1

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

***Standardization Issues in
Distributed Database Management***

by

Bruce L. Catley

(McGill University, B.Sc.)

**A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of**

Master of Science, Information and Systems Science

**Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada**

April 1990

**© copyright
1990, Bruce Catley**

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-59064-5

The undersigned recommend to the
Faculty of Graduate Studies and Research
acceptance of the thesis
"Standardization Issues in Distributed Database Management"
submitted by Bruce L. Catley, B.Sc.
in partial fulfilment of the requirements for
the degree of Master of Science.

Prof. M. Niktash
Thesis Co-Supervisor

Dean J.S. Riordon,
Thesis Co-Supervisor

Prof. S. Mahmoud,
Chairman, Department of Systems
and Computer Engineering

Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University
April 1990

ABSTRACT

This thesis examines the applicability of international data management and interprocess communication standards in addressing the problems faced by vendors building both homogeneous and heterogeneous distributed database management systems.

The objectives of information technology standardization are discussed, followed by a review of the requirements to be addressed by distributed database management systems. Existing and developing standards with potential impact on the standardization of DDBMSs are examined. An architectural framework permitting the integration of various data management and interprocess communication standards is proposed. The "design" of the standards components necessary to facilitate heterogeneous distributed database processing suggests the need for modifications to the manner in which single-site data management facilities are being specified and implemented. Mechanisms for supporting various distributed database distribution alternatives are discussed.

ACKNOWLEDGEMENTS

The author wishes to thank his wife and children, who have offered their constant support, encouragement and tolerance.

The author also wishes to express thanks to his thesis supervisors, Professor Morteza Niktash and Dean J.S. Riordon, for their assistance and guidance during the process of this research.

TABLE OF CONTENTS

ACCEPTANCE SHEET	ii
ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
LIST OF FIGURES	viii
GLOSSARY OF TERMS AND ACRONYMS	ix
CHAPTER 1. INTRODUCTION AND PROBLEM DEFINITION	1
1.1 Introduction	1
1.2 Problem Definition	2
1.3 Thesis Outline	3
CHAPTER 2. IDENTIFICATION OF THE PRIMARY GOALS OF STANDARDIZATION	4
2.1 Introduction	4
2.2 Application Portability	6
2.3 Interworking.	8
2.4 Advantages and Disadvantages of Standardization ...	8
2.5 Reference Models	11
2.5.1 OSI Reference Model	11
2.5.2 Need for a Reference Model for Data Management in a Distributed Environment ...	14
CHAPTER 3. A SURVEY OF REQUIREMENTS FOR DISTRIBUTED DATABASE MANAGEMENT SYSTEMS	19
3.1 Introduction	19
3.2 Database Access Requirements	20
3.2.1 Data Accessibility Requirements	20
3.2.2 Communication Requirements	20
3.3 Requirement for Object-Oriented Facilities	21
3.4 Data Dictionary System Requirements	22
3.5 Requirements for Security	25
3.5.1 Access Control	25
3.5.2 Authentication	25

3.6	Requirements for Transaction Management	25
3.7	User Interface and Presentation Requirements	30
3.8	Requirements for Distribution Support	31
3.8.1	Overview	31
3.8.2	Distribution Alternatives	35
3.8.3	Replication Alternatives	35
3.8.4	Requirements for Data Location Information ..	37
3.8.5	Administrative Requirements for Supporting Distribution	37
3.8.6	Federated Databases	38
3.8.7	Performance Considerations	39
3.9	Heterogeneity	42
3.10	Requirements for Export/Import	42
3.11	Requirements for Support of Additional Data Types ...	43
CHAPTER 4. REVIEW OF RELEVANT STANDARDS POTENTIALLY USEFUL IN DDBMS STANDARDIZATION		45
4.1	Introduction	45
4.2	Database Language SQL	45
4.3	Database Language NDL	50
4.4	Information Resource Dictionary System	51
4.5	Remote Database Access	56
4.6	Directory Services	62
4.7	File Transfer, Access and Management	65
4.8	OSI Management	67
4.9	Abstract Service Definition Conventions	70
4.10	Commitment, Concurrency and Recovery	72
4.11	Language Independent Data Types	74
4.12	Abstract Syntax Notation One	76
4.13	Reference Model of Data Management	78
CHAPTER 5. DESIGN CONSIDERATIONS FOR STANDARDIZED COMPONENTS OF DISTRIBUTED DATABASE MANAGEMENT SYSTEMS		79
5.1	Introduction	79
5.2	Consideration of Database Access Requirements	80
5.2.1	Consideration of Programming Language versus Distributed Access Requirements	80
5.2.2	Consideration of Communication Requirements	83
5.2.3	Provision of Database Access Facilities	87
5.2.3.1	Data Modelling Facility Used	87
5.2.3.2	The External Schema	88

5.3	Consideration of Requirements for Object-Oriented Facilities	91
5.4	Consideration of Data Dictionary System Requirements	93
5.4.1	Dictionary Data versus Directory Data	100
5.5	Consideration of Requirements for Security	102
5.6	Consideration of Requirements for Transaction Management	104
5.7	Consideration of User Interface Requirements	105
5.8	Consideration of Requirements for Distribution Support	107
5.9	Consideration of Requirement for Heterogeneity	109
5.10	Consideration of Requirements for Export/Import	110
5.11	Consideration of Requirements for Support of Additional Data Types	111
CHAPTER 6.	PROPOSED ARCHITECTURE FOR DISTRIBUTED DATA MANAGEMENT SYSTEMS	114
6.1	Requirement for a Reference Model	114
6.2	Description of Proposed Architecture	119
6.2.1	Database Controller	120
6.2.2	Distribution Controller	124
6.2.3	Global Data Directory Processor	126
6.2.4	User Application Processor	127
6.2.5	Remote Database Access	128
6.2.6	Access Control Processor	129
6.2.7	Features of the Proposed Architecture	129
6.3	Use of the Architecture for Information Resource Dictionary Access	131
6.4	Examples of the Use of the Architecture in Handling Sample Requests	134
6.4.1	Retrieval of Application Level Data	134
6.4.2	Retrieval of System Documentation Metadata	137
CHAPTER 7.	THESIS CONTRIBUTIONS AND FUTURE STUDIES	140
REFERENCES		145

LIST OF FIGURES

Figure	Page
2.1 Gateway approach to interconnection	13
2.2 OSI approach to interconnection	13
4.1 SQL standard architecture	48
4.2 IRDS processor interfaces	52
4.3 IRDS data levels	55
4.4 The RDA application service	59
4.5 Execution of an SQL query using RDA	62
4.6 Architecture of the OSI FTAM Service	66
4.7 An atomic action tree	73
5.1 Components of an information processing system	94
6.1 Proposed architecture for distributed data management standardization	118
6.2 Proposed architecture for Information Resource Dictionary access	132

Glossary of Terms and Acronyms

Abstract Syntax: A set of formal rules for describing the structure of objects within a universe of discourse, without regard to the details of how these objects are represented in a physical medium (within the storage of a computer, on paper, on a transmission medium, etc.). In particular, an abstract syntax will often disregard the order in which the components of an object are to appear, and will omit symbols whose sole purpose is to act as separators between components.

Application Association: A cooperative relationship between two application entity invocations for the purpose of communication of information and coordination of their joint operation. This relationship is formed by the exchange of application protocol control information using the service provided by the presentation layer.

Application Context: A set of rules shared in common by two application entity invocations governing their behaviour in a given application association in order to enable their cooperative operation. An application context is a shared conceptual schema for the universe of discourse for communication.

Application Database: A collection of data instances intended for manipulation in the environment of an information system.

Application Entity Invocation: Performs the functions of an application entity to fulfil a specific set of communication requirements of an application process invocation.

Application Entity: An entity within the application layer.

Application Layer: The uppermost layer within the OSI Basic Reference Model, directly providing facilities for use by application processes.

Application Process Invocation: An instance of an application process in a real open system to perform its functions for a specific occasion of information processing.

Application Process: An element within a computer system which performs the information processing for a particular application.

Application Schema: A description of the data instances which comprise the application database.

Application Service Element: A set of functions that provide a capability for the interworking of application entities for a specific purpose on a single application association.

Application: A particular activity or group of activities associated with some function within an enterprise.

ASE: application service element.

Association: A relationship between application entities for the purpose of communication between them.

Atomic Action: An action which is either carried out completely or not at all.

CCITT: Comité Consultatif International Télégraphique et Téléphonique - a United Nations organization through which the national telecommunications bodies coordinate their activities.

CCR: Commitment, Concurrency and Recovery.

Centralized Database: A database in which both its data and metadata is stored on, and managed by a single computer system (a single-processor machine, or a multi-processor machine with shared memory or otherwise tightly coupled). A centralized DBMS is one that can create and manage only centralized databases.
Client: A user requesting the services provided at an interface of a server.

Client: A user requesting the services provided at an interface of a server.

Conceptual Schema: A description, usually in a formal language, of a universe of discourse.

Concrete Syntax: A syntax describing the way in which objects within a universe of discourse are physically to be represented by concrete sequences of symbols (bits for example) in a physical medium.

Conformance: The degree to which an implementation of a standard within a specific system satisfies the requirements of a standard.

Data Independence: The independence of process objects from data objects such that certain data objects may be changed without unnecessarily affecting the process objects.

Data Manipulation Language: A language for declaring the database procedures and executable statements of a specific database application program.

Data Modelling Facility: A class of data structure rules and associated rules for data manipulation operations.

Data Type: A grouping of data objects according to some rule for distinguishing the group from others. Typical rules involve the set of values which the objects of the group may assume and the set of operations which may meaningfully be carried out on objects of the group.

Data: A representation of facts or ideas in a formalized manner suitable for communication or manipulation by some process.

Database Management System: A set of software/hardware processing elements which together provide the services needed to support database management. A number of processing elements which together support and control the creation, use, and maintenance of a database.

Database: A collection of interrelated data objects stored together in the same container with controlled redundancy according to one schema to serve one or more applications.

DBMS: Database Management System.

DDB: Distributed database.

DDBMS: A distributed database management system.

DIS: Draft International Standard.

Distributed Database Management System: A distributed database management system (DDBMS) is a set of software/hardware processing elements which together provide the services needed to create and manage a distributed database.

Distributed Database System: A distributed information system in which data objects are distributed.

Distributed Database: A database in which its data or metadata (or both) are stored in parts or in multiple copies (or both) on more than one computer system, where the systems are connected by a communications network, and the database is managed in a coordinated fashion as a whole entity.

Distributed Processing: Distributed processing comprises that class of information processing activities in which discrete components of the overall processing activities may be located in more than one system, at more than one geographical location, or where there is any reason which necessitates explicit communication among the components.

DML: Data manipulation language.

DP: Draft Proposal.

Draft International Standard: Within ISO, a final draft of a standard produced for public discussion and comment within all member countries. This is the stage immediately preceding issuance of an ISO International Standard.

Draft Proposal: Within ISO, an initial draft of a standard produced for public discussion and comment within all member countries. Depending on the result of a vote among those members which are voting members of the relevant ISO Sub-committee, the Draft Proposal may be advanced to the status of Draft International Standard or may be sent back for revision.

End System: A real system.

Enterprise: An undertaking or venture in the real world.

Entity: Within the OSI Basic Reference Model, the conceptual objects responsible for performing the functions within a layer. The model does not place any restrictions on how entities are implemented in real systems.

Formal Language: A language formally defined in terms of an alphabet of basic symbols together with a syntax describing how these symbols can be put together to produce more complex linguistic objects within the language.

Gateway: A system which is responsible for transforming the conventions used within one system (or network of interconnected systems) to the conventions used in another.

Heterogeneous Systems: Systems which differ significantly from one another in their structure, or in the conventions used for representing information.

IEC: International Electrotechnical Commission.

Information Processing: The systematic execution of operations on data.

Information System: An organized storage and manipulation of information about a universe of discourse.

Information: The meaning that a human associates with data in a particular context through knowledge of the conventions of representation used.

Interconnection: Association of application processes with one another so that they can exchange in a dialogue.

Interface: A defined set of services made available by a processor.

International Electrotechnical Commission: A standardization organization whose responsibilities include standards for microprocessor systems and information technology equipment. IEC forms a part of JTC1 with ISO in the area of information technology.

International Standard: Within ISO, the final version of a standard on which international agreement has been reached.

Interoperability: Ability to interwork.

Interworking: Collaboration between application processes which enables them to perform distributed processing.

Information Resource Dictionary System: The set of software/hardware tools which together provide the services needed to support the knowledge of objects in an information system.

IRDS: Information Resource Dictionary System.

IS: International Standard.

ISO: The International Organization for Standardization, a body dedicated to the production of standards through international agreement. The ISO technical committee dealing with information technology (JTC1) is a joint committee comprising the former Technical Committee 97 of ISO and Sub-committees 47B (Microprocessor Systems) and 83 (Information Technology Equipment) of IEC.

JTC1: Joint Technical Committee 1, entitled "Information Technology", was formed between ISO and IEC to prevent duplication between the two organizations.

Language Binding: A defined relationship between the syntax and semantics of a programming language and that of some supporting service.

Layer: In OSI, a part of a hierarchically structured architecture, whose functioning is dependent on the layer below, and which provides a service to the layer above.

Local Syntax: A syntax used internally within a system.

Local System Environment: Those aspects of a system which do not have bearing on its interconnection with other systems.

LSE: local system environment.

Mapping: A set of objects having a defined correspondence with the members of another set.

Metadata: Data about data.

Open System: Within the OSI Basic Reference Model, a system which obeys OSI standards in its communication with other systems.

Open Systems Interconnection Environment: The set of definitions of the standardized services, protocols, and data structures which enable the interconnection of systems.

Open Systems Interconnection: A set of standards for the exchange of information among systems that are "open" to one another by virtue of their mutual use of the applicable standards.

OSI Basic Reference Model: A standard, produced by ISO, which describes the conceptual structure of systems which are to communicate with one another.

OSI: Open Systems Interconnection.

OSIE: Open Systems Interconnection Environment.

PDU: protocol data unit.

Peer-to-peer Protocol: A protocol used for communication between entities associated with the same layer of the OSI Basic Reference Model.

Presentation Layer: That layer within the OSI Basic Reference Model which is responsible for the representation of data to be transferred between systems in a form acceptable to the application entities within those systems. This service is provided to the application layer, and depends on the service provided by the session layer.

Processing Element: An abstract conceptualization of an executable piece of code.

Processor: A processing element.

Protocol Control Information: Information exchanged between communicating peer entities to coordinate their operation.

Protocol Data Unit: A unit of data specified in a protocol consisting of protocol control information and possibly user data.

Protocol: A set of rules for the interaction of two or more parties engaged in data transmission or communication.

Real Open System: A real system which complies with the requirements of OSI standards in its communication with other real systems.

Real System: a set of one or more computers, the associated software, peripherals, terminals, human operators, physical processes, information transfer means etc., that forms an autonomous whole capable of performing information processing and/or information transfer.

Remote Operations: A model, notation, service definition and protocol specification defined in ISO DIS 9072. The remote operations execution paradigm is based upon objects, in which the functionality of an object is modelled as a set of operations available at its interface.

Remote Procedure Call: A procedure call in which the calling procedure and the called procedure do not share an address space and may be in physically separate locations.

RO: Remote operation.

ROSE: Remote operation service element.

RPC: Remote procedure call.

Semantics: The meaning associated with an object or its representation.

Server: A role filled by a processing element when it provides services to another processor.

Service Element: A self-contained operation (or group of operations) associated with the activation of a particular facility within a service.

Service: In general, a capability provided by one processing element to other processors. In the OSI Basic Reference Model, the facilities offered by a layer to the layer above.

Session Layer: That layer within the OSI Basic Reference Model which is responsible for the organization and synchronization of dialogues between presentation entities. This service depends on the service provided by the transport layer.

Syntax: A set of formal rules for constructing sequences of symbols to represent objects within a universe of discourse.

System: A real system.

Transaction: A transaction is an atomic unit of execution; i.e., it is a sequence of operations which either are performed in entirety or are not performed at all.

Transmission: The process of sending (for example, data) from one place to another.

Transport Layer: That layer within the OSI Basic Reference Model which is responsible for providing transfer of data between arbitrary systems, with control on an end-to-end basis. This service is provided to entities within the session layer, and depends on the service provided by the network layer.

Universe of Discourse: That part of the real or hypothetical world which is understood in the same way by communicating partners and about which they agree to communicate.

Chapter 1

Introduction and Problem Definition

1.1 Introduction

The potential for multi-vendor communication networks is increasing as vendors implement products conforming to Open Systems Interconnection standards. Concurrently, data management, programming language, and operating system standards are being developed for the single-site environment. However, distributed systems products must become more "open" to allow the development of heterogeneous distributed systems. Many distributed database systems stem from research prototypes, supporting one or a small number of database management systems using vendor-specific communication protocols, without reference to the evolving standards.

Such developments are beneficial in advancing knowledge of distributed database management systems (DDBMSs) and of the practicality of different methods of implementation, but do little to address the urgent commercial requirement for implementation of distributed applications requiring access to databases controlled by different DBMSs. This requirement stems from a desire to take advantage of substantial investments in existing databases. The direction of the research undertaken for this study is predicated by a belief that the only pragmatic solution to the problems of great diversity which must be addressed by heterogeneous DDBMSs is to attempt to reduce the complexity to tractable proportions. The only practical way to accomplish this is to develop an integrated and comprehensive set of data management and interprocess communication

standards such that product vendors have a target to which the peculiarities of particular implementations can be mapped.

Standards work on distributed databases is in the preliminary stage. There is still no recognized framework addressing such problems as the determination of the location of data fragments or facilities for control of data replication. However, the Database Working Group of Subcommittee 21 of the Information Technology Technical Committee of ISO/IEC (JTC1) is developing a remote database access service that defines a service combining database language standards with OSI protocols to allow a "client" to read and update data at a "server".

1.2 Problem Definition

The major objective of this thesis is to evaluate the applicability of existing and developing data management and interprocess communication standards to the requirements of distributed database processing. This research has been done with a view towards providing recommendations concerning the direction of evolving standards for support of a distributed database environment. The focus of this research is centered on the design of distributed database management systems, rather than on the manner in which existing DDBMSs may be used in supporting particular application databases. The requirements that must be addressed by users developing applications of DDBMS technology are considered.

1.3 Thesis Outline

Chapter 2 addresses the goals of information technology standardization. In order that the scope of the area being studied is clear, Chapter 3 elaborates requirements that must be addressed by DDBMSs, many of which must also be addressed by single-site DBMSs. Chapter 4 reviews the progress of standards that have been finalized or are currently being developed insofar as they relate to the problems faced in a distributed database environment. Chapter 5 reviews the requirements identified in Chapter 3, and the existing situation as regards relevant standards, in suggesting considerations for component standards required to address distributed data management requirements. In this way the proposed "design", elaborated in the form of an architecture in Chapter 6, is similar in nature to the design that would follow the requirements definition and analysis stages of typical information systems developments. Chapter 6 outlines a proposed "model" for the Reference Model of Data Management that has been under development for some time within the database working group of Subcommittee 21 of the Information Technology Technical Committee of ISO. This reference model, similar in concept to the OSI Basic Reference Model, attempts to prescribe a framework in which existing and developing standards can be placed and the interfaces of new standards can be determined.

Chapter 2

Identification of the Primary Goals of Standardization

2.1 Introduction

In the past, information technology standards have focused individually on data communication, data processing (as in manipulation), and data storage and retrieval. Interprocess communication standards are exemplified by the various components of Open Systems Interconnection (OSI). Information processing standards have been developed for a wide variety of programming languages. Database standards have been developed in order to facilitate definition and access to data from varying programming languages.

Information technology standards have generally addressed the separable problems involved in:

- 1) the portability of applications, and
- 2) the interworking of several applications.

Database and programming language standards have addressed the concerns of application portability. OSI standards, on the other hand, have been more concerned with interoperability, that is, attempting to ensure that different application processes are able to interwork to achieve application objectives.

This chapter addresses the goals of standardization as they relate to

the area of distributed database technology. The timing is appropriate in that standards addressing such technology are still in their infancy. Since distributed database technology is concerned with interoperability as well as with the portability of database applications, it is beneficial to consider the goals, advantages, and disadvantages of information technology standards as they relate to both of these standards objectives.

There are three basic types of information technology standards:

- *de facto* standards from manufacturers;
- industry or private standards; and
- official national and international standards.

De facto and industry standards are important to both vendors and users in formulating an information technology strategy. However, national and international standards are of primary interest since they are nonpartisan, and will form the context in which standards are addressed in this study. The International Organization for Standardization (ISO), the International Telephone and Telegraph Consultative Committee (CCITT), and the International Electrotechnical Commission (IEC) are the international organizations having the greatest influence on information technology standardization. Standards may be thought of as "global technical treaties" benefiting both suppliers and users for technical, business and economic reasons.

There are a number of reasons for standardizing information technology hardware and software components. The need for connectivity, forced by the integration of computers and communications, is one of the major factors in making information technology standards a prominent issue. The absence of standards can lead to the development of multiple similar solutions addressing the same problem, with consequent duplication

of effort and interfacing difficulties. The goal of Open Systems Interconnection, for example, is to mask the effects of the differing technologies and specific designs of different vendors from the interconnection process. The criticality of standards in the telecommunications industry has long been recognized due to the need for telephones to communicate with each other. However, standardizing a technology before it has become sufficiently mature may be detrimental to the propitious development of the technology.

In considering the timing which may be appropriate for standardization of a specific area of information technology, it is relevant to examine the goals of information technology standardization. The two major goals are application portability and interworking. Application portability is addressed in section 2.2; section 2.3 considers the need for interworking in a distributed database environment.

2.2 Application Portability

Application portability is the ability to move an application from one computing environment to another with little or no change to the application. Application portability makes it possible to distribute applications and data throughout an organization according to business needs rather than according to system compatibility. The accessibility of the data processed or managed by an application also affects the extent to which an application is portable.

