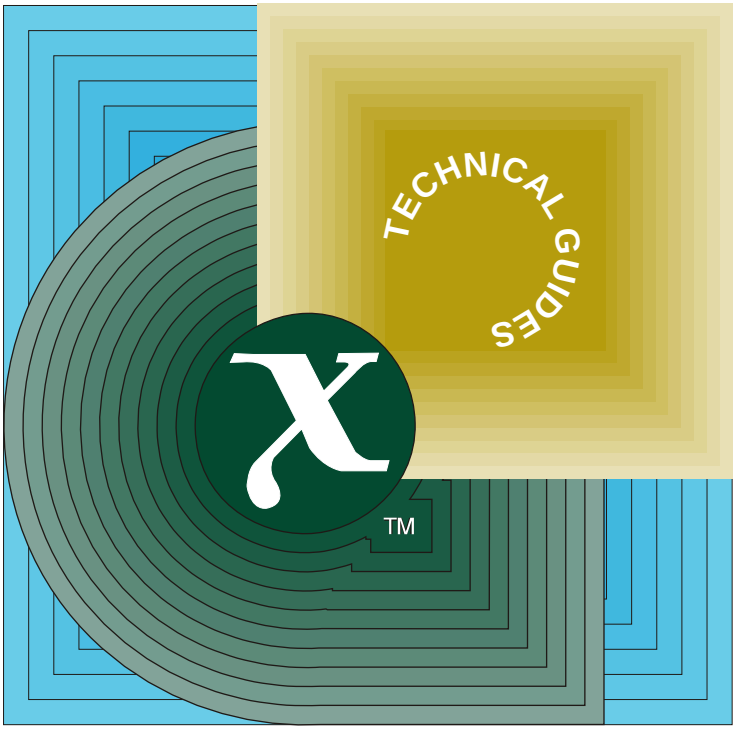
Guide

Data Management: Reference Model



THE Open GROUP

[This page intentionally left blank]

X/Open Guide

Data Management: Reference Model

X/Open Company Ltd.

X/Open Guide

Data Management: Reference Model

ISBN: 1-85912-155-1

X/Open Document Number: G505

Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to X/Open at:

X/Open Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

XoSpecs@xopen.org

Contents

List of Figures

Preface

X/Open

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable open system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining.

X/Open defines this CAE in a set of specifications which include an evolving portfolio of application programming interfaces (APIs) which significantly enhance portability of application programs at the source code level, along with definitions of and references to protocols and protocol profiles which significantly enhance the interoperability of applications and systems.

The X/Open CAE is implemented in real products and recognised by a distinctive trade mark — the X/Open brand — that is licensed by X/Open and may be used on products which have demonstrated their conformance.

X/Open Technical Publications

X/Open publishes a wide range of technical literature, the main part of which is focussed on specification development, but which also includes Guides, Snapshots, Technical Studies, Branding/Testing documents, industry surveys, and business titles.

There are two types of X/Open specification:

- *CAE Specifications*

  CAE (Common Applications Environment) specifications are the stable specifications that form the basis for X/Open-branded products. These specifications are intended to be used widely within the industry for product development and procurement purposes.

  Anyone developing products that implement an X/Open CAE specification can enjoy the benefits of a single, widely supported standard. In addition, they can demonstrate compliance with the majority of X/Open CAE specifications once these specifications are referenced in an X/Open component or profile definition and included in the X/Open branding programme.

  CAE specifications are published as soon as they are developed, not published to coincide with the launch of a particular X/Open brand. By making its specifications available in this way, X/Open makes it possible for conformant products to be developed as soon as is practicable, so enhancing the value of the X/Open brand as a procurement aid to users.

- *Preliminary Specifications*

  These specifications, which often address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations, are released in a controlled manner for the purpose of validation through implementation of products. A Preliminary specification is not a draft specification. In fact, it is as stable as X/Open can make it, and on publication has gone through the same rigorous X/Open development and review procedures as a CAE specification.

  Preliminary specifications are analogous to the *trial-use* standards issued by formal standards organisations, and product development teams are encouraged to develop products on the basis of them. However, because of the nature of the technology that a Preliminary specification is addressing, it may be untried in multiple independent implementations, and may therefore change before being published as a CAE specification. There is always the intent to progress to a corresponding CAE specification, but the ability to do so depends on consensus among X/Open members. In all cases, any resulting CAE specification is made as upwards-compatible as possible. However, complete upwards-compatibility from the Preliminary to the CAE specification cannot be guaranteed.

