre 6), il y a des cas dans lesquels nous avons quelques données historiques intégrées dans un entrepôt de données. Une nouvelle version d'IXIA peut couvrir ces données en intégrant l'entrepôt de données (DW) comme source dans le médiateur d'IXIA. La figure 1 montre notre proposition. Dans cette nouvelle version, un wrapper peut être développé pour le DW et ses données participeront au système d'indexation d'IXIA. Grâce à cette architecture tolérante nous obtenons simultanément les paradigmes hybrides verticaux et horizontaux.
Interface de Requête Développer une interface pour poser des requêtes plus sophistiquées peut être une autre perspective de ce travail. Dans l'implémentation actuelle, les requêtes qui peuvent être traitées par IXIA ne peuvent pas faire les opérations complémentaires telles que l'addition, la soustraction ou un calcul de moyenne. Pour de telles requêtes, une nouvelle interface doit être développée.

2.2 Perspectives de Recherche

Nous considérons deux perspectives principales de recherches pour cette thèse.
Vérification de la cohérence par le médiateur d'IXIA nous croyons qu'en employant les données d'indexation dans le médiateur, nous pouvons vérifier la cohérence de certaines données importantes. Puisque toutes contraintes des vues des P-types sont mises en application dans le système d'indexation du médiateur, nous pouvons vérifier si sémantiquement les objets semblables de différentes sources sont cohérents.

Dans l'application de télécommunication, par exemple, tous les Call-Numbers sont dans deux sources de données : la base de données des clients (Subscriber), qui contient tous les Call-Numbers de la région, et une des bases de données de switch. Une solution pour réaliser ce but est de développer un Oid sémantique. Nous avons employé les numéros de téléphone, qui sont strictement uniques, pour créer des Oids sémantique. Dans le système d'indexation un Oid est joint avec le code source dans une paire (Oid, source). Si les Oids qui correspondent aux mêmes objets ne sont pas dans la même Eq-Classe, IXIA peut détecter une contradiction.

Optimisation de Requête au niveau Local Les approches traditionnelles de traitement des requêtes utilisent les données statistiques qui sont calculées de manière off-line pour réaliser l'optimisation de requête. C'est généralement efficace dans une base de données standard dans laquelle l'environnement de données et de calcul sont sous la commande stricte de la base de données [Ives, 2002]. Le contexte de l'intégration de données est très dynamique et les techniques qui sont efficaces dans une base de données simple souvent sont souvent inadaptées dans ce contexte. Dans IXIA nous avons employé la connaissance sémantique (i.e., les contraintes des vues) au niveau du médiateur pour réaliser l'optimisation globale pour des plans logiques de requête. Cette optimisation sémantique a été garantie par la matérialisation de la structure d'indexation et des données requises pour la mise à jour incrémentale de cette indexation.

L'optimisation locale de requête, qui est l'optimisation des requêtes partielles sur les sources, est toujours un sujet d'étude.

Dépendance entre deux Sources sur les Attributs Classificateurs Dans l'application d'intégration de télécommunication, si nous employons toutes les procédures de contraintes dans le schéma global, certains attributs classificateurs du P-type de CALL-NUMBER (National-Charge, International-Charge et Mobile-Charge) sont dans les switch et ne sont pas disponibles pour la base de données des clients (subscriber). Dans ce cas, un objet Call-Number du schéma global doit être classé dans le système d'indexation en employant les attributs de deux sources autonomes. On peut proposer plusieurs solutions. Une de ces solutions est de définir seulement un objet au niveau du médiateur et de le classer en employant les attributs correspondants de deux sources. Dans ce cas, le " Source Modification Manager " doit réunir tous les attributs classificateurs reçus de deux " Modification Detector ". Employer un Oid sémantique peut simplifier la tâche pour le " Source Modification Manager ". En l'absence d'un Oid sémantique, une autre solution peut être la matérialisation d'un attribut unique du P-type, " number" dans cet exemple.

Autres Approches d'Indexation Dans cette thèse nous nous sommes concentrés sur fournir une architecture hybride qui peut offrir un compromis optimal entre le temps de réponse à une requête et la fraîcheur des données, et nous avons fait une comparaison entre cette architecture et d'autres approches d'intégration de données, notamment les approches hybrides. Une autre étude comparative pourrait concerner le système d'indexation d'Osiris et d'autres systèmes existants d'indexation d'objet qui fournissent l'utilisation d'indexation de la connaissance sémantique au niveau conceptuel.

Approche d'Intégration Osiview Une autre perspective de cette thèse est l'approfondissement de l'approche d'Osiview (annexe A). Dans cette architecture, les requêtes sont classées dans un système de multi-médiateur dans lequel les médiateurs correspondent à une hiérarchie de vues d'Osiris. Nous pouvons ainsi réaliser le traitement efficace de requête dans un cadre entièrement virtuel d'intégration de données.

Appendix A

Osiview: A View-Based Data Integration Approach

A.1 Introduction

Semantic mediator approaches for the integration of heterogeneous sources usually lack the benefits of a class-based hierarchical structure, mainly because of the constraints of the standard object model where an object can belong to only one class. Using the multi-view model of Osiris system, and taking advantage of its query classification, we propose a multi-level mediator approach where each mediator is a point of view over the corresponding sources. A global mediator at the upper level provides a global view of the system and constitutes the global schema of the integration system that is used to classify the queries into the hierarchy of mediators. This approach offers some of the advantages of a hierarchical architecture and provides a query optimization for a mediator integration system.

A.2 State of the Art

The mediator approach is one of the main virtual solutions proposed to develop a data integration systems. In this approach, data is requested from local sources by one or several mediators. The architecture of a mediation introduced in [Wiederhold, 1992] consists of two primary elements: a translator and a mediator. The translators provide access to data in the data sources using a common data language. They also translate the decomposed queries from a common query language to the queries in local sources query languages.

The architecture of a mediation approach is one of the main features optimizing the performance of the query answering. In addition, since the global schema acts the interface to the user for accessing data, the choice of the platform and the language for expressing and querying such a schema is crucial [Poggi, 2006].

Most of the projects that develop semantic mediator approaches have a global structure [Calì *et al.*, 2003, Levy, 1996, Kirk *et al.*, 1995]. The global integrated schema is obtained in one mediation step.

In this paper, we present an approach offering a hierarchical structure to semantic data integration by employing the hierarchy of views. This method presents some of the advantages of a hierarchical integration method, notably the optimization of mediator system query answering.

A.3 Osiview: A multi-level mediator system

Our hierarchical mediator approach is based on the notion of view. Considering a predefined global schema which is a set of views, we develop a hierarchy of mediators in which a mediator corresponds to each view. A global mediator is developed that contains the global schema and using this, it classifies the queries in the hierarchy of mediators and it decomposes the more complex queries if necessary.

The method can be decomposed in four steps:

1. After developing a global mediator that integrates all the P-types of the global schema, we design the hierarchy of mediators. Corresponding to each view in the global schema, a mediator is developed that contains information consisting of a query and the information about the sources that will be used to translate the user queries into queries to the local sources.

2. For each source we develop a correspondence between attributes in order to:

   - Classify the source in the hierarchy of mediators
   - Develop the mediators in different levels

3. The classification of local sources in the hierarchy of mediators is performed in this step. We associate to each mediator the local sources which contain the corresponding information. We then make several projections of each local source to the mediators in which it is classified.

4. Query evaluation is performed by a query classification module in the multilevel mediator. When a user's query arrives to the global mediator, using query classification, the global mediator finds the more specialized view that can satisfy this query. Then the query is classified under the mediator corresponding to this view. More complex queries are decomposed into partial queries by the global mediator and are sent to the corresponding mediators. The global mediator then integrates the partial responses. There are cases when a partial query must wait for the responses of other partial queries.

