TECHNICAL REPORT

ISO/IEC TR 10032

First edition
2003-11-01

Information technology — Reference Model of Data Management

Technologies de l'information — Modèle de référence pour la gestion de données

Reference number
ISO/IEC TR 10032:2003(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In exceptional circumstances, the joint technical committee may propose the publication of a Technical Report of one of the following types:

— type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

— type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

— type 3, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC TR 10032, which is a Technical Report of type 3, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

Introduction

ISO, in specifying a Reference Model of Data Management, recognizes that there are many implementors of data management systems. It is inevitable that different implementors use different terms to specify or refer to similar data management functions. Furthermore, the use of the same term to describe different functions is also common. There is a clear need to standardize the data management functions. This Technical Report fulfils that role by presenting a Reference Model of Data Management and defining the areas of this model which lend themselves to standardization.

This Technical Report defines the Reference Model of Data Management. It provides a common basis for the coordination of standards development for the purpose of data management, while allowing existing and emerging standards to be placed into perspective.

The term “data management” includes the description, creation, modification, use and control of data in information systems. Such data management functions may be performed as a common service for information systems applications. Alternatively, each application may define and control the data relevant to it. In the case in which data management functions are performed as a common service, it is desirable to provide standardized facilities for data access and control in order to permit the sharing of data by a number of users. Such standardization requires the determination of a number of interfaces for which individual standards may be developed.

The objectives of this Technical Report are to provide a framework allowing, within the scope specified in Clause 1, for the following:

a) identification of interfaces;

b) positioning all such interfaces relative to each other;

c) identification of facilities provided at each interface;

d) identification of the process which supports each interface and, where appropriate, of the data required for such support;

e) positioning the use of the interfaces in terms of an information systems life cycle; and

f) identification of the binding alternatives associated with each appropriate identified interface.

There are three major objectives which are applied in this Technical Report to data management standardization. These are as follows:

a) Shareability of resources;

b) Minimize cost of supporting an information system over its life cycle;

c) Optimum use of standardization effort.

The shareability of resources objective applies to both information resources as represented by data in databases and to processor resources of the kind described in Clause 6. There is particular emphasis on the shareability of information resources located at different places and developed using different hardware and software. All shareability of resources is subject to access control.

The objective of minimizing the cost of supporting an information system applies to all phases of the information system life cycle, including design, development, operation and maintenance costs.

The objective associated with the optimum use of standardization effort refers to reducing the number of standards required and to simplifying the content of such standards.

This Technical Report identifies areas for developing or improving standards, and provides a common framework for maintaining consistency of all related standards.

This Technical Report provides a framework which allows teams of experts to work productively and independently on the development of standards for different components of information systems.

This Technical Report has sufficient generality to accommodate the development of new standards in response to advances in technology.

The description of the Reference Model of Data Management given in this Technical Report is presented as follows:

— Clause 4 introduces data management and the requirements based on information systems;

— Clause 5 explains the data concepts that are required for the Reference Model and how they relate to each other and the process concepts;

— Clause 6 provides an architectural model within which different data and processing components relevant to data management can be placed;

— Clause 7 describes the objectives and principles for data management standardization;

— Annex A is a list of related International Standards;

— Annex B shows how the existing and future SC 21/WG3 standards relate to the architectural model described in Clause 6;

This Technical Report specifies the classes of services that are expected to be provided by data management, and it provides a framework which describes the way in which they are related to each other. However, data management does not exist in isolation but within an environment providing other services such as data storage and communication, as is described in Clause 4.

Prior to completion of work on this Technical Report, data management standards were developed within ISO/IEC as indicated in Annex A of this document. The positioning of such International Standards using this Reference Model of Data Management is described in Annex B.

Information technology — Reference Model of Data Management

1 Scope

This Technical Report defines the ISO Reference Model of Data Management. It establishes a framework for coordinating the development of existing and future standards for the management of persistent data in information systems. See Annex A for references to existing data management standards.

This Technical Report defines common terminology and concepts pertinent to all data held within information systems. Such concepts are used to define more specifically the services provided by particular data management components, such as database management systems or data dictionary systems. The definition of such related services identifies interfaces which may be the subject of future standardization.

This Technical Report does not specify services and protocols for data management. This Technical Report is neither an implementation specification for systems, nor a basis for appraising the conformance of implementations.

The scope of this Technical Report includes processes which are concerned with handling persistent data and their interaction with processes particular to the requirements of a specific information system. This includes common data management services such as those required to define, store, retrieve, update, maintain, backup, restore, and communicate applications and dictionary data.

The scope of this Technical Report includes consideration of standards for the management of data located on one or more computer systems, including services for distributed database management.

This Technical Report does not include within its scope common services normally provided by an operating system including those processes which are concerned with specific types of physical storage devices, specific techniques for storing data, and specific details of communications and human computer interfaces.

A data management standard defines services provided at an interface. It does not impose limitations on how processes are implemented.

2 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

The definitions provided in this clause aim to specify the most technical use of the terms in this Technical Report. The introduction to each term may be presented in a simpler informal description. Some of the terms are defined in other standards, but the following definitions are provided for use in the specific context of data management.

2.1
access control
the prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner. For data management purposes, access control relates to the enabling of authorized access to data and the prevention of unauthorized access. Access control determines the processes which a user may perform

2.2
access control data
a collection of data associated with the definition or modification of access control privileges

2.3
access control mechanism
a mechanism which may be used to enforce a security policy

2.4
application
the data manipulation and processing operations that are related to specific requirements of an information system

2.5
application process
a process which is specific to the requirements of a particular information system

2.6
application system
a collection of application processes which utilizes the services provided by the human-computer interface, communications facility, and data management system to perform the processing necessary to meet the requirements of the information system

2.7
audit trail
a record of the activity taking place in an information system over a period of time

2.8
authorization
a definition of privileges for a specific user identifier

2.9
binding
a process which involves relating a process to specific data definitions

2.10
client
a role filled by a processor when it requests the services provided by another processor (i.e. a server)

2.11
client-server relationship
the relationship between a client and a server which is established at the moment that a client asks for a service to be performed by a server

2.12
communications linkage
a means for exchanging data between computer systems, or between a user and computer systems

2.13
computer system
a collection of hardware which is managed as a single unit by software such as an operating system which may also provide common services such as access control, interprocess communications, and a graphical user interface

2.14
configuration
a set of processes comprising an information system and the way in which the processes are interrelated

2.15
configuration management
an activity of managing the configuration of an information system throughout its life cycle

2.16
constraining rule
a rule which is part of a Data Modelling Facility and which controls the specification of the constraints which may be expressed upon a collection of data

2.17
constraint
a restriction on the values permitted for a given collection of data

2.18
data content standard
a logical specification of a collection of data which is of sufficiently general applicability to be of use in many application systems

2.19
data definition
a description which determines the rules to which one or more collections of data instances must conform

2.20
data export
a data management service which retrieves a set of data from a database and creates a copy of that data organized according to a data interchange format

2.21
data import
a data management service which inserts into a database a set of data organized according to a data interchange format

2.22
data independence
the independence of processes from data such that the data definition may be changed without unnecessarily affecting the processes

2.23
data integrity
conformance of data values to a specified set of rules

2.24
data interchange format
a set of data structuring rules that determine a format for data to enable the export of data from one data management system and its import by another data management system

2.25
data interchange standard
a standard which defines a set of data according to a set of data structuring rules so that the set of data can be interchanged between one computer system and another

2.26
data management
the activities of defining, creating, storing, maintaining and providing access to data and associated processes in one or more information systems

2.27
data management environment
an abstract conceptualization of the data and associated processing elements involved in a computer system

2.28
data management service
a service provided by a data management system

2.29
data management session
a period of time during which a set of data management services are being used by a client of a data management process

2.30
data management system
a system which is concerned with the organization and control of data

2.31
data manipulation process
a process the semantics of which are prescribed by the data manipulation rules of a Data Modelling Facility

2.32
data manipulation rule
a rule which either must be followed when specifying a process or else is automatically followed by a data management system when a process is executed

2.33
data modelling facility
rules for defining a schema and the data manipulation rules for operating on data stored in accordance with the schema

2.34
data structuring rule
a rule specifying how a collection of data may be structured

2.35
data type
a named, formal specification which governs the common static and dynamic properties of all instances of that data type

2.36
database
a collection of data stored according to a schema and manipulated according to the rules set out in one Data Modelling Facility

2.37
database controller
an abstract representation for the collection of services which conform to and implement a Data Modelling Facility

2.38
database environment
a database and its associated schema and database controller

2.39
database language
a language with a formal syntax to be used for defining, creating, accessing and maintaining databases

2.40
database management
creating, using and maintaining databases

2.41
database management system
DBMS
a collection of integrated services which support database management and together support and control the creation, use and maintenance of a database

2.42
dictionary system
an information system containing information about an enterprise, its operations, activities, processes and data that are related to one or more application systems

2.43
distributed database
a collection of data which is distributed across two or more database environments

2.44
distributed information system
an information system, the data and associated processes of which are distributed across two or more database environments

2.45
distribution data
the data which defines location, replication and fragmentation information about data objects in a distributed database system

2.46
fragmentation
a partitioning across more than one database environment of the data values for the instances of one data type in a distributed database

2.47
functional standard
a standard which consists of an assembly of other standards showing how they fit together

2.48
horizontal fragmentation
a fragmentation where the partitions are formed from all data values for a subset of instances

2.49
information system
a system which organizes the storage and manipulation of information about a universe of discourse

2.50
interchange data modelling facility
a data modelling facility that supports the interchange of data between data management systems

2.51
interface standard
a standard which defines the services available at an interface to a process

2.52
level pair
a modelling concept which groups a schema with its associated database. There are two adjacent data levels. The upper level will always contain the definition of data stored on the lower level

2.53
management domain
a domain encompassing a set of two or more information systems, any of which may be distributed, which have been designed and constructed to interchange data and processes

2.54
persistent data
data which is retained in the information system for more than one data management session

2.55
privilege
the authorization given to an identified user to allow the use of a particular data management service to access specific data or processes

2.56
process
a process is an active component of an information system

2.57
processing linkage
a representation of a possible interaction between processors

2.58
processor
a modelling concept that represents some combination of hardware and software that can provide services either to one or more other processors or to a human user

2.59
schema
a description of the content, structure, and constraints used to construct and maintain a database

2.60
server
role filled by a processor when it provides services to another processor

2.61
service
a capability provided by a processor to other processors, or by a process to other processes

2.62
services interface
a defined set of services made available by a process or processor

2.63
session
a period of time during which a client may have many interactions with a server and both the client and server maintain data about each other

2.64
source schema
a data definition or set of data definitions prior to transformation to a schema

2.65
transaction
a set of related operations characterized by four properties: atomicity, consistency, isolation and durability. A transaction is uniquely identified by a transaction identifier

2.66
transient data
data which is either flowing into and out of an information system, or, in the case of a distributed system, between two computer systems

2.67
user processor
a processor which provides services to a human user and which is a client (directly or indirectly) of a database controller

2.68
variant
a configuration of all or part of an information system which coexists with another having a different configuration but providing the same facilities

2.69
version
a configuration of all or part of an information system at a specific point in time

2.70
vertical fragmentation
a fragmentation where the partitions are formed from the same type of data values for all instances

3 Symbols and abbreviations

The purpose of this clause is to identify the symbols and abbreviations used in this Technical Report.