Portability increases the range of environments in which a product can be applied with reasonable cost. The result is that the costs of developing a product can be amortized over a broader base. The effective lifetime of a portable piece of software can be expected to be longer, since it is possible to

transport it to more modern hardware at a later date. Portability of software has become a dominant issue in information processing, particularly in the public sector where procurement policy forces procurement requests to be open to the widest possible competition.

Perfect portability implies that an application can be ported in object code (binary executable) form. Much of the standardization effort associated with programming languages has focused on source code compatibility, which can often be completely adequate for the porting of applications from one environment to another with reasonable expense. Much of the standardization work relating to databases also has addressed the source code (database language) level and, as will be discussed subsequently, this situation presents limitations for the interworking which is required for distributed database processing.

Portability of application software can result in benefits for both users and vendors. For users, an increased degree of application portability implies a greater degree of vendor independence and a greater degree of protection of investment in applications software. Vendors are provided with opportunities for access to markets which would otherwise be inaccessible as a result of high costs of conversion. Standards thus force competition based on factors other than product differentiation for the sake of differentiation. If the functionality to be provided by a vendor product can be measured by conformance to a particular standard, then evaluation of products can be based on performance, user interfaces, and other criteria of particular interest for the requirements being addressed.

2.3 Interworking

Distributed systems considerations require that components of an information processing system which are physically separated must be capable of working together in order to perform a data processing task. In the case of distributed database management, clients at remote sites must be able to access data at servers located remotely. A primary goal of standardization is to address such requirements for interworking (or interoperability). Satisfying such requirements goes far beyond normal requirements for application portability, which often are addressed satisfactorily by source language standardization. Effective interworking requires more than source language compatibility, in that recompilation of procedures is not feasible from an operational viewpoint for each execution of the procedure. Distributed database processing is one area where effective interworking is an important standardization concern.

2.4 Advantages and Disadvantages of Standardization

The demand for interconnection has been the primary motivating factor leading to the rapid proliferation of information technology standards. Since requirements for interconnection are ultimately international, the standards with the greatest potential to affect the future of the information technology landscape are international standards.

The heavy costs of research and development by vendors, and the equally high costs of information systems redevelopment by users, are the

main reasons that both governments and industry support standards. Standards are designed to ensure that such investment is long-term. Standards are intended to increase the stability of the information technology environment, but in order to be successful they must be formulated with great care and foresight. Formally agreed standards are fairer to information technology vendors, in that no single manufacturer's specifications are the basis for interconnection. The definition of standards requires agreement on a generic, abstract form of specification unlike that used by any particular vendor. Standards limit the number of solutions to the same problems; in doing so it is possible to concentrate on improving a smaller set of techniques. The result is that users become less "locked in" to the products of a particular vendor, increasing competition and the choice of available products for many applications.

However, standardization does have the potential to stifle progress, particularly in a fast-moving field such as information technology. Although it is true that the premature introduction of an inappropriate standard has the potential to cripple new technology, the failure to reach agreement on a single standard can cause great inconvenience and expense in the longer term. An example is the requirement for standardization of electrical voltage, frequency and physical connection parameters.

One of the disadvantages of standardization is the overhead which may be associated with certain standard solutions. For example, OSI standards must account for many differing ways of communicating, and thus must include functions which may not be necessary in certain implementations. Another disadvantage of standards is an increased degree of rigidity, which can tend to restrict innovation. The slow development of

standards causes consumers to delay purchases and vendors to hold back investments until they are sure that what they buy or sell conforms to a forthcoming standard.

One of the fundamental motivations for users to adopt standards is to achieve a greater degree of stability in their information systems environments. Making changes in established standards is very difficult since users will tend to resist changes which make existing systems incompatible with new standards. This results in the need for adequate planning and design of information technology standards so that necessary modifications can be facilitated. As information technology fields such as data communications and data management become more integrated, the need for determination of how various standards components will fit together increases. If information technology standards are to provide the stability which is the motivating factor in their development, the design of the components must provide for evolution into the future.

The determination of standards components that will fit together optimally in order to provide a coherent framework for developing user applications is similar in nature to the development of any information processing system. First of all, requirements must be addressed taking into consideration the existing system. Next, such requirements must be analyzed and a design formulated in which existing components can be placed. Also, the design will prescribe the functionality to be addressed by required components. Only by preparing such a design can a normal project management approach permitting parallel development by numerous project participants be accommodated. As associated with an information systems development project, the project phase dealing with the

identification of components and the manner in which these components interrelate is known as the design phase. In standards development work, this phase often is addressed by the preparation of a reference model.

2.5 Reference Models

2.5.1 OSI Reference Model

Perhaps the best known reference model within the information technology standards arena is the Basic Reference Model of Open Systems Interconnection [52]. At a time when a number of different network architectures were in existence or being developed, the OSI reference model was an invaluable aid to standardization in that it provided a framework and methodology in which standards for each of the seven layers could be developed in parallel.

The objective of the OSI Reference Model was to establish a framework for coordinating the development of existing and future standards permitting interworking between systems. Such interconnection was intended to address specifically the interprocess communication aspects of "open systems". The reference model outlined the functions to be addressed by each of the layers so that replacement of the software or firmware addressing this functionality would be feasible.

The approach of OSI to the requirements of interprocess communication was to provide a standard set of conventions and mechanisms for representing information to be communicated. Prior to the availability of such an approach, the only method in which differing network architectures could be interfaced was by means of a "gateway"

approach. In this approach (see Figure 2.1), the functions and conventions of one architecture are mapped individually to each differing architecture to which communication is desired. The problem with this approach is that the number of interfaces which must be developed in order to interface "n" differing environments is equal to $n(n-1)/2$. In Figure 2.1, it can be seen that the number of different sets of conversion facilities that must be developed in order to successfully provide interfaces between 4 differing information processing systems (hardware and software) is 6. In contrast, the OSI approach to interconnection, shown in Figure 2.2, requires only as many sets of conversion facilities to be built as there are different information processing systems. Each vendor is required only to convert their proprietary representation to a common set of conventions, which is defined by the various standards of OSI. Thus the number of separate conversion facilities needed to permit interworking between 4 different systems is limited to 4.

The logistics of interfacing merely 10 differing system environments, requiring 45 sets of conversion facilities, point to the obvious advantages of the OSI approach to interconnection. In the OSI approach, the effort required to provide interfaces between 10 differing system environments is only 10, where each vendor has responsibility for mapping its proprietary architecture and conventions for data representation and communication to the vendor-independent standard represented by OSI. The economic advantages of the OSI approach to interconnection are obvious, although it can be expected that sufficiently influential vendors will attempt to set the standard as their proprietary standard and require that other vendors interface with such a standard.

The manner in which Open Systems Interconnection standards have evolved is a major change in the approach to development of national and international information technology standards. In the past, standards were often formalized after *de facto* standards had been established, and the rate of development of such standards was viewed as painfully slow. In the case of OSI standards, vendors are developing products **after** the standards have been developed, or are near enough to finalization in order to justify the expense of product development. While vendors in the past have "extended" standards-specified facilities (in nonstandard fashion) in order to make up for weaknesses in standards-defined functionality, many standards now contain advanced features not available in product offerings. This is partially a consequence of user pressure motivating vendors to actively participate in the standards development process and to develop standard-conforming products.

2.5.2 Need for a Reference Model for Data Management in a Distributed Environment

Since the early 1980s a group responsible for producing a reference model for data management standardization has existed within the International Organization for Standardization. However, no equivalent of the OSI reference model has yet emerged for data management standardization. There are a number of reasons for this. Data management standards have long been associated with programming language standardization. This is to be expected considering the requirement for database information to be accessible to users operating in differing programming language environments. There is also an increasing need for database access from remote locations. Data Dictionary System standards are being developed in parallel with database language standards. Since

these standards (referred to as Information Resource Dictionary System (IRDS) standards) have evolved in parallel with database language standards, and have not assumed the availability of standards-defined database facilities, such as IRDS standards have developed independent data definition and manipulation language facilities. Database language standards initially focused on the Codasyl-style Network Database Language, which became an ISO and ANSI standard, before the value of the network data modelling facility was diminished by the superiority of the relational model. Remote database access standards, initially proposed by the European Computer Manufacturers' Association, have been progressing within ISO since 1985. In essence each of the three related data management standards efforts, that is, database language standards, IRDS (data dictionary) standards, and remote database access (RDA) standards, have evolved independently, without the benefit of a reference model framework addressing the optimal manner in which the component standards would best fit together.

The reason that such a reference model has not emerged relates to the complexity inherent in achieving agreement on how programming language, database language, IRDS, operating system, and RDA and other interprocess communication standards should interface. This, in turn, provides some indication of the degree of maturity of data management standardization. Until such a reference model of data management standardization is agreed upon by the responsible national and international standardization committees, it can be expected that progress in individual areas of data management standardization will be less than optimal, and that coordination problems will exist. The result is likely to be similar in nature to an information system where an adequate design has

not been finalized prior to the development of the various programs and modules that make up the system.

The effect of such a situation on the possibility of standard distributed database functions can be expected to be profound. Since coherent and integrated single-site standards form a necessary prerequisite to standardized distributed database functionality, it is necessary that any complete reference model for data management standardization must address the provision of data management facilities both locally and in the distributed environment. Given the plethora of vendors supplying data management products, or products with a data management component, the need for an "open system" environment in order to facilitate heterogeneous distributed database management is apparent. However, OSI standards only address the communication aspects of such an open system environment, with the standardization objective of interworking. Database and IRDS standards developments have focused on the standardization objective of application portability. Distributed database standards must be concerned both with application portability and with interworking.

In recognition of the extent to which an agreed reference model of data management standardization is a prerequisite to progress in distributed database management standardization, much of the effort of this study has been directed towards the advancement of such a reference model. In fact, inclusion of requirements for distributed database management as a part of the functionality which must be addressed by such a model has been beneficial in providing insight into problems relating to the interfacing of standards components in a single site environment.

The question remaining is the extent to which the field of distributed database management is ready for standardization. In the opinion of the author, the lack of progress in standards for such facilities is *the* major reason for the slow rate of development of commercially viable DDBMS product offerings which address heterogeneous environments (heterogeneous with respect to hardware, database software, and communications facilities). However, as was the case with single site DBMS database standards, there are many aspects of distributed databases not requiring standardization. For instance, much of the research in distributed databases addresses distribution and replication alternatives. As with single site DBMSs, these alternatives, dependent upon parameters such as the ratio of queries to updates, are largely performance oriented and fall outside the range of facilities requiring standardization. Conversely, a standardized manner of storing and materializing data location information is crucial to the functioning of heterogeneous distributed databases.

The degree to which a field of information technology is ready for standardization may be determined by the degree of consensus reached on a framework for placing existing and developing standards and prescribing new standards. The remainder of this thesis addresses requirements for data management functionality both in local and distributed environments, with emphasis on those aspects of distributed database systems that are amenable to standardization. Existing and developing standards are considered in Chapter 4, focusing on the extent to which they are able to address the requirements outlined in Chapter 3. Chapter 5 then presents analysis and design considerations for addressing such requirements in a distributed environment. Chapter 6 suggests a "model" of a reference model of data management, using the object modelling approach adopted by

the Reference Model of Data Management Rapporteur Group of the Database Working Group (JTC 1/SC 21/WG 3) of ISO.

Chapter 3

A Survey of Requirements for Distributed Database Management Systems

3.1 Introduction

The concept of a distributed system involves "dividing up" activities usually associated with a tightly-coupled, centralized information system over a number of processing sites. The activities of such a distributed system must be coordinated and global rules must apply across sites, but at the same time each site should be allowed as much local autonomy as possible.

This chapter is concerned with requirements which must be addressed in a distributed system attempting to provide data management functionality. Although many of these requirements also must be addressed in a single-site data management environment, the needs outlined below center on data management objectives for which the impact of implementation in a distributed environment must be considered.

In identifying requirements, an attempt has been made to distinguish between the functionality needed and differing mechanisms for achieving that functionality.

3.2 Database Access Requirements

3.2.1 Data Accessibility Requirements

There is a requirement for all data management services to be accessible both in a remote environment and in a local environment. While such a requirement may seem obvious to practitioners attempting to implement distributed database management technology, the manner in which single site DBMSs have been specified and built has frequently assumed a local environment only. In a local system environment, shared memory may be assumed. Products and standards specifications that are predicated on such an assumption cannot be expected to demonstrate the flexibility required of a distributed systems environment.

The nature of databases is such that their data must be utilized in various standard or nonstandard programming languages environments, for both local and for remote access. This stems from the natural expectation that different users of database information are going to be using the information for diverse purposes, requiring different types of manipulative facilities. In a distributed environment, there exists the possibility that the identity of the server(s) from which information is being requested may not be known, or alternatively, information may be available from one of a number of sites holding replicated data.

3.2.2 Communication Requirements

Interprocess communication requirements of distributed databases, while conceptually independent from data access requirements, cannot be addressed in isolation. The need for integration of data management and communication services is of considerable importance if DDBMSs are to

provide access utilizing differing vendors' database and communication facilities. The requirements for integration stem from the need for processes in different sites to communicate effectively in addressing requirements such as distributed concurrency control. Interprocess communication facilities are needed in order to permit interworking, not just data communication, and are relevant in both local and wide area network environments. In the same way that future DDBMS products must be concerned with differences in the local DBMS implementations which may be supported in a DDBMS, such products must also be concerned with the differing network architectures and communication protocols supported by different vendors.

3.3 Requirement for Object-Oriented Facilities

In recent years the abstraction mechanisms available within programming languages have been enhanced considerably, primarily in keeping with the approach exemplified by the object-oriented paradigm. Abstraction permits the definition of data objects in the software in a manner that is natural for the objects that occur in the problem domain. Another distinct advantage of this approach is that a programmer can define operations on these data objects so that they may be used without worrying about the details of their implementation. Furthermore, abstraction constructs which allow object type definition to be grouped together with permissible operations provide the possibility of "data hiding", in which manipulation of an object can only be done via the given operations. The benefits of preventing access by programmers to an object's implementation, so that they cannot circumvent the use of predefined subprograms, have been well recognized as regards the potential provided

for averting subsequent problems with debugging and maintenance of the software. Such encapsulation of data structures and operations, further enhanced by programming language facilities for definition of object types and their operations in separately compiled reusable modules, is provided by more recent languages such as Ada and Modula II. The encapsulation associated with these programming languages is also relevant for database access procedures.

For example, in order to specify an Information Resource Dictionary System standard, it is necessary to define data structures that will be under the exclusive control of the IRDS implementation. In order to ensure the integrity of this data, a requirement exists to "encapsulate" the facilities relating to such data so that it may only be accessed by means of the "legal methods" of the object-oriented approach.

3.4 Data Dictionary System Requirements

Users in an information systems analysis, development and maintenance environment have very similar requirements for control of the information objects related to their endeavors. The result has been the development of software products which attempt to address both the data objects to be maintained and accessed in an information system, as well as other types of information (such as information on programs, modules, data flows, manual or automated processes, users, etc.) required for analysis and control purposes.

The purpose of a data dictionary/directory is to control information relating to data entities in a manner which facilitates the effective development of information systems. Since a great deal of diversity existed in hardware and software, data dictionary systems were created as

standalone products independent of the DBMSs and file systems controlling the application data. Only by such mechanisms could the valuable resource represented by the "metadata" be given a degree of stability.

Frequently, the efforts of data administrators to utilize data dictionary tools have been unsuccessful. One of the major reasons for difficulty has been the lack of stability of data dictionary system products, and the extent to which these products have been limited in their ability to assert "active" control over information system objects stored in system files and computer databases. The standalone "passive" dictionary systems of the past have suffered from being "bypassable". For instance, the most up-to-date information regarding the program modules that update a particular data item value should be available in the data dictionary system. However, whether the data dictionary information is, in fact, the latest information reflected in current program listings is controlled only by administrative arrangements, and is thus subject to error. The fact that dictionary information *may not* be current tends to negate its value. The result has been that maintenance of data dictionary information is often neglected. This can easily culminate in a data dictionary that is of little or no use.

The requirement for a definitive source for information related to the data and other types of information entities can be addressed by data dictionary systems which are "active" with respect to the other software products utilized in information systems. Requirements for control of data objects stored in a database can optimally be addressed by a dictionary system where the metadata about data entities is the catalog, or schema, of the database system. These requirements could not be adequately addressed before standardized database system facilities provided some stability in

the DBMS environment. The lack of integration of related software components has been the major technical reason for the failure of many well-intentioned data administration and information resource management staff.

The facilities to be supported by the Information Resource Dictionary System standard, from [28], include the following:

- to provide a shareable repository for all definitions and descriptions relevant to the analysis, design, construction and use of a system;
- to support database administration, document administration, information resource management and data administration;
- to support distributed processing and the distributed database environment;
- to provide a source and object library management system;
- to support version control;
- to provide a common means of specifying data definitions accessible to programming language compilers and database management systems;
- to provide facilities for programs written using ISO standard programming languages to use SQL and NDL (Network Database Language) data manipulation languages to access information stored in the IRDS at execution time;
- to allow access to its data from another Open System using standard OSI protocols such as Remote Database Access; and
- to provide for portability of skills for users of standard-conforming IRDSs, by means of appropriate user interface facilities such as a concrete command language or panel (screen) interfaces.

In a distributed environment, control of metadata must be extended to support the fragmentation of data among various sites.

3.5 Requirements for Security

3.5.1 Access Control

The requirements for control of metadata, from section 3.4 above, indicate the need for a number of different levels of data to exist in defining and controlling the data management environment. In order to protect the integrity of this data and to protect the data from unauthorized disclosure, the need for access control provisions relating to database and data dictionary data (at all levels) is well recognized. Such access control provisions are needed relative not only to data types being used, but also to specific instances of these data types. Access control facilities are needed to prevent breaches of security from both intentional and unintentional misuse. In order to provide more specific control of the operations which are permissible in updating or retrieving such data, it is also necessary to provide the capability to restrict operations on data to specific procedures, as suggested by the object-oriented paradigm.

3.5.2 Authentication

The requirement for authentication is to ensure that a user of database resources is in fact the identified user. While such conditions existed in centralized DBMSs, the requisite verification procedures are more complex in a distributed database because of the greater potential for unauthorized access to information and for corruption of the database.

3.6 Requirements for Transaction Management

The issues of integrity, recovery, and concurrency control are strongly interrelated in database systems. Integrity deals with ensuring that

database information is maintained accurately and consistently according to the rules and constraints which pertain to the data. Recovery deals with restoring values to a consistent state when all of the operations of a particular update request cannot be completed. The requirement for a number of users to perform operations at the same time introduces the need for concurrency control provisions. The solution to these problems for both local and distributed database processing centers around the provision of transactions.

A transaction is defined as a delimited sequence of database operations which together perform a logical unit of work. The concept of a transaction is essential to reliable database management. Where database modifications are involved, a transaction is a sequence of update steps which change the content of a database from one consistent state to another. The nature of what constitutes a consistent state is known only to the application. Thus the scope of the transaction must be defined by the application. A primary goal of database management is the preservation of the integrity of the data given the possibility of hardware or software failure. In a batch processing environment a job which made changes prior to a failure can simply be rerun, while in an interactive environment changes must be recorded separately so that they can be reinstalled if a failure leaves the database in an inconsistent state. The possibility of a device failure or a software failure at the time that updates are made to the stored data cannot be eliminated. Since such failures can leave the database in an unknown and potentially inconsistent state, it is necessary to record the progress of transactions in a log file held in stable (disk) storage in order to be able to recover from various failures.

Haerder and Reuter [55] have studied techniques to support transaction-oriented database recovery, and have proposed a generally accepted "ACID" test of a system's quality, notably that a transaction must be:

- Atomic** - all changes must be present in the database at the completion, or none must remain;
- Consistent** - at completion the transaction leaves the database in a logically consistent state;
- Isolated** - changes made by a transaction must be invisible to other concurrent transactions (and vice versa);
- Durable** - once committed the system must guarantee that the results of the transaction survive subsequent malfunctions.

The specification of requirements for ensuring that such properties of transactions are realized has resulted in the concept of "serializability". This requires that the concurrent execution of several atomic transactions must be serially equivalent in the sense that the effect of performing them concurrently is always the same as if they had been done one at a time in some arbitrary order.

In a local DBMS environment, the requirement for serializability can be explicitly stated and the mechanisms used to provide such a facility can be known only to the DBMS supplier. In distributed processing, aspects of transaction management are related both to the features of the local operating system, especially process management and interprocess communication, and to the features of the communication network. In order for heterogeneous DDBMSs to be realized, mechanisms for ensuring serializability cannot be strictly proprietary to a particular DBMS vendor - they must be standardized so that individual suppliers can be expected to utilize the same mechanisms and protocols. In a distributed database system, a component of the database management system must manage a

transaction at the originating site, and cooperating components must manage the transaction steps performed at each participating site.

The need for communication components in DDBMSs implies that distributed database systems not only face more complex synchronization problems than centralized DBMSs, but also more and different modes of failure. The synchronization problems are not substantially different from those encountered in centralized transaction processing; the major differences arise from the modes of communication failure:

- messages may be lost in transmission;
- messages may not arrive in the order sent due to transmission errors and consequent retransmission; or,
- certain failures make it impossible to tell if it is a remote site or the connection to that site that is down.

The latter case can result in the partitioning of a network by a communication failure so that it is divided into two (or more) separate groups of sites. Each group comprises one or more sites that can communicate with each other, but not with any site in another group. Database agents in one group may continue to operate in isolation from other groups, with consequent decisions which could affect the integrity of the database if there are outstanding transactions whose effects span the partition.

In order to prevent loss of database integrity as a result of such failures, it is necessary to know the degree to which data is replicated. This requires the availability of location information concerning replicated data in a form that is understandable to each of the cooperating DDBMS components.

