

Information Integration among Heterogeneous and Autonomous Applications

Ammar Benabdelkader

Copyright ©2002 by Ammar Benabdelkader.

All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without written permission from the author.

ISBN 90-5776-093-2
Febodruk Enschede.

Information Integration among Heterogeneous and Autonomous Applications

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam,
op gezag van de Rector Magnificus
prof. P. F. van der Heijden
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Aula der Universiteit
op Dinsdag 12 November 2002 te 10.00 uur

door

Ammar Benabdelkader

geboren te Bir El Arch, Sétif, Algeria

Promotiecommissie:

Promotor: Prof. dr. L.O. Hertzberger

Co-promotor: Dr. H. Afsarmanesh

Overige Leden:

Prof. dr. P.W. Adriaans
Prof. dr. R.J. Meijer
Prof. dr. A.P.J.M. Siebes
Prof. dr. L.M. Camarinha-Matos
Dr. ir. Paul W.P.J Grefen

Faculteit:

Natuurwetenschappen, Wiskunde & Informatica
Kruislaan 403
1098 SJ Amsterdam
Nederland

The research described in this thesis was partially supported by the European Commission under ESPRIT project-22186 Waternet, the HPCN MegaStore project, and the Dutch ICES/KIS program under project-1544516 Virtual Laboratory.

Contents

Acknowledgments	xii
1 Introduction	1
1.1 Major Requirements in terms of Information Management	3
1.2 Application Cases: an Overview	4
1.3 Thesis Contribution	7
1.4 Organization of the thesis	8
2 Information Integration Approaches, Mechanisms, and Tools	11
2.1 Introduction	11
2.2 A Taxonomy for Information Integration	12
2.2.1 Distributed Systems	14
2.2.2 Integrated Systems	17
2.3 Further Classifications and Categorizations	28
2.4 Discussion	29
3 WATERNET: Intelligent Supervision and Control in Heterogeneous and Distributed Application	31
3.1 Introduction	32
3.2 Water Environment and General application requirements	33
3.2.1 Water Network Structure and Management	35
3.3 Information Management Approach	38
3.3.1 The Waternet Architecture	38
3.3.2 Simple Scenario for Subsystems interaction	39
3.4 Distributed Information Management System (DIMS)	40
3.4.1 The PEER Federated Layer	41
3.4.2 Schemas Management in WATERNET Using PEER	42
3.5 Extended Integration Approach	45
3.5.1 Data Adapters Supporting Openness	47
3.5.2 The WATERNET System Implementation	48
3.6 Conclusion and Discussion	48
3.6.1 Major Characteristics and Benefits of Federated Approach in Waternet	49
3.6.2 Contribution to GFI ₂ S	49

4	MegaStore: Advanced Web Databases for Music Industry	51
4.1	Introduction	51
4.1.1	E-Commerce Applications: Attempts and Aims	52
4.2	Problem Analysis and Required High Level Architecture	52
4.2.1	Database Design	53
4.2.2	ODL Schema definition	55
4.3	The MegaStore System Architecture	57
4.3.1	The Internet-Shop Interface	58
4.3.2	The Shop-in-a-Shop Interface	58
4.3.3	Server Architecture Extension	59
4.4	Music Audio and Video content	60
4.4.1	Bandwidth and Encoding Algorithm	60
4.4.2	Data Volume Estimation	61
4.5	Music Data Manipulation	62
4.5.1	Objects Loading Strategies	62
4.5.2	Extensions	65
4.5.3	Database Administration	66
4.6	MegaStore Interfaces - Advanced Features	67
4.6.1	Dynamic Browsing	69
4.6.2	Ordering System	70
4.6.3	System Security	71
4.6.4	Current Implementation Status	72
4.7	Derived Applications	73
4.7.1	LuisterPaal Interface	73
4.7.2	Music Sheet Application	75
4.8	Conclusion and Discussion	78
4.8.1	Major Characteristics and Benefits provided to MegaStore Application	79
4.8.2	Contribution of the MegaStore's Information Management Approach to GFI ₂ S	80
5	Information Management for Scientific Applications	81
5.1	Introduction	81
5.2	Virtual Laboratory Architecture Design	82
5.2.1	The VL Information Management for COoperation - VIMCO Module	85
5.3	Multi-Media Scientific Data Sets Manipulation	85
5.3.1	Storage of Large Scientific and Engineering Data Sets	87
5.3.2	Scientific Data Archiving and Cataloguing Using Dublin Core Standard	93
5.4	Universal Database Access - Based on Standards	99
5.4.1	Database Connection Module	102
5.4.2	Query Execution Module	102
5.4.3	Results Presentation Module	102
5.4.4	Object Creation Module	103
5.4.5	Further Benefits	103
5.5	Data Access Security and Information Visibility (Safe/Reliable Data Export)	105
5.5.1	Role-based Access Control Definition	106
5.5.2	Flexible Role-based Access Interface	108
5.6	Physical Database Performance Analysis	109

5.6.1	Specific Functions to Access Binary Large Objects (Blobs)	110
5.6.2	Benchmarking Tests For Matisse Database System	110
5.6.3	Observations	112
5.6.4	Lessons Learned	112
5.7	Conclusion and Discussion	112
5.7.1	Contribution to GFI ₂ S	113
6	GFI₂S - Generic and Flexible Information Integration System	115
6.1	Introduction	115
6.1.1	Focus of GFI ₂ S	117
6.2	GFI ₂ S Information Integration Approach	119
6.2.1	Local Adaptation Layer (LAL)	122
6.2.2	Node Federation Layer (NFL)	125
6.2.3	Application of Database Standards and Middleware Solutions in GFI ₂ S	144
6.2.4	GFI ₂ S in Action	145
6.3	Conclusion	147
7	Conclusions and Future Work	149
7.1	Overview	149
7.2	GFI ₂ S Compared to Other Approaches	152
7.3	Lessons Learned	153
7.4	Future Work	154
A	Application of Database and Middleware Standards in FGI₂S	157
A.1	Object-Oriented Standards and Extensions Adaptation for GFI₂S	157
A.1.1	Object Definition Language – ODL	158
A.1.2	Query Languages – SQL, SQL3, and OQL	160
A.1.3	Object Interchange Format - OIF	161
A.2	Web Standard and Middleware Adaptation for GFI₂S	162
A.2.1	Object Database Connectivity - ODBC	163
A.2.2	Use of JAVA for Application Programming	164
A.2.3	Use of XML for Information Exchange	165
	Samenvatting	180
	Abstract	182
	Résumé	184

List of Figures

2.1	Information Integration Approaches - Classification	14
2.2	Distributed Database Architecture	15
2.3	Example of a Simple Database Model	15
2.4	Fragmentation in Distributed Database Systems	16
2.5	Data Distribution and Replication among Distributed Databases	17
2.6	Two Side Dependent Translation	19
2.7	Access Through the Common Data Model	20
2.8	Access to the Global Shared Database	21
2.9	Schemas Representation in PEER	24
2.10	The PRODNET Reference Architecture	25
2.11	General DIMS Architecture Approach	25
2.12	WebFINDIT Components are grouped in four Interactive Layers	27
3.1	Logical Units for the Waternet System architecture	34
3.2	Water Management Environment	36
3.3	Information Management Architecture for the Water Network in Terms of Units	39
3.4	Simple Scenario for Subsystems Interaction in Waternet	40
3.5	PEER Federated Layer Representation	41
3.6	Basic Integration Architecture	45
3.7	Extended Integration Architecture	46
3.8	DIMS Layer – Federated Data Process using Adapters	47
4.1	Base Schema Definition for the MegaStore System	55
4.2	MegaStore Server Architecture Description	57
4.3	Data Storage Mechanisms	63
4.4	Music Input - Format A	64
4.5	Music Input - Format B	64
4.6	Music Input - Format C	65
4.7	DBA Interface - OIF Loader	67
4.8	An Activity Diagram for the Internet-Shop Interface	68
4.9	Main MegaStore Interface	69
4.10	Album Songs Interface	70
4.11	State Diagram for Orders	70
4.12	Custom Order	71
4.13	Conceptual Model for an e-MegaStore Application	73
4.14	LuisterPaal User Interface	74
4.15	Database Model for the Music Sheet Application	76

4.16	Music Sheet User Interface	77
4.17	e-MegaStore System Architecture	79
5.1	Functional layers within the Virtual Laboratory Environment	83
5.2	File System Approach	88
5.3	External Data Link Approach	89
5.4	One-Database Storage Approach	91
5.5	Architecture for the Parallel/Distributed Database Server	92
5.6	Parallel/Distributed server architecture: an Application Case	93
5.7	An Object-oriented schema for the Dublin Core meta-data	96
5.8	Enhanced Object-oriented schema definition for the VL archiving environment based on the Dublin Core standard	97
5.9	Example simplified Data Model for Authors and Publications	100
5.10	Universal Database Access Interface	101
5.11	Schema Definition of the role-based Access Control with Export Views	106
5.12	Interface for Views Definition	107
5.13	Safe/Reliable Data Export Interface	108
5.14	Database performance when storing/retrieving large objects	111
6.1	The GF2IS Architecture following the Node-to-Node Federation	120
6.2	Communication Model among GF2IS components	121
6.3	Components of the Local Adaptation Layer	123
6.4	Node Federated Layer Representation	125
6.5	Schemas representation adopted at the node federation layer	128
6.6	Integrated Scenario for Systems Interoperation	130
6.7	Export Schema (Exp2) Definition and Derivation - Example	131
6.8	Export Schema (Exp2) Derivation - an example	132
6.9	Integrated Schema Definition and Derivation - Example	133
6.10	Integrated Schema Derivation – an example	133
6.11	Schema Definition and Derivation Specification - an example	141
6.12	Classes and Attributes Instantiation - an example	141
6.13	Federated Query Processor – The Steps	142
6.14	Federated Query Processor – Performing Mechanism	143
6.15	GFI₂S - Global Overview	145
6.16	Global Architecture and Interfaces to GFI ₂ S	146
A.1	Application Access to Remote Database via ODBC	163

List of Tables

1.1	List of Requirements of Today's and Forthcoming Applications	6
2.1	Commercial Systems Evaluation in Terms of Information Integration	23
2.2	Approaches Evaluation based on the Developed Systems	28
2.3	Approaches Evaluation based on the Application Requirements	29
3.1	Simple Network Local Schema in Control Unit Node.	42
3.2	Simple Network Export Schema (EXP1) in Control Unit Node	43
3.3	Simple Local Schema (LOC)	43
3.4	Simple Imported Schema (IMP7)	43
3.5	The Integrated (INT) Schema in Optimization	44
4.1	ODL Schema for the MegaStore Database	56
4.2	An OIF Example	66
5.1	Dublin Core: Elements Description	95
5.2	Enhanced ODL schema for the VL archiving environment based on the Dublin Core standard	98

List of Abbreviations

ADO: ActiveX Data Object
API: Application Programming Interface
BLOB: Binary Large Object
CDM: Common Data Model
CGI: Common Gateway Interface
COMCOL: Communication and Collaboration Layer
CORBA: Common Object Request Broker Architecture
DBA: Database Administrator
DBT: Data Browsing Tool
DC: Dublin Core
DCMI: Dublin Core Meta-data Initiative
DIMS: Distributed Information Management System
DLL: Dynamic Link Library
DTD: Data Type Definition
EC: Electronic Commerce
ER: Entity Relationship
EXP: Export Schema
FL : Federated Layer
FDBMS: Federated Database Management System
FRS: Federated Resources Specifications
FQP: Federated Query Processing
GFI2S: Generic and Flexile Information Integration System
IDL: Interface Definition Language
IIS: Internet Information Server
IMP: Import Schema
INT: Integrated Schema
IPR: Intellectual Propriety Rights
JDBC: Java DataBase Connectivity

LAL: Local Adaptation Layer
LAN: Local Area Network
LOC: Local Schema
LRS: Local Resources Specifications
MACS: Material Analysis for Complex Surfaces
NFL: Node Federation Layer
ODBC: Open Database Connectivity
ODBMS: Object Database Management Systems
ODL: Object Definition Language
ODMG: Object Database Management Group
OIF: Object Interchange Format
OLAP: On Line Analysis Processing
OLE-DB: Object Linking and Embedding for Databases
OMT: Object Modeling Technique
OQL: Object Query Language
ORDBMS: Object-Relational Database Management Systems
RDBMS: Relational Database Management Systems
RMI: Remote Method Invocation
SDDL: Schema Definition and Derivation Language
SMT: Schema Manipulation Tool
SQL: Structured Query Language
SQL/CLI: SQL Call-Level Interface
SQL/PSM: SQL Persistent, Stored Modules
STEP: Standard for the Exchange of Product Model Data
UML: Unified Modeling Language
UDBA: Universal DataBase Access Interface
VIMCO: Virtual-laboratory Information Management for COoperation
ViSE: Virtual Simulation and Exploration
VL: Virtual Laboratory
VLAM-G: Grid-based Virtual Laboratory Amsterdam
VL-e: Virtual Laboratory for Experimental Science
XML: eXtensible Markup Language
WAN: Wide Area Network

Acknowledgments

The work described in this dissertation is the result of more than five years of research performed at the Informatics Institute of the University of Amsterdam. Many people have supported my research effort amongst whom I would like to mention a few especially.

First of all, I would like to thank my promoter Bob Hertzberger and my co-promoter Hamideh Afsarmanesh for their supervision and for giving me the opportunity to become a member of the COoperataive Information Management (CO-IM) group. Without their confidence and support this thesis would not have reached its final stage. Their final reading and comments greatly improved the content and the readability of this thesis.

I am very grateful for the continuous support of the members of my group with whom I shared many professional and social activities. In particular, Adam Belloum, Anne Frankel, Ersin Cem Caletas, Victor Guevara, Mohand-Amokrane Abtroun, and Uzgul Unal. Special thank is given to my officemate Cesar Garita, who was my research companion during all my studies at the UvA. Many thanks to all other people at the institute with whom I have shared the work and expertise: Hakan Yakali, Arnoud Visser, Philip Jonkergouw, and Zeger Hendrikse. They always helped me keep up with the good work, and they supported me in the final preparation of the thesis document.

I also want to express my appreciation to the members of the evaluation committee for the time they dedicated to the promotion activities. I am sincerely grateful to all of them for their many constructive comments and the principal advisory role they played during the final writing process of this thesis.

I am also thankful to the partners of Waternet, MegaStore, and Virtual Laboratory projects. Without their collaboration, the application cases, which constitute an important part of this research would not have been so faithfully developed. Special thank if given to Ronald Schut, Manager of the PCC company, and to Pieter de Haan who participated in the final development of the MegaStore project.

During my stay in Amsterdam I have made many friendships at several places and for different periods of time, who have in one way or another influenced my life in Amsterdam and made it more enjoyable. Just to mention a few of them: Ahmed, Nikos Massios, Fouzi, Djomaa, Hiddad, Youssef, Rachid, and others. I thank you all for your after-hours company and for your true friendship and constant encouragement.

Furthermore, during the elaboration of this thesis, I received the assistance of many colleagues and friends from the Faculty of Science. I cannot mention all the names, however, I would like to mention the people of the secretariat, they were always very attentive and accessible to my requests. In particular, Jacqueline, Virginie, and Erik offered me a lot of help during the final stages of this thesis.

Finally, I want to thank my parents, my sister, and my brothers. They were always my greatest supporters in every way, they provided me with the necessary strength, and they motivated me to study from the very beginning. But most of all I would like to thank my wife Yasmina, she always encouraged me to continue and challenged me to go further with my research work.

Ammar Benabdelkader
Amsterdam, September 2002.

Chapter 1

Introduction

The design and development processes of advanced applications in scientific and system engineering domains consider different data modeling and information management strategies. Data models define the data structures and relationships among the data, to reflect the proper representation of the information each application needs. The information management strategies however, depend on the global architecture design and the chosen database system to fulfill the functionalities required by the application.

Diversity of the used information management approaches is usually due to different characteristics and requirements of each application. Due to the complex requirements of emerging applications, several scientific and business oriented organizations from biology, medicine, physics, astronomy, engineering, e-commerce, etc. have realized the need to reconsider their information management systems towards better addressing of collaborative work. Therefore, these organizations are required to provide appropriate products and services, and to better react to the new information management requirements in terms of data integration. Traditionally, information integration and data translation among different heterogeneous and autonomous sites were considered a completely manual process, where either the user or the database administrator must do the data translation and exchange. Nowadays, the problem of data integration and information exchange among heterogeneous data sources has become a challenging issue to be studied, and different integration approaches are being examined and evaluated.

This thesis addresses the issue of *information integration for systems interoperation* among heterogeneous and autonomous applications, and mainly addresses solutions related to the requirements for:

- ☞ *Data integration* from different sources distributed over a network of nodes.
- ☞ *Interoperation* and *information exchange* among a number of sites, which are heterogeneous, autonomous, and of distributed nature.
- ☞ *Methodology* design and *generic tools* development to support the information integration amongst a number of networked applications.

Within the different chapters of the thesis we propose *methodologies*, develop *standard tools for information access/exchange*, and validate *generic solutions* that serve the information management requirements. Generic solutions fit several applications emerging from various domains, and facilitate systems flexibility and configurability. The design and development of generic solutions, for information management and interoperation, is achieved

via the deployment of standards and middleware solutions during the different development phases of an application, which evolve from modeling and design, to development and validation. In addition, the proposed solutions for information integration and systems interoperation consider the combination of emerging advances in databases and Web technologies, and mainly deploy the related standard concepts for data modeling, data definition, information storage and retrieval, information exchange, multi-platform-programming environment, and Communication infrastructure. Considering the main characteristics defining advanced applications, described above, the structure of the thesis is motivated by the following facts:

- ① There is wide variety of emerging networked applications in the diverse domains of science, business, engineering, education, e-commerce, tourism, etc. Among the common characteristics of these applications, we can enumerate distribution diversity, site autonomy, and information heterogeneity. In addition, the use of different data modeling approaches, information management strategies, and database management systems complicate the interoperation among these applications.
- ② Based on the type of applications and the global objectives targeted by each of them, the information management requirements differ from one application to another.
- ③ To fulfill the information management requirements of these applications, several approaches have been proposed and developed, addressing the information integration problem. Most of these approaches are application specific, while attempts in the direction of using standard tools and generic solutions are quite a few.
- ④ In order to validate different approaches addressing specific requirements of these applications, it is necessary to design and develop some prototypes as proof of concepts in different applications.
- ⑤ Based on the studies and prototypes developed within the different projects addressed in this thesis, a 'challenging' generic and flexible approach is designed and presented. This approach benefits from previous methodologies and extends them in order to provide generic solutions that can be applied within a wide variety of applications.

The five major points enumerated above illustrate the reason for the structure of the thesis. The application cases, mentioned in point 4 above and addressed in chapters 3, 4, and 5 of this dissertation, provide the base for the Generic and Flexible Information Integration System (**GFI₂S**) addressed in chapter 6. Mainly, the **GFI₂S** system reflects the results of the lessons learned in three research projects, and thus, addresses the integration of data sources, and the interoperation among diverse, heterogeneous, and autonomous sites in a network.

This introductory chapter briefly emphasizes the main characteristics of emerging applications and outlines the major requirements of these applications in terms of information management and interoperation. Chapter 2 describes some of the existing related approaches, mechanisms, and tools for information integration. Three specific research projects from different application domains are described in chapters 3, 4, and 5 of this thesis. These chapters focus mostly on the study and analysis of the information management requirements for distributed and heterogeneous applications, as inspired by the advanced cooperative applications in the domains of water systems industry, e-commerce, and e-science. The E-Science¹ domain addressed in chapter 5 mostly focuses on how to

¹E-Science is about global collaboration in key areas of science, and the next generation of infrastructure that will enable it. *John Taylor, Director General of the Research Councils, OST.*

use scientific methods, and how to apply new software packages and web resources in the analysis and solutions of real application problems, and in specific for the experiments in research laboratories. Chapter 6 of the thesis addresses an open and flexible approach for information integration and systems interoperation (**GFI₂S**). First it outlines the main requirements for emerging applications, and then it defines a common integrated approach that can support many emerging applications from different application domains. The approach also applies object-oriented standards and middleware solutions in order to provide more generic utilities.

1.1 Major Requirements in terms of Information Management

This section outlines the major requirements for future advanced applications in terms of information management and systems interoperation. These requirements are mainly identified through the three application cases addressed in the thesis, in addition to some other research work and literature. A detailed description of these requirements is out of the scope for this introductory chapter, appropriate descriptions however, are given in different chapters of this dissertation. It is also necessary to mention that these requirements are identified to support the information sharing and the data integration among networked applications, while preserving their local autonomy, heterogeneity, and distribution.

Hereafter, we enumerate a list of the major requirements that need to be addressed when designing and developing appropriate information management strategies for advanced applications. In order to give a better understanding and a clear overview to the reader, these requirements are grouped into six main categories:

R₁: Information Integration for Systems interoperation

- Transparent access to data located at different sources, via on-line integrated views.
- Interoperation among different systems in term of information exchange and services.
- Information sharing within a large community of internal and external applications and users.
- Data integration from heterogeneous and distributed sources.

R₂: Security for access and visibility levels

- User authentication based on pre-defined access rights.
- Information visibility levels based on pre-defined import/export schemas.
- Separation between public and proprietary information.

R₃: User facilities

- Provide easy access to data independent of its internal format and structure.
- Support user friendly interfaces facilitating the exploration of information, characterized by its complex structure.

R₄: Use of standards and middleware solutions

- Universal accesses to data regardless the underlying database management system.
- Use of standards and middleware solutions for data modeling and information exchange.
- Support for multi-platform applications development.
- Scientific data classification and cataloguing via the deployment of emerging standards in the field such as Dublin Core for scientific data description and NetCDF² for array-oriented data representation.

R₅: System efficiency and effort Minimization

- System efficiency and performance for data manipulation.
- Short response time for on-line requests.
- High bandwidth for data transfer.
- Good strategies for data storage and data archives.
- Effort and cost minimization in both modeling and development phases.

R₆: Advanced features

- Support for new data types introduced within the scientific application domain.
- Support the management of large data sets.
- Combine databases and advanced web technologies.
- Combine object-oriented concepts and emerging standards.

1.2 Application Cases: an Overview

The application cases presented in chapters 3, 4, and 5 of this dissertation illustrate three examples of modern applications. These application cases are primarily addressed by their data modeling concepts and information management strategies, that are required to support each application domain. The choice of these three application cases have covered different domain criteria on data integration mechanisms. In addition, these applications constitute some of the research areas within the CO-IM³ group at the University of Amsterdam, targeting the COoperative Information Management among autonomous and heterogeneous applications. Furthermore, the diversity of these application domains have added more value to our research work by introducing various requirements and thus, adding new challenges. The three applications addressed in this dissertation include:

- Intelligent supervision and control in heterogeneous and distributed water environments (Waternet project),
- Interoperation and collaboration among large distributed databases for music industry and e-commerce applications (MegaStore project), and
- Scientific data archiving and cataloguing for e-science within the Virtual Laboratory environment (Virtual Laboratory project).

²Network Common Data Form

³CO-IM: COoperative Information Management group of the University of Amsterdam (<http://www.science.uva.nl/~netpeer>)

The study of these applications shows that even while addressing only the information management issue, the requirements are quite complex and specific for each application domain. Thus, the modeling constructs, the designed methodologies, and the used systems differ from one application to another. More precisely:

1. In water supply industries (Waternet) distinct functionalities required in this industry are supported by independent, heterogeneous, and autonomous subsystems. Each subsystem performs its specific activity, but their co-working and complex information exchange needs to be properly supported in order to assure a continuous supply, to meet the quality standards, to save energy, to optimize pipeline sizes, and to reduce wastes.
2. In music industry application (MegaStore), we address the design and development of advanced and efficient internet-based Electronic Commerce (E-Commerce) services to support necessary requirements for the buyers of different goods. In addition to the traditional user requirements for every application environment, the developed system properly addresses several efficiency, organization, and multimedia related issues. Among the addressed issues, we enumerate: the data catalogues and information classification, short response time for on-line requests, high system performance, and high data transfer rates.
3. In experimental life science application (Virtual Laboratory), the information management framework aims at developing digital libraries and toolkits to enable scientists and engineers to work on their problems via experimentation. Especially, the VL information management framework addresses some emerging issues related to the management of large multimedia scientific data, information integration from a variety of data sources, and collaborative developments in e-science environment.

As depicted in Table 1.1, a large list of requirements is addressed in each of these research projects. On one hand the three projects are characterized by needs from different application domains, thus the requirements differ slightly from one project to another. On the other hand, considering the fact that the projects are carried out in different periods also shows that some requirements change in time, to cope with the level of advances in different applications areas. These application cases are not to be directly compared with each other, rather they, in one way or another, complement each other in creating a more comprehensive set of requirements and contribute to the design and development of the (**GFI₂S**) solution. In each of the application cases, some of the requirements do not apply due to the type of application (symbolized by Φ); and some others were not addressed due the main aim of the application (symbolized by X). As such, the Waternet is a specific peer-to-peer system in which, the use of standards and middleware was not obligatory, while the MegaStore project presents a system for a large community in which, the use of common technologies in databases and Internet is mandatory.

Even at the level of each individual requirement, its consideration may be partially addressed within different research projects. For instance, the requirement of *the use of standards and middleware solutions* represents several concepts, which are partially addressed within the three projects. As such, the MegaStore system addresses the standards at the level of data modeling (e.g. UML), data definition (e.g. ODL), and information storage/retrieval (e.g. SQL/SQL3, OQL). While, the Virtual laboratory information management framework extends the use of standards to also support universal data access and scientific data mod-

eling. Similarly, the data integration and the information sharing are addressed at different levels of complexity within each project.

Major Requirements in terms of Information Management	Waternet 1996-1998	<i>MegaStore</i> 1999-2000	Virtual Lab 1999-2003	GFI₂S
Information integration for systems interoperation - Transparent access to data - Systems Interoperation - Information sharing - Data integration	✓ ✓ ✓ ✓	✓ Φ Φ Φ	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓
Security for access and visibility levels - User authentication - Information visibility levels - Separation between public and private data	✓ ✓ Φ	✓ Φ ✓	✓ ✓ ✓	✓ ✓ ✓
User Facilities - Easy access to data - User friendly interface	X X	✓ ✓	✓ ✓	✓ ✓
Use of standards and middleware - Universal access to data - Standard Data modeling - Multi-platform development - Data classification and catalogs	X Φ X Φ	Φ ✓ X ✓	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓
System efficiency and minimization - System efficiency and performance - Short response time for user requests - High bandwidth for data transfer - Good strategies for data storage - Effort and cost Minimization	X X X X X	✓ ✓ ✓ X ✓	✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓
Advanced Features - Support for new data types - Management of Large data sets - Combine advanced databases and web technologies - Combine O-O concepts and emerging standards	Φ Φ Φ X	✓ ✓ ✓ X	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓
Notation: ✓ :Addressed Φ: Does not apply X: Not addressed				

Table 1.1: List of Requirements of Today's and Forthcoming Applications

As depicted in column 4 of Table 1.1, the richness of the VL advanced applications, which emerge from various scientific domains, requires the consideration of most defined requirements. In addition, the Generic and Flexible Integration System (**GFI₂S**) presented in chapter 6 also considers the totality of these requirements in order to achieve a more flexible and open solution, serving the interoperation among advanced applications.

The list of requirements presented in Table 1.1 can also be categorized into two main categories of:

- Several information *Research Challenges* that need to be addressed in order to support the integration of information among networked applications, while preserving their local autonomy, heterogeneity, and distribution. Some of the *Research Challenges* that are addressed in different chapters of the thesis document include: information integration, security for access and visibility levels, and other advanced features such as supporting new data types and applying standards to object-oriented concepts.
- Several *standard tools* and *advanced Web technologies* that need to be applied for information management, to facilitate the information exchange among interoperable systems. Namely, standard data access, user friendly interfaces, and support for new data types.

The contribution of this thesis to the area of information integration and system interoperation is to investigate some of the research challenges and to apply emerging technologies and database standards to data integration mechanisms. Previous work on information integration for systems interoperation has developed some specific solutions for information exchange and data integration. These solutions are characterized by their specific tools and languages, which are mostly difficult to learn and hard to maintain. Our approach however, applies emerging standards and Internet technologies for information integration among autonomous and heterogeneous sites. This approach presents an open information integration facility for heterogeneous systems, while preserving their autonomy and distribution.

1.3 Thesis Contribution

The emerging advanced applications from scientific and business organizations present new challenges to the research in the domain of information management and interoperation. The challenges, primarily include: handling multi-media data types from the scientific domain, and deploying new co-working environments mainly based on the Internet technology, middleware, and standard solutions. MiddleWare in this thesis document is a general term representing any developed software that serves to “glue together” or mediate between two separate and usually pre-existing programs. For instance, a common application of middleware is to support programs written at one site with a particular database for access to other databases. As standard solutions to be applied to the information integration for systems interoperation, this thesis addresses: ODL for data definition, OIF/XML for objects representation, ODBC/JDBC for database connection, and XML as a flexible way to create common information format and share the format together with the data on the World Wide Web, intranets, and elsewhere.

Research in the domain of information exchange, interoperability⁴, and data source integration is still an open area for advanced architectures design, integration mechanisms, and tools development. Furthermore, the new emerging Internet technology for applications communications and middleware solutions for universal data access via standards offer promising approaches for the research in this area. Thus, there is a growing need for new

⁴Interoperability is basically the ability of a system or a product to work with other systems or products without special and extra effort on the part of the end-user

approaches to be designed and tools to be developed, in order to support interoperable information systems and to facilitate their collaboration mechanisms.

The work presented within this dissertation is an “*application driven database research*”, to better support the real information management needs and requirements of advanced emerging applications. It also introduces new approaches to integrate different types of data from heterogeneous applications to achieve broader information access, minimizing specialized development efforts, and attain competitive advantages.

In order to handle the new emerging data types, the work described in this dissertation document addresses two complementary trends. The first is to extend the traditional database systems to handle new and different data types and migrate these types of data into the DBMS. The second is to apply the middleware approach to provide standardized interfaces for all types of data, maximizing interoperability and reusability, while leaving the data in the place where it is generated or heavily used.

The main aim of the thesis is to address the design and partial development of a generic system to support information sharing among a wide variety of applications, and to assist their proper collaborative working environment, and flexible information integration.

Based on the expertise gained in the design and development of the various R&D projects during this Ph.D. study and based on the investigation, evaluation, and validation of the methodologies and systems discussed in chapters 3, 4, and 5 of the dissertation, an open and flexible integration approach is presented in chapter 6. The flexibility of this approach is achieved via some database extensions and through the deployment of object-oriented standards, emerging Internet technologies, and middleware solutions. The database modeling and the scientific data cataloguing and archiving, are supported via the deployment of the object-oriented standards. While, the emerging Internet technologies and middleware solutions allow universal access to the data, ease the information exchange mechanisms, and facilitate the multi-platform applications development. Therefore, *the thesis contribution to the research area of information integration resides in the specific combination of emerging standards in the filed with the fundamental research approaches and the way in which they are inter-linked.*

The approach we propose does not only take into consideration the work that has been done in the area of information exchange and interoperation. But, it also considers the utilization of both advanced web and database technologies in order to support the new requirements from the scientific applications for handling large multimedia data sets, and applies the new emerging technology for Internet communications and universal data access through middleware and standard solutions. As such, the developed framework and the designed methodology enable easy cooperative work among existing systems, while preserving their autonomy and heterogeneity.

1.4 Organization of the thesis

The remaining of this dissertation document is organized as follow:

- In Chapter 2, we study, analyze, and discuss a number of classifications for information management and the state of the art in data integration approaches. This study considers both distributed and integrated systems, and illustrates the need for these distinct approaches in order to support the complex requirements of different advanced cooperative applications from system engineering to scientific domains.

- Chapters 3 and 4 present two case studies in the field of intelligent supervision/control in heterogeneous and distributed applications, namely, the water distribution management and the advanced word wide databases for e-commerce. The designed and developed frameworks for these specific cases will be evaluated. This evaluation allows the validation of the most important and relevant features, that need to be taken into consideration when designing the flexible integration approach. These beneficial features are further addressed and deployed within the integration approach presented in chapter 6.
- Chapter 5 addresses a third application case from the scientific domain; namely, the information management framework of the Virtual Laboratory project. The Virtual Laboratory project addresses the issue of manipulating large scientific and engineering data sets in terms of data acquisition from heterogeneous resources, information modeling of complex and varied application types from several large scientific emerging domains, data archiving mechanisms for very large objects (e.g. binary and text), and resources cataloguing based on using the Dublin Core standard model. The information management approach of VL, focuses on the use of middleware and standard *de facto* solutions as a means to enforce and standardize the information access/retrieval processes among multidisciplinary applications. The proposed approach incorporates several advanced key features to support system efficiency and performance, enforced by the provision of security for access, and visibility rights to information, database indexing, cataloguing mechanisms, and database performance analysis.
- Chapter 6 proposes a Generic and Flexible Information Integration System (**GFI₂S**) based on the investigation, evaluation, and validation of the methodologies and systems discussed within the previous chapters. The extensibility of **GFI₂S** is achieved via several data management functionality extensions and through the deployment of object-oriented standards, emerging Internet technologies, and middleware solutions. An important distinction between the system we are designing and other integration systems resides in the introduction of the two components in the architecture of **GFI₂S**. The first component of the architecture (called *Local Adaptation Layer*) assures proper communication between the local data source and its federated layer. While, the second component (called *Node Federation Layer*) presents the node's window to the outside world for information sharing and interoperation.
- Finally, chapter 7 concludes the thesis and summarizes the main conclusions derived from this research and provides examples of the **GFI₂S** deployment in real applications.

Chapter 2

Information Integration Approaches, Mechanisms, and Tools

2.1 Introduction

A wide variety of information management and data integration approaches, mechanisms, and tools are introduced and being used for diverse applications in the domains of life sciences, engineering, education, health care, business, tourism, and art. These approaches are mostly designed and developed to cope with the specific requirements of each application. Among the major requirements for today's and forthcoming applications, which are described in Table 1.1 of Chapter 1, we enumerate: site behavior, user facilities, security for access and visibility rights, collaboration and interoperation, use of standards and middleware solutions, system efficiency and minimization, and other advanced features. The diversities of the proposed approaches for information management and data integration are due to different aspects considered for each application, and which relate to the DBMS architecture, data storage approaches, and system interoperation mechanisms.

This chapter presents a survey on several research and development approaches that have directly or indirectly contributed to the issue of information integration and interoperation mechanisms among autonomous, distributed, heterogeneous systems. Considering the main emphasis of the thesis that addresses the integration of heterogeneous data sources and interoperation among autonomous sites, special emphasis is given to the *data distribution*, *information exchange*, and *interoperability* issues. Thus, the three concepts described below are used as the base for the classification of existing approaches and mechanisms for information integration:

- ☞ **DBMS architecture:** For most of the emerging applications, and considering the enormous improvement in networking and communication protocols, it is clear that the client/server architecture has become the key issue for application development. However, depending on the type of application, decision has to be taken regarding whether to use a centralized, distributed [SBD⁺83, TBD⁺87, PH 98, NDL⁺00], or even a federated approach [HM 85, SL 90, TA 93, HTH⁺99, GAH 01] for proper sup-

port of the application requirements in term of information management and systems interoperation.

- ☞ **Data storage/access:** The manner in which data is being stored and accessed plays an important role in defining the proper integration mechanism, to be adopted for a given application. For instance, applications in the area of data mining and Online Analysis Processing (OLAP) [Sho 97, VS 99, PP 00, WB 97, FS 96, CD 97, and Kar 98] are based on the access to catalogs and repositories of data, which are reformatted and prepared for certain analysis tasks. While, in most applications from scientific and system engineering domains [WMP 98, PWD⁺99], the data processing mechanisms require that information has to be fetched on-line, processed on-line, and the proper decision has to be made on-line when necessary [BAK⁺00, ABK⁺00, AKB⁺01]. Similarly to the access mechanism, the data storage processing requirements may differ from one application to another. Furthermore, certain applications keep results locally and private, others publish the results immediately after generation, while a third type of application may require an evaluation time in order to validate the results before they get published and made available to the outside users.
- ☞ **System interoperation:** The interoperation mechanism for exchanging information and services among a set of users/applications within a collaborative community also defines the coupling mechanisms between those systems [LA 86, LMR 90, ZK 96, BA 98a, THB⁺98]. When data is integrated from external sources, it is very important for each system (e.g. data source) to clearly define the interaction mechanisms with other systems. Some criteria for this interaction involves handling the inter-linking among distributed data over different locations, and the format in which data is going to be exchanged.

It is clear that the complexity of the proposed approach for every advanced application depends very much on the specific characteristics of the application and its required level of integration. Therefore, *the more complex and higher is the requested level of integration in the application, the more complex and difficult becomes the development process.*

In this chapter, we study the main approaches introduced so far for information exchange and interoperation for different application domains. Furthermore, these approaches are discussed, evaluated, and those that can support some requirements for different applications and better fit the general integration purposes are validated for the general use cases. However, this is not a simple task since the requirements in terms of information handling are quite different in centralized applications from those that are distributed or federated applications. Some of the applications rely on the use of centralized archived data, others require homogeneous information replicated at different sites, a third type of applications may require run-time generation of information for decision support, and so on. Thus, *it becomes unrealistic to validate and conclude that one specific and concrete solution can support all types of today's and forthcoming applications.*

2.2 A Taxonomy for Information Integration

There are many classifications and initiatives aiming at the provision of a taxonomy for the information integration approaches. Some classifications look to the problem of integration/interoperation from the data access dimension, some others from the distribution/centralization dimension, and still some others from their heterogeneity and autonomy dimension [SL 90, RK 97, DD 99, ACM 00].

In order to describe other approaches suggested by research related to the subject of this thesis, and specially considering their wide diversity and distinct focus, we have resorted to the definition of certain classifications. In the proposed classification approach, since the main subject of the thesis consists of integrating heterogeneous data from distributed sources, we address the categorization architecture from the distribution/Integration point of view. The main purpose of this classification (and the names that are suggested and associated with each category) is to be able to discuss the common features among a group of systems that follow one general approach. As such, these “category names” might be defined differently in some other publications, since in general there is no common consensus among database researchers on the exact definition of these category names and the database systems they represent. Also this classification of information integration approaches is specifically focused on the main characteristics emphasized in the thesis, namely the design of a flexible environment to support many heterogeneous application domains, as it is the case for instance in scientific and engineering domains, like Virtual Laboratory environment.

Later in Chapter 6 of this dissertation, we will use the results of our investigation represented in this chapter and our classification of approaches, for identification and validation of a set of generic methodologies that can better support the general collaboration processes among heterogeneous nodes in a network of databases and applications. Provision of a set of generic tools and methodologies for information management facilitates the collaboration process among distributed and heterogeneous nodes. In such methodology the use of standard tools and middleware solution will play an important role in solving many issues related to the use of multi-platform systems, different architectures, and various information modeling methodologies.

Figure 2.1 illustrates a taxonomy diagram for information integration approaches. As depicted in this figure, at the *first level* of the proposed hierarchy, the management of heterogeneous data sources (information) is classified into two main categories of *Distributed Systems* and *Integrated Systems*. Distributed systems typically support applications that share common database software at both DDBMS servers and their applications. Integrated systems however, support database applications that address similar tasks in different manners or using different representations and data modeling systems. Within each of these two categories several approaches are identified, studied, and evaluated based on the applications’ requirements.

At the *second level* of this taxonomy, different and various approaches for information integration are derived from the two categories illustrated at level 1 of Figure 2.1. On one hand, distributed database systems can follow *horizontal* fragmentation, *vertical* fragmentation, or *hybrid* fragmentation [EN 00, OV 99]. On the other hand, when the application becomes more complex and requires additional functionalities, most research on related approaches, focusing on the needs for data heterogeneity resolution, result in a variety of *integrated systems*. Although, a number of researchers in this area still consider all these approaches as heterogeneous distributed systems.

Within the integrated approach, Domenig and Dittrich [DD 99] present a taxonomy of systems for querying heterogeneous data in which, they distinguish between materialized and virtual approaches. Similarly, Florescu et al. [FLM 98] make an important distinction, in building data integration systems on whether to take a warehousing or a virtual approach. In our classification, we name the two approaches respectively *Physical Integration* and *Virtual Integration*. In a *Physical Integration* the data originating from local and remote sources are integrated into one single database on which all queries can operate. In *Virtual Integration* (also referred to as multidatabase systems in some literatures), data remains on

the local/remote sources, queries operate directly on them and information integration has to take place on the fly during the query processing.

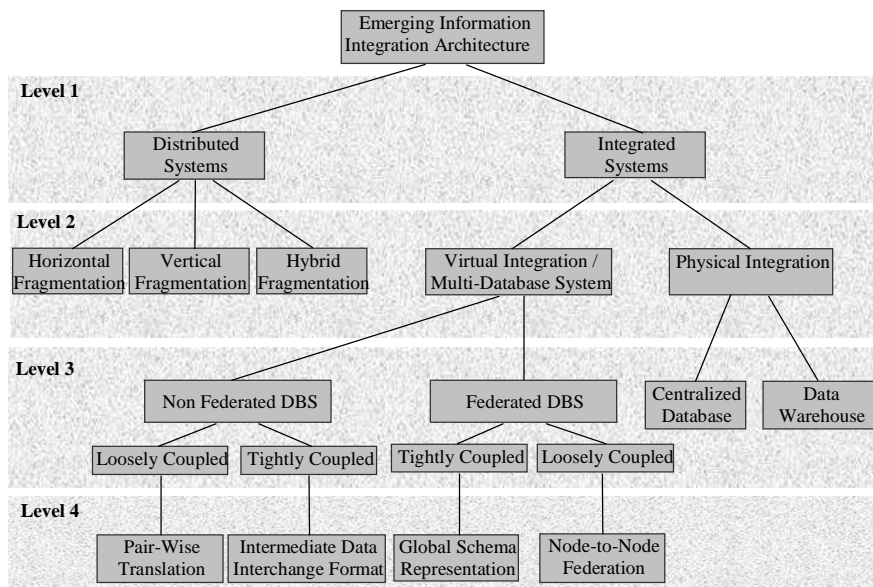


Figure 2.1: Information Integration Approaches - Classification

At the *third level* of the taxonomy, the variety of the proposed approaches becomes more and more specified and complex, this is due to the advanced functionalities and complex features required by advanced applications emerging in the domains of system engineering, medicine, biology, etc. Nowadays, those applications are of wide use, significant importance, and real necessity.

- The physical integration expands into *centralized databases* and *data warehouses*. In a centralized database, information is migrated from various sources into a universal DBMS, while in data warehousing information may be imported in different form and volume than it exists in its originating sources.
- The virtual integration derives into *federated* and *non-federated* systems. Each of these systems can be either *loosely* or *tightly coupled*.

At the *fourth level* of the taxonomy, we categorize the *intermediate data interchange format* and the *pair-wise translations* as examples of non-federated systems and *global schema representation* and *Node-to-Node federation* as examples of federated multidatabase systems.

The next sections present and discuss in more details these different approaches illustrated on Figure 2.1; more focus will be put on Virtual Integrated systems, due to their relation and direct impact on the subject of the thesis.

2.2.1 Distributed Systems

In distributed systems, the applications considered by the network have similar functionalities and share similar type of information. Furthermore, most of the involved sites use the

same software for data modeling, information management, and accomplishment of their global functionalities. In such architecture, distributed database systems (DDBMS) are defined as a collection of multiple logically interrelated databases distributed over a computer network. A distributed database management system (DDBMS) is a software system that manages the distributed database, while making the distribution transparent to the user.

Figure 2.2 illustrates a distributed database architecture, in which information is stored within four distributed databases located at different remote sites and inter-linked through the communication network. In such a system, distribution is transparent to the user in the sense of hiding the details of where the data is physically stored within the system. Thus, from the point of view of the operational details of the network, the user has the freedom concerning the data location transparency and objects naming transparency [EN 00].

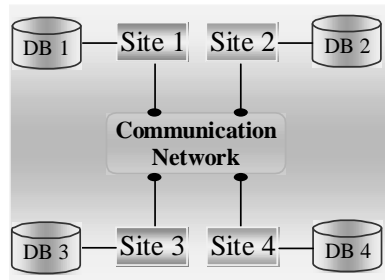


Figure 2.2: Distributed Database Architecture

Figure 2.3 presents an example of a simple database model. Its structure consists of two classes: *Product* and *Customer* each characterized by a set of attributes defining the two classes and a relationship among them. The example will be used within this section to illustrate some of the concepts related to distributed database systems.

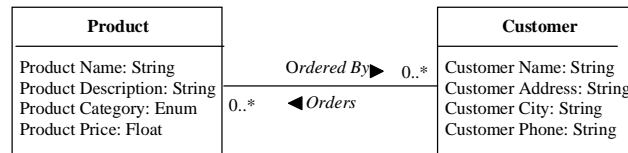


Figure 2.3: Example of a Simple Database Model

2.2.1.1 Data Fragmentation and Allocation

Figure 2.4 illustrates an example of the three types of fragmentation, which are possible in distributed systems:

- ① **Horizontal fragmentation** distributes a relation into sets of tuples (rows). The database model conserves the same structure, restriction however is applied to the set of records (instances) based on some conditions.
- ② **Vertical fragmentation** distributes a relation into sub-relations where each sub-relation is defined by a subset of the columns of the original relation. All objects are partially present in the fragmented class, restriction is made for some attributes of the class (table).

- ③ **Hybrid fragmentation** partitions a relation by applying the horizontal and vertical fragmentation strategy one after the other. Restriction is made on both attributes and instances of the fragmented class.

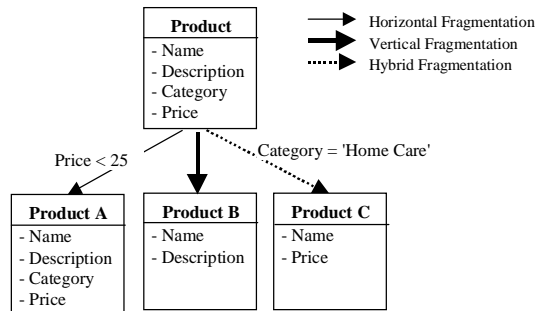


Figure 2.4: Fragmentation in Distributed Database Systems

Figure 2.4 illustrates the three types of data fragmentation:

- Horizontal fragmentation: the class *Product* is restricted at the instances level to only preserve products of '*Price < 25*'.
- Vertical fragmentation: the class *Product* is restricted at the attribute level to *Name* and *Price*.
- Hybrid fragmentation: the class *Product* is restricted at the attribute level to *Name* and *Price*, and at the instances level to only hold products of category '*Home Care*'.

In distributed systems failure at a single site does not make the entire system unavailable to all users; when the data and software fail on one site the other sites continue to operate, which improves both system reliability and availability¹ in comparison to centralized approach. Furthermore, safeness can also be achieved and improved by replicating data and software at multiple sites.

Figure 2.5 gives a global overview and an example on how fragmentation and replication can be established within a distributed system. Here, the data fragmentation and distribution is illustrated using the simple data model presented in Figure 2.3. In addition to the three types of fragmentation, a mirror site is defined as a part of the distributed system. The mirroring site assures in fact the availability and safeness of information by holding a replicate of all the data, which is distributed among the other sites.

The distributed database architecture is still valid and used, a representative example and a good candidate for distributed systems is the banking application environment where all the distributed sites share the same database schema and participate in achieving a global and unique task. Another example with a small difference in the distributed database model can be the case of administrating a main hospital with many branches and care centers performing similar activities.

However, distributed systems are not fully appropriate for instance in emerging scientific applications, since it is very hard to guarantee the usage of the same data model and software tools for all cooperating/integrating sites.

¹Reliability is defined as the probability that a system is running at a certain time point, whereas availability is the probability that a system is continuously available during a time interval [EN 00]

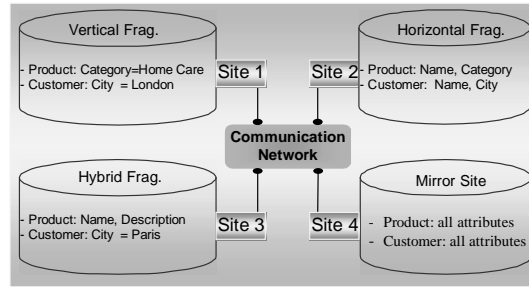


Figure 2.5: Data Distribution and Replication among Distributed Databases

2.2.2 Integrated Systems

In the main area of integrating heterogeneous and distributed information sources, the information integration generally implies uniform and transparent access to data managed by multiple databases. The task of an integrated database system is to answer queries that may require extracting and combining data from multiple local/remote data sources. An important distinction in classifying the different strategies for the manipulation of the data is whether to take a physical or a virtual integration approach [FLM 98].

1. In the physical integration, data from multiple sources is loaded into one single comprehensive new database on which all queries are applied. This requires that the new database needs to be updated when data changes, but the advantage is that adequate performance can be guaranteed at query time.
2. In the virtual integration, the data remains at the local/remote sources where it belongs. Queries to the integrated system are decomposed at run time into sub-queries to be applied to the local/remote sources, and then the integration of the sub-queries results must take place on the fly during the query processing. In this approach data is not replicated and is guaranteed to be up-to-date at query time.

The following sections will describe in more details both the physical and virtual integrations and address different categories derived from these two approaches. The variations in manipulating and handling the exchange of information reside in the manner the data is gathered and accessed.

2.2.2.1 Physical Integration – warehouses/, malls, or marts

The physical integration (also known as materialized approach in some literatures) involves extracting data from a variety of heterogeneous distributed systems and applications, standardizing it to fit a global format, transporting it from the place where it belongs, and loading it in the local database in desired formats. There are essentially two variants of materialized systems [DD 99]:

- a. Data from the remote systems is extracted, integrated, and stored in a *centralized database*, therefore, the remote systems are not used after data is extracted. In such a system, in case of updates to both data and meta-data of every remote site, the centralized database must be adjusted to reflect eventual changes in the remote systems. The main drawback of this approach is that existing applications have to be rewritten to fulfill the new database model.

- b. Data from the remote sources are imported into one DBMS, *the data warehouse*. The difference with the previous case is that the underlying data sources are still operational and the warehoused data is typically not imported directly in the same format and volume as it exists in local systems. It is mostly transformed, cleaned, and prepared for certain analysis tasks, like data mining and OLAP (Online Analysis Processing).

The physical integration model is considerably adopted and used within data warehousing environments [FS 96, CD 97, WB 97, Kar 98] and OLAP databases [Sho 97, VS 99, PP 00]. Recently, several commercial DBMS products provide certain solutions for data warehousing and OLAP. To name a few of them, Oracle Migration Workbench² [Daly 01], DB2 Warehouse Manager [ED 01], Sybase warehouse studio [Syb 99], SAP/R3 Hummingbird [Hum 00], Hyperion [Hyp 01], Cognos [Cog 00], and Comshare [Com 00]. They provide capabilities that specifically address the specifications and requirements of data warehousing. Furthermore, These software houses provide complete open warehouse design and meta-data management environments that simplify the process of building and managing warehoused data while delivering impressive flexibility.

Within the Virtual Laboratory³ project, the physical integration approach can be validated for some applications to support data archiving, cataloguing, and publishing, where organizations with similar working environments are able to gather and collect their finalized data in a global repository to be used for other activities. Archives in scientific applications provide storage of information off-line, and make it available when necessary to be used as input for experiments during a mining session, or an analysis process. Published data in the area of scientific experimentation refers to the final successful experiment results that an organization/system wishes to make available to outside users. Both archived and published data represent a huge volume of data that is not susceptible to be changed frequently.

However, the physical integration is not fully suitable for large scale interoperable applications, in which the design and development of a real federated system proves to be more appropriate.

2.2.2.2 Virtual Integration - Multi-Database Systems

The main aim in the virtual integration is to give the user the impression of working with a single DBMS, while, in fact, the data is managed by several individual DBMS. This approach is more appropriate for building systems where the number of databases is large, the data is changing frequently, and there is little global control over the participating local/remote data sources.

In the area of implementing integrated information systems, in which different systems need to collaborate, the development of software layers and tools turn out a necessity. These tools facilitate the exchange and the sharing of data among collaborative systems by translating data between different applications and making the information distribution transparent to the users. From the implementation point of view of multidatabase systems, many different approaches, dealing with this issue, are designed and developed.

²<http://otn.oracle.com/tech/migration/workbench/content.html>

³The Virtual Laboratory project (1999-2003) supported by the Dutch ICES/KIS foundation aims at designing and developing hardware and software reference architecture and a digital library framework to enable scientists and engineers to work on their problems via experimentation in the field of technical and scientific applications, making optimum use of by modern Information Technology.

Within the next sections we will present and discuss different approaches that have been investigated in the area of integrating heterogeneous and distributed data sources, namely, the pair-wise translation, the intermediate data format, global schema definition, and the node-to-node federation.

Pair-wise Translation

Using the pair-wise translation, we keep and preserve the various representations of different applications then, we provide translation tools from one application to another. As depicted in Figure 2.6, within this approach we have to develop a two-side dependent translator tool for every application within the system. Any change in any of the applications, requires rewriting some parts of code, the integration of a new application requires the development of $2*N$ new translators in which N represent the number of communicating applications in the network.

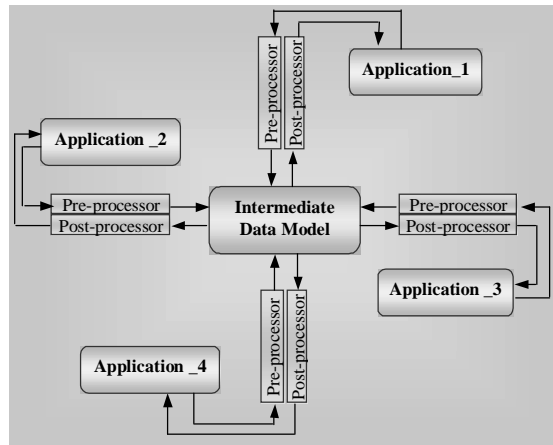


Figure 2.6: Two Side Dependent Translation

The two-side dependent translation approach provides the following advantages [AE 95]:

- Every application will be designed in a manner that optimizes the considered tasks.
- Every module may be extended without any necessary updates in the other parts of the system.
- The specific translation to each application reduces the volume of information that needs to be exchanged, since every module can execute locally some tasks and send the results in a condensed format.
- A more flexible exchange for data and services can be obtained using database standards for data modeling and information access/exchange (e.g. ODL, OQL, and XML). These standards facilitate the logical links and interfaces between different pieces of information within those interconnected applications.

The pair-wise translation approach better suits in small environments where the number of communicating applications is very limited.

Intermediate Data Interchange Format - Common Data Definition Model

The integration approach through intermediate data format develop a general representation for all the anticipated data types, this representation must include some dependencies among different types. The intermediate data model also makes assumption for its entities both for their representation and the methods of handling them. An entity, e.g. a *Triangle* for instance, can be represented by its three points in application_1 and by its three segments in application_2. Thus, the handling methods for this entity change from one representation to another.

Figure 2.7 illustrates the architecture used by the intermediate data format, in which every application participating in the system must have at its disposal two translators:

- a- The *pre-processor* from the application to the intermediate format, which translate data from the format used by the application and make it available according to the intermediate common format,
- b- The *post-processor* from the intermediate format to the application, which takes information available in the intermediate format and translates it to fit the local format adopted by the application.

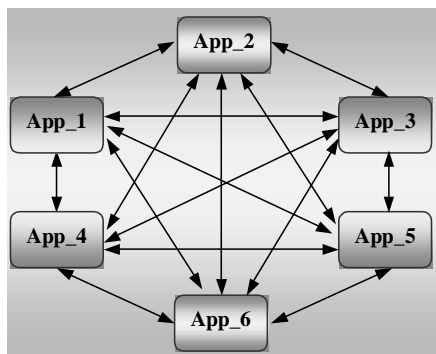


Figure 2.7: Access Through the Common Data Model

Examples of standards that deploy the intermediate data format and consider it for modeling object structures and representations include *STEP*, *IGES*, *DXF*, *SET*, *VDA*, etc. [Fow 95, TM 96]. These standards (also referred to as neutral formats) are designed to fulfil a number of high level industry requirements, and are based on a number of fundamental principles for shared product databases.

The advantage of using the common intermediate definition over the two-side dependent translation is that each application has to communicate only with the intermediate format instead of communicating with all other applications, thus reducing the number of necessary developed translators, especially when the number of applications grows.

Nowadays, the deployment of middleware solutions and standards can extend this approach, so the inter-modules communication becomes transparent. The *node-to-node* integration approach, presented in Chapter 6, will benefit from a part of this architecture, in which, database standards will play a role similar to the intermediate data model, in order to unify the schema definition, the query language, and the data exchange processes within the integrated applications.

The Global Schema Representation – Schema and Data Integration

The global schema representation (also known as universal representation in some literatures) is based on the development of one single representation that manages all types of data required by the networked applications and completely integrates multiple data sources into one global database in order to provide a single and coherent view on the data [SP 94]. As depicted in Figure 2.8, the global schema representation requires that all system components, accessing the single database, shall conform to the shared format and create their specific views based on it.