3.1 Symbols

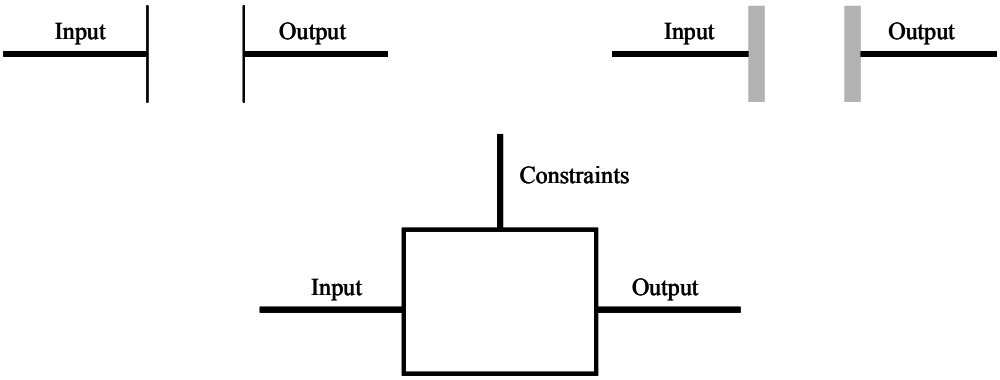
3.1.1 Persistent data

3.1.2 Communications linkage

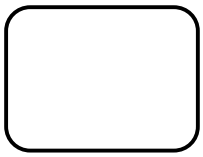
3.1.3 Processing linkage

3.1.4 Process class

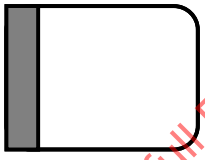
A process class symbol is used to indicate a data manipulation process. A processing linkage at the left edge indicates input, at the right edge indicates output, and at the top indicates constraint.



3.1.5 Processor class



3.1.6 Processor class with service interface



A symbol for a processor class with a service interface is used in diagrams with processing linkages to indicate those interactions in which it participates as a client and those in which it participates as a server. Each processing linkage to a server is connected to the shaded service interface.

3.1.7 Class names

A class of processor is referred to by a capitalized name whereas an instance thereof has only lower case letters.

3.2 Abbreviations

ACID	the set of Atomicity, Consistency, Isolation and Durability properties
DBMS	Database Management System
IRDS	Information Resource Dictionary System
NDL	Network Database Language
OSI	Open Systems Interconnection
RDA	Remote Database Access
SQL	Database Language SQL

4 Data Management Requirements

4.1 Purpose

The purpose of this clause is to describe the following:

a) relevant concepts of information systems,

b) aspects of information systems which place requirements on data management,

c) scope of data management.

4.2 Information systems

In order to function, an enterprise needs to collect, keep, and process information about its own operations, its external environment, and its interaction with its environment. A system which handles these tasks for an enterprise is called an **information system**. Each information system supports a set of organizational requirements, and an enterprise may have one or many information systems to meet its total needs. An information system may be located on one computer system. However, an information system may be spread across two or more computer systems. In the latter case, the information system is classified in this report as a **distributed information system**.

Data flows into and out of an information system, and these interactions may be with either persons or processes, including other information systems. Many interactions may occur concurrently. Each interaction may require an approved authorization.

This Technical Report distinguishes two kinds of data referred to as **transient data** and **persistent data**. Transient data is either flowing into and out of an information system, or, in the case of a distributed information system, between two computer systems. Persistent data always has a representation which is retained in the information system over a period of time.

Data management is concerned with the organization and control of persistent data. A system which performs this function is called a **data management system**.

4.2.1 Context of Data Management in an Information System

Figure 1 shows how a Data Management System is positioned relative to a Computer System and the parts of an Information System such as the Applications Processes, Communications Facilities, and the Human-Computer Interface.

For the purposes of this Technical Report, a Computer System is a collection of hardware which is managed as a single unit by software such as an operating system which may also provide common services such as interprocess communication and a graphical user interface.

All parts of an Information System may be distributed across two or more computer systems. Within a computer system there may be a number of instances of any part of an Information System.

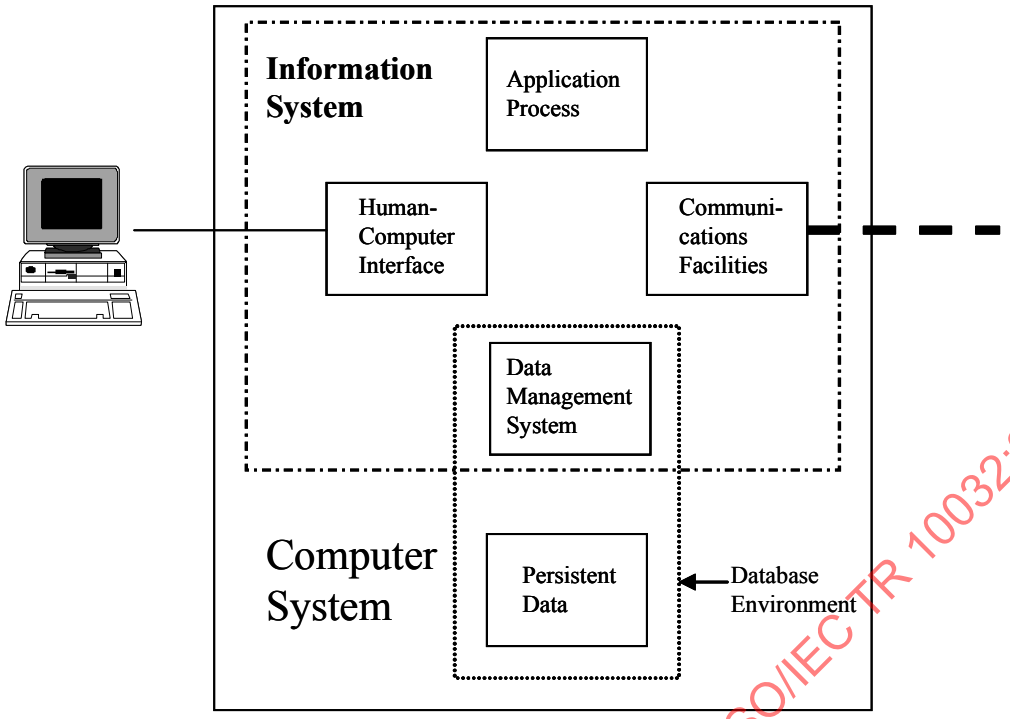


Figure 1 — Position of Data Management System within an Information System

Figure 1 shows the following:

a) The Data Management System provides services that manage a collection of Persistent Data.

b) The interface to human users is provided by the Human-Computer Interface.

c) The Application Process provides capabilities specific to the requirements of a particular Information System.

d) The interface to other Data Management Systems, Information Systems, and Computer Systems is provided by Communications Facilities.

e) Services provided by the Human-Computer Interface may be used by the other parts of the Information System.

f) The Application Process may use services provided by the other parts of the Information System.

g) Each of the parts of an Information System may use services provided by the Computer System.

h) The combination of a Data Management System and Persistent Data is referred to as a Database Environment.

4.3 Database and schema

The persistent data in a database environment comprises a schema and its associated database. A schema is a collection of data definitions which determine the content, structure and constraints used to construct and maintain a database. A database is a collection of persistent data defined by a schema.

A data management system uses the data definitions in a schema to enable and manage access to data in the database defined by the schema.

4.4 Data Modelling Facility

A schema is prepared according to a set of **data structuring rules**. Each set of data structuring rules may have an associated set of **data manipulation rules** which define the processes which may be performed on data structured according to the data structuring rules.

The data structuring rules and data manipulation rules are together called a **Data Modelling Facility**.

It is important to distinguish between the rules inherent in a Data Modelling Facility which are used when preparing a schema and the rules specific to an information system which are defined in a schema. The latter rules represent constraints which are used by a data management system when populating a database conforming to the schema.

A Data Modelling Facility may be specified either in terms of the service provided at the service interface to a data management system, or by a **database language**.

A database management system (DBMS) embodies a data management system and other processes which support the development and use of a database.

A database language is used to define a schema according to data structuring rules and to define processes according to the associated data manipulation rules.

Examples of three classes of Data Modelling Facility are relational, network, and hierarchical. The data structuring rules for Data Modelling Facilities in different classes may be very similar, as for network and relational, but the associated data manipulation facilities may be different.

4.5 Data independence

An objective of data management is data independence which makes it possible for additive changes, and possibly modifications, to be made to the schema in an information system without having to make unnecessary changes to the already existing application processes. This objective is normally achieved in three complementary ways.

The first is by binding the application process to a schema in such a way that the application process is aware only of that part of the schema, namely the application schema, which is needed by the application process.

The second is by ensuring that the application processes do not depend on the physical representation of the data.

The third way of improving data independence is by including as many of the constraints as possible in the schema rather than in the application processes. The extent to which such constraints may be included in the schema depends on the ability of the Data Modelling Facility to define the constraints that are used when defining the schema.

4.6 Data management services

Data management services are provided at the service interface of a data management system. These services support the use of a data modelling facility (whether specified in terms of services or a database language) and all other facilities required for managing persistent data.

Any process may request the use of a data management service available at a services interface. There is a requirement that a services interface is independent of the way the service may be implemented by a data management system and the physical representation of persistent data.

A sequence of requests from one process for data management services relating to one database environment constitutes a **data management session**.

4.7 Processors and interfaces

A data management process may be invoked by a user, data management processes or other processes. The processes will be performed by **processors**, each of which will have an interface. The interface to the processor needs to be specified. Such interfaces may be dependent on the standard programming language used to specify the process using the interface.

At any interface, there will be factors of which the user (human or a process on behalf of a user) will need to be aware to be able to use the underlying processor. These factors should be kept to a minimum to provide as much independence at an interface as possible.

4.8 Access control

In any organizational context, there are particular requirements of access control that may be expressed in terms of a **security policy**. A security policy determines what form of access each user of an information system requires, and an information system must have appropriate **access control mechanisms** which may be used to enforce the security policy.

Within the scope of data management, the requirement is to be able to determine whether to allow any specific request for a data management service to access specific data occurrences by an identified user which is either a person or a process. Access control should be based on an appropriate combination of user identifier, process identifier and referenced data. An access control privilege is assigned to a user to enable the user to perform specific processes on data under specific conditions.

Access control requirements in a data management context can be divided into two categories. Firstly, it must be possible to define and subsequently modify access control privileges. Secondly, it must be possible to enforce, at any time, the need for access control privileges which are required at that place at that time.

4.8.1 Definition and modification of access control privileges

Facilities are required for the definition of privileges, which includes their initial creation, modification, suspension, and deletion. The process of allocating privileges to users is called **authorization**. There is a requirement to identify a global authority who is required to define and modify other access control privileges in a **data management environment**.

Privileges may be defined in terms of the identifier of the user, restrictions on the use of application processes, databases, schemas, data, dates, times, locations and the period of validity of the privilege. Privileges may also be allocated using combinations of the previously mentioned aspects.

Additional information may be required, such as the identifier of the user who authorized the privilege. The data describing privileges is referred to as **access control data**. This data must be stored and managed in the same way as any other data within the scope of data management, including having access controls applied on itself.