Since database systems must service a number of simultaneous interactive users, concurrently executing transactions must be synchronized so that they do not interfere with each other. Concurrency control provisions must address a variety of problems that may compromise the consistency of the database, such as lost updates and inconsistent reads. There are three common solutions to the problem of concurrency control: locking, optimistic concurrency control, and timestamps. Concurrency control based on the two-phase commit procedure has most often been used in distributed database and file services developed to date, but more optimistic timestamp-based methods are emerging and may offer a more flexible and efficient solution in the future.

Another requirement associated with concurrency control is prevention of deadlock (or deadly embrace). This can be done by several methods, such as cycle checkers, which choose a victim when deadlock is detected and abort one transaction; or time-outs, which abort a transaction on the presumption that it is involved in a deadlock (after which all of its locked records are released).

It has been shown [12] that there are no statistically significant differences in the relative performance of implementations using each of the three different methods of concurrency control, unless the database size is very small. In normal operational situations conflicts are rare, with typically less than 0.1% of transactions being required to wait. Given this information, the most important aspect of runtime concurrency control performance is the processing overhead for each transaction, as opposed to the time taken for rectifying conflicts. The primary requirement for distributed database processing is that the mechanism used for concurrency

control must be consistent across participating sites, whereas more vendor latitude is possible in each local DBMS.

3.7 User Interface and Presentation Requirements

Reference [12] divides the technology relevant to information systems into five categories: hardware, software, communications, data, and people, where "people" is clarified to mean essentially the interaction between people and machines. It is generally acknowledged that the man-machine interface is a critical aspect to the acceptance of a system by its users.

The requirement for object-oriented interfaces (sect. 3.3) can be particularly important in the development of user interfaces. The "objects" that are visible to a database user are often database data structures. However, the data structures optimal for DBMSs and DDBMSs are not necessarily the same data structures which all classes of users may wish to have presented to them. For example, users of an accounts receivable system wish to deal with objects of interest (such as invoices) for the purposes of the particular application system, which may not be constructed as third normal form relations. Other users may wish to view the data in the form of some hierarchical view. For control purposes, the database system should provide facilities to limit user access and update to previously defined routines, such that database integrity can be maintained.

Another example of a user interface requirement of common concern is the need for data administration tools such as database design aids. Such requirements typically support data analysis and design activities in the systems development cycle. These tools frequently make use of data modelling facilities which can be readily understood by human users,

typically by means of diagramming techniques. Chen's entity-relationship model [76], proposed in 1976 and modified subsequently into a number of variations, has gained wide acceptance as such an analysis/design tool.

A requirement for data administrator support is the ability to view data in terms of data structures appropriate for data analysis. The entity-relationship data modelling technique is one possibility; other approaches such as the binary relationship approach (discussed in [38] and evaluated as a candidate for a conceptual schema language) are equally useful in analyzing data and determining dependencies between data objects.

Another user interface requirement relates to the style in which access to database data is to occur. Possibilities are various types of programmatic interface and interactive dialogues with the database control system of the DBMS or DDBMS. Using programmatic interfaces, data manipulation language requests are usually compiled and combined with the output of programming language compilation in creating a reusable object module. More interactive types of interface require compilation prior to execution of each database access request.

User interface requirements to DDBMSs must account for differing styles of access and differing degrees of encapsulation of the access and update requests.

3.8 Requirements for Distribution Support

3.8.1 Overview

A primary requirement of DDBMSs is to provide distribution transparency, that is, the independence of application programs from the

distribution of data. Regardless of the data model employed, any database, whether distributed or not, can be described by a schema, sometimes referred to as a *global* schema in a DDBMS. The global schema defines the totality of data available in the database. Such a schema can be defined in the same way for a distributed database as for a single site database. The relational model is used most frequently in describing the desirable attributes of DDBMS implementations, largely as a result of the complications that can occur in distributed implementations of other data modelling facilities such as the hierarchical and network models. These complications are caused primarily by the limitations of "one record at a time" navigationally oriented DML facilities, and as a result of difficulty in utilizing pointers which cross site boundaries. The fact that an international standard has been accepted for the SQL database language further increases the desirability of utilizing the relational model for defining the global schema for DDBMSs.

Fragmentation refers to the splitting up of the data defined by the global schema into a number of non-overlapping portions called fragments. Such fragments may be allocated, redundantly or non-redundantly, to the various sites of a network.

Data fragmentation is conceptually distinct from data allocation, the former being associated with the DDBMS property of fragmentation transparency, while the latter is associated with a lower degree of distribution transparency known as location transparency. With fragmentation transparency, the user is insulated from the existence of a number of fragments, and, in relational terms, the user works with global relations. Location transparency is a lower degree of distribution

transparency in which the user must take into consideration the different fragments existing in the DDB. However, the user need not be aware of the location(s) of the fragments. Another desirable feature of distributed databases is that the degree of redundancy can be controlled, that is, the degree of replication of fragments for performance and availability purposes can be regulated.

Global relations may be fragmented (or partitioned) in several ways. Horizontal fragmentation consists of partitioning a global relation into subsets by means of allocating different tuples (records) of the table to different sites. In horizontal fragmentation data is assigned to sites by the value of some attribute in each tuple; e.g., tuples of a relation with a given value of attribute X are assigned to site A, those with a different value to site B, etc. Vertical fragmentation of a global relation is the subdivision of its attributes into groups. In vertical fragmentation, some defined attributes of a relation are assigned to one site, while others are assigned to another site.

Control of replication must be provided by means of algorithms that ensure coordinated updates within transactions. The simplest such algorithm is the "Read Any Write All" technique. This may be an acceptable approach if updates to the replicated data are rare with respect to reads. However, such an approach suffers from communication failures in that updates are not possible if any copy is inaccessible. Other algorithms, generally of increased complexity, may be more appropriate depending on the frequency of retrieval requests to updates. All such algorithms must take into consideration the need to ensure that all replicated copies are eventually updated, although all schemes other than the "Read Any Write

All" algorithm must be expected to tolerate some degree of ephemeral discrepancies between replicated copies of the same data. In many database applications, coping with differences between replicated copies, although increasing considerably the complexity, is much preferable to disallowing updates if an affected site is unavailable. In determining optimal replication schemes, the level of data being accessed must be considered. For example, most access to dictionary-level information is for retrieval, not update. Thus to ensure high availability and enhance performance, multiple copies of dictionary information may be appropriate within a network. This permits many look-ups to be local, rather than involving network interaction and delay. The ratio of retrievals to updates may be considerably different for the application level data which is controlled by dictionary information, and thus the appropriate replication strategy may be different.

Requirements for robustness of distributed database systems favor the autonomy of sites to the greatest extent possible. This is the case for both technical reasons, such as performance and availability, and for administrative reasons such as accounting and authentication.

The degree to which the various types of transparency described above will be supported by differing DDBMSs can be expected to vary from implementation to implementation. Requirements common to all DDBMSs are the need to define the global schema, the need to be able to define segments of the occurrences of that data (fragments) which may exist at one or more sites, and the ability to provide for allocation of such fragments to one or several sites in order to support operational requirements for database access.

3.8.2 Distribution Alternatives

In the same way that a DBMS conceals the details of physical data storage in order to provide data independence, it is desirable that the details of data distribution are concealed from the application programmer or end-user. Such masking of information from the user, as is required in providing location transparency or replication transparency, applies both to application level data and to data dictionary information (or metadata). Regardless of the extent to which data may be replicated, a desirable DDBMS feature is location transparency. The objective of location transparency is to permit programs to be written as if the database was not distributed, although the speed of execution of the programs may differ depending on whether data is stored locally or remotely. The requirement for support in a distributed database environment is to permit data to be distributed according to the usage patterns of a particular application, while supporting the types of transparency outlined in section 3.8.1.

3.8.3 Replication Alternatives

The extent to which DDBMS data is replicated is a primary consideration affecting the performance possibilities for DDBMS implementations. The issues associated with the optimal degree of replication are basically cost/performance in nature, where we must be concerned both with the cost of communications and the cost of replication. Communications costs can be reduced by replicating data where queries are significantly more frequent than updates. However, since updates must be propagated to all copies, at a cost in communications services and time, replication is a costly strategy where updates are more frequent. Extensive

research has been done on the optimal strategies for replication of dictionary (or directory) information. Allocation of such information may be (from [12]):

- totally centralized. This alternative relies on the availability of the central site for data access. Dictionary information is easily updatable, but the central site can become the bottleneck of the system.
- fully replicated at each site. This alternative not only increases the availability of dictionary information at each site, but also increases the difficulty of performing updates, since each site must be updated.
- locally partitioned. In this alternative, dictionary data is fragmented such that the information is stored at the same site as the objects to which it refers. This alternative increases the volume of traffic on the network if there are a number of non-local queries, but is effective when most of the queries are at the local site.
- extended centralized dictionaries. In this alternative, each application which requires a given part of the dictionary brings the information to its local site during its execution. This approach reduces the number of remote queries for dictionary information, but requires that the dictionary information is kept up-to-date by the propagation of updates to sites which have copied dictionary information.

According to Chu [56], the ratio between updates and queries turns out to be the most significant optimization parameter regarding distribution relating to dictionary or directory information. If the ratio is below 10 percent, then the preferred alternative is the fully replicated dictionary. If the ratio is between 10 percent and 50 percent, then the preferred approach is the extended centralized dictionary. If the ratio is over 50 percent the preferred solution is the locally partitioned dictionary.

It is apparent that the distribution alternatives which must be supported by DDBMSs must be flexible, varying from full replication to no replication at all, depending on the usage patterns of the data to be controlled by such systems.

An additional requirement relating to data replication is the degree to which local autonomy of each site in managing its own data is required. Certain DDBMSs (such as System R*) support full autonomy of sites, while many others, such as INGRES-STAR and SDD-1, have not insisted on site-autonomy as a design consideration.

3.8.4 Requirements for Data Location Information

In a centralized DBMS, physical data independence is achieved by isolating application programs from physical data storage attributes (disks, cylinders, tracks, etc.) so that application programs need only be concerned with logically defined structures. The management of a distributed database also requires a "database" of data fragment location information, which may itself be distributed. In the case of the centralized DBMS, the location information relates to the particulars of the specific hardware/software environment in which the DBMS is operating, and thus can be completely proprietary in nature. The distributed location information of DDBMS implementations must be accessible to all participating sites. Thus a requirement for heterogeneous implementations is that data location information must be globally available within the DDBMS network - in practice this necessitates some degree of standardization in the format utilized to represent such location information.

3.8.5 Administrative Requirements for Supporting Distribution

An operational requirement for distributed databases is for tools to support administration in distributed environments. Tools such as data dictionary systems to support database administration have become viable in centrally located databases. Generally, the tools do not yet exist to

support database administration and network management in a distributed environment. Events that require the cooperation of all the sites of a network, such as the addition of a new site to the distributed database, are very difficult to manage manually in a distributed environment. From an operational viewpoint, as local autonomy increases, it becomes increasingly difficult to bring database administration, which involves users, organization management, systems operators, and applications maintenance staff, under a single authority.

3.8.6 Federated Databases

The development of distributed systems requires that problems of software integration be overcome. Implementors of distributed systems must be concerned with a variety of differing software on a number of different machines, and communication between dissimilar software components must be possible. Complications in supporting a distributed database environment include the problem of heterogeneous databases based upon different data models or different implementations of similar data models, as well as problems associated with differing operating systems and communication network architectures on different machines.

Attempting to integrate differing software and hardware for the purposes of a particular distributed database configuration may well prove to be extremely difficult and expensive. Vendors must live with the problems of integration in order to protect their existing customer base, while users must contend with the problem of integrating existing systems because of large investments in systems based on existing DBMS implementations. Both users and vendors cannot afford to ignore the considerable investment in existing systems and databases when

considering DDBMS implementation. Thus a DDBMS requirement is the ability to incorporate existing DBMS implementations and databases into future DDBMSs.

3.8.7 Performance Considerations

The response time perceived by each user of a distributed database management system is the sum of three components: the processing time at each site involved in a request, communication delays in both the forward and return paths to each affected site, and the queuing delays at each node. The problem of query optimization has been analyzed in considerable detail in order to determine optimal access plans to access data that may be replicated at various sites in the network.

A number of factors affect the performance of a DDBMS implementation. Important factors are the degree to which data is replicated, and the extent to which information is available concerning the delay associated with accesses to differing nodes of the network. Data compression, for both storage and communication, should be totally transparent to the user, and is important for certain types of data, such as images. Another factor of considerable significance is the manner in which binding is accomplished, i.e. the choice between pre-compiled queries and dynamically interpreted queries. The option is between analyzing and optimizing a query each time a query is processed if interpretation is used, but only once if queries are pre-compiled. This choice reflects the potential conflict between efficiency and flexibility. There is definitely a requirement for support of both repetitive queries and ad hoc queries, both by single-site DBMSs and by their distributed counterparts. As well, there is a need to ensure that pre-compiled queries are not rendered inoperable by database

changes such as deletion of views or schema changes, or at least to ensure that the validity of such pre-compiled queries is flagged when these changes occur. This problem becomes increasingly difficult in distributed situations where data relocation possibilities and local autonomy issues increase the need for flexibility, while at the same time make it harder to keep track of the pre-compiled queries that may be stored remotely.

The question of query optimization has received considerable attention both in the literature and in terms of strategies utilized in DBMS and DDBMS implementations. Since the cost of explicitly forming the Cartesian product in the processing of a query involving a join is very costly, a number of other techniques have been devised. The "nested loops" method proceeds in a manner analogous to forming a Cartesian product, but for each tuple in one table the relation to be joined to it is scanned serially. The "merge-scan" technique relies on sorting the relations on the join attribute, and performing the selections in the read and merge phases. The "semi-join" technique involves breaking the query processing steps into a projection of the attribute or attributes of interest, and communicating only the restricted subset of the information to another site to be joined to another relation, after which the results are sent back to the originating site in order to complete the join.

All of these techniques for query optimization rely on the effective use of available information. Where such information (order being considered to be information as well) is unavailable, it is often more efficient to create the information rather than to proceed without it. Query processing in a distributed environment must contend with the same problems of concern to local DBMS processing, plus the problem of the cost, both in time and

dollars, of data communication. In a distributed environment, the choice of optimization technique and the order in which actions are carried out influence not only the total number of file read and write operations, but also the amount of data and the number of messages to be transferred from site to site. Information which may affect the choice of query optimization technique include the network parameters. For example, in a wide area network, the cost of transmitting an entire relation may be prohibitive, but this is not necessarily the case for a local area network environment.

It has been shown that for the general case the query optimization problem is equivalent to the traveling salesman problem, that is, it is computationally intractable to achieve perfect results [12]. However, algorithms can be devised to achieve satisfactory results if sufficient information is available to the optimizer. The relational model is amenable to large-scale non-procedural operations, and has proven to be much more suitable to optimization than hierarchical or network data models, on which data manipulation language operations are more procedurally oriented and navigational in nature.

Adequate performance must be considered by any viable DDBMS implementation, and is integrally related to the degree of data replication imposed by individual applications using the DDBMS implementation. In distributed database systems used in support of real-time operations, such as teleconferencing, performance considerations may become a paramount concern. The requirement common to all DDBMS implementations is that adequate information be available regarding the location of information at various sites of the network, so that optimization can occur.

3.9 Heterogeneity

A homogeneous DDBMS refers to a DDBMS with the same DBMS at each site, even if the computers and/or the operating system are not the same. A heterogeneous DDBMS is defined to involve at least two different DBMSs. If distributed database systems are to be developed in a top-down manner with no regard for existing DBMSs (that is, not the federated DDBMS situation), then it is preferable, from the perspective of the implementing vendor, to make such distributed databases homogeneous in nature. However, from a user perspective, homogeneous DDBMSs do not address the problems of integration faced when attempting to federate distributed databases from disparate existing databases. It has long been recognized that the problems of heterogeneous DDBMSs, particularly the aspect of mapping between different data models, are very difficult. This has resulted in the current situation where most heterogeneous DDBMSs are research prototypes, while commercially available systems that provide some degree of support for heterogeneity typically can afford only limited support for data model translation.

Since user and vendor requirements do not permit us to ignore existing investments in database technology, it cannot be assumed that distributed databases can be built in a top-down manner. It is clear that a requirement exists for providing mechanisms to support heterogeneity.

3.10 Requirements for Export/Import

In order to share all or portions of database information, including not only application level data but higher data levels (dictionary and dictionary

definition level information), there exists a requirement to support the transfer of data between data management systems without establishing any connection between these systems. This is the type of functionality normally associated with database unload and reload utilities. However, export/import involving differing DBMS implementations dictates some standard format for the transfer of this data if it is to be intelligible to more than one DBMS. Export/import facilities are particularly important in permitting the sharing of analysis and design information between dictionary systems and software engineering tools. There also exists a requirement for a stable format for storage of database information for archival purposes.

3.11 Requirements for Support of Additional Data Types

The requirements for access to information which is "unstructured" in the way that database information is normally thought of as being structured (that is, as separately defined columns within a relational table) are continuing to increase. Electronic documents containing numerous types of "structure" information, are becoming more prevalent and include not only text but scanned images, geometric graphical objects and digitized voice. The requirement to integrate information available in different forms and stored on different media has resulted in the need to store this data in the increasingly economical storage devices used by database systems. Management of this information in a database demands the subsequent storage of control information about the information resources in data dictionary systems.

Since most database systems support neither the storage and retrieval of information nor operations (for example, proximity searches) on these data types, there is an growing requirement to provide these functions. This requirement exists in both centralized DBMS and distributed database environments.

Chapter 4

Review of Relevant Standards Potentially Useful in DDBMS Standardization

4.1 Introduction

The chapter will review standards having an impact on distributed database technology. The review will not be an in-depth evaluation, but will point out the salient features as they relate to the design and implementation of distributed database management systems.

4.2 Database Language SQL

Standards produced by the International Organization for Standardization progress through four recognized stages of development, reflecting their degree of technical stability. These stages are the *Working Draft*, *Draft Proposal* (DP), *Draft International Standard* (DIS), and the full *International Standard*. The standard for database language SQL reached the full international standard stage in June 1987, and was published by ISO as standard ISO 9075-1987. Subsequently it has been published as a national standard by a number of countries participating in the ISO work, including Canada [57] and the United States (as an ANSI standard).

The initial base document for the SQL standard was contributed to ANSI by IBM, and work on this standard was initiated in ANSI database committee X3H2 in 1982. ANSI had been concentrating on the

standardization of a Network Database Language since the formation of the committee in 1978. Both projects were submitted to ISO shortly thereafter, and further development subsequently has been synchronized between ISO, ANSI and other countries participating in standards development.

The SQL standard [57] specifies the syntax and semantics of two separate interfaces to the database control system. The first is the schema definition language, which is used to declare the structures, integrity constraints, and view declarations of a relational database. The second is a module language containing data manipulation language (DML) statements for declaring the executable procedures of particular database sessions. The module language is the underlying interface to the database from external languages, and may be supported directly by implementations. Alternatively, conformance to the SQL standard may be claimed on the basis of conformance to an embedded syntax for inclusion of SQL DML statements in COBOL, FORTRAN, Pascal, or PL/I programs. Such hybrid application programs are defined to be equivalent to standard-conforming programming language programs, where the SQL statements have been replaced by standard procedure or subroutine CALLs of SQL procedures in a separate SQL module.

The SQL standard defines a number of data types that fit into three categories: character string, exact numeric, and approximate numeric. The standard specifies the data structures and operations of the relational data model, including capabilities for defining, querying, protecting and altering databases. The primary data structure is the table, defined in terms of rows and columns. Tables may be base tables, having persistent storage representation and table description, derived tables, or viewed tables. A

derived table is an ephemeral table created from one or more base tables during the execution of a database statement. A viewed table is a derived table that has a persistent description.

Every utilization of an SQL standard-conforming database occurs with conceptual reference to a module, whether or not the module language is explicitly supported. Each module contains a language clause, specifying one of the four supported programming language environments in which the module will execute. The manner of communicating data between the database control system and each procedure of a module is dependent on the language environment of the containing module. Thus the SQL standard explicitly defines the mapping between database defined data types and the data types defined within supported programming languages. Figure 4.1 shows the elements addressed by the SQL standard and those that are not.

Since the publication of the ISO 9075-1987 standard, an Addendum to the standard has reached the full International Standard level, and has been incorporated into ISO 9075-1989. This addendum, published as a part of [57], specifies integrity enhancements in the form of UNIQUE, CHECK and referential constraint (FOREIGN KEY) definitions. The search condition associated with a CHECK constraint definition is prohibited from containing a subquery.

Continuing work on relational database language standards has resulted in the registration of SQL2 as a Draft Proposal. SQL2 is a forward-compatible enhancement of [57] and includes a large amount of additional functionality when compared with the original SQL standard.

The document is much larger than [57], and contains the following additional facilities:

- additional data types (date and time, for example);
- national character sets are supported;
- the ability to define collating sequences (which relates to the ordering for comparison predicates and order clauses for cursors);
- a domain facility is specified;
- additional integrity constraints and cascade rules are specified;
- additional predicates are supported;
- embedded SQL support is specified for Ada and C;
- access to schema information tables is provided;
- provision of "Dynamic SQL", which extends SQL for interactive execution of SQL statements not known at the time that an SQL module or embedded syntax program is created;
- a schema manipulation language facility is provided (alter and drop tables, revoke permissions); and
- additional transaction management facilities are defined.

While the SQL2 specifications are not fully upward compatible with SQL, the situations where upward compatibility is lacking relate to implementor-defined features in the original standard which are more fully specified in SQL2. The SQL2 specification is expected to reach DIS status in July 1991, and full International Standard status in June 1992.