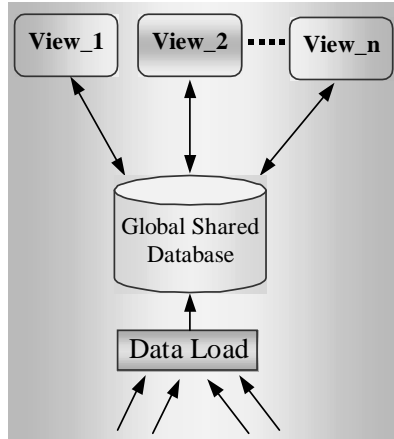


Figure 2.8: Access to the Global Shared Database

Modular and extensible systems make this approach not attractive for the following reasons [AE 95, EK 91]:

- There is no guarantee that new applications do not need different formats for their internal operations,
- In today's applications, there is always the introduction of new methods that require particular information structure to be reflected in the underlying database model of the applications,
- The use of such a representation, which supports an exhaustive structure, represents a supplementary load for systems that manage simple applications.

In addition, the global integrated schema is hard to maintain and to automate. Furthermore, systems autonomy is often sacrificed in order to solve semantic problems.

Node-to-Node Federation

The information dealt with in federated systems consists of different pieces of information gathered from disparate local and remote sources and integrated into one coherent view, known as integrated schema. Therefore, the integrated schema is constructed by merging the structure of the local schema with the various imported schemas. The imported schemas represent the part of information that other systems wish to share with the outside world.

There are two main features distinguishing a federated system from a traditional database system [FLM 98]:

1. A system within the federation does not communicate directly with a local storage manager. Instead, the query execution engine communicates with the corresponding wrappers on top of it. Those wrappers are defined within the federated layer developed for each system.
2. The user does not pose queries directly on the schema in which the data is stored. Instead, the user poses queries on an integrated (or mediated) schema defined at the federated layer of his system.

In the federation approach, every site/user is responsible for integrating the schemas they need for their applications. Support is therefore provided by a federated or multidatabase language that contains the syntactic constructs needed for accessing and manipulating disparate and autonomous databases [BBE 99]. In addition to these constructs, a federated system requires a set of sources defining the mapping rules, the semantics description, and eventually the data exchange format. These sources also define the manner in which information will be imported, integrated, and accessed.

The way in which information is imported, integrated, and accessed in multidatabase systems differs from one approach to another. They all consider the data distribution, the heterogeneity of information, and the autonomy of the systems participating in the collaboration. However, visibility level for the data and the heterogeneity at the DBMS level are not fully considered within most of those systems. In addition, extensibility for new DBMSs to be considered within the federation is not supported.

Some of the commercial products such as Sybase, Oracle, Ingres/Star, and UniSQL/M have extended the functionalities of their systems to support some of the general requirements of the multi-database systems [HBP 94] [SYE⁺90]. Therefore, each of these DBMSs provides some type of federation⁴ where the federated schema is commonly defined for all the nodes that participate in the collaboration. The so-called federated schema is created based on each node decision about what information is commonly shared. A node can also decide at any time to extend, restrict, or remove its shared data. However, the federated approach provided by those systems represent in fact a very limited federation where a node is only allowed to share the same data with all the other nodes or do not share any thing.

The UniSQL/M, for instance, is an object-relational multi-database system that enables interoperability of multiple databases. Currently, UniSQL/M supports the integration of the following other database types: UniSQL/X, Oracle, Informix, and Sybase. UniSQL/M gives its applications and users access to those databases through a single SQL interface, however, it does not address the visibility level of shared data at the local databases. Through the registration process, a database has to decide to share every thing or do not share; thus, there is no support for full federation process as required in different types of applications. Furthermore, user access is defined at the global schema without support for any access level, which is very critical in today's applications.

The multidatabase approach, as addressed and provided by some commercial products, better suits specific applications within the same organization in which, different software tools need to adjust their input/output. In such a case (within the same environment), visibility levels may not be highly required and heterogeneity of used systems and data modeling is not very complex.

⁴Federation provides applications and queries with access to data stored in multiple databases without requiring knowledge of their distributed location.

Table 2.1 illustrates and evaluates some of the commercial database systems in terms of integrating information from different sources. The table shows the product name, the used data model, support for information visibility levels, and the consideration of standards in terms of information modeling, query language, and information exchange. The evaluation illustrates that most of these products are extension to the existing relational DBMSs, in which the use of standards is well considered, while the information visibility levels and the node-to-node federation are not fully taken into account. Data is usually collected from different sources and locally (physically) stored according to the global schema definition.

Product	Data Models	Global Schema	Visibility Levels	Use of Standards	Data Export	Query Language
UniSQL/M	Relational, UniSQL, Oracle, Informix, Sybase	Yes	No	Yes	No	SQL
Sybase	Relational	Yes	No	Yes	No	SQL-Like
Oracle	Relational	Yes	Yes	Yes	No	PL/SQL
Ingres/Star	Relational	Yes	No	Yes	No	SQL-Like

Table 2.1: Commercial Systems Evaluation in Terms of Information Integration

The remaining of this section will address three examples of federated systems for which, the adopted architecture is close to satisfy the requirements of today's applications emerging from scientific and industrial domains. The prototypes concern:

1. *PEER system* [ATW⁺94]: an object-oriented federated information management system supporting the import, export, and integration of heterogeneous and autonomous schemas.
2. *DIMS of PRODNET II* [CA 99]: Distributed Information Management System for an IT platform supporting industrial virtual enterprises, and providing mechanisms for inter-operation and information exchange in real time.
3. *WebFINDIT* [BBO⁺99a]: a system supporting the database equivalent to the World Wide Web and addressing interoperability in Web accessible Databases.

Example 1: PEER Federated System⁵

PEER [ATW⁺93, ATW⁺94] is an object-oriented federated information management system designed and developed at the University of Amsterdam to support the management, sharing, and exchange of heterogeneous information in a network of loosely/tightly coupled nodes. Using *PEER*, each node in the federation network can autonomously decide about the information that it locally manages, and which part of its local information it wishes to export and share with other nodes. Each node can import information that is exported by other nodes and then transform, derive and integrate (a part of) the imported information to fit its interest and corresponds to the local interpretation. *PEER* is a pure federated system; namely there is no need for a single global schema to be defined on the information to be shared by different nodes, and there is no global control among the nodes.

The *PEER* integration infrastructure helps the human users in a cooperative team, by supporting their information integration at different levels of granularity, e.g. to support

⁵More details related to the *PEER* federated system and its application in Waternet will be later provided in Chapter 3.

the global task, or among different activities and sub-activities. The PEER system provides an environment for the cooperation and information exchange among different nodes in a network, where every node is composed of one server process and may consist of several client processes.

The PEER information management strategy supports the sharing and exchange of information among nodes, without the need for data redundancy and/or creation of one global schema. Therefore, the problems of data consistency, referential integrity and update propagation, and the need for a common glossary of concepts and definitions are eliminated.

As depicted Figure 2.9, every node in the PEER layer is represented by several kinds of schemas; one local schema, a number of import and export schemas, and one integrated schema.

- The *local schema* is the schema that models the information that is available and stored locally within the node,
- The various *imports schemas* model the information that the node needs to access from other local/remote nodes,
- The *export schema* models the information that a node wishes to make accessible to other nodes, and
- The *integrated schema* presents coherent pool of information on all accessible local and remote information.

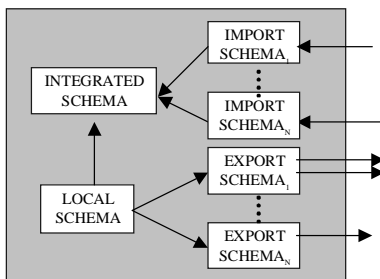


Figure 2.9: Schemas Representation in PEER

Example 2: DIMS of PRODNET II

The European ESPRIT project PRODNET II (1996-1999) [CA 99] designs and develops an open IT platform to support industrial virtual enterprises, with special focus on the needs of Small- and Medium-sized Enterprises (SMEs). The PRODNET project provides a VE support infrastructure, in which the involved SME companies are able to inter-operate and exchange information in real time so that they can work as a single integrated organization, while at the same time keeping their own independence and autonomy.

The proposed infrastructure for PRODNET II [GR 01], depicted in Figure 2.10, is composed of three main components: the Internal Module (including the PPC and other engineering modules), the Advanced Coordination Functionalities, and the PRODNET Cooperation Layer (PCL). More details related to the description of different components of PRODNET can be found in [GAH 01, FAG⁺00, KRS⁺99, CL 99, GCL 99, Sch 99, and OAB

99]. This section however, will give a brief description of the DIMS component of the PCL layer, due to its relation with the subject of the thesis.

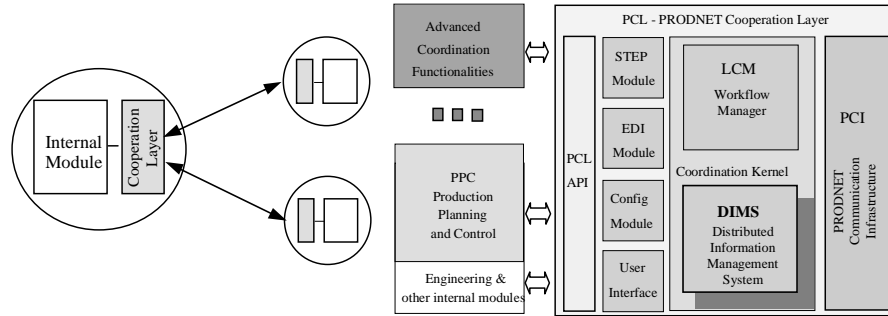


Figure 2.10: The PRODNET Reference Architecture

The DIMS (Distributed Information Management System) is responsible for modeling and managing the exchange of all integrated VE cooperation-related information, while preserving the autonomy and information privacy of the involved enterprises [GAH 01, FAG⁺00].

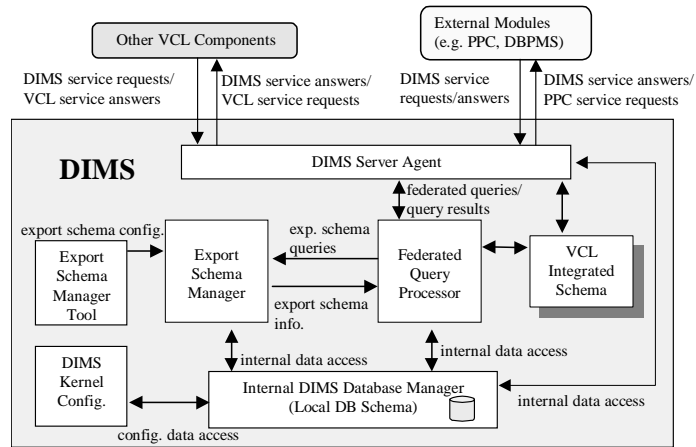


Figure 2.11: General DIMS Architecture Approach

The general reference architecture for DIMS embodies the following components (see figure 2.11):

- *VCL⁶ Integrated Schema*: provides a unified definition of both the local and the distributed VE information that can be accessed by end-users and applications at each VE node. Other VCL components and external enterprise modules issue federated database queries on this schema through the DIMS server agent, which takes care of the interaction with the federated query processor, or with the local database schema.
- *Export Schema Manager and Tool*: encloses the functionality to create and maintain

⁶VCL: VE Cooperation Layer.

the hierarchy of export schemas that are defined on the VCL local schema, based on the visibility of access that need to be specified for a given node.

- *Federated Query Processor*: transparently supports the access to data distributed over the nodes of the VE network, taking into account the specific visibility access rights (represented by export schemas) defined for every node.
- *DIMS Server Agent*: receives and dispatches all the DIMS service requests issued by other VCL modules. The Server Agent first determines the nature of the service requests and then triggers the activation of the involved DIMS internal components.
- *Internal DIMS Database Manager*: represents the server tier that provides the fundamental functionalities expected from a database management system including: transaction management, data storage and retrieval facility, stored procedures management, SQL support, database triggers, etc.
- *DIMS Kernel Configurator*: allows the user to specify the configuration of certain DIMS operation parameters (e.g. DIMS users and access security definitions (accounts and passwords), Communication port number of DIMS server, and Timeout duration for distributed queries).

Example 3: WebFINDIT a System for Querying Web Databases

WebFINDIT [BBO⁺99a] [BBH⁺99][BBO⁺99b] focuses on the design and implementation of an architecture to support appropriate tools to manage the description of, location, and access to data in the context of highly dynamic networks of information sources. It proposes the use of flexible organizational constructs called coalitions and service links, to facilitate data organization, discovery, and sharing among Internet accessible databases.

In order to achieve broad and flexible access to those remote information sources, WebFINDIT provides the WebTassili as language that supports the definition and manipulation of information using two levels mechanism for Querying Web Databases. At the meta-data level, users/applications can explore meta-information about a particular database, while at the data level, they can query actual information stored in databases. WebTassili also educates users about the information available and focuses here on those aspects of the language designed specifically for locating information sources and educating users. The WebTassili framework translates WebTassili queries to the native local languages, and translates results from the native systems format to WebTassili.

The WebFINDIT prototype has been implemented using advanced object technology, including CORBA as a distributed computing platform, Java, and Database Connectivity Gateways to access native databases and to connect to databases. The combination of technologies such as CORBA and Java offers a compelling middleware infrastructure to implement wide-area enterprise applications, in which CORBA is used to provide a robust communication infrastructure while Java is used to allow a dynamic deployment of the system over the Web. Different database management systems⁷ have been used as a test-bed for the prototype.

As shown in Figure 2.12 [BBO⁺99a], the WebFINDIT components are grouped in four interactive layers:

1. The **Query Layer** gives users access to WebFINDIT services through two components: the browser and the query processor,

⁷According to [BBO⁺99a], The actual WebFINDIT prototype interconnects 26 databases, which are implemented using four different DBMSs: Oracle, mSQL, DB2, and ObjectStore.

2. The **Communication Layer** manages the interactions among WebFINDIT components and mediates requests between the query processor and co-database servers,
3. The **Meta-data Layer** consists of a set of co-database servers that store meta-data about the associated databases (e.g. information type, location, and coalitions).
4. The **Data Layer** has two components: databases and Information Source Interfaces (ISIs). An ISI provides access to a specific database server by delivering requests from the communication layer and retrieving results from the database.

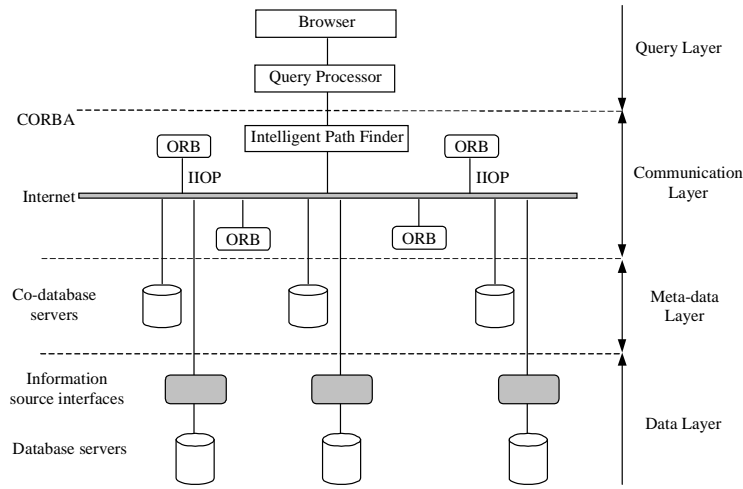


Figure 2.12: WebFINDIT Components are grouped in four Interactive Layers

Discussion of the three Examples

In comparison to the main aspects of the approach suggested in this dissertation for information integration, hereafter, we briefly discuss the three examples presented above. These examples are discussed in accordance to the detailed requirements of distributed, heterogeneous, and autonomous sites, as described in section 1.1.

The PEER system uses its specific languages for modeling and querying the data, which make the knowledge of PEER a must for every application participating in the federation community. The fact that PEER is not based on standards, makes the process of learning its languages and maintaining its applications a hard task that require appropriate skills and expertise in the field. In addition, PEER by itself is a full DBMS that is required to be used as the database management system for each application within the federation community. In our opinion, a better approach to information integration must preserve the involved systems with their existing DBMSs, and provide a higher level of integration that logically links those DBMSs in a suitable manner.

The DIMS approach, proposed for PRODNET, provides in fact a comprehensive federated solution for a certain type of applications that cooperate together towards the achievement of a common and global goal of the VE. Within the DIMS approach of PRODNET, users and their access rights are well defined, however, the heterogeneity issue of different

sites of the collaboration (e.g. legacy systems) is left to be resolved by the sites themselves. In other words, different collaborative sites share a common data model and a common query language.

WebFINDIT provides an architecture to manage and query data in the context of highly dynamic networks of information sources. Similar to many other solutions, WebFINDIT uses its specific languages for data coalitions and information retrieval, which also require appropriate knowledge and expertise of these specific languages and modeling concepts. In addition, users authentication and their access rights to the information sources are not addressed to the required level. From the global architecture point of view, it seems that WebFINDIT proposes a global solution to access data from external sources, rather than providing an integration mechanism where different application can cooperate and share each others data and services.

2.3 Further Classifications and Categorizations

In this section we evaluate some of the approaches for information integration based on two classifications variant, namely, (1) a classification based on the developed approaches, and (2) a classification based on the application requirements.

Approach	Application Area	Advantages	Disadvantages
Distributed Systems	Banking, hospital chains, etc.	Availability Reliability	Off time updates at central nodes, Performance
Data Warehouse	Archiving, statistics, decision support, and OLAP	Uniform access to data Adequate performance guaranteed at query time	Updates need to be reflected Data is accessed off-line, so not up to date
Common Data Definition Model	Archiving, statistics, and decision making	Uniform access to data Data is always fresh	Hard to automate Hard to maintain
Pair-Wise Translation	Specific Applications	Well optimized	Specific interfaces
General Federated Approaches	Business, and scientific applications	Uniform access to disparate databases	Visibility level is not fully addressed
Node-to-Node Federation	Scientific domains. Business applications.	Full federation, Good visibility level. Data is not replicated. Data is guaranteed to be fresh at query time.	Hard to build and to maintain. Difficult to reach an agreement.

Table 2.2: Approaches Evaluation based on the Developed Systems

Table 2.2 evaluates some of the approaches for information integration as developed to fulfill the specific requirements in different application domains. It also outlines the advantages and disadvantages of each approach. Therefore, the deployment of an approach,

to be used by an application domain for their information management, is strongly based on the specific requirements of every application. In addition, the final decision concerning the approach to follow also rely on the evaluation and estimation of the gained benefits and advantages against the sacrificed features, which relate to system autonomy, data freshness, and information privacy.

In Table 2.3, we classify the application domains into four categories of Chain Systems, Data Archiving and Cataloguing, Data Publishing, and large scale Applications. The classification, which is based on the main characteristics of each application domain, also determines the application type and provides examples of these applications.

Application Type	Approach	Characteristics	Examples
Chain Systems	Distributed Systems	Similar environments at geographically distributed sites. Applications may use similar data models and management systems. Data is susceptible to change.	Banking, hospitals, supper-markets, and insurance organizations.
Data Archiving and Cataloguing	Centralized Databases	Data is gathered from different sources in order to be archived and used by a third party (service). Archived data is not susceptible to change frequently. Archives are usually copyrights protected.	Scientific results, Biology, reference databases, and product information.
Data Publishing	Data Warehousing	Data is collected from distributed sources, cleaned, filtered, and integrated into one format. Data is susceptible to change and the new updates need to be reflected.	Data mining, and On-line Analysis Processing (OLAP).
Large scale Applications	Federated Systems	Data needs to be accessed from different remote resources. Data needs to be fresh.	Science environment, health care, business, and industry.

Table 2.3: Approaches Evaluation based on the Application Requirements

2.4 Discussion

Several approaches, among the ones presented in this chapter, address the main aspects of multidatabase systems. Namely, heterogeneity, distribution, and autonomy are addressed with a certain level of details. However, the requirements of today's applications are more complex than what is provided, more precisely:

- **Distribution** must be addressed in a manner were data remains at its originating source while access to it is gained on-line, when needed and in the requested format. Under the normal consideration, up-to-date data must be gathered from multiple disparate data sources.
- **Heterogeneity** must be supported at the level of data representation (including semantic and syntactic heterogeneity) and at the level of DBMSs (e.g. relational model,

CODASYL network model, object-oriented model, and flat files).

- Each system within the collaboration community must fully preserve its *local autonomy* in terms of controlling its information management.
- The *import/export of information* at each node must be well controlled and very flexible. A node must be able to define as many export schemas as required by different applications. Each export schema shall represent the part of local information the node wishes to make available for other specific applications. Similarly, a node shall import as many import schemas as needed from its different applications; moreover, information is imported in the format desired by the requesting applications.
- *Bilateral agreements* need to be established between the members of a collaboration network in order to define the information to share, the manner to access it, and the circumstances under which the shared information will be used.

In addition, generally most of the proposed approaches lack the means for generalization and only address specific domain-dependent cases. Adding a new site to the cooperation requires considerable expertise and effort in order to interface it with all communicating systems. Thus, support for applications extensibility and evolutions are not guaranteed in most of these approaches. For instance, the PEER federated system provides a loosely coupled federated environment where autonomy is preserved, users visibility rights are supported, and its federated schemas are well defined. However, the existing implementation of PEER requires that, for their interaction/cooperation, all the nodes (agents) within the federation community use the same data model (the PEER model) and the same query language (PEER language) in their “cooperation layer”. Similarly, WebFINDIT proposes an architecture that allows dynamic couplings of Web accessible databases using a common data model and a common query language called WebTassili. Thus, for both systems, expertise is required to build cooperation layers when initiating a new collaboration network or extending the existing federated nodes to support new members.

The approach we propose in chapter 6 is mainly based on and extends the PEER architecture in defining the type of information managed by each node (namely local, import, export, and integrated schemas). In addition, the approach allows different applications to follow widely understood formats and common languages for modeling and querying the data. Further, it preserves all application’s autonomy and builds a federated layer on top of each application. Different than the original development of the PEER database system, the information integration approach, proposed in chapter 6, supports the inter-nodes communication based on the usage of middleware solutions and standards; namely, ODL for schema definition, OQL for query formulation, and XML/OIF for data exchange. Moreover, the solution we propose can be considered as a higher abstract level that is used to interconnect different information sources, rather than being a new DBMS intending to replace the existing DBMSs in all networked applications.

Chapter 3

WATERNET: Intelligent Supervision and Control in Heterogeneous and Distributed Application

This chapter describes the design and implementation of the Waternet integrated/federated environment, which allows the coupling of distributed, heterogeneous, and autonomous subsystems in the water distribution/management system [ABH 98a, ABH 98b]. The Waternet ESPRIT IV project (No. 22.186), aims at developing an evolutionary knowledge capture and an information management system. Waternet supports the control, optimal operation, and semi-authorized decision making for drinkable water distribution networks.

The main goal of the Esprit project Waternet is stated as *Knowledge Capture for Advanced Supervision of Water Distribution Network*. It involves: the development of several different subsystems and their integration into a coherent environment, in which they can easily access and exchange the information they need. In order to support the requirements of advanced distributed control and management of water distribution networks, there is a need to develop a strong interoperable information management system. The interoperable information management system supports the cooperative heterogeneous subsystems with their exchange and management of large amount of data.

The Development of the Waternet information management system is based on the use of the PEER federated system. PEER provides basic means for information integration and interoperation among heterogeneous systems without the need for data redundancy or replication at multiple sites. The information management framework for Waternet, presented in this chapter, is augmented with the development of adapters serving the need for system flexibility and openness. Therefore, adapters facilitate the insertion of new components to the system. Our contribution to the Waternet project consisted of:

- ☞ Analysing the requirements of the Waternet subsystems (units) and the design of the database structure and communication mechanisms for their managements of information. Waternet units included: supervision, simulation, optimization, water quality management, and machine learning. These other units in the Waternet system were

developed by other partners of this project. The complete list of partners included: ESTEC, UNINOVA, SEBETIA, WBE, University of Amsterdam, ADASA, Smas-Sintra, Universitat Politecnica de Catalunya, University of Naples, and ALFAMICRO.

- ☞ Development a strong integration system for distributed information management to properly support the data exchange and information sharing among the different units of the Waternet system.

Our design and development of the innovative integration architecture and framework achieved within the Waternet project, present an implementation prototype of a federated environment, which allows different subsystems of a typical water company to work in collaboration. Regarding the information management approaches presented in chapter 2, the Waternet framework can be considered as an implementation prototype of the node-to-node federation, in which import/export schemas are well defined and node autonomy is preserved. The designed approach although addressing the Waternet requirements in specific, is generalized enough to be applied to other complex application environments, that involve the interoperability among heterogeneous and autonomous subsystems.

3.1 Introduction

Water supply industries nowadays lack a global overview of the status of the production and the water distribution system. Distinct functionalities required in this industry, e.g. optimization, water quality, etc. are supported by independent, heterogeneous, and autonomous subsystems. Each subsystem performs its specific activity, but their co-working and complex information exchange needs to be properly supported. Typically, there is none or little coordinated control in order to assure a continuous supply of water with a better quality monitoring, minimize the costs of exploitation, meet the quality standards, save energy consumption, optimize pipeline sizes, and reduce wastes. Furthermore, these systems are heterogeneous and of different levels of automation and reliability.

The main focus of the WATERNET project is two-fold: (1) the development of several subsystems performing the necessary functionalities (i.e. the supervision, the simulation, the machine learning, the models manager, the optimization, the remote unit, and the water quality); and (2) the integration of these subsystems into a coherent environment, in which the subsystems can easily access and exchange the information they need from the other subsystems, in order to function properly.

The integration/interoperation architecture designed for WATERNET, involves the development of Distributed Information Management system (DIMS) for every subsystem, that provides all mechanisms necessary for such interactions among the subsystems. Considering the fact that the DIMS plays the role of the interlocutor/integrator among all other subsystems, its implementation must reflect the inter-operation requirements specific to the design of WATERNET system and its subsystems. The two main requirements to be considered involve:

1. The need for provision of the information produced by every subsystem, for access by any other subsystem.
2. The general requirements of “openness and flexibility” to support the WATERNET system life cycle.

Subsystems in Waternet cooperative environment are independent and autonomous modules developed by different individual partners within the community. The best approach

to support point (1) above, while preserving subsystem autonomy, is the federated database approach. The PEER federated database system is used for the integration of subsystems' information through their DIMSs, and properly supports this point. The PEER federated information management system developed by the CO-IM¹ group of the University of Amsterdam is used for the development of the Distributed Information Management System (DIMS) layer for every subsystem in the WATERNET environment. The federated schema management of PEER [TA 93] employs a common means for information representation; namely, a common object-oriented schema representation that acts as the “mediator” representation of all existing information within the subsystems.

In order to support point (2) of openness above however, in addition to the federated information management, we have chosen an integration mechanism and approach, that develops *Data Adapters* for every subsystem. The primary role of Data Adapters is to provide specific interfaces for the input/output data used by every subsystem program from/to the information representation in the common mediator schema in its DIMS. This mechanism in turn supports the openness of the system as an environment, to which different functionalities can be simply added or removed, as required by any cooperative environment, in order to adjust to its specific needs.

This chapter first briefly describes the WATERNET infrastructure and its main components and then addresses the architecture and mechanisms developed for the information integration in WATERNET system. Furthermore, the chapter describes how the integration architecture supports the required openness, flexibility, and future expansion requirements for the water management systems. In order to provide a better view of how the information is represented and exchanged between PEER nodes, several examples are provided in the chapter. These examples show how the information that is stored for a given subsystem, can be imported and used by another subsystem within the cooperating network.

The remainder of this chapter is organized as follow. Section 2 addresses the description and analysis of the water management environment and presents the logical architecture for the water supply network. Section 3 describes the information management approach and designs the federated architecture for Waternet system. A general and open implementation framework for the distributed WATERNET system is described in section 4. This framework includes a brief description of the PEER distributed/federated system. Section 5 describes the main integration architecture of WATERNET supported by the PEER federated system, in which the information sharing and data exchange is supported through the integrated schema; in this section, an extensible integration approach supporting systems flexibility and application modularity through the use of the “adapter” components is also presented. Finally, section 6 concludes the chapter and enumerates the major characteristics and benefits of the extended federated integration/interoperation approach developed for the WATERNET system.

3.2 Water Environment and General application requirements

Water industries today require the cooperation of heterogeneous subsystems (also called units in this document). Each subsystem, e.g. optimization, water quality, simulation, supervision, machine learning, etc. performs a distinct function and their co-working and information exchange are of complicated nature. In principle, a number of activities may

¹Cooperative Information Management Group (<http://www.science.uva.nl/~netpeer>).

be assumed by every subsystem. Clearly, the number of units and the complexity of every system depend on the size and functionalities of the water industry. In Europe, water industries constitute a wide range, for example as small as a company where all modules run on a single system that is located in the control room of its headquarter, or as large as a water company with geographically distributed control, processing, and distribution sites.

Independent of the size, water companies today *lack a global overview* of the status of production and of the water distribution network. Control of such systems, is often carried out locally, based on the operators' experiences. Typically, there is none or little coordinated control, that is needed to assure a continuous supply, meet the quality standards, save energy, optimize pipeline sizes and reduce wastes [URB 97].

A good System analysis begins by capturing the requirements of an application, and modeling the essential elements in its environment. To support the requirements of complex applications, such as Waternet, the information management system must deal with heterogeneous sources of information from geographically distributed sites. The interconnection among the cooperating nodes is established through a variety of wide area (WAN) and local area (LAN) networks, in which a node (agent in the community) may need to access, in run time via remote queries, one or several sources of information in other nodes' data sources. In general nodes can be independent and self-serving with a large variety of data that they generate and handle. Therefore, any assumption in the direction of centralization and replication of data or unification of data descriptions (schema) in different nodes is unrealistic. Namely it is preferred to have no global schema or redundant storage of data in the network.

Subsystems involved in water management system are heterogeneous and geographically distributed. In this network, every subsystem constitutes one such unit. Every unit serves a specific function in the integrated system and thus units are intrinsically of different kinds [BA 98b, BAG 98].

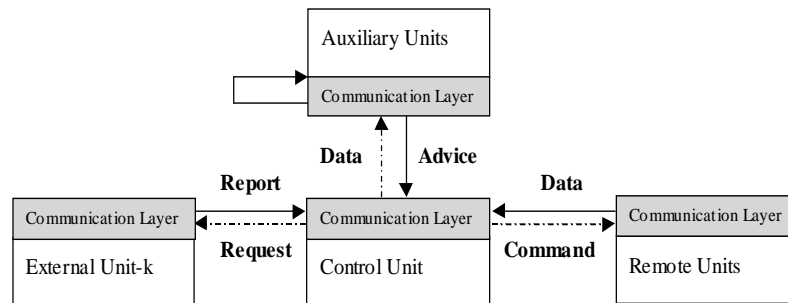


Figure 3.1: Logical Units for the Waternet System architecture

Through the analysis of the existing water supply and management networks [BAG 98] we have identified and classified the heterogeneous and distributed water management subsystems into four categories of units, namely: *Control Unit*, *Remote Unit*, *Auxiliary Unit*, and *External Unit* (see Figure 3.1):

- ① The *Control Unit* performs some central supervision and control of the water supply and distribution system. Usually, under the normal conditions, the Auxiliary Units can only “suggest” certain actions to the Control Unit, but the latter will make the final decision.
- ② The *Remote Unit* represents the concept of a site where the information is gathered from a set of sensors and control devices.

- ③ The *Auxiliary Unit* is a unit that complements the work of the Control Unit with other functionalities. Examples of Auxiliary Units include the machine learning unit, simulation unit, optimization unit, and water quality monitoring unit. These units will read the information from the Remote Units and/or the Control Units, and give the proper feedback in terms of certain suggestive actions (commands or parameter modifications) in order to achieve a better performance.
- ④ The *External Unit* is a unit that typically functions outside the water management system, but needs to be accessed to provide some information and/or services. Examples of external units can include: the geographic information systems that could be provided by some services provider, the companies that can perform water network device maintenance, and the centers that can collect users complaints.

3.2.1 Water Network Structure and Management

Existing systems for control and monitoring of water production and distribution are heterogeneous and of different levels of automation and reliability. In such a cooperative environment, the proper functionality of all subsystems involved in the water control system depends on the *sharing* and *exchange* of data with other subsystems. Typically, direct connection between these subsystems is none existent or at best exists as point-to-point, where the exchanged information for example consists of documents, phone calls, and electronic mails. Some earlier publications address the problem of developing an infrastructure and/or mechanisms to support the systematic sharing and exchange of information [URB 97, Wang 97, CQ 97], but the suggested solutions still lack a coherent environment to provide a global overview of the status of water production and the water distribution, an integration strategy for the considered subsystems and their activities, and support for the *openness and flexibility* requirements [AWH 94].

Figure 3.2 illustrates an example of a real water environment as designed and validated by the Waternet partners. In such an environment, the Waternet units mentioned above (control unit, remote units, and auxiliary units) are interconnected through a communication Intranet network via an Application Program Interface (API) that ensures data exchange and data security. Each unit (node) in the system has the full autonomy on its local data, can export a part of its local information, and can import some information that is exported by other nodes.

Following are the different kinds of subsystems that are considered necessary for the development of the WATERNET system [BAG 98]:

1. ***Remote Unit Subsystem:*** remote unit represents the concept of a site where the information is gathered from a set of sensors and control devices, and some local control is executed. Every remote unit keeps track of the local information of the site (basically device information, status readings, alarm events, and commands) and is able to handle some local events by itself.
2. ***Supervisory Subsystem:*** supervisory element performs some central supervision and control of the water supply and distribution system. In some cases, there could be only one supervisory subsystem in the network, but then some level of fault-tolerance needs to be implemented. However, it is also possible to have multiple supervisory systems distributed along the network. Usually, under the normal conditions, only the supervisory system makes the final control decisions in the system to modify the behavior of a Remote Unit. In this case, the other units (e.g. simulation, optimization,

etc. described below) can only “suggest” certain actions to the supervisory system, but the latter will make the final decision at the end.

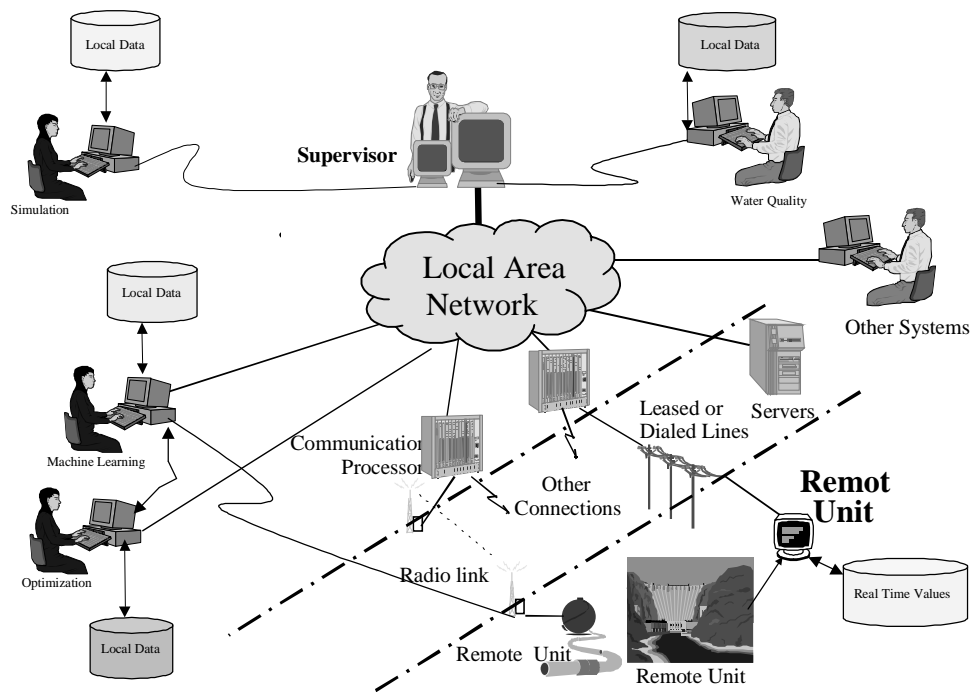


Figure 3.2: Water Management Environment

The functionalities of the supervisory system include:

- a- **Planning:** planning is a daily process that allows a supervisor at the control headquarter to define the set of actions (production plans) to be taken for the coming hours, days, or weeks. The production plans are executed at fixed schedules and according to the set of pre-defined steps of each action. Planning also includes the set of parameters to be kept in the network and their range of values. In order to achieve a good water supply, two sets of results, described below, can be used: the water demand forecast generated by the Machine Learning subsystem and the optimized strategy proposed by the optimization subsystem.
- b- **Controlling:** controlling is the main regulatory task of the water management network, it focuses on three operations: choosing a plan and making it the practical strategy for remote units, manipulating the devices, and adjustment of set points. Such a process allows to recognize failures in the system, to identify the non-optimized operations, and to take the proper recovery actions.
- c- **Monitoring:** monitoring is an important process in water distribution network since it watches at the system by reading values of all devices and checks if everything runs properly (parameters should be within the defined range limits, default values, rules checking for pressure, level, flow, etc.). The monitoring system can monitor all remote units for the company regarding their actual running status and collected information, periodic readings of network devices, alarms, and

the sensor values about the pressure, flow, and quality for all remote units. The monitoring interface supports the browsing of Network Current Status, Historical Data, and Graphic Display of the device information and statistical analysis.

- d- **Alarms Handling:** once an alarm is detected it will be presented to the system supervisor at the control room, who shall then make an expert decision on how to react properly. In this case the knowledge/rules extracted by the Machine Learning subsystem (described below) can be used in order to help the supervisor to react properly, taking the suggested rules into consideration [CM 99].
3. **Simulation Subsystem:** simulation assists the operator if he/she decides to look forward in time (e.g. for a few hours) to spot potential problems that can develop if the network is not monitored aggressively. Finding such problems can be supported through the use of the most up-to-date consumption forecasts for the network. In this case, the simulation process looks at what will happen during the next hours, with the goal to spot the eventual problems before they actually develop and occur, a set of simulated network results will be produced by this subsystem and presented to the supervisory system
 4. **Machine Learning Subsystem:** that complements the work of the supervisory subsystem by other functionalities [CM 97]. Two activities are supported by the machine learning subsystem.
 - a- The *Knowledge Extractor* process that uses the network model information and the historical data for knowledge and rules extraction, in order to be used by the supervision system for an advanced monitoring of the system.
 - b- The *Water Demand Forecasting* process that needs to extract some knowledge to be used in forecasting future water demand, and giving information on how the network is evolving. Its objective is to predict the water consumption for a region of the network in the near future [CM 99].
 5. **Optimization Subsystem:** optimization in the water networks refers to the optimized operational strategies for the elements controlling water transfer in the network such as the pumps or valves related to cost, quality, etc. The optimized operational strategies are based on the forecast of future demands over the time horizon using a simplified model of the water network's dynamic behavior.
 6. **Water Quality Subsystem:** quality in the water management comprises a large set of parameters, however the most important are related to the quality of supply (pressure, flow, continuity, etc.) and biological characteristics. The quality monitoring process gets actual values of the sensors for quality measurement from the supervision system and then it generates a list of possible abnormal situations for which a set of alarms will be generated and presented to the supervisor at the control room.
 7. **Other External Subsystems:** external subsystems may run outside the water management system, while they are needed to be contacted in order to provide information/services necessary for water distribution. For instance, the geographic information systems and/or the water network maintenance systems that can be contacted by the supervisory subsystem or others, when their information/services are required.

3.3 Information Management Approach

In water supply and distribution network, typically the information about the water characteristics and network devices, is gathered in remote units and processed at different stages of network simulation, network behavior learning, strategy optimization, and water quality checking. Furthermore, the proper planing and strategies for water management and processing untreated water are achieved under the supervision of the system supervisor.

Units involved in the water control network (e.g. supervision, simulation, optimization, machine learning, and water quality) function properly if and only if they can access the information produced by other units. Therefore, the sharing and exchange of information among subsystems must be properly supported, while the proper independence and autonomy of the units needs to be also preserved. For instance, the control unit, or an external unit, are autonomous units, while the remote unit has only partial control over its functionality and takes orders from the Control Unit. Similarly, the heterogeneity of information representation in different units and its varied classification needs to be supported. In general, the same piece of information is viewed differently by two units, and different levels of details can be associated with it [URB 97, RPR⁺94, SP 94]. The database schemas involved in the definition of the Waternet subsystems are described in details [BAG 98].

Some earlier publications have addressed the problem of data representation and information modeling for the operational control of water distribution systems [Wang 97, CQ 97]. However, they mostly lack a comprehensive approach that involves the entire set of components and their activities, and takes into consideration the distribution and evolution of the system. In general, subsystems are independent and self-serving, with a large variety of data that they generate and store. Therefore, any assumption of centralization, replication, or unification of data descriptions in different subsystems (through one global schema) is unrealistic. It is preferred to have no centralized global schema or redundant storage of data within the entire network.

3.3.1 The Waternet Architecture

In order to support the complex information management requirements in water environments and their applications, we have designed a comprehensive architecture for the Waternet system. Within this Architecture, presented in Figure 3.3, the integration among components and their information exchange is clearly defined and represented. Some earlier publications address the problem of data representation and data modeling for the operational control of water distribution systems [Wang 97, CQ 97]. But, they mostly lack a comprehensive approach that involves the entire set components and their activities, and takes into consideration the distribution and evolution of the system.

The information management architecture of the water network illustrated in Figure 3.3 is a general and comprehensive architecture that supports the autonomy and heterogeneity of information representation in all sites involved in the water management. In practice, the case of every water company is different and may require only a subset of this comprehensive and open architecture. Here the purpose is to define an architecture that is capable of handling an advanced federated control network.

The DIMS (Distributed Information Management System) is augmented with every unit in the network in order to support the capability of information sharing with other units in the Waternet network in a transparent way. Therefore, the DIMS layer ensures the runtime access to information stored in other subsystems (via remote queries). The DIMS

information management is supported via the federated schema facilities and the federated query processing of PEER. Moreover, DIMS extends the PEER approach with the data adapters mechanism, in order to better support the information exchange between the PEER system and the specific information systems used at each unit in Waternet (see section 3.5.1).

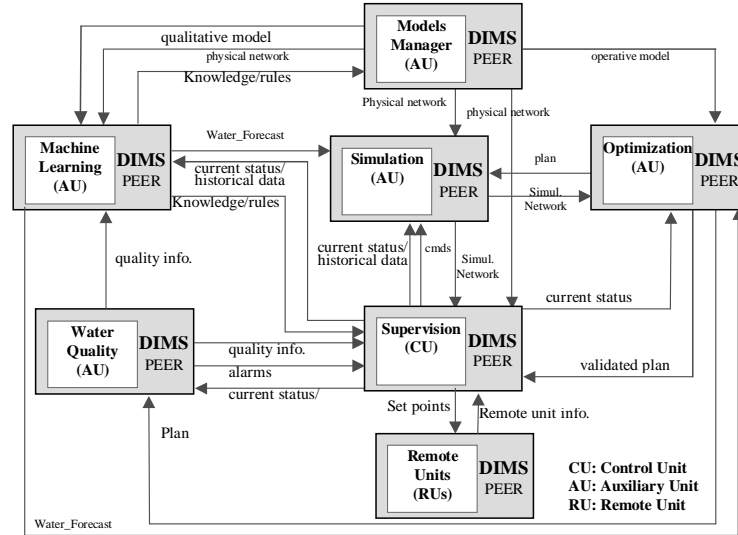


Figure 3.3: Information Management Architecture for the Water Network in Terms of Units

The federated schema definition facilities provide the capability of information sharing with other units in a transparent way. This also implies that every such unit can handle different kinds of information: the information which is going to be stored locally, the sharable information for public access, and the information which needs to be imported from other units. Consequently, through the federated query processing, from the user point of view the access to physically distributed information along the network is the same as a local access. In general, the *Control Units* and *Auxiliary Units* may retrieve the information from other different sites, and are in some cases able to provide suggestions/strategies to improve the behavior of the *Remote Units* and *Control Units*. Every unit will read the information, which is required from the other units, at the time that is needed. Therefore, the information that is accessed from other units is always up-to-date and there is no repetition of information among the units in the network.

3.3.2 Simple Scenario for Subsystems interaction

The WATERNET system operation requires a real cooperative environment in terms of the integration and the exchange of information between different subsystems. In order to give the reader an overview on the complexity of the interactions between the WATERNET subsystems, for data sharing and some results validation, a simple scenario for the process involved in developing an optimized strategy is presented in this section.

The cooperative work required to develop an optimized strategy can be considered as *"a part of the bigger cooperative environment required every day"*, to identify many operations to be carried out the next day. As depicted in Figure 3.4, the cooperative process needed for the simple scenario involves the optimization, machine learning, simulation, water quality,

and the supervision subsystems. The steps involved in the execution of the scenario are described below:

- **First**, in order to generate a *management plan* for the next day operation of the network, the *Optimization* asks the *Supervision* for the network devices information; asks the *Machine Learning* for the forecasting results; and asks the *Models Manager* for the operative model. The management plan generated by the optimization subsystem, primarily consists of a sequence of commands to be performed at specific times on the network, e.g. opening a valve at 2 AM, stopping a pump at 5 PM, and so on.
- **Second**, the *Simulation* and the *Water Quality* subsystems are invoked by the optimization. These two subsystems must access the *generated plan* information from the *Optimization* subsystem, perform some processing and give their feedback about the consequences of the *generated plan* on the ability of the system to support the proposed plan and/or how this plan affects the quality of the water.
- **Third**, the *Optimization* subsystem needs to access both the *Simulation* and the *Water Quality* subsystems, in order to check their evaluation results of its earlier generated plan and in order to decide either to recommend the plan to the *Supervision* as an *optimized plan* or to reject it. If the plan is rejected, the whole process described above needs to be restarted to develop and test a new plan. Otherwise, the plan will be approved and presented to the supervisor for acceptance. If accepted by the supervisor, the plan will be loaded at the remote units by the system supervisor at the control room; otherwise it will be canceled and re-planning starts again.

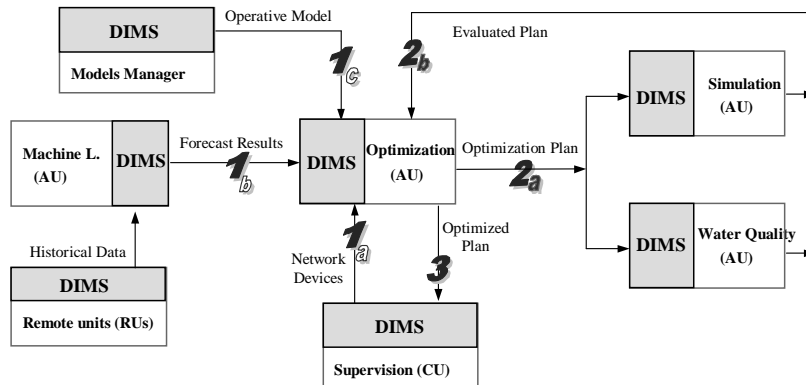


Figure 3.4: Simple Scenario for Subsystems Interaction in Waternet

3.4 Distributed Information Management System (DIMS)

In the general architecture, as presented in Figure 3.3, every component of the WATERNET system being a remote unit, a control unit, an auxiliary unit, or an external unit, constitutes a PEER node. In principal, one unit can either run on an individual workstation (or PC), or several units can run on the same system. The PEER system and the development of the PEER federated layer for DIMSs are further described in this section.

3.4.1 The PEER Federated Layer

The PEER system provides an environment for the cooperation and information exchange among different nodes in a network, where every node is composed of one server process and may consist of several client processes. The federated schema management and the federated query processing of PEER [AWT⁺94] support the sharing and exchange of information among nodes, without the need for data redundancy and/or creation of one global schema. Therefore, the problems of data consistency, referential integrity, and updates propagation are eliminated.

The federated schema management of PEER organizes four different kinds of schemas for every node (Figure 3.5). The local data at the node is defined by the schema called LOC. Every node can create other several schemas called exported schemas (EXP), to represent a part of its local schema (LOC); and only authorized users at remote nodes can access data of this node through some EXP schemas. The authorized nodes can import the EXPs of other nodes that will be called imported schemas (IMP). The imported schemas (IMPs) are then merged with the LOC to build the integrated schema (INT) for the node. Hence, every node in the federated community can access both its local and the remote information (from other nodes) through its INT schema, as if all the data is local information. At the same time, the physical and logical distribution of information becomes completely transparent to the users. The four kinds of schemas for the subsystems are defined using the SDDL (Schema Definition and Derivation Language) of PEER [TA 93]. Several examples of these schemas defined for the WATERNET subsystems are included and described in earlier publications [ABH 98b, BAG 98].

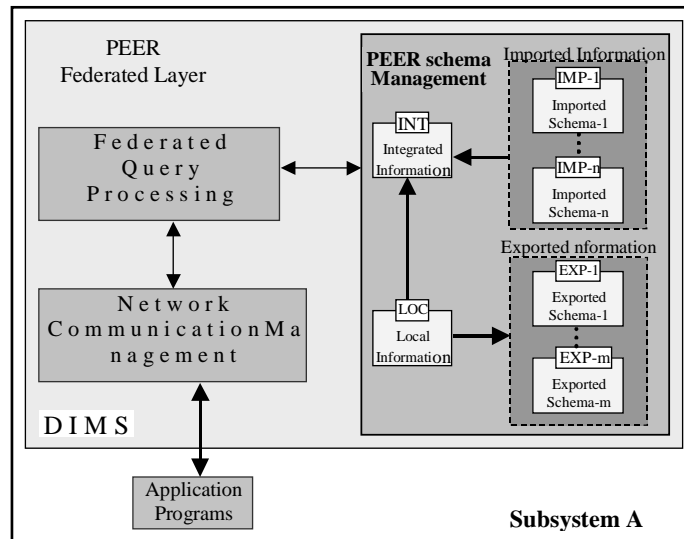


Figure 3.5: PEER Federated Layer Representation

The PEER layer development for every unit, supports the following features:

- Integration and filtering of information accessed from a set distributed units.
- Support for local autonomy and heterogeneity at every unit.
- Access to updated data with no redundancy of stored information.

- Flexible support for potential network expansion with new functional units.
- Differences in data structures, modeling approaches, and objects naming in different nodes are solved through the definition and derivation of the integrated schema.

3.4.2 Schemas Management in WATERNET Using PEER

To provide a better idea about the information that is represented and exchanged between PEER nodes, simple examples are provided below. In general, these examples show how different pieces of information that are stored at a given node, can be imported and used by another different node. What is represented in these examples is a small part of the PEER schemas developed for certain units in the implementation of the Waternet system. The definition of the schemas in these figures is based on the use of the Schema Definition and Derivation Language SDDL of PEER [ATW⁺93, AWT⁺94].

Table 3.1 shows a part of the LOC schema of the Control Unit CU. In order to export some information from the CU to the auxiliary Units (or in general, to any other unit), one or more export schemas need to be defined at the CU node. The export schema EXP1 defined at CU (Table 3.2) contains network devices information in *Nodes* (subtypes *E_Tank* and *Reservoir*) and *Devices* (subtypes *Pipe*, *Group*, and *Valve*) types. This information is derived from the LOC schema of CU and due to the preference of the autonomous CU node, it is defined rather differently than in the corresponding LOC schema type definition. For instance, the type *Head_Dep(ending)* is not exported in EXP1 from the local schema of CU, as well as several attributes such as *low_level_alarm* and *high_level_alarm* in *Tank* and *broken* in *Network_Devices*.

```

define_schema LOC type NETWORK_NODES
  code_, site_code_: STRINGS
  coord_x, coord_y, coord_z: REALS
  outflow: INTEGERS
type TANK subtype_of NETWORK_NODES
  area: STRINGS
  low_level_alarm, heigh_level_alarm: REALS
  weir_elevation: INTEGERS
  inflow: REALS
type RESERVOIR subtype_of NETWORK_NODES
type HEAD_DEP subtype_of NETWORK_NODES
  coef_discharge: REALS
type NETWORK_DEVICES
  start_node, end_node: NETWORK_NODES
  discharge: REALS
  broken: BOOLEANS
type PIPE subtype_of NETWORK_DEVICES
  pipe_length, pipe_diameter: REALS
type GROUP subtype_of NETWORK_DEVICES
  power: REALS
type VALVE subtype_of NETWORK_DEVICES
  open_time, close_time: DATE
end_schema LOC

```

Table 3.1: Simple Network Local Schema in Control Unit Node.

Other attributes have been exported, sometimes with different names or under different types (i.e. a subtype). Thus, through the attribute transformation, it is possible to export information with a different representation that is more simplified to better support other

units' purposes. Table 3.3 and Table 3.4 represent two schemas in the optimization unit, and in a specific its LOC schema and the IMP7 from the Control Unit (CU) respectively.

<pre> Derive_schema EXP1 From_schema LOC type NODES code_id: STRINGS code_site: STRINGS x, y, z: REALS flow: INTEGERS type E_TANK subtype_of NODES hq2: STRINGS elevation: INTEGERS inflow: REALS type RES subtype_of NODES type DEVICES start_node: NODES end_node: NODES discharge: REALS type PIPE subtype_of DEVICES pipe_length, pipe_diameter: REALS type GROUP subtype_of DEVICES power: REALS type VALVE subtype_of DEVICES open_time, close_time: DATE derivetion_specification ----- </pre>	<pre> NODES = NETWORK_NODES@LOC code_id = code_@LOC code_site = site_code_@LOC x = coord_x@LOC y = coord_y@LOC z = coord_z@LOC flow = outflow@LOC E_TANK = TANK@LOC hq2 = area@LOC elevation= weir_elevation@LOC inflow= inflow@LOC RES = RESERVOIR@LOC DEVICES = NETWORK_DEVICES@LOC start_node = start_node@LOC end_node = end_node@LOC discharge = discharge@LOC PIPE = PIPE@LOC Pipe_length = pipe_length@LOC Pipe_diameter = pipe_diameter@LOC GROUP = GROUP@LOC Power = power@LOC VALVE = VALVE@LOC Open_time = open_time@LOC Close_time = close_time@LOC end_schema EXP1 </pre>
--	--

Table 3.2: Simple Network Export Schema (EXP1) in Control Unit Node

<pre> define_schema LOC type OPT_VALVE device_code: STRINGS site_code: STRINGS min_flowrate: REALS max_flowrate: REALS min_position: REALS max_position: REALS type OPT_PIPE opt_coeff: REALS type OPT_GROUP opt_coef: REALS min_pos: REALS max_pos: REALS type OPTIMIZATION_NODE opt_name: STRINGS opt_address: STRINGS end_schema LOC </pre>	<pre> Define_schema IMP7 same_as_export_schema EXP1 from_agent Control_Unit type NODES code_id, code_site: STRINGS x, y, z: REALS flow: INTEGERS type E_TANK subtype_of NODES hq2: STRINGS elevation: INTEGERS inflow: REALS type RES subtype_of NODES type DEVICES start_node, end_node: NODES discharge: REALS type PIPE subtype_of DEVICES pipe_length, pipe_diameter: REALS type GROUP subtype_of DEVICES power: REALS type VALVE subtype_of DEVICES open_time, close_time: DATE end_schema IMP7 </pre>
<p>Table 3.3: Simple Local Schema (LOC)</p>	<p>Table 3.4: Simple Imported Schema (IMP7)</p>

The Optimization Unit will access some up-to-date and current information from the Control Unit through the import schema IMP7. An import schema (IMP) always has the same structure as the definition of its corresponding export schema (EXP) at its origin. For every import schema, the name of the node and the name of the export schema from that node are specified.

Finally, as shown in Table 3.5, the Optimization Unit will define an integrated schema, derived from its local schema and other imported schemas, such as IMP7, using some types and maps derivation operators (e.g union, restrict, subtract, rename, threading, etc. from the SDDL language of PEER). However, for simplicity reasons, here the examples only show the UNION operation. The INT schema represents the proper database view for the optimization applications and optimization programs. Once the integrated schema is defined and created, the user at the Optimization node can formulate his queries against this global and complete schema that represent an overview of all the information accessible from this site. Based on the integrated schema definition, when a query arrives, it will be decomposed into several sub-queries, each related to a different remote node where the needed information is available. The result from different remote queries will then be merged with the local one and then presented to the user as a coherent response for his request.