4.8.2 Enforcement of access control

The decision to allow any particular access to data is based on the privileges of the user.

The enforcement of access control requires that users, and processes acting on behalf of users, be identifiable, and that the authority to request a service which accesses certain data can be checked at the moment when the request is made.

4.8.3 Security external to data management

The following aspects of security are related to the control of access to data, but are outside the scope of data management for the purpose of this Technical Report:

a) authentication of user identification;

b) protection of stored data so that it may only be accessed by a data management system;

c) protection of communicated data so that it may only be accessed by a data management system;

d) action to be taken (such as reporting and logging for subsequent analysis) in the event of breaches or attempted breaches of security by users;

e) any breaches, or attempted breaches, of security by users without valid privileges must be both reported and logged for subsequent analysis.

4.9 Operational requirements to support data management

There is a set of operational requirements which information systems place on data management, independent of the particular requirements of an information system to store and manipulate data. These operational requirements are as follows:

a) information systems life cycle support;

b) configuration management, version control and variants;

c) concurrent processing;

d) database transaction management;

e) performance engineering;

f) referencing data;

g) extensible Data Modelling Facility;

h) support for different Data Modelling Facilities at user interface;

i) audit trails;

j) recovery;

k) logical data restructuring;

l) physical storage reorganization.

Data management provides generalized facilities to meet these requirements, so that it is not necessary to develop specific solutions for each information system.

4.9.1 Information systems life cycle support

Each application system goes through a life cycle consisting of several phases. This may start with an information systems planning phase to determine which application systems are needed in an enterprise and continue through analysis, design, construction, operation, revision and phase out. The precise definition of a life cycle is considered outside the scope of this Technical Report.

There are three types of requirements which arise from supporting an information system life cycle. Firstly, the fact that an application system may need to be revised after it has entered an operational phase places major requirements on the functionality to be provided by a data management system, for example the ability to modify a schema or a process. Secondly, there is a requirement for recording information about the life cycle and providing means of controlling the actions within and between each of the phases. Thirdly, there is a requirement to store, modify and retrieve process definitions and data definitions with version control.

4.9.2 Configuration management, version control and variants

The activity of managing the changes made to the configuration of an information system over a period of time is called **configuration management**. The activity may be a continuous one with no configuration having any special status. However, it is frequently necessary for very practical reasons to identify discrete **versions** of the system configuration at specific points in time. In addition to managing the changes to the system configuration, it is also necessary to keep track of the configuration which belongs to each specific version.

While an information system is at certain phases in its life cycle, the persistent data and processes which are part of that information system may be required to exist concurrently in different forms of representation. During a revision phase, new data types and new processes may need to coexist with the others which have been already tried and tested.

Alternatively, two forms of a process may be regarded as different **variants**. This means that each variant meets different requirements (such as differing internal storage representations) and the one variant is not intended to replace the other.

4.9.3 Concurrent processing

An information system is a resource which may be shared by several users concurrently. This concurrency requirement is significant even if only one user is involved in using the information system. A user can initiate a request for the services of a data management system which can be handled more expediently if access to the data can be made concurrently. The data management environment must ensure that the separate intent of each user is carried out in a way which is consistent with their perception of the data.

It is a requirement that concurrent interactions must not interfere with each other. It is also a requirement that the data integrity shall not be affected by the concurrent processing.

4.9.4 Database transaction management

A database transaction is defined as a delimited sequence of database interactions which together form a logical unit of work. In the cases of database update, a database transaction is a sequence of update steps, including addition and deletion, which change the content of a database from one consistent state to another. The consistent state of the database of an information system should be evaluated in terms of the schema to which the database conforms, and possibly in terms of the rules embedded in application processes.

The ACID test of a system's quality identifies the following requirements for the management of database transactions:

a) Atomic — the effects of all changes must either persist in the database after the database transaction has been completed or else none may remain.

b) Consistent — at completion, a database transaction leaves the database in a consistent state.

c) Isolated — uncommitted changes made by a database transaction must be invisible to any other concurrent database transaction and vice versa.

d) Durable — once committed, the system must ensure that the results of the database transaction survive any subsequent malfunctions.

Ensuring that such properties of database transactions are realized is identified as the concept of “serializability”. The concurrent execution of several database transactions must be serially equivalent in the

sense that performing them concurrently is the same as if they are executed to completion in some arbitrary order.

4.9.5 Performance engineering

There is a requirement to be able to improve the performance of any computerized information system, whether it is an application system or a dictionary system or a system in which the two are highly integrated.

The basis for making such improvements depends on the collection of statistics regarding the frequency of use of processes and the frequency of access and change to data.

4.9.6 Referencing data

All data in a database environment must be uniquely distinguishable from other data in the same database environment. Where necessary qualifications should be used to achieve this.

A name may be a user assigned name or a name assigned by the data management system. In the latter case, the name may have no meaning to users.

This naming requirement exists for application systems, dictionary systems and for any other kind of information system. Each database environment in a computer system needs to be distinguishable from others.

4.9.7 Extensible Data Modelling Facility

While one Data Modelling Facility may be the standard for a data management system, there is a requirement to be able to add data types and the associated processes special to such data types. An example of this requirement is that of full text processing in conjunction with the processing of structured data.

4.9.8 Support for different Data Modelling Facilities at user interface

A user may prefer to manipulate data according to a Data Modelling Facility different from that provided by the data management system. Hence, there is a requirement to be able to map the data between the data management system's preferred format and the user's preferred format.

4.9.9 Audit trails

It may be necessary for auditing purposes to be able to maintain a record of successful changes to data in a database, and in some cases to keep a record of the transactions which query the data and generate reports. This record may include the relevant data values, details of the transaction and identification of the initiating user. These **audit trails** may be specified as being required for all data in a database, selected types of data or specified data occurrences.

4.9.10 Recovery

There is a requirement to be able to return a database to a prior consistent state. This requirement may arise because of an erroneous transaction, a system failure or loss of the stored data. Various mechanisms can be used to meet this requirement, such as the recording of all changes made to a database and keeping backup copies of all or part of the database.

4.9.11 Logical data restructuring

Logical data restructuring is defined as the process of changing a data definition after an information system has been in use for some time. A change may be additive to an existing data definition or it may comprise a modification to a part of the existing data definition.

A purely additive change will usually affect only those application processes which need to take advantage of it. However, certain modifications may affect existing data definitions and it is necessary to ensure, as part of the restructuring, that the data and schema are consistent.

4.9.12 Physical storage reorganization

Physical storage reorganization is defined as the process of changing the representation of the persistent data in a storage medium.

4.10 Additional operational requirements to support data management in a distributed information system

In addition to the operational requirements arising from any information system, there is a set of further requirements which arise from distributed information systems. In such cases, the data being managed is stored in more than one computer system.

In a distributed information system, data belonging to this information system is stored in two or more database environments, each of which is contained within one computer system.

It is possible that the identity of the system from which the service is being requested is not known to the requester. Alternatively, the service being requested may be available from a number of sites holding replicated data.

The operational requirements dependent on the data being distributed are as follows:

a) distribution control;

b) database transaction management;

c) communications;

d) export/import;

e) distribution independence;

f) system autonomy;

g) recovery of a distributed database.

Some of these requirements are also applicable to an information system which includes more than one database environment within a single computer system.

The first refers to the degree to which the distribution of the data can be managed. This varies between two extremes.

a) At the one extreme, there is no management of the distribution of the data. This means that each application process is responsible for naming (but not necessarily locating) the database environment in which any required data is available. Each database environment is autonomous.

b) At the other extreme, the distribution of the data across two or more database environments is fully coordinated to the extent that an application process is not aware of whether or how the data is in fact distributed. The collection of all data is referred to as a **distributed database**. This distributed database is managed so that it is internally consistent and conforms to the definitions in one schema. This schema may itself be distributed in some way.

The second categorization refers to the distribution scenarios which describe alternative ways in which a distributed information system may be developed or evolve. These are as follows:

a) a distributed database system in which the constituent database environments are designed in such a way that interaction between the database environments is possible;

b) a federated database system in which two or more separately designed database systems are brought together in some sense after a period of separate use and made to function as a single distributed database system;

c) a situation in which each database environment conforms to a set of standards and is hence able to interact (possibly on an *ad hoc* basis) with other database environments each of which was designed separately but conforming to the same standards.

4.10.1 Distribution control

Distribution control includes control of **fragmentation** and control of replication.

Fragmentation is a partitioning across more than one database environment of the data values for the instances of one data type in a distributed database. It is required for reasons of performance and availability to be able to fragment the data in a distributed database in various ways. The two most commonly identified ways are **horizontal fragmentation**, which is a fragmentation where the partitions are formed from all data values for a subset of instances, and **vertical fragmentation**, which is a fragmentation where the partitions are formed from the same type of data values for all instances.

Horizontal fragmentation makes it possible to ensure that each database environment includes those instances of each data type that are required by local application processes.

Vertical fragmentation makes it possible to ensure that each database environment includes those values for all instances of each data type that are required by local application processes.

Different strategies for the distribution of data may be required:

a) have no fragmentation;

b) have only horizontal fragmentation;

c) have only vertical fragmentation;

d) allow any combination of horizontal and vertical fragmentation.

When fragmentation is supported in a distributed environment it is an additional requirement that the user of the information system should not need to be aware of how the data is fragmented or distributed between computer systems.

It is often necessary for reasons such as performance or protection against the failure of one computer system to provide a copy of all or part of a database. Such replicated data may be stored in a computer system different from that in which the data is initially created and subsequently controlled. The requirement for fragmentation may be combined with the requirement for replication such that copies of a set of fragments are assigned to two or more database environments. Information about which data is in which database environments needs to be accessible (directly or indirectly) in each database environment.

The requirement for providing copies of all or part of a database creates the requirement for being able to maintain the consistency of all copies of data when the data is being updated. The algorithms which provide control over the replications must also ensure coordinated updates within the transactions.

4.10.2 Database transaction management

In a distributed database, there is a requirement to synchronise the effect of local transaction management systems to ensure that changes to distributed data result in a consistent state for each database and also for the totality of databases.

Processing in one computer system may proceed concurrently with processing in another computer system without affecting the data integrity at each computer system.

4.10.3 Communications

It may be necessary to communicate with an information system from a location remote from that of the information system itself. In addition, an information system located at one computer system may need to be able to communicate with an information system located at a different site

Such communication requirements can lead to failures in a distributed information system such as

a) messages may be lost in transmission;

b) messages may not arrive in the order sent due to transmission errors and consequent retransmission;

c) in certain circumstances, failure of a connection is indistinguishable from failure at the remote site.

In order to prevent the loss of database integrity due to such failures, it is necessary to know the degree to which the data is replicated.

4.10.4 Export/import

It is required to be able to take a copy of some or all of a database, with or without its data definition, and to insert it into the same or a different (possibly remote) database environment. The data is said to be exported from one environment and imported into the other. Once exported, the data can be imported into as many other environments as required, and may also be retained for archival purposes.

4.10.5 Distribution independence

Application processes should be independent of the distribution of the data. There are several kinds of distribution independence. In local independence, the application process must be aware of different fragments existing in the distributed database, but not of their location. With fragmentation transparency, the application process may not need to be aware of the existence of fragments.