Features which may be noted concerning SQL are as follows:

- SQL is not merely a "query language", as its acronym "Structured Query Language" implies. It is a full featured data manipulation language;
- SQL is but one possible language for describing the data structures and operations of a relational data modelling facility. Whether it is the optimal relational language is debatable. In any case, the extensive capabilities of SQL and the degree to which it has been accepted in the marketplace secure its place of importance in the future of database technology in both local and distributed environments;
- Although the dynamic SQL facility of SQL2 is beginning to provide interactive access facilities which are more amenable to access in a distributed system, the capabilities of both SQL and SQL2 are tied in closely with the data type specifications of programming languages;

- The power of the relational facilities provided by the SQL Database Language, both in terms of the ability to specify integrity verification parameters, and in terms of the capabilities and data types supported by the DML, is continuing to increase, and can be expected to do so in the foreseeable future.

4.3 Database Language NDL

Reference [58] defines the data structures and operations of a network database language. Although standardization work on the network database standard began much earlier in ANSI X3H2 (in 1978), publication as an ISO standard did not occur until 1987, at the same time as the SQL standard. The NDL specification originated from prior work in CODASYL, including the preparation of an original Data Base Task Group report, and subsequent work by the Data Description Language Committee of CODASYL.

Most of the work done within ANSI and ISO on the original CODASYL proposals focused on increasing the power of the network-structured data modelling facility while attempting to provide for greater data independence. Much of the work on the latter objective centered on the removal of physical constructs (such as AREAs) from the specification. However, the arrival of the relational approach signaled the beginning of the end for the network database language. The attributes of the relational model have been shown superior to those offered by network and hierarchical implementations, particularly as performance optimization work continues on relational implementations. The total lack of products for the NDL database language standard eliminate it from further serious consideration in both local and distributed database environments.

4.4 Information Resource Dictionary System

In 1980, both the American National Standards Institute and the National Bureau of Standards of the United States Department of Commerce initiated efforts to develop standards for data dictionary software. The ANSI and NBS work was integrated in 1983 when the ANSI X3H4 committee voted to adopt the draft Federal Information Processing Standard for Data Dictionary Systems as its base document. An American National Standard Information Resource Dictionary System standard [59] has now been finalized and adopted as a FIPS standard for the United States federal government.

Realizing the importance of data dictionary system standards, the International Organization for Standardization approved a project addressing Information Resource Dictionary System standardization in January 1984. An Information Resource Dictionary is defined as a repository for data about information processing systems. Such information resources may include any type of information of interest in an information processing environment. An IRDS provides facilities for "extensibility", that is, for defining entities of interest for which occurrences will exist in the IRDS.

The ANSI IRDS standard is based on an entity-relationship data model, permitting only binary relationships. This data modelling facility is defined as the first half of a very lengthy document (approximately 1200 pages). The utilization of the binary-relationship model in fulfilling dictionary system requirements is then specified in the remainder of [59].

Figure 4.2: IRDS Processor Interfaces (source [45])

The introduction of the ANSI document into ISO resulted in prolonged debate on the merits of the selected E-R data modelling approach. There was general satisfaction with the requirements addressed by the ANSI specification. However, the manner in which ANSI dealt with these requirements, and the potential degree of difficulty of integrating the IRDS work with database language standards, were extensively criticized. Participants in the ISO work also questioned whether it was appropriate to specify a "line at a time" command language, or a panel (screen) interface, as the interfaces to be standardized.

The result was that participants in the ISO work opted for the specification of a "services interface", specified in terms of language bindings to standard programming languages [39]. The ISO work also diverged from the ANSI work in terms of the data modelling facility used. The ISO IRDS was specified in terms of an object-association data model that was closer in expressive power to the relational data model, although differing from the SQL data modelling facility. Although many attempts were made to reconcile the ANSI and ISO IRDS standards, it became clear that such reconciliation, at the level of the underlying data modelling approach, was impossible.

Figure 4.2 shows the major IRDS processor interfaces (see section 6.2 for a description of the symbols in the Figure). The ISO standard addresses the IRDS Services Interface, while the ANSI standard specifies a panel (screen) interface, and a command language designed as well for a human user but usable by another automated processor. Subsequent to the initial ANSI IRDS work, and recognizing the importance of a services interface to

IRDS functions, ANSI has begun work on a services interface standard compatible with the other ANSI IRDS standards.

Figure 4.3 shows the levels of data of concern in an IRDS environment. Application level data is not of direct relevance to the IRDS, although the metadata stored in an IRDS relating to data types may control the instances stored as application level data. The Information Resource Dictionary (or simply dictionary) level data is the prime area of concern for IRDS standardization. This is where schema level information for application databases is stored. As well, other analysis and design information relevant to the information systems analysis, design, development and operational phases may be stored at the IRD level. The form that this information will take depends on the methodology being used in developing information systems. To provide flexibility in defining the types of entities about which such information is stored, IRDS systems provide for "extensibility". Extensibility is the ability to create types at the IRD Definition level for which occurrences will exist at the IRD level. The fundamental level is visualized in [45] as the level for defining the constructs (data modelling approach) utilized at the IRD Definition level.

Figure 4.3: IRDS data levels (source [45])

4.5 Remote Database Access

The Remote Database Access (RDA) standards under development in the International Organization for Standardization are intended to support interworking between an application program in one system and a database management system in a communicating system. As with other OSI standards, RDA defines the peer-to-peer protocols, while the local program interfaces to the communication facilities are implementor-defined.

The RDA standards have been divided into a Generic Remote Database Access Service and Protocol [40] and an SQL Specialization of RDA [44]. The generic standard defines the common aspects of remote database access, with the possibility of several specializations, each addressing a different type of database access. The generic RDA standard provides a template for an abstract class of RDA applications, and is not directly usable without a corresponding specialization. The specialization references a particular database language standard and stipulates the types and format of data to be carried in argument and result parameters. Currently, the only specialization being standardized is the SQL specialization. Other specializations such as the bibliographic retrieval application are currently being explored.

The remote database access standards define in detail the communications aspects concerned with database access, and reference other standards for all other factors. RDA standards reference database language standards for the data management aspects, and OSI application layer and presentation layer standards for the communications features.

The RDA work began in the European Computer Manufacturers' Association with the publication of ECMA Technical Report 30 in December

1985. At that time ECMA proposed the establishment of a new work item in ISO, and the work was assigned to the ISO Database Working Group, SC 21/WG 3. Since then, much of the work within ISO has focused on ensuring that other OSI facilities were optimally exploited, rather than re-specified. The RDA specifications in ISO are at the Draft Proposal level, and thus will not become full International Standards for approximately two years.

One of the major changes made to the RDA specifications since the ECMA work has been the use of Remote Operations model, notation, service definition, and protocol specification ([16] and [22]). The Remote Operations standard uses an object-oriented execution paradigm in which the functionality of an object is modelled as a set of operations available at the object's interface. The Remote Operation Service Element (ROSE) is designed to support many types of interactive applications in a distributed open systems environment. In the OSI environment using remote operations, communication between application processes is represented in terms of interactive exchanges between a pair of application entities (AEs) using the presentation service. Operations invoked by one AE are performed by the other AE. Operations may be classified according to two possible operation modes. In synchronous operations, the invoker requires a reply from the performer before invoking another operation, while the invoking AE may continue to invoke further operations in asynchronous mode.

The Remote Operations notation is based on the macro concept of Abstract Syntax Notation-1 (see section 4.12). Application Service Elements available to the user element of the AE at the operation interface are defined using the Remote Operations notation. This facility provides applications with independence from OSI communication services. RDA

uses Operations Class 2, which is an asynchronous class reporting success (with a result) or failure (with an error returned). The asynchronous operation mode is the main difference between a ROSE operation invocation and a programming language procedure call.

Since RDA makes use of the Remote Operations notation in its definition, implementors are required to support the ROSE service and protocol. The benefits of using Remote Operations notation and service are similar to those of using high level programming languages. Aside from the considerable reduction in the resulting size of the RDA specification, automated tools are available to process the formal definitions and generate the software supporting an RDA service. The code thus generated will, of course, depend on the intended execution environment.

Figure 4.4 shows the structure of two intercommunicating RDA implementations. One such implementation must support the database server, while the other is assumed to be a workstation containing a client program that requires access to the database in the database server. Such interactive interworking is achieved by making calls on the RDA Application Service using an interface defined by the implementor. The RDA Application Service is envisaged as including all of the hardware and software necessary to support the client/server communication.

Within the RDA Application Service, the client and server communicate via a pair of peer protocol machines, which exchange data through the service provided by the Presentation Layer. The RDA standards define the structure and meaning of the protocol messages exchanged by means of the presentation service. The functions provided by the RDA

service may be grouped into three classes: dialogue management, transaction management, and database access.

Dialogue management makes use of the Association Control Service Element (ISO 8649) to establish an association between peer application layer entities. This service, which is a Common Application Service Element used by most application layer standards, communicates common parameters such as identity of caller, account codes, and Presentation Address. The Presentation Address is assumed by the service to be known to the RDA client. Establishing the Presentation Addresses of appropriate database servers is a major difference between remote database access,

which may be viewed as a "post office", and a full DDBMS implementation. A dialogue in the remote database access standards may last for longer than a single association, and thus permit the re-establishment of an association that has been interrupted for any reason.

Transaction management in RDA is dependent on the Application Context. The Basic Application Context provides no special protocol support for transaction management, and would be inappropriate for distributed interworking involving multiple clients wishing to update a database server concurrently. The Commitment, Concurrency and Recovery standard (see section 4.10) provides a facility enabling transaction management when a transaction may involve a set of end systems. A third application context is designed to allow remote database access standards to be used in conjunction with the evolving ISO transaction processing standards.

Figure 4.5 depicts the functions associated with an implementation of RDA in the context of a relational database query processor. Beginning with the formulation of a query by a user, processing follows a sequence of operations following down the left side of the Figure and up the right side. The dotted box entitled "RDA service" represents the approximate division between the functions that are executed in the workstation and those executed in the server. The RDA facility provides the mechanism for communication between the accessing workstation and the remote database server.

The work of analyzing and executing a client query is split between a front end in the workstation, and the back end in the server. The exact division has been subject to some modifications during the development of the RDA specification for the SQL Specialization. Initially, an ASN.1

encoding of SQL DML statements was indicated. This was followed by optional textual representation of SQL statements, and has recently been modified to prohibit the ASN.1 encoding. Skepticism by the implementors of RDA servers of the quality of the syntax and semantic analysis performed in the client workstation is the primary reason for these changes. Similar considerations are common in the design of reliable distributed systems. Analysis of text-form SQL statements may involve additional overhead when such statements are syntactically incorrect. This disadvantage is counterbalanced by requiring the client to have less information about the server database.

The RDA service permits a user to use the same front end to access several different databases. Similarly, a single database server may be shared by a number of clients with different workstations and styles of man-machine interface.

RDA specializations may support more than one form of data representation, and the actual form to be used may be negotiated when the association is established. This facility allows private encodings for use between compatible or identical implementations, as well as a standard representation for open interconnection. Database language standards do not define the representation of data types to be supported by conforming DBMSs as bit strings, but rather in terms of their mappings to programming languages. For exchange of database data the encoding of data values must be agreed between the client and server. Abstract Syntax Notation One (ASN.1) is specified as a mechanism to accomplish this for open systems interworking. ASN.1 is widely used in ISO and CCITT for

defining data types and values, and provides a standard encoding [27] for these values.

4.6 Directory Services

The Directory Information Base specified by the Directory Services series of standards [68-70] is intended to facilitate communication between, with or about objects such as application entities, people, terminals and

distribution lists. The Directory function provides capabilities for "user-friendly naming" and for name-to-address mapping, which allows the binding between objects and their locations to be dynamic. The Directory is not intended to be a general-purpose database system, in that a number of assumptions are made as to the nature of the application being supported. It is assumed that there is no need for global consistency of data across an OSI network, and that there is a considerably higher frequency of queries than of updates.

The success of OSI interprocess communication standards motivated the need for the Directory Services standards. These standards define a global distributed database system, designed jointly by ISO and CCITT. Attempting to develop a specification for standards addressing the Directory Services requirements with currently available technology was an ambitious undertaking. Compromises made in order to produce viable specifications have resulted in a number of weaknesses in the current standards, many of which are being addressed by ongoing standards efforts.

The current Directory Services standards support only a very limited number of operations. The Directory Services specification is also lacking an access control facility. Other problems center on the lack of flexibility of access mechanisms. For example, it is necessary that an entire name, not just a portion of the whole name, be known in order to ascertain the network address. This has brought about the requirement for shorter "nicknames" to be used. As another example, there is a requirement that the Directory Services standard not only provide a global automated "telephone book white pages", but also a "yellow pages" facility. A requirement for such yellow pages functionality could be to determine the print servers in close

proximity that were attached to a network and able to support certain required features (such as accepting Postscript format). This type of search, using several attributes and involving boolean logic, is addressed adequately as a database access requirement by most DBMSs. Also, the current Directory Services standard does not encompass the requirements addressed in section 3.6 for transaction management - requirements which are addressed within database standards by demanding serializability of transactions.

The lack of functionality and flexibility of much of the current Directory Services standard can be directly attributed to the limitations of the data model which has been utilized - the Directory Services standards use a hierarchical data modelling facility. While the inherent weaknesses of the hierarchical data model relating to such capabilities as flexible access criteria have long been appreciated by database practitioners, the developers of the Directory Services work, typically communications engineers, may have been less than totally familiar with the additional power offered by a relational data modelling approach. In addition it must be appreciated that the state of development of relational database standards was not far advanced when the Directory Services work began.

The Directory Services standard provides access capabilities to a certain specialized type of information that is of interest in an open systems environment. The Directory Services function is basically a standardized distributed database management application. As a result of the importance of this standard in facilitating national and international communications, and as a result of the relatively slow progress of distributed data management standards in relation to OSI standards, the Directory Services

standard has proceeded in isolation of database standards and in advance of DDB standards efforts.

Attempting to overcome some of the limitations of the existing Directory Services standards may result in duplication of effort with standards work relating to general-purpose distributed data management. It is ironic that this is occurring at a time when implementors of Directory Services standards are considering the usefulness of relational database facilities as an implementation vehicle. Thus more general purpose database facilities are being used to provide the restricted subset specified in the Directory Services standard, when in fact the direction being considered by current work on the Directory Services standards is to allow more flexible general-purpose mechanisms. Additionally, one of the requirements to be supported within general purpose distributed database management systems is the need to transform a "logical" site name into a physical network address for the establishment of an association. This is the type of information that the Directory Services standards were designed to provide.

4.7 File Transfer, Access and Management

The File Transfer, Access and Management (FTAM) standards ([63-66]) are OSI application layer standards. They allow data to be added to or removed from an information storage facility, and the description of the data maintained, without knowing how the filing system is implemented. FTAM relies on the presentation layer of OSI to resolve syntactical differences between open systems, and on the session layer to manage the dialogue between open systems.

The FTAM standards reference a virtual filestore to provide independence from the incompatibilities of a wide range of file systems. The virtual filestore is a conceptual model supported by conforming implementations, regardless of the filing systems used in the Local System Environment (LSE) of end systems. The FTAM standards define the universe of discourse for a communication utilizing FTAM standards. The virtual filestore contains many of the attributes found in a variety of real file systems. Any system conforming to the specified file protocols in the role of responder is considered to provide a virtual filestore. The FTAM standards do not specify the interfaces to filing systems within the local systems.

Figure 4.6 shows a model of the FTAM service. The service supports a set of actions on the virtual filestore which is determined by the file

attributes, the current state of the filestore (e.g. as regards concurrent access), and the negotiated parameters established when the data transfer regime was initiated. FTAM supports the exchange of accounting information and an access control facility based on the concept of an access control list. It provides concurrency control at the file level by means of declaration of intentions (such as shared or exclusive access). It also supports a finer level of granularity of locking, at the File Access Data Unit (FADU) level. A FADU is a component of a hierarchical structure in which the number of levels and the number of arcs originating from each node are not restricted. Each subtree, including all data units contained within the subtree, comprises a FADU. It is possible to use FTAM in conjunction with CCR (section 4.10) to provide a recovery mechanism.

The FTAM standard [63] contains a non-normative annex giving an example of the use of FTAM to support remote database access. This annex is somewhat misleading, in that the ability of FTAM to support remote database access is comparable to the ability of file systems to support database operations. FTAM may be of value in the export/import of database data (see section 3.10), but its usefulness in support of distributed databases is limited.

4.8 OSI Management

OSI Management is concerned with the operations and communication associated with the management of resources within the Open Systems Interconnection Environment. OSI Management defines the characteristics of the resources to be monitored, controlled, or coordinated within a network, and is the OSI equivalent of what is commonly referred to as "Network Management" of communication networks. This family of

standards also defines the communication protocols for exchanging management information between systems, and the structure of the information that is maintained in the "Management Information Base".

Management of the communications environment is a distributed information processing application. As with other OSI-defined facilities, the communicating end systems are assumed to be completely heterogeneous in nature. Thus the operations and notifications communicated between the "manager" role and the "agent" role are defined abstractly and without any reference to the manner in which management information is represented and stored in a real open system.

A managed object is the OSI Management view of a system resource that is subject to network management, such as an entity of one of the OSI layers, a connection, or an item of physical communications equipment. OSI Management capabilities are categorized as belonging to one of five functional areas: fault, configuration, accounting, performance, and security management. The standards relating to these areas define the "universe of discourse" of the shared communication between a manager and an agent. The managed resources are modelled as objects, with attributes, that can be manipulated by operations.

The distributed systems approach taken by the developers of OSI Management is similar in nature to the approach used in specifying the Directory Services and FTAM standards (sections 4.6 and 4.7). A protocol-driven object-oriented approach was selected, which models both data and process constructs in defining the behavior of managed objects. All three of these standards incorporate the definition of "data modelling facilities", i.e., they define the data types and the permitted operations on

such data types. However, the information communicated between end systems could equivalently be represented, in the specification of the standards, by a general-purpose data modelling facility such as is prescribed by the SQL standard. Similarly, the operations defined by OSI Management and the other standards could then be conveyed between communicating systems by means of the exchange of data values which were understood between both partners in a communication. The OSI Management standards seek to define a mechanism for determining the content of protocol messages between communicating sites for network management purposes. An alternative approach is to reference a common data model (such as SQL) and to use a standard communication mechanism (such as RDA), with the consequence that the complexity of OSI Management standards would be considerably reduced. Thus the "manager" role could function in RDA-client mode, sending appropriate data or requests to an "agent", which acts in RDA-server mode. The server would respond to the client with appropriate notifications and data (such as the value of an error counter).

Directory Services, FTAM, and OSI Management are all standards made overly complicated by the lack of adequate standards for distributed data management. However, the nature of the application areas is fundamentally no different from any other distributed application, and the suggested "data directed" approach has the potential to facilitate interworking between cooperating real open systems while significantly reducing the burden on both standards developers and implementors. Note that it is not necessary that each such communicating end system understand the data structures of SQL in order for this to occur. The suggested approach provides an appropriate mechanism for specifying the

structure and content of the protocol messages which are exchanged between systems. The processing involved in reacting to incoming protocol messages remains unstandardized as a part of the Local System Environment. This does not, however, preclude vendors from defining a database based on the data elements standardized as a part of application areas such as OSI Management, and controlling the communications environment by means of behavior triggered by the current status of database records.

4.9 Abstract Service Definition Conventions

The manner of specification of both data management standards and of OSI standards has been the subject of considerable discussion and controversy for some time. OSI specifications designate the peer-to-peer protocols which precisely define the exchanges of protocol data units (PDUs) between peer layer entities. However, the interface to OSI services in a particular end system has not been specified in a precise manner, and the method of specification of the services available has differed for individual OSI standards.

Many OSI-defined standards have been specified with reference to ISO Technical Report 8509, which outlines OSI Service Conventions. The natural language specifications of OSI services have frequently been augmented by the use of more precise definitions using Formal Description Techniques (FDTs). FDTs such as LOTOS, Estelle and SDL have been developed in order to permit the formal specification of communication protocols. An important advantage of FDTs, aside from greater precision in the specification, is that these specifications are amenable to the use of automatic tools for implementing the protocol. So far, FDTs have not been

used in the specification of any data management standards. However, work is underway at the U.S. National Institute for Standards and Technology in Washington to provide an Estelle specification of Remote Database Access.

The FDTs mentioned are useful in the definition of communication protocols, but specification is also a concern for database services or for the local invocation of OSI facilities. To simplify the development of a store-and-forward Message Handling System within CCITT and ISO, the Abstract Service Definition Conventions standard [30] was developed. This specification allows the abstract definition of the services available from objects at identified ports. The concrete realization of the service then depends on the context in which the service is invoked. The provision of abstractly defined services in an open system environment is provided by the direct mapping to Remote Operations (see section 4.5) such as bind, unbind, and operation. However, these abstractly defined services may be invoked in a local system environment by means of vendor-dependent realizations.

An advantage of the use of the Abstract Service Definition Conventions is the automated generation of protocol elements. A disadvantage of the use of the Abstract Service Definition Conventions, which are based on the macro notation of ASN.1, is that the semantics of the communicated data types can only be specified as comments. This form of specification is less precise than is achievable with FDTs for communication protocols.

The Abstract Service Definition Conventions work is potentially useful in allowing a single specification of data management services for invocation either locally or remotely.

4.10 Commitment, Concurrency and Recovery

A distributed application is represented in OSI by the joint cooperation of application entities (AEs) interconnected by application associations. The Commitment, Concurrency and Recovery (CCR) service definition and protocol specification ([60] and [61]) provide a means of coordinating the activities of separate application exchanges by defining services for starting and ending semantic exchanges and related activities. The purpose of the CCR standards is to provide a mechanism to support distributed application processing performed as sequences of operations having all of the characteristics of "atomic actions" (see section 3.6). The CCR standards support such processing even in the presence of communication and application failures. CCR is an OSI Common Application Service Element specification and is used only in conjunction with other referencing application layer standards.

The use of the CCR standards requires that cooperating application entities organize their activity into a tree structure. An atomic action tree (depicted in Figure 4.7) may be statically pre-defined, or constructed dynamically using CCR services by including new branches as the atomic action develops. An atomic action tree is a hierarchical relationship between CCR service users. CCR prescribes the protocol interaction between a single pair (comprising an atomic action branch) of communicating sites. In the event of application or communication failure, the atomic action branch may endure and recovery may proceed with a subsequent association. It is the

responsibility of the related application layer standard to provide the higher level control function over multiple associations associated with the transaction tree.