<pre> Define_schema INT from LOC, IMP7 Type OPTIMIZATION_NODE opt_name: STRINGS opt_address: STRINGS type NODES node_id: STRINGS node_site: STRINGS x, y, z: REALS type ARCS start_node: NODES end_node: NODES discharge: REALS type OPT_VALVE subtype_of ARCS open_time: DATE close_time: DATE min_flowrate: REALS max_flowrate: REALS min_position: REALS max_position: REALS type OPT_PIPE subtype_of ARCS pipe_length: REALS pipe_diameter: REALS opt_coeff: REALS type OPT_GROUP subtype_of ARCS power: REALS opt_coef: REALS min_pos, max_pos: REALS derivation_specification ----- </pre>	<pre> OPTIMIZATION_NODE = OPTIMIZATION_NODE@LOC opt_name = opt_name@LOC opt_address = opt_address@LOC NODES = UNION (E_TANK@IMP7, RES@IMP7) node_id = code_id@IMP7 node_site = code_site@IMP7 x = x@IMP7 y = y@IMP7 z = z@IMP7 ARCS = UNION (PIPE@IMP7, GROUP@IMP7, VALVE@IMP7) start_node = start_node@IMP7 end_node = end_node@IMP7 discharge = discharge@IMP7 OPT_VALVE = UNION (OPT_VALVE@LOC, VALVE@IMP7) open_time = open_time@IMP7 close_time = close_time@IMP7 min_flowrate = min_flowrate@LOC max_flowrate = max_flowrate@LOC min_position = min_position@LOC max_position = max_position@LOC OPT_PIPE = UNION (OPT_PIPE@LOC, PIPE@IMP7) pipe_length = pipe_length@IMP7 pipe_diameter = pipe_diameter@IMP7 opt_coeff = opt_coeff@LOC OPT_GROUP = UNION (OPT_GROUP@LOC, GROUP@IMP7) power = power@IMP7 opt_coef = opt_coef@LOC min_pos = min_pos@LOC max_pos = max_pos@LOC end_schema INT </pre>
---	--

Table 3.5: The Integrated (INT) Schema in Optimization

3.5 Extended Integration Approach

An important outcome of the DIMS integration approach is that any subsystem in WATERNET can develop its application programs without the need of knowledge about the format, structure, and/or location of the data produced somewhere else in another subsystem.

Figure 3.6 represents the main integration architecture of WATERNET based on PEER, in which each subsystem within the WATERNET is augmented with its DIMS (a PEER-based federated layer). Within the DIMS layer, the information sharing and data exchange is supported through the integrated schemas. Using the provided mechanisms, users and application programs in a subsystem can specify the queries for data retrieval or data insertion through the subsystem's integrated schemas. The defined queries can be specified on line by human users, using the on-line PC-interface, or within an application program using the programming languages-interface. Once a query arrives, it will be decomposed into several sub-queries. The query decomposition is based on the definitions in the integrated schema at the DIMS layer. Each sub-query is then sent to the proper remote subsystem. Finally, the local partial-result for the query is merged with the remote partial-results, and produces the complete result to the query, that will be presented to the end-user and appears just the same as if it was handled completely local at this node. The approach described above allows a complete integration of the data stored in different subsystems in a transparent way, while preserving access security issues, and the execution of concurrent transactions.

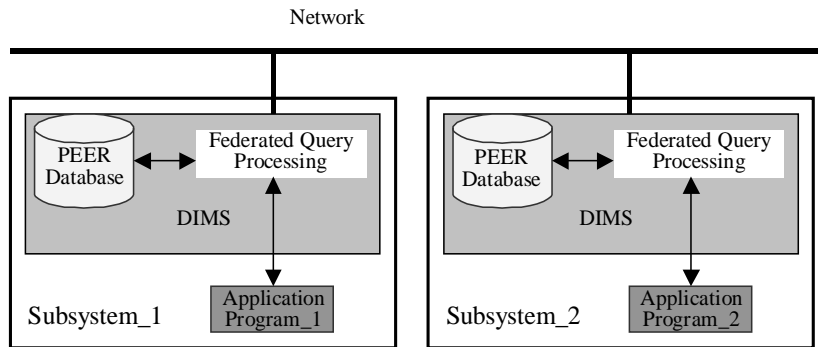


Figure 3.6: Basic Integration Architecture

As represented in Figure 3.6, an application program within a subsystem can then simply receive its input from (and similarly send its output to) the PEER database server. However, considering the specific characteristics that define every application domain, this implementation architecture may not sufficiently support all the requirements within the WATERNET environment. In specific, to support the water management applications and the wide variety of subsystems within the configuration of different water companies, in our requirement analysis stage we have identified the need for a more “open and flexible” architecture [BAG 98]. For instance, from time to time different subsystems (mostly pre-existing and some commercial, e.g. new simulation software) may need to be added to (or removed from) the WATERNET system, in order to better support the specific needs of the company. Even as a product, some of the existing subsystems may need to be disconnected from WATERNET, and/or replaced by other existing or new commercial products that run in the company. Using the federated architecture and approach as described above, these alterations within the subsystems require that subsystem developers must have database

language expertise to properly add/remove/replace the subsystems to the federated architecture. For instance, the knowledge of PEER database language commands is mandatory, to generate the appropriate PEER commands to be included within any application program written in a subsystem in order to develop the interaction between a new subsystem and its DIMS. A similar problem arises when a unit decides to use as input (for its application programs) some other resources outside the DIMSs that may be available from external applications or databases. Hence, there is a need for an open and flexible integration architecture. Under the influence of this “openness” requirement, we have extended the integration architecture of the DIMS to also include the “*Adapter*” (or data adapters) components, described in section 3.5.1.

This extensible integration approach supports the system flexibility and application programs modularity for the WATERNET subsystems. The extended approach, as depicted in Figure 3.7, through (a) preserves the main properties for cooperative working in multi-agent environment, such as the data-location transparency, access security, transactions concurrency, etc. (similar to the architecture described in section 3.4), but additionally through (b) with the **adapters**, supports the openness requirement. Among other features, the adapters support adding/removing new subsystems within the WATERNET system that can be developed independently from the WATERNET project. Using the adapters, an application can receive its input either from the remote DIMS or from external application. Similarly, the generated output (in addition to storing it in DIMS) if needed can also be stored locally in a storage facility (or another simple database system) and made available to external applications that may not even be allowed to access and retrieve information stored in different WATERNET DIMSs. Clearly, within the WATERNET system, the data of a subsystem stored within its DIMS can always be accessed by other WATERNET subsystems through the DIMS to DIMS interconnections.

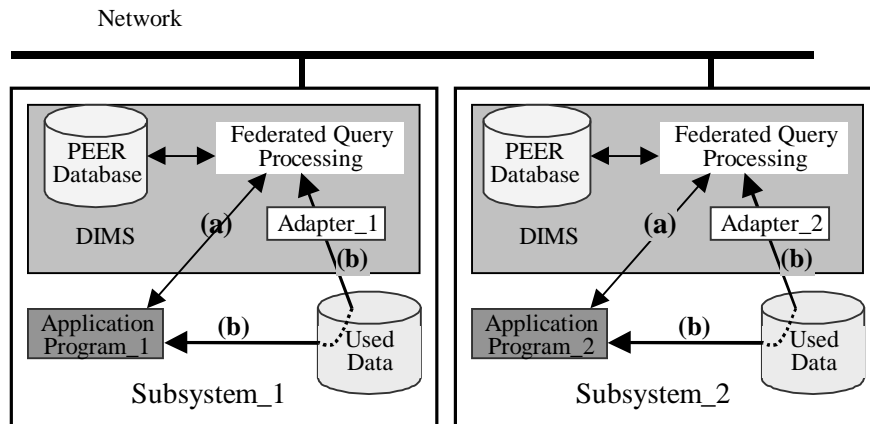


Figure 3.7: Extended Integration Architecture

The adapter framework supports the following:

- a- Provides the storage of the *exact output* of the application programs in every subsystem within its DIMS layer. In fact, a module called *pre-processor* takes the output of every subsystem’s program in the exact format that it is produced, (being a set of values, or a record, etc.), reformats it according to the object definitions in the subsystem’s LOC schema and stores it in the DIMS.

- b- Supports every subsystem's (e.g. supervision's) access to the data stored in other subsystems (e.g. optimization, machine learning, and others) in the *exact format* that is required by every application program. In fact, a module called **post-processor** provides access to imported data through the DIMS for every application program but changing its format to the exact format as desired to be read by the required application program (e.g. supervision's).

Therefore, the **pre-processor** and **post-processor** (in Figure 3.8) together provide the access *to* and *from* the PEER database for every application program in every subsystem. This mechanism in turn supports the modularity and autonomy of nodes within the cooperative community, while also supporting their desired specific application-program-dependent input/output formats for data.

3.5.1 Data Adapters Supporting Openness

The Adapter framework, as represented in Figure 3.8, provides flexibility and openness, and facilities for the development of application programs. In other words, the programs can read/write their data in the most convenient way to them. For every subsystem the adapters constitute a set of dual *pre-processor* and *post-processor* components, where each pair supports the input/output of one of its application programs.

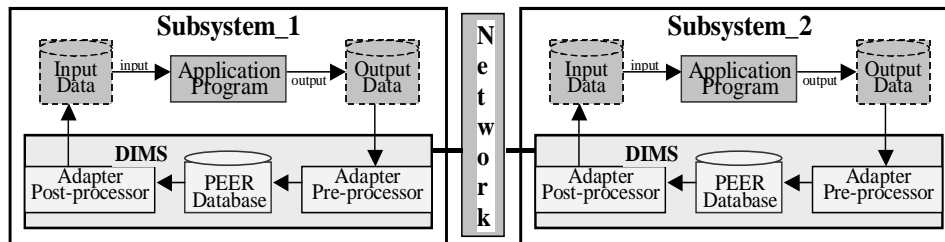


Figure 3.8: DIMS Layer – Federated Data Process using Adapters

Considering the above clarification, the DIMS integration architecture makes the development process of every subsystem (as well as adjustment to other environment configurations) very convenient, as it proved itself in practice during the development phase in the WATERNET project. Namely, every subsystem developed its application programs completely independent of the others, and it was enough to just specify to the DIMS developer the desired format for the *input* and *output* of those programs and not being concerned with how this data is produced by others. For instance, a program in machine learning subsystem produces as output "a file" for which the record format represents: " $r_1, r_2, r_3, r_4, r_5, r_6$ ". Meanwhile, for instance, a program in the simulation subsystem, reads its input from "a file" with the record format: " $r_3, r_5, n_7, r_2, r_1, d_1$ ", while here the " r_3 " need to be imported from the DIMS of the machine learning subsystem, " n_7 " needs to be imported from optimization subsystem, and " d_1 " is a computation result using different imported and local values. At the last stage the imported information and other values need to be re-arranged according to the record format required by the simulation application program.

3.5.2 The WATERNET System Implementation

The architecture designed for the WATERNET system is comprehensive enough to support different possible implementation strategies adopted in water companies. Namely, it can support a wide range of companies. For instance, it can support on one hand the case of a small water company where all modules of the WATERNET system run on a single system in the control room at the headquarter, and the remote units only send their collected data to this headquarter. At the same time, it can support a medium to large size water company with many geographically distributed control sites, even if different modules of the WATERNET system, for instance, the forecasting, machine learning, and water quality management each run on different sites and are connected only through the communication network. The PEER federated database system [TA 93], is used as the base for the implementation of the information management in the WATERNET project and supports the communication and interoperability among these subsystems. However, the PEER system was extended to better adjust to both to the specificities of the water management environment and the specific development strategies of different subsystems in WATERNET.

Some extensions enhanced the portability of the PEER federated system. For example, the development of two interfaces: the *on-line PC interface* and the *programming languages interface* for PEER [BA 98b]. Considering the facts that PEER is Unix based, while most WATERNET subsystems are developed and run on PCs, the on-line PC interface developed for PEER, allows a user to interact with any database within the cooperative community in order to check, retrieve or update the information for which he/she has gained the appropriate access rights. The Programming Languages Interface includes the necessary functions that allow programmers to develop their own programs, while interfacing with PEER through several different applications programs written in C, C++, Pascal, etc.

3.6 Conclusion and Discussion

In this chapter, a general approach for the design of an open and flexible architecture for the integration between different WATERNET system units, and the mechanisms used for their implementation were presented. The implementation of the designed architecture for the WATERNET framework is based on the PEER federated information management system, since it properly supports the cooperation and information exchange among different nodes involved in an intelligent cooperative environment. To better support the "openness and flexible" requirements in water management environments, the implementation architecture of the DIMS was extended to include the adapter framework. In addition to the main properties provided by the PEER federated layer in the DIMS implementation, the extensions with adapters provide among other features: (1) support for the systems expansion (addition, removal, or replacement of subsystems), (2) the adjustment to subsystems evolution (new/modified application programs), (3) the use of external media (resources from external application) as the input information, and (4) the storage of generated output in a different media, in order to be made available to external applications that may not even be allowed to access the information stored in different DIMSs within the community.

The development of the Waternet project has provided a good environment for implementing a prototype of a federated environment. In this environment an open architecture for distributed/federated information management system was designed. The development of the federated DIMS system for Waternet was based on the detailed study and analysis of the water network production environment. The designed architecture and the approach de-

scribed in this chapter, or a substantial part of it, can be applied to any other manufacturing and production application domain, in which several heterogeneous and some autonomous nodes need to cooperate and exchange their information.

3.6.1 Major Characteristics and Benefits of Federated Approach in Waternet

The federated schema management and federated query processing mechanisms of PEER, in addition to the adapters framework presented in the sections above provide a flexible, and an open environment for the development of a strong water management system. Following characteristics resulted in this environment represent the major benefits gained from the approach taken in the project for the design and implementation of the DIMS.

- System openness, so that different modules can be added to/removed from the WATERNET system, as needed, in order to support the specificities of different water companies. This characteristic strongly supports WATERNET as a flexible product, since in order to install the WATERNET system in a company, some of its subsystems may need to be disconnected from this product, and/or replaced by other existing products that already run in the company.
- No need for the development of a single global schema for all Waternet subsystems (being centralized or distributed).
- No need for data redundancy/duplication among the subsystems (no data transmission unless needed). As a result, the problems of data consistency, referential integrity, and update propagations are eliminated.
- Complete transparency of logical/physical distribution of information among the nodes in the network, to the end user.
- Retrieved data is always accessed directly from its origin and as a result it is always up to date.
- The WATERNET development environment has become totally flexible. In fact, all subsystems continued developing *their functionalities and application programs*, while simultaneously the gradual and dynamic development of the DIMS adjusted itself to their extensions and modifications.

3.6.2 Contribution to GFI₂S

The designed architecture and the implemented approach described in this chapter can be applied to many other cooperative environments, in which several heterogeneous nodes need to interact and exchange their information. Several of the developed aspects and lessons learned, during the design and implementation of the Waternet system, are in fact conforming the base for the design of the **GFI₂S** system described in chapter 6. In specific, the DIMS architecture of Waternet contributes to GFI₂S at two levels:

- *At the local system level of GFI₂S, we adopted the approach of developing data adapters of Waternet and developed the Local Adaptation Layer component (LAL) serving the system openness.* In Waternet, similar to many other systems, the role of an adapter is to adapt and translate data from one application to another. In **GFI₂S**, we take this approach, but further extend it such that the role of an adapter is to

also include the local resources specifications. Such specifications include among other features the access rights to the data. When a query is submitted by an external user/application, the Local Adaptation Layer (LAL) of **GFI₂S** only delivers the data to authenticated users. In addition, data is delivered according to the local specifications for users access rights and their corresponding visibility levels. Therefore, the LAL in **GFI₂S** controls the access to the local system and preserves its autonomy (see section 6.2.1).

- *At the Node Federation Layer (NFL) of **GFI₂S**, we adopted the federated schema management of Waternet (local, import, export, and integrated schemas), in tackling the fundamental schema management challenges.* Additionally, in **GFI₂S**, we extend the definition of schemas by augmenting by the specification of the federated resources and the semantic descriptions. Such specifications provide a better understanding of the exchanged information and facilitate the development of schema integration mechanisms (see section 6.2.2).

Furthermore, we must also admit to the fact that during the development of different federation components of the Waternet system, we faced two major obstacles, which made the development of the system requiring a lot of efforts. On one hand, the knowledge and expertise of the PEER system was needed at the development phase of the various Waternet subsystems. On the other hand, development of the adapter components was specific to every legacy system used at the components. Therefore, along with the development of the Waternet System, for each new module, adapters needed to be developed.

In chapter 6, these two issues will be addressed in the design and development of the integration architecture of **GFI₂S**.

- To solve the first issue, related to specific languages, we suggest the use of standard languages for data definition and information access (e.g. ODL, SQL, and OQL). These standard languages are widely understood by a large community, thus, less effort will be needed when defining federated schemas and accessing data through them.
- To solve the second issue, we use middleware and standard solutions for information exchange, and communication protocols, serving the need and requirements of openness and extensions. The use of middleware solutions play an important role in reducing the number of intermediate tools, unifying the access to shared information, and facilitating the interoperation process among heterogeneous units.

Chapter 4

MegaStore: Advanced Web Databases for Music Industry

4.1 Introduction

The MegaStore¹ system described in this chapter, aims at the design and set-up of the necessary database structure and platform architecture for advanced e-commerce applications, and in specific addresses the CD and music industry. Unlike most existing systems, the database design of MegaStore is general enough to include additional information about the music. Furthermore, the search engine can benefit from the additional stored information about the music composer/performer and lyrics for the titles, among others. Similarly, the storage of complete audio/video clips can serve for a future extension to this system towards what is known as *on-line Music from the Wall*, with a small effort and reduced cost.

☞ From the usage point of view, the Internet-based CD shopping system, called the *Virtual MegaStore*, consists of a *front-end* system with two main components. The first component being the *Internet-Shop*, that can be accessed by all Internet users and the second component constituting a so-called *Shop-in-a-Shop*. The *Shop-in-a-Shop* interface can be installed inside an existing physical music store and it offers the store keepers the unique and strong ability to immediately and at the run-time respond to the requests of customers visiting the music shop, through downloading the raw music data and producing CDs tailored to the customers requests.

☞ The MegaStore *front-end* system is based on a *back-end* component that includes a distributed object-oriented database and a high performance server architecture. The database supports geographically distributed multi-media information and the designed extensible server architecture assures the required high data transfer rate and the short response time for on-line requests.

The proposed system architecture best suits the e-commerce application, by separating the public and general information from the private information, and supports the large data sets that need to be securely kept at predefined Internet sites. Furthermore, the necessary inter-stores communication requirements are studied and supported based on each identified activity within the system.

¹The Virtual MegaStore project has been supported by the Dutch HPCN foundation whose partners are the University of Amsterdam, the Frame Holding BV, and the International Music consortium BV.

4.1.1 E-Commerce Applications: Attempts and Aims

To support necessary requirements and flexibility to the buyers of different goods, advanced and efficient internet-based Electronic Commerce (E-Commerce) services must be designed and developed. In addition to the traditional user requirements for every application environment, the new developed system must properly address several efficiency and organization related issues, among which the data catalogues and information classification, short response time for on-line requests, high system performance, and high data transfer rates must be considered.

One major technical problem hampering the realization of suitable implementation for electronic shopping is the lack of possibilities to integrate a wide variety of data in a single coherent environment, which bases on a comprehensive system architecture and advanced database technologies [BAH 00, BAH 99, Atz 99, Atz 98, BBH⁺99]. In the context of web-based systems, several approaches have been investigated and worked out during the last few years. These approaches cover different domains of interest and address several application requirements. As such, these applications address different domains ranging from simple search engines that allow users to find information of their interest using a user friendly interfaces [NK 97, MMR 97] to advanced systems that manipulate multimedia information taking into account the emerging Internet technologies [BBO⁺99a, AAC⁺99, MHH 97]. Nowadays, more e-commerce applications embracing electronic shopping via virtual stores are also emerging (e.g. CD Now² and Amazon³). However, Web-related systems are still involving the development of a large number of tools for data manipulation [BEM⁺98, FGN 98, Frt 99], which require a lot of efforts for their internal maintenance and operation. The work described in this chapter overcomes some of these problems and addresses the need to provide the user of electronic shopping with an environment through which he can experience as sufficiently close to real life shopping experience.

The structure of this chapter is organized as follow. In section 4.2, the music industry application is studied and analyzed, and the necessary requirements are identified. The section also describes the design of complex music library information, and accomplishes the MegaStore base schema structure using the UML notation and the ODL definition. Section 4.3 focuses on the general design of the server architecture for the MegaStore system, and mainly outlines the necessary requirements for: the *Internet-Shop* interface, the *Shop-in-a-Shop* interface, and the server architecture extension. In section 4.4, a brief descriptions of different audio/video music formats supported by the MegaStore system are presented, while section 4.5 addresses the manipulation of music data contents and their storage mechanisms. Section 4.6 describes the MegaStore advanced features and outlines the current implementation status of the system. Section 4.7 presents the Luisterpaal interface and the Music Sheet application; two applications derived from the MegaStore framework. And finally, section 4.8 concludes the chapter and introduces some possible extensions to the system.

4.2 Problem Analysis and Required High Level Architecture

The analysis of the music data to be stored and transferred between music shops and the users plays an important role in defining the MegaStore server architecture. The MegaStore

²<http://www.cdnow.com>

³<http://www.amazon.com>

network must be designed and build in such a way that it provides high bandwidth for huge amount of data transfer in a very short response time while taking into consideration the information visibility rights and security of access. Due to the music data specification and copyrights, a main characteristic of the MegaStore application is that the real music data is protected by certain music label centers, and neither can it be centralized in one common database, nor can it be freely or randomly replicated at different sites.

To properly support the requirements of the MegaStore environment, the designed system architecture involves the following components: [BAH 00]

1. The *back-end* system, including the database engine and the predefined networking connection between the MegaStore system components.
2. The *front-end* system, including (1) the *Internet-Shop* interface, where a user from home (or work place) can search for music, listen/watch to the audio/video clips, and order CDs, and (2) the *Shop-in-a-Shop* interface, where the music storekeeper can fetch on-line the real music data from its original source in order to burn at run-time the requested music CDs.

Under the specifications of the MegaStore system enumerated above, we have identified the need to design and build a dedicated networking infrastructure for this application, where the following aspects are studied:

- The music data is geographically distributed over the network
- Information about music is classified into two main categories: the general information stored at the Directory Services that can be accessed by any user connected via the *Internet-Shop* Interface and the raw music data that can only be accessed by the music storekeepers at music centers or burning towers.
- Depending on the user profile and authorization, only a part of the information can be accessed, and users need not to know about the data distribution.
- The real music data is securely transferred through a dedicated Network connection among music centers.
- The system must benefit from intelligent caching mechanisms, which is being further investigated at the University of Amsterdam, in order to improve the performance of the system [BA 98a, BA 98b].
- High bandwidth connection is necessary to handle raw music data that needs to be passed between the real music storage centers and the burning towers.
- Low latency network connection for the *Internet-Shop* interface is necessary to support the huge number of users expected to connect to the system.

4.2.1 Database Design

The database design for MegaStore is achieved in collaboration with the experts in the music industry domain. For design of the database schema, mnemonic names are chosen, taken from the music context, and thus objects are named for what they represent. This choice helps for instance the storekeepers to easily understand the elements of the database schema and use that in formulating their requests.

Based on the study and analysis of the data to be managed (i.e. stored/transferred) within the music industry, we have identified two main categories of information:

- The general information needed for the *Internet-Shop* interface, which represents the complete information about each song, artist, album, customer, order, etc. (except for the raw music data itself).
- The raw music data (real tracks) for the *Shop-in-a-Shop* interface that represents the real data used for the on-line CD burning.

The dynamism and flexibility of the *Virtual MegaStore* system mostly depends on its database design and how open it is in supporting several application domains with different structures and different size [BAH 99a]. Different pieces of information about the *MegaStore* application domain are defined and stored as a set of inter-linked objects of different kinds, grouped by their domain of interests, e.g. artists, songs, CDs, consumers, stores, burning towers, etc. We have also taken into consideration the support for the following needs:

- ✓ To enrich the database, in order to support data of different types (text, html, images, audio, video, postscript, etc.) [BAH 99a].
- ✓ To capture the inter-relationship semantics among the objects, through the storage of a large set of relationships among different pieces of information.
- ✓ To build a distributed environment where data must be stored at different music centers, without replication or redundancy at different sites [BAH 99, BBO⁺99a].
- ✓ To extend the database in a way that the Graphical User Interface application requirements can be supported through the Web server.

The schema represented in Figure 4.1 shows the static view of the MegaStore database catalogue (also called Directory Services) in terms of classes and relationships among them. The Database catalogue defines the general music information as described within the section. The real music data however, is stored at the secure parallel/distributed database server described in section 4.3.3. The name of a class is derived from the problem domain and must be as unambiguous as possible. The attributes define the characteristics of the class and capture the information that describes and identifies the class; every attribute has a type, which specifies what kind of data it represents. The relationship association between classes is drawn as a solid line and has a name and a multiplicity range [UML 98].

The part of the database schema design for the MegaStore system that is presented in Figure 4.1 describes the detailed structure of its Directory Service. For instance, a customer can order some albums, where each album consists of a set of Songs, and it is possible that one or more artists sing every Song. A song may also have a link to its music composer, music performer and/or some instruments. Such a specification may help in satisfying users through many points of views. For instance, a user may be more interested in his/her search, in the music performer, the music composer, or the used instruments, rather than being interested in a search based on the artist name or the song title.

Following is a brief definition of the variety of information within the schema defined for the MegaStore database catalogue:

- The class *Song* represents the main entity in the MegaStore system. Within this class, the information about each song is defined. The richness of MegaStore system strongly depends on the availability of such information in the database. Since the

Internet-Shop is a dynamic Web interface, for which Web pages are created on the fly depending on the user request, the system automatically checks the database and provides the user with the most complete information that it finds in the database. The class *Song* has three links to the classes *Album*, *Artist*, and *Instrument* via the defined relationships “*Song Of Album*”, “*Sung By*”, and “*Uses*”.

- The *Album* entity represents the class for CDs, tapes, and other means of music titles collection. Mainly, an album has some characteristics, consists of a set of songs, and one or several artists sing the songs in the album. The class *Album* links to other classes such as *Artist* and *Song*, through the specified relationships “*Album Artists*” and “*Album Songs*”.
- The class *Artist* is the entity that holds all the information about each artist. Under the normal consideration some attributes such as the artist name, artist photo, and a short biography, are enough for the artist description.
- The class *Customer* keeps the necessary information about the customers. Each time a user makes an order, the system automatically checks the user’s identity based on the information available in the database, and decides whether to directly access the information about him/her from the database, if it exists, or requests it from the user, if it is not.

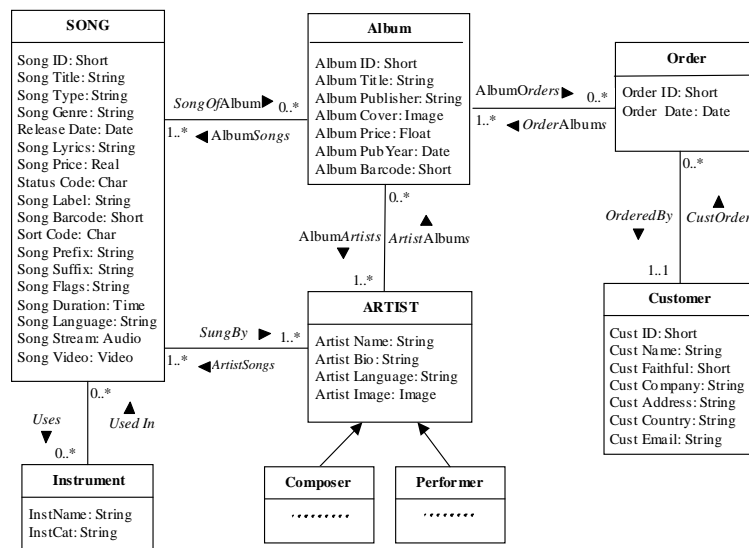


Figure 4.1: Base Schema Definition for the MegaStore System

4.2.2 ODL Schema definition

The ODL schema definition of the MegaStore Database presented below is a conversion of the music library database as presented in Table 4.1 which, corresponds to the Unified Modeling Language (UML) into an Object Definition Language (ODL). The ODL schema presents the big advantage that it can be automatically loaded into the Matisse database [Mt 01] or any other ODMG compliant database [CBB⁺00]. In addition to the standard ODL definition, Matisse ODL supports the following extensions:

- ✓ Multimedia types (e.g. audio, video, and image) are implemented, in Matisse DBMS, as a list of 8/16 bit Unsigned Short
- ✓ The *entry_point_dictionary* function serves as an entry point for the object to which the attribute belongs. Entry points are associated with the attributes by using the keyword *entry_point_of* and also serve as an efficient index.
- ✓ The cardinality is specified in between square brackets after the name of the relationship. The first digit in between the square brackets specifies the minimum number of successors, and the second digit specifies the maximum number (-1 means no limit). The default cardinality for a relationship association is [0, -1].
- ✓ The empty initialization for some attributes declares a null default value for these attributes.

<pre> interface SONG : persistent { attribute String SongTitle; entry_point_dictionary epSongTitle entry_point_of SongTitle; attribute Audio SongStream; attribute String SongLabel = ""; attribute String SongGenre; attribute String SongLyrics = ""; attribute Audio SongSample; attribute String SongType; attribute Date ReleaseDate; attribute Float SongPrice; attribute String StatusCode; attribute String SongLanguage = "En"; attribute Short SongDuration; attribute Video SongVideo; relationship List<Album> SongOfAlbum inverse Album::AlbumSongs; relationship List<ARTIST> SungBy inverse ARTIST::ArtistSongs; }; interface Album : persistent { attribute String AlbumBarCode; attribute String AlbumTitle; entry_point_dictionary epAlbumTitle entry_point_of AlbumTitle; attribute Image AlbumCover; attribute Float AlbumPrice; attribute String AlbumPublisher; attribute Date AlbumPubYear; relationship List<SONG> AlbumSongs[1,-1] inverse SONG::SongOfAlbum; relationship List<ARTIST> A_Artists[1,-1] inverse ARTIST::ArtistAlbums; relationship List<ORDER> AlbumOrder inverse ORDER::OrderAlbums; }; </pre>	<pre> interface ARTIST : persistent { attribute Image ArtistImage; attribute String ArtistName; attribute String ArtistBio = ""; entry_point_dictionary epArtistName entry_point_of ArtistName; relationship List<CDAlbum> ArtistAlbums inverse Album::AlbumArtists; relationship List<SONG> ArtistSongs inverse SONG::SungBy; }; interface ORDER : persistent { attribute Date OrderDate; attribute Float OrderPrice = 0.0; relationship List<Album> OrderAlbums[1,-1] inverse Album::AlbumOrder; relationship List<Customer> OrderedBy inverse Customer::CustomerOrders; }; interface Customer : persistent { attribute String CustomerName; attribute Short CustomerFaithful; attribute String CustomerCompany; attribute String CustomerAddress; attribute String CustomerCountry; attribute String CustomerEmail; relationship List<ORDER> CustomerOrders inverse ORDER::OrderedBy; }; interface User : persistent { attribute String UserLogin; attribute String UserPassword = ""; attribute String UserDescription = ""; attribute short UserLevel = 1; }; </pre>
---	--

Table 4.1: ODL Schema for the MegaStore Database

Furthermore, the Matisse ODL supports other object-relational extensions that are not illustrated in the example in Table 4.1. Such extension includes the definitions of indices and methods for the attributes as well as the specification of check functions and triggers for both attributes and relationships.

4.3 The MegaStore System Architecture

This section focuses on the design of the server architecture and its extension to support the *Virtual MegaStore* system. The choice of the server architecture extensions are due to the specific needs of this application for supporting its information management and the data transfer requirements.

As depicted in Figure 4.2, the MegaStore system consists of two components for data storage: the database catalogue at the Directory Services, and the parallel/distributed database server at the back-end system. All the music information including the short clips of the converted streaming audio/video that do not require high security protection will be placed at, and accessible through, the database catalogue, to be made available to the Internet users. However, the real raw music data that serves for CDs burning is securely kept at different distributed music centers linked with each other via a secure predefined network connection, so that only authorized users can access and manipulate the raw music data. The two sections below respectively describe in more details (1) the *Internet-Shop* interface to which ordinary users can connect in order to access the general music information and place their orders, and (2) the *Shop-in-a-Shop* interface to which only authorized users from the music stores can be connected.

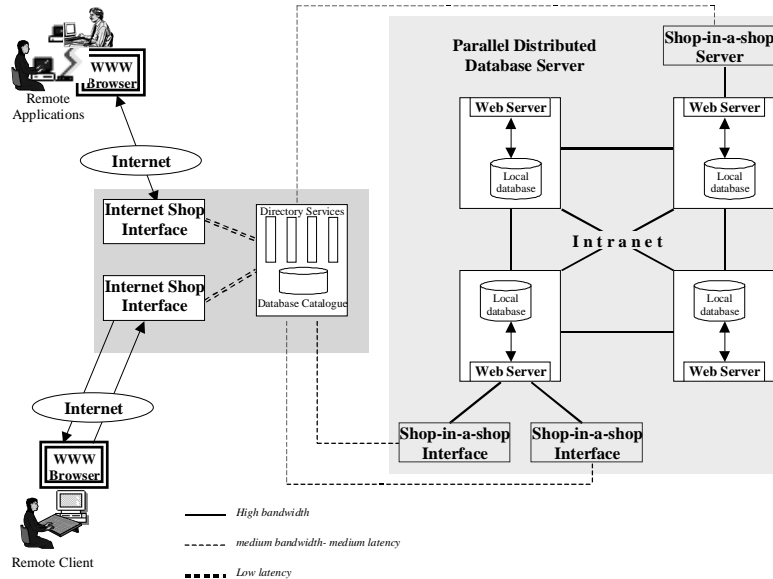


Figure 4.2: MegaStore Server Architecture Description

There will be three kinds of connections that need to be established between the components of the MegaStore system, namely:

- ① High bandwidth connection, which is required for the case of transferring a considerable amount of data. This is usually the case, for transferring raw music data between geographically distributed music centers where the studies show that a minimum of 1 Gigabyte per second Internet connection is required [BAH 00].
- ② Medium bandwidth with a medium latency connection, which is required for transfer of medium size data between the MegaStore system components. This is usually the

case, for updating the directory services when new music albums are produced (e.g. the connection from the *Shop-in-a-Shop* interface to the directory services).

- ③ Low latency connection, which is required to support the huge number of high end-users who access the MegaStore system and require the transfer of small amount of data (e.g. the connection between the Internet-shop interface and the Directory Services). The analysis, of this application domain, shows that a huge number of users must be supported [BAH 99], the data to be transferred between the Internet user and MegaStore server however, will be in the range of medium to small size.

The design of the innovative architecture and technology for the MegaStore distributed environment although applied to the music industry application, is general enough to be applied to other similar complex application environments. Such complexities are found in many other e-commerce applications. For example, for the complex applications where the research and market findings need to be combined with the commercial advertising data, as it opens the collaboration among different enterprises based on common interest in system architecture platform and database content and technology.

4.3.1 The Internet-Shop Interface

The *Internet-Shop* attempts to give the user the same feeling as when he or she is visiting a real music store. The music store, as we know it today, is a place where items can be touched, where intuition plays a bigger role than knowledge, and where music albums are usually bought in an unplanned fashion. This is how most people behave during shopping and this is also what makes the shopping itself fun. The real shop is not limited to a search engine, which finds products in a faultless but also emotionless manner. Even for the more expensive products in general, the final decision is often not objective, but is based on the emotions caused by the color, form, brand, and price. Therefore, the user interface of the *Internet Shop* needs to invoke the same kind of emotion as in a real shop.

This means that in an ideal case, a consumer from home wishes to request tracks/titles that he/she wants to be included in one CD of his/her own compilation, and if he decides to buy this tailored CD, he wishes to pay, in a secure way, by means of electronic payment. If the payment is accepted, the system must automatically allocate this customer's order to the closest music shop, in relation to the location of the customer. At that shop the titles are eventually burned and delivered to the user.

4.3.2 The Shop-in-a-Shop Interface

The *Shop-in-a-Shop* interface adds a new dimension to music shopping. Normal music shops only have a limited stock. If a customer is looking for a specific piece of music that is not in the stock, he or she cannot be served. To avoid this problem, the *Shop-in-a-Shop* system places a *customer terminal* either in the music shops or other shops, e.g. photo developing shops, or even supermarkets, that supports the browsing and selection of music that is stored in a database server. If the customer has decided to purchase a piece of music, a CD is produced by downloading the raw music data from the database server. Also, additional information like booklet, the CD label, and the CD cover is downloaded and produced. Naturally, the music store must have a very high bandwidth Internet connection to be able to retrieve the large raw CD data in a few minutes time. The advanced internet technology although perhaps lacking behind the support for vastly increasing number of customers for

e-commerce, due to the costs and required efficiency, dedicated fast connections are expected to be available in the near future to properly support the music industry application. Also, the production equipment like CD burners and high quality printers are expected to be sufficiently fast in the near future.

4.3.3 Server Architecture Extension

This section addresses the server architecture extension to support both the information management and data transfer requirements for the MegaStore system. The main requirements to take into consideration includes:

1. Design and implement the extensions needed for the existing parallel server system [PH 98], in order to support all identified *Virtual MegaStore* database functionality. These extensions support:
 - (a) The functionality needed by the HTTP daemons (Web server) front-end, in terms of support for the Web user interface, including the streaming of audio and video data.
 - (b) Easy database administration.
2. Develop and implement a mechanism that supports the entry of music and associating data into the database system [BAH 99b].

4.3.3.1 Distributed Parallel Server Extension:

To provide the MegaStore web server with efficient access to the raw music data, a parallel/distributed database framework is designed and developed [PH 98]. With this implementation, the nodes (music stores) of the distributed MegaStore server are inter-connected, making it possible for specific users to connect to any node in the distributed server and to request an object, without the need to know where that object actually resides.

Due to the music data specification and copyrights that do not allow data replication or redundancy, this data must be securely kept at the site where it belongs. The distributed database supports the following required functionalities:

- Provides a way for managing huge amount of data
- Data is securely kept at geographically distributed music centers
- Data is stored only at the point(s) where it belongs
- Data is visible from any node (music center) within the cooperation community
- Data is efficiently transferred between the nodes in short response time

4.3.3.2 Data Storage and Manipulation:

The MegaStore data manipulation concerns two components: the database catalogue at the Directory Services and the parallel-distributed database server at the back-end system. All the music information including the short clips of the converted streaming audio/video that do not require high security protection will be placed at and accessible through the Directory Services, to be made available to the Internet users. However, the real raw music data that serves for CDs burning is securely kept at different distributed music centers linked

to each other via a secure network connection, so that only authorized users can access and manipulate the raw music data.

The music data loading for MegaStore is a two-fold process. On one hand it stores the raw music data at the secure distributed server, and on the other hand it updates the Directory Services with the general information concerning newly acquired albums and titles. The storage of the music data is provided locally at each site (music centers) by the music producer framework that can be in some cases integrated with the *Shop-in-a-Shop* interface. At this level, in order to keep the system up-to-date and more consistent, not only the music data entry mechanisms are provided, but also the music data conversion and data formatting are considered [BAH 99b].

4.4 Music Audio and Video content

This section addresses some issues related to the music data conversion and briefly describes the variety of different music formats supported by the MegaStore system, as well as it provides information concerning the music encoders.

Music Audio and Video clips consist of previously captured digital audio or video files, which can also be recorded from many types of media device. Currently, the MegaStore System supports most of the existing audio and video formats including Real Audio, MPEG, CD Tracks, Waveform, QuickTime, etc.

In addition, the MegaStore system is open to support other emerging standard formats, such as the Secure Digital Music Initiative (SDMI)⁴. However, most efforts investigated on music data conversion for the MegaStore system are mainly focussed on the Real Audio [RA Inc] and the MPEG (MP3) formats, due to the various advantages of these two technologies over the others. The RealAudio has an advantage of producing both audio streaming and video clips, and the generated files are of smaller size. The MP3 however presents the advantage that it produces near CD-quality music and it is widely used over the world.

4.4.1 Bandwidth and Encoding Algorithm

Bandwidth, also known as bitrate, is the amount of data that can be sent through an Internet or network connection during a set period of time. Bandwidth is measured in kilobits per second (Kbps). Standard modems are commonly referred to by the bitrate they are able to receive, for example, 14.4, 28.8, and 56 Kbps. In addition to the standard bandwidths, music clips can be recorded for bitrates up to 100 Kbps, 200 Kbps, or more. These higher bandwidths, however, are generally more typical of corporate LANs or entertainment-based Web sites.

When audio files need to be processed or digitized, an encoder and an encoding algorithm must be selected. Most Encoders can encode using different algorithms. Each encoding algorithm is optimized for a particular type of audio and connection bandwidth. Such a dynamic selection of audio/video clips allows the system to provide the Internet-user with the best quality connection his/her system can handle, without the user having to explicitly choose from separate clips recorded for different speeds. In the MegaStore system, for the digitized music clips we mainly use RealAudio and MP3 formats. RealAudio 56K ISDN, Music – Mono and Stereo template, best suits the Internet-Shop needs since we expect Internet users via ISDN or similar connection. While, MP3 200kps - CD quality music, is

⁴<http://www.sdmi.org/>

required for the Shop-in-Shop burning system. This process is transparent to users, and the MegaStore system is configured to automatically serve the appropriate streaming file. As such, we can reach the widest possible audience, while still providing the best listening experience to users with a high bandwidth connection.

4.4.1.1 MP3 Encoders:

The MP3 audio format has become the standard for this main reason that it produces near CD-quality music and it is widely used over the world. MPEG Layer 3 files can fit up to a minute of CD-quality (44.1 KHz, 16-bit stereo) audio in a single megabyte. In comparison to an audio CD that contain a maximum of 74 minutes of music, filling a 650MB CD-R disc with MP3 files would result in more than ten hours of music.

The default bandwidth encoding in the programs is 320kbps, however a 128 kbps is recommended because it provides near CD quality music and smaller outputs. Any bandwidth higher than 320 kbps causes overkill, and will make the resulted files much larger which also implies longer download time and increases transfer costs.

MusicMatch Jukebox⁵ allows creating MP3s and WMAs from cassette or microphone with line-in recording. Ultimate Encoder⁶ is a high quality MP3 Audio encoder & decoder that supports MPEG Layer 1, 2 and 3. The XingMP3 Encoder⁷ processes files up to 8 times faster than other encoders. Faster encoding means you get to spend your time listening to your music, not waiting for it to encode.

4.4.1.2 RA Encoder:

The RealAudio Producer⁸ can encode using several different templates. Each encoding template is optimized for a particular type of audio and connection bandwidth. The RealAudio has an advantage over the MP3 in the sense that it can produce both audio streaming and video clips, and the generated files are of smaller size.

In the MegaStore system, for the digitized music clips we used RealAudio 56K ISDN, Music – Mono and Stereo. These templates best suit our needs since we expect Internet users via ISDN and similar connections. Using these algorithms to encode songs of 5 minutes duration for instance, produce digital audio clips of 1.37 MB per Song ($4.9 \text{ KB} * 300 \text{ s} = 1.37 \text{ MB}$) which, may result to up to 40 hours of music per CD.

4.4.2 Data Volume Estimation

This section gives some estimation about the data volume that needs to be handled within the *Internet-Shop* and the *Shop-in-a-Shop* interfaces.

For the *Internet-Shop* interface, the average disk space requested per CD will be around 8 MB (a CD album contains about 15 titles: $550 \text{ KB} * 15 = 8.25 \text{ MB}$). Thus, a prototype system of 5000 albums requires about 40 Gigabytes of disk space.

For the *Shop-in-a-Shop* interface we should add to the total disk space needed for the *Internet-Shop* an average of 65 MB per CD representing the complete raw data (in MP3 format), which is around 350 Gigabytes for a system dealing with 5000 albums.

⁵<http://www.musicmatch.com/jukebox/>

⁶Ultimate Encoder Inc., CopyCopyright ©1998-2001 (<http://www.usro.net/products/uencoder>).

⁷<http://www.xingtech.com/mp3/encoder/index.html>

⁸<http://www.realnetworks.com/products/producer/>

The MegaStore system is designed in a flexible way that supports different implementation strategies. For instance, if the system needs to be extended to support more albums and titles, one strategy that preserves the system performance is to store the audio/video clips together with the raw music data at the distributed/parallel database server and not at the Directory Services. This approach not only extends the system in term of supporting a considerable amount of music data, but it also preserves the performance of the system including short response time and data security, by keeping the directory services as efficient as possible. The Directory Services plays a major role in defining the general MegaStore information and specifying where the related raw data for each song or album is located within the distributed system.

4.5 Music Data Manipulation

Due to the huge number of instances to be loaded into the database and the relationships to be maintained among those objects, the data loading process into the database is an error prone task to be done manually. Some automatic mechanisms and tools have been developed in order to ease the data loading process. The developed tools expect the data to be available in specific format in order to be automatically scanned to a standard format and loaded into the database.

4.5.1 Objects Loading Strategies

For the data translation process among heterogeneous data sources, a number of approaches are described in chapter 2 (section 2.2.2.2). In this section, we describe two similar approaches, which serve the general data storage mechanism in the MegaStore projects.

- ① The first approach builds specific interface to directly store the data from its available format into the database. This approach requires the development of two-side dependant interface for each input format. If we consider N different input formats to be loaded into M databases, the number of interfaces to be developed will be $N * M$ interfaces (Figure 4.3-a). Thus, the number of interfaces to be developed increase as more input/output formats are considered. The mapping of ten different data inputs into five different databases, for instance, requires the development of fifty specific interfaces ($10 * 5$).
- ② The second approach uses an intermediate step first by storing the data into an intermediate standard format (e.g. OIF, XML), and second by loading this data into the database system. The advantage of using intermediate standard format is that, it can be loaded into any DBMS that is compliant to standards. As depicted in Figure 4.3-b, the number of interfaces to be developed for N different input formats will be reduced to $N+1$ (i.e. one-dependant side interface for each input format plus one standard interface (the standard interface is usually supported by the DBMS itself)). In this case, the mapping of ten different data inputs requires the development of ten specific interfaces regardless the number of target database systems (outputs).

The intermediate format through standard only requires the development of a one-sided-dependent interface for each input that needs to be stored into the database. It has the advantage over the first approach of being ready to communicate with any other database system or application program that are compliant to standards. Thus, this approach, based

on intermediate standard format, is certainly preferred for most cases. However, in order to cover all possible cases, we also foresee the necessity for supporting the first approach in some cases that are simple and not expected to change.

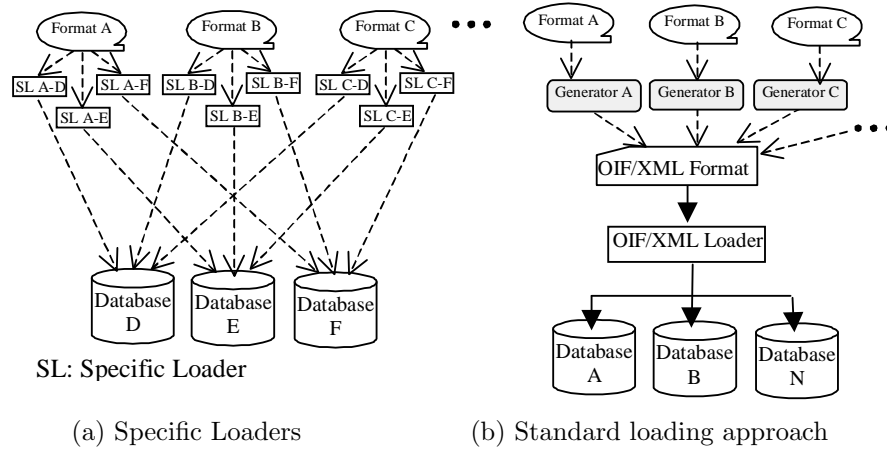


Figure 4.3: Data Storage Mechanisms

For the music data storage into the database, depending on the input data set structure, several algorithms are available and can be used to fill the database with the existing data that is available in different format.

Regarding the first approach we have developed a specific interface to directly read the data from a Dbase file “.DBF”⁹ and store it into the Matisse Database [Mt 01]. The DBF file was provided by a partner from the music domain and it contains more than 50.000 record about song titles and artist names. However, the provided data does not include any audio/video streaming or images.

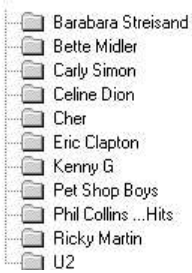
For the second approach, we have developed three algorithms for data loading. Each of those algorithms generates an OIF standard format of different levels of complexity for which a generic OIF loader is used to fill the database. The three data sets entry algorithms supported by the system are described within this section, namely:

Format A: expects a set of subdirectories, where each subdirectory is named same as artist name and contains a list of audio/video files (digitized songs) for that artist stored locally. The format of each file¹⁰ consists of the complete song name with the proper extension that reflects the audio/video type, and the file size that can be used in estimating the duration time for each song. From the description of the input data set, as presented in Figure 4.4, the developed tools allow the creation of the Artist objects and for each artist, the complete set of songs is created with the proper links between the two classes *Artist* and *Song* via the relationships *ArtistSongs* and *SungBy*.

⁹A dBASE file, a format originated by Ashton-Tate, but understood by Act!, Clipper, FoxPro, Arago, Wordtech, xBase, and similar database or database-related products.

¹⁰In order to preserve music labels and copyrights, the real name and type of the titles are not shown in the example.

Eric Clapton



(a) Artists

Name	Size
🔊 Song 1.ra	1,069KB
🔊 Song 2.ra	1,102KB
🔊 Song 3.ra	972KB
🔊 Song 4.ra	1,434KB
🔊 Song 5.ra	1,608KB
🔊 Song 6.ra	1,028KB
🔊 Song 7.ra	1,086KB
🔊 Song 8.ra	2,332KB
🔊 Song 11.ra	1,132KB
🔊 Song 12.ra	1,901KB

(b) Artist Songs

Figure 4.4: Music Input - Format A

Format B: As depicted in Figure 4.5, Format B expects a random list of audio/video files stored locally, the format of each file consists of the artist name between parentheses followed by the complete song name and its extension. The system, for instance, can extract from the “(Dire Straits) Money for Nothing.ra” entry, the name of the artist “Dire Straits”, the title of the song “Money for Nothing”, the type of the audio format “Real Audio”, and the audio streaming itself. This information will be stored in the database together with other generated set of relationships such as “Artist Songs” from artist “Dire Straits” to song “Money for Nothing”, and “Sung By” from song “Money for Nothing” to artist “Dire Straits”. If the file format does not contain information about the Artist, only objects of the class Song will be created with the necessary information and stored into the database.

English Songs

Name	Size
🔊 (Artist Name) Song Title	5,130KB
🔊 (Artist Name) Song Title	4,353KB
🔊 (Artist Name) Song Title	2,446KB
🔊 (Artist Name) Song Title	1,699KB
🔊 (Artist Name) Song Title	1,586KB
🔊 (Artist Name) Song Title	1,335KB
🔊 (Dire Straits) Money for Nothing.ra	787KB
🔊 (Artist Name) Song Title	1,106KB
🔊 (Artist Name) Song Title	1,050KB
🔊 (Artist Name) Song Title	1,029KB
🔊 (Artist Name) Song Title	940KB
🔊 (Artist Name) Song Title	884KB
🔊 (Artist Name) Song Title	854KB

Figure 4.5: Music Input - Format B

Format C: can be considered as an extension of formats A and B. In addition to the description given within format B, the Directory structure will be divided into several subdirectories and each subdirectory is named same as the Album name and contains the list of the Album titles (see Figure 4.6). Same as it happened in real, an album consists of a set of songs and one or several artists are singing each song. This format

is the most suitable, since it allows the creation of three set of class objects (Album, Artist, Song) with the necessary relationships among them. In addition to the specification supported in Format A and format B, the system will automatically extract the Album name and creates the necessary links to the classes Artist (via *AlbumArtists* and *ArtistOfAlbum*) and Song (via *AlbumSongs* and *SongOfAlbum*).

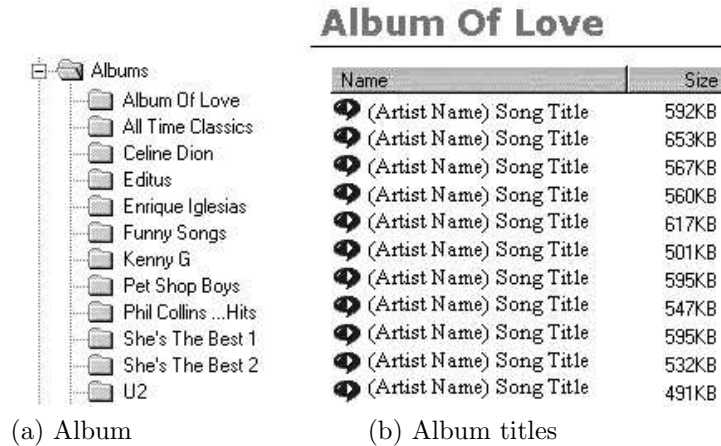


Figure 4.6: Music Input - Format C

The three algorithms are of different levels of complexity, the two first formats support the creation of the two inter-linked classes *Artist* and *Song*, while the third format augments the data structure by creating the class *Album* and adding the necessary links to it.

4.5.2 Extensions

From the three formats presented above, it is clear that some information can be easily extracted from the music data set such as the artist name, song title, streaming size, etc. However, some other information can also be automatically derived from the music data set as presented in the three formats above.

- The type of music can be extracted from the extension of the streaming file (e.g *.rm, *.ra, and *.ram extensions are Real Audio specification; *.mp3, and *.mp2 are MPEG specification, etc.). The music type helps in invoking the proper music player for the selected audio/video streaming. During a web session for instance, the proper plug-in will be automatically activated by the system based on the type of the music stream.
- In addition, since the music encoder for audio/video tracks is known or can be extracted. Then, the song duration can be calculated and provided to the database system. If we consider a Real Audio encoder using a 56K Dial-Up, Music – Stereo connection that provides a 32 kps streaming audio. We can easily estimate the song duration of *Money for Nothing.ra* to 3 min and 44 seconds, since the file size is 787 KB and the used algorithm require 3.5 kilo byte disk space per second.

An Object Interchange Format (OIF) example is presented in Table 4.2, in this example information about Songs, Artists, Albums are specified and the different links between these

entities are defined. This example depicts a part of a simple standard exchange format generated by the algorithm described in format C.

The reader of this document may notice that, within the example presented above, there are additional information such as the *ArtistImage* and the *CDCover* that are not present in any of the three formats presented above. For that we consider a set of images that are already loaded into the database, the images are supposed to be titled same as artist name and album title. During the process of music data loading into the database, the system automatically checks if there exists a corresponding image for the artist or the album, and if so a link to that image will be created and maintained for future use.

```

Song{SongTitle "Beatiful Girl", SongStream "Beatiful Girl.ra", SongDuration "2:29"}
  SungBy{Artist{ArtistName "CharlieRich", ArtistImage "CharlieRich.jpg"}}
  SongOfAlbum{Album{AlbumTitle "All Time Classics", AlbumCover "All Time Classics.jpg"}}
Song{SongTitle "How 'bout Us", SongStream "How 'bout Us.ra", SongDuration "4:30"}
  SungBy{Artist{ArtistName "Champaign", ArtistImage "Champaign.jpg"}}
  SongOfAlbum{Album{AlbumTitle "All Time Classics", AlbumCover "All Time Classics.jpg"}}
Song{SongTitle "When I see your Smile", SongStream "Smile.ra", SongDuration "4:00"}
  SungBy{Artist{ArtistName "Beat English", ArtistImage "Beat English.jpg"}}
  SongOfAlbum{Album{AlbumTitle "All Time Classics", AlbumCover "All Time Classics.jpg"}}

```

Table 4.2: An OIF Example

4.5.3 Database Administration

Some database administration facilities for the management and maintenance of the MegaStore back-end system are provided. Among other features, the database administrator interface (DBA), implemented in C++ and in Windows NT environment:

- Allows the creation of the database schema based on the data model defined for the MegaStore system,
- Eases the generation of ready for input data files via the scanning of storage disks and reformatting the data to fit the standard Object Interchange format (OIF),
- Supports the automatic loading of the music data from standard (non standard) input format and the creation of the necessary links between inter-related pieces of information, and
- Provides a mean for dynamic Web pages generation based on specific user requests.

In order to preserve the data security and confidentiality, users of the system need at first to connect to a running database server, where they must provide the system by the database name, the host of the machine running the database, and eventually the user name and password.

Figure 4.7 shows the main menu of the DBA interface for the music data objects management. The modules presented on that interface are the OIF files generator for each of the mechanisms described in section 4.5.1, and the OIF objects loader where the user do not need to do more than selecting an existing OIF file and validates it to be loaded into the database.



Figure 4.7: DBA Interface - OIF Loader

4.6 MegaStore Interfaces - Advanced Features

Some major benefits for a Web application to deploy a database management system are: the dynamism, flexibility, cataloguing, and searching facilities. In the domain of e-commerce, customers want to be able to find and view different products, make comparison between those products, and select the products that best suits their needs in an efficient manner. Another challenging feature in such applications is the fact that customers may have different preferences based on their needs, their cultures, or depending on the occasion. This section briefly describes the Internet-Shop user interface for music titles and CD album provisions, more details about this interface can be found in [BAH 99b]. The Internet-Shop interface is a Web server for music titles and CD albums, it allows a user from home (or at work) to search for music in an efficient way, listen to partial/complete music tracks, and order the music he likes.