4.10.6 System autonomy

In some distributed information systems there is a requirement for the computer system storing the distributed data to be autonomous. An autonomous computer system needs to be capable of functioning independently of other computer systems. Such a requirement relates to the need for performance, the availability of data during communication failure and administrative concerns such as accounting and authentication of users.

A requirement for autonomous computer systems in a distributed information system places significant demands on the management of data for such an information system.

4.10.7 Recovery of a distributed database

Modified data that is distributed in more than one database should be recovered in such a way that the end result must be a consistent state and the state across databases must be consistent.

4.11 Dictionary systems

There are a number of requirements for data management which relate to data about information systems. This data is managed by a particular type of information system called a **dictionary system**.

In order to distinguish the particular purpose of a dictionary system from other information systems, the latter are referred to as application systems. A dictionary system is an information system containing data about one or more application systems.

Some detailed requirements of particular concern to dictionary systems are support for life cycles, version control, configuration management, auditing and performance engineering.

5 Concepts for data level pairs and related processes

5.1 Purpose

The purpose of this clause is to explain the data constructs that are required for this Technical Report, how they relate to one another, and how they relate to the process constructs that are required for this Technical Report.

5.2 Level pairs

The **level pair** construct is a means of explaining the relationships between database and schema. The graphic notation of Figure 2 is employed to illustrate the essential coupling between a database and its definition.

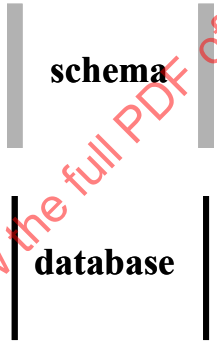


Figure 2 — Level pair construct

The impact of the level pair construct is that each database conforms to the data structures specified in its associated schema. The data values in a database can be processed only by data manipulation processes which are bound to the schema of the database. The schema determines the exact form of processing permitted. The level pair construct therefore illustrates a means for achieving consistent operations among many data manipulation processes.

The representation and interpretation of the data values depend on the schema. There can be no processing until the schema has been defined and activated. When changes to the schema are made, then any associated database has to be changed accordingly to maintain consistency.

5.2.1 Interlocking level pairs

A particular schema not only defines data but is itself a collection of data which must be created and protected and may be modified. Data management techniques are therefore appropriate for the management of schemas. Specifically, the schema in a level pair can be represented in a higher-level database, the data structure of which may be defined by a higher-level schema. This database and schema constitute another, higher-level level pair. The two level pairs are said to be “interlocking”, as shown in Figure 3.

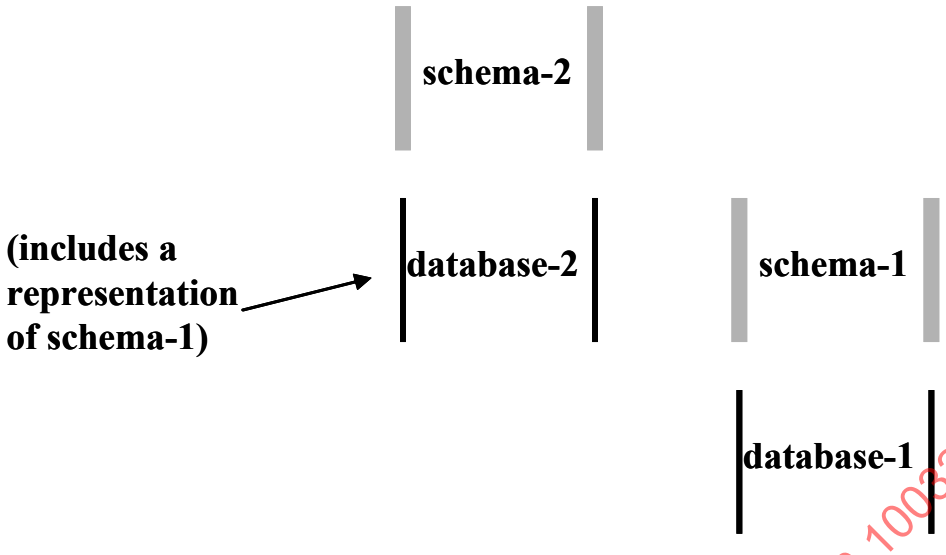


Figure 3 — Interlocking level pair

Figure 3 shows that database-1 conforms to schema-1. The data in database-1 may be processed by data manipulation processes which are bound to schema-1. Figure 3 also shows that database-2 conforms to schema-2 and that the data in database-2 may be processed by data manipulation processes which are bound to schema-2. The representation of schema-1 in database-2 is called a **source schema**. The source schema can be selected from the database or otherwise processed by data manipulation statements in the same way as any data in a database.

The two level pairs are at different levels of data definition. Since it is also possible for schema-2 in Figure 3 to have a representation in the form of data instances recorded in a database, it can be seen that the concept of interlocking level pairs is a recursive concept and can be used with two, three, or more level pairs. The recursion stops when the data definition cannot be modified in any way.

5.2.2 Recursive use of level pairs

With a potential multiplicity of level pairs, a mechanism for referencing each level is required. General labels of (N), (N+1), etc. are used in this Technical Report to indicate increasingly higher levels when common properties are considered.

The interlocking of level pairs takes place by associating the schema of a level pair (N) with the database of the next level pair (N+1). The former is called an (N) schema, and the latter an (N+1) database. Figure 4 applies this labelling to Figure 3.

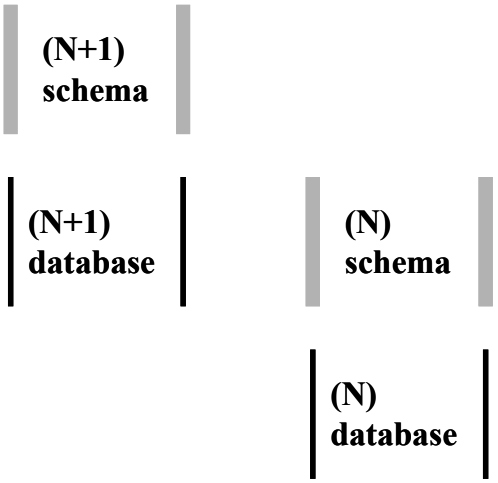


Figure 4 — Generalized interlocking level pairs

5.2.3 Operations on level pairs

The implementation of a database involves processes of creating and maintaining data definitions, making those definitions available to the data manipulation processes and then executing operations to select and modify data in a database.

The above processes are related to Figure 4 as follows:

a) The (N) database represents the data which is actually intended for manipulation at level (N).

b) The (N) schema represents the schema which has been made available to control the processes for level pair (N). This schema contains data definitions for the (N) database only. A process is said to be bound to the schema if the process conforms to the schema of the level pair.

c) The (N+1) database contains data definitions that have been created during the design process for database (N) and maintained during system operation. Such a collection of data definitions is called a source schema. An (N+1) database may also contain data other than data definitions, such as descriptions of those data definitions and the designs and specifications of processes which use them.

d) An (N+1) database may contain representations of one or more (N) schemas in source form. After one of these (N) source schemas has been selected, an **activate** process may be used to convert the (N) source schema into a form (called an **object schema**), such that an associated (N) database may be populated. An (N) source schema may be activated more than once and each activation produces a separate (N) object schema with an associated (N) database which may be populated using data manipulation processes.

The interlocking has a lowest level pair for which the data on the lower level of this pair does not contain data about a schema and its components and hence cannot be activated. This level pair is then part of an application system and the data in the lower level of this level pair is an application database.

The interlocking also has a highest level pair for which the schema on the higher level of the level pair is not recorded in a higher level database. This schema is then implicit in the Data Modelling Facility used by the data management system.

5.3 Dependence of level pairs on a Data Modelling Facility

The level pair construct and the concept of a Data Modelling Facility are closely inter-related. A Data Modelling Facility comprises a set of data structuring rules and an associated set of data manipulation rules.

5.3.1 Level pairs and data structuring rules

A Data Modelling Facility includes a set of data structuring rules which are to be used for defining a schema. These rules include rules for the specification of constraints which may be part of a schema. Each schema must be complete and consistent according to the data structuring rules of the associated Data Modelling Facility.

In Figure 4 the (N+1) schema constrains the data that can populate an (N+1) database. As a result of this, the (N+1) schema has an impact on each (N) source schema which is contained in an (N+1) database.

5.3.2 Level pairs and data manipulation rules

A Data Modelling Facility also includes the rules for the semantics of data manipulation processes. For an (N) schema, the constraints which are based on the data structuring rules also have an impact on the semantics of updating data manipulation processes performed on the database.

5.4 Level pairs and associated processes

The data in a database may be retrieved or modified by a series of data manipulation processes. In addition, if part of this data comprises a source schema, it is possible to perform an activate process on the data.

Figures 5 and 6 illustrate how the data structuring rules used to define an (N+1) schema affect the data manipulation processes used to manipulate data in an (N) database.

The following five data manipulation processes are used: (1) select, (2) activate, (3) bind process to data structuring and manipulation rules, (4) bind process to data definitions and (5) select, add, modify or delete.

The numbers 1 through 5 refer to steps that are explained following the figures. Note that Figures 5 and 6 are an expansion of Figure 4 to show activation, schema linkages, intermediate collections of data, and the preparation and operation of other data manipulation processes.