Figure 4.7: An atomic action tree

The scope of the CCR service is to support the initialization and completion of a single atomic action branch. In conjunction with the facilities offered by the referencing application layer standard (such as the remote database access standards), CCR provides a means of ensuring that the "ACID" properties of a transaction (see section 3.6) are maintained. This is done using a two-phase commitment protocol, which in CCR operates individually on superior-subordinate associations. In a two-phase commit, the controlling (superior) site sends a *prepare* order to all of the nodes to which it communicates directly. For example, in Figure 4.7 site M would issue CCR *prepare* requests to sites A, B, and C. Site A would not respond with a CCR *ready* protocol message until it had confirmed that sites D and

E were ready to commit; D would ensure that site H was ready to commit, etc. Each participating site must acknowledge the *prepare* with either an *accept* or a *reject*; if it *accepts*, a site may not thereafter unilaterally abort its part of the transaction. If all participating sites responding directly to a superior at the root of the tree acknowledge with *accept*, the controlling site will issue a *commit* order to all direct subordinates. If any site responds with *refuse*, or does not respond within a time-out period, the controlling site must issue a *rollback* order. *Rollback* should be clearly distinguished from *restart*, which is an element used to recover from application and communication failures.

4.11 Language Independent Data Types

As defined in [22], "A syntactic metalanguage is a notation for defining the syntax of a language by using a number of rules. Each rule names a part of the language (called a non-terminal symbol of the language) and then defines its possible forms. A terminal symbol of the language is an atom that cannot be split into smaller components of the language." Since the time of Algol 60, the syntax of programming languages has been defined formally using syntactic metalanguages. Algol 60 was defined using a notation now known as BNF or Backus Naur Form. Although this has proved a suitable basis for subsequent programming languages, it has frequently been extended or altered. Current programming languages generally use some form of syntactic metalanguage for the specification of the elements of the language, including the definition of the data types supported by the language. Similarly, database language standards use a variation of BNF as a syntactic metalanguage in defining the elements and data types supported. Relating database language data types to

programming language data types requires the correlation of the syntactic metalanguage specifications used in defining the data types of the two facilities. It is then necessary to state correspondence rules for moving these data types between database and programming language environments. The problem with this approach is that it is a "gateway" approach (see Figure 2.1), in that the correspondence between programming language data types and database data types must be resolved individually for each programming language for each data type supported.

The correspondence thus established between database and programming language data types is at the syntactic metalanguage level. There is no concern for the actual encoding of the data within any particular implementation which permits access to database facilities from programming languages. To implement a "services interface" to database services efficiently, it is necessary to avoid the "gateway" approach to mapping between the data types of common system facilities and programming language data types. For distributed interworking it is also necessary to be concerned with the precise encoding of such data types for transmission.

In recognition of the extent of the work required to specify and implement individual mappings between the data types of system facilities (graphics facilities, database facilities, IRDS services) and numerous programming languages, a work item addressing the question of common language-independent data types was established in 1985 in the Languages Subcommittee of ISO/JTC1. The intent of this work is to avoid the necessity for numerous conversions by specifying a common set of data types. These

need not be directly incorporated into various services and programming languages, but each facility should map individually to the common set.

In establishing an appropriate set of common data types, the ASN.1 specifications [26] deserve careful consideration. ASN.1 specifies supported data types using a variant of BNF as a syntactic metalanguage. ASN.1 has proven to be a very useful type-length-value specification for data interchange for Open Systems Interconnection.

The direction proposed by the common language-independent data types project appears appropriate, although the results have not yet eliminated the need for numerous data type mappings.

4.12 Abstract Syntax Notation One

Application layer standards communicate by means of sets of data elements of different types defined for specific applications. Application layer standards do not define the encoding (binary representations) of these data elements, either as used in a specific computer system, or when being transferred between systems. Specification of such encoding is not a function of application layer standards.

For two systems to be able to interchange data, they must agree on the manner of representation of the data to be transferred. This can be done for simple data structures by using protocol elements to indicate the structure, or by embedding delimiters within the data. This approach is not practical for complex data structures, and it is preferable to separate the definitions of the data structures from the communications protocol.

One way of doing this is to define the data structures in terms of an *abstract* syntax [26], and to utilize a common method of encoding data

elements defined by such an abstract syntax into a *transfer* syntax for the communication [27]. Such a transfer syntax may differ from the *concrete* syntaxes used to represent information in either of the communication systems. In the case that the local syntaxes of communicating partners are the same, presentation layer facilities allow for the negotiation of a transfer syntax from the set known and supported by the systems concerned. Such a syntax need not correspond to the local syntax used by either of the end systems. It is essential, if communication is to occur, that both systems agree on a transfer syntax and be capable of transforming information from their local syntax to the transfer syntax selected.

Presentation layer standards provide capabilities relating to the data encoding and transformation rules used over a particular presentation connection. They enable distributed application processes to communicate without concern for the differences in concrete syntax employed by each of the communicating systems. If the two sites involved in an association cannot communicate using the same code set (e.g. EBCDIC to EBCDIC), then the sender would map to the communication service using the ISO 8825 encoding defined in [27]. The receiver would subsequently translate from this transfer syntax into its required encoding. ASN.1 standards define character sets by reference to other ISO standards.

In principle, the syntax rules used by a facility such as ASN.1 must make it possible to represent any value of any conceivable data type without ambiguity. Thus a transfer syntax must contain provisions for recursively defined structures, whose components are also structured. Such provisions are applicable, for example, to the sending of relational tables as data structures. The ASN.1 standards achieve this by permitting data types to be

defined as either *primitive* (without internal structure) or *constructor* (made up from one or more distinguishable components).

Currently only one standardized notation for an abstract syntax and one set of encoding rules exists. It is possible that further encoding rules will be defined in the future to address requirements for efficiency or encryption.

4.13 Reference Model of Data Management

Since the early 1980s, efforts have been underway within ISO to prepare a reference model establishing the future direction of data management standardization. This would be done in the same manner in which interprocess communication standards have been guided by the OSI Reference Model [52]. The objective of this work is to establish a framework for coordinating the development of existing and future standards for the management of persistent data in information systems. The scope of this standard was initially limited to a single site environment, but has recently been widened to include consideration of distributed databases. This work is currently in the Draft Proposal stage in ISO [5].

In pictorial terms, the purpose of the Reference Model of Data Management is to prescribe the standards needed to specify the data management "block" of functionality depicted in Figure 5.1. It should also establish the manner of interaction of the required data management functions with the other components in Figure 5.1.

Chapter 5

Design Considerations for Standardized Components of Distributed Database Management Systems

5.1 Introduction

This chapter submits the data management requirements identified in Chapter 3 to an analysis/design effort, while taking into consideration the facilities provided by the existing and developing standards reviewed in Chapter 4.

The standards needed to support distributed database management relate to both data management and interprocess communication, and must build upon existing standards in these areas. Their degree of complexity makes it unlikely that comprehensive standards will emerge without prior agreement on a framework. This framework must consider the requirements which these standards must address and summarize a workable "design" that incorporates existing OSI, programming language, operating system and data management components. The considerations documented in this chapter provide the overall parameters for the design of an architectural model incorporating the needs of both centralized and distributed databases; such an architecture is suggested in Chapter 6.

2

of/de

2

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

5.2 Consideration of Database Access Requirements

5.2.1 Consideration of Programming Language versus Distributed Access Requirements

Of the three programming language access alternatives, the SQL standards support the explicit procedure call approach and the implicit procedure call approach (in which a preprocessor extracts flagged data manipulation language statements that have been written into the program source code). In the native syntax approach, the programming language is extended to incorporate database data manipulation capabilities in a manner that is natural to the programming language environment. All three approaches assume the availability of shared memory and the ability to link compiled programs with database modules for local execution. Such assumptions are not valid in a distributed systems environment. Currently, the SQL database language standards prescribe the data transformations which must take place to map between standard programming language data types and the database supported data types. The SQL language is defined in terms of the syntax and semantics of data definition and data manipulation statements in source form. SQL defines a module as a persistent object which includes one or more procedures (each containing one SQL statement), and a language clause that specifies the programming language environment in which the module is to be invoked. A single module is associated with one compilation unit (a segment of executable code) during its execution. The manner of specifying the association between the module and the compilation unit is implementor-defined. Such

an approach evolved from the original designs of database management systems; there was an overriding requirement to provide for access from a number of different programming languages if data was to be shared among multiple users.

While such an approach is workable in a single site database environment, it is less useful in a distributed environment, where the question of the location of the "SQL module" must be addressed. Since the SQL standards assume interface with a standardized programming language environment, and that environment needs to exist at the client end of a remote database access association, the fact that the "SQL module" must exist at the client end of an RDA connection is implied. This makes it difficult to optimize for performance because the server DBMS does not have access to the SQL module in advance of execution. The RDA server application entity is in effect providing a "services interface" (see section 4.5). SQL data manipulation language requests are received in standard format (as an SQL text string), and tables that result from SQL select statements are returned to the client in standard format (in ASN.1 encoded format). However, because the manner in which SQL is defined involves dependencies with programming language environments, the "database services interface" which RDA defines cannot be directly utilized by a client process.

In effect, this results in several parts of RDA being undefined. The interaction between the client application program and the RDA client process in the application layer remains undefined. Similarly, the interaction between the RDA server application process and the local DBMS of the server remains undefined, and thus subject to differing

implementations by different vendors. The lack of definition of the actual interfaces with the RDA service is the result of the degree to which the definition of SQL services has been integrated with programming language source code.

While such a situation is to be expected considering the evolution of database standards, it is not necessary. The current design lacks orthogonality. This often occurs in information systems development when requirements change and designs formulated to satisfy earlier requirements are modified.

It is possible to optimize RDA access during the course of an association by means of the R-DefineDBL and R-InvokeDBL operations. These prevent the need for repetitive parsing and compilation of DML statements by having the server assign a "command handle" which uniquely identifies a particular operation. The client may subsequently use the command handle to cause re-invocation of the defined DML statement. This RDA process corresponds to the normal embedded SQL approach, while an RDA invocation of a previously unspecified SQL DML statement corresponds to dynamic SQL statements. However, the period for which RDA-defined DML statements exist is only for the length of the association. While there are logical reasons for this somewhat arbitrary duration for the validity of defined DML statements (relating to unique identification of the DML statements to be executed), this duration may at times be arbitrarily short when associations are established and broken frequently. It may also be too long a duration in certain circumstances. For instance, if schema changes relevant to the data being accessed in a defined DML statement are

required during the course of the association, previously defined DML statements may be invalidated.

The above noted difficulties raise a number of questions concerning the viability of the manner of specification and implementation of centralized database access requirements. If the existing approach of performing mappings between programming language and DBMSs is termed a "gateway" approach (see Figure 2.1), the identification of an intermediary set of common language-independent data types could be termed a non-gateway approach (as in Figure 2.2). The standardization work on establishing such a common set of language independent data types (see section 4.11) has been demonstrated to be a viable approach. Unfortunately, the Languages Subcommittee (JTC1/SC22) of ISO appears to be progressing rather slowly with its work on this standard. The requirement, however, becomes even more important in a distributed environment where it cannot be assumed that the database user and the database reside on the same machine.

5.2.2 Consideration of Communication Requirements

Interprocess communication for distributed systems must take into consideration the situation in which the identity of the server(s) containing the required data is unknown. Broadcast techniques have the potential to provide a more efficient method of communication under these circumstances, particularly in high-traffic networks with extensive replication.

As noted in section 4.5, the Remote Database Access service and protocol [40] references a number of other OSI facilities. These include the Association Control Service Element (ACSE), the Commitment,

Concurrency and Recovery standard (section 4.10), and the use of Remote Operations notation (section 4.5). The peer-to-peer relationships of the OSI family of standards, which assume no hierarchy nor central control, are well suited to the provision of flexible (as regards adding or removing sites from the network) interprocess communication facilities for distributed database management.

However, there are technological developments underway which deserve consideration in the future development of OSI interprocess communication standards. The Reference Model of Open Systems Interconnection [52], and most of the OSI series of standards, are predicated on end systems being heterogeneous in nature. This assumption was valid when the OSI Reference Model developments began (1978), but is no longer as valid. Many of the developments in operating systems interfaces, as with the POSIX standard for the UNIX environment, and in the database area, have been working towards the achievement of a greater degree of homogeneity in the end systems. The concept of the "universe of discourse" is introduced in section 6.1 [38]. As has been realized in the development of the Application Layer Structure standard of OSI, which is still in the formative stage, there is a need to take into consideration the degree of commonality of the shared universe of discourse of communicating applications if efficient interprocess communication is to occur.

OSI standards such as the File Transfer, Access and Management standard (section 4.7) have been developed assuming heterogeneity of communicating end systems. Each vendor developing an FTAM implementation references the abstractly defined attributes of the shared

"virtual filestore". The manner in which individual vendors may actually implement their FTAM implementations is a part of the Local Systems Environment (LSE) and thus not of concern to OSI. OSI is concerned only with the aspects of interprocess communication that lie in what is defined as the Open Systems Interconnection Environment (OSIE). It must be appreciated that the distinction between the LSE and the OSIE is strictly a conceptual difference. For example, the "user element" is a part of the application entity within the OSIE while being a part of the application process.

FTAM and a number of other OSI defined services may be thought of as a "protocol driven" approach to interworking. Protocol exchanges between application layer entities consist primarily of commonly understood data elements. An alternative approach is to use distributed data management operations in order to modify data element values, which then control the behavior of the end systems involved in the communication. The latter method has the potential to considerably reduce the number of protocols needing to be defined and implemented. However, to facilitate such distributed data management operations requires the finalization and implementation of effective and widespread standards for remote database access.

The fact that standards being developed for the LSE are resulting in greater commonality between communicating end systems results in opportunities for simplification of communication protocols, and for greater efficiency in interworking. The RDA SQL Specialization [44] permits the same operations to be performed on a remote server as are allowed for a local database server. Interprocess communication facilities for distributed

database interworking should provide flexibility in taking into account communication in both wide area and local area networks, and also communication between multiple processors in a single site. Such parallel processing may require not only communication between UNIX-type processes, each having distinct address spaces, but may also require support of "lightweight" processes for efficiency. Lightweight processes utilize shared memory in order to communicate, often using monitor constructs to control access to such shared memory. The "threads" of DBMS servers may utilize a monitor to enclose the operations which access cache blocks, in order to ensure the consistency of simultaneous updates to the same block by different threads.

Various types of interprocess communication require differing underlying communication facilities. However, in a distributed systems environment the superiority of the message-passing paradigm is partly due to its potential applicability in all of the situations outlined above, whether shared memory is available for communication or not. Thus standards to support distributed database management should be based on "services interfaces", which react in a predefined manner when a particular message is received, and respond as well in message format. This will require a considerable change in the manner in which database language standards are currently specified.

While the OSI "building blocks" should be used in providing basic communication services, the extent to which the semantics of supported operations are contained in the communication protocol (as in FTAM and in OSI Management) should be kept to a minimum. The use of protocol elements to support such operations should occur only when such operations

relate directly to the distributed environment, as when using the two-phase commitment protocol of the Commitment, Concurrency and Recovery standard. A major advantage of such an approach is that the same services interfaces can be specified and implemented for both local and remote invocation.

5.2.3 Provision of Database Access Facilities

5.2.3.1 Data Modelling Facility Used

Much of the complexity of existing distributed database management may be directly attributed to the difficulties encountered in mapping between differing data models. It is suggested that such difficulties could be considerably reduced by utilizing an "OSI approach", meaning a "non-gateway" approach (see Figures 2.1 and 2.2), to addressing disparities between the data models of different DBMSs in a heterogeneous DDBMS.

The data modelling facility to be selected as the "go between" data model used as the standard format in a DDBMS network is of considerable importance. However, it cannot be expected that it will never change, and any architecture for distributed database management standardization should take into consideration this possibility. Given the current situation as regards candidate data models to which other DBMS models may be mapped in a DDBMS network, there is little doubt that the relational model, as specified in the SQL standards ([43] and [57]), is presently the only possible choice.

A hierarchical or a network data model is a possibility. Although there exists a standard [58] for a network data model, the lack of implementations eliminates this standard from further consideration. The fundamental structures of the hierarchical and network models can cause

side effects in a distributed environment that lead to serious inefficiencies. With both of these models the links (or pointers) contained within the structure must cross site boundaries. The relational model is preferable in this regard since the operations on relations use sets of tuples at a time, and therefore can be partitioned site-by-site. This reduces the requirement for inter-site messages during processing, while the duration of the operation is proportional to the size of the sets, not to the unknown order of a chain.

The operations of the relational model are based on mathematical operations on sets, and subscribe to the algebraic principle of closure. This implies that the result of an operation on two objects is an object of the same type. Thus in the relational model operations on tables produce tables. Additional advantages associated with the relational model result from the fact that the programmer can remain unaware of the underlying file structures and is not subjected to any navigational problems.

Other data models have been proposed and used for differing aspects, such as data analysis and design (see section 3.7). These models are inappropriate as the data modelling facility to be referenced in a standardized DDBMS environment since they do not include a data manipulation language facility. Such models include the entity-relationship approaches, and the binary relationship data model [38].

The desirability of the relational data modelling facility in the role of providing a standard DML for DDBMSs is increased further by the existence of the widely-accepted SQL standard.

5.2.3.2 The External Schema

Much of the theory concerning DDBMS possibilities references the ANSI/SPARC DBMS framework [71]. This framework defines the need for

conceptual, internal and external schemata. The conceptual schema has been the subject of considerable study since the ANSI/SPARC study group report (as in [38]), in an attempt to define an all-encompassing schema permitting mapping to differing underlying schemata. A conceptual schema with the expressive power to perform as proposed by the ANSI study group has been an elusive goal. This has negatively impacted distributed database technology relying on such a concept. In the meantime, the expressive power, in terms of both integrity specification and data manipulation facilities, of practical data modelling facilities (such as SQL relational) has continued to increase.

Concerning the internal schema of the ANSI/SPARC model, it is generally accepted that the requirement for a standardized data storage description facility, addressing storage structures and access methods, is not as crucial as is the need for other standard data management services. The capabilities of storage devices differ, and the number of users affected by such a standard would be considerably fewer. Application programmers are not normally affected by the details of storage structure. It is primarily the Database Administrator role which is concerned with such matters.

The positioning and functions of the ANSI/SPARC external schema (subschema) are often referenced in architectural work relating to distributed database management. However, it is apparent that in most relational systems the role of the subschema has changed considerably, from a statically defined subset of the data structures of the database to a more powerful facility which is closer in nature to a pre-defined query. The subschema was conceived at a time when the technically feasible possibilities for providing a restricted user view were very limited. At that

time the definition of a statically defined portion of the overall schema data structures was foreseen as being sufficient, in responding to needs for access control provisions, data type mappings to programming languages, and indication of update intentions. This type of limited functionality has been exceeded by the view definition facilities of SQL.

The subschema has evolved in the literature to be a mechanism for addressing several largely orthogonal requirements. The subschema is not defined as a solution to a problem; it is defined to handle more than one requirement. As well as subsetting and access control and possible use for performance optimization purposes, the subschema has been assigned responsibility for the manner in which database data is to be presented to a user, that is, the mappings to data types of the user programming language. This is a separable issue, comparable in nature to the functions of the presentation layer in OSI. If it is not separated, it becomes extremely difficult to define a database services interface, which is needed for the efficient definition of data management standard services in a distributed environment. Thus it is suggested that the "presentation" transformation function normally associated with the external schema is a requirement which is orthogonal to data management and should not be a part of the functionality delivered by the database control system of future DBMSs. Until an architecture allows the definition of a "services interface" to data management services, we cannot merely define how a particular service functions. We must be concerned with how a server operates in differing implementation environments.

5.3 Consideration of Requirements for Object-Oriented Facilities

The need for database access facilities which demonstrate the properties advocated by the object-oriented paradigm immediately leads to consideration of the types of interfaces provided by database facilities. The Reference Model of Data Management [5] defines "interface" as: "A defined set of services made available by a processor". However, no standard database service interface exists as an interface that can be accessed by using a known concrete or abstract syntax since SQL defines functionality which depends on the language environment of the executing module. The database access requirements identified highlight the need for access from diverse operating environments both locally and remotely, and the need to prevent duplication of effort in the specification of available services. This signifies the urgency for a database services interface which is not specified in terms of language bindings to any particular programming language, but in a more object-oriented manner.

Although the reference model document has been described in object-oriented terms, existing (SQL) and developing (SQL2 and IRDS) standards have not been rigorously defined using the concept of an abstract data type. An abstract data type is an object type together with a collection of operations applicable to objects of that type. Rigid definitions of the properties of an object type require that the characteristics and precise interfaces to objects of that type be defined, first abstractly (preferably), with one or several possible implementations. The extent to which the specification of database language and IRDS standards has been tied to the

programming languages world has resulted in standards which are not defined as objects should be defined. They cannot be visualized as a "block of functionality" accessible to any authorized user. The action to be taken by products conforming to existing SQL and IRDS standards is dependent upon the language environment to which the "database controller" or "dictionary controller" happens to be bound. The fact that the "binding" itself is so loosely defined is an obstacle to application portability. The form of definition of data management functionality which has evolved from programming language origins is now resulting in difficulties for the provision of remote access to data. A change in the manner of specification is required as we advance into the field of distributed data management. In any case, such a change is required in order to eliminate the need for numerous language bindings to programming languages.