In addition, X/Open publishes:

- *Guides*

  These provide information that X/Open believes is useful in the evaluation, procurement, development or management of open systems, particularly those that are X/Open-compliant. X/Open Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming X/Open conformance.

- *Technical Studies*

  X/Open Technical Studies present results of analyses performed by X/Open on subjects of interest in areas relevant to X/Open's Technical Programme. They are intended to communicate the findings to the outside world and, where appropriate, stimulate discussion and actions by other bodies and the industry in general.

- *Snapshots*

  These provide a mechanism for X/Open to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A Snapshot represents the interim results of an X/Open technical activity. Although at the time of its publication, there may be an intention to progress the activity towards publication of a Specification, Guide or Technical Study, X/Open is a consensus organisation, and makes no commitment regarding future development and further publication. Similarly, a Snapshot does not represent any commitment by X/Open members to develop any specific products.

Versions and Issues of Specifications

As with all *live* documents, CAE Specifications require revision, in this case as the subject technology develops and to align with emerging associated international standards. X/Open makes a distinction between revised specifications which are fully backward compatible and those which are not:

- a new *Version* indicates that this publication includes all the same (unchanged) definitive information from the previous publication of that title, but also includes extensions or additional information. As such, it *replaces* the previous publication.

- a new *Issue* does include changes to the definitive information contained in the previous publication of that title (and may also include extensions or additional information). As such, X/Open maintains *both* the previous and new issue as current publications.

Corrigenda

Most X/Open publications deal with technology at the leading edge of open systems development. Feedback from implementation experience gained from using these publications occasionally uncovers errors or inconsistencies. Significant errors or recommended solutions to reported problems are communicated by means of Corrigenda.

The reader of this document is advised to check periodically if any Corrigenda apply to this publication. This may be done either by email to the X/Open info-server or by checking the Corrigenda list in the latest X/Open Publications Price List.

To request Corrigenda information by email, send a message to info-server@xopen.co.uk with the following in the Subject line:

```
request corrigenda; topic index
```

This will return the index of publications for which Corrigenda exist.

This Document

This document is a Guide (see above). It provides a functional description of the X/Open Data Management Reference Model (the X/Open Model), a software architecture that allows multiple applications to share data resources in a distributed processing environment using common data definition, manipulation languages and remote access protocols.

The Guide is structured as follows:

- Chapter 1 on page 1 is an introduction.
- Chapter 2 on page 3 provides some basic definitions.
- Chapter 3 on page 5 describes general data management requirements.
- Chapter 4 on page 9 provides a description of basic concepts.
- Chapter 5 on page 13 describes the X/Open Model.
- Chapter 6 on page 27 provides a detailed description of the functions provided by the component parts of the X/Open Model.
- Chapter 7 on page 43 shows some applications of the X/Open Model.
- Appendix A on page 49 looks at the X/Open **RDA over OSI and Non-OSI Transport Providers** specification (published as an X/Open Consortium Specification).

Intended Audience

This guide is intended to introduce the X/Open Model to application developers, system administrators and product builders and suppliers. It assumes prior knowledge of database and transaction processing.

Typographical Conventions

The following typographical conventions are used throughout this document:

- **Bold** font is used in text for keywords and references to published standards and specifications.
- *Italic* strings are used for emphasis and to identify the first instance of a word requiring definition.

Trade Marks

UNIX® is a registered trade mark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X/Open® is a registered trade mark, and the ‘‘X’’ device is a trade mark, of X/Open Company Limited.

Referenced Documents

The following documents are referenced in this guide.

X/Open Data Management Documents

CLI
: X/Open CAE Specification, February 1995, Data Management: SQL Call Level Interface (CLI) (ISBN: 1-85912-081-2, C451).

RDA
: X/Open CAE Specification, August 1993, Data Management: SQL Remote Database Access (ISBN: 1-872630-98-7, C307).

RDA over OSI and Non-OSI Transport Providers
: X/Open Consortium Specification, January 1995, SQL Remote Database Access over OSI and Non-OSI Transport Providers (J501).

SQL
: X/Open CAE Specification, August 1992, Structured Query Language (SQL) (ISBN: 1-872630-58-8, C201).