The adopted MegaStore web server architecture, described in section 4.3, provides the Internet-Shop interface with an efficient access to the raw music data where, the distributed/parallel database framework is adapted and extended to handle the huge amount of raw music data required for burning Compact Discs [BAH 99b].

Figure 4.8 illustrates an activity diagram for the MegaStore Web interface using UML notation [UML 98]. The activity diagram for MegaStore has a starting point and several end points. A starting point (also called initial state) is shown as a solid filled circle, and an end point (also called final state) is shown as a circle surrounding a smaller solid circle. A state in a diagram is shown as a rectangle with rounded corners. Between the states are states transitions, shown as a line with an arrow from one state to another. In this diagram, for simplicity and clarity reasons only the "Show Help" state shows the possibility to end the web session or move to another entry point (state in the diagram). However, the system implementation strategy allows the possibility of ending a session or moving to another point at/from any state presented in the diagram.

This diagram helps the reader of this document to get a global overview concerning the MegaStore Web Interface implementation and shows how the different modules inside this

interface are connected to each other. Namely, level 1 corresponds to the modules that are accessible from the “Main Menu”, level 2 matches the interfaces invoked from the above level and displayed on the “First browsing Area”, and level 3 presents the modules showed on the “Second Browsing Area”.

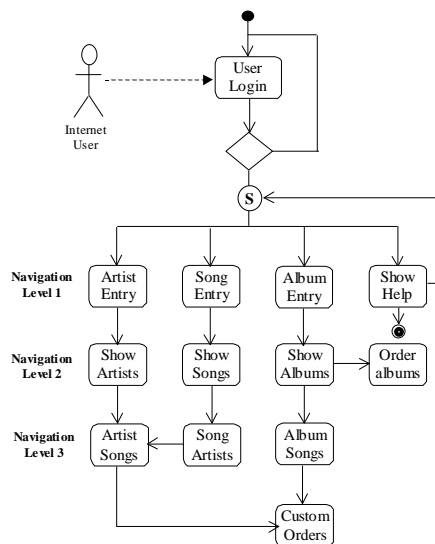


Figure 4.8: An Activity Diagram for the Internet-Shop Interface

Figure 4.9 presents a screen shot of the MegaStore Internet user Interface. As depicted on this figure the MegaStore system interface consists of three main areas of information presentation:

- ① The Menu (top left) presents the starting entry from which the user can explore the MegaStore system for music titles, CD albums, artists, and gets help when requested.
- ② The main browsing area (bottom left): at the starting of the system (the first time a user connect to the MegaStore system), this area will be the target for top music hits, best selling and advertisements for which we want to notify the user and bring his/her attention. Later on, this will be the area where results of the user requests are presented in an organized way. If the user for instance request all the available albums (from the menu by clicking the album’s button and then submit the Search Album button) the result of his request will be presented on this main browsing area. In this case, the result consists of a sequence of web pages representing information about each album.
- ③ The Second Browsing Area (to the right): At first, on this area a short description of the MegaStore system will be provided to the user. During the user exploration of the system, this will be the browsing area where goes the additional results to the user request. For instance, if the user requests a CD album from the main menu (1) by simply specifying the album name, the response to his request will be organized and presented in the main browsing area (2). From this result the user can click the album cover and gets more information about the album. Mainly, he can check the titles within the Album and order it, he can also listen to the music of each song, etc.

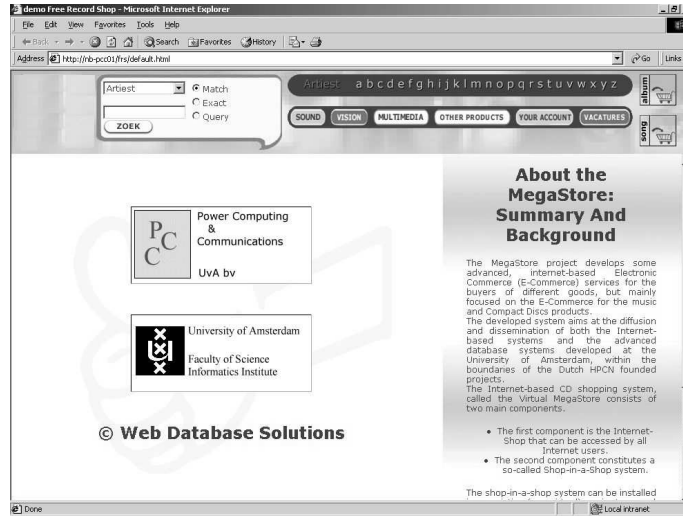


Figure 4.9: Main MegaStore Interface

The MegaStore Web interface allows users to navigate through the system and find the information of their interest in a very efficient way. This section presents a brief description of some innovative features supported by the MegaStore Internet-Shop interface, such features include: the system security, ordering mechanisms, dynamic browsing, embedded plugging, etc.

4.6.1 Dynamic Browsing

Using the MegaStore Internet interface, user queries are immediately sent to the database and the result of each query is merged on the fly with the corresponding predefined html formatting, in order to be viewed to the user who is clicking a way the MegaStore Web Interface. Thus, web pages are created on the fly based on the user request, and information availability in the database.

Figure 4.10 presents the result of a query submitted by a user requesting all the available information for the Album “She is the Best 1”. The system for instance, provides the more relevant information and makes the embedded links for audio/video clips based on the availability/non-availability of this information. Moreover, the system is built in such a way that the user will not feel data incompleteness in the interface. In this case for instance, in addition to Album name, Album cover Image, and the list titles within the album, the MegaStore system provides the following functionalities:

- The image button at the bottom of the album image, titled ‘Add Album to your Cart’, allows the user to add the current CD album to his/her standard order. In this case the user makes standard order by selecting existing albums,
- the optional speaker icon (🔊) after each song, indicates the audio stream availability for the corresponding title. This option gives the Internet user the possibility to listen to single partial/complete music tracks from the album. In addition, the speaker icon on the top-right, after ‘Album Songs’, gives the possibility to listen to all album’s songs by a single mouse click on that icon,

- The optional camera icon (📷), after each song, indicates the video clip availability for the corresponding title,
- The check-in box marks the tracks licensed for customs compilation, and
- The button at the bottom, titled 'Add to Cart', adds the checked titles to the user customized order. In this case the user can make his own compilation of titles to be included within an Album (see section 4.6.2). Please notice that the user can always add new titles to his custom compilation from different screen sessions. A tailored order will not be taken into account until being validated by the user.



Figure 4.10: Album Songs Interface

4.6.2 Ordering System

The MegaStore Internet Shop do not only allows users to search for artists, titles, albums, and listen to audio/video clips, but it also allows a user from home or at work to make his own orders and place them into the system.

Figure 4.11 presents a state diagram for orders using the UML notation [UML 98].

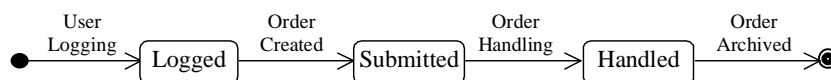


Figure 4.11: State Diagram for Orders

The MegaStore ordering system has the capability to produce both standard Album orders and custom compilation orders.

- **Standard Orders:** A user can create an order based on existing albums. Each time he/she adds a new item to his/her CD shopping cart, a general overview about the current order status will be shown including: the list of ordered items, the price of

each item, and the total charge for the order including the handling and shipment costs. The user will always have the possibility to add/remove items to/from the order, and the order will not be taken into consideration unless it is validated (checked out). The order validation process includes gathering the ordered items, the customer information, and the payment procedure.

- **Customized Orders:** As depicted on Figure 4.12, the user customized Albums feature allows users to select the set of titles to be included in the album. While the user navigates through the Internet-Shop interface he/she can select the songs to be added to his/her tailored album compilation. The number of titles to be included in the compilation mainly depends on the total space/time available for the songs support (CD, tape, etc.). The total price of the customized album is gradually calculated based on the price of each song and other handling, shipment, and compilation costs.



Titles in your Custom CD: 1		
Song Description	Price	Remove
Memory	€0.7	Remove
Where Does My heart Beat Now	€0.6	Remove
La Vie En Rose	€0.8	Remove
Don't Worry	€0.7	Remove
Miss You Nights	€0.7	Remove
I Don't Want To Lose You	€1	Remove
Against All Odds	€.6	Remove
Dance Into the Light	€.9	Remove
True Colors	€1.1	Remove
CD+S+H		€ 6.50
Totaal		€ 13.60



Figure 4.12: Custom Order

In order to preserve music labels and artist rights, only music tracks specifically licensed by music labels and artists for custom compilation can be included in tailored orders. Currently, most major music labels only provide their music for full Album sales.

4.6.3 System Security

The MegaStore System is a multi-level security system. Several kind of users are expected to use the system including (three roles defined for accessing the MegaStore system):

- The ordinary user (access level 1),
- The storekeeper (access level 2), and
- The system administrator (access level 3).

Within each session a user must identify himself (herself) to the system using a user name and a password, the system will automatically checks the user authentication against the information available in the database and allocate an accessibility level to the user. This level will be valid within the same session and the user can only see the parts of the information

(system) that corresponds to his/her visibility/access level. In addition, authorized users can always re-identify themselves with a different access level for the same session.

4.6.4 Current Implementation Status

For the implementation purpose of the MegaStore Internet-Shop interface, we combined different technologies to support the application requirements as described in section 4.2. Based on the detailed study of the MegaStore application functionality needs, the appropriate approach to apply and the technologies to use are identified [BAH 99]:

- The designed database is being implemented on top of the Arches computers at the University of Amsterdam. The Arches system is currently composed of 20 nodes containing each a dual Pentium II with 512 MB ram and 9 GB disk, and it supports several network communications.
- The Matisse database system [Mt 01] is used as the object-oriented distributed database system (ODBMS). Among other features Matisse supports transaction management, concurrency control, historical versioning, indexing mechanisms, high speed for data access, multi-media streaming, and standard programming interfaces using C/C++, Java, ODBC, etc.
- Different Internet infrastructures are deployed depending on each functionality of the MegaStore system (a high bandwidth communication between the music stores for huge amount of data transfer and a low latency communication to the Directory Services for high end-users access).

Some experiments are performed using an NT front-end machine to run a DBA interface and an Internet-Shop server, that is in turn connected to the Matisse database running on the Arches machines, using the ODBC driver. The database administrator (DBA) interface, implemented in C++ and Windows NT environment, provides some administration facilities including the automatic loading of the music data and the creation of the necessary links between inter-related pieces of information. And The Internet-Shop server is implemented using a combination of the most recent and relevant software technologies including JAVA Script and Visual Basic Script for tips programming, Active Database Objects (ADO) for database connection, and HTML for text formatting. The implementation of the server is made possible, using the Active Server Pages programming environment that allows the combination of all these different software technologies in one single environment.

4.7 Derived Applications

This section describes the implementation prototypes of two innovative interfaces related to the MegaStore system, namely, the *LuisterPaal*¹¹ interface and the *Music Sheet*¹² application. The LuisterPaal interface provides a simplified version of the MegaStore system, while the Music Sheet application provides an extended implementation of the MegaStore general concept. Figure 4.13 presents a conceptual model for e-MegaStore applications. This model shows the applicability of the comprehensive MegaStore framework to the design and implementation of similar systems in the area of manipulating multimedia large data sets. Thus, the two applications proof the general implementation approach for the MegaStore system, from which other applications that share the same characteristics can also benefit.

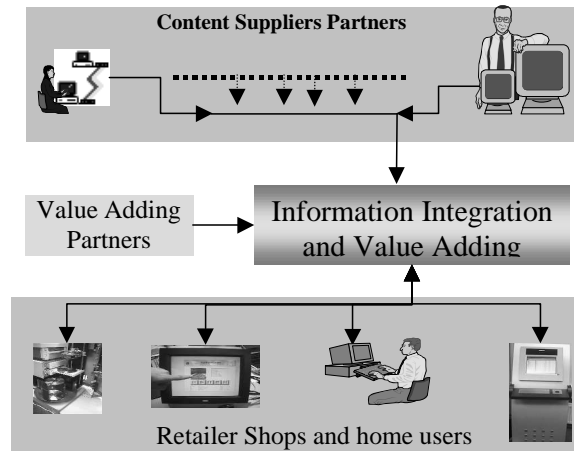


Figure 4.13: Conceptual Model for an e-MegaStore Application

4.7.1 LuisterPaal Interface

The LuisterPaal interface [BAR⁺01] is a listening booth facility, which enable users visiting a music store to listen to music. In order to facilitate interactions with the system, the self-service LuisterPaal interface uses a scanner (barcode reader) and a touchscreen¹³. The barcode reader facilitates the searching mechanism for the user, while the touchscreen is a finger selection technology that makes the interaction with the system more reliable. Music selection is simplified to the scanning of the barcode of a given album, therefore, the entry to the database is achieved via the unique barcodes of the albums.

From the implementation point of view the LuisterPaal does not differ much from the MegaStore concept. Its user-friendly interface is based on the use of (1) a database catalog holding the Album's information (e.g. titles, duration, artists) and (2) a multimedia database server holding the real audio/video clips for each album.

¹¹The LuisterPaal interface is a joined project between the University of Amsterdam, Power Computing and Communication PCC - UvA, Free RecordShop NL, and Siemens Nixdorf.

¹²The Music Sheet application is a joined project between the University of Amsterdam, Power Computing and Communication PCC - UvA, MondriCom b.v, and Attitude NL.

¹³Touchscreens are specialized hardware products aimed at assisting interaction with computer learning devices. Touchmonitors emulate a mouse, giving the user "Free-Rein" in selecting appropriate items on the screen, by touching the requested features.

The LuisterPaal system is simple to use and very advance regarding the functionalities it provides. It consists of a very friendly user interface, where a user can simply scan the barcode of a CD and follow a real interaction with the system. The barcode scanning is realized via a fixed barcode reader (scanner) attached to the touchscreen. The barcode reader and touchscreen are mandatory for simplicity reason; the system also runs on a PC with a normal keyboard and mouse.

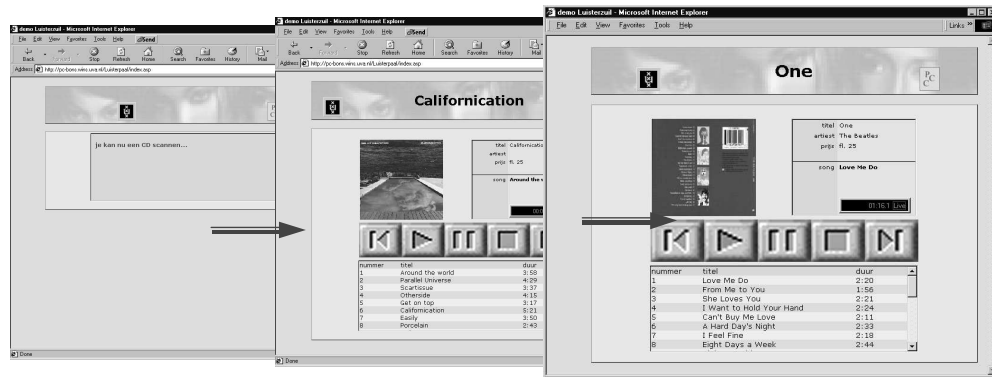


Figure 4.14: LuisterPaal User Interface

Within the LuisterPaal system, the user in a music store can grab a case of a CD and scans its barcode to the system. At the music stores, instead of using the case of a CD to scan the barcode, the availability of simple catalogs for Albums including barcode's specification provides a more convenient solution. When a user enters the barcode of an Album, the system connects to the database catalog, extracts the necessary information to be browsed to the user on the screen, and launches the audio/video streams to be played on-line. At any time during the listening process, the user can introduce the barcode of a new Album or freely interact within the audio/video clips of the album. Figure 4.14 shows three screens of the LuisterPaal interface. As depicted in the figures, the user can do the most audio/video operating functions (e.g. play, stop, pause, play next, play previous, play first, and play last). The LuisterPaal interface provides the following benefits and advantages:

- * Very simple interface in which, a user has only to scan the barcode of an album, the rest is automatically done by the system,
- * Flexible interface based on a large collection of music data stored in a database catalog,
- * Support for video clips to be played on the screen,
- * Audio/video data streaming are fetched on-line from the multimedia database server,
- * Interactive system in which, a user can freely navigate across albums and titles, using specialized hardware devices (barcode reader and touchscreen),
- * Unlike existing kiosks, in which a listing point is requested for each album, a single LuisterPaal interface is linked to the totality of albums within the database catalog.
- * Most users in music stores are not aware on how to make complex queries based on keywords. Thus, The use of a barcode reader and a touchscreen facilitate the user interaction with the system,

- ✱ The LuisterPaal approaches reduce the costs and efforts in making such facilities for ordinary users. The estimated costs and efforts are limited to the development of the database catalog and the set-up of the interface.
- ✱ The use of database standards and middleware solutions during the implementation of such a system eases its extension and makes it reusable.

4.7.2 Music Sheet Application

Music Sheet is a new emerging application in the domain of music notes and their performing issues. It consists of the design and development of a complete system, which model the complete life cycle of music production in one general concept. The development phases of the global concept rises from the lyrics writing phase, to the performing phase, to copyrights and publishing phase. At the same time it addresses the degree of difficulty, the used instrumentation, and Intellectual Propriety Rights (IPR) related to music production.

The Music Sheet server [BAR⁺01], briefly described in this section, provides a catalog that lets retailers form different music bookstores to search for music books, retrieve music notes, and place their orders into the system.

Among the highlighted development characteristics of the Music Sheet application are the extended data model and the flexible search interface. The complete data model allows the proper structuring of the application and links together related real world entities, while the flexible search interface provides a mean to navigate through a complex set of data inter-linked to each other via many relationship associations.

Figure 4.15 illustrates the data model for the Music Sheet application and shows the added entities to the data definition concepts, among which music notes, different instrumentation types, and instrumentation elements are supported. Within this model, music publications are well defined and classified into several categories of music books and folders. Each of these publications is composed of a set of titles; a title in turn is composed of several parts.

The richness of the Music Sheet data model has led to the development of a very flexible interface for the sheet music application. The search mechanism of the interface uses and combines several entities as entries to the database system. Search for publication titles and music notes is based on a combination of author name, title, instrument type, music genre, keyword, etc. A user for instance, can search for a piece of music notes of genre 'classic', composed by 'Bach', and played with both 'organ' and 'string' instruments.

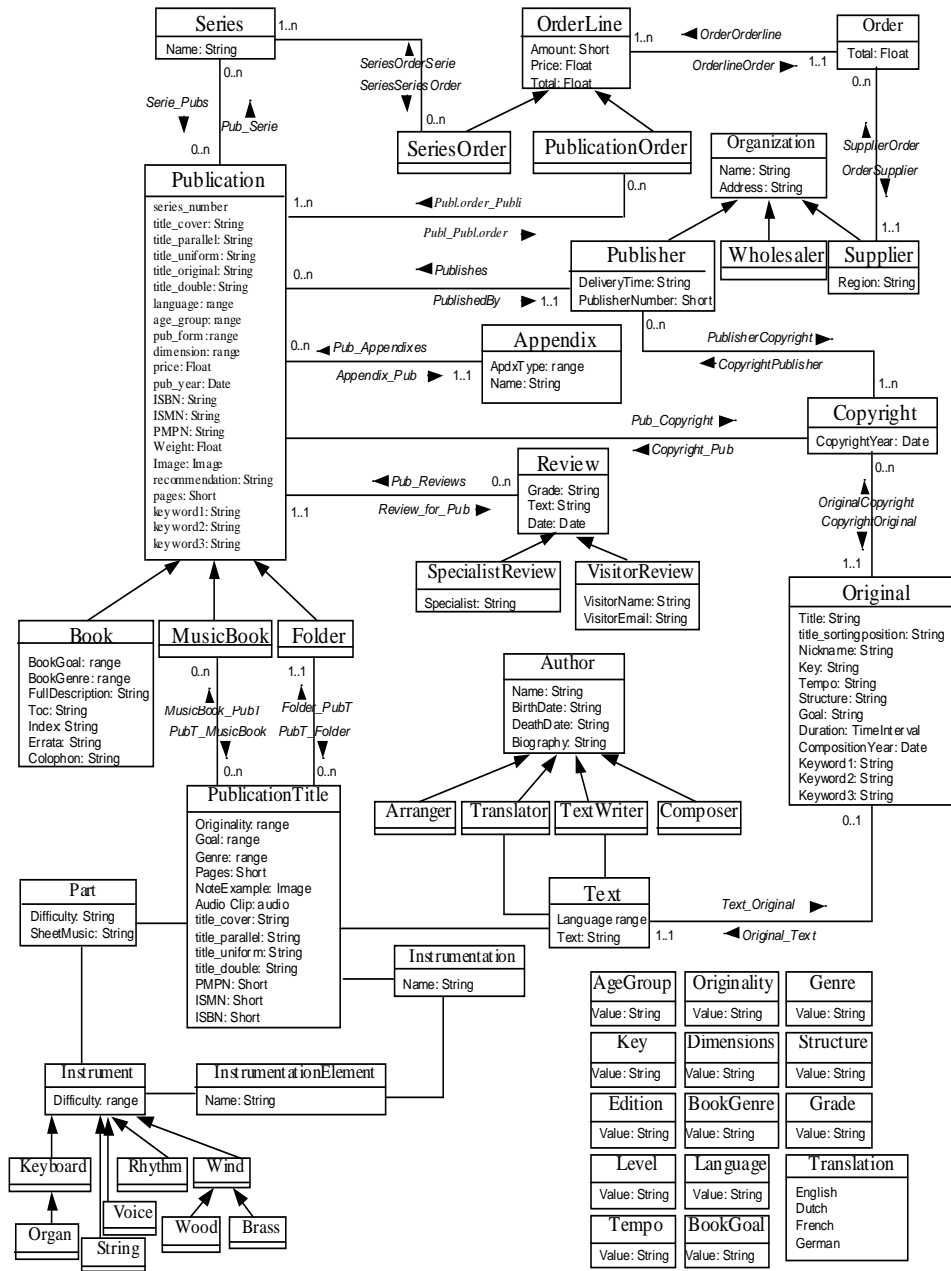


Figure 4.15: Database Model for the Music Sheet Application

Figure 4.16 shows four screens from the Music Sheet interface, those interfaces illustrate few samples among many other interfaces¹⁴ dedicated to explore and navigate through the complete structure of the database model. Following are brief clarifications concerning the four samples presented in Figure 4.16:

1. *Subscription* interface is mandatory when a user decides to order some music books or music titles
2. *Music search* interface allows a very flexible entry to the system, in which a user can specify and combine different keywords related to his/her search criterions (e.g. title, composer/author, number, genre, and instrument).
3. *Composer search* interface explores in more detail information about authors. This interface is accessed via the main list of authors or via direct links from publications and publication titles.
4. *Music notes* example provides a sample of the music notes available for the publication titles. Usually, those samples are free to download and to print.

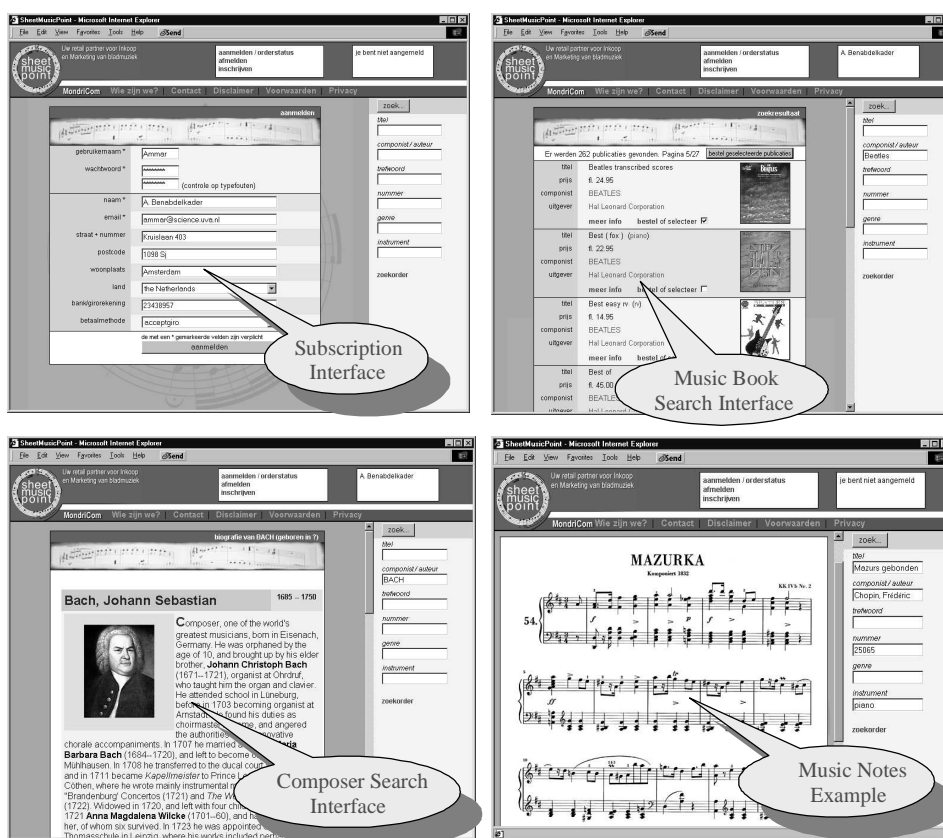


Figure 4.16: Music Sheet User Interface

¹⁴ Among them, we enumerate interfaces for publisher, suppliers, reviews, copyrights, detailed titles descriptions, and interfaces for music books ordering.

Advantages gained within the development of the sheet music application:

- * The Music Sheet application benefits from the MegaStore concept and adopts its full ordering system.
- * Publications and publication titles are well defined and more descriptive elements are introduced to their definition. Among these elements, we enumerate music note samples, degree of difficulty, reviewers' recommendations, and support for several types of keyword.
- * A clear distinction is made between the different authors participating in the production of a given piece of music. The authors range from text writer or translator, to the music arranger, to the music composer.
- * The system is open for specialists, reviewers, and system visitors to give their comments and recommendation for the different part of the music.
- * The designed database model is comprehensive enough to express the intellectual propriety rights (IPR) such as text originality, copyrights, and publishing organizations.
- * The Music Sheet interface provides a very flexible entry to the database based on the combination of keywords from different entities.
- * The interface is more dedicated to music bookstore retailers, but it is also simplified and can be used by ordinary users.
- * The system is extended with a component, which give users the possibility to describe in more details a desired item that it could not be found using the search interface. User requests will be treated by a specialist in the field and results (if any) are sent back to requesting user.

4.8 Conclusion and Discussion

This chapter addressed the innovative design methodology of an open architecture for the MegaStore application. The chapter first described in details the application requirement analysis and the database design, and then it addressed the general design and the server architecture for the Internet-Shop and the Shop-in-a-Shop interfaces. Further discussions are focussed on the issues of music conversion mechanisms, extensions to the distributed multimedia server, and the adaptation of the MegaStore framework to other e-commerce applications within the music industry. A distributed/parallel multimedia database server is adapted and extended to handle the huge amount of raw music data required for burning Compact Discs.

The development of MegaStore system and its extension with further sub-projects, such as the *LuisterPaal interface*, and the *Music Sheet server*; have proven the strength and generality of both the database design and the system architecture developed for these applications. The system analysis and the database design for the applications, developed for MegaStore, are achieved in collaboration with the experts from the music industry. Thus, the database schema description and names chosen for the schema components were taken directly from the music context, and also the object naming strategy in the e-MegaStore framework is mnemonic.

4.8.1 Major Characteristics and Benefits provided to MegaStore Application

As depicted in Figure 4.17, within the MegaStore framework, that is adjusted to support more e-commerce application, we have designed and developed an e-MegaStore architecture that seamlessly fits the current Music Store models of operations. In addition, in supporting their main goals, the e-MegaStore framework offers the following additional advantages:

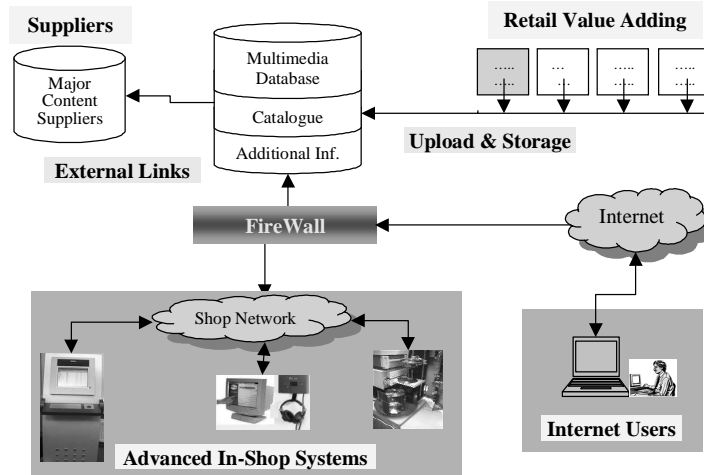


Figure 4.17: e-MegaStore System Architecture

- Value added information can be gathered from different major suppliers/retailers and gradually added to the database catalog. The database catalog can make references to multimedia data stored either at the multimedia server or detained by external suppliers. At the same time, it is possible that external catalogs link their information systems to the audio/video data stored within the multimedia database server, developed for the e-MegaStore applications.
- The use of Matisse object-oriented database system [Mt 01] made it possible to fulfill several requirements for MegaStore database development and its extensions, in specific to support multimedia data types, large objects handling, indexing, versioning, and high performance. The use of Matisse object-oriented database system together with a parallel/distributed database server for the development of the e-MegaStore applications provide the following advantages:
 - Allows flexible navigation through complex Web objects.
 - Supports scalability as necessary for multimedia large objects.
 - Provides high performance as required by multi-users applications.
 - Supports manipulation of new data types (e.g. for author biography, song lyrics, images/photos data, and audio/video streams).
- The security and data encryption issues for audio/video data streams are fully addressed and considered within different applications. The parallel/distributed database server for MegaStore provides three levels for security and encryption:

- Short audio/video clips (data) are securely stored but not encrypted.
- Medium audio/video tracks are encrypted and securely stored.
- High quality audi/video streams are highly encrypted and securely stored.

4.8.2 Contribution of the MegaStore's Information Management Approach to **GFI₂S**

The main idea behind the design and set-up of the necessary database structure and system architecture for the MegaStore application was to develop a comprehensive system that supports applications with two specific characteristics: (1) to facilitate the storage and manipulation of large data sets and (2) to provide a flexible information classification and clear separation between public and proprietary data. The experience gained from the approach followed in MegaStore Contributes to the following three features in the design and development of the **GFI₂S**:

- *System Reusability*, through the deployment of database standards and Internet middleware, for data definition and information access. Thus, making optimum use of the developed components, unifying the access to data, and reducing the development efforts.
- *System Efficiency*, through the development of parallel/distributed database servers and through the deployment of good strategies for the storage and management of large and complex data. These strategies have proven their strength and flexibility through different applications within current MegaStore-like models of operation.
- *User Assistance*, through the development of user friendly interfaces. In MegaStore, these interfaces are addressing the needs of ordinary end-users in E-commerce applications. While, in **GFI₂S**, they are planned to also assist advanced users in creating federated schemas, specifying the access rights on the shared information, defining the schema derivation mappings, and so on.

Chapter 5

Information Management for Scientific Applications

5.1 Introduction

This chapter identifies the basic information management requirements of emerging applications in e-science and presents the main approaches taken within the Virtual Laboratory project addressing these requirements. The Virtual Laboratory project (VL¹) initiated at the University of Amsterdam and supported by the Dutch ICES/KIS-II program, aims at the design and development of a generic environment and a flexible architecture supporting scientific applications and scientists with their experiment definition and control, data handling facilities, and access to distributed resources. The cooperative federated information management framework developed by the CO-IM group for the VL, aims at providing the necessary information services to enable scientists and engineers to work on their experimentations, and to properly handle all related data/information [AKB⁺01, ABK⁺00]. However, the work presented in this chapter describes only the contribution of this author to the partial design and development of the VL information management functionalities.

This chapter first briefly describes the architecture design of the Virtual Laboratory and then focuses on specific advanced features, functionalities, and facilities introduced and developed for management of information in scientific applications. The addressed features include:

- ☞ The *strategies* for storage and retrieval of multimedia scientific information (addressed in section 5.3.1),
- ☞ The *use of standards* for scientific data modeling and archiving, supporting the integration of data from heterogeneous sources (addressed in section 5.3.2),
- ☞ *Universal and schema free access* to scientific data, stored within various local and remote database management systems (addressed in section 5.4),
- ☞ *Access security* to the data available within the VL archive is based on *predefined visibility restricted schemas*. As such, appropriate access rights and visibility levels

¹Virtual Laboratory (VL) is also referred to as VLAM-G that stands for Grid-based Virtual Laboratory AMsterdam.

for individual users and groups are presented. This approach, addressed in section 5.5, is mainly adapted for *VL Scientific Results Publishing*,

- ☞ At the end of this chapter *performance issues* of the suggested implementation approach is addressed. *Benchmarking tests*, for the storage/retrieval of massive amount of data is performed. The presented results address the specific features that assure database efficiency and performance, including the information access security and the short response time for data transfer (addressed in section 5.6).

If we define a site as an organization with several application systems and databases, *this chapter provides generic tools and advanced facilities that can be adopted by every site to enable it as a node in a cooperation network*. On one hand, sites can benefit from these generic tools and advanced facilities in order to make their applications and databases stronger and more efficient. On the other hand, the use of standards and the consideration of state-of-the-art techniques, when developing these tools, enable these sites for appropriate collaborations taking advantages of (1) interoperability for data access and communication, (2) information sharing/exchange for interfacing and federation, and (3) collaboration within Virtual Laboratories and Virtual Organisations.

Later on in Chapter 6, the design and development of a more generic and flexible collaborative information management framework is described, addressing the cooperation among different centers and scientists, required within the e-science and other emerging collaborative applications.

5.2 Virtual Laboratory Architecture Design

A main goal of VL is to provide a science portal² for distributed data analysis in applied scientific research. The VL supports scientists with all the steps involved in conducting their experiments, using the Virtual Laboratory facilities. The steps may involve experiment definition and control, access to local and remote sites, process and analyze the retrieved data, archive and publish the resulted information, control of external devices from their experiments, use advanced tools to simulate and visualize the results, collaborate with other scientists and centers, and so on. Thus, Virtual Laboratory (VL) provides *a user-oriented environment and a science portal, supporting collaborative, Grid-based distributed analysis in applied sciences, using cross-institutional integration of heterogeneous information and resources* [ABB⁺01].

The general design of the VL architecture is based on multiple functionality layers, so that the required application-specific and domain-specific computational and engineering features can be separated and dealt with differently from the generic computing and data management aspects. The generic aspects serve a broad range of scientific domains, and include features related to parallel/distributed computing and networking infrastructure, basic middleware tools for information management and collaboration, and generic environment for simulation and visualization techniques. Further, domain-specific or application-specific features, are defined on top of these generic features.

²A science portal refers to delivering science information and services to industry, investors, and to the research community; using cross-institutional integration of heterogeneous information and resources.

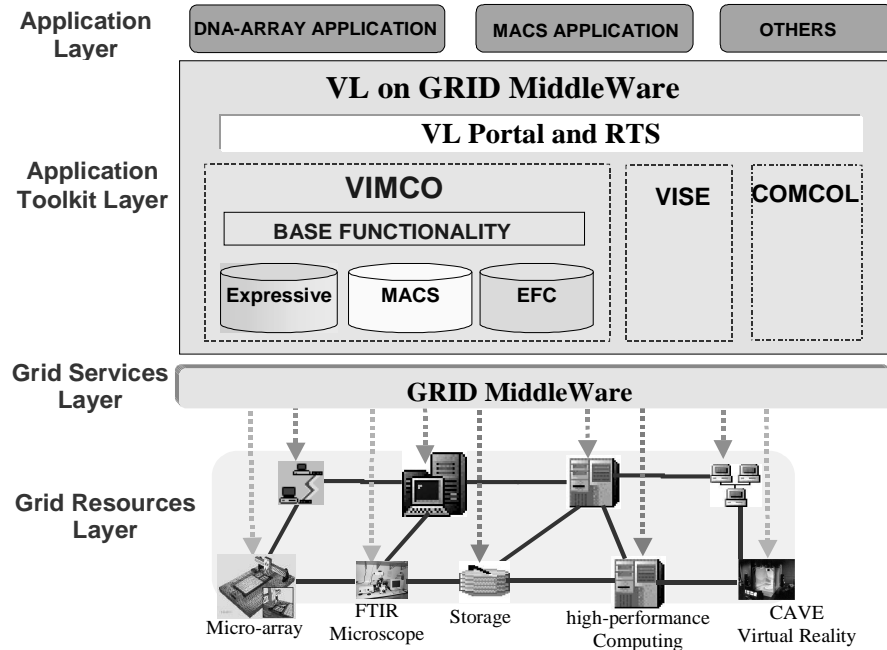


Figure 5.1: Functional layers within the Virtual Laboratory Environment

As illustrated in Figure 5.1, the VL reference architecture is primarily composed of four functional layers [ABK⁺00, AKB⁺01, ABB⁺01]. This multi-layered design primarily represents the functionalities supported by the VL, and does not imply that the development of the layers at the “upper” levels is dependent on the “lower” layers. Rather, it represents the fact that on one hand every layer can be developed simultaneously and independently of the others without the need for extensive interaction during the design. On the other hand, it allows to focus on one layer at a time and it provides possibilities for a clear description of the primitive Virtual Laboratory operations and components, and their individual functionalities at different levels. The four-layer architecture of VL is briefly described below.

Layer 1: Grid Resources Layer

The Grid Resources Layer provides the high-bandwidth low-latency communication platform, which is necessary both for accessing the underlying large data sets and for the physical or logical distribution of the connected external devices and the client community that uses the laboratory facilities.

Layer 2: Grid Services Layer - Grid MiddleWare

The gigabit networking technology being set at the University of Amsterdam, and the Globus distributed resource management system [FK 98], are used for the development of VL Grid middleware environment. The Globus system addresses the needs of the high performance applications that require the ability to exploit diverse, geographically distributed resources. The VL Grid MiddleWare provides basic mechanisms for the communication, authentication, network information, and data access. These mechanisms are used in VL to construct various higher level services, such as parallel programming tools and schedulers for multidisciplinary scientific applications. Since the Globus system offers the resource management required for distributed computing, it is used for the development of some

VL internal components (e.g. remote data access, resource allocation, and secure access to external devices).

Layer 3: Application Toolkit Layer - VL on Grid MiddleWare

Two main roles are targeted by the development of the VL middleware layer. First, it provides a set of generic functionalities for the various applications from advanced scientific domains such as physics, chemistry, system engineering, medicine, and biology. Second, it bridges the gap between those advanced applications and the Grid-services layer. The major components of VL middleware support unique information provision and provide means for resource integration and collaboration. These components include:

- ① The VL *Portal and RTS*, provides a Grid-based science portal, supporting the VL functionalities for a wide range of users, while hiding the details related to the distributed data computation from the end-users [BHG⁺01]. The VL portal can be easily used to define a new VL application, while the RTS (Run Time System) forms the VL core component constituting the interface between the VL Grid MiddleWare and the underlying Grid services (Grid MiddleWare).
- ② The VL *Information Management for Cooperation* (VIMCO Module) provides archiving services as well as the information handling and data manipulation within the Virtual Laboratory. This module supports a wide range of functionalities ranging from the basic storage and retrieval of information (e.g. for both the raw data and processed results), to advanced requirements for intelligent information integration and federated database facilities, and to supporting the collaboration and information sharing among remote centers.
- ③ The *Communication and Collaboration* (ComCol Module) enables the communication with external devices connected to the laboratory, as well as the secure communication and collaboration between users within and outside the laboratory.
- ④ The *Virtual Simulation and Exploration* (ViSE Module) presents a generic environment in which scientific visualization, interactive calculation, geometric probing and context-sensitive simulations are supported.

Layer 4: Application Layer

At the top layer of the architecture is the application dependent part of the Virtual Laboratory framework. Within this layer, interfaces are present, and application-specific and domain-specific tools are provided in order to enable users to make their specific experiments, using the functionality provided by the other layers in the architecture. Among the domain specific application cases that are currently in development phase within the VL project, we enumerate: MACS- material analysis for complex surfaces [FAE⁺01, EAG⁺01], EXPRESSIVE- genome expression in biology [KAB⁺01], dynamic exploration and distributed simulation within interactive environments [BS 00, SKH⁺99], and EFC – Electronic Fee Collection and intelligent transport systems [VWH 00, DHA⁺98, HDB⁺97].

Due to the focus of this chapter of the thesis on the information management within VL, the following sections only describes the contribution of this author to the partial design and development of VIMCO module.

5.2.1 The VL Information Management for COoperation - VIMCO Module

The VIMCO module is being designed as a multi-level information management system and environment to support the classification and manipulation of the data within the Virtual Laboratory environment. Considering the wide variety and large amount of data handled within different layers of the VL, the required information management mechanisms may vary. Namely, the need for parallel database extensions, distributed database facilities, and federated/integrated information management; must be considered. These extensions are necessary to better support the information management requirements of advanced scientific applications. Therefore, the design of the information management system must support structured as well as binary data access, data integration from several sources, location transparency for remote data access, secure and authorized access to shared data among networked applications, and the intelligent data handling.

The general design objectives of the VIMCO system within the VL cover the areas of fundamental database research and development to support complex domains. The VIMCO development primarily addresses two main focus areas:

- ① The first area focuses on the *Data Archive*: storage and retrieval of a wide variety of scientific and engineering data necessary to be handled within the VL, supporting their categorization, storage, and scientific data publishing (that is the focus of this chapter).
- ② The second area concentrates on the development of a *Generic and Flexible Information Integration System (GFI₂S)*: a flexible collaborative framework preserving systems autonomy and supporting the import/export of data based on information visibility and access rights defined among systems (that is the focus of Chapter 6). *GFI₂S* is designed as a generic approach that serve the information integration in a highly dynamic network of applications. Therefore, its deployment within the Virtual Laboratory environment can improve the accessibility to large databases for data intensive applications and can provide access to a variety of distributed sources of information.

The work on data archiving focuses on the design and development of an information brokerage system to archive the wide variety of data with different modalities and from different sources. This includes all the data generated through specific research and application domains supported by the VL framework. For instance, for the information handled by *ViSE* and *ComCol* modules of the VL as well as other VL applications, a catalogue/archive schema has been developed using the Dublin Core MetaData standard. This catalogue/archive schema has been refined to achieve a more scalable and extendable archive meta-metadata, able to capture comprehensive information about the complete experimentation process (e.g. raw/processed data, experiment parameters, scientist information, hardware devices, and software characteristics). The designed schema is extendable to cope with the future modifications and with the flexible addition of new experiment types.

5.3 Multi-Media Scientific Data Sets Manipulation

One common characteristic of the scientific domains such as biotechnology, physics, astronomy, and complex engineering applications, is that they all produce large data sets. The

data generated from different experiments in each of these domains needs to be inter-linked and referenced, so that the scientific applications can fully utilize the outcomes of the experiments.

The physical data storage approach plays an important role in the long-term strategy for data management in organizations. The data storage approach chosen for an application environment mostly depends on the requirements of the application, specifically for data archiving and information access. Simple applications, for instance, are built on top of the file system. Other applications from system engineering domain, however, require the deployment of proper physical database design for storing their data. Nowadays, more complex applications such as in bio-informatics [Gelb 98], biology [BDH⁺95], and medicine [BAS⁺99] require much more advanced solutions. In such solutions, the design of a proper system architecture and physical database approach can help in solving problems related to information security and efficiency of access, as well as facilitating the cooperative working processes among different experimenters and sites. The approach must also take benefits from using database systems in terms of user view definition, information sharing, and system integration.

Considering the complexity and the distribution of data in advanced scientific applications, the proper management of the domain information and knowledge is challenging. The scientific data storage/archiving and data access/retrieval mechanisms must be addressed in such a way that data sets can be properly searched, retrieved, compared to other existing data sets, published, and inter-linked. In addition, the necessary mechanisms for information security, performance issues, and the means to distinguish and protect the private data, together with the necessary support for the data to be published and shared with remote users must be provided.

Therefore, the objectives of a system for scientific data management must go beyond just providing a networked “hierarchical storage management” system [HSM 01], which only enhance the notion of a traditional file system made up of a hierarchy of directories and files. Traditional mechanisms for storing large volumes of scientific data are inadequate to satisfy the long-term cataloging, access and retrieval needs of scientific experimentations and their meta-data.

In scientific applications, meta-data refers to the annotations and added “information” to the scientific data sets, which is for instance different than the meaning of meta-data, referring to schemas in databases area. Scientific meta-data is an essential component of a data archive; its storage and maintenance help users to understand the structure of the archive and provide necessary information to correctly interpret the data [JCF 95]. Scientific applications commonly include a metadata database utilizing database management systems to provide users with a powerful query facility. The metadata database is often managed separately from the archive as a facility for locating data [JCF 95, GSB 95]. In many cases, recent research combines DBMS technology with the so-called hierarchical storage management systems (HSM). The DBMS is re-engineered to use an HSM as its storage medium resulting in metadata databases with virtually huge data sets located at various storage facilities [SS 95, BFL⁺95].

The Scientific Data Management approach in [HA 96, AHW⁺98], built on top of the traditional file system, addresses this challenge by providing a hierarchical storage management system to store data files, as well as a database that captures information about those files. Thus, it provides researchers with enhanced facilities for storing, locating, retrieving, and interpreting archived data. Similarly, the Intelligent Archive (IA³) provides scientists

³Intelligent Archive, Lawrence Livermore National Laboratory – LLNL (<http://www.llnl.gov/ia/>).

with advanced capabilities for organizing and searching the information. One drawback to these approaches is that keeping the metadata database synchronized with the archive can be difficult.

5.3.1 Storage of Large Scientific and Engineering Data Sets

So far, depending on the requirements and criteria, a variety of approaches are applied to the storage of such data. We categorize the approaches discussed in this section, addressing the management of large-scientific multi-media data sets, into:

1. The traditional *file system* approach, described in section 5.3.1.1.
2. The *external data link* approach, defined for access to external data sources, described in section 5.3.1.2.
3. The *one-database storage* approach, described in section 5.3.1.3.

The *file system* approach, traditionally used in the past, has shown its inefficiency especially in maintaining the links between the inter-related information pieces, searching the available information, and comparing results of different experiments. For instance, consider a group of scientists that every day perform different, but inter-related experiments within the bio-informatics⁴ application domain. Not only the generated experiment results are complex and cannot be managed using a relatively simple file system, but also the complete set of information concerning each experiment cannot be properly inter-linked in order to give the entire experiment more value. Furthermore, the results of an experiment cannot be fully exploited if information about the experimenter is missing, neither if the input data sets or environment parameters are not coherently available.

The second technique, *external data link* approach is much more effective than flat files for storing and managing large data sets. In this case, a database catalogue is used together with the file system, to efficiently manage distributed scientific information. The database will provide references to all objects that are stored locally or remotely at geographically distributed sites. This approach solves the problems related to the database overloads with huge objects and improves the system performance. The advantages of this approach includes, among others, the provision of a mean for managing huge amounts of data, provision of data in a secure manner, and distribution of data among geographically distributed nodes; where the nodes are usually those in which the data is generated or where it belongs to. [BAH 99, PWD⁺99].

The last approach, the so-called *one-database storage* approach, consists of storing both the binary data and the other general information of the application, into the same database. However, talking about binary data means data of very large size, which results degradation of performance when loading this huge data into the database itself. The database catalogue needs to be much better exploited for metadata cataloging, indexing, and proper searching facilities.

However, the emerging scientific and large engineering applications generally require more than what is provided by these approaches. New approaches need to be developed for the management of large data sets in order to improve the performance of the system while preserving the consistency of data and information visibility levels. One possible solution, suggested in this section, is to merge and extend the second and the third approaches

⁴Bio-Informatics is an interdisciplinary science that studies and explores biological issues and cases using and benefiting from the methods of informatics.

presented above into one distributed system consisting of a front-end catalogue database and a back-end distributed database server. In this approach, described in section 5.3.1.4, and referred to as the “*parallel/distributed database server approach*”, the meta-data (description of the data) and the annotations are kept in the database catalog, while the large multimedia data sets are stored at the distributed servers [BCG⁺97, WMP 98]. Queries on the data data are formulated against the database catalogue, and since this is generally a small amount of data, a good performance can be achieved by the query processing.

Following sections describe different strategies for data storage and mainly focus on an extended approach, which deploys a parallel/distributed database server for on-line object delivery to authenticated users/applications.

5.3.1.1 File System Approach

File systems store information in O/S files, and allow the storage of a very large amount of data over a long period of time. However, these files are in different formats and the programs accessing them are coded in different languages, which may result in data redundancy and inconsistency. Data in files is not automatically backed up, in order to guarantee its availability, a recovery system must be developed and set up [GUW 02].

The file system approach is still used by applications in which, the implementation is based on the manipulation of regular files. The developed application programs are fully dependent on the data files (see Figure 5.2). Thus, the structure of data files is embedded in the access programs and any changes in the structure of a file require re-compiling all programs that access this file. This approach, used by several applications in the past, cannot fully support many application requirements of the advanced scientific domains, where information needs to be inter-linked, compared to other data, and easily accessed. The file system by itself is not able to handle complex inter-linked huge data sets. Thus, applications based on file systems are hard to maintain and to extend [BAK⁺00, EN 00, GUW 02]. The inefficiencies of this approach are evident in:

- Maintaining the link between the inter-linked pieces of information, and comparing related data in different applications,
- Supporting the ability to query and modify the data using an appropriate query language, and their support for schema being limited to the creation of directory structures for files.
- Searching the stored information and supporting access to data items whose location in a particular file is not known.
- Preserving the system coherency, data scattered in various files of different formats.

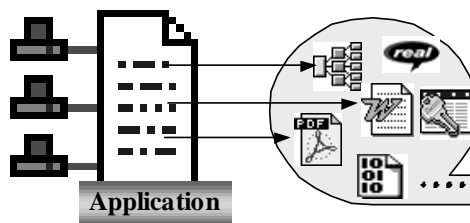


Figure 5.2: File System Approach

Furthermore, the file system approach cannot properly support the requirements for:

- Security for access to data, which can only be enforced at the level of operating systems and not further.
- Critical concurrent access control situations to files by several users, which may not be prevented.
- Data “sharing” though the Internet, due to the lack of proper security for access to data and concurrency control mechanisms.
- Individual user views on the data are not supported, and data items, for which the location within the file is not known, are difficult to locate and retrieve.

However, for simple cases inside one organization, where the data structure and the application requirements are simple, well defined, and not expected to evolve, it may be more desirable to use regular files. The usage of a database management system in such a situation may involve unnecessary overhead costs that would not incur in traditional file processing [EN 00].

5.3.1.2 External Data Link Approach

The external data link approach uses a database catalogue together with the file system, to efficiently manage distributed multimedia data sets. In the external data link approach, a single repository (catalogue) for meta-data and general information is maintained. The database catalogue (also referred to as metadata database) is defined once and then accessed by various users and applications. Scientists can use the metadata to locate and interpret data stored in the archive, including data generated by other scientists. This metadata provides permanent documentation of the data and becomes an integral part of the scientific data management system. Thus, the database catalogue stores the structure of the data, documents the contents and context of the data stored in the archive, and references the large binary objects that are available either locally or remotely at geographically distributed sites (Figure 5.3). This approach solves the problems related to centralized database overloads with huge objects, provides better retrieval performance, and improves the access to the data through a single database catalogue.

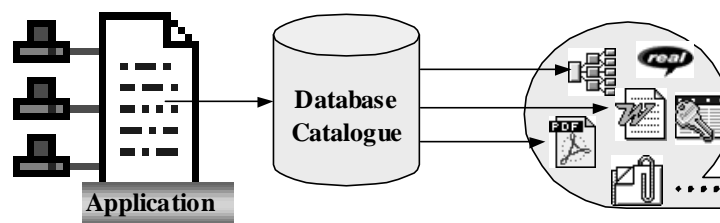


Figure 5.3: External Data Link Approach

In addition to the main characteristics listed above, the usage of this approach provides scientific applications with the following advantages [BAH 99, PWD⁺99]:

- Data is stored separate from the access programs (program-data independence).
- Data is stored at one place, either at the point where it is generated or where it belongs.
- Data is distributed so that it can be physically located closest to its intensive usage.
- Data is visible from any node within the cooperation community.
- Data distribution reduces the access bottlenecks at individual sites.

However, there are of course several limitations associated to this approach. The first problem faced when using this approach relates to the database catalogue consistency. In general, the referenced external objects are stored independently of the database catalogue. This later only contains a simple link to the external objects that are stored as regular files on different local and remote systems. Thus, these files can be updated or removed by local users within each system, without notifying the database catalogue maintainer, which may result to inaccurate or incorrect reference links.

The problem of database catalogue consistency could be addressed by developing a specific module that automatically and periodically checks the availability of the referenced objects against the database catalogue content, then the database catalogue will be updated regarding the new changes. If an external referenced object is missing for instance, the link to it will be removed from the database catalogue in order to avoid inaccurate references. Furthermore, log information can also be gathered and stored. The log information can support keeping track of the system updates based on some comparison of the size, the date, and the author of the referenced external objects.

A second problem concerns the security issues for the stored objects, where a more critical situation relates to data privacy and user visibility rights. Under normal considerations the files are usually hold in a public location, so when users request information from the system, their requests are evaluated against the database catalogue and they will be provided with the links to the proper objects that they can access and retrieve via http, ftp, and other data transfer protocols. Thus, typically the objects are not secure.

To solve the security problem within this approach, some applications develop a number of remote file servers, through which, the file access and user authentication are controlled based on the database catalogue information [BAH 99, PWD⁺99].

5.3.1.3 One-Database Storage Approach

As depicted in Figure 5.4, the *one-database storage* approach consists of storing the binary objects together with their meta-data and the other general information of the application, within the same database (being centralized or distributed) in a unified way, and typically in digitized format. This approach requires substantial re-engineering or extension of traditional DBMSs to directly query and manipulate the contents of the files stored in the database. However, this problem is not a barrier for research anymore, since several database systems (e.g. Matisse⁵, Oracle⁶, Jasmine⁷, Informix⁸, and DB2⁹) already support the storage of multi-media and binary objects of different formats (e.g. postscript, images, audio, video, documents, etc.). The storage of these types of data is made possible by the DBMSs via a set of binary large objects (the so-called blobs).

⁵Matisse: The Object Developer's Database System (<http://www.fresher.com>).

⁶Oracle Database System (<http://www.oracle.com>).

⁷Computer Associatestm-Jasmine (<http://www.cai.com>)

⁸Informix[®] information management solutions (<http://www.informix.com>)

⁹IBM DB2 Universal Database System (<http://databases.about.com/cs/db2/>).

The one-database-storage approach solves the problem of keeping the metadata database synchronized with the archive, by making the metadata and the archive one-and-the-same.

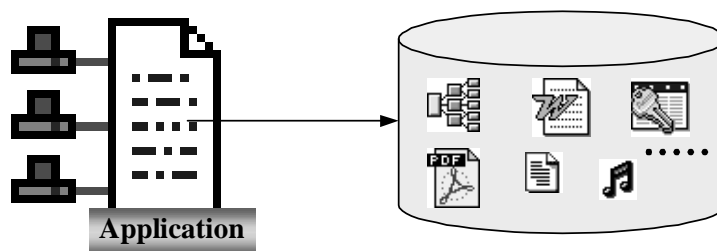


Figure 5.4: One-Database Storage Approach

However, here the binary data refers to data of huge size, where even the loading of these objects in the database affects the database performance, which can be better exploited for cataloging, indexing, and searching facilities. Another problem may be the access mechanisms to these objects, which require extra encoding/decoding facilities (embedded plug-ins) to be supported by the database software in order to properly manage the compact data formats of the binary objects.

In addition, the studies made on engineering and management of data from different applications in the domain of music industry [BAH 99a, BAH 99b], water distribution management [ABH 98a], biology and medicine [BAS⁺99], show that the information dealt with is of two main categories: the general information (so-called meta-data) and the raw data. The meta-data represents the description of a wide variety of data, which is accessed by a large number of users; where the user queries are issued against the meta-data. The raw data, on the other hand, is generally accessed and processed by the end-user scientists and experimenters.

From the usage point of view, the *one-database storage* approach that stores large binary objects in the same database together with the general information of the application is easier and more desirable. However, it reduces the database performance and efficiency. Let us consider one example application that is now being developed at the University of Amsterdam. This application is focused on the management of large data sets and the information handling in a bio-medical¹⁰ application [BKS 98], where the data originates for instance either from a simulation experiment or from medical scanners (CAT, MRI)¹¹. In such applications, the complete information about scans are “slice wise” stored in the database, and delivered at run-time, on demand. The largest objects that are stored in the database are the raw data sets for the slices, used for on-line simulation, visualization and exploration. Considering a normal bio-medical application that deals with dynamic grids of 1024 x 1024 16-bit pixels with 5000 slices over 50 time steps, the amount of raw data reaches several hundreds of gigabytes (around 524 GB). Our detailed study on these applications also shows that the general information only requires small disk storage capacity compared with the raw data that is, in most cases, in the range of hundreds of times larger.