It may be useful to examine the possibilities available for the specification of data management services in a more abstract manner. One possibility, which appears to be sufficiently powerful for such specification is the Abstract Service Definition Conventions work developed in JTC1/SC18 (Text and Office Systems standardization) for Message Handling Systems (X.407), and subsequently used in the Directory Services standards [68-70]. The Abstract Service Definition Conventions standard uses ASN.1 and the macro facility of ASN.1, and maps definitively to Remote Operations ([15] and [16]) for those instances where implementation is to be by means of an OSI realization. ASN.1 is prescribed as the means for specifying the abstract syntax of the arguments and results of procedures. There are a number of possible realizations of the functionality described abstractly by means of Abstract Service Definition Conventions.

5.4 Consideration of Data Dictionary System Requirements

Every request to access data at level n intrinsically implies the need for access to schema data about that data, (i.e. access to data at level $(n+1)$). The requirements for definition and access to data at different levels are not inherently different. The fact that distinct standards have been developed to separately support very similar functions is the result of the evolution of these standards; it is not the result of any "plan" which seeks to provide for product integration. The number of levels accessed is determined by the semantics of the application, and is not limited to the number of levels currently envisioned in IRDS. IRDS standards currently provide for two levels of data, the dictionary level and the dictionary definition level, while recognizing the existence of application level data which is not controlled by the IRDS service.

Figure 5.1 attempts to show pictorially components of a single information processing system, any of which are potentially the subject of standardization. While any attempt to categorize information systems components is likely to be subject to controversy, the five major identified components, Application Support (provision of processing functionality such as is provided by programming languages), Man-Machine Interface Management, Data Management, Information Exchange (other Information Processing Systems) and System Control (provision of operating system functionality) appear to be logical and separable components which may be useful in delineating various standardization activities.

Figure 5.1: Components of an information processing system (source [49])

To a large extent, the information technology standards currently being developed actually do follow the breakdown suggested in Figure 5.1. System Control functions are the subject of the POSIX standardization for operating system interfaces in a UNIX environment. Application Support functions are standardized within the Languages Subcommittee of ISO (JTC1/SC22). Most of OSI can be viewed as addressing the Information Exchange component. MMI (Man-Machine Interface) Management is partially being addressed by Virtual Terminal standards and by the

standardization of user interface facilities such as X-Windows, although there remains a considerable amount of work to be done in this area. Data Management standards are addressed by the SQL and IRDS work, while RDA is the bridge between the Information Exchange component and the Data Management component.

IRDS can, and should, be treated as a standardized application of the use of Data Management services. A standardized application is one where a large number of users from different organizations and in different countries have similar requirements. IRDS standards have developed in parallel with the development of the basic data management functionality defined by SQL and SQL2. As a result, there has been a natural tendency to respond to IRDS requirements separately and in isolation from the specification of the basic data definition and data manipulation language facilities of SQL.

However, the IRDS work has also defined a different data model for access to dictionary data. Although the history of parallel development of IRDS standards with database language standards has resulted in different data models (entity-relationship and object-association data models) being developed to support the dictionary application, there are compelling reasons for attempting to prevent any proliferation of data modelling facilities being supported by standards. One reason relates to the requirements which are going to be placed on IRDS, if it is capable of handling them, in a distributed systems environment. It is unrealistic to expect that vendors of database management products are going to wish to use different data models depending on the level of data being accessed. For example, is it reasonable to expect a vendor to make use of a different data

model for metadata than has been used for application level data? Should the same vendor be asked to use possibly another data model for accessing data fragment location information required for distributed data management? If a different data model does exist for dictionary data, then access protocols for distributed access to such metadata will need to be developed. The result may be very similar to the situation with the Network Database Language; IRDS standards which provide an alternative, rather than a value-added functionality, are likely to be ignored by a marketplace that does not need multiple solutions to the same problem. The functionality being addressed by IRDS standards can be categorized as separable from the basic data definition and manipulation functionality defined by SQL.

Section 3.4 identifies objectives to be addressed by an Information Resource Dictionary System. The manner in which such IRDS requirements were addressed in the formulation of standards followed a pragmatic approach in attempting to make IRDS services available in support of various programming language environments. This is consistent with the scope identified in [28], which references support for all ISO standard database languages (NDL and SQL are specified) and programming languages. However, the IRDS objectives documented also identify functionality to support ISO standard Open System Management and Directory Services as being a requirement. Also, it was required that an IRDS should be able to provide facilities for programs written with ISO standard programming languages to use the SQL and NDL Data Manipulation Languages to access information stored in the IRDS at execution time. Reference [28] further noted the need to allow access to its data from another open system using RDA protocols.

It is suggested that the architectural mechanisms used to address the requirements documented in [28] are of paramount importance in achieving most of the objectives identified above.

The scope statement of [28] does not include reference to a distributed environment, although the need to access IRDS data remotely is noted as is the requirement to support OSI Management and Directory Services, both of which are distributed system applications. The fact that support for NDL would become as meaningless as it has may not have been foreseen at the time that reference [28] was written (November 85). However, the impact of envisaging a requirement to support more than one standard database language has had considerable influence on the manner in which IRDS standards have been developed. It has also been a factor in motivating the use of a distinct data model for the provision of IRDS services - a consideration similar in nature to the motivation for the standalone data dictionary products of the past which were internally independent of any database product offering. But the existence of an effective ISO standard database language changes this situation considerably. It is possible that the difficulty of accessing IRDS data using different data models may have been underestimated.

In reviewing the requirements envisaged for IRDS standards in November 1985 as compared to the current situation, it is interesting to speculate on the extent to which IRDS objectives may have changed with evolving technology. The extent to which support for different database languages would be required was overestimated. The effort involved in supporting a variety of different programming languages, at least by current means, may yet be underestimated. So far, IRDS functionality has

not been considered for access (via individual language bindings) from any programming languages other than Pascal. Requirements for distributed access to IRDS data were not adequately considered.

With reference to the requirements identified for object-oriented facilities (see section 5.3), IRDS must provide capabilities relating to the idea of "encapsulation" or packaging of functionality. It has long been appreciated that controls are necessary if data dictionary systems are to be more useful in the future than some of the "passive repositories of out-of-date documentation" of the past. The implementation of the object-oriented paradigm, such as has been done with "packages" in Ada and with "modules" in Modula II, permits the "hiding" of private portions of the package specification such that only visible definitions are accessible to (authorized) users.

Location information concerning data fragments for distributed database management must be hidden from DDBMS users for integrity control purposes, and must only be accessible to the DDBMS itself. Furthermore, such location data is relevant not only to application level data but to dictionary data and dictionary definition level data, since these may also be distributed in a distributed database.

Much of the current development on IRDS standards has been predicated by the need to provide a human-usable interface to dictionary data. With reference to Figure 5.1, the user-friendliness of the man-machine interface component, and the data model presented to a user involved in data analysis or database design, is a separable issue from the provision of basic data management functionality. As we move towards distributed systems, the underlying components which we define must be

machine-accessible. Thus the requirement for an Information Resource Dictionary System to be usable in a distributed database environment is fundamental to the usefulness of the standard in the near future. Provision of standards for a human-usable facility addressing data analysis is of value, but is of lesser importance than providing an IRDS which will be useful in a distributed systems environment.

The degree of integration of DBMS schema tables with the portion of dictionary level data representing the database schema has been rather limited. This results in difficulty when attempting, in the distributed situation, to provide location data for the information resources which are "documented" in the IRDS and stored as database objects. The degree to which such data objects are integrated must increase in the future if standardized means are to be found for accessing the location information for heterogeneous DDBMS implementations.

In horizontal partitioning of a DDB, standardized location information must be available concerning the ranges of values of fields in a record which can be found in one, or more (in the case of replication) locations of the DDB. In vertical partitioning, the location information must be defined allowing a user to view a logical table which is, in effect, instantiated by a join involving multiple sites of the DDB. Effective interworking involving DBMS implementations from various vendors requires a greater degree of standardization of both the database catalog (schema) information, and of the format in which location information is maintained.

It is suggested that IRDS should be explicitly providing a value-added functionality over and above the basic data management facilities currently provided by SQL, and increasingly provided by SQL2 and SQL3 [72].

Distributed database management considerations should form part of the requirements which IRDSs are to address, as well as the assumed IRDS added-value functions such as versioning, life-cycle support and provision of programming language specific data definitions. Thus an IRDS which must operate in a distributed environment should be usable with SQL (and RDA) facilities in order that data held remotely may be accessed, regardless of whether the structures used for interface with a data administrator differ from SQL. IRDS data should be stored as SQL tables in individual sites of standard-conforming distributed database systems, and should be accessible as SQL tables where permissions are granted to do so. This does not preclude any degree of encapsulation of specific IRDS modules, possibly having exclusive update permissions on different elements of IRDS data, where control and/or man-machine interface considerations warrant such measures. This recommended strategy is similar in nature to the approach used by a number of relational DBMSs. These DBMSs have implemented their database descriptions as tables embedded in the database they describe. Such DBMSs are said to be *homeomorphic*. IBM's DB2 follows this practice with a catalogue containing several relations: SYSTABLES is a relation naming and describing the tables in the database, while SYSCOLUMNS is a table naming and describing the attributes (columns), etc. SQL queries may be performed on the catalogue.

5.4.1 Dictionary Data versus Directory Data

The name "Information Resource Dictionary System" relating to the standards reviewed in section 4.4 does not contain the word "system" by accident. The IRDS standards prepared by both ANSI and ISO have been designed as if they were a "system" solution to a standalone user

requirement. However, a distributed architecture which is going to be viable in satisfying the diverse user requirements documented in Chapter 3 must consist of a number of "building block" components designed to fit together, as was the case in the development of OSI standards. The manner in which this can best be done, in the case of data dictionary standards, may relate to the inherent separation between "dictionary" system requirements and data "directory" requirements.

At a certain point in the evolution of data dictionary standards, the term Data Dictionary/Directory was commonly used. This term highlights the inherent distinction between dictionary system functions and the directory function related to locating information stored within an information systems environment (possibly distributed). The term "directory" is seldom used currently in relation to dictionary functionality. This is largely because of the fact that database systems typically provide location information (cylinder and track addresses, etc.) in a centralized system, and dictionary systems have not yet had much impact on the distributed environment. It is suggested that the dictionary and directory components should not be viewed as a part of an Information Resource Dictionary "System", but as distinct functions which are separable, and which need to be separable in order to address adequately distributed database management concerns.

Many of the requirements documented in section 3.4 are the needs which a large class of users (individuals with responsibilities relating to the analysis, design, development and maintenance of information processing systems) have in common. These requirements result in the specifications for a "standardized" application. The "directory" requirement is an

orthogonal function needed to identify the location of data or other information resources, and is of concern particularly in a distributed database environment. A DDBMS, regardless of the application which it may serve, has a requirement for a data directory function at each site in the network. Each individual DDBMS application (including the standardized data dictionary application) has requirements for locating various fragments of data which may be stored at any of a number of sites in the network. Normally, the DDBMS will utilize such directory location information in order to provide some degree of transparency to such applications. The consolidated results of a request for information which requires data from multiple sites will, for example, be merged prior to presentation of the data to the requesting application. Thus the "dictionary" component is the component which addresses the application semantics of the standardized dictionary application. The "directory" component is needed in every DDBMS application and is concerned with the provision of location information which can identify the site(s) of a network where data resources can be found. The directory information is established by and used only by the DDBMS in order to respond to application requests.

5.5 Consideration of Requirements for Security

Whether a permissive or a restrictive approach to access control is desired, access control provisions must be concerned both with data (including values of individual occurrences), and with the services which are available to reference the data. A restrictive access control scheme assumes no permissions unless specifically granted, while a permissive scheme assumes access by all users to all data is allowable unless specifically prohibited. Since both the data structures and the operations are dependent

on the data model, access control provisions concerned with database data must be dependent on a particular data model.

The access controls of SQL, that is, the granting and revoking of permissions for the execution of the procedures of SQL modules based on the value of authorization identifiers, provide a level of security to basic data access functionality. SQL views containing a query specification and thereby selectively delimiting the exact subset of database information to be available to a particular user, are effective in restricting the data which may be retrieved. Limitations on the updatability of views make the view mechanism less useful for update than for retrieval. In addition, many theoretically updatable views are not specified as being updatable in the SQL standards, although such restrictions are being relaxed in the SQL2 standards. Similar access controls apply to data at different levels, notwithstanding the possible requirement to add additional access controls for standards-defined processes such as may be defined in IRDS "modules".

An IRDS dictionary function is an important standardized application, but as regards basic database management functions, it is nonetheless an application. If an environment is put in place wherein IRDS application processes are the only procedures with access permissions to data above the application level, then the IRDS will have exclusive access to such metadata. This may include query as well as update procedures. Separate access control facilities for access to such IRDS procedures are important in order to protect data above the application level, but they are distinct from the access controls to data under the control of an SQL-conforming implementation.

Similar requirements for control of application level data also exist. Once again the facilities of SQL are useful in providing access control for retrieval and update of base tables (tables which physically exist in the database as opposed to derived or viewed tables, which need not). However, encapsulation mechanisms with exclusive access permissions to application data may provide additional control over permissible operations on application data, when the access control permissions of the basic data management function are insufficient. This is very similar in concept to the situation outlined above for IRDS-controlled data.

5.6 Consideration of Requirements for Transaction Management

As noted in section 3.6, the transaction concept generally is accepted as the appropriate mechanism for addressing the issues of integrity, recovery, and concurrency control within database systems. The "ACID" properties of a transaction can be easily specified (if not so easily implemented) by the use of concepts such as serializability in a local DBMS environment. While the manner in which this is accomplished in a local DBMS is typically proprietary in nature, heterogeneous distributed database interworking requires that an appropriate mechanism be selected from among the various possibilities outlined in section 3.6, and that such a mechanism be standardized.

Fortunately, the Commitment, Concurrency and Recovery (CCR) standard has been developed as an OSI Common Application Service Element which prescribes the necessary protocol interaction to facilitate concurrency control between two communicating sites. Use of the facilities of the CCR standard are provided through the Remote Database Access

standards. The DDBMS, as an application process, is then responsible for providing the higher level of control associated with a particular transaction tree. This ensures, for example, that all participating sites have responded to a "secure" directive before issuing the "commit" order. The participating local DBMS implementations are then responsible for ensuring that the ACID qualities are upheld within local sites once appropriate DDBMS-initiated protocol messages have been received by the RDA server components.

Facilities such as are specified by the CCR standard provide specific mechanisms for addressing a number of transaction management requirements necessary for distributed database interworking. However, consistency and integrity control in a distributed database require a number of additional controls to be incorporated into the DDBMS implementation. Many of these controls are associated with the supported distribution and replication alternatives and with whether less than 100% consistency is to be permitted. Local autonomy issues favor the possibility of performing updates when one or more sites containing replicated data or metadata may be unavailable. Whether or not such updates are permitted, the key to maintaining integrity is the availability of directory information identifying the location of all replicated data.

5.7 Consideration of User Interface Requirements

As represented pictorially in Figure 5.1, one component of any information processing system may be considered a man/machine interface component responsible for coordinating the interaction with the user. In the past the representation of data in data management software may have

been bound too closely to the manner of presenting such information to users. However, as has always been the case during the evolution of computer hardware and software, the prevention of duplication of effort in systems development requires the separation of different layers of functionality which are able to form the "building blocks" from which applications are developed.

In addition, it is suggested that there is a need for the separation of the data structures visible to human users from the data structures which are appropriate for DDBMS interworking. This is similar in nature to the considerations discussed for Data Dictionary System requirements. Users have understandable requirements for viewing data using structures and operations that seem natural to them. Many types of interface are currently used, such as forms or a natural language query interface, and many more are possible in the future. Such user interfaces, as suggested in Figure 5.1, should optimally be a completely orthogonal consideration to the provision of basic data management facilities, regardless of whether such facilities are concerned with centralized or distributed systems. Standardization of user interfaces, if and when appropriate, should be completely independent of database and distributed database standardization.

One performance related user interface requirement which needs to be directly supported by data management standards is the requirement for both interactive access to database information and for the repetitive invocation of previously compiled and optimized requests. A services interface format supporting this requirement and also supporting identification of previously validated requests may be the most appropriate, primarily because of its applicability for both local and remote invocation.

5.8 Consideration of Requirements for Distribution Support

Ceri and Pelagatti [1] note that the three most important objectives which motivate the features of an architectural framework for understanding distributed databases are the separation of data fragmentation and allocation, the control of redundancy, and the independence from local DBMSs. The latter permits local mapping transparency, and reduces the need to take into consideration potentially differing data models of local DBMSs. In order to make progress in the development of DDBMSs, it is necessary to reduce the extent to which consideration of a potentially large number of different data models complicates the picture. Attempting to insulate architectural features from the nuances of local DBMSs appears to be a desirable design objective. However, this does not imply that local DBMSs cannot utilize differing data models; it only suggests that the local mapping of the fragments of a DDB which are allocated to a particular site should be a secondary consideration when determining an architectural framework for distributed databases.

The data modelling facility to be used throughout a distributed database environment is an issue of paramount concern and has been examined in section 5.2.4.1. Only with reference to the relational model has the central issue of query optimization been isolated, clearly formulated, and analyzed. The problems which must be addressed by the database control system in a single site DBMS are fundamentally similar in nature to query optimization issues faced in a distributed environment. The results of analysis efforts focusing on query optimization apply to both local and distributed environments, although additional consideration is required in

the DDBMS context in order to factor in the increased cost of different forms of communication between sites.

The optimization of DDBMS queries is a feature which differentiates various products, in the same manner that query optimization results in performance differences in local DBMSs. However, it is not difficult to determine the information which must be available in order for such query optimization to be possible. Directory information concerning the location of data fragments, and the manner of accessing such information, must be the subject of standardization if heterogeneous distributed database interworking is to be supported. The mechanisms incorporated in the Remote Database Access standards have the potential to support access to any type of data, including directory information. The facilities associated with the SQL relational data model provide a means of structuring such directory information. Thus it is suggested that these mechanisms should be utilized in providing standardized access to data directory information, both locally and remotely.

Other requirements which must be addressed in achieving an architectural model able to support distributed database management include:

- ensuring that consistency is maintained across all sites containing replicated data;
- ensuring that if updates are permitted when sites containing replicated data are not available, that mechanisms are put into place to ensure that consistency is restored and that out-of-date data is not accessed in the interim;
- ensuring that adequate data directory information is available in order to verify referential integrity constraints which cross node boundaries;
- providing support for the wide variety of distribution and replication

alternatives outlined in sections 3.8.2 and 3.8.3, both for application level data and for different levels of metadata.

The provision of viable DDBMS facilities that adequately address such demanding criteria is a difficult task, particularly considering the mathematical complexity of the query optimization problems. It can be expected that support for such DDBMS requirements will only slowly become available in commercial DDBMS offerings. Fortunately, most of these requirements are directly related to the degree of performance expected to be delivered by DDBMS implementation, and need not affect the development of an architecture within which such problems may be addressed by individual DDBMS vendors. Much of the current research focusing on optimal distribution and replication alternatives, distributed access plans, and the extent to which directory information should be replicated, can be addressed within a standardization framework which concentrates on the information needed for such optimization.

5.9 Consideration of Requirement for Heterogeneity

The criticality of protecting both user and vendor investments in existing database systems and software is such that concerns for heterogeneity must be addressed in any viable architecture for distributed data management standardization.

Commercially available DDBMSs typically provide very limited support for translation between data models, referred to in [1] as the "really hard problem". However, addressing this difficult problem is not a prerequisite to enabling DDBMSs to interwork, and perhaps the problem should explicitly not be addressed, any more than the problem of interfacing

various communications network architectures by means of a gateway approach (see Figure 2.1) should have been addressed. It is possible that the somewhat slow pace at which viable DDBMS implementations are becoming available is related to the extent to which the approach to the provision of support for heterogeneity is inadequate.

Thus it is suggested that there is a possibility of providing for heterogeneous DDBMSs where the database language (and data model) to be used for interactions between nodes of the network is standardized. While it is apparent that the builder of a distributed DBMS utilizing several existing DBMSs must be concerned with the differences in data models and data manipulation languages, perhaps the optimal manner of addressing such heterogeneity is to take the OSI approach to interconnection (see Figure 2.2). Thus the recommended approach to the construction of DDBMSs which may consist of disparate DBMSs is to develop standard specifications for the interactions between DDBMS sites to which vendors of database products can then map.

5.10 Consideration of Requirements for Export/Import

Export and import facilities, as with database unload and load facilities, have usually been addressed as a completely separable issue from normal retrieval and updating operations. The reasons for this stem from the degree to which data modelling facilities and data formats have been proprietary to individual vendors.

It is suggested that the selection operations required to indicate the portion of a database of interest for producing an export/import file do not differ significantly from the semantics of SQL "select" statements. It is

further suggested that if a "services interface" approach is taken to the specification of the operations to be supported by standard database facilities, then the format of an export/import file could be precisely the octet stream which would result from normal DML retrieval operations. The manner of representation of schema level data associated with the lower-level data as a potential part of an export/import file must also be considered. However, since such metadata is inherently no different from lower level data, such as application level data, and can be represented using the same data modelling facility, there is no reason that the same approach cannot be used for all levels of data for which the creation of an export/import file may be appropriate.

It is thus suggested that the selection operations associated with standard database languages be used in specifying the portion of a database for which an export/import file is desired. It may be necessary to extend DML retrieval semantics in order to account for the differing selection requirements for export/import purposes, such as ensuring that schema level data accompanies application level data. Taking a "services interface" approach to the output of such a DML request will mean that the result of such retrieval operations will be a "message". Such a message will consist of a flat-form representation of the relational tabular structure, which can simply be used for the purpose of storage or communication as a file rather than presentation to a user.

5.11 Consideration of Requirements for Support of Additional Data Types

The requirements for integration of a number of different data types into the domain of what were previously considered databases of

"structured data" can be expected to increase in the near future. Examples where such requirements are prevalent are in fields such as document management or the storage and retrieval of engineering drawings. In the past there has been a tendency to assume that the DBMS should have knowledge of every type of data stored within it, when in practice this is probably far from realistic. For instance, in the case of compound electronic documents which conform to the Office Document Interchange Format standard, it is not realistic to include in a data management component (a DBMS) knowledge of the semantics of the Office Document Architecture (ODA) standard [73]. However, what may be realistic, if it is worthwhile to put such information in a database, is to separate out the type of information which would be used to retrieve such complex data types from the database. This is typically database information - the type of information which would be stored as document profile (header) information in ODA.