The figure A.1 shows the architecture of the multilevel mediator approach. In the next section, we describe our model in an example related to relational database sources.

A.3.1 Query classification using views in Osiris

Query optimization in Osiris is done either by classifying the query into the views of the P-type, or by using the indexing of objects through Eq-classes [Simonet, 1996]. We consider here only the first kind of query optimization, which we call query classification. Query classification is also supported by the classification network. It is done in five steps:

1. for each attribute with domain restrictions in the query (corresponding to the where part in SQL), named query attribute, determination of its SSDs of interest.

2. for each SSD of interest of a query attribute, assignation of valid (V) or a potential (P) value depending on all (/only some) the elements of a SSD are valid to the query.

3. Determination of the Eq-classes tuples. For each tuple, the SSD of a query attribute is an explicit SSD; for other attributes the corresponding SSD is the generic SSD, represented by *.

Figure A.1: Multilevel Mediator Architecture

4. Assignation to each tuple of a truth value. This value is obtained by the conjunction of the V and P values of its explicit SSDs .

5. Classification (through the classification network) of the Eq-classes tuples in order to determine the valid and potential views of the query.

Let us Consider the definition of the P-type PERSON in figure B.2. By using the predicates on the attributes Age, Military Service and Sex, we obtain the following partitioning of the attributes:

```
SSDs of Age: d11=[0, 18[, d12=[18, 27], d13=]27, 65[, d14=[65,140]
SSDs of Sex: d21={"f"}, d22={"m"}
SSDs of MilitaryService: d31={"no"}, d32={"‘yes"’, "exempt", "deffered"}
```

Now, consider the query person $\|Age > 25 and Sex = F[Name, BirthDay], then$:
SSDs of interest and their value

Query attributes	**SDS of interest**
Age	{(d12,P), (d13,V), (d14, V)}
Sex	{d21, V}

Eq-classes and their truth value

Eq-class and associated value
(<d12, d21, *>, **P**)
(<d13, d21, *>, **V**)
(<d14, d21, *>, **V**)

A.3.2 Example

We consider three sources containing medical information. In each source, there is a unique attribute for each person.
S1:

```
T1:  Patient( code, name, sex, birth-date, consultation-reason )
T2:  Hospitalization( code, date, reason, service)
T3:  Doctor( code,name,birth-date, specialty, service)
T4:  Pregnant( code, type, ddate, test, result, comment)
```

S2:

```
T1:  Patient( code, nom, sex, datedenaissance, raison,docteurs)
T2:  Medecin( code, nom, service)
T3:  Service( scode, nom, medecin)
T4:  Grossesse( code, test, result)
```

S3:

```
T1:  MedicalTest(code, name, test-service)
T2:  PrognantTest( code, case, month, service)
T3:  Radiology(code, machine,case, result-delay)
```

In the first step, we develop the global schema and then the corresponding multilevel mediator as is shown in figure A.3. Each mediator is a view of the corresponding local sources and contains a query for each of these local sources. We define the object corresponding to each mediator that will be the base of its predefined queries for the associated sources. Each mediator composes the user query with these predefined queries to build the queries toward local sources. Figure A.4 shows examples of these objects for the *Person*, *Patient* and *Pregnant* mediators.

In this step we must associate the corresponding local sources to each mediator. For this purpose the local sources have to be indexed. In the relational model this indexation is made with the names of tables and fields. For this purpose we must build the attribute correspondence of each source. The result of local sources classification in this example is shown below. Using the attribute correspondence in this step could be done automatically.

```
view PERSON - Minimal view of P-type PERSON
  attr Code:  int
       Name:  STRING
       Sex:  CHAR
       BirthDay:  DATE
       MilitaryService:  STRING
       followedBy:  setof DOCTOR; -doctor is a view of the P-type person
  key - External Key ; not mandatory
  methods
       - other function specification
  assertions
       - Domain Constraits
       Sex in "f", "m";
       0 <= age <= 120
       Military service in "yes", "no", "deferred", "exempt";
       - Inter-Attribute Dependencies
       Age < 18 -> MilitaryService = "no";
       Age >= 18 and sex = "m" -> MilitaryService in "yes", "deffered", "exempt";
       Sex = "f" -> MilitaryService = "no";
end; - The minimal view automatically contais a private attribute OID: toid

view PATIENT : PERSON - PATIENT specializes PERSON
  attr ConsultationDate :  DATE ;
       ConsultationReason:  STRING;
end;

view HOSPITALIZEDPATIENT : PATIENT
  attr Service :  SERVICE; - a view of another P-TYPE
       IncomeDate:  Date;
end ;

view DOCTOR : PERSON
  attr Speciality :  setof STRING in "cardiologist", "nephrologists",
                  "diabetologist", "rheumatologist";
       experience:  int;
end ;

view ADULT : PERSON assertions age >= 18 end ;
view MINOR : PERSONNE assertions age < 18 end ;
view SENIOR : PERSONNE assertions age >= 65 end ;
view WOMAN : PERSONNE assertions sex = "f" end ;
```

Figure A.2: P-type of PERSON

Person: S1, S2	Patient: S1, S2	Doctors: S1, S2
Hospitalized: S1	Pregnant: S1, S2(partially)	Specialist: S1
Test: S3	Radiology: S3	Pregnant: S3

Figure A.3: Multilevel Mediator Schema

Figure A.4: Mediator attributes

To translate the user query into comprehensible local source queries we need the wrapper information that consists of attributes correspondence to translate the terms of CDM to local source terms and the logics to interpret the queries. We have modulated the wrapper functionality in our mediators so that each mediator saves a block of information to each of its associated sources. We show in figure A.5 the content of the pregnant mediator as an example:

A.3.3 Query evaluation in Osiview

In the previous section, we showed that we find the more specialized mediator for a query using the query classification mechanism of Osiris. We explain it now in the continuation of our example. Suppose that the query Q1 arrives at the global mediator:

```
Q1:  (pregnant patient | case = prevision-operation) [test, result]
```

Then

```
(X :: case = prevision-operation)  and  (Y :: test, result )
```

The Global mediator sends this query to the Osiris module and it classifies the query under the *Pregnant* view. Then the global mediator sends the query to the *Pregnant* mediator. In the *Pregnant* mediator we have two associated sources: S1, S2 .
S1:

```
X'~ type = Caesarean Y' ~ test, result
--> select Y'  from  T4  where  X'
```

Then the S1 query is:

```
Pregnant
  Sources:  S1, S2
  Mediator Query:  [ ( code, case, deliveryDate, test,result )| X ] [Y] *
  S1:  T4, key:  code

    attributes correspondence
        code code
        case type
        deliveryDate ddate
        test test
        result result
    Query Logic:  select Y' from T4 where X' **
    S2:  T4, key:  code

    attributes correspondence
        code code
        case Null
        test test
        result result
    Query Logic:  select Y' from T4 where X' **

* X: condition of query    Y: the desired attributes
** X' and Y' are built with the replacement of source's terms using attributes
correspondence
```

Figure A.5: Pregnant mediator

```
select   test, result   from   T4   where   type = sezarian
```

S2:

```
case = null -->  response is null
```

There are cases where a query contains attributes that are not in the same mediator. In these cases, we send the key of the results objects and the rest of the desired attributes to the global mediator and the global mediator sends a partial query to the corresponding mediator.

A.4 Advantages

Source and query classification in this method optimizes query evaluation. In addition, we have decomposed the wrapper information in the mediators, which optimizes the translation process by reducing the wrapping space to the necessary attributes for each query.

We make a fusion of the user's query and the mediator query using simple query logic for each source. It modulates and optimizes the translation process and offers a semi-automatica wrapping method.

Another advantage of this method is that a source becomes useful when it is classified in the mediators and we build its query logic in each mediator.