SQL, Version 2
: X/Open Preliminary Specification, April 1995, Data Management: Structured Query Language (SQL), Version 2 (ISBN: 1-85912-093-8, P446).

X/Open Distributed Transaction Processing Documents

CPI-C, Version 2 (contains Peer-to-Peer)
: X/Open Preliminary Specification, November 1994, The CPI-C Specification, Version 2, X/Open Document Number P415, ISBN: 1-85912-057-1.

DTP
: X/Open Guide, November 1993, Distributed Transaction Processing: Reference Model, Version 2 (ISBN: 1-85912-019-9, G307).

TX
: X/Open Preliminary Specification, October 1992, Distributed Transaction Processing: The TX (Transaction Demarcation) Specification (ISBN: 1-872630-65-0, P209).

TxRPC
: X/Open Preliminary Specification, July 1993, Distributed Transaction Processing: The TxRPC Specification (ISBN: 1-85912-000-8, P305).

XA
: X/Open CAE Specification, December 1991, Distributed Transaction Processing: The XA Specification (ISBN: 1-872630-24-3, C193 or XO/CAE/91/300).

XA+
: X/Open Snapshot, July 1994, Distributed Transaction Processing: The XA+ Specification, Version 2 (ISBN: 1-85912-046-6, S423).

XATMI
: X/Open Preliminary Specification, July 1993, Distributed Transaction Processing: The XATMI Specification (ISBN: 1-872630-99-5, P306).

Other X/Open Documents

mOSI
: X/Open Preliminary Specification, August 1994, Minimum OSI Functionality (contained in Networking Services, Issue 4, Appendix H).

Networking Services, Issue 4
: X/Open CAE Specification, August 1994, Networking Services, Issue 4 (ISBN: 1-85912-049-0, C438).

XAP
: X/Open CAE Specification, September 1993, ACSE/Presentation Services API (XAP) (ISBN: 1-872630-91-X, C303).

Non-X/Open Documents

ISO C
: ISO/IEC 9899: 1990, Programming Languages — C (technically identical to ANSI standard X3.159-1989).

ISO CLI
: ISO/IEC DIS 9075-3, Information Technology — Database Language SQL, Part 3: SQL Call Level Interface (SQL/CLI).

ISO COBOL
: ISO 1989: 1985, Programming Languages — COBOL (technically identical to ANSI standard X3.23-1985).

ISO RDA
: ISO/IEC 9579-1: 1993, Information Technology — Open Systems Interconnection — Remote Database Access — Part 1: Generic Model, Service, and Protocol.

ISO RDA SQL Specialization
: ISO/IEC 9579-2: 1993, Information Technology — Open Systems Interconnection — Remote Database Access — Part 2: SQL Specialization.

ISO RMDM
: ISO/IEC 10032: 1995, Information Technology — Reference Model of Data Management.

ISO SQL
: ISO/IEC 9075: 1992, Information Technology — Database Language SQL (technically identical to ANSI standard X3.135-1992).

Minimal OSI
: ISO/IEC DISP 11188-3, International Standardized Profile — Common Upper Layer Requirements — Part 3: Minimal OSI Upper Layers Facilities, Version 6, 1994-04-14.

NIST FIPS 127-2
: Federal Information Procurement Standard (FIPS) 127-2, Database Language SQL, NIST, 25 January 1993.

OSI Transport Protocol
: ISO/IEC 8073: 1987, Information Technology — Telecommunications and Information Exchange Between Systems — Open Systems Interconnection — Protocol for Providing the Connection-mode Transport Service.

OSI Transport Service
: ISO 8072: 1986, Information Processing Systems — Open Systems Interconnection — Transport Service Definition.

RFC 1006
: ISO Transport Service on Top of the TCP, Version 3, May 1987, Marshall T. Rose and Dwight E. Cass, Network Working Group, Northrop Research and Technology Center.

TCP
: Transmission Control Protocol, RFC 793 (Defense Communication Agency, DDN Protocol Handbook, Volume II, DARPA Internet Protocols, (December 1985).

The following documents are useful as supplementary reading, but are not directly referenced:

Internationalisation Guide
: X/Open Guide, July 1993, Internationalisation Guide, Version 2 (ISBN: 1-859120-02-4, G304).

IP
: Internet Protocol, RFC 791, September 1981.