¹⁰Bio-medical Engineering is an interdisciplinary field between medicine and engineering, which applies the most advanced technologies, principles, and skills developed in engineering to the world of medicine.

¹¹CAT stands for Computerized Axial Tomographic and MRI stands for Magnetic Resonance Imaging.

5.3.1.4 A New Approach: Parallel/Distributed Database Server

In many collaborative application domains, for instance, the electronic commerce and the large scientific applications, access to databases is concentrated on large structured (and unstructured) objects of “value”, and their related information (meta-data). In order to gain better database performance and access efficiency, learning from previous approaches, an alternative solution would be the utilization of a “distributed database server” to manage the large binary objects, separate from the general information. The proposed architecture deploys a parallel/distributed database server, that delivers objects (files) based on user authentication [BAH 99a, PH 98].

Figure 5.5 illustrates the architecture deployed within the parallel/distributed database server for scientific applications, where the followed strategy for data management takes advantages of both the *external data link* approach and the *one-database storage* solution. This approach uses a database repository to store the general information and a parallel/distributed database server (instead of file server) to store large objects. The utilization of database systems instead of file servers in such architecture enforces the issues related to security for access, concurrency control, and information visibility rights mentioned in previous sections 5.3.1.1 to 5.3.1.3.

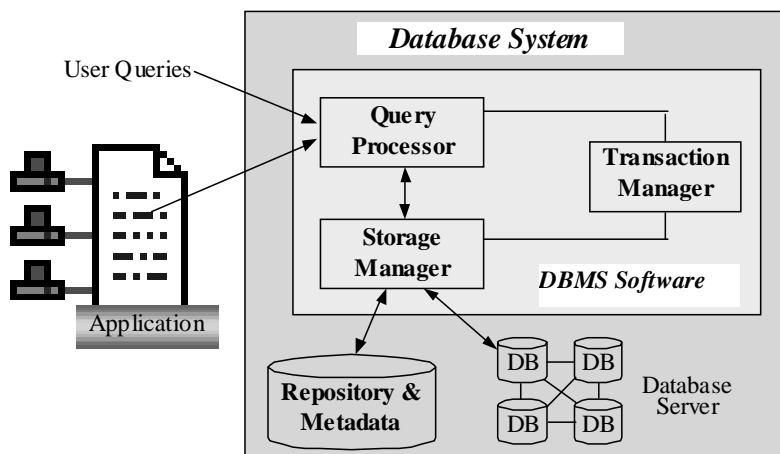


Figure 5.5: Architecture for the Parallel/Distributed Database Server

The parallel/distributed database server is well suited when high security is required (objects are encrypted in the database as lists of bytes that are difficult to access due to database security and hard to decode since they require special encoders/decoders that are only delivered to authorized users). This feature is very important for scientific applications, since it assures a minimum level of security for the private data, where users of the system will also be authenticated by the database server before getting served.

The parallel/distributed database server provides the proper base framework that can be adapted to handle huge amounts of raw scientific data. With this architecture, the database nodes of the distributed server are inter-connected, making it possible for specific experimenters to connect to any database server within the distributed server, and request an object, without the need to know where the object actually resides. If the request cannot be handled locally at that database node, it is automatically broadcasted to the other database nodes, and the result is forwarded back to the user.

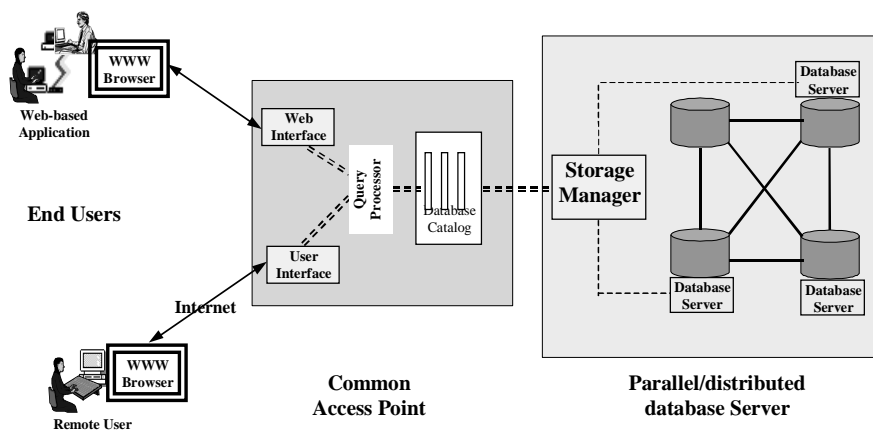


Figure 5.6: Parallel/Distributed server architecture: an Application Case

The followed approach is also suitable in the sense that the database can be used for different purposes. Figure 5.6 presents an application case of this architecture, in which, for instance in a Virtual Laboratory environment, the outside users and application interfaces are based on, and supported, through the database catalogue, while the scientists' experimental interfaces are based on and supported by both the database catalogue and the parallel/distributed database server.

5.3.2 Scientific Data Archiving and Cataloguing Using Dublin Core Standard

Scientific and industrial organizations are nowadays focusing on building technology infrastructures that are cost-effective and conform to their application practices, in which the scientific meta-data plays an increasingly important role, and brings a considerable value in terms of information retrieval, application maintenance, data integration, and support for user requests. Such infrastructures maximize the sharing and re-use of data, eliminate redundancy, and facilitate application integrity [DCMI 99, BA 00].

To improve the applications' flexibility and the end user's usability, current systems must provide information about the content and the quality of the data they hold and manage. This information can be provided through the so called meta-data repositories [BA 00]. Scientific meta-data is information about the application data, that gives descriptive information about the context, conditions, and characteristics of the data. Thus, meta-data serves as the binding mean that ties the various tools and technologies together at the application level.

This section aims at the design and development of a database system, to archive a wide variety of large scientific data sets. The process of data storage/acquisition, from different sources, within scientific applications, can follow one of the two approaches, which are presented and described in section 4.5.1.

1. The first approach builds a specific two-side-dependent interface, to directly store the data from its origins into the corresponding databases.

2. The second approach uses an intermediate step by storing the data in an intermediate standard format (probably OIF and/or XML). The standard format can be loaded into the compliant database systems.

The main focus of this section is to apply the second approach to scientific data, using the Dublin Core Meta Data Standard [WKL⁺98]. The Dublin Core (DC) standard describes better the content of scientific data and the context related to its generation. This section addresses the design of a data archive, using Dublin Core, as follows:

1. First it gives a brief description of the Dublin Core Meta Data Standard; a specific higher-level cataloguing/archiving schema for the scientific raw/processed data.
2. Second, it addresses the object-oriented representation of the Dublin Core meta-data; a representation that better suites to scientific applications and makes the definition of the DC elements and the relationship among them more elaborate.
3. Third, it extends the object-oriented DC model with additional meta-data terms; in order to support the scientific applications regarding their experimentation and data processing.

5.3.2.1 Dublin Core based Meta Data Design and Implementation

The DC Meta-data Initiative [WKL⁺98] is a cross-disciplinary international effort to develop mechanisms for the discovery-oriented description of diverse resources in electronic environment. The Dublin Core Element Set comprises fifteen elements [DCMI 99], which together capture a representation of essential aspects related to the description of data sources (e.g. publishing). The meta-data definitions, presented in [DCMI 99], and provided in Table 5.1, include both the conceptual and representational form of the Dublin Core elements. The *Label* provides a mnemonic single-word specification for the DC meta-data *Elements*; labels are simple enough to identify the corresponding elements in the schema. While, the *Definition* captures the semantic descriptions for these elements. For simplicity reasons, some detailed definitions are summarized.

Element	Label	Definition
Title	Title	A name given to the resource, by which the resource is formally known.
Creator	Creator	An entity primarily responsible for making the content of the resource. The Creator entity can be a person, an organization, or a service.
Subject & Keywords	Subject	The topic of the resource content, expressed as keywords, key phrases or classification codes.
Description	Description	An account of the resource content. Description may include but is not limited to: an abstract, table of contents, reference to a graphical representation of content or a free-text account of the content.
Publisher	Publisher	An entity responsible for making the resource available. Examples of a Publisher include a person, an organization, or a service.
Contributor	Contributor	An entity responsible for making contributions to the content of the resource. Examples of a Contributor include a person, an organization, or a service.
Date	Date	A date associated with an event in the life cycle of the resource. Typically, the Date is associated with the creation or availability of the resource. The date value is defined in a profile of ISO 8601.
Resource Type	Type	The nature or genre of the resource content. Type includes terms describing general categories, functions, genres, or aggregation levels for content.
Format	Format	Specifies the physical or digital manifestation of the resource. Format may include the media-type or dimensions of the resource and may be also used to determine the software, hardware or other equipment needed to display or manage the resource.
Resource Identifier	Identifier	An unambiguous reference to the resource within a given context. Example of formal identification systems include URI, URL, ISBN, and DOI (Digital Object Identifier).
Source	Source	A Reference to a resource from which the present resource is derived. The resource is referenced by means of a string or number conforming to a formal identification system.
Language	Language	A language of the intellectual content of the resource. The values of the Language element include a two-letter Language Code (ISO639), followed optionally, by a two-letter Country Code (ISO3166). For example, 'en' for English, 'fr' for French, or 'en-uk' for English used in the United Kingdom.
Relation	Relation	A reference to a related resource. The resource is referenced by means of a string or number conforming to a formal identification system.
Coverage	Coverage	The extent or scope of the content of the resource. Coverage includes spatial location (a place name or geographic coordinates), temporal period (a period label, date, or date range) or jurisdiction (such as a named administrative entity).
Rights Management	Rights	Information about rights held in and over the resource. The Rights element defines the rights management statement for the resource, or references a service providing such information. Rights information often encompasses IPRs and Copyrights.

Table 5.1: Dublin Core: Elements Description

5.3.2.2 Dublin Core Object Modeling

Current implementations for the Dublin Core model are addressed by “pseudo-hierarchical dot notation” (e.g. DC.Creator.Email) or in best cases, using the relational model. However, complex applications within e-science domains, require data models to properly reflect the real world entities similarly to their existence in reality. To properly model the Dublin Core elements and benefit from the utilization of the DC model, we address extension of the DC meta-data definition in terms of object-oriented modeling and object references. For instance, the object oriented modeling of the *creator*, *publisher*, *contributor*, and *rights*, better suits in scientific applications by allowing the creation of a more comprehensive relationships among objects. The object-oriented modeling of these elements allows their representation as real-world entities, and makes them related to each other via the means of relationship associations.

Figure 5.7 illustrates the design of an object-oriented database definition for Dublin Core schema elements that we introduce for use within the VL project, using the Unified Modeling Language [UML 98]. The Object Definition Language (ODL) schema for this meta-data is also provided as a subset of the enhanced object-oriented Dublin Core ODL schema definition, presented in Table 5.2. Users of Dublin Core can find more details and useful references related to the use of Dublin Core in [Hill 01].

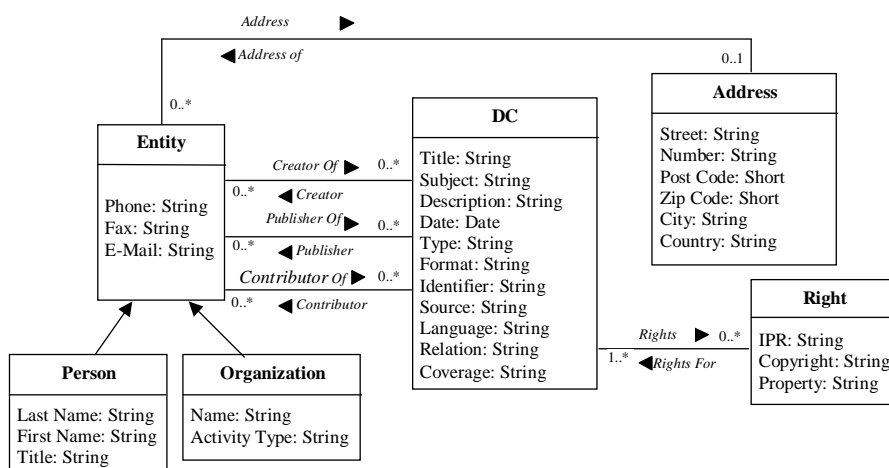


Figure 5.7: An Object-oriented schema for the Dublin Core meta-data

5.3.2.3 Enhanced Dublin Core Data Model

In order to satisfy the complex requirements for advanced scientific applications in terms of scientific data representation, additional meta-data vocabularies related to scientists and their performed experiments need to be added to the DC data model. As such, the extensions we introduce must allow to distinguish between *raw* and *processed* data, to specify the different kind of *processes* used for data manipulation, to indicate the *devices* from which data is being collected, and to support the *reviewing* for scientific experiment results.

Figure 5.8 illustrates a generic object-oriented schema representation for the VL data archiving. This schema is based on and enhances the DC meta-data model presented in Figure 5.7. The extensions to this schema first allow the distinction between raw and processed data, and second provide facilities for reviewer's comments, processing methods definition, and specifications about the devices generating this data.

The extended part of the DC schema in Figure 5.8 presents the necessary additional entities required for the VL experimental environment. For instance, a *Process* is defined by a *name*, a *description*, and a set of *parameters*; it uses one or several *raw data* as input, and produces a set of *processed data*. It is possible that one or more persons review the produced data. The DC schema elements are related to each other through a number of relationships describing the associations between these entities. The multiplicity range (cardinalities) of the relationship associations define constraints on the data, a process for instance must have at least one input (raw data) and produce one or more output results (processed data).

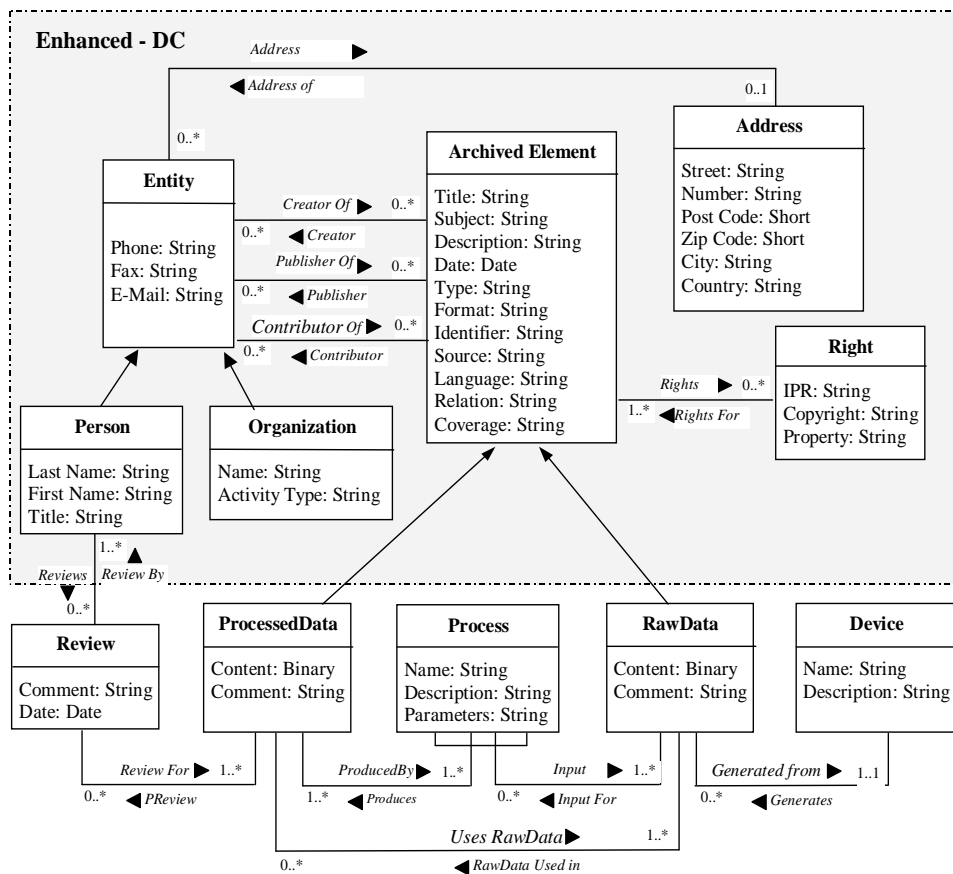


Figure 5.8: Enhanced Object-oriented schema definition for the VL archiving environment based on the Dublin Core standard

Table 5.2 presents the ODL schema definition for the enhanced object-oriented DC meta-data, illustrated in Figure 5.8. The provision of the ODL definition presents the advantage of direct loads/creation of the DC schema into any database that is ODMG compliant. The use of enhanced Dublin Core metadata for modeling scientific applications has proven its

validity via its deployment, as a base, for the data definition within different applications, which are being developed within the VL information management framework. Among these scientific applications, we enumerate a generic model for experimental environment [KAH 01], DNA micro-array and gene expression [KAB⁺01], Mass Spectrometry metadata analysis [EAG⁺01], and information management for material science applications [FAE⁺01].

<pre> //Extended Dublin Core ODL Definition //----- interface ArchivedElement : persistent { attribute String Title = ""; mt_make_entry "make-entry"; attribute String Subject = ""; mt_make_entry "make-full-text-entry"; attribute String Description = ""; attribute Date Date = ""; attribute String Type = ""; attribute String Format = ""; attribute String Identifier = ""; attribute String Source = ""; attribute String Language = ""; attribute String Relation = ""; attribute String Coverage = ""; relationship List<Entity> Creator inverse Entity::CreatorOf; relationship List<Entity> Publisher inverse Entity::PublisherOf; relationship List<Entity> Contributor inverse Entity::ContributorOf; relationship List<Right> Rights inverse Right::RightsFor; }; interface Right : persistent { attribute String IPR = ""; mt_make_entry "make-entry"; attribute String Copyright = ""; attribute String Property = ""; relationship List<ArchivedElement> RightsFor[1,-1] inverse ArchivedElement::Rights; }; interface Entity : persistent { attribute String Phone = ""; attribute String Fax = ""; attribute String EMail = ""; relationship List<Address> EntityAddress[0,1] inverse Address::AddressOfEntity; relationship List<ArchivedElement> ContributorOf inverse ArchivedElement::Contributor; relationship List<ArchivedElement> PublisherOf inverse ArchivedElement::Publisher; relationship List<ArchivedElement> CreatorOf inverse ArchivedElement::Creator; }; interface Person : Entity : persistent { attribute String LastName = ""; mt_make_entry "make-entry"; attribute String FirstName = ""; attribute String Title = ""; relationship List<Review> Reviews[0,-1] inverse Review::ReviewedBy; }; interface Organization : Entity : persistent { attribute String Name = ""; mt_make_entry "make-entry"; attribute String ActivityType = ""; }; </pre>	<pre> interface Address : persistent { attribute String Street = ""; mt_make_entry "make-entry"; attribute String Number = ""; attribute Short PostCode ; attribute Short ZIPCode ; attribute String City = ""; attribute String State = ""; attribute String Country = ""; relationship List<Entity> AddressOfEntity inverse Entity::EntityAddress; }; //Dublin Core Meta Data Extention //----- interface Review : persistent { attribute String Comment; mt_make_entry "make-entry"; attribute Date Date; relationship List<Person> ReviewedBy[0,-1] inverse Person::Reviews; relationship List<ProcessedData> ReviewFor[1,-1] inverse ProcessedData::Preview; }; interface ProcessedData:ArchivedElement:persistent { attribute Binary Content; attribute String Comment; relationship List<Review> Preview inverse Review::ReviewFor; relationship List<Process> Uses[1,1] inverse Process::UsedFor; relationship List<RawData> UsesRawData[1,-1] inverse RawData::RawDataUsedIn; }; interface RawData : ArchivedElement : persistent { attribute Binary Content; attribute String Comment ; relationship List<Process> InputFor inverse Process::Input; relationship List<ProcessedData> RawDataUsedIn inverse ProcessedData::UsesRawData; relationship List<Device> GeneratedFrom[0,1] inverse Device::Generates; }; interface Process : persistent { attribute String Name = ""; mt_make_entry "make-entry"; attribute String Description ""; attribute String Parameters ""; relationship List<RawData> Input[1,-1] inverse RawData::InputFor; relationship List<ProcessedData> UsedFor inverse ProcessedData::Uses; }; interface Device : persistent { attribute String Name = ""; mt_make_entry "make-entry"; attribute String Description ""; relationship List<RawData> Generates inverse RawData::GeneratedFrom; }; </pre>
---	--

Table 5.2: Enhanced ODL schema for the VL archiving environment based on the Dublin Core standard

5.4 Universal Database Access - Based on Standards

The work presented in this section describes the universal database access interface (called UDBA [Ben 00a]). The UDBA is a web-based framework, which is achieved through a set of functions for data and meta-data manipulation, that are supported by the combination of database technologies with current standard tools for data access such as ODBC and JDBC. The provision of these functions for accessing the internal structure of a database (data and schema) facilitates the process of creating intelligent web-based and non web-based applications. Therefore, universal database access is achieved via the development of generic tools through which, users will be able to access several data sources regardless of their internal structure, data types, and location.

The development of universal database access interface brings three main advantages to the work in the area of information management and interoperation.

1. First, it presents a flexible interface to easily manage the database schema and objects through a web environment to which ordinary users are quite familiar.
2. Second, it allows to better explore the advanced features supported by the object-oriented/object-relational DBMSs and provides means to improve some of these features supported by their object database connectivity mechanisms (e.g. class inheritance, object identifier, and cross-reference relationships).
3. Third, it extends the implementation of the data types as required by complex scientific applications such as the multi-media information, large data sets, and complex inter-linked objects.

The implementation of the universal database access interface is based on the following software technologies:

- Matisse object-relational DBMS [Mt 01] for schema representation and data storage.
- Active Server Pages programming model [ASP 01], which allows dynamic and interactive Web pages to be generated on the fly from the Web server.
- ActiveX Data Objects [ADO 01] programming extension for database connectivity. The primary benefits of ADO are ease of use, high speed, and low memory overhead.
- Matisse ODBC driver for accessing the heterogeneous databases, structure and objects, via a common interface
- JavaScript and VBScript languages, which overcome the limitations of HTML and allow the creation of functions, embedded within HTML code.

Using the universal data access interface, users do not need to know much about the data structure of the underlying data source. Users are only asked to specify the data source's name. Then the application connects the user to the specified database, reads the schema structure, and automatically presents to the user, in a very flexible manner, a set of concepts, through which he/she can freely navigate and explore the database structure (schema) and its instances (data).

For the VL information management system, the universal database interface provide a facility similar to what the database vendors deliver as a "client interface" to manage the data and the schema for the database that runs on the DBMS server. The main difference is that the universal database access interface can be considered as a complementary tool

to the specific database interface that can also be accessed by different users through the Internet. Therefore, universal database interfaces bring to the DBMS server the advantage of: (1) being able to run on a web server environment, which makes it platform independent and widely used, and (2) since the UDBA interface is built on top of the middleware and standard solutions, it can be considered as a generic framework that can interface many databases that are compliant to standards.

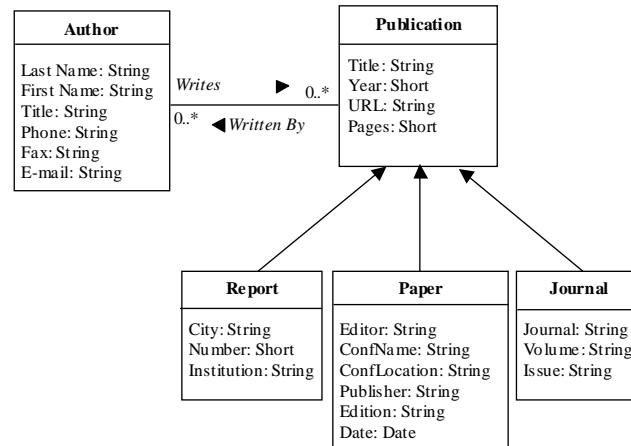


Figure 5.9: Example simplified Data Model for Authors and Publications

Figure 5.10 presents a screen shot of the universal database access interface and illustrates the four main components of this interface, namely, the *Database Connection*, the *Query Execution*, the *Results Presentation*, and the *Object Creation* modules. The example, shown in this section, is based on a database for *Authors* and *Publications*, for which a simplified model is presented in Figure 5.9.

As depicted in Figure 5.10, the implementation of the UDBA Web interface utilizes the frames mechanism. Therefore, the interface consists of four browsing areas (frames), each handling a set of sub-tasks of the system. The use of frames in this context allows the proper representation of complex objects and facilitates the navigation among those objects that are related to each other. The four frames of the UDBA interface consist of (1) database connection (at the top left), (2) schema exploration and query formulation (at the top right), (3) results presentation (at the bottom left), and new objects creation (at the bottom right).

The general steps for information access through the universal database access, pointed to in the figure by circled numbers 1 to 4, are briefly described below. Further details of the functionalities provided by the universal data access are described in section 5.4.5.

1. First, the user has to specify a data source name, and press the *Connect* button. The interface then connects to the corresponding database, reads its internal structure, and presents to the user a friendly interface. The database *Connection Module* handles this step, and provides the user with the complete structure of the underlying database through which he/she can freely navigate among the database components (tables/classes), select the desired items (or specify a query), and specify his/her desired output format for the results.
2. Second, the user has either to select a class/table name or specify an SQL query to be submitted to the data source server, choose his/her preferred output format for the

results, and press the *Query Execution* button. The *Query Execution* module extracts user inputs, checks for syntax and semantic errors, and builds the corresponding query to be sent to the database.

- Third, the query results are returned to the user. The *Result Presentation* module allows the presentation of the query results to the user according to his/her desired format. The output format can be HTML, in the form of a table, or using a standard data exchange format such as XML and OIF. At this stage, the user can also choose to add/create a new object for the selected data type into the database.
- Finally, the user has the possibility to create new objects in the database. The *Object Creation* module provides a flexible facility to dynamically create objects and store them into the database. This facility is provided to the user within the data representation module.

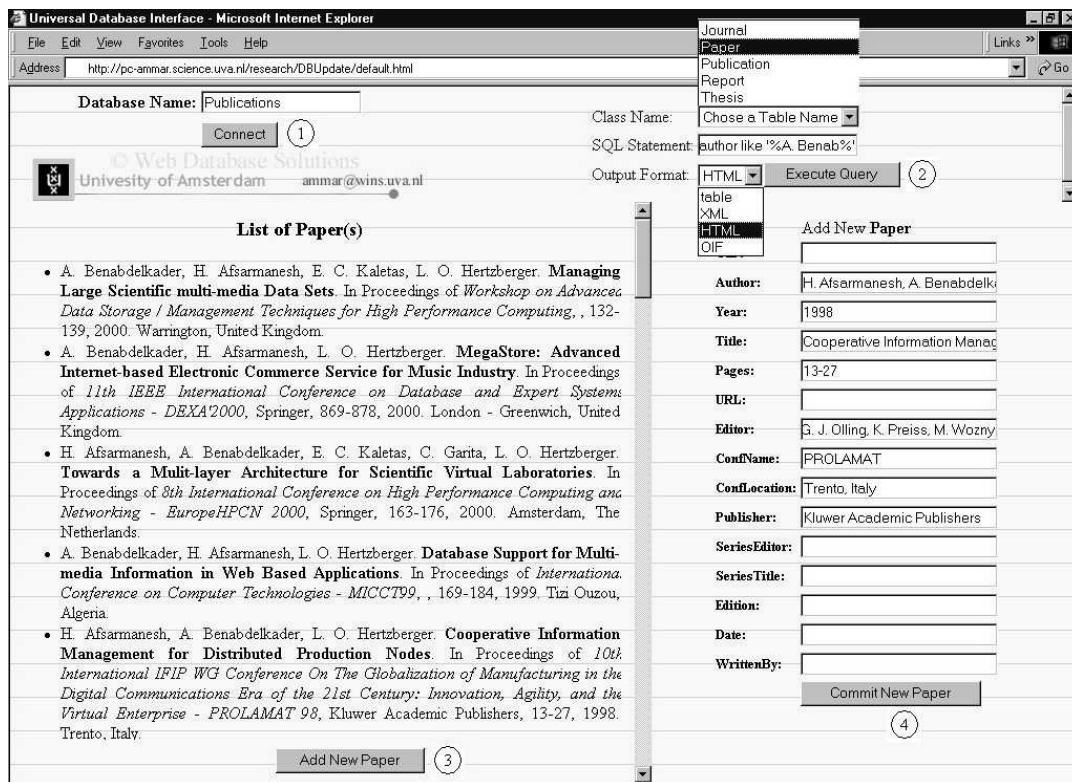


Figure 5.10: Universal Database Access Interface

Please notice that for every step, there is a module supporting its functionality. These modules that constitute the universal database access framework are further described in more details within the following sections.

5.4.1 Database Connection Module

As depicted in frame (1) of Figure 5.10, the connection to a database server is simply done by specifying a database name and requesting the connection¹². When activating the connection module, this later automatically connects to the specified database, reads the structure of its schema, and organizes it in a simple format for the user. The format consists of:

- A pull-down scroll menu consisting of all the *database classes*, from which, the user can choose one class to browse or manipulate its instances.
- An input text area, where the user can specify an *SQL statement* to be submitted to the database. The SQL statement can be a select query, an update statement, an insert or a delete command.
- A pull-down list, consisting of all possible output *representation formats* for the selected class objects.
- The *query execution* button through which, the user validates and activates the query execution process.

5.4.2 Query Execution Module

The query execution module extracts the user input/selection, checks for errors, builds the corresponding query according to the user request, and launches the query execution process. The query execution module performs according to the following strategy:

- If an SQL statement is specified, it executes it without checking the class name in the pull-down menu
- If a database class/table name is selected, the query execution module reads the selected class name from the pull-down scroll menu, constructs the proper select query, for instance *select OID, * from <class name>*, and performs the query execution.
- If neither of the two inputs is specified, the query execution module prints a warning message asking the user to either select a class name or specify an SQL statement.

5.4.3 Results Presentation Module

The result presentation module extracts the corresponding output information for the user's requests and organizes the query results according to the specified format. Different layout possibilities are illustrated for presenting the information to the user. The user can choose either an HTML presentation, an XML format, an Object Interchange Format (OIF), or a table format presentation.

The result presentation module builds a dynamic output for presenting information to the end user. The presented output is dynamic in terms of columns and lines; and is based respectively on the selected attributes, and on the objects available for the chosen class.

Frame (3) in Figure 5.10 illustrates the output format for publications and authors using an HTML format. While, another example is presented in frames (3) and (4), which show on the left hand side the ArchivedElements with the XML format, and on the right hand

¹²Specification for the underlying database name, location, and access mechanisms are supported and provided via the ODBC data sources facilities.

side the IPRs¹³ related data using the table output format. Query results are represented on the screen and one page at a time, through which the user can navigate and scroll forward and backward.

5.4.4 Object Creation Module

In addition to the information retrieval and data representation, the dynamic creation of new instances for these classes that are being explored, can be achieved through the *Add New<object>* button, which generates the corresponding form for the data input and loads it into the database.

The object creation module allows the generation of dynamic forms for input to be filled up by users, to create new instances. Input forms are based on the structure of the current class that is being explored. Frame (4) in Figure 5.10, for instance, presents an input form for creating a new instance of the class *Paper*.

After filling the input form, the data specified by the user will be submitted to the database in order to create a new object (instance) for the considered class. Submission of the input data is accomplished by the *Commit Event* module, which dynamically reads the user inputs, builds the SQL command, and inserts the new object into the database.

In case of any input data errors (e.g. wrong type, invalid range), these errors will be identified by the system and reported to the user. Therefore, the updates to the database are not committed unless the user fixes all errors. At this level, providing the user with the possibility to go one screen backward and correct his/her errors (without losing any data), facilitates the objects creation process.

5.4.5 Further Benefits

In addition to the various advantages of the UDBA framework as presented within the previous sections, in the following sub-sections, we will provide more details and exemplify the three concepts that illustrate the strength simplicity, openness, and flexibility of the universal database access framework.

5.4.5.1 Dynamic Query Definition and Results Presentation

The universal database access framework presents a dynamic facility for query definition and results presentation. The flexibility of the framework is made possible through the run-time access to data and schema structures of the underlying data source.

The flexibility in data representation is supported in terms of database structure characterization, object instances creation, and layout for the output results formatting.

- Depending on the defined attributes within each class, the user can select a set of properties and constraints to be considered for the query execution.
- Through the SQL statement, the user can also restrict the objects to be retrieved, via the specification of the condition predicates.
- According to the user specifications for the layout in data presentation, the results of a query are formatted to fit the selected formatting template.

¹³ArchivedElement and IPR concepts are defined in section 5.3.2.3

The flexibility for query specification and conditions definitions are provided through the support of SQL commands for INSERT, UPDATE, DELETE, as well as for database schema manipulation. The following examples present in more details these features through a set of examples from the ‘*Authors and Publications*’ data model defined in Figure 5.9:

- Creating new instances for classes defined within the database schema are supported through the INSERT command. For instance, the statement: `INSERT INTO Paper (Title, ConfName) VALUES ('MegaStore', 'DEXA 2000')`, creates a new instance of the class *Paper* and assigns the values *MegaStore* and ‘*DEXA 2000*’ respectively to the attributes *Title* and *ConfName*.
- Updates to the database are supported through the UPDATE query statement. For instance, the SQL statement: `UPDATE Paper SET (Pages, ConfLocation) VALUES ('869-878', 'London, UK') WHERE title LIKE 'MegaStore%'`, assigns new values ‘*869-878*’ and ‘*London, UK*’ for respectively *Pages* and *ConfLocation* of the class *Paper*, when the condition title “LIKE ‘MegaStore%’” is satisfied.
- Removing objects from the database is supported through the DELETE command. For instance, the simple delete query: `Delete Paper where Title = ''`, removes all instances of the class *Paper* for which the *Title* is empty, thus, it can be used for instance to clean the database from some incomplete data.
- Exploring the structure of the underlying database is supported through a set of concepts for schema manipulation and is implemented using the Matisse DBMS.

5.4.5.2 Multi-level Navigation Through Relationships

Complex Objects in scientific and system engineering applications are characterised by a set of relationship properties that link related objects together and form a network of entities. A good example for complex inter-linked entities is the conceptual schema for the Virtual Laboratory archiving environment, presented in section 5.3.2.3. In this environment, for instance, the entity *ArchivedElement* is linked to several entities such as *Publisher*, *Creator*, and *Right*. These entities are inter-linked through the concept of relationship definition.

During a user session and along his/her navigation, the universal database access interface automatically determines all the links among the objects in the database and provides the possibility of navigational access among them. An example that illustrates the navigational facility provided by the framework is presented in Figure 5.13 (frames 3 and 4), in which the presented results are also augmented with the relationship links from each object to all other objects that are related to it. For instance the possibility to navigating from an *ArchivedElement* object and explore other objects related to it (e.g. *Right*, *Contributor*, *Creator*, and *Publisher*). These reference links allow the user to easily navigate through the database objects and freely explore the complete scientific data sets as they exist in real world.

The navigational mechanism of the system is implemented via the relationship capability offered by the Matisse DBMS¹⁴. This mechanism provides fast and direct access among inter-linked objects; thus, it eases the user interaction and navigation through the database objects.

¹⁴In addition to its support for SQL ANSI standards, the Matisse ODBC driver provides extra features related to object-orientation and multi-media data manipulation.

5.4.5.3 Multi-Task Modules

The universal database access interface is an advanced Web framework that provides a flexible interface based on the knowledge acquired from the database structure, object instances characterization, and the output results presentation.

The development of the UDBA framework, presented above, benefits from modular design and implementation. Its complete development consists only of four modules that perform all the tasks required for the schema and data manipulation. The multi-task functionality of each of these modules is supported via a set of parameters, which are dynamically adjusted based on the data characterization and users' specifications. The *Result Presentation* module for instance, is responsible for representing in a simple and flexible way any type of data generated either by a select query or an update statement, according to the user's specifications and preferences. Similarly, the *Object Creation* module is a flexible component, which generates correspondent input data forms for classes defined within the database schema, and builds the proper insert query to be sent to the database, regardless of the structure of the information and the size of data that it contains.

5.5 Data Access Security and Information Visibility (Safe/Reliable Data Export)

The work described in this section concentrates on designing and implementing an interface to support role-based access rights facility, allowing organizations to share part of their data with certain other organizations. More details concerning a full description regarding data access control principles and mechanisms is outside the scope of this thesis, the reader can refer to the various approaches that have been published over the past few years. Among the published proposals, [SS 94] addresses principles related to access control, [Bald 90] focuses on naming/grouping privileges to simplify security management in large databases, and [SC 96, SF 94, Tho 91] proposed access control extension mechanisms to support role-based models. Other publications concentrate on the applicability of a role-based data access control to solve security problems in Intranet environment [TC 97], and to support information access rights and visibility levels in Virtual Enterprises [FAG⁺00, GAH 01].

The safe/reliable data export framework is a user-friendly web interface to access and retrieve a wide variety of data, based on predefined export schemas¹⁵. The access to data, through those export schemas, defined on top of existing databases, is achieved via a role-based access control.

The implementation of the safe/reliable data export interface is based on the universal database access framework presented in section 5.4, and extends it to support access rights and visibility levels to a subset of information, based on export views definition and user's (organization's) roles assignment. Roles are used to simplify the description of allowed access characteristics to database objects. The approach for granting user permission to access a subset of the information within each application is achieved through role assignment.

The scope of access to database objects using the Safe/Reliable data export is achieved through the following steps:

- Create roles according to the job functions (e.g. within the organizations),

¹⁵Export schemas definition within this section refers to views definition augmented with the concept of role-based access control.

- Grant users permissions (access authorization) based on these roles,
- Define export views based on database objects, and
- Assign the defined export views to the roles on the basis of their responsibilities.

The remaining two sections illustrate the adaptation of a role-based access control to existing applications, in which section 5.5.1 addresses a role-based access control definition, a facility helping the application developer in defining the necessary mechanisms for creating a role-based access control; while section 5.5.2 provides an interface for data publishing based on predefined export views and users authentication.

5.5.1 Role-based Access Control Definition

Figure 5.11 illustrates the schema definition for the meta-data characterising the role-based access control to export views. This schema is defined for the implementation of the safe/reliable data export interface. As described later in this section, when necessary, this generic data model will be augmented to different database descriptions, within several applications of the VL, in order to control and restrict external access to their data. In order to support the database administrator with the creation of export views based on the fusion of several database objects, a class named *Element* is defined to cope with this issue, thus, the designed model supports the definition of export views based on several joint classes.

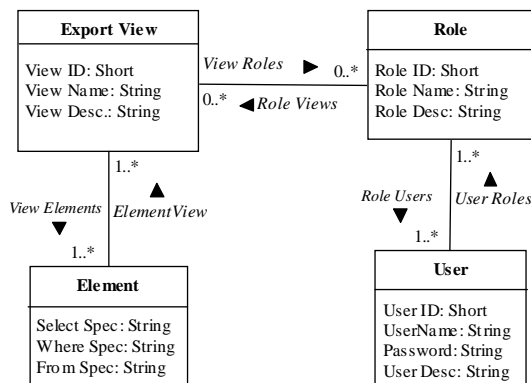


Figure 5.11: Schema Definition of the role-based Access Control with Export Views

The implementation of the safe/reliable data export framework is supported via the development of related tools to create export views, define roles, and assign users to those roles based on their privileges and access rights. As an example of these tools, we present in Figure 5.12 a screen shot of a user-friendly interface to create export views (similar interfaces are also available for defining users permissions and roles). This example also illustrates the steps involved in defining export views, including:

1. The database administrator (DBA user) connects to a given data source by simply specifying the database name and pressing the *Connect* button.
2. DBA user selects a class name from a list provided by the system. The list corresponds to the set of classes extracted from the structure of the underlying data source.

- DBA user proceeds with the export view creation, by selecting a set of attributes and specifying optional condition predicates.

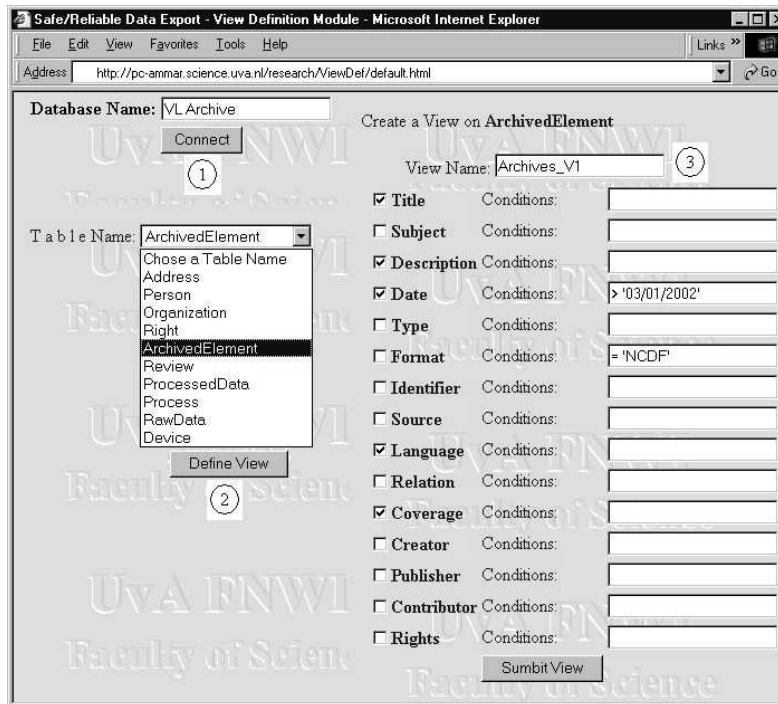


Figure 5.12: Interface for Views Definition

Once the user presses the *SubmitView* button, the corresponding commands (statements) for the view creation will be built on the fly and committed to the specified database. Two specific cases are supported here:

1st Case: If the underlying database system supports the definition of views (e.g. Oracle DBMS), an SQL command is formulated and executed, creating the export view. In this case the corresponding SQL statement will be:

```
Create View Archives_V1 as
Select Title, Description, Date, Language, Coverage
From ArchivedElement
Where (Date > '03/01/2002' and Format = 'NCDF').
```

2nd Case: If the underlying database system does not support the built-in components for the definition of views and roles (e.g. Matisse DBMS), as it is the case for most object-oriented database systems, the database schema for the corresponding application must be augmented with the data model of the role-based data access control. In this case the following specifications are realized for the elements of the export view defined in Figure 5.12:

- **Select Specification:** *Select Title, Description, Date, Language, Coverage*
- **From Specification:** *From ArchivedElement*
- **Where Specification:** *Where (Date > '03/01/2002' and Format = 'NCDF')*

Other interfaces are also developed for the purpose of creating export schemas, including user definition, and the roles assignment for users and groups. In terms of flexibility and user-friendly facilities, the implementation of these interfaces is similar to the one for export view definition.

5.5.2 Flexible Role-based Access Interface

The role-based access interface, primarily developed for data publishing to outside users, relies on safe and reliable mechanisms for data export and publishing. The interface is safe and reliable in the sense that it is based on the definition of roles, defined at different access levels. A VL user must be authenticated by the system before any attempt to access information. In addition, users are not aware about the complete structure of the underlying database, each user only sees the part of the database for which he/she has gained the proper access rights.

The screenshot displays the 'Safe/Reliable Data Export Interface' in a Microsoft Internet Explorer browser window. The address bar shows the URL: `http://pc-ammar.science.uva.nl/research/DataExport/default.html`. The interface includes a login section with the following fields and buttons:

- Database Name:** VL Archive
- User Login:** ammer
- User Password:** [masked]
- Class Name:** ArchivedElement
- SQL Statement:** [empty]
- Output Format:** XML
- Buttons:** Connect (1), Execute Query

Below the login section, there is a 'List of ArchivedElement(s)' section (3) displaying XML data for three records. The XML data includes fields such as Title, Subject, Description, Date, Type, Format, Identifier, Language, Relation, Coverage, Publisher, and Creator.

To the right of the XML data, there is a table titled 'Browsing Right Instances' showing details for three records. The table has the following columns: Num, OID, IPR, Copyright, Property, and RightsFor.

Num	OID	IPR	Copyright	Property	RightsFor
1	655	RHAD	© 1997	Rights History and Archives Division	62b
2	658	ASL	© 1997	Arizona State Library	62b
3	65b	APR	© 1997	Archives and Public Records	62b

Below the table, there is an 'Upload Data' button (4).

Figure 5.13: Safe/Reliable Data Export Interface

Figure 5.13 illustrates an example of the safe/reliable data export interface for an external user of VL connected through the Web. The interface tool, first checks and authenticates the user connection for the selected data source, in this case the VL archive database (following the example of figure 5.12), and only then, it provides the user with a set of information (in

this case the class names including the `ArchivedElement`) based on his granted access rights and visibility level. Therefore, the “user” access to the database objects is restricted via the assigned roles, and users that do not gain permissions to the data source are prohibited from any access to those objects.

Once the user is authenticated for accessing a selected data source, a connection will be established to the underlying database, and the system will provide the user with some interaction facilities to those database objects for which access permissions are granted. From the safe/reliable data export interface, the user can perform the following tasks:

- Select a class/table name and browse or retrieve its instances. At this level, the user can also specify an SQL query to be executed and evaluated against the underlying data source.
- Choose an output format for the data presentation, currently the framework supports the following formats: HTML, XML, OIF, and table format.
- Submit a query for execution on the database, and receive back the results according to the specified format.

The Query Execution module checks the user input, creates the appropriate query, restricted to his/her export view, to be sent to the database, and launches the Results Presentation module, which takes into consideration the output format specified by the user.

In addition to presenting the results on the screen, the framework also provides its users with the following possibilities (Figure 5.13, frame (3)):

- Possibility to *Upload* the query results and save them locally for future use and examination. The results are stored according to the format chosen by the user.
- Possibility to *Navigate* among the database objects through the defined links and relationships to other objects. Each object is augmented with a set of links referencing all the objects to which it is related.

5.6 Physical Database Performance Analysis

Nowadays in all organizations large or small, databases constitute the most critical elements, as the brain handling the organization’s information. They ease the storage and retrieval of massive amount of information, and provide the proper facility for multi-user information sharing and concurrent access control. DBMSs provide major facilities to the application developers, thereby, they are also considered as an important factor on determining the application performance in terms of data storage and information retrieval.

A main approach in this direction is to address the issue of scientific database system’s performance, especially for storage/retrieval of large binary objects within the DBMS itself. In order to achieve the most efficient physical implementation for VL, the main focus within this section will then be to address some benchmarking tests at the physical level of the Matisse DBMS, which is currently used within the VL project. Therefore, this section addresses the results of the physical database performance analysis regarding the manipulation of large scientific data sets, when using the object-oriented Matisse database system as the base for implementation.

In order to properly conduct these data access performance tests at the physical level of Matisse DBMS, first we have designed and developed of a set of basic functions to access

large database objects within the Matisse database. Second we have performed some tests to evaluate and analyze performance of the Matisse database when storing and retrieving large binary objects.

5.6.1 Specific Functions to Access Binary Large Objects (Blobs)

A set of specific functions to read/write large objects (blobs) from/to Matisse database are defined and implemented. These functions facilitate the access to large objects, which are directly stored/retrieved into/from the Matisse database as a list of binary bytes. Four functions are developed for the management of large binary objects [Ben 00b]. Each of these specific functions is implemented based on a set of elementary functions, provided by the Matisse C++ API interface.

- **ConnectDatabase** *<Host Name>* *<Database Name>* : Connects to a distributed database *<Database Name>* that runs on a remote host *<Host Name>*
- **LoadBlob** *<Class Name>* *<Attribute Name>* *<File Name>*: Loads a binary large object *<File Name>* into the database as the value for the attribute mentioned as *<Attribute Name>* defined within the class *<Class Name>*.
- **ReadBlob** *<Class Name>* *<Attribute Name>* *<Entry Point>* *<Object ID>*: reads a binary object *<Object ID>* using the entry *<Entry Point>* from the attribute *<Attribute Name>* defined within the class *<Class Name>*.
- **DisconnectDatabase** *<Host Name>* *<Database Name>*: Disconnects from the database *<Database Name>* on host *<Host Name>*.

These functions hide the programming complexities from the user when accessing binary objects, thus, provide easy and simple access for the user. These functions are also used for the implementation of the benchmarking tests, described in the next section.

5.6.2 Benchmarking Tests For Matisse Database System

As a part of the design of most efficient physical implementation for the VL project, a set of benchmarking tests are performed on the Matisse DBMS to evaluate the performance of different implementation approaches for storing/retrieving very large binary objects in the database. Large real data sets from the application case of VISE, focused on visualization and simulation functionalities of the VL project (described in section 5.2), are used as input for the benchmarking tests. These tests are performed for read and write accesses using the functions described in section 5.6.1 and for two different implementation-configurations in Matisse. Namely, first a Matisse database that uses a normal disk (regular file managed by the system), and second using a raw disk partition (managed by the DBMS itself). For this benchmarking of the Matisse database system, the following test case input was designed and applied:

- 100 objects are used for both purposes of storage and retrieval accesses,
- The size of these objects range between 73 Kilobytes and 12 Megabytes,
- The total size of the 100 objects (loaded into the database) is around 570 Megabytes,
- The objects' storage/retrieval starts with object number one (which is of the smallest size: 73 KB) and gradually increases in size till object number 100 (which is of the largest size: 12 MB),

- The difference in size between every two loaded objects N and $N + 1$ is approximately 70 Kilobytes, where $N = 1 .. 99$,
- The database software runs on the Arches machine at the University of Amsterdam, which consists of dual Pentium II processors with 512 MB memory, 9 GB disk storage, and supports several network communications (Fast-Ethernet, Myrinet, and Gigabit Ethernet),
- To reach the best physical performance, the database configuration, which is chosen based on the available hardware for VL and also considering the suggestions from the Matisse DBMS developers, is as follows:
 - Database silo size (total storage capacity): 2 Gigabytes
 - Database Bucket size: 64 (64 * 512 = 32 KB)
 - Database cache size: 4000 (4000 * 32 K = 128 Megabytes)

These tests are very important in the sense that they allow us to determine the best data access performance that can be reached when storing large objects within the DBMS.

Figure 5.14 illustrates the Matisse database performance when storing/retrieving the 100 objects, of different sizes, one by one. The X-axis values represent the size of the individual objects being stored/retrieved, each illustrated by its size expressed in Megabytes. The Y-axis values represent the database access time for both storing (write) and retrieving (read) of each object, expressed in seconds. More details concerning the input/output data used for the benchmarking tests are given in [Ben 00b].

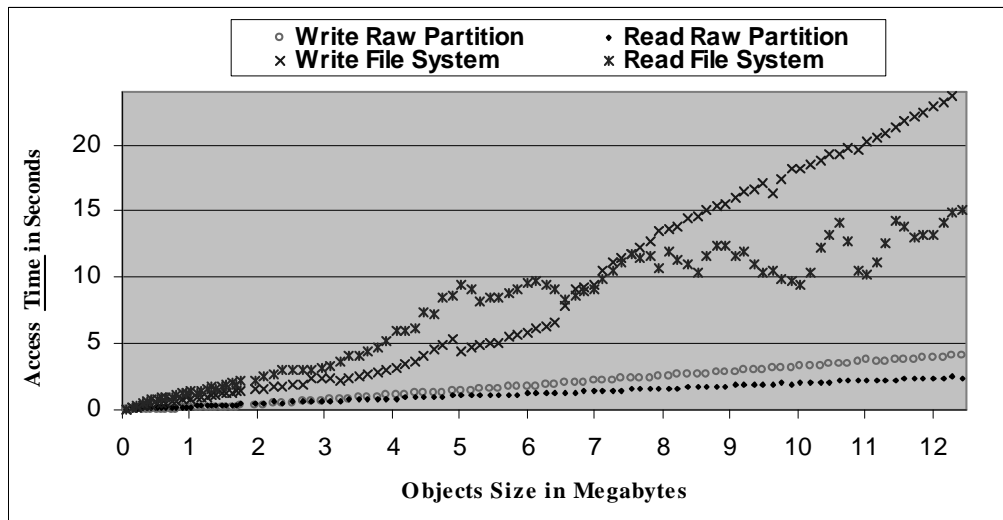


Figure 5.14: Database performance when storing/retrieving large objects

For the first case, in which the database uses regular file system for the data storage, the average data access time for the write operation reaches 664 KB per second and for the read operation reaches 784 KB per second.

For the second case, in which the database uses the raw disk partition for the data storage, the average data access time for the write operation reaches 3.35 MB per second and the read operation reaches 4.92 MB per second.

5.6.3 Observations

From the benchmarking tests performed, according to the chosen configuration parameters for the test case described in previous section, we conclude the following observations:

Observ 1: On the average, for the case of using regular file system and for objects that are smaller than 3 megabytes, the performance of Matisse database system is almost the same for both read and write operations. However, for larger objects, the difference between read and write operations is rapidly increasing.

Observ 2: For the case of using the raw disk partition, both read and write operations almost take the same amount of time for objects up to 6 megabytes. For larger objects, the difference between the read and write operations has a minor increase.

Observ 3: On the average, the Matisse database system performance when storing/retrieving large objects improves up to 6 times with the use of the raw disk partition.

Observ 4: The database access time is more regular when using raw disk partition than in the case of using normal file system, which is probably influenced by other I/O performed by the operating system.

5.6.4 Lessons Learned

To better support the complex requirements of scientific applications in VL, and to design the most efficient physical implementation for the database, some data storage/retrieval benchmarking tests are necessary to be performed. The test cases must be carefully designed to help with the evaluation of the approaches for physical database design, while considering the performance of the DBMS itself for managing large scientific multi media data sets. Besides the tests performed above, additional tests may become necessary for instance to consider the hardware configuration in terms of (1) using several raw disk partitions, (2) increasing the database cache size, and (3) using powerful machines, mainly dedicated to run the database server for large applications.

The knowledge gained from benchmarking plays an important role in defining the physical database design and specific data storage strategy for each application, based on the data input/output requirements. For instance, considering the performance results achieved above, a small to medium size application that does not require more than 5 MB per second for its input/output, can follow an archiving strategy where the large data sets are stored together with their meta-data in the same Matisse database. However, more complex applications that may require better performance or proper storage of larger data sets can benefit from a different strategy, where the database only holds a link to the large objects, that can in fact reside at the place where they are generated or heavily used.

5.7 Conclusion and Discussion

This chapter addressed concepts for building a robust application to efficiently manipulate large and complex scientific data sets in the VL environment and introduced flexible interfaces for data access in the VL. The addressed concepts, however, are not only specific to

the VL, rather, most concepts presented here also facilitate, for instance, the creation of general digital libraries for scientific applications and can support the manipulation of their large multimedia data sets.

In different sections of the chapter, we illustrated possible strategies for storing scientific data, and outlined the major benefits gained when considering standards during the modeling and implementation phases of advanced and complex applications. Such concepts can be applied to different applications and serve for improving their development life cycle from information classification strategies to modeling the constructs, and from the design of information access mechanisms to the analysis of performance criteria.

The concepts presented in this chapter extend and complement the work presented in previous chapters. The extensions addressed in this chapter focuses on the following:

- The combination of database standards and Web technologies, to facilitate the process of developing information integration mechanisms among networked applications.
- The use of universal data access mechanisms to provide common interfaces for accessing heterogeneous databases through a common set of code.
- Standard information modeling for scientific applications to provide a common understanding about the data and meta-data dealt with, in scientific applications.
- The considerations of scientific multimedia information, large data sets, and complex inter-linked objects.
- Provision of mechanisms and tools for scientific data publishing, based on tailored and restricted access on sharable data, while preserving systems' autonomy; and hiding private data from outside users.

5.7.1 Contribution to **GFI₂S**

The concepts introduced in this chapter addressed a number of important issues that are applied at every site to better enable it as a node within the cooperation network. The data storage strategies and the performance issues assure the database efficiency at the local sites. While, the definition of restricted schemas and the development of universal and schema free tools to access them, allow sites to share part of their data. The contribution of the approaches described in this chapter to the **GFI₂S** integration approach presented in Chapter 6 is two-fold:

- On one hand, since the **GFI₂S** integration architecture preserves autonomy for individual sites, those sites can benefit from the use of the important concepts developed for VL information management, in order to efficiently build their applications independently of other sites.
- On the other hand, the use of standards at individual sites as adopted for VL, especially for the information modeling and universal data access, helped the development of information integration mechanisms for **GFI₂S**, which was necessary to support collaboration among heterogeneous and autonomous sites.

Chapter 6

GFI₂S - Generic and Flexible Information Integration System

6.1 Introduction

Nowadays, for information management in complex organizations, a large variety of different database management systems (DBMSs) are used, which are at best chosen to meet the specific characteristics and requirements of every application environment. Existing application environments differ in their main characteristics and features. On one hand, they differ in their distributed/centralized architecture, their size, complexity, and the type of data they handle. On the other hand, their requirements depend on the global functionalities that they need to provide and on the required level of integration.