A.5 Conclusion

In this chapter, we have presented a multi-level semantic mediator approach based on views. Like other multi-view systems, [Rundensteiner, 1992] as an example, Osiris has the properties that make it possible to develop an efficient hierarchical architecture for a semantic mediator. We try to optimize performances and also to take advantage of the structural mediator approach by using this hierarchical view architecture. Each mediator is a view of its related sources. The global mediator contains the global schema and uses this schema to classify the application queries. Query evaluation and optimization are developed based on the view classification in Osiris. When a query arrives to the global mediator, it can recognize the most specialized mediators that satisfy or potentially satisfy the query. Each mediator has a query and the logic information for all of its corresponding sources. Using this information, it mixes the user query and mediator query to obtain the sources queries. This approach simplifies the mediator and the translator development. Future work will focus on the semi-automatic development of the mediators.

Appendix B

An example of Object Classification and Query Evaluation in OSIRIS

In this annex we follow an Osiris example from the definition of a P-type to query evaluation. This is a simple and complete example whose purpose is to show the object indexation and query evaluation processes in Osiris. Figure B.1 shows the hierarchy of views of the P-type PERSON which is a simplified version of the P-type PERSON presented in chapter 4; figure B.2 presents its definition.

Figure B.1: Hierarchy of views of the P-type PERSON

Views of a P-type

The P-type definition consists of the minimal view PERSON and other views which specialize the minimal view. As presented in figure B.2 , each specialized view contains constaints over the attributes of the P-type.

Object Space Partitioning The domain constraints of an attribute in all the views participate in the partitioning of its domain into Stable SubDomains. While the value of this attribute for an object varies within a SubDomain, the object continue to satisfy exactly the same elementary (domain) predicates of this attribute. For example figure B.3 presents the partitioning of the domains of for the classifying attributes of the P-type PERSON.

Eq-class

The partition of the domain of each attribute of a P-type is prolonged into the partition of its object space and constitutes the classification space of the P-type. In our example, the classification space is a three-dimensional space as shown in figure 4.4. Each element of the

```
view PERSON // minimal view of P-type PERSON
  attr Code :  INT;
    Lastname :  STRING;
    Sex :  CHAR in { F, M }; // assertion a1
    Birthdate :  DATE;
    Age :  INT in ]0,140] // assertion a2
    MilitaryService :  STRING in
        {yes, no, ongoing, done}; // assertion a3
    key code
  assertions
    a4:  Age < 18 ==> MilitaryService = no;
    a5:  Age >= 18 and
        Sex = M ==> MilitaryService in {yes, ongoing, done};
    a6:  Sex = F ==> MilitaryService = no;
end ;
view TEACHER : PERSON
  attr salary :  REAL;
    status :  STRING in {Assistant-Professor, Professor};
end TEACHER ;
view STUDENT : PERSON
  attr studies :  setof STRING in { MASTER, MASTER2, THESIS};
    EndYear :  INT;
end STUDENT;
view TEACHING-ASSISTANT : STUDENT, TEACHER // specialize STUDENT and TEACHER
  assertions Age <= 35
end ;
view SPORTSMAN : PERSON // specialize PERSON
  attr IsSportsman:  BOOL;
  attr Disciplines:  STRING;
end SPORTSMAN;
```

Figure B.2: P-type PERSON.

classification space is called an Eq-class. When several attributes of an object are modified while remaining in the same Stable SubDomains, the object continues to satisfy the same set of predicates. Two object of the same Eq-class satisfy the same assertions and consequently validate the same views.

For each P-type, the valid Eq-classes of the P-type are those which satisfy the assertions of the minimal view. For example in the P-type PERSON:

```
age >= 18 -->  military-service in {yes, ongoing, done}
      (F / [18-35] / no)  : VALID
      (M / [18-35] / no)  : INVALID
```

Figure B.3: Stable SubDomains

B.1 Object Classification

In this section we follow our example by presenting the different steps of the Object Classification process.

View classification

The classification of an object in the views of a P-type aims at answering the following question :

Which are the Eq-classes that validate or invalidate a view ?

Figure B.4: View Classification

To simplify, in the rest of this example, we consider the views with assertions on two attributes: *Age* and *Sex*. The value of the attribute *MilitaryService* is indifferent for these views. These views are as follows:

```
View MAN:  Sex = M
View WOMAN: Sex = F
View MINOR Age < 18
View MAJOR Age >= 18
View TEACHING-ASSISTANT as defined in the P-type
```

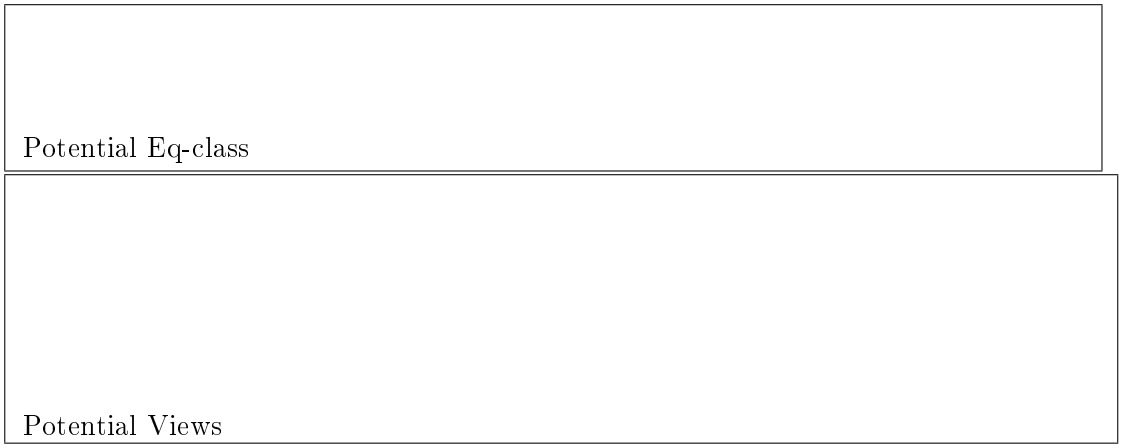

Figure B.5: Example of an Incomplete Object

Figure B.4 shows the *valid* and *invalid* Eq-classes for several views.

```
Age : d11 = [0, 18[, d12 = [18, 35], d13 = ]35, 140]
Sex : d21 = {F}, d22 = {M}
```

We can pose the above question in another way:

Which are the views that satisfy the objects of an Eq-class.

The result of this step is the classification of the Eq-class and consequently its objects in the views. In other words, we determine the valid, invalid and potential views for each Eq-class and thus for all objects of the Eq-classes. The result of this step is shown in the table B.1. However

Eq-classe	WOMAN	MAN	MINEOR	MAJOR	TEACHING-ASSISTANT
[d11,d21]	Valid	Invalid	Valid	Invalid	Invalid
[d11,d22]	Valid	Invalid	Invalid	Valid	Valid
[d11,d23]	Valid	Invalid	Invalid	Valid	Invalid
[d12,d21]	Invalid	Valid	Valid	Invalid	Invalid
[d12,d22]	Invalid	Valid	Invalid	Valid	Valid
[d12,d23]	Invalid	Valid	Invalid	Valid	Invalid

Table B.1: Table of Valid Invalid and Potential Views for all EqClasses

views in Osiris are not materialized and the views which satisfy the objects of an Eq-class are saved in the indexation structure. This will be shown in the next section.