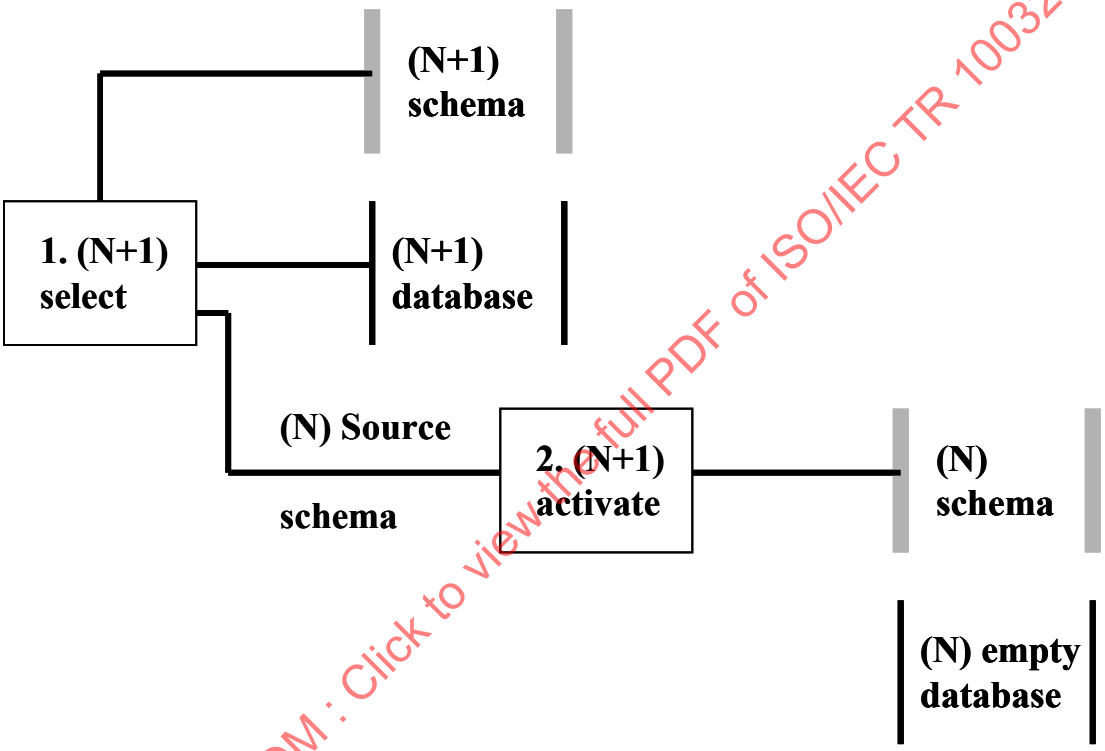


Figure 5 — Creation of an empty database

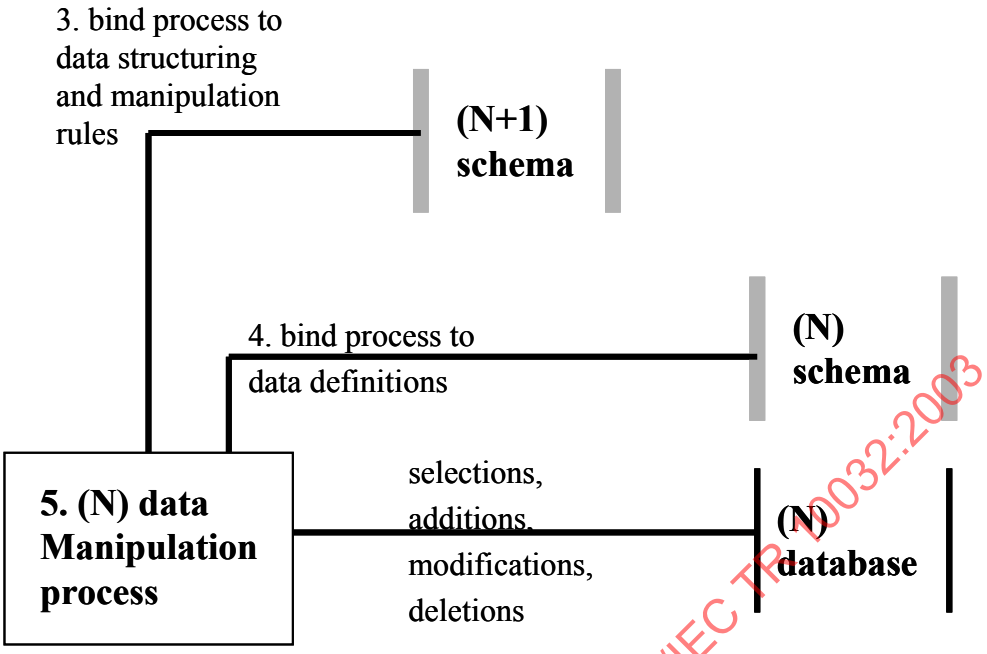


Figure 6 — Binding and data manipulation

The following steps describe corresponding processes in Figures 5 and 6. They show how the rules of the Data Modelling Facility enable an (N) data manipulation process to operate correctly on an (N) database:

1. A data manipulation process selects an (N) source schema from the (N+1) database, using the rules and data structures of the (N+1) schema. The representation of the (N) source schema in the (N+1) database is persistent data that can be modified. The (N) source schema after selection may or may not have a representation form of persistent data.

   For example, the select operation may simply set a flag in the (N+1) database, or it may fetch data from the (N+1) database and store it in a different database from that in which the (N+1) database is stored. This other database must also conform to the (N+1) schema.

2. The source schema is activated to create an (N) object schema and an empty (N) database. Analysis to ensure that the source schema is a valid schema must be performed prior to activation. Such analysis may be performed in whole or in part by the (N+1) data manipulation process performing the select, by another process that analyses the source schema, or in conjunction with the activate.

   Activation may result in changing the setting of a flag associated with the previously selected (N) schema in the (N+1) database. Alternatively, activation could result in physical movement of (N+1) data and in a change in the form of representation. In either case, the activated schema must be protected from any change that would result in the database no longer conforming to the schema.

3. The data structuring and manipulation rules which govern the operation of the (N) data manipulation process accessing the (N) database are the same as or based on those associated with the (N+1) schema.

4. Each (N) data manipulation process which is to access the (N) database is bound to the Data Modelling Facility rules data definitions in the (N+1) schema. When the (N+1) level is the top level, this binding is implicit.

5. In order for a data manipulation process to be able to access an (N) database, the (N) schema must be active.

Steps 1 and 2 provide for the creation of an empty database. Step 3 makes explicit the relationship between the Data Modelling Facility defined at level (N+1) and the (N) data manipulation process. This step also makes explicit an important attribute of interlocking level pairs. Both level pairs are required in order for the data management processes to be able to operate correctly.

The binding may be implemented in a variety of ways (e.g., by reference to the schema, or by incorporating the schema into the process) and at a variety of times (e.g., during execution or during program compilation). The choice made may affect execution time, storage space, and the ease of maintaining the consistency of the processes and the schema as the latter is modified. The choices should not otherwise affect the results of the processes.

5.5 Access control for level pairs

All data and the processes on that data are subject to access control. Access control is equally applicable to any level pair.

5.6 Schema modification

There is a requirement to be able to change the structure of a database, which means that the associated schema must be modified. After the schema has been modified, it is necessary for the data from the original database to be correctly represented according to the new schema.

This can be accomplished by modifying the source schema in the (N+1) database, activating it to produce a new (N) schema and an empty database as shown in Figure 5 and transferring the original data into the new database. Incremental modification can also be used to satisfy this requirement, with the same effect. The mechanisms for the transfer of data from the original version of the database to the new version of the database are outside the scope of this Technical Report.

6 Architectural model

6.1 Purpose

The purpose of this clause is to provide an architectural model within which different data and processing components relevant to data management can be placed. This architectural model takes into account the requirements for data management outlined in Clause 4 and the data structuring and processing concepts described in Clause 5. It begins with a generic model which identifies the processing common to all data management and then elaborates the model in order to accommodate the particular requirements (identified in Clause 4) for a database, a distributed database environment, for export and import of database data and for access control facilities.

The principles used to identify potential data management standards within this architecture are described in Clause 7, and the relationships to and prescription of existing and future data management standards are given in Annex B.

6.2 Modelling concepts

For the purposes of this Technical Report, a processor is defined as a modelling concept that represents some combination of hardware and software that can provide services either to one or more processors or to a human user.

The architectural model is presented in terms of various processor classes, where each processor class focuses on particular aspects of data management. It is not intended that such a model should represent an implementation architecture for engineering computer systems.

6.2.1 Characteristics of Reference Model processors

a) A model is expressed in terms of processors which interact as client and/or server. These terms are used to refer to processors which for a specific interaction have these roles. A particular processor may have a different role in another interaction.

b) In a client-server interaction, a client makes a request for a service where the request may include the specification of any data values needed to invoke that service. The server provides a response, such as one or more of the following:

   1) an indication that the requested service has been completed;

   2) a collection of data that resulted from the requested service;

   3) a message that the service is not available (either because the requester has no authority to request the service or because it is not provided by the processor);

   4) a message that the requested data is not available;

   5) a message that the request is not correctly formulated.

c) Each processor is defined by the external interface it presents as a server, which specifies the services it is capable of providing and the type of data to which the services apply. For any processor, the internal representation of data, the way it performs its processing and any use of services provided by other servers are not relevant to the definition of its interface. However, the interaction of a processor with other servers is relevant for modeling purposes.

d) Each processor is an instance of some class. The class determines the services common to all processors which are instances of the class. Some classes define the type of data to which their services apply and some classes require a separate schema to define the data to which their services apply.

e) A processor may be a client of many servers at any time; some servers may support several concurrent clients.

f) Processors are used as building blocks to provide data management services. The use of some processors may be recursive.

6.2.2 Levels of abstraction

The Reference Model processors are only concerned with necessary details relevant to data management. Details of how any processor is implemented or how it relates to other processors in a computer system are not relevant. In particular, the interactions identified in the architectural model generally allow a client processor and server processor to be either within one computer system, in which case their interaction is supported by operating system facilities, or in different computer systems. Their interaction is supported by operating system facilities or by communications facilities.

When more detail is required for some aspects of a model, the following forms of refinement are used:

a) specialization, where a subclass of a generic processor class is defined as having a distinct name and services which are additional to or modified forms of the services of the generic processor;

b) decomposition, where the services of a processor class are shown to be provided by two or more processor classes with interactions between them.

6.2.3 Notation for processors

In the model diagrams and associated descriptions, it is necessary to identify whether a class or an instance of a processor is being referred to; a notation to distinguish these two concepts is that a class is referred to by a name with an initial upper case letter whereas an instance has only lower case letters. For example, one

processor of the architectural model is the class User Processor, which has instances which are referred to as user processors.

Diagrams are used to show the processor classes involved in a particular type of processing, together with the client-server interactions between classes. The general diagram notation for a processor class is shown in Figure 7, where the shape encloses the name of the class and the shaded part of the boundary represents the interface. The rest of the text in this diagram explains the use of this construct.

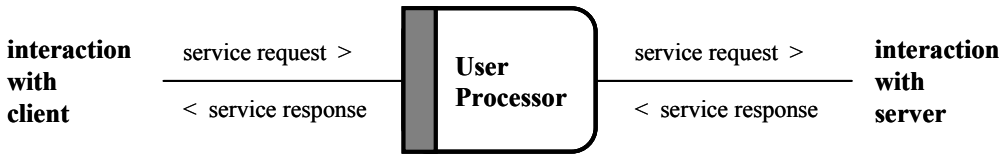


Figure 7 — Example processor diagram

For any processor there are two forms of interaction with other processors; first, as a server, in which case there is a connecting line representing service requests from some client to the interface and, second, as a client of the services of another server, in which case there is a connecting line from a point on the boundary (not the interface) to the interface of the other server. Each line represents a processing linkage which can support a sequence of one or more interactions between instances of the client and server processors.

The architectural model includes a number of diagrams constructed using this notation, which are explained as they are presented. In each case, it is necessary to determine how instances of the processors may interact. In some situations there may be constraints on the allowed interactions between processor instances and in other situations there may be freedoms which may not be obvious from a class diagram. In such cases, an example set of interacting instances is given to provide an interpretation of the class diagram.

6.3 The generic model of data management

The generic model represents the characteristics of the processor classes common to all data management, relating to the general context of a class of users concerned with the definition of and access to data stored in databases. Figure 6 shows the components of the generic model.