ISO 7498
: ISO 7498: 1984, Information Processing Systems — Open Systems Interconnection — Basic Reference Model.

ISO 8326
: ISO 8326: 1987, Information Processing Systems — Open Systems Interconnection — Basic Connection-oriented Session Service Definition.

ISO 8327
: ISO 8327: 1987, Information Processing Systems — Open Systems Interconnection — Basic Connection-oriented Session Protocol Specification.

ISO 8649
: ISO 8649: 1988, Information Processing Systems — Open Systems Interconnection — Service Definition for the Association Control Service Element.

ISO 8650
: ISO 8650: 1992, Information Processing Systems — Open Systems Interconnection — Protocol Specification for the Association Control Service Element.

ISO 8822
: ISO 8822: 1988, Information Processing Systems — Open Systems Interconnection — Connection-oriented Presentation Service Definition.

ISO 8823
: ISO 8823: 1988, Information Processing Systems — Open Systems Interconnection — Connection-oriented Presentation Protocol Specification.

ISO 8824
: ISO 8824: 1990, Information Technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1).

ISO 8859-1
: ISO 8859-1: 1987, Information Processing — 8-bit Single-byte Coded Graphic Character Sets — Part 1: Latin Alphabet No. 1.

OIW RDA
: Stable Implementation Agreements for Open Systems Interconnection Protocols: Part 19 — Remote Database Access, Output from the December 1992 OSE Implementors' Workshop.

OSI DTP

ISO / IEC 10026: 1992, Information Technology — Open Systems Interconnection — Distributed Transaction Processing.

Chapter 1

Introduction

1.1 Overview

The X/Open Data Management Reference Model (the X/Open Model) is a software architecture that allows multiple application programs to define and access shared data resources managed by heterogeneous database management systems using a variety of interfaces. It supports both local and remote database management systems and allows the work of the application programs to be managed by the X/Open Distributed TP facilities.

The X/Open Model comprises three basic components:

- an application program (AP), which defines data and specifies creation, manipulation and retrieval of defined data
- database management systems (DBMSs) which provide facilities for the definition and storage of data and for the insertion, deletion, modification and retrieval of data in that store
- Structured Query Language (SQL), which specifies the language with which an AP communicates with DBMSs.

In addition to these three data management components, the X/Open Model also references the transaction manager (TM) and communication resource manager (CRM) components of the X/Open Distributed TP Model.

X/Open Data Management publications based on this model specify application programming interfaces (APIs), communication protocols and system-level interfaces that facilitate:

- portability of application program source code to any X/Open environment that offers those APIs
- interchangeability of SQL clients, SQL servers, RDA clients, RDA servers, TMs and CRMs from multiple vendors
- interoperability in the same global transaction of SQL clients, SQL servers, RDA clients, RDA servers, TMs and CRMs from multiple vendors.

1.2 Benefits of X/Open Data Management

Data management provides the necessary mechanisms for an organisation to manage and control one of its most valuable resources — its information. It allows the definition, storage and retrieval of the data representing this information in a manner which is independent of location and independent of the hardware and software platform used. It enables the construction of applications that manipulate data consistently using multiple products simultaneously either locally or through some form of network connection.

The portability, interchangeability and interoperability aspects allow application writers to benefit from a wider choice of database management systems and defined interfaces to other related areas such as transaction managers and communications resource managers. Application writers also gain the freedom to select from alternative products and hardware and software platforms.

Published interface specifications simplify software design. For example, some software products that might once have been implemented using customised interfaces may now be implemented and viewed as interchangeable data management components. Branding of products assures the purchaser that the product meets all the conformance requirements of the applicable component and profile definitions and indicates the commitment of the supplier to the X/Open specifications. This approach shortens development time and extends the above benefits to a greater number of products.

All parties benefit from the X/Open method of achieving consensus and communication among open systems providers, with a significant voice for system vendors, independent software vendors (ISVs) and users.

1.3 Areas Not Addressed

The X/Open Model does not address all the issues of importance in data management, for example:

- data modelling
- information resources directories, dictionaries, and so on
- import/export
- specification of benchmarks
- *ad hoc* query tools
- reorganisation/restructuring
- access control
- configuration management
- auditing
- recovery/backup
- physical storage structures.