Incomplete Object

An object is said to be incomplete when the value of at least a classifying attribute is unknown. An incomplete object does not belong to one Eq-class, but potentially to several ones. Figure B.5 presents the view classification for the followed object:

```
Object (lastname: HAMIDI , firstname: Soroush , age: 3 , sex: ?)
```

B.2 Object Indexation

The objects of a P-type are indexed in an indexing structure called ISD (Indexing Structure Descriptor). As shown in figure B.6, it contains :

1. A vector of Stable SubDomains which represents an (or several) Eq-class(es)
2. A vector of views providing the status (Valid, Invalid, Potential) of each view for the objects indexed by this ISD
3. A reference to the set of objects of the ISD
4. The total number of objects indexed by the ISD

Figure B.6: ISD Fields

For complete objects, an ISD contains its valid Eq-class and the view classification for this Eq-class. For an incomplete object, an ISD contains all its potential Eq-classes (see figures B.2).

The vector of views for the incomplete object of figure B.2 is shown in the figure B.2.

Eq-classe	WOMAN	MAN	MINEOR	MAJOR	TEACHING-ASSISTANT
[d11,d21]	Valid	Invalid	Valid	Invalid	Invalid
[d12,d21]	Invalid	Valid	Valid	Invalid	Invalid
ISD01	Potential	Potential	Valid	Invalid	Invalid

Table B.2: Example of an ISD for an incomplete object

Object

```
(lastname: HAMIDI, firstname: Soroush,
           age: 3, sex: ?)
```

Classifing Space

ISDs

```
[d11,d21]
[d12,d21]
```

Figure B.7: Example of the indexation of an incomplete object

Classifing Space

Query

```
Q1:(WOMAN , TEACHING-ASSISTANT | age<26)
```

ISDs

```
sex:              age:
   d11= {'F'}        d21= ]0, 18[
   d12={'M'}         d22= [18,35]
                     d23= ]35,150]
```

Figure B.8: Finding the valid Eq-classes for a query by using view classification

B.3 Query Evaluation

The general form of a questioning (object-preserving) query in Osiris is as follows : (As we described in Chapter 4, we use in our integration approach only this type of query)

```
(context | conditions ) [attributes]
```

A query can be considered as a dynamic view (a view which does not participate in the conceptual schema of Osiris). If the "context" of a query contains non-minimal views, they will participate in the query processing as a vector of response ISDs (see the example of the figure B.3).

The "condition" part of a query is used to find its satisfying Eq-classes. Table B.3 presents the valid, invalid and potential Eq-class(es) for the following query.

```
Q2: (PERSON | age<26 and military-service = no)
(age < 26 : C1) (military-service = no : C2)
```

Eq-class-code	Eq-classe	C1	C2	Q
111	F/0-18/no	V	V	V
211	M/0-18/no	V	V	V
121	F/18-35/no	P	V	P
131	F/35-150/no	I	V	I
222	M/18-35/'yes'/'done'/'differed'	P	I	I
232	M/35-150/ 'yes'/'done'/'differed'	I	I	I

Table B.3: Valid, Invalid and Potential Eq-classes for query Q2

Generally, these two parts (context and conditions) are used simultaneously to find valid or potential ISDs. There are three rules to find satisfying objects :

1. The objects of the ISDs which contains only Valid Eq-classes satisfy the query
2. The objects of the ISDs which contain an Invalid Eq-class do not satisfy the query
3. The object of other ISDs (potential ISDs) have to be verified.

We consider the following ISDs for this example :

```
ISD 01 = {Eq-class 111, 211} Valid and Valid   --> Valid
ISD 02 = {Eq-class 211, 131} Valid and Invalid --> Potential
ISD 03 = {Eq-class 222, 232} Invalid and Invalid --> Invalid
```

Appendix C

PSTN Switches Monitoring Subsystem

During the operation process, if any fault or abnormality happens to certain parts, the monitoring subsystem will detect them in real time and inform the fault-processing program to perform the required operations. At the same time, the subsystem will generate various alarm information and signals. The power alarm status and the critical & major alarm status of each module are connected to the alarm console. The alarm console, in turn, gives out an audible and visual alarm prompt. This prompts the operator to handle the abnormality immediately.

Alarms are sent to the alarm console in real time and are saved in the database for querying, statistics, printing (see figure C.1) this way, maintenance personnel can detect the abnormal status of the equipment and handle them in time according to the alarm level.

Figure C.1: Typical monitoring subsystem interaction with switch DBMS

The Qazvin OMC project aims to provide a uniform monitoring subsystem whiches fetches alarms from all switch in real time, prompts the related employees using predefined media such as portable alarm box, SMS[26], email. The subsystem archives alarms information for querying, statistics and so on.

[26]Short Message Service

Bibliography

[Abiteboul and O.Duschka, 1998] Abiteboul, S. and O.Duschka (1998). Complexity of answering queries using materialized views. *In On Principles of DatabaseSystems (PODS'98)*, pages 254–265. the 17th ACM SIGACT SIGMOD SIGART Symp.

[Adali *et al.*, 1996] Adali, S., Candan, K. S., Papakonstantinou, Y. and Subrahmanian, V. (1996). Query caching and optimization in distributed mediatorsystems. *In the ACM SIGMOD Int. Conf. on Management of Data*, pages 137–148.

[Ahmad *et al.*, 2009] Ahmad, H., Kermanshahani, S., Simonet, A. and Simonet, M. (2009). Materialized approach to the integration of xml documents. *In ICOSE 2009 (International Conference on Ontiological and Semantic Engineering), WASET confernece proceedings (ISSN 2070-3740)*, pages 187–198.

[Alasoud *et al.*, 2005] Alasoud, A., Haarslev, V. and Shiri, N. (2005). A hybrid approach for ontology integration. *In VLDB '05: Proceedings of the 31st VLDB Conference*, Trondheim, Norway.

[Algergawy *et al.*, 2007] Algergawy, A., Schallehn, E. and Saake, G. (2007). A unified schema matching framework. *In Grundlagen von Datenbanken*, pages 57–61.

[Amann *et al.*, 2002] Amann, B., Beeri, C., Fundulaki, I. and Scholl, M. (2002). Querying xml sources using an ontology-based mediator. *In On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, pages 429–448, London, UK. Springer-Verlag.

[Arens *et al.*, 1994] Arens, Y., Chee, C., nan Hsu, C., In, H. and Knoblock, C. A. (1994). Query processing in an information mediator. *In In Proceedings of the ARPA/Rome Lab.*

[Arens *et al.*, 1996] Arens, Y., Knoblock, C. A. and min Shen, W. (1996). Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6:99–130.

[Ashish *et al.*, 1999] Ashish, N., Knoblock, C. A. and Shahabi, C. (1999). Selectively materializing data in mediators by analyzing source structure, query distribution and maintenance cost. *In In Proc. 2nd Int. Workshop On Web Information and Data Management, Kansas City*, pages 33–37. ACM Press.

[Baril, 2006] Baril, X. (2006). *Incremental Method for XML View Maintenance in Case of Non Monitored Data Sources*, chapitre Lecture Notes in Computer Science, pages 148–157. Springer- Verlag Berlin Heidelberg.

[Bassolet, 1998] Bassolet, C. G. (1998). *Approches connexionnistes du classement en Osiris. Vers un classement probabiliste.* Phd thesis, Joseph Fourier - Grenoble1, France.

[Bayardo *et al.*, 1997] Bayardo, Jr., R. J., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A. and Woelk, D. (1997). Infosleuth: agent-based semantic integration of information in open and dynamic environments. *SIGMOD Rec.*, 26(2):195–206.

[Beneventano and Bergamaschi, 2004] Beneventano, D. and Bergamaschi, S. (2004). The momis methodology for integrating heterogeneous data sources. *In 18 th IFIP World Computer Congress.* Kluwer.

[Beneventano *et al.*, 2000] Beneventano, D., Bergamaschi, S., Castano, S., Corni, A., R.Guidetti, Malvezzi, G., Melchiori, M. and Vincini, M. (2000). Information integration:the momis project demonstration. *In the 26th Int. Conf. on Very Large Data Bases (VLDB 2000).*

[Bernstein *et al.*, 2002] Bernstein, P. A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L. and Zaihrayeu, I. (2002). Data management for peer-to-peer computing : A vision. *In WebDB*, pages 89–94.