Following are brief descriptions of the main characteristics, which have a direct impact on the level of complexity of applications in terms of information management. These characteristics need to be considered when designing and developing information integration mechanisms for advanced application domains:

- ☞ **Type of Application:** due to their wide diversity of interest, emerging applications constitute a wide variety. Certain applications are specific to tasks that are internal to organizations, while some others are more dedicated to web environment and e-commerce. A third type focuses on building secure collaborative environments among a trusted community of users, and so on.
- ☞ **Size and Complexity of Applications:** existing organizations are of different complexities and may consist of a wide range in sizes. For example, organizations may be as small and simple as a local or centralized application where all modules can run on a single unit, or as large and complex as a heterogeneous application with many geographically distributed processing units.
- ☞ **Type of Data and Data Integration:** some applications are developed using simple data structures and may perform little information exchange with others. Other applications however, handle inter-linked complex objects and require the integration of large data from external sources.
- ☞ **Level of Integration:** the required global services among heterogeneous and autonomous sites identifies the level of integration to be provided, while supporting the

heterogeneity of the underlying database components, and preserving their autonomy.

The scope and characteristics of application environments as described above, and their required level of interaction and integration with other applications, play the major roles in designing suitable database architectures and database management systems for them. In particular, for every application environment, the underlying DBMS must be carefully chosen to meet its specific requirements.

At the logical level, all existing DBMSs provide certain basic functionalities for application modeling, data structures, and the information storage and retrieval. However, at the physical level (the implementation level), DBMSs differ on the architecture that they rely on, the constructs they offer, the data storage mechanisms they use, and the information retrieval functions they provide. Therefore, existing DBMSs lack the possibility to be efficiently used for all types of applications. Some DBMSs better suit smaller applications, while others are more dedicated to complex environments and focus on the management of, for example, multimedia information and large data sets. Thus, *any attempt in the direction of forcing different applications to use the same database system for the management of all their information services is unrealistic*. Even within the same environment, in certain complex applications, the use of more than one DBMS cannot be avoided.

From the application cases, described in Chapters 3, 4, and 5, it is clear that a main criterion required by most advanced applications, is the provision of collaboration possibilities and data sharing/exchange among distributed, heterogeneous, and autonomous organizations. We also learned from the application cases that providing interoperability and integration among sites, via the deployment of database standards and emerging Internet technologies, is one of the most challenging approaches in the area of integrating heterogeneous information from autonomous sites. The interoperation/integration process eases the collaboration among existing systems, while preserving their autonomy and privacy in manipulating their own data independently of other database systems.

Research in the area of integrated information management systems started in the early 1980's. However, the integration focus since then has been stepwise, from supporting homogeneous centralized systems, to heterogeneous distributed systems, and finally towards the federated solutions. A number of these solutions are described in details in Chapter 2, where a comprehensive classification architecture for these approaches is also provided and illustrated. Considering the subject addressed by this chapter, we briefly summarize the main categories addressing the integration, as follow:

- ***Tightly coupled homogeneous systems*** denoting logically centralized, but physically distributed, databases systems [CP 84, SBD⁺83, AVF⁺92]. This approach lacks the support for data representation heterogeneity and does not preserve the autonomy of individual database systems participating in the collaboration network.
- ***Tightly coupled heterogeneous database systems*** where database schemas are integrated in one centralized global schema on top of which, database views can be defined [TBD⁺87]. This approach proves inefficient in terms of flexible information sharing and in preserving full autonomy of the local database systems.
- ***Loosely Coupled federated architecture*** proposed in [HM 85, LA 86], involves the integration of several database systems, stressing the autonomy and flexible sharing of information in a loosely coupled architecture. Examples of loosely coupled federated systems, providing partial autonomy to the involved databases in the network, are MRSDM [Lit 85], OMNIBASE [REM⁺89], and Calida [JPS⁺88]. To this architecture, Kim et al. [KCG⁺93] provides suitable techniques for resolving representational

conflicts within the context of multidatabase schema integration. Another example of federated information management approaches is proposed by the PEER federated system [AWH 94, ATW⁺94]. PEER follows the pure federation supporting information heterogeneity and preserving the full autonomy of nodes in the federation.

This chapter focuses on the design of a Generic and Flexible Information Integration System (**GFI₂S**). *Flexibility* in **GFI₂S** resides in its ability to add/remove new system to/from the federation with involvement of minimum effort. Flexibility of **GFI₂S** is supported through the use of the specific two-component architecture, while, its *Genericness* is achieved through the incorporation of database standards, emerging Internet technologies, and middleware solutions.

The design of **GFI₂S** is based on the investigation, evaluation, and validation of the methodologies and systems discussed within Chapter 2; and motivated by the expertise gained within the development of the various R&D projects addressed in Chapters 3, 4, and 5 of the dissertation. Thereby, the architecture of **GFI₂S** is inspired on, and extends the architectural design and development for these systems. A number of these aspects and design considerations are described in previous chapters, hereafter, we briefly summarize their main contribution to **GFI₂S** as follow:

- The Waternet system contributes to the design of **GFI₂S** at (1) the Node Federation Layer by tackling the fundamental schema management challenges, and at (2) the Local Adaptation Layer by serving the system openness through the adaptation and extension of the data adapter components.
- The MegaStore system contributes to the design of **GFI₂S** through (1) the deployment of database standards and Internet Middleware supporting system reusability, (2) the development of a parallel/distributed database server assuring system efficiency, and (3) the development of user friendly interfaces assisting advanced/ordinary users in accessing the underlying information sources.
- The VL information management framework contributes to the design of **GFI₂S** by addressing a number of important issues that are applied at every site to better enable it as a node within the federation. On one hand the data storage strategies and the performance issues assure the database efficiency at the local sites. On the other hand, the definition of restricted schemas and the development of universal and schema free tools to access them, allow sites to properly share part of their information.

6.1.1 Focus of **GFI₂S**

The architectural components of Generic and Flexible Information Integration System (**GFI₂S**) supports the integration of different types of data from heterogeneous applications, in order to achieve broader information access capability, and minimize development efforts. The architecture of **GFI₂S** is composed of two main components: (1) Local Adaptation Layer (LAL) that facilitates the access to the underlying databases in the node; and (2) Node Federation Layer (NFL) that provides links to the information and applications outside the node and supports the information sharing and interoperation. This two-component architecture of **GFI₂S** supports a wide variety of existing applications with efficient means for their interconnection and interoperation, while preserving their *heterogeneity*, *distribution*, and full *autonomy*:

- **Heterogeneity** refers to the fact that each database may apply its own distinct DBMS, and data representation is heterogeneous in terms of structures and semantics.
- **Distribution** refers to the storage and processing of information from distributed data sources, located on different host computers.
- **Autonomy** refers to the fact that each site within the federation community is an independent system. Typically, a local site is pre-existing to the creation of a cooperation network and has its own administration policies, and users groups.

The distinctive features of the **GFI₂S** integration approach resides in: (a) the *specific combination* of database standards and Internet middleware with the fundamental research approaches, and (b) the *way in which they are deployed and interlinked* within the two specific components of the **GFI₂S** architecture. These two considerations make the **GFI₂S** approach distinct from all other existing federated/integrated approaches, and introduce **GFI₂S** as a *generic solution* providing a *flexible architecture*, and an *open facility* for integration/interoperation among heterogeneous, distributed, and autonomous sites. Following are clarifications related to genericness, flexibility, and openness of **GFI₂S**:

1. The use of object-oriented standards and middleware solutions, in the development of the **GFI₂S** system, makes its *architecture generic*. Each site that wishes to join the federation only needs the knowledge about its “underlying database system” and about the “common format” adopted at the federation layer. The local users at each site gain proper expertise about the underlying local database characterization and specifications. At the same time, the common format adopted at the federation layer is widely understood by these users.
2. The use of a two-component architecture within **GFI₂S** makes the system *flexible*. This flexibility is supported from two sides, on one hand, sites can easily join or quit the federation, on the other hand, the schema integration strategy followed at the node federation layer allows for a customized integration, which can be tailored to the need of each site. Within each site, the Local Adaptation Layer (LAL) facilitates the access to the underlying databases in the node, in their native format, while the Node Federation Layer (NFL) provides links to the information and applications outside the nodes, to support the information sharing and interoperation.
3. The bounding of database standard mechanisms (e.g. ODMG standard) with emerging advanced tools in the domain of information management (e.g. Java, JDBC, and XML) makes **GFI₂S** an *open facility* for sites integration. The deployment of advanced and standard tools facilitates the information integration and the interoperation among multi-platform and multi-language applications.

The main concepts related to the architecture of **GFI₂S** system and the deployment of standard mechanisms and advanced tools are further described within the remaining sections of this chapter. Section 6.2 addresses the information integration approach of **GFI₂S**, Generic and Flexible Information Integration System, and outlines its architecture. The architecture of **GFI₂S** constitutes two components of Local Adaptation Layer (LAL) and Node Federation Layer (NFL). Further details of LAL and NFL layers of **GFI₂S** are presented in sections 6.2.1 and 6.2.2. Within these two sections, detailed descriptions are also provided regarding the various components and concepts constituting both the local adaptation layer and the node federation layer. Among other components, section 6.2.2 also describes federated schema management, federated resources specification, federated query processing,

and the various schema definition and schema derivation primitives. Section 6.2.3 discusses the deployment of database standards and middleware solutions within **GFI₂S**, to facilitate the collaboration among a large number of organizations. Section 6.2.4 describes an example operational case that shows the quality of **GFI₂S** approach and illustrates the main benefits gained when deploying its architecture. Finally, section 6.3 concludes the chapter and emphasizes the major achievements of **GFI₂S** in terms of information integration and systems interoperation.

6.2 GFI₂S Information Integration Approach

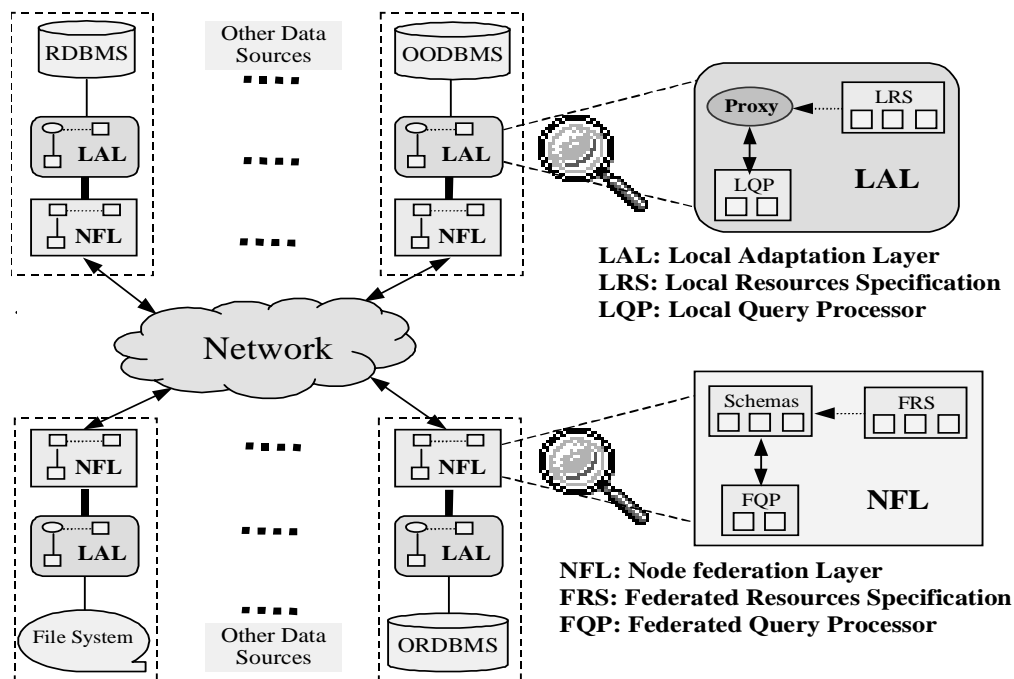
This chapter addresses the design of a flexible approach for information exchange and interoperation among heterogeneous applications. The proposed approach follows the node-to-node federation described in section 2.2.2.2. Additionally, this approach extends the existing solutions via the deployment of standard solutions, middleware, and the emerging Internet and database technologies. Therefore, the information integration in **GFI₂S** is addressed from two perspectives:

1. There will be a design and partial development of a generic and extensible integrated information management approach referred to as Generic and Flexible Information Integration System (**GFI₂S**), and
2. It will involve the definition and extension of database standards and middleware solutions serving the interoperability requirements, and providing support for openness and flexibility to the **GFI₂S** approach.

For federated schema management, the architecture of **GFI₂S** has its roots in the PEER federated/distributed database system [TA 93, ATW⁺94], which was previously developed within the CO-IM group of the University of Amsterdam. Moreover, the designed solution takes advantages from other related research and approaches. Among the applied features we enumerate the use of database standards [BFN 94, PWD⁺99], object-oriented technologies [CBB⁺00], mediated systems [DD 99], wrappers [THH 99], Grid distributed technologies [ABB⁺01, BHG⁺01], and semantics resolution [KCG⁺93, SP 94, HM 99]. In the **GFI₂S** system, the nodes store and manage their data independently, while in the case of common interest they can work together to form a cooperation network at various integration levels. This nesting of nodes at different integration levels allows for a variety of configurations, where, for instance, certain kinds of cooperation are formed to enhance the performance of data sharing, while others are formed to enhance the complex data sharing in a distributed or federated manner.

Figure 6.1 shows the high-level architecture of the **GFI₂S** integration approach among several sites. The two main components participating in the design and development of the integration layer at each site within the federation community are the *Local Adaptation Layer* (called LAL) and the *Node Federation Layer* (called NFL). The local adaptation layer defines the set of concepts and mechanisms that: (1) facilitate the access to the underlying local data sources in their native format, and (2) control their access rights. The node federation layer acts as a mediated common interface that sits between the local system and other external data sources participating in the federation.

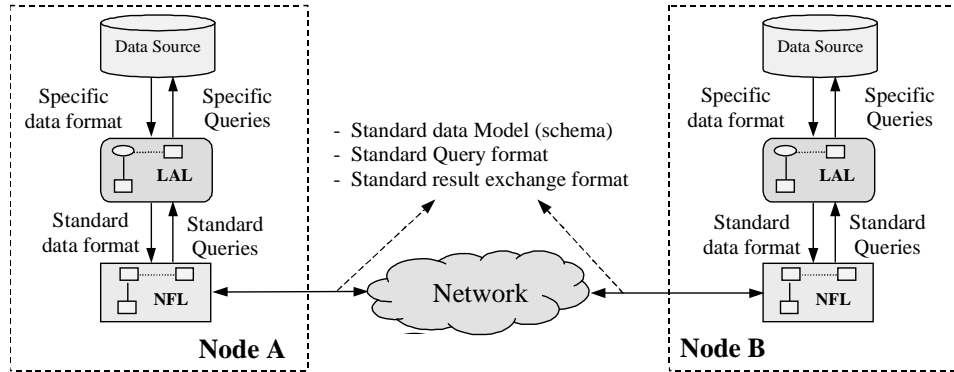
Previous work in the area of federated database systems has involved a considerable effort in integrating new systems to the federation community, where each federated system is considered to interface with all other native systems in the cooperation. Our approach

Figure 6.1: The GF₂S Architecture following the Node-to-Node Federation

considers the use of standards at the federated layer and it is preferred that each site that wishes to join the federation defines and specifies the mapping to and from the federated layer to its internal data source. Thereby, all data source components in the federation only communicate using the standard languages and common data format, adopted at the federated layer, which are widely understood by a large community of users in the field.

Figure 6.2 illustrates the communication mechanism among two sites using **GF₂S** architecture, where the LAL overcomes the specificities of the underlying data source in terms of query languages and data format, and the NFL overcomes the data models specificities among several nodes within the federation community. More precisely:

- The LAL handles the communication between the standard data and query format and the specific data model and query language deployed at the underlying data source. It consists of a set of specifications and tools supporting the communication to the underlying data source using a common data model and a common query language.
- The NFL at each site: (1) defines the part of the information to be imported/exported from/to other sites within the federation community, (2) defines the mapping between import, export, and integrated schemas, and (3) specifies the circumstances under which external users can access exported information.

Figure 6.2: Communication Model among GF₂S components

The **GF₂S** architecture supports several features, which ease the integration process among autonomous and heterogeneous sites. These features are mostly achieved through specific design considerations:

1. The *global federation* is build based on the federated resources¹ defined at the nodes federation layers (NFLs). The nodes federation layers are based on the *use of standards* for data modeling, for information sharing, and for query languages (e.g. ODL, XML, OQL, and SQL). These standard mechanisms are widely adopted in most applications, thus, facilitating the interoperation/integration process.
2. The *Heterogeneity* at every node is supported at the two levels of:
 - Database management systems, where different data modeling approaches are considered, ranging from relational to object-oriented and object-relational, and from file systems to legacy database systems.
 - Data representation, in which the same data may be defined differently (different schema definitions), in terms of its semantics and structures.
3. The proposed architecture makes it possible to add a new application to the federation, just as easy as to take one away. To join the federation, each application only needs knowledge about its underlying database system and about the common format adopted at the federated layer.
4. The *user assistance* in remote data sources integration and data access, is achieved via flexible interactions with the common data model at the federated layer, and through common query languages. The Universal Database Access and the Safe/reliable Data Export interfaces, described in Chapter 5, illustrate two examples of tools assisting **GF₂S** users in integrating heterogeneous information sources and defining information visibility levels on them.
5. The *use of standard mechanisms* and *middleware solutions* for applications modeling, data structuring, and information retrieval ease the interoperation/collaboration process among a wide variety of preexisting applications. The database schema evolution and the multi-media data handling are supported via the deployment of object-oriented

¹The term resources in this context refers to information sources.

standards, while the emerging Internet technologies and middleware solutions allow universal access to the data, ease the information exchange mechanisms, and facilitate multi-platform applications development.

6. The access to the local data sources is supported via the specification of mappings between the local data sources and the common data model at the NFL, where the LAL, at each node, supports the *controlled transfer* of data to the NFL, while *preserving full local autonomy* of that node. For the development of mappings of process also considers the use of object-oriented approaches that combine both elements of object-oriented research and heterogeneous distributed DBMS research (e.g. applying object-oriented concepts to heterogeneous and distributed database environment is considered).
7. To enforce the database integration process and facilitate the federation of a wide variety of heterogeneous information originating from distributed sources, the **GFI₂S** approach also defines the concepts of *Semantics description* and *Dictionary of Terms*. These two concepts, eventually provided by each participating database, help the conflict resolution about the meaning and similarity of objects in terms of their naming and structural representation. The importance of these components resides in providing a clear definition for all the objects exported by a database system, and facilitates their integration by other systems.

The remaining sections describe in details the design of **GFI₂S** and outline the features and considerations enumerated above, which are covered by the two components of **GFI₂S**, namely, the Local Adaptation Layer and the Node Federation Layer.

6.2.1 Local Adaptation Layer (LAL)

In order to make the information sharing among heterogeneous and autonomous sites a reality, local sites must develop a set of mechanisms to facilitate access to their local data, to define the concept models, and to specify the circumstances under which external users/applications can gain access to a part of their information.

A good strategy for information integration must clearly distinguish between the tasks to be performed locally (specific individual tasks) and the common tasks that involve the contribution of several networked applications (common tasks of the federation). The design strategy we have followed considers that users at the federated layer only operate on common data models using standard mapping tools and querying languages, without caring about the specific operations for query translation and results transformation, which are left to each local site. Thus, it is preferable to keep the local mapping rules and derivation primitives, related to the query and to the result translations, local to each layer. For security and efficiency reasons, these specifications can be better supported as built-in components within the local adaptation layer. On one hand, performing these mapping locally at each site prohibits external users from knowing or needing to know the underlying data structures of the local application. On the other hand, the local execution of the queries and the transfer of the resulting data in a condensed format improve the interaction process and reduce the communication time between interoperable database systems.

The main goal of the *Local Adaptation Layer*, created at each local node, is two-fold: (1) to translate external queries into queries for the local querying language, and (2) to transform the local results of the queries into a format that is readily understood by other applications within the integrated system. Thus, the local adaptation layer at each node

provides a direct link between the local data source and the integrated system, forms the local database gateway, and implements the interfaces to the federated layer.

Figure 6.3 shows the main components of the local adaptation layer. Mainly: (1) the *Local Interoperation Agent* that forms the main gateway to outside for the local node; (2) the *local resources specification* that defines the access rights, the query mapping rules, and the results transformation mechanisms; and (3) the *local query processor* that interfaces to the local data source for query translation and execution. The dashed arrows from the local resources specification to the Local Interoperation Agent and to the local query processor, means that the operation processes of these latter components in terms of data access rights and query translation are performed and controlled based on the local resources specification. The next sub-sections further describe in more details the components of LAL.

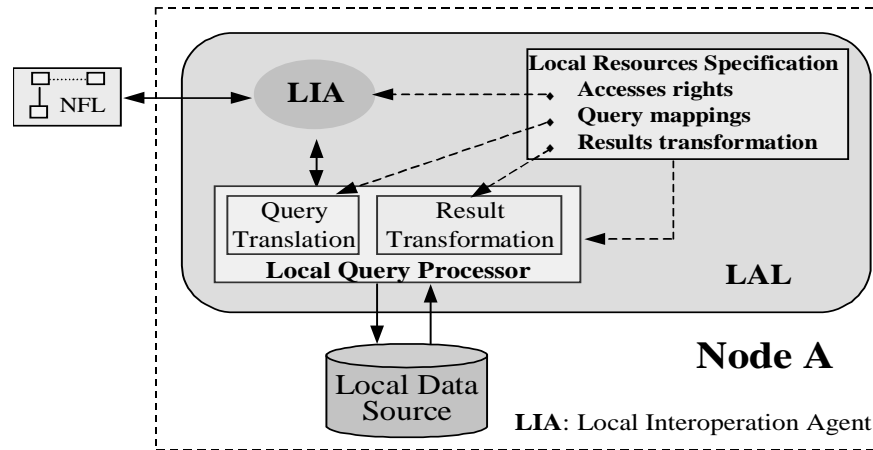


Figure 6.3: Components of the Local Adaptation Layer

6.2.1.1 Local Interoperation Agent (LIA)

The *Local Interoperation Agent* component acts as the main gateway to outside for the local node. Its main functionality include: checking and controlling the access rights to data, validating the query commands against the local resources specification, and coordinating the local query execution and result transformation. To properly accomplish the main tasks assigned to it, the Local Interoperation Agent operates according to the local resources specification defined at the Local Adaptation Layer of the corresponding node.

6.2.1.2 Local Resources Specification (LRS)

The *Local Resources Specifications* (LRS) help the Local Interoperation Agent and the local query processor with performing their tasks and controlling the access to the sharable data at the local node. In this repository, the local resources are specified in terms of the following features:

- *Access rights and user visibility levels*, which define the set of users that can access the data, and specifies the part of information to be shared with them.

- **Query mappings**, which define the set of rules for query mapping that help the local query processor (LQP) in translating the arriving queries from the common format into the local format used at each local database system.
- **Results transformation**, which define the set of rules for data translation that help the local query processor (LQP) in transforming the results of a local query into the common format. The common format is based on the use of standards, which are widely understood by the networked applications.

More details describing the resources specification and outlining the operational relationship between local and federated resources specification are given in section 6.2.2.2. The latter section outlines the schema definitions, mapping derivations, access rights definition, and semantics description related to the sharable information among distributed sites.

6.2.1.3 Local Query Processor

The *Local Query Processor* is a software component that offers a uniform querying interface, to the node federation layer. This querying facility applies to the schema defined on the underlying local data source. The role of the local query processor is to pose queries to the local data source and extract an answer resulting from that data source. The extracted result must be reorganized into the common format that can be understood and manipulated at the federated layer.

The local query processor, at each site's local adaptation layer, is usually created with two distinct parts, which consist of the *query translation* and the *result transformation*. The query translation component transforms a query in the common format into a query in the local querying language in order to be executed locally. The result transformation process transforms the local data resulting from the local query execution into a format that is readily understood at the integration layer.

Local Query Translation & Execution: At the local adaptation layer, before executing the submitted query, first the query must be translated from the common format into the local format of the underlying data source, and second the transformed query will be executed against the local data source. We assume that the federated query processor, as will be presented in details in section 6.2.2.4, only performs the decomposition of the federated query into a set of sub-queries that are still in the common format. The task of translating the sub-queries from the common format to each local format of the database systems participating in the federation is left to the local adaptation layer of each node. If we consider the integration approach introduced in Figure 6.1, the node making the integration does not need to know the specifications of local query languages at participating database, neither it needs to know about the local DBMSs used by these databases. So it is easier and safer to process the relevant sub-queries translation at each local node than having the database integrator transforming all the queries from the common format to the various local formats.

Local Result Transformation: The local result transformation process translates the result of a sub-query into the common format defined at the federated layer. Similarly to the query translation process, since the integration layer follows a common model for data representation, the results of the local sub-queries are locally transformed at each node and sent back to the user/application that has requested it. The *Result Assembler* at the federated layer, as will be presented in details in section 6.2.2.4, will

merge this data and organize it to fit the federated schema specified by the requesting user/application.

Below, we summarize the steps involved within the execution process of a given query at the Local Adaptation Layer (LAL), which include:

1. A query is submitted to the local application through its LAL. The query is expressed using standard languages (e.g. OQL).
2. The Local Query Processor (LQP) translates the submitted query in order to be conforming to the local native schema definition and local query language. The query is then sent to the local source where the real data is stored.
3. Through the result assembler of LQP, the returned results (expressed according to the local format) are transformed and reorganized to fit the export schema defined at the node federation layer.
4. The global results, expressed using the standard format (e.g. XML, CDM), are returned to the user.

6.2.2 Node Federation Layer (NFL)

In recent years, database research is increasingly focused on the area of systems interoperation and information integration, where the research is approached differently through different methodologies such as the multidatabase systems (MDBS) [LMR 90], federated database systems [SL 90], or mediated systems [FLM 98]. There are however, a large number of challenges still need to be addressed and solutions to be found in this area. Therefore, further and deeper investigations are required in this field.

The *Node Federation Layer* of the integrated system, addressed in this section, describes the integration mechanism of **GFI₂S**. The global architecture of the node federation layer, presented in Figure 6.4, is composed of (1) a *Federated Schema Management* defined using the Object Modeling Language (ODL), (2) a set of *Federated Resources Specification* for the information sharing, and (3) a *Federated Query processor* based on standard query languages. The modeling language within the **GFI₂S** system allows the definitions of the set of entities, their attribute names, and relationships that are used to build the schema components. While, the common query language is the mean to formulate queries against those schemas.

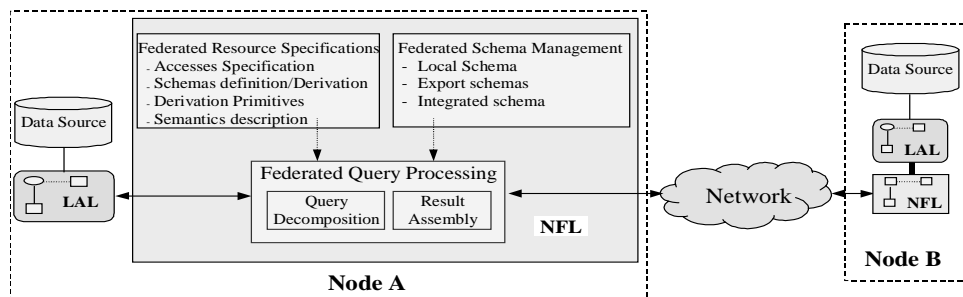


Figure 6.4: Node Federated Layer Representation

In addition to supporting heterogeneity of the database components and preserving their autonomy, the NFL layer at each node allows every site within the federation to:

- Define as many export schemas as required; each export schema defines a part of the sharable information for certain external users, and is tailored to the specific need of those users.
- Only import the required parts of information from the schemas exported by other sites; related information is imported in coordination with the node federation layer of the exporting site.
- Design and build its own integrated schema; where classes and their attributes in the integrated schema represent the global overview of all the data that can be accessed by the site. The integrated schema represents the local information merged with all imported information.

The node federation layer is responsible for maintaining the local, export, and integrated schemas consistency, decomposing global queries into sub-queries executable by the local database at different sites, coordinating the execution of the sub-queries, and reorganizing the returned sub-query results to fit the global view of the integrated schema. These tasks are accomplished at the node federation layer based on the set of federated resources, which define the necessary specifications related to the sharable information, its derivation operations, and the circumstances under which external user can gain access to that information.

- ***From the usage point of view***, the NFL at each node provides the means to create a seamless federated database, hiding details such as the database location, data representation, and heterogeneity from the users and application programs. Moreover, the integrated schema within the node federation layer is customized to the need of the users and to the requirements of their applications.
- ***From the implementation point of view***, the aim of the NFL is to remove the need for static global schema integration and allow each application to have more control over the sharable information; control, is therefore decentralized. The node federation layer brings an open component-based architecture to build an integrated system with advanced database integration features, which makes it a favorable solution located in between the two extremes of no integration and total integration.

The Object Database Standard (ODMG 3.0) [CBB⁺00] is chosen for specifying the modeling constructs and the querying language at the federation layer. In this direction, the object definition language (ODL) is used for describing the data structure and constraints, while the Object Query Language (OQL) is used for operating and retrieving the sharable information.

Besides following the standards in defining the database schema components, the ODMG model provides the following advantages:

- It is simple enough so that it can be readily understood and implemented.
- It is semantically expressive to capture the intended meaning of conceptual schemas that may reflect several kind of heterogeneity.
- It contains the basic features common to most semantic, hierarchical, relational, and object oriented models. It supports modeling primitives, type membership, object properties, and object behavior.

- It includes the ability to encapsulate the functionality of shared objects, its extensible nature, and object uniformity.
- It is an open standard for new extensions regarding languages programming, Java binding, and interfacing with other emerging standards and technologies.

The use of a common data model, that is object-oriented, does not rule out the participation of other relational, hierarchical, or file system based models. Rather, it assures their full integration via the support of their semantics heterogeneity through the expressive object data model adopted at the federated layer.

The remaining of this section will address in more details different components that constitute the federated layer at each node participating in the federation community. Namely, the following sub-sections describe the steps required for defining:

1. The ***Federated Schema Management***, which provides the necessary mechanisms for the definition and derivation of the local, export, and integrated schemas (described in section 6.2.2.1),
2. The ***Federated Resources Specification (FRS)***, which defines the set of concepts related to the sharable information, to their derivation, and to their access right policies (described in section 6.2.2.2),
3. The ***Federated Derivation Primitives***, which specifies the mapping rules for schemas derivation and query decomposition (described in section 6.2.2.3), and
4. The ***Federated Query Processor (FQP)***, which processes queries that may require extracting and merging data from multiple sources (described in section 6.2.2.4).

6.2.2.1 Federated Schema Management

The federated schema management is based on the integration of information from several remote and local data sources. As depicted in Figure 6.5, Every site participating in the integrated system is represented at its node federation layer by several schemas:

- The ***Local Schema***, which models the data stored locally. This schema represents the part of data owned by the node.
- The ***Export Schemas***, which model the part of information that this node wishes to make accessible to other nodes/users.
- The ***Imported Information***, which represents the part of information that this node wishes to import from other exported schemas participating in the federation.
- The ***Integrated Schema***, which merges local and imported information into a coherent view that, satisfies the node's requirements in term of information management.

The schemas representation adopted for the node federation layer, illustrated in Figure 6.5, guarantees the following features:

- ***Autonomy is preserved***: each node has full control on its local data, decides on the part of data to be shared with external nodes, and only imports the part of information needed from the various schemas exported by other nodes.

- *As many export schemas as needed*: each node individually decides on the part of data that will be shared with the external nodes based on other node's privileges and rights.
- Each node decides on the *layout and the format of the data* that they want to export and make available to the outside world.
- The information managed by the integrated system is *maintained* and *stored* at *several local and remote heterogeneous data sources* where it is generated or it belongs.

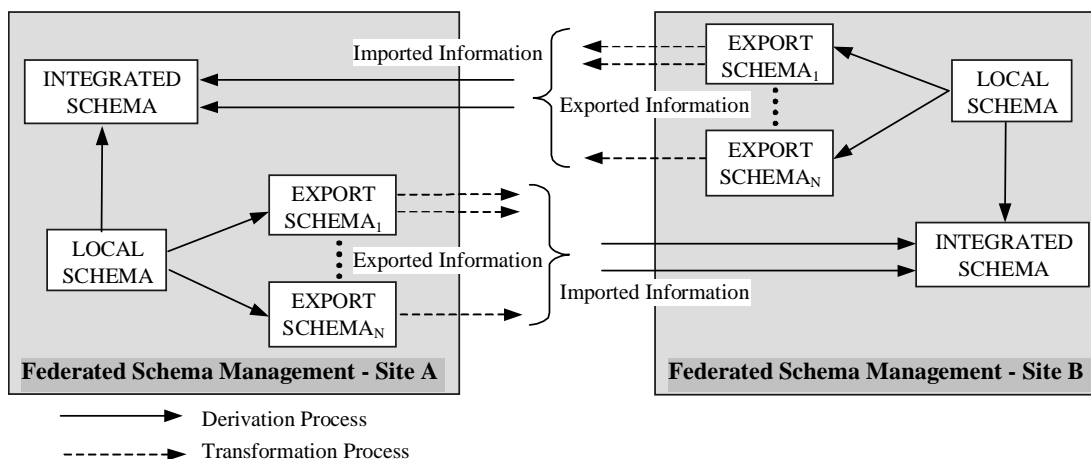


Figure 6.5: Schemas representation adopted at the node federation layer

The following sections describe in more details the components of the Federated Schema Management that support the definition of local, export, and integrated schemas; while their corresponding resources specification, semantics description, and derivation primitives are further described in sections 6.2.2.2 and 6.2.2.3.

Local Schema

The *local uniform schema* at the NFL reflects the data structure of the underlying local data source, and it is defined using a *common-format* adopted at the level of the federation layer to represent all the local schemas at the sites. Under normal situations, this schema can be semi-automatically constructed from the native schema of the underlying DBMS, with the help of a local user.

The local uniform schema supports the derivation of a set of export schemas (serving the requirements of other applications) for information sharing. The local uniform schema defines a consistent data model facilitating the manipulation of the shared information.

For the derivation of export schemas, another strategy that can be adopted is to define them directly on the local underlying data source, without passing through the definition of the “local uniform schema”. However, the usage of the local uniform schema as an intermediate step, eases the definition and further expansion of the export schemas at different phases of the integration process, as well as making the federation more flexible when new information sharing policies are required.

The native schemas are then transformed using the standard data modeling constructs adopted at the federation layer. Thus, the problems related to the data modeling heterogeneity among applications are solved, while representational heterogeneities are also treated at the node federation layer, for which related concepts to schemas derivation and semantics description are presented in sections 6.2.2.2.

Export Schema

The export schema defines a part of local information that a node desires to share with the outside world. It also specifies the “mapping rules” for translating the data from the local data source to the various export schemas. These mappings are necessary when a node decides to export data in a different format than the local representation. Such a variation is necessary for security reasons and due to some required layout when data needs to be exported. A node for instance may create a single export schema for each user/application or it may define a common export schema that can be exported to different nodes.

A set of resources are defined within each export schema, these resources specify the derivation mappings from the local schema to the export schemas, define their access rights, and provide the corresponding semantics descriptions, which help during the integration process. These resources are further described in section 6.2.2.2.

Integrated Schema

Within each node’s federation layer, the Integrated Schema (INT) represents a global and coherent overview of all accessible local and remote information. Its definition is based on both local and imported information; that are merged in a way that makes the physical/logical distribution of information transparent to the user. The integrated schema can be interpreted as one user’s global classification of objects that are organized differently by the schemas in other data sources.

At the NFL of every node, local and remote information from different sources are modeled into one data model, representing the integrated schema². The constitution of the integrated schema of a node is based on the need of the application and on the part of information that this application is authorized to access and retrieve from other sites. Therefore, the integrated schema represents the structure of the local and imported sharable information. The use of a common data modeling language in defining the integrated schema plays a vital role in resolving some aspects of the data model heterogeneity and eases both the schema integration task and the mapping specifications.

The integrated schema for each system incorporates sub-schemas of the local data sources. The Local Adaptation Layer (LAL), developed at each local data source, provides the link that transforms the local database instances into instances of the integrated schema for the node. Consequently, queries on the integrated schema can access the data that resides in remote data sources of the network.

If the local schema uses a different data modeling constructs and a different query language than the languages used within the federation layer, the local schema is seen as an export schema for which, again the mapping specification and the development of the Local Adaptation Layer become necessary. Thus, the import of information from the local schema to the integrated schema will follow the same process as if importing it from a remote node.

²The schema integration component also considers both schema and database browsing, schema modification and enrichment, and the interactive, incremental construction of integrated schemas for use by different agents within the cooperation community.

The diagram in Figure 6.6 provides an example representing the schema integration and systems interoperation within a collaborative environment among three departments of *Manufacturing*, *Sales*, and *Customer* management. Within this environment, users commands are by default, formulated and evaluated against the integrated schema. However, it is always possible, for instance, for database administrator users to specify and submit queries against a local, or export schema as long as they hold proper access rights.

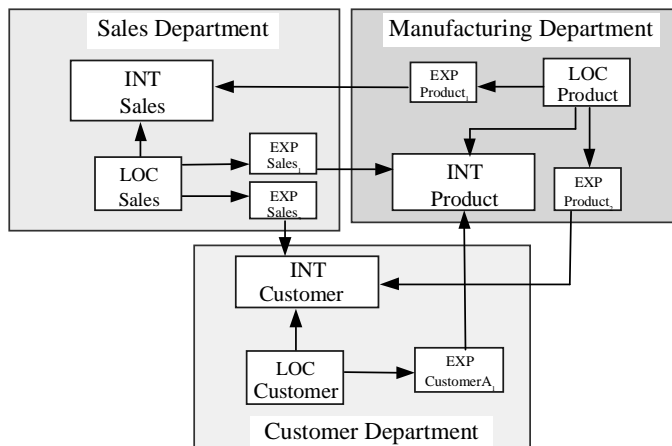


Figure 6.6: Integrated Scenario for Systems Interoperation

The exchange of information between integrated systems within a federation community is done through messages, where a message content is of two types query or data. Queries are always sent from the integrated database system to the local and remote information sources, while data is passed from the remote nodes to the integrated system as an answer to its query. The content of a message, query or data, is expressed using respectively the common query language (e.g. SQL and OQL) and the common data format for information exchange (e.g. XML and OIF).

Within the integrated schema, a different data representation can be used. Similarly to the case of data export in which an application decides on the layout for representing the information to be shared with the outside world, a node at the integrated system can also decide on the way to reorganize or merge the imported data with the local information. Therefore, there is also a need to specify a set of resources that serve the federation process. These resources, referred to as *Federated Resources Specification (FRS)*, define the necessary concepts for deriving the integrated schema and specifying the access rights to it.

6.2.2.2 Federated Resources Specification

In order to facilitate and control the access to their data, each node specifies a set of resources, serving the federation process. These resources define the set of schemas and concepts related to the description of the sharable information and also specify the circumstances under which a node can join and register within the federation community. The main distinction between the federated resources specification and the local resources specification (presented in section 6.2.1.2) is that: the local resources specifications define the mapping rules and the access rights to the local underlying data source, while the federated resources specify the concepts related to the integration through the Node Federation Layer.

Within each node participating in the collaboration community, The federated resources include definitions and specifications of the following five components:

1. **Export schemas definition and derivation**, respectively using the object definition language and the schema derivation language. An export schema definition is identified by an '.odl' extension (<export schema>.odl), while its derivation is identified by a '.drv' extension (<export schema>.drv).
2. **Export schemas registration**, in order to enable other sites to access and share its data, each site must register the set of export schemas together with their access rights specifications.
3. **Integrated schema definition and derivation**, similarly to the export schema definition and derivation, the integrated schema is defined by two components, its definition and its derivation (e.g. <integrated schema>.odl and <integrated schema>.drv).
4. **Access rights definition for export and integrated schema components**, which are defined by the extension '.acs', namely, <export schema>.acs and <integrated schema>.acs.
5. **Semantics description for export and integrated schemas**, which consist of a dictionary of terms and a dictionary of semantics, respectively expressed by <export schema>.dic and <integrated schema>.dic.

Export Schemas Definition and derivation

Export schema definitions constitute subsets of the local schema definition. To define the export schema, we preferably use the ODL object definition language. Figure 6.7 shows a simplified diagram of a basic example for the export schema definition. In this example, a new class *Person* in export schema *EXP2* is derived from the class *Employee* in local schema *Loc*. The structure and the attribute names in *EXP2* are defined differently than in the local schema *Loc*.

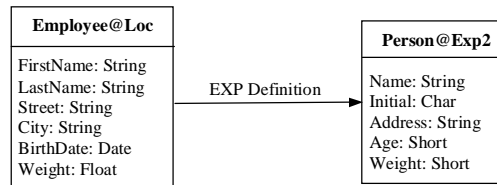


Figure 6.7: Export Schema (Exp2) Definition and Derivation - Example

Figure 6.8 shows example derivation mappings for the class *Person* within the export schema *Exp2*, derived from the local schema *Loc*. The derivation process is based on a set of rules, which define the relationship between the class and attributes in the derived schema (*Exp2*) and those in the base schema (*Loc*). The example below shows the use of three types of derivation: rename, user-defined functions, and type casting.

- The class *Employee* and its attribute *LastName* are renamed in *Exp2* respectively as *Person* and *Name*.
- Attributes *Initial*, *Address*, and *Age* are derived using user-defined functions (respectively *GetInitial*, *Concatenate*, and *DateToAge*). The concept of user-defined function

(that will be later addressed in section 6.2.2.3) is simple and powerful; it allows a user to define his/her specific derivation's functions and algorithms.

- The attribute *Weight* is derived through type casting by converting the *float* value of *Weight* in the base schema into a value of type *short* in the derived schema.

```
// Derivation specification of class Person in export schema Exp2
Person@Exp2 = Employee@Loc
  Person.Name@Exp2 = Person.LastName@Loc
  Person.Initial@Exp2 = GetInitial(Employee.FirstName@Loc)
  Person.Address@Exp2 = Concatenate(Employee.Street@Loc, Employee.City@Loc)
  Person.Age@Exp2 = DateToAge(Employee.BirthDate@Loc)
  Person.Weight@Exp2 = (Short) (Employee.Weight@Loc)
```

Figure 6.8: Export Schema (Exp2) Derivation - an example

The example described above shows the flexibility provided by our approach for users to define the derivation mapping from the base schemas. More details about the different derivation mapping concepts and their possible implementation strategies will be provided in section 6.2.2.3.

Export Schemas Registration

Database administrators at each local site must identify and register the set of export schemas that can be accessed by other sites and external users. The registration process of an export schema identifies its name and its location, along with other information needed for user access.

The following BNF-like notation defines the syntax for export schema registration:

```
Registration ::= <Export Schema> [<description>] <host> <port> <mode>
Export Schema ::= identifier
port ::= number
Mode ::= R | W | R/W
```

The example below shows the registration process defined for two exports schemas Dublin Core (DC) and Traffic System (TS):

DC	Dublin Core	www.science.uva.nl	8800	R/W
TS	Traffic System	146.50.1.188	8900	R

The export schema registration allows the database integrators at other sites to identify the set of export schemas that can participate in the definition of their integrated schemas.

Integrated Schema definition and Derivation

The integrated schema defines the complete set of information that a site can access and retrieve. Therefore, this schema is derived from the local schema and from various schemas imported from other nodes within the federation network. Figure 6.9 for instance, shows a simple example of the integrated schema (INT) definition. In this example, a new class *Organization* in integrated schema *INT* is derived from the existing classes *Department* in schema *Exp7* and *Faculty* in schema *Exp3*. This example involves the definition of new class *Organization*, which contains information from two other classes *Department* and *Faculty* that belong to different schemas *Exp7* and *Exp3*.

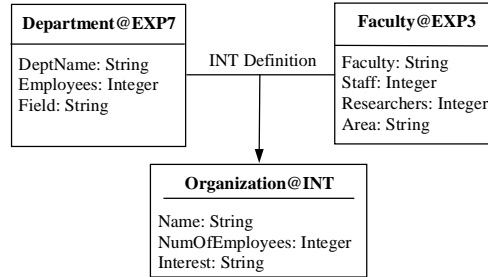


Figure 6.9: Integrated Schema Definition and Derivation - Example

Figure 6.10 shows the derivation mappings of the integrated schema (*INT*), which is based on a set of rules defining the relationship between the classes and attributes in the derived schema (*INT*) and those in the base schemas (*Exp7* and *Exp3*). This example primarily shows the use of the *Union* mapping and a user-defined function (*Sum*); both operations imply the combination of data originating from different schemas. The derivation mapping is reached using an extended language for schema derivation that supports the derivation operations (e.g. rename, union, subtract, and user-defined functions). More information about different derivation mapping languages and primitives will be provided in section 6.2.2.3.

```

//--- Derivation Specification of class Organization in integrated schema INT
Organization@INT = Union (Derive(Department@Exp7,
                                Name=DeptName,
                                NumOfEmployees=Employees,
                                Interest=Field),
                          Derive(Faculty@Exp3,
                                Name=Faculty,
                                NumOfEmployees=Sum*(Staff, Researchers),
                                Interest=Area))
* int Function Sum (int n, int m) return (n + m);
  
```

Figure 6.10: Integrated Schema Derivation – an example

Access Rights Specification for Users and Applications

This process limits the user accesses to the export/integrated schema components and identifies the set of allowable operations submitted on the sharable information. The access to data will be limited to the set of authorized users/applications. This process acts as a semantic integrity checker and access controller sitting between export/integrated schemas and the components accessing them. The access specifications are defined for each export and integrated schema. The following BNF-like notation is adopted for their specifications:

```

Access ::= <user> <password> | <user> <host> <port> <mode>
User   ::= identifier
port   ::= number
Mode   ::= R | W | R/W
  
```

The example below shows the access rights for three different users, defined for a given export or integrated schema:

John	*****	8800	R/W
Toto	www.science.uva.nl	8800	R
Toto	*****	8800	W

Semantics Description

Structural characteristics and assigned names within the database schema do not sufficiently describe their real-world meanings among a number of interoperable systems. In order to support sharing of information among a collection of heterogeneous and autonomous sites, we must overcome the heterogeneity³ of type definitions among these databases. Thus, it is necessary to have an explicit dictionary that clearly defines entities of the database model using a natural language to describe the meaning of names used for types and their attributes. Availability of such a dictionary facilitates the integration of new export schemas from other sites, without the need for on-line support from external users, submitting those schemas. The semantics description also helps in developing *Syntax Match Assistance* tools, which facilitates the automatic/semi-automatic resolution of semantic and representational differences that occur among local and imported related data objects in different systems. Hammer and McLeod [HM 99] distinguish between two aspects in resolving those representational differences: (1) determine the relationships between sharable objects in different components, and (2) detect possible conflicts in their structural representations. The development of such a facility within the federation layer assists and helps in applying methods for resolving representational differences among a number of databases participating in the federation.

Existing approaches for conflict resolution in integrated database systems are based on heuristics [HR 90, NEM⁺86, SLC⁺88], classification [SSG⁺91], semantic proximity concept [SK 93], or fuzzy and incomplete knowledge [FN 92, VH 93], etc. These approaches may lead to inaccurate resolution of the semantic problems. Our approach considers the enforcement of structural representation and semantics resolution by the definition of a *Dictionary of Terms* and a *Dictionary of Semantics* both provided for each database schema at the Node Federation Layer. The *Dictionary of Terms* serves for automatic conflict resolution about the meaning and resemblance of objects in terms of their naming and structural representation, while the *Dictionary of Semantics* helps the schema integration process and to manually solve conflicts that are not automatically detected. The database schema integrator also uses the dictionary of semantics in order to understand the structure of each database schema for the participating sites in the federation.

For the integrated schema, one global dictionary is required per application environment to describe the terms and the semantics (Terms.dic, Semantics.dic). For the export schemas, also one dictionary is required for each export schema (e.g. <Exp>Terms.dic and <Exp>Semantics.dic).

³Heterogeneity in names and data structures is a natural consequence of independent creation and evolution of autonomous databases that are tailored to the specific requirements and characteristics of each application.

Hereafter, are the BNF-Like descriptions of the dictionary of terms and the dictionary of semantics, followed by some examples:

Dictionary of Terms

```

Term          ::= <identifier><relationship><identifier>, []
Identifier    ::= literal
Relationship  ::= X-equal | M-equal | Synonym
X-equal      ::= syntactically equal
M-equal      ::= semantically equal

```

Example of Terms:

```

'Hardware'    M-equal    'Device'
'Device'      Synonym    'Machine'
'Serial number' M-equal  'Code'

```

Dictionary of Semantics

```

Semantic Concept ::= <class name>:[<class description>]
                  [<attribute definition>]
class name       ::= identifier
class description ::= literal expression
attribute definition ::= <attribute name>:<attribute description>
attribute name   ::= identifier
attribute description ::= literal expression

```

Example Semantics:

```

Hardware: this class can be a Device or a machine
         h_name: hardware name
         h_SN: hardware serial number

```

6.2.2.3 Federated Derivation Primitives

Research work in the direction of defining a set of derivation primitives still lack standards and comprehensive necessary concepts for schema derivation and for mapping specification. Radeke [Rad 96] proposes an ODMG extension for federated database systems, in which the schema derivation is supported through the filtering process, based on the DROP command. Within the exported schema the DROP command specifies the attributes, relationships, and methods to be filtered from the local schema. However, the approach does not define a clear export schema for the shared data, rather it gives the list of concepts to be filtered from the local schema. Therefore, the complete elements of the local schemas at the underlying database are not hidden from the user. While, the main aim in collaborative systems is to limit the external access to only the part of information to which users gain access rights. Similarly, Busse et al. [BFN 94] describes an approach to introduce the notion of virtual classes for the evolving object database standard ODMG. However, virtual classes are represented as views, for which the derivation and mapping information is given as a queries that provide full instantiation of the virtual classes. Since the approach assumes a one-to-one correspondence between the integrated instances and due to the complexity in

integrating the relationships, the approach does not properly adapt to distributed databases and support for their different query languages and data models is not fully supported. Recent work presented by Roantree et al. [RKB 01] provides an extended object definition language for view schema integration, based on the standard model for object-oriented databases. It also provides ODL_v language for view specifications and ODL_w language for wrappers specifications. In our approach, we consider the schema integration to be addressed from two perspectives: (1) providing means for export and integrated schemas definitions, and (2) defining the derivation operations that maps the classes and attributes in the base schemas to those in the derived schemas.

Regarding point (1), for export and integrated schemas definition, we use an object definition language specifications to define these schemas. The ODL syntax is used to define the classes of the derived schemas and to specify the relationships between these classes. Thereby, an export schema, for instance, differs from other schemas by only being a subset of an existing local schema, while an integrated schema is a subset of the union of a number of local and export schemas of other sites.

Regarding point (2), related to derivation operations, a strong language for schema derivation must be designed and developed. The schema derivation language must define the derivation mapping for the exported/integrated classes and their attributes. The approach we propose for the derivation operations, distinguish between two types of derivation: class derivation and attribute derivation. These derivation primitives define the mapping between the derived classes/attributes in the derived schema and the basic classes/attributes in the base schemas. The schema derivation language suggested here is based on the PEER Schema Definition and Derivation Language SDDL [AWT⁺94, WA 94], and extends their definitions in order to achieve an open strategy for the schema derivation. Our approach also considers derivations based on the user-defined functions, especially, in the case of the attributes instantiation. For the class derivation the following constructs are supported: *rename*, *union*, *restrict*, and *subtract*. The attribute derivation can use any user-defined function to derive new instances from the base schemas. The *rename* construct is also supported for the attribute derivations.

Two approaches are suggested for the efficient implementation of the class and attribute derivation. In the first approach, the various functions for the class and attribute derivation will be provided in form of shared libraries or dynamic link libraries (DLLs), which will be linked to the node federation layer (NFL) of each site. A second possibility, is to define these functions as persistent stored modules (SQL/PSM) or Call-Level-Interfaces (SQL/CLI) within the SQL environment. Shared libraries will be provided by the developers for interfacing through the programming languages within C, C++, Pascal, Delphi, etc. while, SQL/PSM and SQL/CLI are provided for interfacing through standard database connectivity mechanisms such as ODBC, JDBC, and Embedded-SQL.

The following sections describe in more details the mapping operations/constructs for attribute and class derivation. In the given descriptions: C stands for ‘*class-name@schema-name*’ and A_C represents the set of all attributes in class C . The domain of C is denoted by $Dom(C)$ and the domain of A_C is denoted by $Dom(A_C)$.

$$\mathbf{C} = \{ \text{all legal classes of schema } S \}$$

$$A_C = \{ \text{all legal attributes in class } C \}$$

$$\mathbf{A}_C = \{ \text{all legal attributes of schema } S \}$$

$$\text{Let } I : \mathbf{C} \rightarrow Dom(\mathbf{C})$$

$$I(C) = \{ \text{the set of all instances of class } C \}$$

$I(C) \subseteq Dom(C)$, where

$$Dom(C) = \prod_{a \in A_C} Dom(a)$$

Attribute Derivation

A derived attribute is defined by an attribute derivation expression as follow:

```
derived-attribute-definition := derived-attribute-name = <a-expr>
a-expr := class-name.attribute-name[@schema-name] |
        user-defined-function
```

Attribute derivation is accomplished by specifying an attribute either as a *rename* of another attribute or by a *user-defined function*. The semantics of the primitives defined for the attribute derivation are provided below, where a represents an attribute in class C and $Dom(a)$ represents all the legal values of attribute a .

$$Dom(A_C) = \bigcup_{a \in A_C} Dom(a)$$

The two constructs, defined for the attribute derivation, are defined below.

1- Rename

The attribute *Rename* operation derives a new attribute a from the attribute a_1 , where a and a_1 are two attributes in classes C and C_1 .

$$C.a = C_1.a_1 \tag{6.1}$$

Represents:

$$\forall o_1 \in I(C_1), \exists o \in I(C) : o.a = o_1.a_1$$

Where:

o, o_1 are two objects (instances) of classes C and C_1 respectively, and a, a_1 are attributes of classes C and C_1 respectively.

Example:

```
Organization.Name = Department.DeptName@Exp7
```

The attribute *Name* in the class *Organization* is derived from the attribute *DeptName* of class *Department*.

2- User Defined Function

The *User-defined-function* derives a new attribute a from a set of attributes a_1, a_2, \dots, a_n by applying the function f , thus,

$$a = f(a_1, a_2, \dots, a_n) \tag{6.2}$$

Represents:

$$f : Dom(a_1) \times Dom(a_2) \times \dots \times Dom(a_n) \rightarrow Dom(a)$$

Where a is the derived attribute by applying the function f to the attributes a_i in the base classes C_j , $1 \leq j \leq n$.

Following is a simplified BNF-Like notation for the user-defined function concept:

```

user function    :: <type> <identifier> ([param-list]) <function-body>
<param-list>   :: <type> <identifier-list>,
function-body   :: <statement-list>
statement-list  :: a list of statements in a certain programming language
identifier-list :: a list of identifiers
identifier      :: an identifier in a certain programming language

```

Example:

```

Organization.NumOfEmployees@INT = Sum (Faculty.Staff@Exp3, Faculty.Researchers@Exp3)

```

The attribute *NumOfEmployees* of class *Organization* in the integrated schema (*INT*) is derived as summation of the attributes *Staff* and *Researchers* in export schema (*Exp3*) from the class *Faculty*, where *Sum* is a user-defined function with two parameters of type *Integer*. The **Sum** function can be defined as follow:

```

int Sum (int n1, int n2) {return = n1 + n2;}

```

Class Derivation

In schema derivation, a derived class is constructed from other classes using *Union*, *Restrict*, *Subtract*, and other primitives as described below. A derived class is defined by a class derivation expression as follow:

```

derived-class-definition := derived-class-name = <c-expr>
c-expr := class-name@schema-name |
        union (<c-expr>, <c-list>) |
        Subtract (<c-expr>, <c-expr>) |
        Restrict (<c-expr>, restriction) |
        Derive (<c-expr>, <derived-attribute-definition>)
c-list := <c-expr> | <c-expr>, <c-list>

```

The following constructs provide the semantics for the class derivation.

1- Rename

The class *Rename* operation derives a new class C from another class C_1 .

$$C = C_1 \quad (6.3)$$

Represents:

$I(C) = I(C_1)$, where C is the derived class and C_1 is a class in the base schema S_1 .

Example:

```

Organization@INT = Faculty@Exp3

```

The class *Organization* in the integrated schema *INT* is derived from the class *Faculty* in export schema *Exp3*.

2- Subtract

The class *Subtract* operation derives a new class C by subtracting instances of class C_2 from class C_1 .

$$C = \text{subtract}(C_1, C_2) \quad (6.4)$$

Represents:

$I(C) = I(C_1) \setminus I(C_2)$, where C is the derived class and C_1 and C_2 are classes in base schemas S_1 and S_2 .