Operations which are not currently database DML operations, such as proximity searching, can be done on the data once it is retrieved from the database. The document profile information would be stored in the IRDS at the Dictionary level. The ODIF octet string which represents the document itself would be stored as database data at the application level. Such an approach would have considerable appeal in organizations attempting to use data management services effectively in order to control electronic documents, or hard-copy documents (although for these no database data would exist at the application level). As the number of database-supported data types increases, it can be expected that additional operations on such data types will permit increasingly flexible access to textual and other data types in a distributed environment. For example, a Structured Full-Text

Query Language has been proposed by the Air Transport Association [74]. This facility avoids the problem of differing application program interfaces to the textual search engines of different vendors, by specifying an SQL-like language for the required operations. Incorporation of such facilities into SQL standards would then permit their utilization in both local and distributed environments.

Chapter 6

Proposed Architecture for Distributed Data Management Systems

6.1 Requirement for a Reference Model

In the past there has been a tendency to treat the field of DDBMS as a distinct area, at least from a standardization perspective. As depicted in Figure 5.1, any attempt at standardization must necessarily address a number of components which together form a whole. To a great extent this has been the difficulty with many of the "fourth generation languages" - they provide powerful application development facilities but have a tendency to define their operating environment in isolation of other software products with which they must interface. The preceding discussion of DDBMS requirements, and the analysis of these requirements as related to current standards efforts, clearly point to the need for an agreed "master plan". This is essential if future information technology standards are to interwork successfully in supporting the distributed database environment. It is equally apparent that solutions to current problems with the progression of heterogeneous DDBMS should be considered with reference to the OSI approach, as opposed to the gateway approach, to interconnection (see Figures 2.1 and 2.2).

The proposed architectural model thus assumes that interconnection of heterogeneous DDBMS components should be based on a common interface

language, relating both to aspects of the communication services and protocols as well as to the data modelling facility which is to be used. As was the case with the OSI Reference Model, the possibility exists for the development of an "Esperanto" in the DDBMS area, but, as with OSI, successful realization of such a goal must occur in a timely fashion. Thus it is important that international agreement be reached prior to the implementation of numerous different languages and dialects for DDBMS communication. If such agreement is not reached in the near future, we will inevitably need to support numerous (in the order of n -squared) gateways between disparate implementations.

Reference [38] defines a "universe of discourse" as "that part of the real or hypothetical world which is understood in the same way by communicating partners and which they agree to communicate about." In order for heterogeneous DDBMS components to communicate, it is necessary that there be greater commonality in their shared universe of discourse. In an application process, the representation of the formal description of the universe of discourse is referred to as the conceptual schema for the communication. The objective of the conceptual schema in the context of communication between application processes is to define the semantics of the data elements that can be referred to in the communication, and the allowable operations that can be carried out on these data elements. The ultimate goal is the 100% principle [38], that is, to be able to define all of the semantics of database data and permissible operations as a part of the specification of the conceptual schema.

Much of the DDBMS literature refers to the conceptual schema as if it actually existed and could be used, as proposed in the ANSI/SPARC DBMS

Framework [71], in mapping between differing data models. At present, the relational database language standards (references [43] and [57]) are the only widely accepted reference point for defining the format and semantics of database information. The SQL2 language is gradually coming closer to achieving the objectives of the conceptual schema as proposed in [38]. Currently and in the near future this language is considered the most appropriate candidate for the data modelling facility that will represent the data-related portion of the shared universe of discourse between communicating DDBMS sites. Existing and developing OSI standards are thought to be appropriate as communications facilities in a standard DDBMS environment.

The proposed architecture, using a number of "building blocks" of required functionality, recognizes that the nature of distributed systems demands numerous asynchronous processes able to communicate via messages. This corresponds with the recent school of thought advocating that the most attractive model on which to base a distributed database is the client-server model. In this model, the clients (or requesters, or data users) react with local or remote servers (the data providers) by means of messages. Advantages of communication via message passing are that the side effects of unexpected process interactions are eliminated, and the internals of each process are isolated from other processes, thereby making it easier to change the way a process operates without affecting others. The peer-to-peer nature of the OSI interprocess communication adds to the flexibility by permitting greater ease in adding or removing servers.

The need for effective interworking between DDBMS components will necessitate standardization at the level of the "services interface". It is

possible to view such an interface as an application protocol which may be invoked either locally or remotely. At present, standardization work in distributed database and data dictionary systems is being impeded by the extent to which previous standardization work has focused on the "language" used to define or access a database or dictionary system. There is an increasing realization of the need for the standardization of services interfaces in order to circumvent the implementor-defined nuances of differing implementations of the same data manipulation source code. These differences can considerably hinder interoperability. Since the standardization of such services interfaces has latent impact on application portability requirements in the local system environment as well as on interoperability requirements, it is suggested that the proposed architecture has the potential to prevent considerable duplication of effort in resolving both database and DDBMS standardization issues.

Proposed Architecture for Distributed Data Management Standardization

Figure 6.1

6.2 Description of Proposed Architecture

Figure 6.1 shows a proposed architecture for the components of data management standards. This architectural model takes into account the requirements outlined in Chapter 3 and the status of existing and developing standards outlined in Chapter 4, while reflecting the design considerations discussed in Chapter 5. The proposed model does not include consideration of facilities falling outside the bounds of data management standardization, such as the facilities offered by file servers utilized by DBMSs.

Concerning the notation of Figure 6.1, the cylindrical objects at the right of the Figure are intended to represent containers for the storage of persistent data. The rounded rectangles with a hatched area represent processing elements (or processors). Such abstract processors are a modelling construct used to indicate architectural components. The component diagram shows the client/server interactions between the processing element components necessary to support distributed data management. The services interface associated with each processor is indicated by the hatched area at the left of each box. The ellipses are used to depict invocation of communication services, which in an open systems environment would be provided by various OSI standards from the presentation and lower layers. Remote Database Access is an application layer service which, in conjunction with the other associated application layer services, provides the bridge between data management services and the interprocess communication facilities defined by OSI.

6.2.1 Database Controller

The Database Controller represents a class of abstract processors that provide the services typically associated with database management. These are facilities to store and retrieve data and to manage transactions within a set of defined constraints. Such services must relate to a single Data Modelling Facility. Each defined Data Modelling Facility has a prescribed set of data manipulation rules that define the functionality available at the interface to the Database Controller. This interface is identified as the "Database Services Interface". The Database Services Interface is specified in a manner that is independent of all parameters of the accessing environment, such as the programming language being used by an accessing processor.

The Database Controller function is the basic building block that supports the storage of all types of data required within distributed data management. It is used to store data of interest both to user applications and to other processors included within the data management architecture. Each processor making use of the data management services provided by the database controller is responsible for defining and manipulating its own data; this data may be hidden from the view of other processors, or may be available for query and/or update.

The data thus defined by user applications or by other defined processing elements will necessarily involve at least two data levels (the schema level and the instances that relate to such data definitions). Such data may exist in a hierarchy comprising more than two data levels. Thus a user application (or a standardized user application such as an IRDS) may

define and control data at a number of different levels, as required by the application semantics.

Data manipulation requests to the Database Controller will contain an input indication as a part of the Database Services Interface which references the level of data to be accessed. Access to data at any level implies that a container conforming to a particular schema has been created for this data. Such a container may thereafter be populated, creating a database which may be referenced by a level designator (eg n-database). Data in an n-database that consists of schema data for a lower level may subsequently be selected and processed to form an activated schema, for which a related container provides a repository for lower level data occurrences. A populated database at the lower level may be referred to as an (n-1) database.

The Database Controller will have no knowledge of the difference in data semantics associated with different level pairs of data. It will make use of the same insert and modify commands to update data at higher levels as it uses to update data at the application level. As a result, the Database Controller will have no need for separate languages to create and update schema data. This does not imply, however, that "macro" functions cannot be utilized by data management system implementations in order to carry out predefined series of data manipulation commands relevant to data at a certain level (e.g. corresponding to update of schema level data). It does, however, imply that there is no need for separate standards for storage and retrieval of data at differing levels.

Thus the Database Controller will be responsible for storage and retrieval of database data at a number of (identified) levels, but the syntax

and semantics of update and retrieval DML commands, and the utilization of access control facilities, will be the same. The format for accessing a database that is provided at the database services interface may be either data manipulation language commands or references to previously defined database language statements.

The intimation of the term "Database Services Interface" is that the composition of the messages crossing the boundary can be precisely determined, independently of the invoking application environment. The format of the data which crosses the Database Services Interface can be very similar in nature to the manner of presentation used elsewhere within the Open Systems Interconnection Environment. Considerations resulting in the type/length/value specification of ASN.1, such as the separation of the abstract syntax from the differing transfer syntaxes that may be negotiated, are also applicable for data crossing the Database Services Interface. The use of a mechanism such as ASN.1 is appropriate regardless of the efficiency of a particular transfer mechanism. Such a format may imply certain changes in the current manner of utilization of ASN.1 (relating to dynamic rather than static use of the ASN.1). The use of the existing ISO 8825 encoding [27] may also imply some inefficiency when shared memory is available to the client and the Database Controller. However, distributed systems considerations dictate the use of such a "message-passing paradigm" approach in order that the same mechanisms can be used for both local and distributed database access. Performance considerations may result in the definition of additional transfer syntaxes in the future which are based on the same data semantics (i.e. on the same abstract syntax).

The services of the Database Controller may be used to store data of many different data types. Each data object class is associated with differing operations and possible constraints; such operations and constraints are defined in the standards for the database controller function. It is expected that the number of data types supported by standard-conforming database controllers and the number of operations on such data types will continue to increase as databases are used for the storage of increasingly diverse types of information.

An example is the current requirement for a greater degree of integration between the type of structured data normally stored in relational databases and textual information. Textual record access requirements include proximity searches and other full-text retrieval operations. Access to full-text searching facilities is typically provided by proprietary application programming interfaces. This results in difficulty in accessing text under the control of different vendor's text search engines. Consequently, efforts have begun to define a standard interface language [74] for accessing text; it can be expected that, as such standards mature, there will be a requirement to include in the function of the database controller, as an additional supported data type, the textual searching operations then available for distributed database processing.

An immediate advantage of the definition of such a database services interface is that access would be facilitated from other than a programming language environment. For example, there is a requirement for access to databases from electronic compound documents. Many legal documents are 90% boilerplate and 10% data, and a requirement exists to be able to provide access from electronic documents to database applications such that

the data can be extracted automatically from the database for insertion in the document. Using a services interface approach, an electronic document could include a database request which would result in the inclusion of the appropriate updated database information.

6.2.2 Distribution Controller

The Distribution Controller incorporates most of the facilities commonly associated with distributed database management. This function is responsible for obtaining and using data location information in determining an optimal access plan. Many of the functions of the Distribution Controller do not require standardization. These are aspects for which vendor ingenuity is needed in order to meet the DDB requirements for various types of transparency and differing distribution alternatives. For example, if updates are to be allowed during periods when one or more sites requiring update of replicated data are unavailable, it is a responsibility of the Distribution Controller to ensure that such inconsistencies are resolved at a later date.

Supporting various distribution alternatives in a heterogeneous environment requires standardization of the format of the location information accessed by Global Data Directory processors in different nodes of the network. However, most of the functionality provided by the Distribution Controller is strictly in the purview of individual vendors, just as the means of providing the services supported by the SQL database language is the responsibility of DBMS vendors. The Distribution Controller must determine optimal strategies for retrievals involving multi-site joins in the same manner that individual DBMS implementations are responsible for query optimization in the local system environment. The Distribution

Controller can be tailored to satisfy different degrees of transparency, such as fragmentation or location transparency.

The Distribution Controller is shown in Figure 6.1 as a single abstract processor class, whereas in actual fact it may well consist of a number of separable components, some of which may be appropriate for standardization. For example, the Distribution Controller would be required to provide services to support the definition of fragments, and also the allocation of such fragments to various sites of the network. Specification of fragmentation and specification of allocation are separate functions for which different standardized services interfaces may be appropriate in the future.

Note that two possibilities exist for communication between different sites both of which contain Distribution Controller Processors. The first is that the Distribution Controller of one site is totally responsible for execution of any DML request. In this simpler alternative, the Distribution Controller Processors of other sites become "pass-through" functions providing, for any DML request, access to either data directory information or to required data known by the remote Distribution Controller to exist at that site. In a more sophisticated implementation, several Distribution Controller Processors from different sites may cooperate in devising the optimal access plan for a particular DML request. This latter alternative requires a greater degree of coordination of the functions of the Distribution Controller Processors, including a standardized protocol for communicating between such processors. Decomposition of the functions of the Distribution Controller is not currently appropriate since data management standards for the distributed environment are still in their infancy.

6.2.3 Global Data Directory Processor

The Distribution Controller Processor incorporates facilities to control the allocation of data fragments throughout the DDB. The allocation of data is determined by both user indicated preferences specified at the interface to the Distribution Controller Processor, and by the replication and distribution strategies incorporated in individual DDBMS implementations. The function of storing and retrieving information relating to the location of data fragments is logically separable from the more strategic and performance-oriented functions of the Distribution Controller.

The Global Data Directory processor is thus defined to determine the site location of data to be accessed from location information stored in standard format. To do this, it utilizes the facilities of the Database Controller function. This does not imply that complete location information is maintained by the Global Data Directory processor function that resides in each node of a distributed environment. No assumptions are made as to distribution alternatives. The directory information related to data may be totally centralized, replicated in each site, or any alternative between these two extremes. Locally stored data location information may also simply indicate a location where more complete location information is stored. The Global Data Directory processor does not contain detailed physical access parameters for data stored locally; it defers to the DBMS/operating system. Such physical location information is usually proprietary in nature. This is the catalog information associated with each DBMS.

In order to allow for heterogeneous interworking, there is a requirement for the site names of data stored remotely, and the format for

relating this data to the objects of concern to the Database Controller, to be stored in a standardized format. Such location information must be represented in terms of the data modelling facility used in common by all of the Distribution Controller, the Global Data Directory processor, and the Database Controller. The determination of the format of such location information is a necessary standardization effort to be undertaken. The function of the Global Data Directory Processor is to utilize such a standard format in providing data identifying the site(s) at which all or a portion of data being accessed is stored. When data is being updated, the Global Data Directory function will determine where such data must be stored in order to be consistent with previously established allocation plans.

Note that the Global Data Directory processor stores location information for the local site as well as for remote sites, such that information concerning the data stored locally can be made available to the Distribution Controller.

The manner in which the site location information which is referenced by the Global Data Directory processor is translated into an address for a network node is considered to be outside the scope of data management standardization. The Directory Services standard may be useful in this regard.

6.2.4 User Application Processor

When the user is a programming language agent, a component of the User Application Processor will provide the data type translation function between the data types supported by the programming language and the data types supported by the Database Controller. Ideally, this mapping should be defined only once from each programming language to and from

the data types supported by the Database Controller, and the data types supported by the Database Controller should be the same data types used elsewhere in the Open Systems Interconnection Environment.

User applications of data management services may be custom applications, partially standardized applications, or applications which are completely defined by the specification of a standard, depending on the number of users with similar requirements for a certain type of services. As described in section 6.3, the Information Resource Dictionary System standard is an example of a standardized user application which is within the scope of data management standardization. Other such standardized applications may exist in other areas.

The "user" may be either a human user or another processor.

6.2.5 Remote Database Access

The Remote Database Access function is the link between OSI and data management standards. This service provides a client-server communication mechanism between a single pair of communicating real systems. The specification and implementation of remote database access functions is facilitated by the language-independent nature of the Database Controller.

Direct invocation of the remote database access service by the User Application Processor remains possible in the absence of a processor fulfilling the role of the Distribution Controller. This will provide a transition mechanism in the period before the availability of DDBMS Distribution Controllers able to utilize standardized data directory information.

6.2.6 Access Control Processor

Associated with most requests for data under the control of the database controller function is the requirement to control access to such data. This control is essential in order to provide the required degree of "encapsulation" which will permit the "data hiding" demanded by the object-oriented paradigm, while enabling shared use of common data management facilities. Thus all requests for database access to the database controller are screened by the Access Control Processor, and only valid requests are permitted. The Access Control Processor again utilizes the facilities of the Database Controller in order to store the database of information needed in order to verify access control provisions.

6.2.7 Features of the Proposed Architecture

The data modelling facility used by the Distribution Controller, the Global Data Directory Processor, and the Database Controller is assumed to be consistent. While this data modelling facility will ideally be SQL relational as defined in ISO 9075 in the near term, there is nothing in the proposed framework precluding the replacement of SQL with a totally different data model.

The proposed architecture acknowledges the existence of numerous data modelling facilities but avoids the complications involved in such heterogeneity by insisting on commonality of the data modelling facility within a standardized data management environment. This approach is very similar in nature to the OSI approach to interconnection, where heterogeneity is addressed by requiring that vendors of conforming products

map to a common set of conventions for the representation of information to be communicated (see Figure 2.2).

This approach has been taken since it is not practical to provide for standardized data management system interfaces that permit data model translations. However, this does not preclude the provision of data model translation facilities between the standardized Database Controller function and the data model used by a DBMS not conforming to the data modelling facility standard of the Database Controller (such as IMS or IDMS). Also, it does not imply that data model translation capabilities will not exist in conjunction with the User Application Processor. Standardization of such human interface components for specific application areas may be advantageous. Existing standards which focus on user interface aspects, such as the ANSI IRDS standards, may be incorporated into the architecture by converting to the standard data modelling facility.

Figure 6.1 shows a client (initiating) site communicating with a single remote site. This is strictly for illustrative purposes, as any number of RDA associations, from 0 upwards, may be associated with a particular client request. Such RDA associations may be used both for access to remotely stored data directory information and for access to remotely stored data at any level. Note that while Figure 6.1 illustrates the generic component architecture and the relative placement of the various processors, there are other possible implementation scenarios. For example, a particular node may contain only an RDA client process in conjunction with a User Application Processor, in which case the DDBMS control functions

associated with the Distribution Controller will be obtained from another site.

For operation in a local system environment the Distribution Controller and the Global Data Directory processor functions are not required and may be bypassed. Information concerning which data resides at the local site and thus may be accessed in this manner is provided by the Global Data Directory processor function.

No "block server" function (to access operating system controlled blocks of data) is shown in the model. While it is acknowledged that the Database Controller Processor is going to make use of some form of file access facilities, the required functionality is not defined and need not be from the perspective of the users of data management services. Since the need for a data management standard that specifies an interface to such services is not recommended, provision of such facilities is not shown in the proposed model. Similarly, since no requirement for standardization efforts in the area of the internal schema is foreseen, no data storage description services processor appears in the proposed architecture.

6.3 Use of the Architecture for Information Resource Dictionary Access

Figure 6.2 shows the use of the proposed architecture in support of Information Resource Dictionary System access.

The Information Resource Dictionary standards define a standardized application processor (one of many possible such standardized applications) in which added-value facilities are required to build upon the capabilities provided by the Database Controller. As in the specification of Database

Figure 6.2

Controller functionality, the services interface defined by the IRDS application should be defined using an abstract syntax unrelated to specific programming languages.

The processing facilities of the Information Resource Dictionary standardized application may be used in order to provide added-value functionality that encapsulates dictionary services such as versioning and life-cycle control. This function provides a facility permitting an object-oriented view of the metadata being accessed. The IRDS application, as with any application requiring access to data above the application level, identifies to the Database Controller the level of data to be accessed, always with reference to the same basic data manipulation commands. As a result, the IRDS application can be defined to control all access (or all updates) to data above the application level. Other user applications may also access data above the application level if duly authorized (i.e. if they satisfy the requirements of the access control processor). Thus the IRDS application **may** retain exclusive privileges for access to all data above application level data. Update of data above the application level, whether creation or manipulation of such data, can then be controlled by the Information Resource Dictionary application processor.

The Information Resource Dictionary application, when included as an active component of the data management architecture, is assigned responsibility for the control of all "metadata". The IRDS "added value" facility must then satisfy access control provisions using the Database Controller to update and retrieve all data above the application level. Modification of the IRDS and SQL2 standards in order to accommodate this design will eliminate the duplication involved in having separate SQL Data

Definition and Manipulation Languages which replicate the schema definition portion of the IRDS-provided services. A side benefit of such an approach is that the "active" and "dynamic" qualities desirable in a dictionary facility will be provided as a consequence of the standards design, in that the IRDS application shares common data with other applications that have need of it (and are authorized to access it).

Note that utilization of the same DML commands for schema data as are used for application data instances does not preclude the possibility of vendor proprietary (or standardized) "macro" commands that take into consideration the effect of schema changes on populated databases.

6.4 Examples of the Use of the Architecture in Handling Sample Requests

6.4.1 Retrieval of Application Level Data

Suppose that a user of a particular application at site A wishes to find information corresponding to the request (in English language):

"Find all personnel information for employees in departments X and Y".

Suppose further that such personnel information exists in one (normalized) relation, but that the data is horizontally fragmented so that data about employees of department X is contained at the local site (A), while employee tuples for employees of department Y are contained at sites B and C.

In order to satisfy such a request, the following course of events would occur (with reference to Figure 6.1):

Regardless of the man-machine interface or data model characteristics that are visible to the user of the User Application Processor, the user's application is responsible (possibly through a standardized data model

translation facility) for presenting the request to the data management software in standard format. Thus the request received at the local site will be in terms of the data model understood in common by the Distribution Controller, Global Data Directory, and Database Controller Processors. The Distribution Controller Processor will make use of standard format information provided by the Global Data Directory Processor in order to determine if data about employees of department X is available locally. Whether the location data is stored at the local site depends on the selected choice of distribution alternative. However, network traffic would be reduced by not requiring remote access to directory information indicating that the data was stored at the requesting site.