[Bertino and Guerrini, 1995] Bertino, E. and Guerrini, G. (1995). Objects with multiple most specic classes. *ECOOP '95 : Proceeding of the 9th European Conference on Object-Oriented Programming.*

[Blakeley *et al.*, 1986] Blakeley, J. A., Larson, P.-A. and Tompa, F. W. (1986). Efficiently updating materialized views. *SIGMOD Rec.*, 15(2):61–71.

[Borgida, 1995] Borgida, A. (1995). Description logics in data management. *In IEEE Trans. on Knowledge and Data Engineering*, volume 7(5), pages 671–682.

[Bouzeghoub and Lenzerini, 2001] Bouzeghoub, M. and Lenzerini, M. (2001). Introduction tothe special issue on data extraction, cleaning and reconciliation. *In Information Systems*, volume 26(8), pages 535–536.

[Boyd *et al.*, 2004] Boyd, M., Kittivoravitkul, S., Lazanitis, C., McBrien, P. and Rizopoulos, N. (2004). Automed: A bav data integration system for heterogeneous data sources. *In CAiSE*, pages 82–97.

[Bruni *et al.*, 2003] Bruni, P., Arnaudies, F., Bennett, A., Englert, S. and Keplinger, G. (2003). *Data federation with ibm db2 information integrator v8.1.* IBM Corp., Riverton, NJ, USA.

[Calì *et al.*, 2002] Calì, A., Calvanese, D., Giacomo, G. D., and M.Lenzerini (2002). Data integration under integrity constraints. *In the 14th Conf. on Advanced Information Systems Engineering (CAiSE 2002).*

[Calì *et al.*, 2004] Calì, A., Calvanese, D., Giacomo, G. D. and Lenzerini, M. (2004). Data integration under integrity constraints. *Inf. Syst.*, 29(2):147–163.

[Calì *et al.*, 2003] Calì, A., Calvanese, D., Giacomo, G. D., Lenzerini, M., Naggar, P. and Vernacotola, F. (2003). Ibis: Semantic data integration at work. *In CAiSE*, pages 79–94.

[Calì *et al.*, 2001] Calì, A., Giacomo, G. D. and Lenzerini, M. (2001). Models of information integration: Turning local-as-view into global-as-view. *In Foundations of Models for Information Integration. On line proceedings,http://www.fmldo.org/FMII-2001.*

[Calvanese *et al.*, 2000] Calvanese, D., Giacomo, G. D. and Lenzerini, M. (2000). Answering queries using views over description logics knowledge bases. *In the17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 386–391.

[Calvanese *et al.*, 1998] Calvanese, D., Giacomo, G. D., Lenzerini, M., D.Nardi and Rosati, R. (1998). Description logic framework for information integration. *In the 6th Int. Conf. On Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13.

[Calvanese *et al.*, 2004] Calvanese, D., Giacomo, G. D., Lenzerini, M. and Rosati, R. (2004). Logical foundations of peer-to-peer data integration. *In PODS*, pages 241–251.

[Carey *et al.*, 1995] Carey, M. J., Haas, L. M., Schwarz, P. M., Arya, M., Cody, W., Fagin, R., Flickner, M., Luniewski, A., Niblack, W., Petkovic, D., J.Thomas, Williams, J. H. and Wimmers, E. L. (1995). Towards heterogeneous multimediainformation systems: The garlic approach. *In the 5th Int. Workshop on Research Issues in Data Engineering – Distributed Object Management (RIDE-DOM'95)*, pages 124–131. IEEE Computer Society Press.

[Catarci and Lenzerini, 1993] Catarci, T. and Lenzerini, M. (1993). Representing and using inter-schema knowledge in cooperative information systems. *In J. of Intelligent and Cooperative Information Systems*, volume 2(4), pages 375–398.

[Cattell *et al.*, 1997] Cattell, R.-G., Barry, D.-K., Bartels, D., Berler, M., Eastman, J., Gamerman, S., Jordan, D., Springer, A., Strickland, H. and Wade, D. (1997). *The object database standard : ODMG 2.0.* Morgan Kaufmann.

[Chatalic *et al.*, 2006] Chatalic, P., Nguyen, G. H. and Rousset, M.-C. (2006). Reasoning with inconsistencies in propositional peer-to-peer inference systems. *In ECAI*, pages 352–356.

[Chawathe *et al.*, 1994] Chawathe, S. S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. D. and Widom, J. (1994). The tsimmis project: Integration of heterogeneous information sources. *In IPSJ*, pages 7–18.

[Chekuri and Rajaraman, 2000] Chekuri, C. and Rajaraman, A. (2000). Conjunctive query containment revisited. *Theor. Comput. Sci.*, 239(2):211–229.

[Christine and Stefano, 1996] Christine, P. and Stefano, S. (1996). Intégration de bases de données : Panorama des problems et des approaches. École Polytechnique Fédérale de Lausanne.

[Codd, 1993] Codd, E. . A. (1993). Providing olap (on-line analytical processing) to user-analysts: An it mandate. *In White Paper, Commissioned by Arbor Software (now Hyperion Solutions).*

[Cohen, 1998] Cohen, W. W. (1998). Integration of heterogeneousdatabases without common domains using queries based on textual similarity. *In the ACM SIGMOD Int. Conf. on Management of Data*, pages 201–212.

[der Meyden, 1998] der Meyden, R. V. (1998). Logical approaches to incomplete information. *Logics for Databases and Information Systems, Kluwer Academic Publisher*, pages 307–356.

[Diallo, 2006] Diallo, G. (2006). *Une Architecture a Base d'Ontologies pour la Gestion Unifiee des Donnees Structurees et non Structurees.* Phd thesis, l'Universite Joseph Fourier.

[Doan and Halevy, 2002] Doan, A. and Halevy, A. (2002). Efficiently ordering query plans for data integration. *In In Proc. of ICDE.*

[Dou and LePendu, 2006] Dou, D. and LePendu, P. (2006). Ontology-based integration for relational databases. *In SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 461–466, New York, NY, USA. ACM.

[Duschka, 1997] Duschka, O. (1997). *Query Planning and Optimization inInformation Integration.* Phd thesis, Stanford University.

[Ejaz and Revett, 2004] Ejaz, A. and Revett, K. (2004). Utilizing staging tables in data integration to load data into materialized views. *In CIS*, pages 685–691.

[Elmagarmid and Sheth, 1999] Elmagarmid, A., M. R. and Sheth, A. (1999). *Management of Heterogeneous and Autonomous Database Systems.* Morgan Kaufmann Publishers, 3rd édition.

[Fagin *et al.*, 2003] Fagin, R., Kolaitis, P., Miller, R. and Popa, L. (2003). Data exchange: Semantics and query answering. *In ICDT.*

[Fernández *et al.*, 1998] Fernández, M. F., Florescu, D., Levy, A. Y. and Suciu, D. (1998). Warehousing and incremental evaluation for web site management. *In BDA.*

[Friedman *et al.*, 1999] Friedman, M., Levy, A. and Millstein, T. (1999). Navigationalplans for data integration. *In the 16th Nat. Conf. on Artificial Intelligence (AAAI'99)*, pages 67–73. AAAI Press/The MIT Press.

[Garcia-Molina *et al.*, 1998] Garcia-Molina, H., Labio, W. and Yang, J. (1998). Expiring data in a warehouse. *In VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 500–511, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Garcia-Molina *et al.*, 1997] Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J. D., Vassalos, V. and Widom, J. (1997). The tsimmis approach to mediation: Data models and languages. *J. Intell. Inf. Syst.*, 8(2):117–132.