Example:

Organization@INT = **subtract** (department@Loc, Faculty@Exp3)

3- Restrict

The class *Restrict* operation derives a new class C from a class C_1 based on certain restriction predicate P .

First, let us define a comparison predicate *Comp* as follow:

$$\begin{aligned} \mathbf{Comp} = \{ & A\theta V \mid \\ & A \text{ is an attribute} \\ & V \in \text{Dom}(A) \\ & \theta \in \{\leq, \geq, =, <>, <, >\} \} \end{aligned}$$

We add the *Not* operator to the *Comp* predicate:

$$\mathbf{Comp} = \mathbf{Comp} \cup \{ \text{Not}(C) \mid C \in \mathbf{Comp} \}$$

Now we define a global predicate:

$$\begin{aligned} \mathbf{P} &= P_1 \phi P_2 \\ P_1, P_2 &\in \mathbf{P} \text{ OR } P_1, P_2 \in \mathbf{Comp} \\ \phi &\in \{ \text{AND}, \text{OR} \} \\ C_1 &\in \mathbf{C}, P \in \mathbf{P} \end{aligned}$$

$$C = \text{restrict}(C_1, P) \quad (6.5)$$

Represents:

$I(C) = \{o \in I(C_1) : P(o)\}$, where C is the derived class and C_1 is a class in a base schema S_1 .

Example:

Organization@INT = **restrict** (department@Exp7, Employees<25)

Following is a BNF-like description of restrictions:

```
restriction :: <predicate>[ <AND|OR|NOT> <predicate>,]
predicate  :: <attribute><operator><range-attribute>
operator   :: = | <> | >= | =< | < | >
```

4- Union

The class *Union* operation derives a new class C from a set of classes C_1, C_2, \dots, C_n .

$$C = \text{Union}(C_1, C_2, \dots, C_n) \quad (6.6)$$

Represents:

$I(C) = (I(C_1) \cup I(C_2) \cup \dots \cup I(C_n))$, where C is the derived class and C_i are classes in a certain other base schemas S_j , $j \in [1..k]$.

Example:

Organization@INT = Union (Department@Exp7, Faculty@Exp3)

The class *Union* operation will be demonstrated in more details in the next example with the *Derive* construct. The *Derive* operation Derives a new class C from the a class C_1 by applying the attribute derivation operations.

5- Derive

The class *Derive* operation derives a new class C from another class C_1 by applying a number of expressions E_i specifying the attributes derivation.

$$C = Derive(C_1, E_1, E_2, \dots, E_k) \quad (6.7)$$

E_i is an expression defining the attributes derivation. It applies the attribute derivation primitives, namely, attribute rename and user-defined function:

- $E_i : a_i = a_j$, where a_i is an attribute of class C and a_j is an attribute of class C_j .
- $E_i : a_i = f_i(a_1, a_2, \dots, a_j)$, where a_i is an attribute of class C and a_1, \dots, a_j are attributes of classes C_1, C_2, \dots, C_j .

$$Attributes(C) = \{range(f_i)\}_{i \in [1..k]}$$

$$I(C) = \{V \in Dom(C) \mid \exists o_1 \in I(C_1) : V = (E_1(o_1), \dots, E_k(o_1))\}$$

or

$$I(C) = \{V \in Dom(C) \mid \exists o_1 \in I(C_1) : V = (a_1 = E_1(o_1), \dots, a_k = E_k(o_1))\}$$

Figure 6.11 illustrates a derivation example of a simple integrated schema *INT*. First, we define three simple schemas (*Exp7*, *Exp3*, and *INT*), using the ODL specification. Second, we specify the derivation operations of the class *Organization* in the integrated schema *INT*. In this example, the derivation is specified in two steps:

1. Specify the derive operations at each of the classes *Department* in *Exp7* and *Faculty* in *Exp3*. The derive operations specify the expressions related to the derivation of the corresponding attributes of class *Organization*.
2. Derive the class *Organization* in the integrated schema *INT* as a *Union* of the derived classes *Department* and *Faculty* from the export schema *Exp7* and *Exp3*.

```

/-- Definition of Schema Exp7
Class Department : Persistent {
  attribute String DeptName;
  attribute Short Employees;
  attribute String Field;
}

/-- Definition of Schema Exp3
Class Faculty : Persistent {
  attribute String Faculty;
  attribute Short Staff;
  attribute Short Researchers;
  attribute String Area;
}

/-- Definition of Schema INT
Class Organization : Persistent {
  attribute String Name;
  attribute Short NumOfEmployees;
  attribute String Interest;
}

/-- Derivation Specification of Schema INT
Organization@INT = Union (Derive(Department@Exp7,
                                Name=DeptName,
                                NumOfEmployees=Employees,
                                Interest=Field),
                          Derive(Faculty@Exp3,
                                Name=Faculty,
                                NumOfEmployees=Sum*(Staff, Researchers),
                                Interest=Area))

* Int function Sum(int s, int t)
  return (s + t)

```

Figure 6.11: Schema Definition and Derivation Specification - an example

Figure 6.12 illustrates an instantiation example corresponding to the derivation primitives, defined in figure 6.11. The example also shows the cases of attribute rename and user-defined function (e.g. *Sum*).

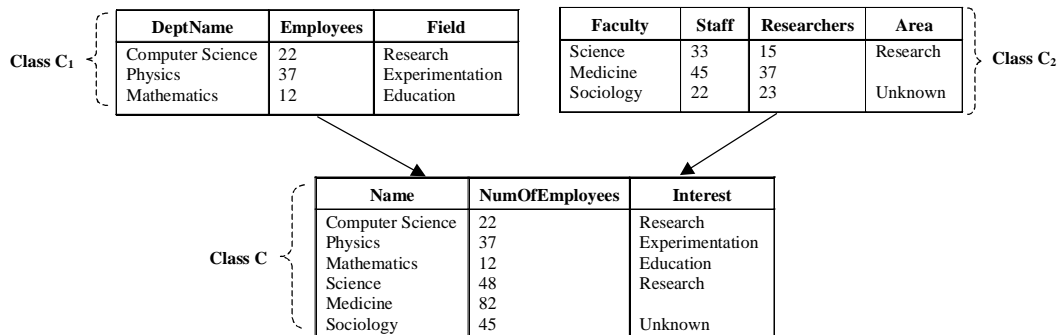


Figure 6.12: Classes and Attributes Instantiation - an example

6.2.2.4 Federated Query Processing

The task of the Federated Query Processor (FQP) is to process and respond to queries that may require accessing authorized data, and extracting and combining data from multiple sources. The sources are usually heterogeneous and geographically distributed. In other words, the federated query processor performs the functions of coordinating the sub-queries execution and provides a gateway to the multiple distributed data sources, where this complexity and distribution are hidden from the user; namely they are indistinguishable to the user from that of accessing a single database.

Figure 6.13 illustrates the steps involved in a federated query execution, which include:

1. The federated query processor first decomposes the query posed against the integrated schema into sub-queries on the local/remote data sources with their own local schema.
2. Various sub-queries are executed locally at the corresponding nodes and the sub-results are sent back to the federated query processor.
3. The federated query processor combines the local partial results into the common format and sends the complete results to the user requesting it.

The local and federated query processors are assisted in their functionalities with the set of resources specified at the local and federated layers.

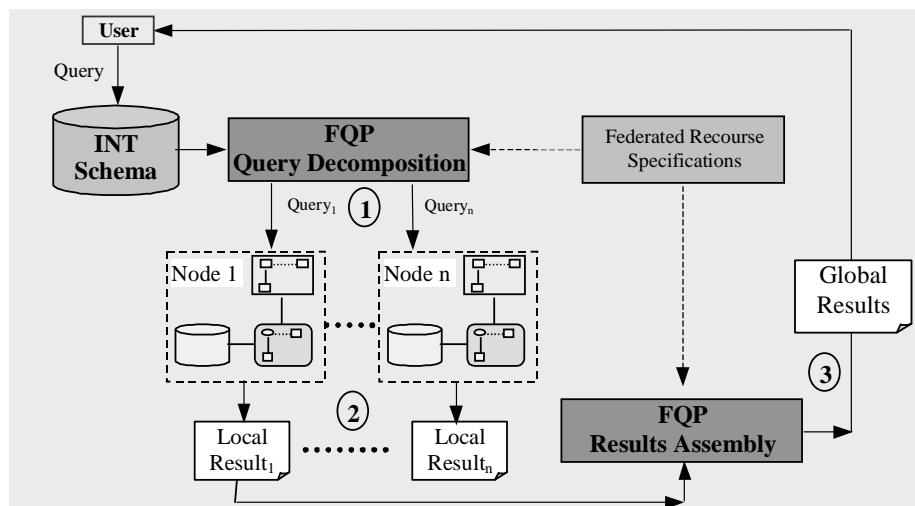


Figure 6.13: Federated Query Processor – The Steps

After defining the federated layer for different data sources within the federation, authorized users can either access the data in a tightly coupled way by creating their integrated systems or in a loosely coupled manner via direct on-line access to the shared data. Therefore, queries can also be individually submitted to an export schema at each node's federated layer without passing through the integrated schema. When a query is formulated against the export schema, the following actions will take place:

- The query will be rewritten by the Local Adaptation Layer (LAL), in order to be conforming to the local query language. The query is translated based on the local resources specifications (LRS) of the underlying data source,

- The query is executed against the local underlying data source from which, the export schema is derived,
- The query result will be reorganized to fit the export schema, and returned to the user that has requested it. The Local Adaptation Layer reformats the results based on the local resources specifications (LRS) of the underlying data source,

Figure 6.14 illustrates the performing mechanism of the federated query processor and its interaction with the local query processor at each local database system. The federated query processor supports:

- Decomposition of the query arrived from an application (in common format) to a set of sub-queries, each to be sent to the corresponding underlying data source.
- Assembly of the partial returned sub-results into a common coherent format that can be understood by the requesting users/applications.

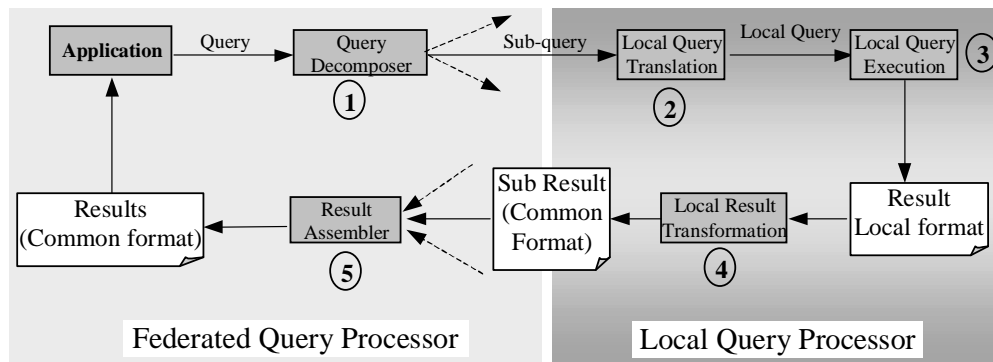


Figure 6.14: Federated Query Processor – Performing Mechanism

Federated Query Decomposer

The Federated Query Decomposer is a component of FQP, which decomposes a federated query to a set of sub-queries, each to be sent to the corresponding underlying data sources to be answered. The sub-queries are expressed using the common query language that is adopted at the federation layer. When a sub-query arrives at a site, the LAL at that site performs the translation of the sub-query to conform to the local query language and then it executes it.

Results Assembler

The Results Assembler is another component of FQP that combines the various sub-query results for every federated query, in the format of the common data model (CDM). Similar to the query translation process, the LAL at each site performs the transformation of the sub-query result from the local format (of its underlying data source) to the common format adopted at the federation layer. The Results Assembler then, merges the data produced by several local sites into a single format corresponding to the CDM.

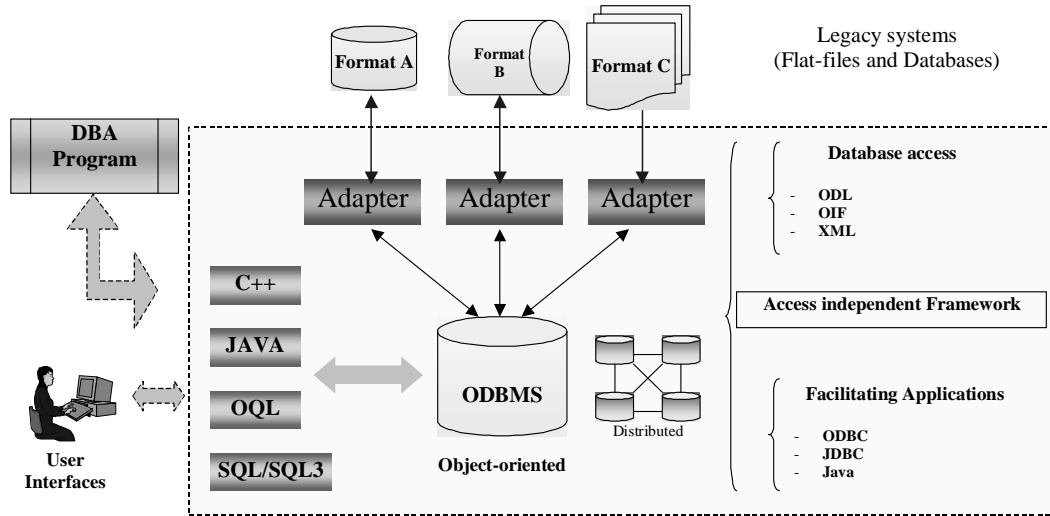
6.2.3 Application of Database Standards and Middleware Solutions in GFI₂S

Considering the wide variety and heterogeneity of networked applications, any efficient solution for their integration/interoperation must involve the *use of standard mechanisms* for applications modeling, data structuring and information retrieval. Standard mechanisms eases the design and building of integration mechanisms for collaborative environments, and solves many barriers faced in the integration process. For instance, the use of emerging standards for application modeling (e.g. UML), for data structuring (e.g. ODL), for information retrieval (e.g. SQL or OQL), and for data exchange (e.g. OIF and XML) reduces the need for construction of individual data translation wrappers among integrated nodes, and facilitates the development of necessary federated query processor. However, similar to many others, in the database area, standards lag behind in supporting new features and extensions provided by certain commercial and research prototypes of database management systems. Most DBMS's extensions nowadays address the development of advanced constructs to better support the ever-growing requirements of emerging applications. The new constructs provided by these database systems address multimedia data types, object-orientation concepts, interoperation/integration facilities, distributed computing, etc.

The GFI₂S federated architecture, as conceived and described in this chapter, presents an open and flexible solution towards a generic approach for information exchange and data integration, applying standards to the extent possible, and suggesting their extension when standards are not available (see figure 6.15).

- On one hand, from the database development perspective, ODL is used in GFI₂S to support the portability of database schemas across conforming ODBMSs. OIF and XML are used to exchange objects between databases and provide database documentation. Simultaneously, independent access facilities among the data sources are supported through middleware solutions (e.g. ODBC, JDBC, and JAVA).
- On the other hand, from users, applications, and database accesses perspectives, the GFI₂S system targets a comprehensive solution, based on standard interfaces and languages (e.g. SQL, OQL, Java, and C++), which:
 - Provide most of the requested information for users and applications,
 - Facilitate the access to the heterogeneous and autonomous data sources,
 - Support flexible access to those underlying data sources based on standard technologies and via secure and reliable export schemas mechanisms,
 - Support multimedia information and large data sets.

The use of *database standards* and *middleware solutions* facilitate the exchange of information among different applications. Namely two aspects are emphasized. First, *full and rich representation* of the schema concepts, query language, and data representation are supported through the object-oriented database standards e.g. ODL, OQL, and OIF. Second, the *user facilities and data transparency* are supported through Web standards and Middleware solutions e.g. ODBC, Java, and XML. More details on the applicability of these standards and middleware to GFI₂S are included in Appendix A. Some of the problems that face the standardization process are also addressed and discussed within these sections.

Figure 6.15: GFI₂S - Global Overview

6.2.4 GFI₂S in Action

The GFI₂S system is designed (and partially implemented) to better solve the information integration issues that were initially tackled in Chapters 3, 4, and 5. The flexible GFI₂S architecture can support a wide variety of collaborative environments, from a fully federated network of expert systems supporting distributed control applications (e.g. Waternet), to a distributed Virtual Laboratory environment (e.g. VL), and even for brokerage of information from remote sites. This section describes one example operational case of the GFI₂S system for brokerage of information for distributed, heterogeneous, and autonomous sites. This example shows the quality of GFI₂S and illustrates the main benefits gained when deploying the GFI₂S approach. The example also demonstrates the integration of GFI₂S with universal access to data, data export, and other tools as explained in Chapter 5.

The example presented in figure 6.16 illustrates a flexible framework, which is based on the GFI₂S architecture. The three-tier (client/server) architecture adopted here for web services satisfies many information management requirements for the entire interoperation and collaboration tasks among distributed networked heterogeneous data sources and applications. It mainly (1) deploys the GFI₂S system, following the node-to-node federation approach, at the lower-tier; (2) a set of interfaces for data access and direct interaction among heterogeneous data sources using middleware and standard solutions, and (3) supports a number of defined users/applications accessing the integrated information system via these interfaces.

In more details, the global architecture presented in figure 6.16 is composed of the following components:

- ❶ At the GFI₂S Federated Layer (*lower-tier*), the approach for integrating autonomous and heterogeneous data sources is addressed from two sides. On one hand, it develops a Local Adaptation Layer (LAL) for each data source, participating in the federated network. On the other hand, it builds the Node Federation Layer (NFL) on top of the conceptual data model (CDM) of the federated schema. The LAL components assure

proper communication between the local data sources and their federated layer. While, the NFL component supports the inter-connection of different information sources at the federated layer.

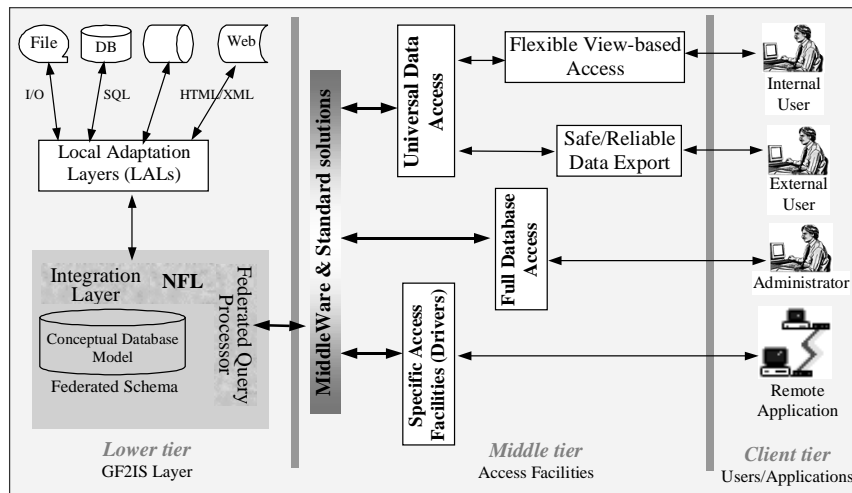


Figure 6.16: Global Architecture and Interfaces to GFI₂S

Additionally, The **GFI₂S** architecture employs middleware and standards to provide standardized interfaces for all types of data (structured and unstructured), maximizing interoperability and reusability, while leaving the data either at the place where it is generated or where it is heavily used.

- ② At the *middle tier* layer, three types of interfaces for local/remote data access are supported. The interfacing components facilitate the access to information for each type of authorized users in a secure and confident manner. The access to data is a dynamic process, which consists of direct interaction with heterogeneous data sources. The developed interfaces allow the access to the underlying heterogeneous data sources via the use of middleware and standard solutions (e.g. ODBC, JDBC, Java, and XML). The three main interfaces illustrated in 6.16 present the global access facilities to the underlying heterogeneous database systems through **GFI₂S**, which mainly include:

- * *Universal data access* and direct interaction with heterogeneous data sources using middleware solutions and standards.
- * *Specific access facilities* (tailored interfaces) to data. These interface also dynamically interact with the remote data sources, but they are application dependent.
- * *Full access* for administrators to heterogeneous remote data sources through the **GFI₂S** tier.

These interfaces can also be used as a base for several enhancements. As such, from the *Universal Data Access* interface two variants were derived: the *Safe/Reliable Data Export* and the *Flexible View-Based Access* interface.

- ③ From the client tier, the four types of users/applications, which can access the data within such a system are:
 - * Database Administrator (DBA) that has full access to all data sources, and can update both the data and meta-data. The database administrator has the ability to access the data through standard tools or via specific access facilities.
 - * Users inside the organization (internal users/developers) are supported by a flexible data access interface, which is based on view definition for information visibility rights and security for access. Internal users may also have the ability to update the part of data of their responsibility.
 - * External users (collaborators) are supported by safe/reliable data export facilities through which, they can access information of their interest in an open and secure manner. In addition to the user friendly and flexible screen-based presentation of the requested information, this interface also supports the provision of data in several physical standard formats (OIF, XML, HTML, etc.) that can be uploaded at the user site. The safe and reliable data export interface only supports restricted read accesses.
 - * Specific applications that are clearly defined and are not expected to change are supported through specific application-dependent interfaces. These applications need self interaction with the database, information is requested from the database when needed and will be delivered at the requesting time.

6.3 Conclusion

The architecture of the Generic and Flexible Information Integration System (**GFI₂S**) is designed as a unified federated object-oriented layer that can support the integration of information among heterogeneous and autonomous data sources. Thus, supporting the information exchange and collaboration among heterogeneous and autonomous nodes. In this system, a data source can represent any database built using a commercial DBMS product, or any information source (e.g. a file system). The integrated schema, defined within the node federation layer of **GFI₂S**, provides users with a consolidated view of the information shared for the purpose of collaboration. The integrated schema is defined at the federated layer of every node in the network, while the data corresponding to it is distributed among the individual local/remote data sources. Thus, data in all of the local and remote nodes is accessible by others as if it belongs to a single local database.

The **GFI₂S** integration architecture is constituted of two main components: the Local Adaptation Layer (LAL) and the Node Federation Layer (NFL). This decomposition properly serves the need for sites' interoperability and information sources integration. The **GFI₂S** architecture presents a flexible approach based on standards and middleware solutions, through which:

1. *The Local adaptation Layer (LAL)* at each node, defines the set of mechanisms and concepts that facilitate the access to underlying local data sources in their native format, and the control of their access rights.
2. *The Node Federation Level (NFL)* at each node, acts as a mediated common interface that sits between the local system and other external data sources participating in the federation.

The **GFI₂S** federated architecture, as conceived and described in this chapter, presents an open and flexible solution towards a generic approach for information exchange and data integration:

- From the *database development perspectives*, ODL is used in the **GFI₂S** to support the portability of database schemas across conforming ODBMSs, and OIF and XML are used to exchange objects between databases and provide database documentation. Simultaneously, independent access facilities among the data sources are supported through standards and middleware solutions (e.g. ODBC, JDBC, and JAVA).
- From the *user access perspectives*, the **GFI₂S** system targets a comprehensive solution, based on standard interfaces and languages (e.g. SQL, OQL, Java, and C++), which facilitate the access to distributed, heterogeneous, and autonomous data sources. The flexible access to those underlying data sources is achieved through standard technologies and via secure and reliable export schemas mechanisms.

As such, the **GFI₂S** federated approach allows the integration of databases and applications that are distributed, autonomous, and heterogeneous.

Chapter 7

Conclusions and Future Work

7.1 Overview

A wide variety of distributed applications are nowadays emerging in diverse domains of science, business, engineering, education, e-commerce, tourism, etc. These applications deploy various database systems for the management of their information, in which the diversity stems from reasons related to the specific information management requirements and the objectives targeted by these applications. Other reasons may also concern suitability, efficiency, and security. In today's organizations, new and existing applications such as design, manufacturing, or decision making environments, require access to data stored in several of pre-existing databases detained at several local and remote sites. To satisfy the new information management requirements of these organizations, a strong information integration system must be designed and developed, serving the need for information integration and interoperation among these organizations.

This dissertation describes the design and development of an information integration approach to support the integration of heterogeneous information sources while preserving their local autonomy and distribution. The first step in this direction consists of a global survey focusing on the related research and approaches for information integration and interoperation among autonomous and distributed systems. The survey of existing approaches emerging in this domain forms the *state-of-the-art* and the related research work for the dissertation. Considering the main emphasis of the thesis, this survey is conducted by a *classification of existing approaches and methodologies* for information integration. Classification of these approaches, as addressed so far by other researchers, is mostly based on three concepts of database architectures, data access and storage mechanisms, and systems interoperation. The taxonomy for information integration approaches, proposed in chapter 2, divides them into two main categories: *Distributed Systems* and *Integrated Systems*. Distributed systems typically share common database control software at both DBMS servers and applications. Integrated systems however, support database applications that address decentralized/autonomous database control, using different representations and data modeling systems. Within each of these two categories several approaches are identified, studied, and evaluated based on the applications' requirements.

Research on integrated systems distinguishes between *Physical Integration* and *Virtual Integration*. In a *Physical Integration* the data originating from local and remote sources are integrated into one single database on which all queries can operate. In *Virtual Integration*,

data remains on the local/remote sources, queries operate directly on them and data integration takes place on the fly during the query processing. At deeper levels of the taxonomy, when the required level of integration becomes more complex and when the requirements are higher, the variety of the proposed approaches becomes more and more specific and complex. On one hand, the physical integration expands into *centralized databases* and *data warehouses*. In a centralized database, information is migrated from various sources into a universal DBMS, while in data warehousing information may be imported in different format and volume than it exists in its originating sources. On the other hand, the virtual integration derives into *federated* and *non-federated* systems. Furthermore, each of these systems can be either *loosely* or *tightly coupled*.

Most approaches presented and discussed in chapter 2, do not properly support the extensibility and the evolution of applications, rather they address specific domain-dependent cases. Adding a new site to the federation, or applying a given approach to a different application domain, requires considerable expertise and effort in order to interface it with all systems participating within the federation. Still, these approaches bring considerable advantages, which can be adapted and deployed. The information integration approach, we presented in chapter 6, benefits from these approaches and follows a strategy, supporting standard languages, generic tools, and middleware solutions. The major benefits from which, the integration approach takes advantages are discussed in details in section 7.2.

The development of several systems and tools to support the management and exchange of information and the data integration purposes, during the preparation of the thesis, has provided the means to better understand the complexity of such processes, and to better deploy standard tools and middleware solutions.

1. The **Waternet** system, presented in chapter 3, aims at an evolutionary knowledge capture and management system supporting the control, optimal operation, and decision making for the management of water distribution in a network of expert systems, provided the proper environment for developing an open and flexible architecture for integration of different Waternet modules. From the development of the Waternet integrated/federated environment, we learned how to design and develop flexible, open, and reliable environments for information management systems supporting the following characteristics:
 - System openness, so that different sites can be added to/removed from the federation community, in order to support the specificities of different application domains.
 - Data distribution, so there is no need to develop a single global schema and a common glossary of concepts among the networked sites.
 - Complete data location transparency to the user, of logical/physical distribution of information among the sites in the network.

The developed aspects and the lessons learned, during the design and development of the *Waternet* system, contributed to **GFI₂S** by tackling the fundamental schema management challenges at the federation layer, and by serving the system openness through the adoption of the data adapters at the node layer.

2. The **MegaStore** framework, presented in chapter 4, aims at the design and set-up of the necessary database structure and platform architecture for advanced e-commerce applications, and in specific, addressing the CD and music industry. It provides a

good example for the deployment of database standards and middleware solutions. Its development is supported through the coupling of Web standards and middleware with advanced database technologies. The main idea behind the developed framework for MegaStore is to design a comprehensive system to support applications with the following characteristics:

- Facilitate the storage and manipulation of multimedia large data sets.
- Provide a flexible information classification and clear separation between public and proprietary data.
- Extend Web services in E-Business applications with the functionalities for flexible navigation through complex Web objects, scalability as necessary for multimedia large objects, high performance as required by multi-users applications, and so on.

The design and development of MegaStore framework contributed to **GFI₂S** through (1) the deployment of database standard and Internet Middleware supporting system reusability, (2) the development of a parallel/distributed database server assuring system efficiency, and (3) the development of user friendly interfaces assisting advanced/ordinary users in accessing the underlying information sources.

3. The *Virtual Laboratory (VL)* Information Management, presented in chapter 5, aims at the design and development of a software layer and an enhanced architecture, supporting scientists in their experimentations, and providing the basic information management requirements for the emerging multimedia applications in e-science. Our contribution within VL Information Management focuses on specific advanced features, functionalities, and facilities introduced and developed for management of scientific data for VL applications. Specific subjects addressed within VL include:

- *Strategies* for storage and retrieval of large scientific data sets.
- *Use of standards* for scientific data modeling and archiving.
- *Universal and schema free access* to scientific data.
- *Access to scientific data based on the predefined visibility restricted schemas,*
- *Scientific Results Publishing, performance issues, Benchmarking tests,* etc.

The concepts, addressed within the VL Information Management framework, addressed a number of important issues that can be applied at every site to better enable it as a node in the cooperation network. On one hand, individual sites can benefit from the use of these concepts to efficiently build their applications independently of other sites. On the other hand, the use of standards at local sites helped the development of information integration mechanisms for **GFI₂S**.

The approaches for *Waternet* and *MegaStore* systems provide specific mechanisms for the design and development of information integration systems, and illustrate the main benefits of using standard solutions to support the information sharing and the data integration. While, the VL Information Management, particularly employs these standard concepts to the information integration mechanism and provides a forward benefit to it. VL information management framework applies the Web and database standards at every site, making its components stronger and more suitable for integration and cooperation, while preserving their full autonomy and specific characterization. The various concepts and lessons learned

during the development of the application cases, described above, have contributed to the design and partial development of a generic and flexible information integration system.

The *Generic and Flexible Information Integration System (GFI₂S)* is designed and partially developed to give its users and applications, access to heterogeneous information sources through generic and flexible interfaces. The distinctive features of the **GFI₂S** integration approach reside in (a) *the specific combination* of database standards and Internet middleware with the fundamental research approaches, and (b) *the way in which they are deployed and inter-linked* within the components of GFI₂S architecture. Its architecture smoothes the transition from relational and object-relational database systems to a system that unifies most DBMSs capabilities. The **GFI₂S** approach, which follows the ODMG standards, is considered to support different data sources and provides application developers with a single, seamless application view, and unified access to all information in those underlying data sources. The **GFI₂S** integration architecture is constituted of two main components of: (1) Local Adaptation Layer (LAL), that facilitates the access to the underlying databases in the node, and (2) the Node Federation Layer (NFL), that provides links to the information and applications outside the nodes and supports the information sharing and interoperation. This two-component architecture of **GFI₂S** provides existing systems with efficient means for their interconnection and interoperation, while preserving their heterogeneity, distribution, and full autonomy. The **GFI₂S** architecture benefits from existing approaches and applies emerging information technology to support the new requirements of scientific and advanced applications. More details regarding the benefits of **GFI₂S** comparing to other approaches are described within the next sections.

7.2 GFI₂S Compared to Other Approaches

The design approach of **GFI₂S** benefits from the careful choice of its specific components. The components are decided based on the state-of-the-art in the area of information integration for systems interoperation. Additionally, in order to support the new requirements from emerging scientific and advanced applications, the **GFI₂S** approach utilizes database standards, Internet technology, and Middleware solutions. The main aspects which are taken into account in the design of **GFI₂S** system are listed below:

- * Standard languages for data modeling and information access are adopted at the federated layer of **GFI₂S**. In the area of information exchange and data integration, several initiatives are emerging in the direction of standardization (e.g. STEP, and NetCDF). Most initiatives consider their specific terminologies for the data representation and manipulation. Thus, new standards are appearing rapidly, while, similar solutions already exists: the database standards. The **GFI₂S** approach benefits from these initiatives and extends the usage of their architectures via the use of database standards in term of data modeling, querying language, and information exchange. In the **GFI₂S** information integration approach, data is not bi-translated, rather, queries are sent from one site and data is received from the other site.
- * The federated schema constitution within **GFI₂S** is based on and extends the PEER approach. PEER uses its specific language for schema definition, mapping derivation, and query formulation. While, **GFI₂S** uses the UML for data modeling, ODL for data definition, and OQL for data access and information retrieval. Such an approach for schema and data management makes **GFI₂S** an open integration facility for other systems, which are compliant to these standards.

- * The usage of XML and OIF data formats within the **GFI₂S** system, facilitate the applicability of database concepts to the federation by enforcing the data exchanges between different organizations in a widely accepted format. The availability of data in standard formats of XML and OIF, within the collaborative environments, reduces the number of wrappers to be developed, and facilitates the data translation among heterogeneous systems when services are requested between them.
- * The use of Object-Oriented database standards as common languages for data modeling and querying provides the possibility for integrating most types of applications ranging from the CODASYL network model, to relational model, to object-oriented and Object-Relational models.
- * The use of extended ODMG mechanisms supports the mapping specification and derivation operations between the underlying data sources and the integrated database schema, at the **GFI₂S** federated layer.
- * The structural representation and the semantics resolution of data from heterogeneous sources are enforced at the **GFI₂S** federated layer by a dictionary of terms and a dictionary of semantics. These dictionaries, which are available for each exported/integrated schema, help users in defining their own federated schemas without the need for external support from experts that devote the sharable (exported) information. The dictionary of terms serves for automatic conflicts resolution, while, the dictionary of semantics reflects in fact the experts' knowledge of the application.
- * The **GFI₂S** federated architecture adapts the approach of defining conceptual wrappers for legacy databases and developed the Local Adaptation Layer (LAL), in order to provide interoperability between legacy systems. The LAL extends the role of wrappers to also include information about users authentication and information visibility levels.
- * The specific data structure and querying language of each node within the federation are preserved. Queries formulated at the **GFI₂S** federated layer are translated to be conform to the local data source query language before being executed. Additionally, the local results are translated to the common format adopted at the federated layer. Therefore, the **GFI₂S** federated approach does not require translating the complete existing data of different databases to the federated layer, rather, it focuses on translating only the part of data that is needed to be exchanged, e.g. the result of a query into a common format.

7.3 Lessons Learned

Generic and Flexible Information Integration Systems must satisfy the requirements of Flexibility and Genericness. From the design of the **GFI₂S** information integration approach, we learned that flexibility of information integration systems resides in the architecture they rely on, while their genericness can be achieved via the deployment of database standards, Internet technologies, and middleware solutions. Thus, the learned aspects are to be considered in the design and development of information integration systems, in order to provide an open facility for integration/interoperation among heterogeneous, distributed, and autonomous sites. Below is a list of the main lessons learned and the expertise gained within the design and development work of the various R&D projects, during the preparation of this dissertation:

- ❶ The use of two components (LAL and NFL) for the information integration among networked sites makes the integration mechanism flexible. This flexibility is supported from two sides, on one hand, sites can join or quit the federation; on the other hand, the schema integration strategy followed at the node federation layer allows for a customized integration, which can be tailored to the need of each site.
- ❷ The use of object-oriented database standards and middleware solutions at the federated layer of the **GFI₂S** makes its architecture generic. Each site that wishes to join the federation only needs the knowledge about its "underlying database system" and about the "standard languages and formats" adopted at the federation layer. The local users at each site gain proper expertise about the underlying local application's characteristics and specifications. At the same time, the standard languages and formats adopted at the federation layer are mostly understood by these users.
- ❸ The use of standard languages for data definition and information access (ODL, OQL/SQL, XML), which are widely adopted by a large community, reduces the efforts needed when defining export and integrated schemas, and facilitates the access to data within networked applications.
- ❹ The use of middleware and standards mechanisms for data access (e.g ODBC and JDBC), information exchange (e.g. XML), and communication protocols(e.g. CORBA) play an important role in reducing the number of intermediate interfacing tools, unifies the access to shared information, and facilitates the data integration among heterogeneous databases and applications.
- ❺ The Consideration of data aspects such as, scientific information, large data sets, and complex inter-linked objects supports the development of complex applications. In addition, the provision of generic mechanisms and tools for scientific data publishing, based on tailored views on the sharable data, preserves systems' autonomy and hides private data from outside users.

Various concepts, enumerated above, provide the base information integration aspects for systems interoperation. The next section will identify some of the remaining issues, in the area of information integration, that need to be further addressed and described.

7.4 Future Work

In order to facilitate the information integration process among heterogeneous applications, attempts to integrate autonomous, distributed, and heterogeneous applications must be strongly based on the use of database standards and middleware solutions. Middleware solutions unify the communication process among interconnected applications, while database standards unify their exchange of data. Thus, the use of database and middleware standards constitute the base for flexible, open, and generic integration among networked applications. Use of standards for data modeling and information retrieval also eases the interoperation/collaboration process with pre-existing application and legacy systems, and reduces the need for construction of individual data translation wrappers.

However, in order to apply standard concepts to the environment of networked applications, certain extensions to database and middleware standards must be addressed to better support the information exchange and data integration among interoperable systems. For

instance, In the database area (similar to many others), standards lag behind in supporting new features and the extensions provided by certain commercial and research database management systems.

Following areas require to be further addressed by the researchers in the field of information management, by the standardization community, and by the DBMS developers:

- The development of advanced standard constructs to better support the specific requirements of complex scientific applications, and to address their data types, object-orientation concepts, interoperation/integrated facilities, distributed computing, etc.
- The extension of database definition language to properly support the mapping constructs and the derivation operations is required, to better support the federation of several heterogeneous databases. As such the object definition language (ODL), for instance, needs to be extended to support the mechanism of schema integration in the area of federated databases. The extensions to the ODL are expected to address the definition of export and integrated schemas, and to provide a set of operations supporting the needs for their derivation mappings.
- The extension of database query language with object-oriented features is required. Currently, different DBMS developers use their specific SQL/OQL extension mechanisms, which differ from one DBMS to another. To overcome the issue of specific extensions, if not by standardization, there must be a consensus among DBMS developers about these extensions; at least a common agreement regarding the main required features such as object identifier, inheritance, path expression, and cross-relationship references must be achieved.
- The consideration of standard data exchange formats, e.g. XML for databases and in particular for information integration, raises several challenges for database research. Having XML focused only on the syntax for data representation partially increases the prospect of its integration. To support information integration at the semantic level, however, there must be a further standardization or agreements upon DTDs (schemas) in XML. Furthermore, at present, the XML data does not conform to a fixed schema; names and meanings of the used tags are arbitrary and the data is self-describing in XML documents. Therefore, several XML-issues need to be addressed to enable the information integration (e.g. languages for describing the contents and capabilities of XML sources, query reformulation algorithms, translation among DTD's, and obtaining source descriptions).

In addition, we must also admit that the issue of information integration is of a very high complexity, especially when the information sources are heterogeneous, distributed, and their local autonomy is preserved. This thesis work described a high level architecture for information integration, in order to give a global overview of a generic and flexible information integration system. Therefore, complete descriptions and full coverage of all the components of **GFI₂S** are out of the scope of this dissertation. Several components of **GFI₂S** are addressed to the required level of details, while others are globally described, leaving a number of issues and problems to be further addressed by other researchers. Among the remaining issues that require further research, we enumerate: updates in federated schemas, derivation mappings to cover the relationship concepts, and better addressing technology-independent issues (e.g. theories, and formal specifications). Some other issues in the domain of information integration were already addressed by the research community, for that reason these issues are addressed but not fully described in **GFI₂S** (e.g. federated query processing, wrappers, and semantics resolution).

However, the **GFI₂S** architecture is flexible enough to be augmented with software components, which are designed/developed by other research/software institutions. For instance, the **GFI₂S** architecture allows an application to use an existing tool for conflicts resolution when defining export and integrated schemas. Such a tool can be based on, and enforced by, the dictionary of terms and dictionary of semantics defined at the federation layer of **GFI₂S**.

Appendix A

Application of Database and Middleware Standards in FGI₂S

The **FGI₂S** federated architecture, as conceived and described in Chapter 6, presents an open and flexible solution towards a generic approach for information exchange and data integration. **FGI₂S** uses object-oriented standards and middleware solutions to the extent possible, and suggests their extension when standards are not available.

- From the database development perspective, object-oriented database standards are used in **FGI₂S** to support the portability of database schemas across conforming ODBMSs, to exchange objects between applications, and to provide database documentation.
- From users, applications, and database accesses perspectives, the **FGI₂S** system targets a comprehensive solution, based on Web standards and middleware solutions to facilitate the access to the heterogeneous and autonomous data sources,

This appendix describes the use of *object-oriented* and *Web standards* to facilitate the exchange of information among different applications and databases. Namely, two aspects are emphasized. First, *full and rich representation* of the schema concepts, query language, and data representation are supported through the object-oriented technologies and standards e.g. ODL, OQL, and OIF, as described in subsections of section A.1. Second, *user facilities and data transparency* are supported through Web standards and Middleware technologies e.g. ODBC, Java, and XML, as described in subsections of section A.2. Some of the problems that face the standardization process are also addressed and discussed within these sections.

A.1 Object-Oriented Standards and Extensions Adaptation for FGI₂S

This section illustrates the benefits gained when deploying database standards for the management of information and data integration. It also addresses extensions to these standards to better support the integration/interoperation process for the schema modeling, the query

formulation, and the data representation; from the various formats adopted at the local data sources to the common format adopted at the integration/interoperation level.

Object-Oriented standards are defined and specified to better support the portability of database schema and database objects across conforming ODBMSs. Portability in object database management systems would allow an application program that is written to access one ODBMS to be able to access another ODBMS, as long as both ODBMSs support the ODMG standard faithfully.

The various components of the ODMG specification, from which the integration (interoperation) process can benefit, include [CBB⁺00]:

- An *Object Model*, which gives database capabilities including definition of relationships, extents, collection classes, and concurrency control.
- An *Object Definition Language* (ODL), which allows defining a database schema in a programming-language in terms of object types, attributes, relationships, and operations.
- An *Object Query Language* (OQL), which includes support for object sets and structures and supports object identity, complex objects, path expressions, operation invocation, and inheritance.
- *Language Bindings* to Java, C++, and Smalltalk, which extends the respective language standards to allow the storage of persistent objects; each binding includes support for OQL, navigation, and transactions.

Currently, ODMG is the only standard interface that allows to store Java objects directly using a standard API that is completely database independent, indifferently of the underlying storage mechanism, being a relational or an object database. If the database system has an interface that conforms with ODMG, it can store objects directly using the standard Java API.

In conjunction to object-oriented databases, object-relational DBMSs emerged as a way of enhancing the capabilities of relational DBMSs with some of the features that appeared in object-DBMSs. These features include wide variety of data types, complex objects, and audio/video data streams. Examples of object-relational DBMSs, which are emerging nowadays, include Informix Universal Server, Oracle, Matisse, and DB2 Universal Server.

The extension of database standards can also be adapted, to properly support the mapping constructs and the derivation operations within a federation of heterogeneous databases. The following sub-sections describe in more details the application of the ODMG standard to schema integration and systems interoperation.

A.1.1 Object Definition Language – ODL

The object definition language (ODL) provides the semantic power for schema definition and offers a standard, which can be extended to support the mechanism of schema integration in the area of federated databases. The extensions to the ODL are expected to address the definition of export and integrated schemas, and the provision of a set of operations serving the need for mapping and derivation. Some research work started a decade ago considering the extensions of ODMG-ODL standard, to also support the schema integration and its evolution. The research is also advancing in the direction of federated schema in ODMG

[BFN 94], extending the ODMG for federated databases [Rad 96, KJR 98], and extending ODL for object-oriented views integration [RKB 01].

Extension of ODL, by researchers and developers in the area of federated databases, provides a powerful mechanism, semantically rich, for integrating schemas from multiple heterogeneous data sources, and to define a virtual database that can be queried just like any stand-alone/centralized database. Extended ODL can be used for export and integrated schema specification from two perspectives. It can be successfully used for: (a) export and integrated schemas definition, and (b) the mapping specifications for the derived schemas, namely operations for the data translation e.g. join, select, and intersect. In **GFI₂S**, Export and Integrated schemas definition comply to the ODL syntax, with only the exception that an export schema is a subset of a local schema, while the integrated schema is a subset of the union of the local schema with a number of imported schemas. Similarly, the derivation operations, for export and integrated schemas, are based on a schema derivation language, which extends the ODL syntax and benefits from other derivation languages developed to better support the information integration among heterogeneous and distributed applications. More details regarding the federated derivation primitives and some examples are provided in section 6.2.2.3.

Hereafter, we present a simple data modeling example that will be used within the next sub-section. In this example from the scientific domain, an *Experiment* is defined by its: *domain*, *results*, *date*, and it is performed by a (*Scientist*). the scientist is in turn, defined by his *name*, (working) *field*, and the *Experiments* that he/she performs . In a relational model, this can be represented as follow:

```
Experiment (Exp_ID, Sc_ID, Domain, Results, Date)
Scientist (Sc_ID, Exp_ID, Name, Field)
```

Using the ODMG-ODL language, the complete definition for this simple scientific application is presented as follow:

```
Interface Experiment:{
  attribute String Domain;
  attribute String Results;
  attribute Date Date;
  relationship SET<Scientist> PerformedBy
    Inverse Scientist::Performs;
}

Interface Scientist: {
  attribute String Name;
  attribute String Field;
  relationship SET<Scientist> Performs Inverse
    Experiment::PerformedBy;
}
```

The links between the two entities in the relational model are expressed through the notion of foreign keys (e.g. *Exp_ID* and *Sc_ID*). While, in object-oriented model, links are explicitly expressed through the relationship concept (e.g *Performs* and *PerformedBy*).

A.1.2 Query Languages – SQL, SQL3, and OQL

Query languages are considered to facilitate the interaction with the database and permits the management and maintenance of its instances. Within the area of database query languages, SQL is continuing its evolution towards a new standard called SQL3, which is extended to deal simultaneously with tables from the relational model and classes from the object model. SQL3 language extends SQL standard by incorporating object-oriented capabilities and features such as complex data types, user-defined routines, inheritance, and indexing extensions.

Similarly to SQL3, the Object Query Language (OQL), enhances the data types, predicates, relational operations, triggers, user-defined types, transaction capabilities, user-defined routines, and extends the SQL language to include object-oriented capabilities. Its syntax for queries is similar to the syntax of the relational standard query language SQL, with some additional features for object model concepts, such as object identity, complex objects, inheritance, polymorphism, path expression, and cross-reference relationships.

To illustrate an example of these extensions to the SQL standard language, let us consider the example from the scientific domain presented at the end of the previous section. An SQL query, for instance, requesting experiment results performed by a scientist named 'John' before 'May 12, 2001', involves condition predicates upon the two entities *Experiment* and *Scientist*. Using the relational model, the query request can be formulated as follow:

```
SELECT Results FROM Experiment WHERE Date <= '12-05-2001' AND Sc_ID IN
(
  SELECT Sc_ID FROM Scientist WHERE Name = "John"
)
```

While using an extended Matisse-SQL, which is based on the object model, a query for the same request, is formulated as follow:

```
SELECT Results FROM Experiment WHERE
  Date <= '12-05-2001' AND PerformedBy.Scientist.Name = "John"
```

SQL Queries in object-oriented/object-relational DBMSs are formulated slightly different than in a standard SQL. Matisse-SQL, for instance, is enhanced to deal with the challenging applications of today by incorporating the object-relational concepts, which mainly concern object identity, inheritance, encapsulation, path expression, and support for multiple data types and complex objects.

Thus, when it comes to extending the SQL standard, different DBMS developers use their specific extension mechanisms, which differ from one DBMS to another. To overcome this issue, there must be certain kind of consensus (or standardization) among DBMS developers concerning these extensions. At least a common agreement regarding the common extensions such as: object identifier, inheritance, path expression, and cross-relationship references.

Hereafter, we illustrate some of the object-oriented/object-relational extensions to the SQL language. These examples are demonstrated using the Matisse DBMSs, and address the concepts of object identifiers, inheritance, cross-reference relationships, and path expression mentioned above.

Matisse DBMS uses the keyword *OID* within a select query to retrieve the object identifier of the database objects; and uses the keyword *ONLY* within a select query, to only retrieve the direct instances of a given class. As such, in the example below, the query “*select only * from scientist*” will not retrieve instances that are subclasses of Scientist. The third example below illustrates the usage of relationships and path expression concepts.

OID : *Select OID, * from Experiment*
Inheritance : *select only * from Scientist*
Relationship & Path expression : *select * from Experiment where PerformedBy.Scientist.Name='John'*

Among the factors that have made SQL a successful standard is its simplicity, especially for non-expert database users. OQL lacks this feature, especially in some cases when querying complex data models involving relationships and complex data types (structures, lists, arrays, blobs, etc.). The manner, in which, queries are defined is very hard to understand, even by users that are quite familiar with database terminology.

Within OQL, we believe that the language must be strongly based on the use of object identifiers, cross-relationship references, and path expression. The following example, for instance, retrieves the *Experiments Performed By a Scientist* with the *OID='0x124'*.

```
SELECT * FROM Experiment E
WHERE E.PerformaedBy.Scientist.OID = '0x124'
```

A.1.3 Object Interchange Format - OIF

Cattel et al. [CBB⁺00] defines the object interchange format (OIF) as a specification language used to dump and load the current state of an ODBMS to/from one or more files. Therefore, the OIF format can be used to exchange persistent objects between ODBMSs, seed data, provide documentation, and derive test suites. Since OIF allows the specification of persistent classes and their states, it can also be considered as a facility for information exchange between database systems and among heterogeneous applications.

In comparison to the XML standard, on certain aspects, OIF provides a similar powerful facility for data representation and information exchange between databases and applications. However, two major and distinct issues separate these two standards from each other. The first issue concerns the format for the data representation, where the OIF format only preserve the types, attributes, and relationship identifiers, as provided by the ODL definition of the ODBMS schema. While, the XML format uses additional naming tags for data representation, which requires extra effort for their handling. The second issue, relates to the consideration and the use of these standards. Even considering the first issue, which seems in favor of the OIF format, the XML standard is far more better and widely used by users and groups in several application domains, than the OIF standard that still has not had the chance for strong considerations by these groups and developers.

During the development of some projects related to the MegaStore framework, presented in Chapter 4, and also related to the Virtual Laboratory project, presented in Chapter 5; an OIF facility is developed serving the requirements of information exchange between databases and applications. The development of such a tool is motivated by the need for adequate facility for archives and backups, for information exchange between different applications, and for preserving the data consistency; specially, to facilitate the projects

in which new versions of DBMSs are released or when a different operating system and platform are to be considered.

The OIF tool developed for these projects (called *Mt.Oif*), is specific only for Matisse ODBMS, supports the database back-ups and recovery, and provides a facility for data exchange between different versions of Matisse ODBMSs with full support for different platforms (e.g. Sun Solaris, Linux, and Windows). All these tasks are performed through the single *Mt.Oif* set of code. Following is the syntax of the *Mt.Oif* tool:

```
Mt_Oif DB@host [user psswd] in|out [oif file]
Usage: Mt_Oif DB@host [user psswd] in|out [oif file]
DB      : Database Name
Host    : Host Name
user    : User Name
psswd   : User Password
in      : load an oif file
out     : generate an oif file
File    : File Name that contains data in OIF format to be loaded/generated
```

The following example command Dumps the data from the database *Sc_Experiment*, which runs on *amelie.wins.uva.nl*, and generates a backup file, named *Sc_Exp_Backup.oif*.

```
Mt_Oif Sc_Experiment@amelie.wins.uva.nl out Sc_Exp_Backup.oif
```

The loading of the generated OIF file into another database called *Sc_Exp_Bk*, which runs on the other host *carol.wins.uva.nl*, passes through the “*in*” parameter as follow:

```
Mt_Oif Sc_Exp_Bk@carol.wins.uva.nl in Sc_Exp_Backup.oif
```

The *Mt.Oif* tool handles all the rich and complete data types supported by Matisse ODBMS, which include among other types: Date and TimeStamps, lists and arrays, large and binary objects, and audio and video streams. As such, the audio and video streams are treated within the OIF format as lists of elements, similar to the way they are handled by the Matisse DBMS itself.

A.2 Web Standard and Middleware Adaptation for FGI₂S

In addition to ODMG, usage of standards like Java, CORBA, and XML can provide interoperability and portability to applications involving databases with different database models and systems. This section addresses a number of emerging information technologies from the perspective of how they could support higher levels of interoperability. Among the emerging technologies, which are relevant to interoperability and more related to the subject of this chapter, we cover in this section:

- Object Database Connectivity ODBC,
- Multi-platform applications development using Java,
- XML standard for information exchange

More standards relevant to interoperability are further emerging nowadays for information handling technologies, among which, we enumerate CORBA¹, Jini², DCOM³ (Distributed Component Object Model), SOAP⁴ (Simple Object Access Protocol), WAP⁵ (Wireless Application Protocol), and High-Portability Programming Languages. These technologies are however not addressed within this dissertation, due to space limitation. Readers interested in investigating these specific technologies in depth can refer to [TB 00, WAP 00, Vin 97].

A.2.1 Object Database Connectivity - ODBC

Providing common interfaces to heterogeneous databases has been a challenging issue in the domain of data access and information retrieval through standards. Open Database Connectivity (ODBC) provides a certain level of standard access using SQL as a standard language for interaction with the underlying database. Figure A.1 illustrates the data access mechanism using ODBC standard, which offers an open facility that provides a common set of API calls to manipulate databases. Using ODBC, users are able to develop a single application program that can access different DBMSs. This mechanism allows developers to build and distribute a client-server type of application without being restricted to a specific DBMS.

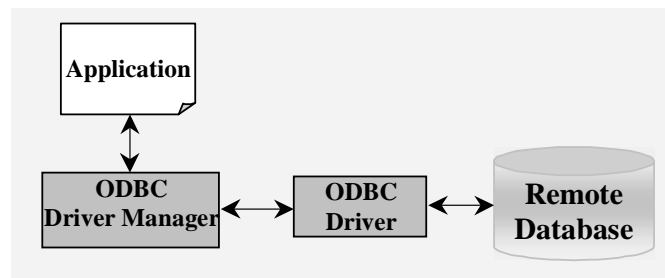


Figure A.1: Application Access to Remote Database via ODBC

In addition, ODBC is considered as a good interface for supplying data. A summary of advantages for using ODBC over native database APIs include:

- Providing an open standard, which is already supported by many development groups and organizations in research, industry, and academia.
- Enabling access, from a common set of code, to data from different relational and object-oriented database systems (e.g. Oracle, Matisse, Sybase, and MySQL).

Although ODBC allows developers to build and distribute client-server applications without targeting a specific DBMS, it has many limitations when used as a programming interface. For instance, consider the case that in many cases, different DBMS vendors support the common basic functionalities defined by standard ANSI SQL 92. However, the various

¹Common Object Request Broker Architecture (<http://www.omg.org/>)

²Jini™ Connection Technology Executive Overview (<http://www.sun.com/jini/overview/>)

³DCOM – Distributed Component Object Model (<http://www.microsoft.com/com/tech/DCOM.asp>)

⁴SOAP - Simple Object Access Protocol (<http://www.develop.com/soap>)

⁵WAP Forum Specifications (<http://www.wapforum.org/what/technical.htm>)

extensions and added-values to different database systems has forced DBMS developers in this area to extend the SQL standard to support their added value features and new built-in data types. This is the case, especially with the appearance of object relational and object-oriented databases, in which the extensions mainly concern object identifiers, abstraction, inheritance, and cross reference relationships. The database system developers therefore find themselves forced to support the new challenging features also via the extensions of the ODBC, which is by origin relational. It is at this level where the differences between the different ODBC drivers become more important and this is one of the reasons, which explains some differences between different ODBC drivers, each specific for a database management system.

Still the use the ODBC standard is of a significant importance, since even if there are differences between the different DBMSs when addressing the object-oriented/object-relational extensions, these differences are minor and does only require minor changes in the development codes. In other words, changing the application's program to access a different database will only require slight changes, which concern the object-oriented/object-relational extension. Therefore, applications can be easily ported from one database to another by switching the ODBC driver and making the required changes instead of rewriting the entire application.

A.2.2 Use of JAVA for Application Programming

Java is the most promising computer language to increase the ability to share software easily across heterogeneous applications of different types. With the appearance of the Internet and the Web, Java fulfills its potential as an object-oriented programming language suitable for supporting networked environment, and becomes the *de facto* standard programming language for Web applications.

The Java object-oriented technology helps in research and development as an important software language for implementing interoperable information management systems, in particular, when used, with its associated Internet technologies (e.g. J2EE⁶, JDBC⁷, and EJB⁸), in web-based applications. It gives users the flexibility and dynamism, and brings a good potential for making legacy systems appear in a more Internet-friendly language. Java applets, for instance, can act as front-ends for accessing the capabilities of remote legacy system, where Internet users can access those legacy systems without having to create mapping code, and without caring about knowledge of the special access mechanisms that are hidden from them. Therefore, the obvious benefit for Java applications developers is that the combination of application and database programming into a single environment means that developers only have to deal with one data model.

There are emerging types of applications, largely driven by Java and the Web, where direct object storage, whether to a relational or an object database, is a clearly superior solution. Java fully supports this feature through the ODMG standard and best suites in environments that need to provide connectivity to a variety of DBMS servers and heterogeneous databases and that require significantly high level of concurrently connected users, where performance and scalability are required.