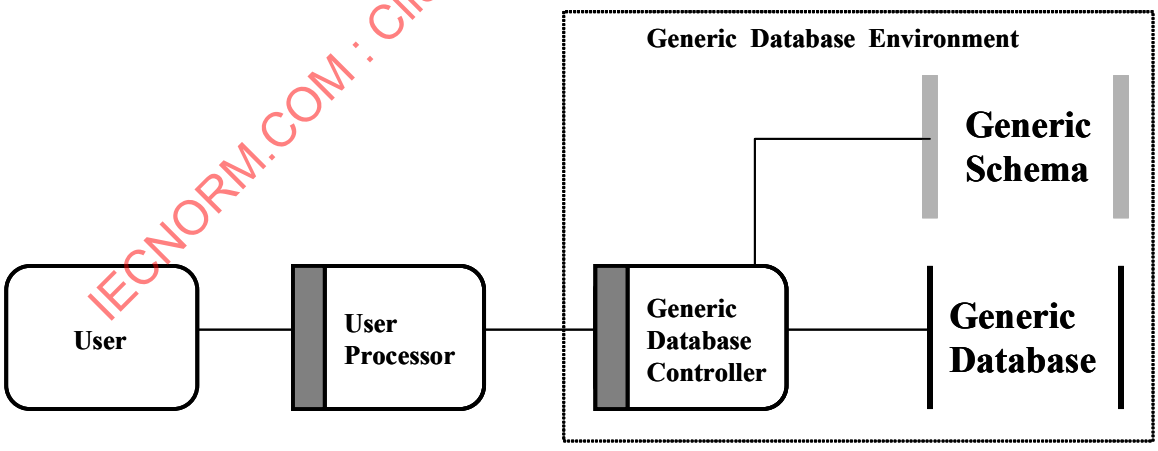


Figure 8 — The generic model of data management

This model is based on the characteristics of a Generic Database class and a Generic Schema class as described in Clause 5. Each processor class and its instances have the following data management characteristics, where the use of “generic” should be assumed for the data management classes.

6.3.1 Generic Database Controller

Generic Database Controller is the name of a class of general-purpose processors that provide data management services for definition of and access to a class of databases; such processors require access to a schema and their services include those related to an associated class of schemas.

A generic database controller is bound to one schema and its associated database, which together form a generic database environment. Each database environment should have a unique identifier.

The data which is accessed by any one database controller in any Database Controller class must be structured and manipulated according to the rules of one defining data modelling facility.

The data which is accessed by any Database Controller may include the rules of one or more defined data modelling facilities. These rules must be represented according to the rules of the defining data modelling facility on which the Database Controller is based.

Typical services provided by a database controller include the following:

a) establish a data management session for a client processor, requiring explicit or implicit binding to a named database environment (requests for the following services are then allowed);

b) add to and modify data definitions in the schema for a database;

c) retrieve data definitions from the schema for a database;

d) add to, modify or delete data in a database;

e) retrieve data from a database;

f) start a database transaction of one or more requests for service;

g) end a database transaction by committing or rolling back;

h) establish backup procedures for a database;

i) initiate recovery procedures for a database;

j) reorganize a database;

k) close a session.

Requests for these services are expressed to a database controller by

a) statements in a database language;

b) messages to a data management services interface;

c) calls to procedures.

6.3.2 User Processor

User Processor is the name of the class of processors that are clients for data management services provided directly or indirectly by database controllers. Each user processor provides a set of services many of which, when requested, make use of data management services. A user processor's services may be for purposes outside the scope of this Technical Report.

The services provided at the interface to a User Processor may make use of a data modelling facility which is different from the one used by the Generic Database Controller. In this case, a definition of the rules of the data modelling facility used at the interface to the User Processor must have been made using the data modelling facility of the Generic Database Controller.

In providing its services, a user processor is a client of the services of one or more database controllers; use of the services of any one database controller requires that a data management session is established directly or indirectly between the user processor and the database controller.

6.3.3 User

User is the name of the class of persons or processors that are clients of the services provided by user processors.

6.4 Specialization of the model in different environments

The generic model given in 6.3 can be applied to the different kinds of information systems identified in Clause 4. At the most abstract level, this can be simply represented by replacing the term "Generic" by an appropriate qualifier. For example the model can be applied to a distributed database environment in terms of a Distributed Database Controller, Distributed Database and Distributed Schema.

These specializations of the model are examined in more detail in the following clauses, by decomposing the abstract classes and by considering the possible instances to show component classes.

6.5 Database environment

In this application of the generic model, a database controller for such a database environment would support the typical services identified in 6.3.1.

Subclause 6.2 described how full interpretation requires understanding of the possible instances of each class. The following examples are given to demonstrate two different arrangements of instances that are possible for a database environment, and the consequences for the use of the services of a database controller.

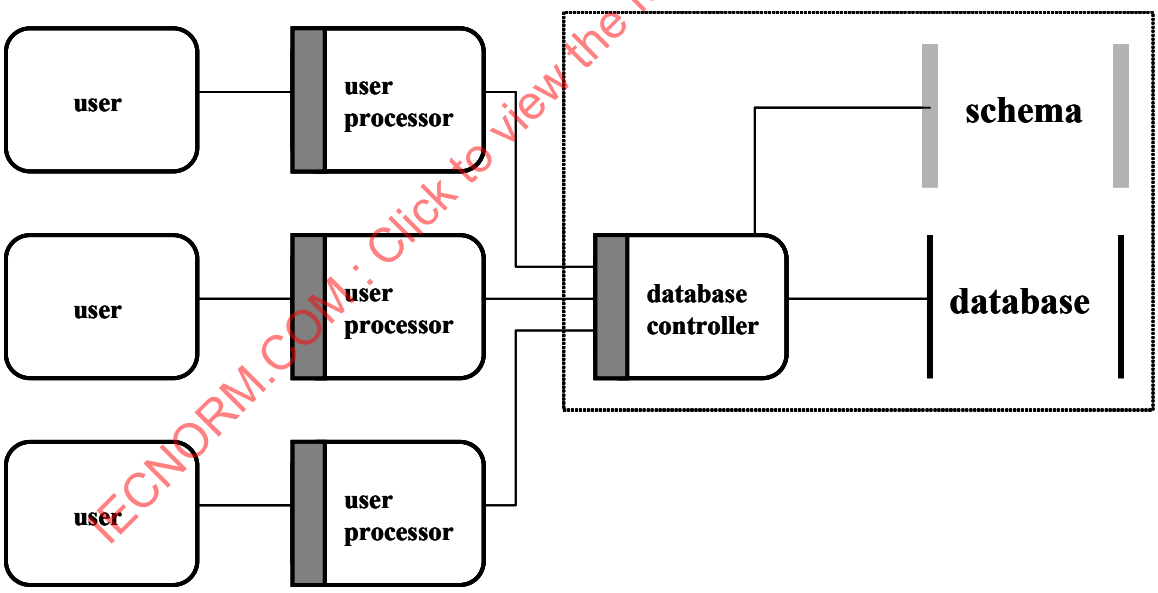


Figure 9 — Example of access to a database environment

The essential feature of the example shown in Figure 9 is that many user processors are using data management services within a database environment. For simplicity just one user is shown for each user processor. This is not intended to be a restriction.

This example illustrates that database controller services must be supported for many client user processors concurrently, yet a client must be able to run database transactions without interference from other clients.

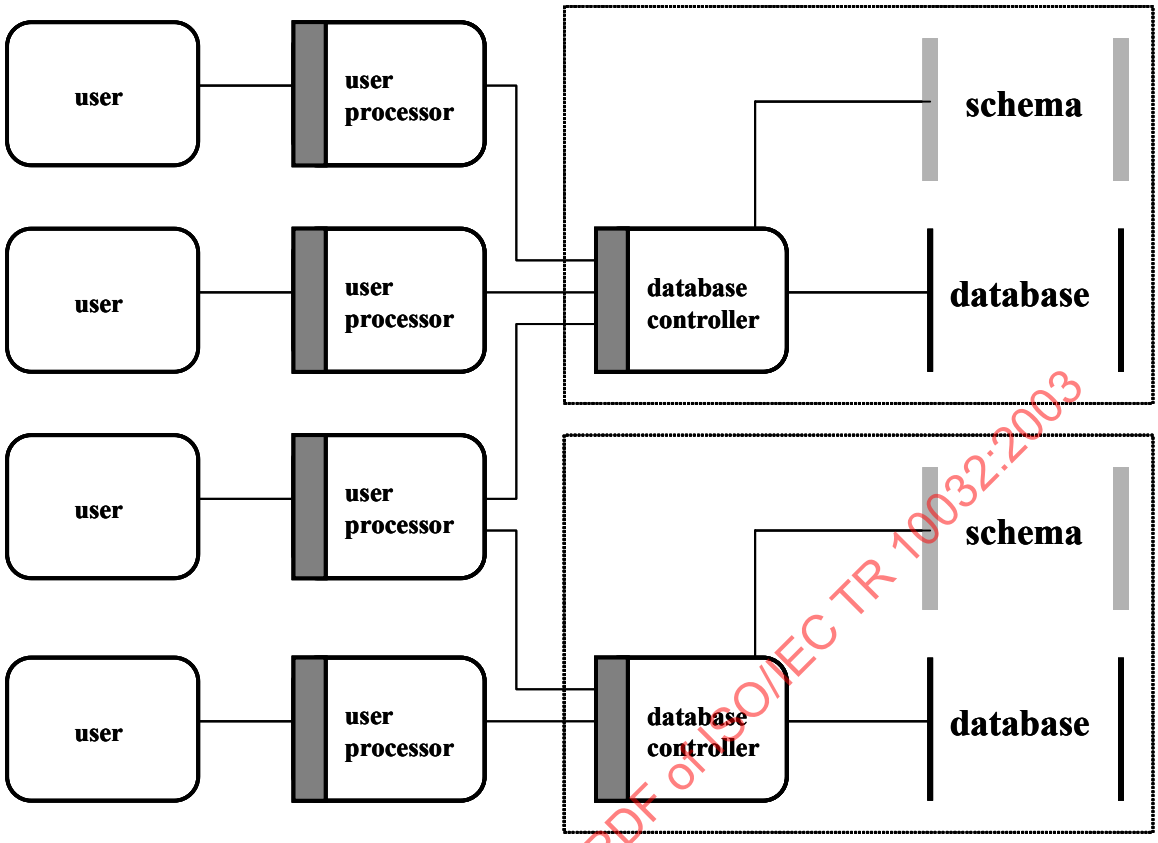


Figure 10 — Example of access to many database environments

The essential difference in the example given in Figure 10 from that in Figure 9 is to show that some user processors are capable of accessing more than one database environment.

In this case, a user processor that accesses more than one database environment must include the capability for directing a request for service to the database environment which contains the data to be accessed. Thus any request for a service must include an identifier of the database environment for which it is intended; the way in which such an identifier is associated with the database environment may be outside the scope of this Technical Report. In addition, the services of a database controller can be provided only within its own database environment. Any database transactions or relationships between data which involve more than one database environment must be supported by some processor other than the database controller.

A database controller may be able to use a services interface to access data which is not under its own control.

This example shows that the services of a database controller do not support the management of data in multiple database environments, though user processors may do so in a specific context.

The above descriptions did not consider the relationship between a database environment and any particular computer system. In Figure 10, the two database environments may be in the same or different computer systems. Also, both examples allow for a user processor to be in a different computer system from a database environment in which case communication protocols are required for the processing linkage. These protocols may be either passive carriers of the interaction or provide special services which support the remote use of data management services.

6.6 Distributed data management

An elaboration of the generic model of data management can be used to address the requirements of a distributed information system.

As indicated previously, distributed data management must handle several different permutations of the categorization schemes presented. One basic requirement common to all is that the services which are provided by a user processor may relate to data stored in one or more different database environments. It is also possible that a service requested by a user processor is available only at some computer system other than the one at which the request originates.

All services of the database controller for the generic model are applicable in a distributed context. In addition, the services related to distribution must be provided, possibly at a different processor interface from that of the database controller.