1.4 Relationship to International Standards

The X/Open Model is compatible with existing International standards in the area of data management. A detailed comparison of the X/Open Model and other X/Open Data Management specifications is given in Section 5.6 on page 24.

Chapter 2

Definitions

This chapter gives fundamental definitions on which subsequent chapters depend. It is structured as follows:

- general definitions
- data definitions
- transaction definitions.

2.1 General Definitions

client
A role filled by a processor when it requests the services provided by another processor (that is, server).

connection
A connection is a temporary association between a client and a server.

information system
A system which organises the storage and manipulation of information about a universe of discourse.

persistent
Continuing to exist indefinitely, unless and until deliberately destroyed.

server
A role filled by a processor when it provides services to other processors (that is, clients).

service
A capability provided by a processor to other processors.

2.2 Data Definitions

catalog
A catalog is a named collection of schemata.

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

descriptor
A descriptor is a conceptually structured collection of data that defines the attributes of an instance of an object of a specified type.

handle
An opaque data value returned by an SQL CLI implementation when a CLI resource is allocated and is used by an SQL CLI application to reference that CLI resource.

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

schema
A schema is a collection of related objects. A schema may contain base tables, views, character sets and collations, and contains any indexes that are defined on the base tables.

2.3 Transaction Definitions

transaction
A set of related operations characterised by four properties. These properties are:

atomicity	The results of the transaction's execution are either all committed or all rolled back.
consistency	A transaction transforms a shared resource from one valid state to another valid state.
isolation	Changes to shared resources that a transaction affects do not become visible outside the transaction until the transaction commits.
durability	The changes that result from the transaction commitment survive subsequent system or media failures.

Chapter 3

Data Management Requirements

The purpose of this chapter is to describe the relevant concepts of information systems and, in particular, those aspects of information systems which place requirements on data management. Thus the scope of data management can be determined.

3.1 Information Systems

In order to function, an organisation needs to collect, keep and process information about its own operations and those of others, its 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 operational requirements and an organisation will normally have several information systems that together meet its total requirements.

An information system may be located on one computer system or it may be distributed across two or more interconnected computer systems.

Data flows into and out of an information system and these interactions may be with either people or other processes, including other information systems, automated machines, and so on. An information system may support many forms of interaction and each interaction may be subject to authorization controls. Interactions with an information system may occur concurrently.

3.2 Context of Data Management in an Information System

Figure 3-1 shows the context and component parts of an information system.

A computer system consists of:

- a collection of hardware that is managed as a single unit by software such as an operating system
- one or more information systems or parts of information systems
- a collection of persistent data used and operated upon by the information systems.

An information system itself consists of several component parts. These are:

- A Human-Computer Interface (HCI) which provides for the ultimate interaction of a human user with the information system. Although not part of the area covered by this guide, X/Open provides several specifications to promote a common approach to the user interface.
- An application program or programs which provide the specific functional logic and control of the information system. Although not part of the area covered by this guide, X/Open provides for the enhanced portability of application programs through the provision of operating system and language specifications.

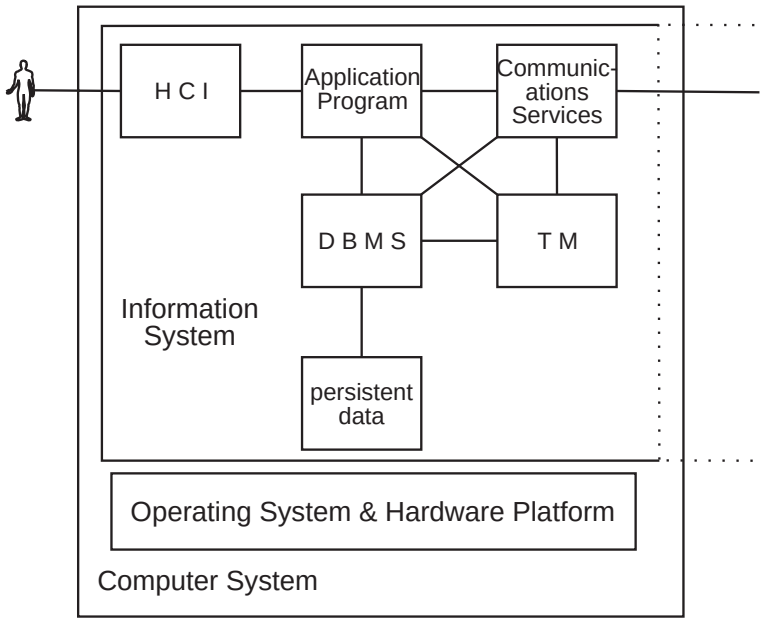


Figure 3-1 Information System Context

- One or more database management systems (DBMSs) which manage the storage and retrieval of the data associated with the information system.
- A transaction manager (TM) that provides for global transaction control over the various database management systems (and potentially other resource managers not shown in the diagram).

and optionally:

- Communications services that provide access to other computer systems containing further parts of the information system or systems present on this computer system or other

independent information systems.