The Distribution Controller Processor would use directory information at site A, if available, in order to determine that data concerning employees of department Y was contained at sites B and C. (It would be probable that information stored at site A would contain information on the relative cost of communication to alternative sites at which replicated data may be stored.) If directory information about employees of department Y was not available locally, the local site would retain information about where to look remotely for such directory information. Access to such remote directory information would be done by means of RDA requests which would target the global data directory processor at remote sites. Such a scheme could be as simple as a pointer to one or more sites where complete directory information was maintained. (If the request had been for data known to be available at the local site, and no updating of replicated data was required, it would be possible to bypass the Distribution Controller Processor completely.)

Having determined that the required information was available at lowest cost at site C, the Distribution Controller Processor would issue a remote database access request to site C. Determination of the network address could be found through an X.500 conforming Directory Services processor in a network supporting such a facility. A request for information stored at the local site would also be issued to the Database Controller Processor of site A. Assuming that the required information concerning employees of department Y was successfully located at site C, the Distribution Controller Processor would combine the resultant table returned from site C with the table resulting from the local request in responding to the User Application Processor. Note that the Distribution Controller Processor would be responsible, in the same way that local DBMSs are, for performing supporting functions such as ordering or aggregation.

If the required information was not obtained successfully from site C within a required interval, the Distribution Controller Processor would be able to issue a subsequent request for the same data to site B.

Note that there is no assumption that the language used throughout the DDBMS network is necessarily standard-conforming SQL. What is assumed is that all of the Distribution Controller, Global Data Directory, and Database Controller Processors are able to converse using the same language. It is further suggested that in the near term this language should be standard-conforming SQL, and, in the medium term, the enhanced facilities of SQL2. The Remote Database Access SQL specialization is then able to provide a service to the Distribution Controller Processor which,

except possibly for performance and cost differences, makes it irrelevant whether or not data to be accessed is located locally or remotely.

There is also no assumption that the DBMS supporting the Database Controller Processor function at each site of the network internally utilizes the data structures and operations of SQL. Any proprietary DBMS may be used as long as a translation to the SQL standard Database Services Interface is supported by the vendor.

6.4.2 Retrieval of System Documentation Metadata

Suppose that a Data Administrator using an Information Resource Dictionary System at site A wishes to find information corresponding to the request (in English language):

"Find all production files containing the social insurance number data element, along with documentation on all programs accessing these files".

Suppose further that "programs" and "files" are entity types known to the IRDS standardized application, and that programs may access files which contain data elements. Assume that such programs may exist at any site of the DDBMS network of interest. Also assume that information concerning all program documentation which is held in machine-readable format is stored at site D, and that the file numbers for locating hard-copy program documentation are stored at site E.

In order to satisfy such a request, the following course of events would occur (with reference to Figure 6.2):

The Information Resource Dictionary Processor would process two separate requests. The first would result in an SQL select statement that would be passed to the Distribution Controller Processor. The Distribution Controller Processor would access the local Global Data Directory Processor in determining which sites must be accessed in order to locate complete information concerning all files known to the system. The request to be sent to each such site would be an SQL request for file information on files where a column description was the social insurance number. The fact that metadata, rather than application level data, was being accessed would have little effect on the actual processing of the query, as long as the User Application Processor (i.e. the Information Resource Dictionary Processor in this case) was entitled to access the information of interest. The sequence of events would be much as described in 6.4.1. The Database Controller Processor function at each site would receive an input indication identifying the level of data to be accessed.

In order to find documentation on each of the programs accessing each of the files thus retrieved, the Global Data Directory processor would report to the Distribution Controller Processor the fact that information about programs was available at sites D and E. The references from site E would be in the form of pointers to the hard-copy records management system where the program documentation was stored. The actual program documentation, in the form of compound electronic documents, would be provided at the database services interface of site D. The length of such documentation would not be of concern since the prescribed services interface would avoid the problem of the maximum parameter lengths associated with the current programming language interfaces. The format

of the electronic document could be an octet stream as prescribed by the Office Document Interchange Format standard [73].

Chapter 7

Thesis Contributions and Future Studies

The advancement of distributed database management as a viable technology demands the establishment of international standards. Such standards must address a number of complex issues that relate to current data management standards and interprocess communication standards. In addition, consideration must be given to the manner in which data management operations are specified and implemented in a shared memory environment.

Specific contributions resulting from the work of this thesis are as follows:

1. A viable approach providing for both homogeneous and heterogeneous distributed database management has been suggested and, as a result of contributions prepared during the development of this thesis, the ISO Reference Model of Data Management has been modified in consideration of the architectural approach suggested in Chapter 6.
2. Contributions resulting from the analysis done during the course of this thesis [34] have contributed to the elimination of the Object-Association data model from the ISO IRDS standard specification. This modification, which has resulted in the use of standard SQL for the specification of the IRDS Services Interface, is consistent with the goal of restricting to one the number of data modelling facilities within a distributed environment.

3. The research done in the preparation of this thesis provides a review of common requirements for the data management component of distributed applications. The approach taken is analogous to an analysis effort on upgrading an information processing system which has been in existence for a number of years. The "existing system" is the current base of data management, programming language, and interprocess communication standards, and the implementations based on these standards. Analysis of an aging information system would likely result in the need for a complete "rewrite". The extent of investment in technology based on existing standards rules out the possibility of a "rewrite" in order to satisfy DDBMS requirements. Rather than start from the beginning, it is necessary to redesign so as to permit the utilization of existing components.

This analysis has resulted in the following observations:

a) Extensive overlap was found to exist among data management standards targeted for the single-site environment; this is not surprising given that various data management standards were not controlled in their scope by any established framework. It has been found that taking a different approach to the specification of the interfaces has considerable potential to provide remote access to data management functions which were previously standardized assuming the availability of shared memory. It is proposed that a standardized data management "block" of functionality can be used extensively in handling differing requirements within the suggested framework. The proposed model allows the use of SQL to provide basic data management services in many contexts. These range from the multiple data levels associated with the IRDS application to the provision of location information for distribution control.

b) Sensible solutions to the realization of the promise of distributed database management must limit the complexity of the problems to be addressed. Just as practical solutions to the intricacies of interprocess communication were resolved using Open Systems Interconnection, pragmatic implementations of DDBMSs must be based on a software structure less complex than a completely heterogeneous DDBMS. The communications aspects should be based on the building blocks provided by OSI standards. While it is necessary to recognize the heterogeneous nature of distributed database management, it is not practical to attempt to build DDBMSs capable of performing the mappings between the data models supported by a myriad of different proprietary DBMSs. What is practical is to base distributed DBMSs on standards for interprocess communication and on standards for database structures and operations. This aids in the reduction of the complexity of DDBMSs by only requiring each database vendor to conform to the requisite interprocess communication and database standards. This is completely analogous to the manner in which OSI reduced to manageable proportions the exponential number of conversions required by the gateway approach. This was accomplished by developing a set of conventions for representing the information to be communicated between open systems. While it is not possible to eliminate the requirement for data model translations, the proposed architectural model limits the number of required conversions to the order of the number of systems involved.

c) The architectural framework which has been suggested proposes a number of modifications to the specification of single-site data management standards in order to provide better support for distributed databases. The suggested architecture focuses on the aspects of distributed database

processing relevant to openness, while attempting to accommodate future technological change. The proposed framework does not constrain topics which are subject to considerable DDBMS research, such as distribution alternatives and replication strategies. The model is viable for both federated heterogeneous databases and top-down DDBMS designs, where a greater degree of homogeneity can be enforced.

Both vendors and users have an interest in achieving a greater degree of stability in order to justify large expenditures on distributed database development. Consensus on a standards framework which addresses distributed database requirements is thus an issue of paramount importance to both vendors and users of distributed database management systems.

The following areas are suggested for further research:

1. Determination of the appropriateness of standardized remote procedure call mechanisms for distributed database interworking. Work in this area should include consideration of the use of "lightweight" processes (utilizing shared memory) to facilitate performance-oriented Remote Procedure Calls for parallel processing architectures. Another important aspect of the use of remote procedure calls is to determine the optimal manner in which OSI facilities and RPCs may be combined, as suggested in [3]. Such an approach may afford a more flexible strategy for the definition and invocation of application layer services supporting the various communication possibilities outlined in section 5.2.2.
2. Analysis of requirements for a general-purpose naming scheme for distributed tables.

3. To research the requirements of Directory Services and OSI Management against the proposed framework in an attempt to prevent the current extent of duplication of effort by using the suggested general-purpose distributed database mechanisms.

4. The Distribution Controller Processor is shown in Figure 6.1 as a single block of functionality, whereas in actual fact it may well consist of a number of separable components. Fragment definition is logically distinct from allocation of such fragments to various sites of the network, and separate standardized services interfaces may be appropriate in the future. Decomposition of the functions of the Distribution Controller Processor is premature given the existing state of development of data management standards relating to the distributed environment, but future standardization efforts will be required in this area.

5. Further work is required in determining the optimal manner of specification of data management operations, with consideration of the need for programming language independence, as discussed in sections 4.9 and 4.11.

REFERENCES

- [1] Ceri, Stefano and Pelagatti, Giuseppe, **"Distributed Databases - Principles and Systems"**, McGraw-Hill, 1984.
- [2] Stocker, P.M., Gray, P.M.D., and Atkinson, M.P., Eds., **"Databases - Role and Structure"**. Cambridge University Press, 1984.
- [3] Standard ECMA-127, **"Basic Remote Procedure Call Using Remote Operations"**, ISO/IEC JTC1/SC 21 N2989, 1988, unpublished.
- [4] **"Each Piece in Its Place"**, Unix Review, June 1987.
- [5] ISO DP 10032 **"Reference Model of Data Management"**, ISO/IEC JTC1/SC 21 Document N2641, 1988, unpublished.
- [6] Olle, T.W., **"Data Modelling Facilities, Level Pairs and Level Triples"**, SC21/WG3 N634, unpublished.
- [7] Whittington, R.P., **"Database Systems Engineering"**, Oxford University Press, New York, 1988.
- [8] Coulouris, George F., and Dollimore, Jean, **"Distributed Systems - Concepts and Design"**, Addison-Wesley, 1988.
- [9] Gibbons, P.H., **"A Stub Generator for Multilanguage RPC in Heterogeneous Environments"**. IEEE Transaction on Software Engineering, Vol.SE-13, No. 1, Jan. 1987.
- [10] O'Connell, Daniel R., **"A Matter of Semantics"**, pp.51-54, Datamation, Dec. 15, 1988.
- [11] Jones, M.B. and Rashid, R.T., **"Mach and Matchmaker: kernel and language support for object-oriented distributed systems"**, ACM Sigplan Notices, vol. 21, no. 11, pp.67 - 77, 1986.
- [12] Slonim, Jacob; Schonbach, Abraham; Bauer, Michael A.; MacRae, Lachlan J.; Thomas, Keith A., **"Building an Open System"**. Van Nostrand Reinhold Company Inc., New York, 1987.
- [13] Panzieri, F. and Randell, B. **"Interfacing UNIX to Data Communications Networks"**, IEEE Transactions on Software Engineering, Vol. SE 11, No. 10, October, 1985, 1016-1031.

- [14] **"A proposal to add a distribution and federation specification capability to SQL"**, Contribution to JTC1/SC21/WG3 by BCS/CODASYL DBAWG, June 1988, unpublished.
- [15] **"Information Processing Systems - Text Communication - Remote Operations - Part 1: Model, Notation and Service Definition"**. Draft International Standard 9072-1.2 (final text), Document ISO/IEC JTC1/SC21 N3881, 1989.
- [16] **"Information Processing Systems - Text Communication - Remote Operations - Part 2: Protocol Specification"**. Draft International Standard 9072-2.2 (final text), Document ISO/IEC JTC1/SC21 N3882, 1989, unpublished.
- [17] **"The Upper Layers of the OSI Reference Model - A Tutorial Review of Current Work on Standards"** W.A. Melendez and E.L. Petersen, ISO JTC1/SC 21 N66, 1986, unpublished.
- [18] Lynch, Clifford A., **"Extending Relational Database Systems for Information Retrieval Applications"**, Ph.D. Thesis, University of California, Berkeley, 1987, (unpublished)
- [19] Gretton-Watson, Paul, **"Distributed Database Development"**, Computer Communications, Vol. 11, October 1988.
- [20] **"Information Processing Systems - Open Systems Interconnection - Distributed Transaction Processing - Part 3: Protocol Specification"**. ISO 2nd DP 10026-3, Document JTC1/SC21 N3363, 1989, unpublished.
- [21] Bozman, J.S., **"Distributed DBMS means cost, trouble"**, Computerworld, Vol. XXIII, No. 6; Feb. 13, 1989.
- [22] **"Method of defining syntactic metalanguage"**, British Standards Institution Standard BS 6154, 1981.
- [23] Tolhurst, M.R. (Ed.), **"Open Systems Interconnection"**, MacMillan Education Ltd., London, 1988.
- [24] Lorin, Harold, **"Aspects of Distributed Computer Systems"**, 2nd ed., John Wiley & Sons, New York, 1988.
- [25] Zajackowski, J.A., British Telecom, **"An introduction to the CCITT/ISO Standard on Transfer Syntax and Notation"**, April 1986, unpublished.
- [26] **"Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)"**. Canadian Standards Association Standard CAN/CSA-Z243.161-89, Toronto, 1989.

[27] **"Information Processing Systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)".** Canadian Standards Association Standard CAN/CSA-Z243.162-89, Toronto, 1989.

[28] **"Statement of IRDS Objectives for Inclusion in ISO Standard",** ISO TC97/SC21/WG3 document N87, unpublished.

[29] **"Approved Commentaries on the Basic Reference Model for Open Systems Interconnection",** Document ISO/IEC JTC1/SC21 N3452, 1988, unpublished.

[30] **"Message Handling Systems - Abstract Service Definition Conventions".** ISO JTC1 Document SC21 N2276, 1987, unpublished.

[31] Holloway, Simon (Ed.), **"The Future of Data Dictionaries",** The British Computer Society Database Specialist Group, Gower Technical, Hants, England, 1989.

[32] **"Data Types for Data Interchange - A report of the SPARC Data Interchange Study Group",** ANSI X3H2 Committee Paper number 85-99, 1985, unpublished.

[33] Catley, B.L., **"Contribution to the Reference Model of Data Management",** ISO/IEC JTC1/SC21/WG3 document N694, 1988, unpublished.

[34] Catley, B.L., **"Canadian Contribution to the Reference Model of Data Management",** ISO TC97/SC21/WG3 document N811, 1989, unpublished.

[35] Gerhardt, R.W., **"Remote Database Access",** Gateway, May/June, 1988.

[36] **"A Component Architecture for Database Management Systems",** U.S. Department of Commerce Report NBS-GCR-81-340, Washington, D.C., 1980.

[37] **"Information Processing Systems - Open Systems Interconnection - Remote Database Access Tutorial",** Document ISO/IEC JTC 1 SC 21 N3343, 1989, unpublished.

[38] **"Information Processing Systems - Concepts and Terminology for the Conceptual Schema and the Information Base",** ISO 9007, Technical Report edited by J.J. van Griethuysen, 1982.

[39] **"Information Resource Dictionary System Services Interface",** ISO/IEC JTC 1/SC21/WG3 document N752, 1989, unpublished.

[40] **"Information Processing Systems - Open Systems Interconnection - Generic Remote Database Access Service and Protocol"**, ISO Draft Proposal 9579, Document ISO/IEC JTC 1/SC 21 N3606, June 1989, unpublished.

[41] **"Data Management Export/Import: Generic"**, ISO/IEC JTC1/SC 21/WG3 document N807, 1989, unpublished.

[42] **"Application Portability"**, Report PSC-ARC003-1, prepared for National Archives of Canada, Protocols Standards and Communication Inc., 1988, unpublished.

[43] **"ISO working draft Database Language SQL2"**, ISO/IEC JTC 1/SC21/WG3 document N985, December, 1989, unpublished.

[44] **"Information Processing Systems - Open Systems Interconnection - Remote Database Access: SQL Specialization"**. Document ISO/IEC JTC 1/SC 21/WG3 N844, June 1989, unpublished.

[45] **"Information Processing Systems - Information Resource Dictionary System Framework"**, Draft International Standard 10027, Document ISO/IEC JTC 1/SC 21 N3426, 1989, unpublished.

[46] **"An Architecture for Database Management Standards"**, U.S. NBS Special Publication 500-86, Washington, D.C., 1982.

[47] **"Reference Model for DBMS Standardization"**, Report of the Database Architecture Framework Task Group of the ANSI/X3/SPARC Database System Study Group. ACM SIGMOD Record, Vol. 15, No. 1, March 1986.

[48] **"Procedure Language Access to Draft Proposed American National Standard Database Language SQL"**, ISO/IEC JTC 1/SC21 document N625, 1987, unpublished.

[49] Oxman, Steven W., **"Data Management Standardization for ACE ACCIS"**, NATO Technical Memorandum STC TM-776, 1985, unpublished.

[50] Cardenas, Alfonso F., **"Heterogeneous Distributed Database Management: The HD-DBMS"**, Proceedings of the IEEE, May 1987.

[51] **"Message Handling Systems - Abstract Service Definition Conventions"**, Document ISO/IEC JTC 1/SC 21 N2276 (DIS 8505/4), 1987, (also CCITT Draft Recommendation X.407), unpublished.

[52] **"Information Processing Systems - Open Systems Interconnection - Basic Reference Model"**. Canadian Standards Association Standard CAN/CSA-Z243.100-86 (ISO 7498-1984), Toronto, 1986.

- [53] Sheppard, Donald A., **"Information Technology Standards: Fact, Fiction or Folly?"**, CIPS Review,, The Canadian Information Processing Society, January/February 1989, Toronto, 1989.
- [54] Ullman, Jeffrey D., **"Principles of Database and Knowledge-Base Systems"**, Volume 1, Computer Science Press, Maryland, U.S.A., 1988
- [55] Haerder, Theo, and Reuter, Andreas, **"Principles of Transaction-oriented Database Recovery"**, ACM Computing Surveys, Vol. 15, No. 4, December 1983, pp. 287-317.
- [56] Chu, Wesley, **"Performance of File Directory Systems for Databases in Star and Distributed Networks"**, AFIPS Conference Proceedings, Volume 45, 1976.
- [57] **"Information Processing Systems - Database Language SQL"**. Canadian Standards Association Standard CAN/CSA-Z243.47-88, ISO 9075-1987), Toronto, 1988.
- [58] **"Information Processing Systems - Database Language NDL"**. ISO 8907-1987), Geneva, 1987.
- [59] **"American National Standard X3-138-1988, Information Resource Dictionary System"**, American National Standards Institute, New York, 1988.
- [60] **"Information Processing Systems - Open Systems Interconnection - Service Definition for the Commitment, Concurrency and Recovery Service Element"**. ISO/IEC/DIS 9804.2, December 1988, unpublished.
- [61] **"Information Processing Systems - Open Systems Interconnection - Protocol Specification for the Commitment, Concurrency and Recovery Service Element"**. ISO/IEC/DIS 9805.2, December 1988, unpublished.
- [62] Lewan, D. and Long, H.G., **"The OSI File Service"**, Proceedings of the IEEE, Vol.71, No. 12, December 1983.
- [63] **"Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 1: General Introduction"**. Canadian Standards Association Standard CAN/CSA-Z243.180-89 (ISO 8571-1-1988), Toronto, 1989.
- [64] **"Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 2: Virtual Filestore Definition"**. Canadian Standards Association Standard CAN/CSA-Z243.181-89 (ISO 8571-2-1988), Toronto, 1989.

[65] **"Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 3: File Service Definition"**. Canadian Standards Association Standard CAN/CSA-Z243.182-89 (ISO 8571-3-1988), Toronto, 1989.

[66] **"Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 4: File Protocol Specification"**. Canadian Standards Association Standard CAN/CSA-Z243.183-89 (ISO 8571-4-1988), Toronto, 1989.

[67] Goldfine, Alan and Konig, Patricia, **"A Technical Overview of the Information Resource Dictionary System (Second Edition)"**, NBSIR 88-3700, National Bureau of Standards, Gaithersburg, MD, U.S.A., 1988.

[68] **"Information Processing Systems - Open Systems Interconnection - The Directory - Part 1: Overview of Concepts, Models and Service"**, ISO/IEC/DP 9594-1, SC 21 document N2751, 1988, unpublished.

[69] **"Information Processing Systems - Open Systems Interconnection - The Directory - Part 2: Models"**, ISO/IEC/DP 9594-2, SC 21 document N2752, 1988, unpublished.

[70] **"Information Processing Systems - Open Systems Interconnection - The Directory - Part 3: Abstract Service Definition"**, ISO/IEC/DP 9594-3, SC 21 document N2753, 1988, unpublished.

[71] **"The ANSI/X3/SPARC DBMS Framework - Report of the Study Group on Database Management Systems"**, Tsichritzis, D.C., and King, A., eds., Computer Systems Research Group, University of Toronto, 1977.

[72] **"ISO-ANSI working draft Database Language SQL2 and SQL3"**, ISO/IEC JTC 1/SC21/WG3 document N986, December, 1989, unpublished.

[73] **"Information Processing - Text and Office Systems - Office Document Architecture (ODA) and Interchange Format - Part 1: Introduction and General Principles"**, ISO/IEC JTC 1/SC 18 document N1163, 1987, unpublished.

[74] Shapiro, N.R., **"SFQL: Structured Full-Text Query Language Specification Version 1.2"**, GE Corporate Research & Development, 1989, unpublished.

[75] **"Information Processing Systems - Open Systems Interconnection - Systems Management Overview"**, ISO/IEC/2nd DP 10040, SC 21 document N4066, 1990, unpublished.

[76] Chen, P., **"The entity-relationship model - Toward a unified view of data"**, ACM Trans.Database Systems 1, March 1976.

END

25/02/91

FIN