[Gardarin *et al.*, 2006] Gardarin, G., Dragan, F. and Yeh, L. (2006). P2p semantic mediation of web sources. *In ICEIS (1)*, pages 7–15.

[Gay, 1999] Gay, E. (Juin 1999). *Vues Osiris sur une Base Relationnelle.* Mémoire dingénieur cnam, p-89, CNAM, Centre de Grenoble, France.

[Giacomo *et al.*, 2007] Giacomo, G. D., Lembo, D., Lenzerini, M. and Rosati, R. (2007). On reconciling data exchange, data integration, and peer data management. *In PODS*, pages 133–142.

[G.Laperrousaz, 2000] G.Laperrousaz (Juin 2000). *Etude et Mise en place dun Environnement pour la Conception et lInterrogation dun Data Warehouse Osiris.* Mémoire dingénieur cnam, p-89, CNAM, Centre de Grenoble, France.

[Goasdoué *et al.*, 2000] Goasdoué, F., Lattès, V. and Rousset, M.-C. (2000). The use of carin language and algorithms for information integration: The picsel system. *International Journal of Cooperative Information Systems (IJCIS)*, 9(4):383–401.

[Goh *et al.*, 1997] Goh, C. H., Bressan, S., Madnick, S. E. and D.Siegel, M. (1997). Context interchange: New features and formalisms for the intelligentintegration of information. *In ACM Trans. on Information Systems*, volume 17(3), pages 270–293.

[Gruber and Gruber, 1993] Gruber, T. R. and Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220.

[Gruninger and Lee, 2002] Gruninger, M. and Lee, J. (2002). Ontology applications and design. *In Communications of the ACM*, volume 45(2), pages 39–41. ACM.

[Gryz, 1998] Gryz, J. (1998). Query folding with inclusion dependencies. *In the 14th IEEE Int. Conf. on Data Engineering (ICDE'98)*, pages 126–133.

[Gupta, 1999] Gupta, A. (1999). *Materialized Views : Techniques, Implementations, and Applications.* MIT Press.

[Gupta *et al.*, 1996] Gupta, A., Jagadish, H. V. and Mumick, I. S. (1996). Data integration using self-maintainable views. *In In Proceedings of International Conference on Extending Database Technology (EDBT*, pages 140–144.

[Gupta and Mumick, 1995] Gupta, A. and Mumick, I. S. (1995). Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Engineering*, 18(2):3–18.

[Haas, 2007] Haas, L. M. (2007). Beauty and the beast: The theory and practice of information integration. *In ICDT*, pages 28–43.

[Haas *et al.*, 2002] Haas, L. M., Lin, E. T. and Roth, M. A. (2002). Data integration through database federation. *IBM Syst. J.*, 41(4):578–596.

[Hacid and Reynaud, 2006] Hacid, M.-S. and Reynaud, C. (2006). Lintegration de sources de donnees. *i3: Information Interaction Intelligence, Web Semantique*, 7–1.

[Halevy *et al.*, 2006] Halevy, A., Rajaraman, A. and Ordille, J. (2006). Data integration: the teenage years. *In VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 9–16. VLDB Endowment.

[Halevy, 2001] Halevy, A. Y. (2001). Answering queries using views: A survey. *Very Large Database*, 10(4):270–294.

[Halevy *et al.*, 2005] Halevy, A. Y., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A. and Sikka, V. (2005). Enterprise information integration: successes, challenges and controversies. *In SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 778–787, New York, NY, USA. ACM.

[Halevy *et al.*, 2003] Halevy, A. Y., Ives, Z. G., Suciu, D. and Tatarinov, I. (2003). Schema mediation in peer data management systems. *In ICDE*, pages 505–.

[Hammer *et al.*, 1994] Hammer, J., McLeod, D. and Si, A. (1994). An intelligent system for identifying and integrating non-local objects in federated database systems. *In HICSS (3)*, pages 398–407.

[Heimbigner and Mcleod, 1985] Heimbigner, D. and Mcleod, D. (1985). A federated architecture for information management. *ACM Transactions on Office Information Systems*, 3:253–278.

[Hsu and Knoblock, 2000] Hsu, C.-N. and Knoblock, C. A. (2000). Semantic query optimization for query plans of heterogeneous multidatabase systems. *IEEE Trans. on Knowl. and Data Eng.*, 12(6):959–978.

[Huang *et al.*, 2007] Huang, S.-M., Chou, T.-H. and Seng, J.-L. (2007). Data warehouse enhancement: A semantic cube model approach. *Inf. Sci.*, 177(11):2238–2254.

[Hull, 1997] Hull, R. (1997). Managing semantic heterogeneity in databases: a theoretical prospective. *In PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 51–61, New York, NY, USA. ACM.

[Hull and Zhou, 1996] Hull, R. and Zhou, G. (1996). A framework for supporting data integration using the materialized and virtual approaches. pages 481–492.

[Inmon, 1996] Inmon, W. H. (1996). *Building the Data Warehouse.* John Wiley and Sons, 1st édition.

[Inmon, 2005] Inmon, W. H. (2005). *Building the Data Warehouse.* Hungry Minds Inc, 4rd édition.

[Ives, 2002] Ives, Z. G. (2002). *Efficient query processing for data integration.* Phd Thesis. Chair-Halevy,, Alon.

[Ives *et al.*, 2004] Ives, Z. G., Halevy, A. Y. and Weld, D. S. (2004). Adapting to source properties in processing data integration queries. *In SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 395–406, New York, NY, USA. ACM.

[Jarke, 2003] Jarke, M., L. M. V. Y. V. P. (2003). *Fundamentals of Data Warehouses.* Springer, ISBN: 978-3-540-42089-7, 2nd édition.

[Jasper *et al.*, 2003] Jasper, E., Tong, N., McBrien, P. and Poulovassilis, A. (2003). View generation and optimisation in the automed data integration framework.

[Josifovski, 1999] Josifovski, V. (1999). *Design, Implementation and Evaluation of a Distributed Mediator System for Data Integration.* Phd thesis, Linköpings universitet.

[Józefowska *et al.*, 2008] Józefowska, J., Lawrynowicz, A. and Lukaszewski, T. (2008). On reducing redundancy in mining relational association rules from the semantic web. *In RR*, pages 205–213.

[Kalfoglou and Schorlemmer, 2003] Kalfoglou, Y. and Schorlemmer, M. (2003). Ontology mapping: The state of the art. *The Knowledge Engineering Review*, 18:2003.

[Kanne and Moerkotte, 2000] Kanne, C.-C. and Moerkotte, G. (2000). Efficient storage of xml data. *In ICDE*, page 198.

[Kashyap and Sheth, 1996] Kashyap, V. and Sheth, A. (1996). Semantic heterogeneity in global information systems: The role of metadata, context and ontologies. *Cooperative Information Systems: Current Trends and Directions*, pages 139–178.

[Kermanshahani, 2008] Kermanshahani, S. (2008). Semi-materialized framework: a hybrid approach to data integration. *In CSTST '08: Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology*, pages 600–606, New York, NY, USA. ACM.

[Kermanshahani *et al.*, 2007] Kermanshahani, S., Ahmad, H., Simonet, A. and Simonet, M. (2007). A semantic view-based multi-mediator architecture. *In Proceedings of the 2007 International Conference on Information & Knowledge Engineering, IKE 2007, June 25-28, 2007, Las Vegas, Nevada, USA*, pages 197–204.

[Kermanshahani *et al.*, 2008] Kermanshahani, S., Ahmad, H., Simonet, A. and Simonet, M. (2008). A view based architecture for a multi-level semantic mediator. *In IKE*.

[Kim and Seo, 1991] Kim, W. and Seo, J. (1991). Classifying schematic and data heterogeneity in multidatabase systems. *Computer*, 24(12):12–18.