The specification of how a database is distributed is a typical service provided by a Distribution Controller. A Distribution Controller determines whether a request for service involves access to another computer system. The Distribution Controller must also determine what services are required to satisfy a request.

For each database environment, there exists a collection of persistent data which is associated with and used by a database controller when processing the requests for service which it receives. This collection of data is called the local distribution data.

The manner in which data is distributed is specified in a database for distribution data. Similarly, the distribution data for a distributed database may itself be distributed in any of the alternative ways.

Figure 11 shows the architectural model of distributed data management. This is based on a decomposition that uses database controllers for localized data management and distribution controllers for the distributed aspects of distributed data management.

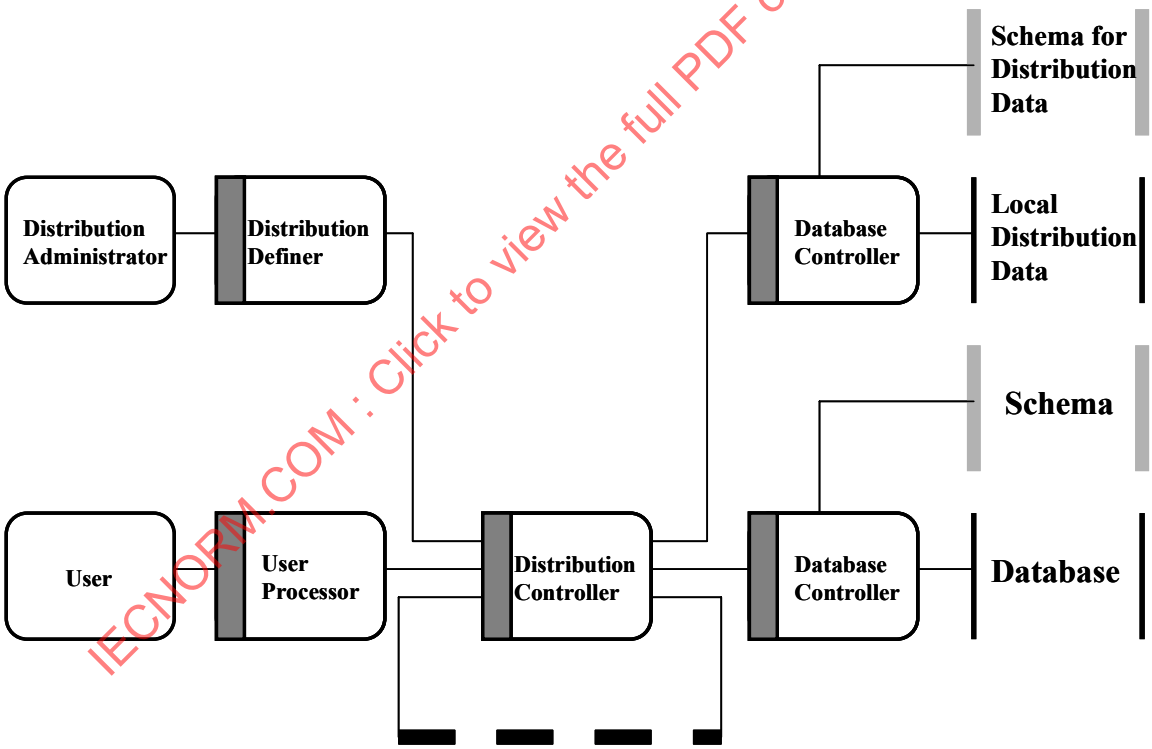


Figure 11 — Distributed data management

In this application of the model, the User, User Processor and Database Controllers are as previously described. The Distribution Administrator is a particular user class concerned with the task of specifying distribution data and the Distribution Definer is a specialized User Processor providing services for distribution administration.

6.6.1 Distribution Controller

A class of processors that provide services for definition of and for the access to distributed databases has the name Distribution Controller. A Distribution Controller must have the capability of accessing the relevant parts of the schema and parts of the distribution data directly or indirectly associated with a distributed database, as well as the distribution data, whether those parts are in the local database environment or the remote database environment.

The distribution data which is used by a Distribution Controller must be accessible according to the same defining Data Modelling Facility.

For any distributed database, it is assumed that each computer system has a distribution controller. Each distribution controller provides access to those parts of a distributed database held in any database environment in the same computer system. In addition, a distribution controller enables access to those parts of a distributed database in remote database environments via communication (shown in Figure 11 by the dashed line) with other distribution controllers in different computer systems.

A particular constraint in Figure 11 is that a distribution definer is the only user processor to require services for access to distribution data. Other kinds of user processors do not require any services which explicitly require access to distribution data, making use of the same database controller services as defined for the generic model. However, a distribution controller provides additional services to other distribution controllers in support of the distributed processing of data.

On receiving an initial request for a service from a user processor, a distribution controller first determines the database environments specified for the referenced data, using the services of a database controller to access the local distribution data in the same computer system. If the required specification is not available as part of the local distribution data, a distribution controller communicates with another distribution controller in another computer system to find the required distribution data.

Once the database environments for the referenced data have been determined, a distribution controller can then invoke the services of any database controllers in the same computer system or communicate with remote distribution controllers to effect part or all of the requested service. The way in which this is performed is determined by a distribution controller in order to achieve optimum use of processing and communication resources.

6.6.2 Role of Distribution Controller and level pairs

A request from a user may relate to any level pair, although the user initiating the request should not need to be aware of the specific level pair involved. When such a request is received by a Distribution Controller, the Distribution Controller has to determine in some way which level pair is involved.

The Distribution Controller then accesses the local distribution data to determine in which database environment the required data is located and also whether the service requested can be performed in that database environment.

6.7 Export/Import model

The export/import model is a specialization of the generic model to exhibit the services of a database controller to include the capability of export and import as described in 4.10.4.

Use of an export service requires first a specification of the data which is to be exported from a database environment; this may include either selected occurrences of a particular type of data, a number of types of data or such data together with corresponding schema definitions. Second, a file has to be named and a type of file format appropriate for the specified data must be chosen.

Use of an import service requires the name of a file into which data has been exported and the file format which has been used.

The architectural model of export and import is based on a decomposition that separates the database controllers which provide the services for any of the types of database environment, as described in 6.4, from the Export/Import Processor which provides the required services. Figure 12 shows the use of export and import service by a user processor. In a distributed environment, the services of the distribution controller could be used.

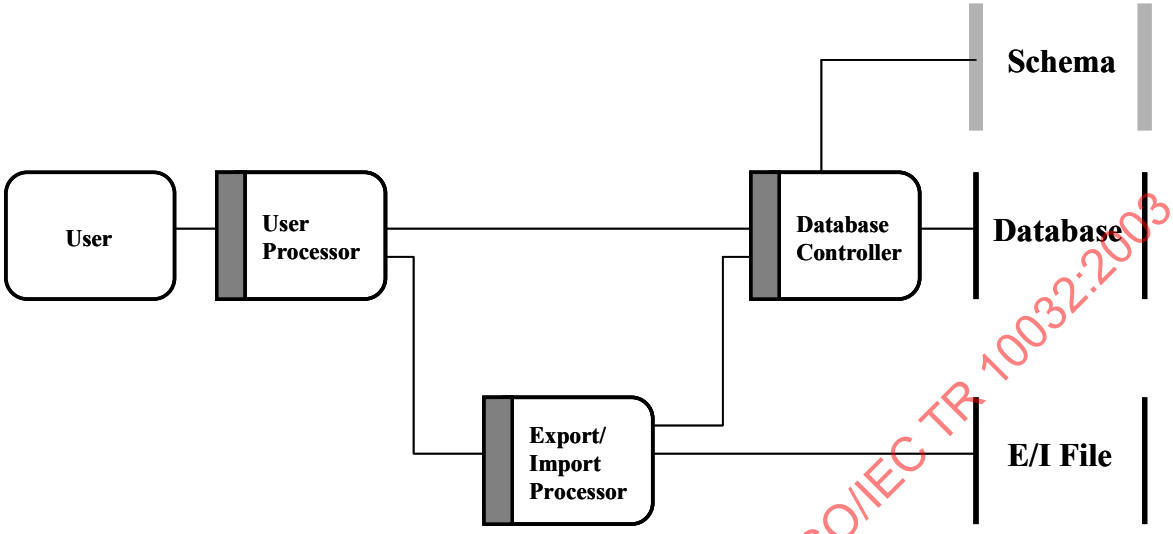


Figure 12 — The model of export/import

6.8 Access Control for Data Management

The capability to control all forms of access control may be required by data management services. Access control may be provided by any data management processor and may include services required for the definition of access control data. Access control may be applicable to a number of processors.

Access control requires that any service provided by any processor may require special authorization. Each request for a service must be associated explicitly or implicitly with an authenticated identifier for the user. This identifier may be implicitly associated with a service request by being declared at the start of a session which involves the use of data management services.

A processor which checks each authorization to use the services provided by a processor is called an Access Control Processor. There are several ways in which an Access Control Processor may relate to another processor, including

a) the Access Control Processor is used by the controlled processor as a server;

b) the Access Control Processor requests the services provided by the controlled processor;

c) the Access Control Processor may be embedded in the controlled processor.

Independently of the approach used, whether a particular request for a service is allowed depends on whether the associated identifier has pre-declared privileges for the processes and data involved. If all appropriate privileges for a request for a service do not exist, the response to the request indicates an access control violation, and this violation may be recorded for general management of access control.

To support access control, the way in which the services are provided by a processor has to take into account the need for appropriate privileges. For most services there is no change in the form of the request for a service but the possible responses may have to allow for rejection because of lack of privileges.

Additional services are required for the definition of the valid identifiers for a database environment and the privileges associated with those identifiers.

One approach to access control is illustrated in Figure 13.

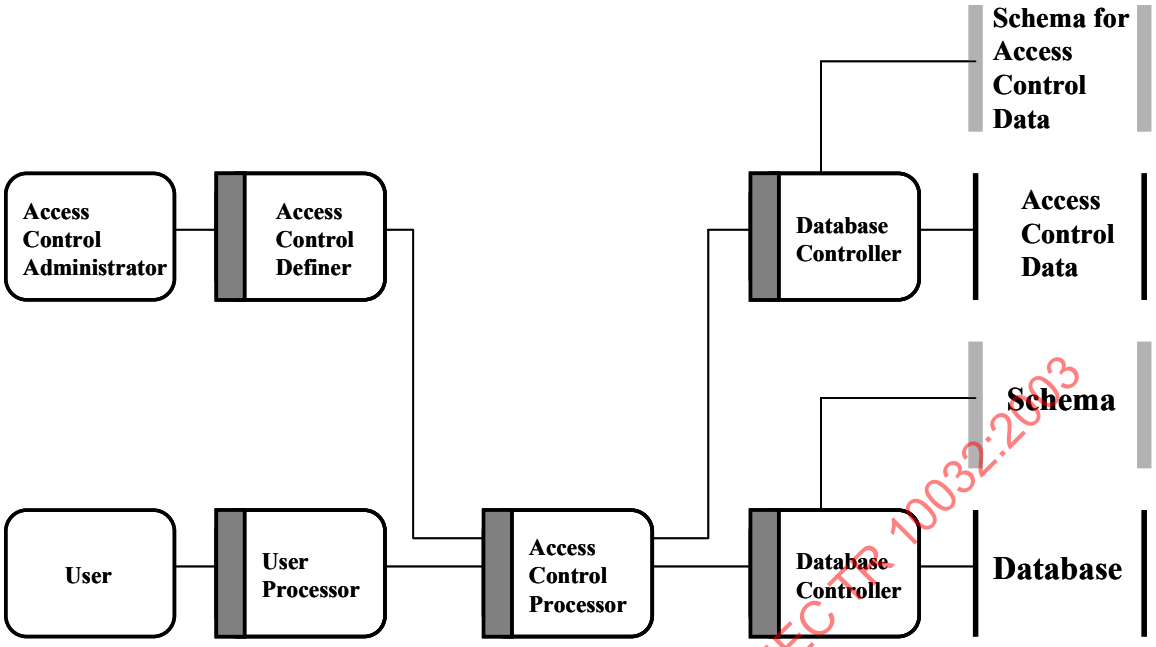


Figure 13 — Access control in a distributed environment

The Access Control Administrator is a particular User Class concerned with the task of specifying access control data and the Access Control Definer is a specialized User Processor for providing services for access control administrators.