⁶ J2EE: Java 2 platform, Enterprise Edition

⁷ JDBC is a trademark name, often though to stand for Java Database Connectivity

⁸ EJB: Enterprise Java Beans

A.2.3 Use of XML for Information Exchange

Very rapidly, since its ratification by the World Wide Web Consortium (W3C) in 1998, XML is already becoming the *de facto* standard for data communication and information exchange among distributed organizations and applications. One of the main applications that benefits from XML is thus, the information exchange and data integration among heterogeneous databases. Similar to ODMG and Java standards, the advantages of using XML standard is to reduce the number of wrappers serving the interoperation among heterogeneous and distributed databases and applications.

Regarding the integration approach of **GFI₂S**, presented in this chapter, its architecture can be augmented with the XML standard for data exchange representation. In such architecture, the data transformation to XML format will be performed at different sites of the federation, where the local XML wrapper processes and transforms the data from a specific representation to an XML notation. Meanwhile, the merging of data is a global process, to be performed at the federated layer, and then organizing the data to fit the integrated schema that is defined at the federated layer. This approach allows the unification of data transformation process at the federated level, where a universal module can transform and merge the returned sub-results simply, by using the XML data and the mapping specifications defined for it.

Integrating XML data across applications and databases is of great interest for the database community; efficient techniques for integrating XML data across local- and wide-area networks are an important research focus. However, the consideration of XML in databases and in particular for information integration, raises several challenges for database research. Having XML focusing only on the *syntax* for data representation only partially advances the prospects of their integration. To support information integration at the *semantic* level however, there must be a certain type of agreement among all involved database nodes upon specific DTDs in XML. For instance, if the XML data does not conform to a fixed schema; names and meanings of the used tags are arbitrary and the data is self-describing in XML documents. According to Alon Levy [AL 99], there are several issues that need to be considered to enable such integration. Among these issues we mention a few important ones; some of these issues are already being addressed by current research.

- *Languages for describing the contents and capabilities of XML sources*, which provide the semantic mapping between the data in the source and the relations in the mediated schema. The main challenges involve (1) the restructured data appearing in XML is richer than in relational data, (2) scaling up to a very large number of XML sources must be supported, and (3) exploiting the knowledge conveyed by accompanying DTDs is required.
- *Query reformulation algorithms*, which require the development of algorithms for efficiently reformulating user queries, posed on a mediated schema, to queries that refer to different underlying XML data sources. The known techniques for reformulation from the relational case do not extend easily to languages for querying XML documents.
- *Translation among DTDs*, which provides the proper tools and facilities to translate XML data conforming to one DTD into an XML document conforming to a different DTD, presumably with semantically related content.
- *Obtaining source descriptions*, which develops methods for automatically (semi-automatically) computing source descriptions for newly introduced XML data sources; becomes significant when the number of data sources grows.

Some challenging research work on using XML for information exchange is reported in [SFP 00]. Some advances are also evolving in the areas of information management and data integration [AL 99, MMA 99, BF 01], views definition [Abt 99], scientific data archiving [PWD⁺99], and graphical querying languages [SFP⁺99]. Among the developed systems, which deploy the XML syntax and the XML query engine: the Tukwila data integration system [IHW 01] can be mentioned, which is designed specifically for processing network-bound XML data sources, and Tox [BBM⁺01]: a repository for XML data and metadata, which supports real and virtual XML documents.

Bibliography

- [AAC⁺99] S. Abiteboul, B. Amann, S. Cluet, A. Eyal, L. Mignet, and T. Milo. Active Views for Electronic Commerce, In *Proceedings of the 25th International Conference on Very Large Databases - VLDB'99*, pages 138-149, Edinburgh, Scotland, UK, September 1999.
- [ABB⁺01] H. Afsarmanesh, R. Belleman, A.S.Z. Belloum, A. Benabdelkader, J.F. Jo van den Brand, T.M. Breit, H. Bussemaker, G.B. Eijkel, A. Frenkel, C. Garita, D.L. Groep, A.W. van Halderen, R.M.A. Heeren, Z.W. Hendrikse, L.O. Hertzberger, J. Kaandorp, E.C. Kaletas, V. Klos, P. Sloot, R.D. Vis, A. Visser, and H.H. Yakali. VLAM-G: A Grid-Based Virtual Laboratory. In *Journal of special issue on grid computing*, IOS press Publishers, 2001.
- [ABH 98a] H. Afsarmanesh, A. Benabdelkader, and L.O. Hertzberger. A Flexible Approach to Information Sharing in Water Industries. In *Proceedings of International Conference on Information Technology – CIT 98*, McGraw-Hill Publishers, pages 135-142, Bhubaneswar, India, December 1998.
- [ABH 98b] H. Afsarmanesh, A. Benabdelkader, and L.O. Hertzberger. Cooperative Information Management for Distributed Production Nodes. In *Proceedings of the 10th International IFIP WG Conference On The Globalization of Manufacturing in the Digital Communications Era of the 21st Century: Innovation, Agility, and the Virtual Enterprise - PROLAMAT 98*, Kluwer Academic Publishers, pages 13-27, Trento, Italy, September 9-12, 1998.
- [ABK⁺00] H. Afsarmanesh, A. Benabdelkader, E.C. Kaletas, C. Garita, and L.O. Hertzberger. Towards a Multilayer Architecture for Scientific Virtual Laboratories. In *Proceedings of the 8th International Conference on High Performance Computing and Networking - EuropeHPCN 2000*, Springer, pages 163-176, Amsterdam, The Netherlands, May 2000.
- [Abt 99] S. Abiteboul. On Views and XML, In *Proceedings ACM Symposium on Principles of Database Systems*, pages 1-9, 1999.
- [ACM 00] P. Atzeni, L. Cabibbo, and G. Mecca. Database cooperation: classification and middleware tools. In *Journal of Database Management*, 2000.
- [ADO 01] ActiveX Data Objects (ADO) - Programmer's Guide. © 1998-2001 Microsoft Corporation, all rights reserved. 2001.
- [AE 95] H. Assal and C. Eastman. Engineering Database as a Medium for Translation. In *Proceedings of the International Conference CIB W-78*, Stanford, CA, 1995.

- [AHW⁺98] D.R. Adams, D.M. Hansen, K.G. Walker, and J.D. Gash. Scientific Data Archive at the Environmental Molecular Science Laboratory. In *Proceedings of the Joint Conference on Mass Storage Systems*, College Park, Maryland, March 1998.
- [AKB⁺01] H. Afsarmanesh, E.C. Kaletas, A. Benabdelkader, C. Garita, and L. O. Hertzberger. A Reference Architecture for Scientific Virtual Laboratories. In *Journal of Future Generation Computer Systems. Special issue: High Performance Computing and Networking*. Volume 17, Number 8, pages 999-1008. North-Holland Publishers, June 2001.
- [AL 99] A. Levy. More on Data Management for XML. University of Washington, May 9th, 1999. (to be completed)
- [ASP 01] Active Server Pages Guide. © 2001 Microsoft Corporation, all rights reserved. 2001.
- [Atz 99] P. Atzeni. Databases and the World Wide Web. In *Proceedings of the SOFSEM'99, Theory and Practice of Informatics, Lecture Notes in Computer Science 1725*, Springer-Verlag, pages 147-159, 1999.
- [Atz 98] P. Atzeni. Web Sites Need Models and Schemes. In *Proceedings of the 17th International Conference on Conceptual Modeling- ER '98*, Lecture Notes in Computer Science, Volume 1507, pages 165-167, Singapore, November 1998.
- [ATW⁺94] H. Afsarmanesh, F. Tuijnman, M. Wiedijk, and L.O. Hertzberger. The Implementation Architecture of PEER Federated Object Management System. Technical Report. Department of Computer Systems, University of Amsterdam, The Netherlands, 1994.
- [ATW⁺93] H. Afsarmanesh, F. Tuijnman, M. Wiedijk, and L.O. Hertzberger. Distributed Schema Management in a Cooperation Network of Autonomous Agents. In proceedings of the 4th *IEEE International Conference on Database and Expert Systems Applications - DEXA'93*, Springer, pages 565-576, Prague, Czech Republic, September 1993.
- [AVF⁺92] P.M.G. Apers, C.A. van den Berg, J. Flokstra, P.W.P.J. Grefen, M.L. Kersten, A.N. Wilschut. PRISMA/DB: A Parallel, Main Memory Relational DBMS. In proceedings of the *IEEE Transactions on Knowledge and Data Engineering*. vol. 4, No. 6, pages 541-554, December 1992.
- [AWH 94] H. Afsarmanesh, M. Wiedijk, and L.O. Hertzberger. Flexible and Dynamic Integration of Multiple Information Bases. In *Proceedings of the 5th International IEEE Conference on Database and Expert Systems Applications - DEXA'94*, Springer, pages 744-753, Athens, Greece, September, 1994.
- [AWT⁺94] H. Afsarmanesh, M., Wiedijk, F. Tuijnman, M. Bergman, and P. Trenning. The PEER Information Management Language User Manual. Technical Report *CS-94-14*, Department of Computer Systems, University of Amsterdam, The Netherlands, 1994.
- [BA 00] A. Benabdelkader and H. Afsarmanesh. Enhanced DC Model for Scientific Data Archive. Internal Report, Faculty of Science, Informatics Institute, University of Amsterdam, The Netherlands, March 2000.
- [BA 98a] A. Benabdelkader and H. Afsarmanesh. A Flexible Interoperation Framework for Distributed Multi-Agent Applications. Technical Report *CS-98-08*, Institute of Computer Science, University of Amsterdam, The Netherlands, 1998.

- [BA 98b] A. Benabdelkader and H. Afsarmanesh. Development of the PEER Federated Layer for Modules Integration in WATERNET. Technical Report *CS-98-07*, Institute of Computer Science, University of Amsterdam, The Netherlands, 1998.
- [BAG 98] A. Benabdelkader, H. Afsarmanesh, and C. Garita. Specification of an Object-Oriented Infrastructure for Data Management in Water Industries. Technical Report *CS-98-06*, Institute of Computer Science, University of Amsterdam, The Netherlands, 1998.
- [BAH 00] A. Benabdelkader, H. Afsarmanesh, and L.O. Hertzberger. MegaStore: Advanced Internet-based Electronic Commerce Service for Music Industry. In *Proceedings of 11th IEEE International Conference on Database and Expert Systems Applications - DEXA'2000*, Springer, London - Greenwich, United Kingdom, September 2000.
- [BAH 99] A. Benabdelkader, H. Afsarmanesh, and L.O. Hertzberger. Database Support for Multi-media Information in Web Based Applications. In *Proceedings of International Conference on Computer Technologies - MICCT99*, pages 169-184, Tizi Ouzou, Algeria, June 1999.
- [BAH 99a] A. Benabdelkader, H. Afsarmanesh, and L.O. Hertzberger. The Virtual MegaStore System Implementation. Technical Report *CS-99-05*, Faculty of Science, Research Institute Computer Science, University of Amsterdam, The Netherlands, 1999.
- [BAH 99b] A. Benabdelkader, H. Afsarmanesh, and L.O. Hertzberger. The Virtual MegaStore System Architecture: Analysis and Design. Technical Report *CS-99-04*, Faculty of Science, Research Institute Computer Science, University of Amsterdam, The Netherlands, 1999.
- [BAK⁺00] A. Benabdelkader, H. Afsarmanesh, E. C. Kaletas, and L.O. Hertzberger. Managing Large Scientific multi-media Data Sets. In *Proceedings of Workshop on Advanced Data Storage / Management Techniques for High Performance Computing*, Warrington, United Kingdom, February 2000.
- [Bald 90] R.W. Baldwin. Naming and grouping privileges to simplify security management in large databases. In *Proceedings of IEEE Computer Society Symposium on Research on Security and Privacy*, pages 61-70, Oakland, 1990.
- [BAR⁺01] A. Benabdelkader, H. Afsarmanesh, R. Schut, and L.O. Hertzberger. e-MegaStore concept for current music stores models: Free RecordShop, LuisterPaal, and Sheet Music. Poster in *ICT KennisCongress*, Den Haag, The Netherlands, September 2001.
- [BAS⁺99] M. Bergman, G.D. van Albada, H. Simon, H. Buchner, P.M.A. Sloot, and V. Friedrich. Brain activity and parallel computing. In *Proceedings of the 5th annual conference of the Advanced School for Computing and Imaging - ASCI*, pages 44-45, Delft, the Netherlands, June 1999.
- [BBE 99] A. Bouguettaya, B. Benatallah, and A. Elmagarmid. An Overview of Multi-database Systems: Past and Present. In *Management of Heterogeneous and Autonomous Database Systems*, A. Elmagarmid, M. Rusinkiewicz, and A. Shet, Eds. Morgan Kaufmann Publishers, pages 1-24, San Francisco, California, 1999.
- [BBH⁺99] A. Bouguettaya, B. Benatallah, L. Hendra, J. Beard, K. Smith, and M. Ouzani. World Wide Database - Integrating the Web, CORBA and Databases.

- In *Proceedings of the International Conference ACM - SIGMOD*, ACM Press, Philadelphia, USA, 1999.
- [BBM⁺01] D. Barbosa, A. Barta, A. Mendelzon, G. Mihaila, F. Rizzolo, and P. Rodriguez-Gianolli. ToX: The Toronto XML Engine. In *Proceedings of the International Workshop on Information Integration on the Web*. Rio de Janeiro, 2001.
- [BBO⁺99a] A. Bouguettaya, B. Benatallah, M. Ouzzani, and L. Hendra. WebFINDIT - An Architecture and System for Querying Web Databases, In *IEEE Journal of Internet Computing*, Volume 3, Number 4, July-August 1999.
- [BBO⁺99b] A. Bouguettaya, B. Benatallah, M. Ouzzani, and L. Hendra. Using Java and CORBA for Implementing Internet Databases. In *Proceedings of the International Conference on Data Engineering*, IEEE Society, Sydney, Australia, March 1999.
- [BCG⁺97] M. Baldonado, C.C.K. Chang, L. Gravano, and A. Paepcke. The Stanford Digital Library Metadata Architecture. In *International Journal of Digital Libraries*, Volume 1, Number 2, September 1997.
- [BDH⁺95] P. Buneman, S.B Davidson, K. Hart, and C. Overton. A Data Transformation system for Biological Data Sources. In *Proceedings of the 21st International Conference on Very Large Databases - VLDB'95*, Zurich, Switzerland, 1995.
- [BEM⁺98] C. Beeri, G. Elber, T. Milo, Y. Sagiv, O. Shmueli, N. Tishby, Y. Kogan, D. Konopnicki, P. Mogilevski, and N. Slonim. WebSuite - A tool suite for harnessing Web data. In *Proceedings of the International Workshop on the Web and Databases WebDB'98*. Valencia, Spain, March 1998.
- [Ben 00a] A. Benabdelkader. Universal Data Access Through Standards. Internal Report, Faculty of Science, Informatics Institute, University of Amsterdam, The Netherlands, July 2000.
- [Ben 00b] A. Benabdelkader. Accessing Large Objects within the Matisse Database System. Internal Report, Faculty of Science, Informatics Institute, University of Amsterdam, The Netherlands, May 2000.
- [Ben 95] A. Benabdelkader. Intégration des Bases de Données Géographiques Hétérogènes - cas des Modèles de Terrains. Master Thesis, INSA de Lyon, Université C.B. Lyon 1, Ecole Centrale Lyon, Université De Savoie, Lyon, France, 1995.
- [BF 01] E. Bertino and E. Ferrari. XML and Data Integration. In *IEEE Internet Computing, Special Issue on Personalization and Privacy*. Volume 5, Number 6, pages 75-76, November/December 2001.
- [BFL⁺95] P. Brown, D. Fisher, S. Louis, J.R. McGraw, R. Musick, and R. Troy. The design of a DBMS / MSS interface. NASA EOSDIS project, University of California, Berkeley and Lawrence Livermore National Laboratory, September 1995.
- [BFN 94] R. Busse, P. Fankhauser, and E.J. Neuhold. Federated Schemata in ODMG. In *Proceedings of the Second International East-West Database Workshop*, pages 356-379, Springer, Klagenfurt, Austria, September 1994.
- [BHG⁺01] A. Belloum, Z. Hendrikse, D. Group, E. C. Kaletas, and L.O. Hertzberger, The VL Abstract Machine: a data and process handling system on the grid. In *Proceedings of 9th International Conference on High Performance Computing and Networking - HPCN Europe 2001*, Springer, Amsterdam, The Netherlands, 2001.

- [BKS 98] R.G. Belleman, J.A. Kaandorp, and P.M.A. Sloot: Interactive environments for the exploration of large data sets. In *Proceedings of the 4th annual conference of the Advanced School for Computing and Imaging (ASCI)*, pages 264-268. Lommel, Belgium, June 1998.
- [BS 00] R.G. Belleman and P.M.A. Sloot. The Design of Dynamic Exploration Environments for Computational Steering Simulations. *SIG Users' Conference 2000*. Krakow, Poland, October 2000.
- [CA 99] L. M. Camarinha-Matos and H. Afsarmanesh. The PRODNET Architecture. In *Infrastructures for Virtual Enterprises - Networking Industrial Enterprises*, L. M. Camarinha-Matos and H. Afsarmanesh, Eds., Kluwer Academic Publishers, pages 109-126, 1999.
- [CBB⁺00] R.G.G. Cattell, D. Barry, M. Berler, J. Eastman, D. Jordan, C. Russel, O. Schadow, T. Stanienda, and F. Velez. The Object Data Standard: ODMG 3.0. Morgan Kaufmann Publishers, 2000.
- [CD 97] S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technology. *ACM SIGMOD Record*, Volume 26, Number 1, March 1997.
- [CL 99] L. M. Camarinha-Matos and C. Lima. PRODNET Coordination Module. In *Infrastructures for Virtual Enterprises - Networking Industrial Enterprises*, L. M. Camarinha-Matos and H. Afsarmanesh, Eds., Kluwer Academic Publishers, pages 147-166, 1999.
- [CM 99] L.M. Camarinha-Matos and F. Martinelli. Application of machine learning in water distribution networks assisted by domain experts. *Journal of Intelligent and Robotic Systems*, Vol. 26, Issue 3/4, pp 325-352, Nov. 99, ISSN 0921-0296.
- [CM 97] L.M. Camarinha-Matos and F. Martinelli. Application of Machine Learning in Water Distribution Networks: An Initial Study. In *Workshop on Machine Learning Application in the real world: Methodological Aspects and Implications*, Nashville, USA, July 1997.
- [Cog 00] COGNOS: Intelligence Investment with Cognos EnterpriseServices. May 2000. Copyright © 1989-2000. Cognos Incorporated (<http://support.cognos.com>).
- [Com 00] Comshare: Management, Planning, and Control. Serial Number 040.01.0800 Copyright © 2000 Comshare, Inc. (<http://www.comshare.com>).
- [CP 84] S. Ceri and G. Pelagati. Distributed Databases: Principles and Systems. New York: McGraw-Hill, 1984.
- [CQ 97] G. Cembrano and J. Quevedo. Optimization in water networks. In *Proceedings of the international Conference on Computing and Control for the Water Industry*. September 1997.
- [Daly 01] D. Daly. The Oracle Migration Workbench Helping you migrate from IBM, Microsoft, Informix, Sybase and other databases to Oracle9i. Copyright© 2001 Oracle Corporation. All Rights Reserved.
- [DCMI 99] Dublin Core Metadata Element Set, Version 1.1: Reference Description. Dublin Core Metadata Initiative (DCMI), July 99.
- [DD 99] R. Domenig and K. Dittrich. An Overview and Classification of Mediated Query Systems. *ACM SIGMOD RECORD*, Volume 28, Number 3, September 1999.

- [DHA⁺98] L. Dorst, A. Hoekstra, J.M. van den Akker, J. Breeman, F.C.A. Groen, J. Lagerberg, A. Visser, H. Yakali, and L.O. Hertzberger. Evaluating Automatic Debiting Systems by modeling and simulation of virtual sensors. In *IEEE Instrumentation and Measurement Magazine*. Vol 1, Num 2, pages 18-25, June 1998.
- [EAG⁺01] G.B. Eijkel, H. Afsarmanesh, D. Groep, A. Frenkel, R.M.A. Heeren. Mass Spectrometry in the Amsterdam Virtual Laboratory: development of a high-performance platform for meta-data analysis. In *Proceedings of the 13th Conference on Mass Spectrometry: informatics and mass spectrometry*, Sanibel Island, Florida, USA, January 19 - 22, 2001.
- [ED 01] S. Ellis and C. Drummond. DB2@Warehouse Manager for OS/390® and z/OS™. White Paper, October 2001, © Copyright IBM Corp. 2001. All Rights Reserved.
- [EK 91] F. Eliassen and R. Karlsen. Interoperability and Object Identity. *SIGMOD Record*, Volume 20, number 4, pages 25-29, 1991.
- [EN 00] R. Elmasri and S. B. Navathe, Editors. Fundamentals of Database Systems. Addison-Wesley Publishers, 3rd edition, 2000.
- [FAE⁺01] A. Frenkel, H. Afsarmanesh, G.B. Eijkel, and L.O. Hertzberger. Information Management for Material Science Applications in a Virtual Laboratory. In *Proceedings of the 12th International Conference on Database and Expert Systems Applications - DEXA 2001*, Munich, Germany, September 2001.
- [FAG⁺00] A. Frenkel, H. Afsarmanesh, C. Garita, and L.O. Hertzberger. Information Access Rights in Virtual Enterprises. In *Proceedings of the 2nd IFIP / MASSIVE Working Conference on Infrastructures for Virtual Enterprises, Pro-VE 2000*, Florianopolis, Brazil, December 2000.
- [FGN 98] G. Falquet, J. Guyot, and L. Nerima. Language and tools to specify hypertext views on databases. In *Proceedings of the International Workshop on the Web and Databases WebDB'98*. Valencia, Spain, March 1998.
- [FK 98] I. Foster and C. Kesselman. The Globus Project: A Status Report. In *Proceedings of IPPS/SPDP '98, Heterogeneous Computing Workshop*, pages 4-18, 1998.
- [FLM 98] D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World Wide Web: A Survey. *ACM SIGMOD Record*, Volume 27, Number 3, pages 59-74, September 1998.
- [FN 92] P. Fankhauser and E. Neuhold. Knowledge based integration of heterogeneous databases. Technical report. Technische Hochschule Darmstadt, 1992.
- [Fow 95] J. Fowler. STEP for Data Management Exchange and Sharing. Volume 8, 214 pages. Technical Appraisals Publishers. November 1995.
- [Frt 99] P. Fraternali. Tools and Approaches for Developing Data-Intensive Web Applications: A Survey. *Computing Surveys*, Volume 31, Number 3, pages 227-263, 1999.
- [FS 96] P.M. Fernandez and D. Schneider. The Ins and Outs (and everything in between) of Data Warehousing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, page 541, Montreal, Canada, June 1996.

- [GAH 01] C. Garita, H. Afsarmanesh, and L.O. Hertzberger. The PRODNET Federated Information Management Approach for Virtual Enterprise Support. In *Journal of Intelligent Manufacturing*, 2001.
- [GCL 99] P. Gibon, J.-F. Clavier, and S. Loison. Support for Electronic Data Interchange. In *Infrastructures for Virtual Enterprises - Networking Industrial Enterprises* (L. M. Camarinha-Matos and H. Afsarmanesh, Eds.), Kluwer Academic Publishers, pages 187-208, 1999.
- [Gelb 98] W. M. Gelbart. Databases in Genomic Research. *CBC NOTE in CR library*, Science 282, pages 659-661, October 98.
- [GR 01] C. O. Garita Rodríguez. Federated Information Management for Virtual Enterprises. PhD Thesis, University of Amsterdam, November 2001.
- [GSB 95] G.S. Barton. Directory interchange format: A metadata tool for the NOAA earth system data dictionary. In Ronald B. Melton, D. Michael DeVaney, and James C. French, editors, *The Role of Metadata in Managing Large Environmental Science Datasets*, pages 19-23. Pacific Northwest Laboratory, Richland, WA, June 1995.
- [GUW 02] Database Systems: The Complete Book. H. Garcia-Molina, J. D. Ullman, J. Widom. Prentice-Hall Publishers, 2002.
- [HA 96] D. M. Hansen and D. R. Adams. A Database Approach to Data Archive Management. In *Proceedings of the First IEEE Metadata Conference*, Silver Spring, MD, April, 1996.
- [HBP 94] A. R. Hurson, M. W. Bright, and H. Pakzad. Multidatabase Systems: An Advanced Solution for Global Information Sharing. IEEE Computer Society Press, Alamos, CA, 1994.
- [HDB⁺97] A.G. Hoekstra, L. Dorst, M. Bergman, J. Lagerberg, A. Visser, H. Yakali, F. Groen, and L.O. Hertzberger. Modelling and simulation of automatic debiting systems for electronic collection on motor highways. In *Proceedings of the International Conference Applied Modelling and Simulation - IASTED*. (M.H. Hamzam, Editors), pages 104-108, 1997.
- [Hill 01] D. Hillmann. Using Dublin Core (<http://dublincore.org/documents/usageguide/>). Copyright © 1995-2001 DDCMI All Rights Reserved. April 2001.
- [HM 99] J. Hammer and D. McLeod. Resolution of Representational Diversity. In *Management of Heterogeneous and Autonomous Database Systems*, (A. Elmagarmid, M. Rusinkiewicz, and A. Shet, Editors), pages 91-117, San Francisco, California, Morgan Kaufmann Publishers, 1999.
- [HM 85] D. Heimbigner and D. McLeod. A federated architecture for information management. In *ACM Transaction on Office Information Systems*, Volume 3, Number 3, pages 253-278, July 1985.
- [HR 90] S. Hayne and S. Ram. Multi-user view integration system (MU-VIS): An expert system for view integration. In *Proceedings of the sixth International conference on Data Engineering*. Los Alamitos, CA :IEEE Computer Society Press, February 1990.
- [HSM 01] A Guide to Hierarchical Storage Management (HSM). White Paper, Copyright © 2001 Computer Associates International, Inc. All rights reserved. (http://www.cai.com/products/hsm_netware).

- [HTH⁺99] J. L. Hainaut, Ph. Thiran, Jan-Marc Hick, S. Bodart, and A. Deflorenne. Methodology and Case Tools for the Development of Federated Databases. In *International Journal of Cooperative Information Systems - IJCIS'99*. pages 169-194, 1999.
- [Hum 00] Hummingbird White Paper. SAP R/3 Data Warehousing and Application Integration, 3034-1W Copyright © 2000 Hummingbird Communication Ltd.
- [Hyp 01] Hyperion Solutions Corporation. Business Intelligence Software. © Copyright 1998-2001, All rights reserved (<http://www.hyperion.com/index.cfm>).
- [IHW 01] Z. Ives, A. Halevy, and D. Weld. Integrating Network-Bound XML Data. In *IEEE Data Engineering*, Bulletin 24/2, June 2001.
- [JCF 95] J.C. French. What is metadata? In *The Role of Metadata in Managing Large Environmental Science Datasets*, (Ronald B. Melton, D. Michael DeVaney, and James C. French, editors), pages 3-8. Pacific Northwest Laboratory, Richland, WA, June 1995.
- [JPS⁺88] G. Jacobsen, G. Piatetsky-Shapiro, C. Lafond, M. Rajinikanth, and J. Hernandez. CALIDA: A knowledge-based system for integrating multiple heterogeneous databases. In *Proceedings of the 3rd International Conference on Data and Knowledge Bases*, pages 3-18, June 1988.
- [KAB⁺01] E. C. Kaletas, H. Afsarmanesh, T. M. Breit, and L.O. Hertzberger. EXPRESSIVE - A database for micro-array gene expression studies. Technical Report CS-2001-02, University of Amsterdam, Informatics Institute, The Netherlands, 2001.
- [KAH 01] E. C. Kaletas, H. Afsarmanesh, and L.O. Hertzberger. Virtual Laboratory Experimentation Environment Data Model. Technical report CS-2001-01, University of Amsterdam, Informatics Institute, The Netherlands, 2001.
- [Kar 98] K. Karlapalem. New Aspects of Data Warehousing Environments (keynote talk). In *International Workshop on Data Warehouse Design and OLAP Technology*, in conjunction with DEXA'98. August 98.
- [KCG⁺93] W. Kim, I. Choi, S.K. Gala, and M. Scheevel. On Resolving Schematic Heterogeneity in Multidatabase Systems. In *Journal of Distributed and Parallel Databases*, Volume 1, Number 3, pages 251-279, 1993.
- [KJR 98] K.T. Claypool, J. Jing and E.A. Rundensteiner. OQL_SERF: An ODMG Implementation of the Template-Based Schema Evolution Framework. *Proceedings of IBM Centre for Advanced Studies Conference - CASCON'98*, November 1998.
- [KRS⁺99] A. Klen, R. Rabelo, M. Spinosa, and A. Ferreira. Distributed Business Process Management. In *Infrastructures for Virtual Enterprises - Networking Industrial Enterprises*, L. M. Camarinha-Matos and H. Afsarmanesh, Editors, pages 241-258, Kluwer Academic, 1999.
- [LA 86] W. Litwin and A. Abdellatif. Multidatabase interoperability. In *IEEE Computer*, Volume 19, Number 12, pages 10-18, December 1986.
- [Lit 85] W. Litwin. An overview of the multidatabase system MRSDB. In *Proceedings of the ACM National Conference*, pages 495-504, New York, October 1985.
- [LMR 90] W. Litwin, L.Mark and N. Roussopoulos: Interoperability of Multiple Autonomous Databases. In *ACM Computing Surveys*, Volume 22, Number 3, pages 267-293, September 1990.

- [MHH 97] S. Mukherjea, K. Hirata, and Y. Hara. Towards a multimedia World Wide Web Information Retrieval. In *Proceedings of the 6th International World Wide Web Conference*. Santa Clara, California, USA, April 97.
- [MMA 99] G. Mecca, P. Merialdo, and P. Atzeni. ARANEUS in the Era of XML. In *IEEE Data Engineering Bulletin, Special Issue on XML*, September 1999.
- [MMR 97] T. Mills, K. Moody, and K. Rodden. Providing World Wide Access to Historical Sources. In *Proceedings of the 6th International World Wide Web Conference*. Santa Clara, California, USA, April 97.
- [Mt 01] Matisse: The Object Developer's Database - System and Manuals. Release 5, 11th Edition, November 2001. Copyright©19922001 Fresher Information Corp. All Rights Reserved.
- [NDL+00] Y. Ndiaye, A.W. Diene, W. Litwin, and T. Risch. Scalable Distributed Datastructures for High-Performance Databases. In *Proceedings of the 3rd Workshop on Distributed Data and Structures - WDAS'2000*, L'Aquila, Italy, 2000.
- [NEM+86] S. Navathe, R. El-Masri, and J. Larson. Integrating user views in database design. *IEEE Computer* 19(1):50-62, 1986.
- [NK 97] J.C. Nortel, R. Kazman. Web Query: searching and visualizing the Web through connectivity. In *Proceedings of the 6th International World Wide Web Conference*. Santa Clara, California, USA, April 97.
- [OAB 99] L. Osorio, C. Antunes, and M. Barata. The PRODNET Communication Infrastructure. In *Infrastructures for Virtual Enterprises - Networking Industrial Enterprises*, L. M. Camarinha-Matos and H. Afsarmanesh, Editors, Kluwer Academic Publishers, pages 167-186, 1999.
- [OV 99] M. Tamer Ozsu and P. Valduriez. Principals of Distributed Database Systems. Prentice-Hall Publishers, 2nd edition, 1999.
- [PH 98] A. J. H. Peddemors and L.O. Hertzberger. A High Performance Distributed Database System for Enhanced Internet Services. In *Proceeding of the 6th International Conference on High Performance Computing and Networking - HPCN'98*, Amsterdam, the Netherlands, April 98.
- [PP 00] T. Priebe and G. Pernul. Towards OLAP Security Design - Survey and Research Issues. In *Proceedings of the 3rd ACM International Workshop on Data Warehousing and OLAP - DOLAP 2000*, pages 33-40, McLean, VA, USA, November 2000.
- [PWD+99] M. Papiani, J.L. Wason, A.N. Dunlop, and D.A.A. Nicole. Distributed Scientific Data Archive Using the Web, XML and SQL/MED. *ACM SIGMOD Record*, Volume 28, Number 3, pages 56-62, September 1999.
- [RA Inc] RealNetworks, Inc. (<http://www.real.com>).
- [Rad 96] E. Radeke, Extending ODMG for Federated Database Systems, In *Proceeding of the 7th International Conference on Database and Expert Systems - DEXA'96*, Lecture Notes in Computer Science 1134, Springer Verlag, September 1996.
- [REM+89] M. Rusinkiewicz, R. El-Masri, B. Czejdo, D. Georgakopoulos, G. Karabatis, A. Jamoussi, K. Loa, and Y. Li. Query processing in a heterogeneous multi-database environment. In *Proceedings of the 1st Annual Symposium on Parallel and Distributed Processing*, 1989.

- [RK 97] E. Radeke and Kruschinski. Nolte OpenDM/Web: WWW Access to Heterogeneous Database Systems. In *Proceedings of the International Symposium on Global Engineering Networking*, Antwerp, Belgium, April 1997.
- [RKB 01] M. Roantree, J.B. Kennedy, and P.J. Barclay. Integrating View Schemata Using an Extended Object Definition Language. In *Proceedings of the 9th International Conference on Cooperative Information Systems - CoopIS 2001*. Trento, Italy, September 5-7, 2001.. Springer-Verlag Lecture Notes in Computer Science (LNCS) Volume 2172, 2001.
- [RPR⁺94] M.P. Reddy, B.E. Prasad, P.G. Reddy, and A. Gupta. A methodology for Integration of Heterogeneous Databases. In *IEEE transactions on Knowledge and data Engineering*, Volume 6, Number 6, December 1994.
- [SBD⁺83] P.G. Selinger, E. Bertino, D. Daniels, L. Haas, B.G. Lindsay, G. Lohman, Y. Masunaga, C. Mohan, P. Ng, P. Wilms, and R. Yost. The Impact of Site Autonomy on R*: A Distributed Relational DBMS. Chapter in *Role and Structure* (C. 151-176). Cambridge University Press, London, 1983.
- [SC 96] R.S. Sandhu and E.J. Coyne. Role-based access control models. In *IEEE Computer*, pages 38-47, February 1996.
- [Sch 99] A. Schreiber. STEP Support for Virtual Enterprises. In *Infrastructures for Virtual Enterprises - Networking Industrial Enterprises*, L. M. Camarinha-Matos and H. Afsarmanesh, Editors, Kluwer Academic Publishers, pages 209-218, 1999.
- [SF 94] R.S. Sandhu and H.L. Feinstein. A tree-tier architecture for role-based access control. In *Proceedings of the 17th NIST-NCS National Computer Security Conference*, pages 11-14, Baltimore, 1994.
- [SFP 00] S. Ceri, P. Fraternali, and S. Paraboschi. XML: Current Developments and Future Challenges for the Database Community. In *Proceedings of the 7th International Conference on Extending Database Technology - EDBT 2000*, pages 3-17, Konstanz, Germany, March 2000.
- [SFP⁺99] S. Ceri, S. Comai, P. Fraternali, S. Paraboschi, L. Tanca, and E. Damiani. XML-GL: A Graphical Language for Querying and Restructuring XML Documents. In *Proceedings of the 7th Italian Conference on Database Systems - SEBD 1999*, pages 151-165, Como, Italy, June 1999.
- [Sho 97] A. Shoshani. OLAP and Statistical Databases: Similarities and Differences. In *Proceedings of the ACM Symposium on Principles of Database Systems - PODS '97*, pages 185-196, 1997.
- [SK 93] A. Sheth and V. Kashyap. So far (schematically), yet so near (semantically). In *Proceedings of the IFIP TC2/WG2.6 Conference on semantics of Interoperable Database Systems, DS-5*. Amsterdam: North-Holland, November 1993.
- [SKH⁺99] P.M.A. Sloot, J.A. Kaandorp, A.G. Hoekstra and B.J. Overeinder. Distributed Simulation with Cellular Automata: Architecture and Applications. In *Proceedings of the 26th Conference on Current Trends in Theory and Practice of Informatics - SOFSEM'99*. Lecture Notes in Computer Science 1725, pages 203-249, Milovy, Czech Republic, November 1999.
- [SL 90] A. Sheth and J. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, Volume 22, pages 183-236, 1990.

- [SLC⁺88] A. Sheth, J. Larson, A. Cornelio, and S. B. Navathe. A tool for integrating conceptual schemata and user views. In Proceedings of the fourth International Conference on Data Engineering, pages 176-183. Los Alamitos, CA :IEEE Computer Society Press, February 1988.
- [SP 94] S. Spaccapietra, C. Parent. View Integration: A Step Forward in solving structural Conflicts, *IEEE transactions on Knowledge and data Engineering*, Volume 6, Number 2, April 1994.
- [SS 95] S. Sarawagi. Database systems for efficient access to tertiary memory. In *Fourteenth IEEE Symposium on Mass Storage Systems*, pages 120-126, Monterey, CA, September 1995. IEEE Computer Society Technical Committee on Mass Storage Systems, IEEE Computer Society Press.
- [SS 94] R.S. Sandhu and P. Samrati. Access control: principle and practice. In *IEEE Communication*, pages 40-48, September 1994.
- [SSG⁺91] A. Savasere, A. Sheth, S. Navathe, and H. Marcus. On applying classification to schema integration. In proceedings of IMS'91. The first International Workshop on Interoperability in multidatabase system, pages 258-261, 1991.
- [Syb 99] N. Ward. Industry Warehouse Studio: A Technical Overview. White Paper, copyright © 1999 Sybase, Inc. November 1999.
- [SYE⁺90] P. Scheurermann, C. Yu, A. Elmagarmid, H. Garcia-Molina, F. Manola, D. McLeod, A. Rosental, and M. Templeton. Report on the workshop on heterogeneous database systems. In *ACM SIGMOD Record*, volume 19, pages 23-31, New York, December 1990.
- [TA 93] F. Tuijnman and H. Afsarmanesh. Management of Shared Data in Federated Cooperative PEER Environment. In *International Journal of Intelligent and Cooperative Systems (IJICIS)*, Volume 2, Number 4, pages 451-473. 1993
- [TB 00] T. Bollinger. A Guide to Understanding Emerging Interoperability Technologies. © 2000 The MITRE Corporation, Washington C3 Center, McLean, Virginia, July 2000.
- [TBD⁺87] M. Templeton, D. Brill, S. K. Dao, E. Lund, P. Ward, A. L. P. Chen, and R. MacGregor. Mermaid: A front-end to distributed heterogeneous databases. In *Proceedings of the IEEE: Special Issue on Distributed Database Systems*, Volume 75, Number 5, pages 695-708, May 1987.
- [TC 97] Z. Tari, H. Chan. A Role-based Access Control for Intranet Security in IEEE Internet Computing, Volume 1, Number 5, pages 24-34, 1997.
- [THB⁺98] Ph. Thiran, J.-L. Hainaut, S. Bodart, A. Deflorenne, and Jan-Marc Hick. Interoperation of Independent, Heterogeneous and Distributed Databases. Methodology and CASE Support: the InterDB Approach. CoopIS 1998, pages 54-63, New York, USA, August 1998.
- [THH 99] Ph. Thiran, J.-M. Hick, and J.-L. Hainaut. Generation of Conceptual Wrappers for Legacy Databases. In *proceeding of the 10th International Conference and Workshop on Database and Expert Systems - DEXA'99*, Pages 678-687, Florence, Italy, September 1999.
- [Tho 91] D.J. Thomsen. Role-based application design and enforcement. In Database Security IV: Status and Prospects, S. Jajodia and C.E. Landwehr (eds), North-Holland, 1991, pages 151-168.

- [TM 96] K.L. Timimi and J. MacKrell. STEP: Towards Open Systems (STEP fundamentals & Business Benefits). Volume 8, 131 pages. CIMdata Publishers, October 1996.
- [UML 98] H.-E. Eriksson and M. Penker. *UML Toolkit*. Wiley Computer Publishing, 1998.
- [URB 97] B. Ulanicki, J.P. Rance, P.L.M. Bounds. A systematic approach to Information integration in water company. *International Conference on Computing and Control for the Water Industry*.1997
- [VH 93] V. Ventrone and S. Heiler. A practical approach for dealing with semantic heterogeneity in federated database systems. Technical report. The MITRE Corporation, Octobre 1993.
- [Vin 97] S. Vinoski. CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments, *IEEE Communications Magazine*, Volume 35, Number 2, February 1997.
- [VS 99] P. Vassiliadis, T. Sellis. A Survey on Logical Models for OLAP Databases. *ACM SIGMOD Record*, Volume 28, Number 4, pages 64-69, December 1999.
- [VWH 00] A. Visser, A.J. van der Wees, and L.O. Hertzberger. Discrete Event Modeling Methodology for Intelligent Transport Systems. In *Proceedings of the World congress on Intelligent Transport Systems*, Torino, Italy, November 2000.
- [WA 94] M. Wiedijk and H. Afsarmanesh. The PEER User Interface Tools Manual. Technical Report *CS-94-15*, Department of Computer Systems, University of Amsterdam, The Netherlands, 1994
- [Wang 97] C.G. Wang. Object-oriented modeling for the operational control of water distribution system. In *Proceedings of the International Conference on Computing and Control for the Water Industry*. September 1997.
- [WAP 00] Wireless Application Protocol. White Paper, Wireless Internet Today, © 2000 The WAPForum, All rights reserved, June 2000.
- [WB 97] M.C. Wu and A.P. Buchmann. Research Issues in Data Warehousing, In *Datenbanksysteme in Buro, Technik und Wissenschaft BTW'97*, pages 61-82, Ulm, March 1997.
- [WKL⁺98] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. Dublin Core Metadata for Resource Discovery. IETF #2413. The Internet Society, September 1998.
- [WMP 98] R. Williams, R. Moore, and J.C.T. Pool, Workshop on Interfaces to scientific data Archives. Technical Report CACR-160, Pasadena, California, March 1998.
- [ZK 96] A. Zisman and J. Kramer. An architecture to support interoperability of autonomous database systems. In *Proceedings of the 2nd International Baltic Workshop on DB and IS*, Tallin-Estonia, June 1996.

Samenvatting^{*}

De Informatieintegratie tussen Heterogene en Autonome Computertoepassingen

Een grote verscheidenheid aan gedistribueerde computertoepassingen ontstaan tegenwoordig op diverse gebieden. Deze toepassingen gebruiken hierbij verschillende database systemen voor het beheren van hun informatie. De bron van deze verscheidenheid moet gezocht worden in de specifieke eisen aan de informatiebeheer en de specifieke doelen die gezet zijn voor deze toepassingen.

Bestaande toepassingen verschillen van elkaar in hun intrinsieke kenmerken en hun vereiste eigenschappen. Bij intrinsieke kenmerken moet men bijvoorbeeld denken aan hun architectuur (gedistribueerd of gecentraliseerd), omvang, complexiteit, en de soort informatie die ze kunnen afhandelen. Hun eigenschappen hangen daarentegen af van de globale functionaliteit die ze moeten bieden en de mogelijkheden tot samenwerken met andere omgevingen. De huidige Database Management Systemen (DBMSs) zijn hoogstens gekozen om te voldoen aan de kenmerken en eigenschappen van één toepassing. Echter, de huidige generatie DBMSs missen de mogelijkheid om efficiënt toegepast te worden voor alle mogelijke soorten toepassingen. Sommige DBMSs zijn beter in kleine toepassingen, andere zijn juist gespecialiseerd in complexe omgevingen en zijn dan ook gericht op, bijvoorbeeld, multimedia of grote datasets. Kortom, *elke poging om echt verschillende computertoepassingen te dwingen tot het gebruik van hetzelfde database systeem voor het complete beheer van al hun informatie is onrealistisch*. Zelfs in een homogene omgeving, voor sommige complexe toepassingen, kan het gebruik van meerdere database systemen niet worden voorkomen.

Verder moet men voor de huidige organisaties, hetgeen wel duidelijk is geworden uit de drie praktijkbeschrijvingen in dit proefschrift, er vanuit gaan dat zowel voor nieuwe als reeds bestaande toepassingen men toegang moet houden tot de data verschillende oudere databases op verscheidene locaties. Daarom is het voor complexe organisaties de mogelijkheden tot samenwerken en integratie mechanismen tussen heterogene en autonome database systemen *het* criterium waar op geselecteerd wordt. In de praktijk hebben we ook geleerd dat juist het bieden van deze interoperabiliteit en informatieintegratie tussen gedistribueerde systemen, met behulp van database standaarden en opkomende Internet technieken, een van de meest uitdagende taken is binnen het kade van integratie van heterogene informatie van autonome omgevingen.

Om deze nieuwe wensen van geavanceerde en complexe organisaties met betrekking tot informatiebeheer te kunnen vervullen, moet een sterk informatieintegratie systeem worden ontworpen en ontwikkeld.

^{*}Vertaling door Arnoud Visser, Philip Jonkergouw, en Zeger Hendrikse

Daartoe is in dit proefschrift een aanvang genomen in het ontwerpen en gedeeltelijk ontwikkelen van een Generiek en Flexibel Informatieintegratie Systeem (**GFI₂S**). Het ontwerp van **GFI₂S** is gebaseerd op het bestuderen, evalueren en valideren van de methoden en technieken beschreven in hoofdstuk 2, en gemotiveerd door de ervaring opgedaan met de ontwikkelingen voor verscheidene onderzoeksprojecten zoals benoemd in hoofdstuk 3, 4 en 5 van dit proefschrift. De *flexibiliteit* van **GFI₂S** schuilt in het kenmerk dat met minimale inspanning een systeem resp. aan een federatie toe te voegen is of uit een federatie te verwijderen is, hetgeen ondersteund wordt door het gebruik van een specifieke twee componenten architectuur. Het *generieke* karakter wordt gewaarborgd door de toepassing van database standaarden, opkomende Internet technieken en ‘middleware’ oplossingen.

Het werk beschreven in hoofdstuk 6 illustreert de architectuur componenten van **GFI₂S** die de integratie van verschillende datasoorten van heterogene toepassingen ondersteunen. De architectuur van **GFI₂S** is gebaseerd op twee componenten: ten eerste een ‘Local Adaption Layer’ (LAL) die de toegang tot lokale databases vergemakkelijkt, en, ten tweede, een ‘Node Federation Layer’ (NFL) die verwijzingen levert naar de informatie en toepassingen buiten deze computer, en ondersteuning biedt voor informatie uitwisseling. De twee componenten architectuur van **GFI₂S** ondersteunt een grote verscheidenheid aan bestaande toepassingen met efficiënte methoden voor het samenvoegen en samen laten werken van hun informatiebronnen, met behoud van de **heterogeniteit**, de **distributie**, en **autonomie** van de bronnen.

- **Heterogeniteit** staat voor het feit dat iedere database zijn eigen DBMS kan gebruiken, en dat de data representatie heterogeen is in termen van structuren en semantiek voor elke omgeving.
- **Distributie** staat voor het opslaan en verwerken van informatie van verscheidene bronnen, gesitueerd op verschillende computers.
- **Autonomie** staat voor het feit dat iedere database in de federatie onafhankelijk is. In de meeste gevallen bestond de database al voor de federatie gevormd werd, en heeft het zijn eigen gebruikersgemeenschap en administratieve procedures.

De onderscheidende eigenschappen van de **GFI₂S** methode zijn samen te vatten in: (a) de **specifieke combinatie** van database standaarden en Internet ‘middleware’ met fundamentele onderzoeksrichtingen, en (b) de **manier waarin deze toegepast en gekoppeld** binnen de specifieke componenten van **GFI₂S**. Deze twee overwegingen maken de **GFI₂S** methode te onderscheiden van alle andere bestaande federatie / integratie methoden, en positioneren **GFI₂S** als een *generieke oplossing* gebaseerd op een *flexibele architectuur*, en een *open faciliteit* voor de samenvoeging en samenwerking tussen heterogene, gedistribueerde en autonome omgevingen.

Abstract

Information Integration among Heterogeneous and Autonomous Applications

A wide variety of distributed applications are nowadays emerging in diverse domains. These applications deploy various database systems for the management of their information, in which the diversity stems from the specific information management requirements and the objectives targeted by these applications.

Existing applications differ in their main characteristics and required features. On one hand, they differ in their distributed/centralized architecture, their size, complexity, and the type of data they handle. On the other hand, their requirements depend on the global functionalities that they need to provide and on the required level of interoperation with other sites. The used database management systems (DBMSs) are at best chosen to meet the specific characteristics and requirements of every application environment. However, currently available DBMSs lack the possibility to be efficiently used for all types of applications. Some DBMSs better suit smaller applications, while others are more dedicated to complex environments and focus on the management of, for example, multimedia information and large data sets. Thus, *any attempt in the direction of forcing different applications to use the same database system for the management of all their information services is unrealistic*. Even within the same environment, in certain complex applications, the use of more than one DBMS cannot be avoided.

Furthermore, from the application cases described in the thesis, it is clear that in today's organizations, new and existing applications require access to data stored in several pre-existing databases detained at several local and remote sites. Therefore, a main criterion required by most complex organizations, is the provision of collaboration possibilities and information integration mechanisms among distributed, heterogeneous, and autonomous systems. We also learned from the development of the application cases that providing interoperability and information integration among distributed systems, via the deployment of database standards and emerging Internet technologies, is one of the most challenging approaches in the area of integrating heterogeneous information from autonomous sites.

In Order to satisfy the new information management requirements of advanced and complex organizations, a strong information integration system must be designed and developed, serving the need for information integration and interoperation among these organizations.

In this context, the work described in this thesis focuses on the design and partial development of a Generic and Flexible Information Integration System (**GFI₂S**). The design of **GFI₂S** is based on the investigation, evaluation, and validation of the methodologies and approaches discussed in chapter 2; and motivated by the expertise gained within the devel-

opment of the various R&D projects addressed in chapters 3, 4, and 5 of the dissertation. *Flexibility* of **GFI₂S** resides in its ability to add/remove new system to/from the federation with involvement of minimum effort. Flexibility in **GFI₂S** is supported through the use of the specific two-component architecture, while, its *genericness* is achieved through the deployment of database standards, emerging Internet technologies, and middleware solutions.

The work described in chapter 6 illustrates the architectural components of **GFI₂S** that support the integration of different types of data from heterogeneous applications. The architecture of **GFI₂S** is composed of two main components of: (1) Local Adaptation Layer (LAL) that facilitates the access to the underlying databases in the node, and (2) Node Federation Layer (NFL) that provides links to the information and applications outside the node and supports the information sharing and interoperation. This two-component architecture of **GFI₂S** supports a wide variety of existing applications with efficient means for their interconnection and interoperation, while preserving their *heterogeneity*, *distribution*, and full *autonomy*.

- **Heterogeneity** refers to the fact that each database may apply its own distinct DBMS, and data representation is heterogeneous in terms of structures and semantics at every site.
- **Distribution** refers to the storage and processing of information from distributed data sources, located on different host computers.
- **Autonomy** refers to the fact that each database within the federation community is an independent database system. Typically, a local database is pre-existing to the creation of a cooperation network and has its own administration policies, and users community.

The distinctive features of the **GFI₂S** integration approach resides in: (a) the *specific combination* of database standards and Internet middleware with the fundamental research approaches, and (b) the *way in which they are deployed and inter-linked* within the specific components of the **GFI₂S**. These two considerations make the **GFI₂S** approach distinct from all other existing federated/integrated approaches, and introduce **GFI₂S** as a *generic solution* providing a *flexible architecture*, and an *open facility* for integration/interoperation among heterogeneous, distributed, and autonomous sites.

Résumé

L'intégration de l'Information entre des Applications Hétérogènes et Autonomes

Une grande variété d'applications réparties émergent de nos jours dans des domaines divers. Ces applications déploient divers systèmes de base de données pour la gestion de leur information, dont la diversité provient des contraintes spécifiques de la gestion de l'information et des objectifs à atteindre.

Certaines applications diffèrent dans leurs caractéristiques principales et des fonctionnalités exigées. Entre autre, ces applications diffèrent dans leur architecture (distribuée/centralisée), leur taille, leur complexité, et le type de données qu'elles manipulent. Leurs besoins sont définis par les fonctionnalités globales qu'elles doivent fournir et du niveau exigé d'interopération avec d'autres sites géographiquement distribués. Les Systèmes de Gestion de Base de Données utilisés (SGBDs) sont choisis pour répondre aux caractéristiques et aux contraintes spécifiques de chaque application. Malheureusement, les SGBDs disponibles actuellement n'offrent pas la possibilité de les adapter efficacement pour tout type d'applications. Certains SGBDs s'adaptent mieux aux petites applications, alors que d'autres ont été spécialement conçus pour les applications complexes ; et par conséquent sont plus adaptés à la gestion de l'information multimedia et aux données de grandes tailles. Ainsi, *toute tentative dans la direction de contraindre différentes applications à utiliser le même système de gestion de base de données pour gérer tous leurs services d'information est peu réaliste*. Même dans le cadre d'un même environnement, il arrive souvent que l'on soit amené à utiliser plusieurs systèmes de gestion de bases de données.

Il est clair d'après les études faites sur les applications, décrites dans la présente thèse, que les nouvelles applications exigent l'accès à des données stockées dans plusieurs bases de données préexistantes, détenues à plusieurs emplacements géographiquement distribués et distantes. Par conséquent, un des critères principaux exigés par la plupart des systèmes complexes, est d'avoir la possibilité de la collaboration et de fournir des mécanismes d'intégration de l'information entre les systèmes répartis, hétérogènes, et autonomes. Il s'est avéré d'après les études faites sur des applications réelles que le développement des systèmes permettant l'interopérabilité et l'intégration des données entre des systèmes répartis et autonomes est loin d'être trivial. Nous avons également appris du développement de ces applications que fournir l'intégration de l'information et l'interopérabilité entre les systèmes répartis, par l'intermédiaire du déploiement des bases de données standards et des nouvelles technologies d'Internet, est l'une des approches les plus prometteuses dans le domaine d'intégration de l'information hétérogènes entre des sites autonomes. Afin de répondre aux nouvelles contraintes de gestion de l'information dans des systèmes avancés et complexes, un système

robuste d'intégration de l'information doit être conçu et développé, servant les besoins d'intégration de l'information et d'interopération entre ces organismes.

Dans ce contexte, le travail décrit dans cette thèse est dédié à la conception et au développement d'un Système Générique et Flexible d'Intégration de l'Information. La conception de **GFI₂S** est basée sur la recherche, l'évaluation, et la validation des méthodologies et approches discutées dans le chapitre 2. En plus, le développement de **GFI₂S** est motivé par l'expertise acquise lors du développement des divers projets de R&D adressés en chapitres 3, 4, et 5 de la présente thèse. La flexibilité de **GFI₂S** réside dans sa capacité à ajouter/enlever un nouveau système à/de la fédération avec le minimum d'effort. La flexibilité dans **GFI₂S** est obtenue grâce à l'utilisation de l'architecture spécifique de deux-composants, alors que, sa Généricité est assurée par le déploiement des bases de données standards et des nouvelles technologies de l'Internet.

Le travail décrit dans le chapitre 6 illustre les composants du système **GFI₂S** qui permettent l'intégration de différents types de données entre des applications hétérogènes. L'architecture de **GFI₂S** est constituée de deux composants principaux : (1) la couche locale d'adaptation qui facilite l'accès aux sources de données dans les nœuds locaux, et (2) la couche de fédération de nœuds qui fournit des liens à l'information et aux applications externes et supporte le partage d'information et d'interopération. L'architecture de **GFI₂S** permet l'intégration d'une grande variété d'applications existantes avec des moyens efficaces, tout en préservant leur hétérogénéité, distribution, et pleine autonomie.

- l'hétérogénéité vient du fait que chaque application peut appliquer son propre système de gestion de bases de données, et sa propre représentation de données hétérogènes en terme de structures et de sémantique.
- la distribution est liée au stockage et au traitement d'information sur des sources de données distribués, situées sur différents sites géographiquement distribués.
- l'autonomie est liée au fait que chaque base de données au niveau de chaque nœud de la fédération est un système indépendant. Typiquement, une base de données locale est préexistante à la création d'un réseau de coopération et possédant sa propre politique d'administration, et ses groupes d'utilisateurs.

Les fonctionnalités principales de l'approche d'intégration de l'information de **GFI₂S** résident dans: (a) la combinaison spécifique des bases de données standards et des nouvelles technologies d'Internet avec les approches fondamentales de recherches, et (b) le moyen par lequel elles sont déployés et liés aux éléments spécifiques du **GFI₂S**. Ces deux fonctionnalités rendent l'approche de **GFI₂S** distincte de toutes autres approches existantes de fédération/intégration, et présentent **GFI₂S** comme une *solution générique* fournissant une *architecture flexible et évolutive* pour l'intégration de l'information et l'interopération entre des sites hétérogènes, distribués et autonomes.