[Kimball and Ross, 2002] Kimball, R. and Ross, M. (2002). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons Inc, 2nd édition.

[Kirk *et al.*, 1995] Kirk, T., Levy, A. Y., Sagiv, Y. and Srivastava, D. (1995). The information manifold. *In the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, pages 85–91.

[Koch,] Koch, C. Data integration against multiple evolving autonomous schemata.

[Kolaitis, 2005] Kolaitis, P. G. (2005). Schema mappings, data exchange, and metadata management. *In PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 61–75, New York, NY, USA. ACM.

[Lattes and Rousset, 2000] Lattes, F. G. V. and Rousset, M.-C. (2000). The use of CARIN language and algorithms for information integration: The PICSEL system. *International Journal of Cooperative Information Systems*, 9(4):383–401.

[Lenzerini, 2002] Lenzerini, M. (2002). Data integration: A theoretical perspective. *In PODS*, pages 243–246.

[Leone *et al.*, 2005] Leone, N., Greco, G., Ianni, G., Lio, V., Terracina, G., Eiter, T., Faber, W., Fink, M., Gottlob, G., Rosati, R., Lembo, D., Lenzerini, M., Ruzzi, M., Kalka, E., Nowicki, B. and Staniszkis, W. (2005). The infomix system for advanced integration of incomplete and inconsistent data. *In SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 915–917, New York, NY, USA. ACM.

[Levy, 1996] Levy, A. Y. (1996). Obtaining complete answers from incomplete databases. *In the 22nd Int. Conf. on Very Large Data Bases(VLDB'96)*, pages 402—412.

[Levy *et al.*, 1996] Levy, A. Y., Rajaraman, A. and Ordille, J. J. (1996). Querying heterogeneous information sources using source descriptions. *In VLDB '96: Proceedings of the 22th*

International Conference on Very Large Data Bases, pages 251–262, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Levy and Rousset, 1995] Levy, A. Y. and Rousset, M.-C. (1995). Combining rules and description logics: An overview of carin. *In ILPS*, page 635.

[Levy *et al.*, 1995] Levy, A. Y., Srivastava, D. and Kirk, T. (1995). Data model and query evaluation in global information systems. *J. Intell. Inf. Syst.*, 5(2):121–143.

[Leymann and Roller, 2002] Leymann, F. and Roller, D. (2002). Using flows in information integration. *IBM Syst. J.*, 41(4):732–742.

[Liu, 2004] Liu, T. W. D. Y. S. T. Y. (2004). Discovering and generating materialized xml views in data integration system. *In Database Engineering and Applications Symposium, 2004. IDEAS apos;04. Proceedings. International*, pages 395–400.

[MacGregor, 1991] MacGregor, R. M. (1991). Inside the loom description classifier. *SIGART Bull.*, 2(3):88–92.

[Manolescu *et al.*, 2001] Manolescu, I., Florescu, D. and Kossmann, D. (2001). Answering xml queries over heterogeneous data sources. *In the 27th Int. Conf. on Very Large Data Bases (VLDB 2001)*, pages 241–250.

[Manolescu *et al.*, 2000] Manolescu, I., Florescu, D., Kossmann, D., Xhumari, F. and Olteanu, D. (2000). Agora: Living with xml and relational. *In In Proceedings of International Conference on Very Large Databases (VLDB*, pages 623–626. Morgan Kaufmann.

[McBrien and Poulovassilis, 2002] McBrien, P. and Poulovassilis, A. (2002). Schema evolution in heterogeneous database architectures, a schema transformation approach. *In CAiSE '02: Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, pages 484–499, London, UK. Springer-Verlag.

[McBrien and Poulovassilis, 2003] McBrien, P. and Poulovassilis, A. (2003). Data integration by bi-directional schema transformation rules. *In ICDE*, pages 227–238.

[McHugh *et al.*, 1997] McHugh, J., Abiteboul, S., Goldman, R., Quass, D. and Widom, J. (1997). Lore: a database management system for semistructured data. *SIGMOD Rec.*, 26(3):54–66.

[McHugh and Widom, 1997] McHugh, J. and Widom, J. (1997). Integrating dynamically-fetched external information into a dbms for semistructured data. *SIGMOD Rec.*, 26(4):24–31.

[McLeod and Heimbigner, 1980] McLeod, D. and Heimbigner, D. (1980). A federated architecture for database systems. *In AFIPS '80: Proceedings of the May 19-22, 1980, national computer conference*, pages 283–289, New York, NY, USA. ACM.

[Mena *et al.*, 1996] Mena, E., Kashyap, V., Sheth, A. P. and Illarramendi, A. (1996). OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *In Conference on Cooperative Information Systems*, pages 14–25.

[Naumann *et al.*, 1999] Naumann, F., Leser, U. and Freytag, J. C. (1999). Quality-driven integration of heterogenous information systems. *In The 25th Int.Conf. on Very Large Data Bases (VLDB'99)*, volume 2(4), pages 447–458.

[Ndiaye *et al.*, 2001] Ndiaye, Y., Diene, A., Litwin, W. and Risch, T. (2001). Amos-sdds: A scalable distributed data manager for windows multicomputers.

[Noy, 2004] Noy, N. F. (2004). Semantic integration: a survey of ontology-based approaches. *SIGMOD Rec.*, 33(4):65–70.

[Papakonstantinou *et al.*, 1995] Papakonstantinou, Y., Garcia-Molina, H. and Widom, J. (1995). Object exchange across heterogeneous information sources. *In ICDE*, pages 251–260.

[Poggi, 2006] Poggi, A. (2006). *Structured and Semi-Structured Data Integration.* Phd thesis, Université PARIS SUD, France.

[Poggi *et al.*, 2008] Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M. and Rosati, R. (2008). Linking data to ontologies. *J. on Data Semantics*, X:133–173.

[Poggi and Ruzzi, 2004] Poggi, A. and Ruzzi, M. (2004). Filling the gap between data federation and data integration. *In SEBD*, pages 270–281.

[Quang, 2005] Quang, H. L. (2005). Integration of web data sources: A survey of existing problems. *In* Brass, C. G. S., éditeur : *17. GI-Workshop über Grundlagen von Datenbanken (17th GI-Workshop on the Foundations of Databases), Wörlitz, 17. - 20. Mai 2005*, pages 78–82. Institute of Computer Science, Martin-Luther-University Halle-Wittenberg.

[Rahm and Bernstein, 2001] Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350.

[Risch *et al.*, 2003] Risch, T., JOSIFOVSKI, V. and KATCHAOUNOV, T. (2003). Functional data integration in a distributed mediator system.

[Roger *et al.*, 2002] Roger, M., Simonet, A. and Simonet, M. (2002). Bringing together description logics and database in an object oriented model. *In DEXA '02: Proceedings of the 13th International Conference on Database and Expert Systems Applications*, pages 504–513, London, UK. Springer-Verlag.

[Roscheisen *et al.*, 2000] Roscheisen, M., Baldonado, M., Chang, K., Gravano, L., Ketchpel, S. and Paepcke, A. (2000). The stanford infobus and its service layers: Augmenting the internet with higher-level information management protocols. Technical Report 2000-42, Stanford InfoLab. Previous number = SIDL-WP-1997-0066.

[Rousset, 1998] Rousset, M.-C. (1998). The use of carin language and algorithms for information integration. *In the PICSEL project, Intelligent Information Integration Workshop associated with ECAI98 Conference.*

[Rousset and Reynaud, 2003] Rousset, M.-C. and Reynaud, C. (2003). Picsel and xyleme: Two illustrative information integration agents. *In AgentLink*, pages 50–78.

[Rundensteiner, 1992] Rundensteiner, E. A. (1992). Multiview: A methodology for supporting multiple views in object-oriented databases. *In VLDB*, pages 187–198.