As shown in Figure 13, the use of a database controller may result in the implicit invocation of associated access control functions in order to verify that the user processor is authorised to perform the required action. Such access control functions will require the retrieval of access control data relating to the particular operation being requested. This is shown in Figure 13 as being explicitly stored as access control data.

Figure 13 illustrates access control functionality as it applies to the database controller. However, access control may be relevant for other processors, such as the Access Control Definer itself. The Access Control Definer is the mechanism for establishing the access control data which is relevant in order to validate that appropriate privileges exist for the operations associated with all such processors.

7 Objectives and principles for data management standardization

7.1 Purpose

This clause identifies the objectives of standardization which are to be used in conjunction with the Reference Model of Data Management when specifying standards.

This clause describes standardization from three complementary points of view:

— technical objectives which are based on the concepts presented in this Technical Report;

— the means by which these technical objectives may be achieved;

— aspects of data management standards.

7.2 Technical objectives associated with data management standardization

The following eight technical objectives are identified for data management standardization. Each of these is associated with one or more of the three major objectives discussed in the Introduction. These objectives are expressed in terms of the concepts introduced in the earlier clauses of this Technical Report:

a) support for all distributed scenarios;

b) location independence;

c) standardized database transaction management;

d) export and import of databases;

e) reduced complexity of handling data;

f) overall performance in distributed scenarios;

g) data independence;

h) application portability.

i) extensible Data Modelling Facility;

j) flexible presentation of data to users.

The last two objectives are noted as being important for data management standardization. These differ from the first eight in that they allow future evolution of the technology and no further means of achieving them can be identified here.

7.2.1 Support for all distributed scenarios

When designing a distributed information system, it should not be necessary for each computer system and for each data management system used to be supplied by the same vendor.

Such a system is referred to as heterogeneous and a database accessed and updated by different kinds of data management systems is referred to as a heterogeneous distributed database.

Various types of interprocess communication require differing underlying communication facilities, which may be provided by operating system facilities, OSI Communications services or other communications services.

Standards to support distributed databases should be based on selected “services interfaces” provided in each data management environment. The processors associated with the interfaces should react in a standard predefined manner when a particular request for service is received, and should respond in a standard manner.

In addition to interworking processors in the same management domain, it is also necessary for a data management system in one management domain to be able to access data managed by another data management system in another management domain, subject to the access control requirements in the server management domain being met. The processors in the two management domains may be provided by the same or different implementors.

7.2.2 Location independence

To a user of data management services through a User Processor, it should not make any difference whether data to be accessed is stored locally or remotely.

In order to provide distributed system communication mechanisms which permit such location independence, data management services and the interfaces at which these services are available, must be designed so that a user is not required to know where the data is stored.

The User Processor should not require awareness of the location of the data being processed. The only exception to this would be a User Processor which is used during system set-up to establish the locations of any database environment with which a system may interchange data.

Remote access, while requiring additional information in order to determine the location of data stored remotely, should not require a fundamentally different approach to the data management services being provided. Requirements for remote as well as for local access to data should be equally applicable to all level pairs.

7.2.3 Standardized database transaction management

The database transaction concept is the appropriate mechanism for addressing the issues of integrity, recovery, and concurrency control within a data management system. Accepted database transaction properties (ACID - Atomic, Consistent, Isolation from the results of other database transactions, and Durability) can be specified by the use of concepts such as serializability in a local data management environment.

Heterogeneous distributed database interworking requires that an appropriate mechanism for database transaction management be selected and standardized.

7.2.4 Export and import of databases

In order to share all or portions of a database, including not only application level data but also data on any level pair, there exists a requirement to support the transfer of data between data management systems.

The possibility of differing data management system implementations indicates a need for a standard representation form for the data transfer if such data is to be understandable to more than one data management system. Export/import facilities are particularly important in permitting the sharing of analysis and design information between dictionary systems and computer aided systems engineering tools.

There also exists a requirement for a standard representation form for archived databases. The same format could be used.

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.

The selection processes required to indicate the portion of a database of interest for producing an export/import file do not differ significantly from the semantics associated with normal retrieval statements.

This leaves consideration of the representation form for the schema level data associated with the lower-level data as a potential part of an export/import file. 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 nothing to prevent the use of the same approach for all levels of data for which the creation of an export/import file may be appropriate.

The Reference Model approach is 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 data manipulation language retrieval semantics in order to account for the specific selection requirements for export/import purposes, such as ensuring that schema level data accompanies application level data.

Taking a “services interface” approach, the response to such a request will mean that the result of such retrieval operations may be a message (which may be lengthy). The response can then simply be used for a different purpose (that is, storage or communication as a file rather than presentation to a user).

7.2.5 Reduced complexity of handling data

Much of the complexity of existing distributed data management systems may be directly attributed to the difficulties encountered in mapping between differing data modelling facilities.

The strategy used to overcome such difficulties is to utilize a standard approach to avoiding disparities between the Data Modelling Facilities of different data management systems in a heterogeneous environment.

The approach advocated by this Technical Report eliminates the need to take into consideration potentially differing data modelling facilities used by local data management systems.

While translation between data modelling facilities may be of concern for vendors wishing to interface with data management systems not conforming to the selected "standard" data modelling facility, it is not necessary that such data model translations be addressed in data management standards.

7.2.6 Overall performance in distributed scenarios

In each of the distributed scenarios, especially the distributed database and the federated database scenarios, it is an objective of standardization work to enable improved performance.

7.2.7 Data independence

Data independence (see 4.5) is important in all information systems in order to reduce the costs involved in modifying and enhancing the system.

7.2.8 Application portability

It is an objective of data management standardization to enable improved portability of application systems from one computer system to another.

7.2.9 Extensible Data Modelling Facility

Any data modelling facility must be capable of being extended as new requirements are perceived and as new technology becomes available.

7.2.10 Flexible presentation of data to users

One processor of any information system which must be considered is the User Processor which is responsible for coordinating the interaction with the user.

There may be a need for the data structures visible to human users to be different from those representable using the interchange data modelling facility.

Such user interfaces should 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.

As such, standardization of the interfaces at various types of user processors is normally outside of the scope of data management standardization.

7.3 Means of achieving objectives

The first eight data management standardization objectives can be achieved by the following means:

a) same data modelling facility for each level pair;

b) same interchange mechanism for all level pairs;

c) same processors usable for all level pairs;

d) standardized services at Database Controller interface;

e) standardized approach to access control;

f) standardized representation of data needed to facilitate interoperability;

g) support data fragmentation;

h) separation of logical and physical structures;

i) access to schema during execution.

These means are described in the terms established in the earlier clauses of this Technical Report.

7.3.1 Same data modelling facility for each level pair

A data management system should utilize the same Data Modelling Facility for all level pairs prescribed by the standards, and should provide the same data manipulation processes on schema level data as on the database controlled by a schema.

The data structuring rules of one data modelling facility may be specified using another data modelling facility. In this case it is important to distinguish between the defining data modelling facility and the defined data modelling facility. It is possible for the data structuring rules of a data modelling facility to be defined and manipulated in accordance with the rules of the same data modelling facility.

To express how other data modelling facilities can be positioned, it is necessary to identify specific level pairs. At earlier stages in the information systems life cycle, it is common practice to develop an application data model using a data modelling facility which is different from that used at the application level pair.

7.3.2 Same interchange mechanism for all level pairs

This Technical Report allows for different Data Modelling Facilities. By standardizing the data modelling facility to be used for interchange between the different data management environments distributed data management can be supported.

The Data Modelling Facility selected as the standard for a distributed data management system is called the interchange Data Modelling Facility.

Which Data Modelling Facility is used as the interchange Data Modelling Facility should not be a factor of which the user of the services provided by a User Processor necessarily needs to be aware.

It is important that the same interchange Data Modelling Facility be supported in different management domains if interchange of data is to be possible between them. The interchange data modelling facility and data modelling facility common to level pairs should be the same data modelling facility.

7.3.3 Same processors usable for all level pairs

An exact representation of the schema in a form appropriate for the standard data modelling facility is needed.

A standardized representation form is essential if processors such as different Database Controllers and User Processors are to be able to share such information.

It should be possible for the services provided by a User Processor to be aware of more than one level pair. It should also be possible for the services provided by a User Processor to be expressed in such a way that the user of these services regards all data as being associated with one level pair.

7.3.4 Standardized services at Database Controller interface

In order to facilitate interoperability of database environments, the services interface to a Database Controller must be standardized.

These services may be a subset of those required to support a Database Language standard, but an alternative and appropriate representation form is required.

7.3.5 Standardized approach to access control

As a consequence of enabling heterogeneous distributed databases and interoperable separately developed systems, a necessary objective is that of a standardized approach to access control.

7.3.6 Standardized representation of data needed to facilitate interoperability

The management of a distributed database requires a database of data fragment location information, which may itself be distributed.

In the case of a centralized data management system, the location information relates to the particulars of the specific hardware/software environment, and can be completely proprietary in nature.

An objective is that data location information must be globally available within the distributed database management system network. In practice, this necessitates standardization in the format utilized to represent such location information.

7.3.7 Support data fragmentation

Standardization of distribution data, and the manner of accessing the data fragments related to such distribution data, is needed if heterogeneous distributed databases and interoperability are to be supported.

A standard data modelling facility provides a means of structuring such information. These facilities should be utilized to provide both local and remote access to such information.

A Distribution Processor must also be able to address horizontal and vertical fragmentation in distributed databases.

A data management system utilizes such location information in order to provide some degree of independence for application processes. The results of a request for information which requires data from multiple sites will be consolidated prior to presentation of the data to the requesting application.

7.3.8 Separation of logical and physical structures

The way in which data is structured according to the data structuring rules of a data modelling facility is referred to as the logical structure of the data. The specification of this must be separate from the way in which a given logical structure is to be represented as persistent data in a storage device.

It is often necessary to change the physical data structure. The need for the change may or may not be triggered by a logical restructuring. In either case, it is necessary to ensure that the physical representation of the data conforms to the data definition.

7.3.9 Access to schema during execution

The requirement for a single definitive source for data under the control of an Information Resource Dictionary System can be addressed by a data dictionary system which is “active” with respect to the other processors utilized in an information system.

The control of data stored in a database is handled by a dictionary system in which the metadata is accessible in the same way as the schema of the database.