[Saïs, 2007] Saïs, F. (2007). *Intégration Sémantique de Données guidé par une Ontologie.* Phd thesis, Université de PARIS SUD, France.

[Scholl *et al.*, 1991] Scholl, M., Laasch, C. and Tresch, M. (1991). Updatable views in objectoriented databases. *In Proc.2nd DOOD conf*, pages 187–198.

[Sheth and Kashyap, 1992] Sheth, A. and Kashyap, V. (1992). So far (schematically) yet so near (semantically). pages 283–312. North-Holland.

[Sheth and Larson, 1990] Sheth, A. P. and Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 1(3):173 – 236.

[Simonet, 1984] Simonet, A. (1984). *Types Abstraits et Bases de Donnees : formalisation du concept de partage et analyse statique de contraintes d'integrite.* Phd thesis, Universite Scientique et Medicale de Grenoble, France.

[Simonet, 1996] Simonet, A. (1996). Classement d'objets et evaluation de requetes en osiris. *In Bases de Donnees Avancees*, pages 273–288, Cassis, France.

[Solar and Doucet, 2002] Solar, G. V. and Doucet, A. (2002). Médiation de données : solutions et problèmes ouverts. *In Actes des 2èmes Assises nationales du GdR I3*, Nancy, France.

[Stanat and McAllister, 1997] Stanat, D. and McAllister (1997). Discrete mathematics in computer science. Technical report.

[Ullman, 1997] Ullman, J. D. (1997). Information integration using logical views. *the 6th Int. Conf. on Database Theory (ICDT'97), Springer*, 1186:19–40.

[Wache *et al.*, 2001] Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. and Hübner, S. (2001). Ontology-based integration of information — a survey of existing approaches. *In* Stuckenschmidt, H., éditeur : *IJCAI–01 Workshop: Ontologies and Information Sharing*, pages 108–117.

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

[Wiederhold, 1998] Wiederhold, G. (1998). Mediators in the architecture of future information systems. pages 185–196.

[Wu and Buchmann, 1997] Wu, M.-C. and Buchmann, A. P. (1997). Research issues in data warehousing. *In Datenbanksysteme in Buro, Technik und Wissenschaft*, pages 61–82.

[Xu and Embley,] Xu, L. and Embley, D. W. Combining the best of global-as-view and local-as-view for data integration. *In In Information Systems Technologies and Its Applications - ISTA*, pages 123–136.

[Xyleme, 2001] Xyleme, L. (2001). A dynamic warehouse for xml data of the web. *IEEE Data Engineering Bulletin*, 24:40–47.

[Zhou *et al.*, 1996] Zhou, G., Hull, R. and King, R. (1996). Generating data integration mediators that use materialization. *J. Intell. Inf. Syst.*, 6(2-3):199–221.

[Zhou *et al.*, 1995] Zhou, G., Hull, R., King, R. and Franchitti, J.-C. (1995). Using object matching and materialization to integrate heterogeneous databases. *In the 3rd Int. Conf. on Cooperative Information Systems (CoopIS'95)*, pages 4–18.

[Zhu, 1999] Zhu, Y. (1999). A framework for warehousing the web contents. *In ICSC '99: Proceedings of the 5th International Computer Science Conference on Internet Applications*, pages 83–92, London, UK. Springer-Verlag.

Abstract

There is a large and increasing volume of documents, data sources and data base management systems available in the world, and many autonomous and heterogeneous sources speak of a same reality while using different words and conceptual structures. Many organizations need to dispose of a system that handles such data in a homogeneous way, which necessitates the integration of these data sources.

The goal of a data integration system is to develop a homogeneous interface for the end users to query several heterogeneous and autonomous sources. Building such a homogeneous interface raises many challenges among which the heterogeneity of data sources, the fragmentation of data, the processing and optimization of queries appear to be the most important.

There are many research projects that present different approaches and each of them proposes a solution to each of these problems. Depending on the integrated view, these approaches can be categorized into two main categories: materialized and virtual approaches; there are also some hybrid approaches when there is a composition of materialized and virtual views. The main advantage of a hybrid approach is to offer a trade-off between the query response time and data freshness in a data integration system. In the existing approaches, query optimization is often privileged for the materialized part of the system.

In this thesis, we develop a hybrid approach which aims to extend query optimization to all the queries of the integration system. It also provides a flexible data refreshing mechanism in order to tolerate different characteristics of sources and their data. This approach is based on the Osiris object indexing system. Osiris is a database and knowledge base platform with a specific object data model based on a hierarchy of views. Its indexation system relies on the partitioning of the object space using the view constraints.

IXIA, the hybrid approach presented in this thesis, materializes the indexation structure of the underlying objects at the mediator level. The Oids of objects, their correspondence with the source objects and the needed data to refresh the indexation data are also materialized.

Our index-based data integration approach offers more flexibility in data refreshing than a fully materialized approach and a better query response time in comparison with a fully virtual data integration system.

Keywords: Data Integration, Heterogeneity, Mediator, Data warehouse, Hybrid approach, Views.

Résumé

Aujourd'hui, il existe un nombre important et croissant de sources de données, qui peuvent être des documents et des données structurées ou semi-structurées. En général, aussi bien les documents que les bases de données sont autonomes et gérés par des systèmes différents. D'autre part, beaucoup de ces sources sont reliées tout en étant sémantiquement hétérogènes : elles modélisent la même réalité externe tout en utilisant des concepts et des structures distincts. Or, les organisations et les entreprises qui sont confrontées à de telles sources de données ont besoin d'en avoir une vision homogène et cohérente. La conséquence est qu'il est nécessaire de les intégrer et de disposer d'un système qui gère ces données.

L'objectif d'un système d'intégration des données est de proposer une interface homogène pour interroger plusieurs sources, qui peuvent être hétérogènes et autonomes. Derrière une telle interface il y a plusieurs défis, parmi lesquels nous soulignons l'hétérogénéité structurelle et sémantique des sources de données, la fragmentation des données, le traitement et l'optimisation des requêtes.

Il existe de nombreux travaux qui présentent des approches distinctes pour l'intégration des données, et chaque approche propose des solutions spécifiques à chacun des problèmes évoqués. On considère en général que ces approches appartiennent à deux grandes catégories : l'approche matérialisée et l'approche virtuelle. Cependant, on peut aussi considérer une troisième approche, dite hybride, qui propose qu'une partie des données du système intégré soit matérialisée et qu'une autre partie des données soit virtuelle.

Dans cette thèse, nous proposons une architecture hybride pour un système d'intégration de sources de données hétérogènes, qui vise à étendre l'optimisation des requêtes à toutes les requêtes du système d'intégration. Elle permet aussi de fournir un mécanisme flexible pour traiter la mise à jour des données afin de tolérer les différentes caractéristiques des sources et de leurs données.

Cette approche est basée sur un système d'indexation d'objets multicritères au niveau de la médiation. Dans notre approche, nous utilisons le système Osiris et son mécanisme d'indexation. Osiris est un système de gestion de bases de données et de bases de connaissance orienté objet, où une famille d'objets est définie par une hiérarchie de vues " object-preserving ". Le système d'indexation d'Osiris est un système multi-attributs, et notre approche propose la matérialisation du sous-ensemble des données directement reliées aux attributs d'indexation.

Le système d'intégration des données proposé, IXIA, matérialise la structure d'indexation des objets sous-jacents au niveau du médiateur. Les Oids des objets, leur correspondance avec les objets des sources et les données nécessaires pour la mise à jour de l'indexation des données sont aussi matérialisées.

Cette approche offre une plus grande flexibilité de rafraîchissement des données qu'une approche entièrement matérialisée, et une meilleure optimisation des requêtes que les méthodes entièrement virtuelles.

Mots-clés: Intégration des données, Entrepôt de données, Hétérogénéité, Médiateur, Approches Hybrides, Vues.