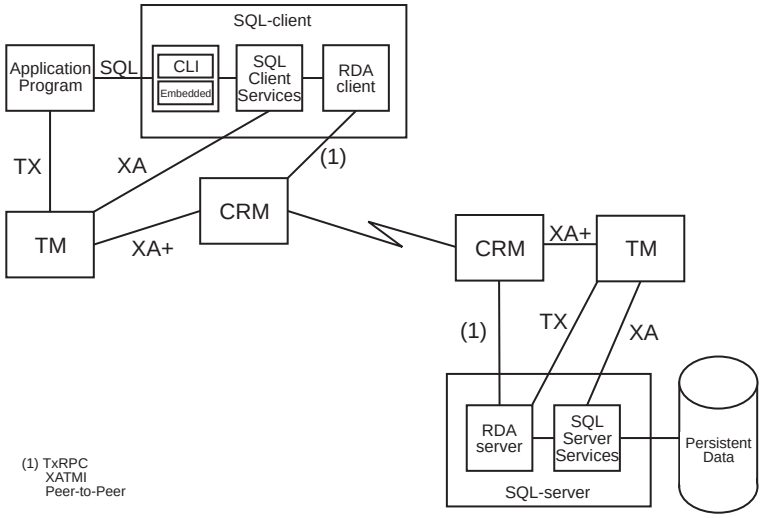


Figure 3-2 Overview of Components, Protocols and Interfaces

This reference model is specifically concerned with the application program, database management system, transaction manager and communications services components of the total picture. These ‘‘data management’’ components and their relationships are shown in more detail in Figure 3-2.

The SQL language, the CLI and Embedded application program interfaces, and the SQL client and SQL server functional components, as shown in Figure 3-2, are described in the various X/Open Data Management specifications.

The protocols TX, XA, XA+, TxRPC, XATMI and Peer-to-Peer and the functional components TM and CRM, as shown in Figure 3-2, are described in the various X/Open Distributed TP specifications.

Chapter 4

Concepts

This chapter describes a few basic concepts which are essential to the understanding of the rest of the document.

4.1 Persistent Data

Persistent data is at the heart of any database management system. The *raison d'être* of any database is to hold and make available information, data, about a particular subject or area of interest.

We can divide persistent data into two classes — data and metadata.

Data represents information about something, usually something in the real world but sometimes abstract ideas or concepts. That is, the data describes some object of interest to an information system using it. Metadata, however, represents information about the data. Thus we can envision a hierarchy as shown in Figure 4-1.

Data and metadata are relative terms, as metadata can also be considered data and might therefore be expected to be described by metadata, or in this case meta-metadata.

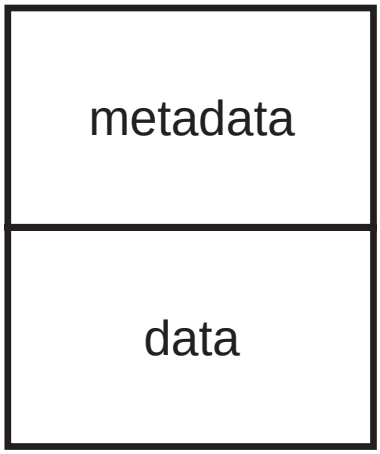


Figure 4-1 Data and Metadata

In the ISO Reference Model of Data Management (ISO RMDM), this relationship is described in terms of ''Level Pairs'', whereby the lower level of one pair (data) forms the upper level (metadata) of the next lower pair. In the ISO RMDM, the number of level pairs is unrestricted. In the X/Open Model, however, the number of level pairs is just two. This is shown in Figure 4-2 on page 10. Here the application data is the lowest level and contains the data required by the applications using the database.

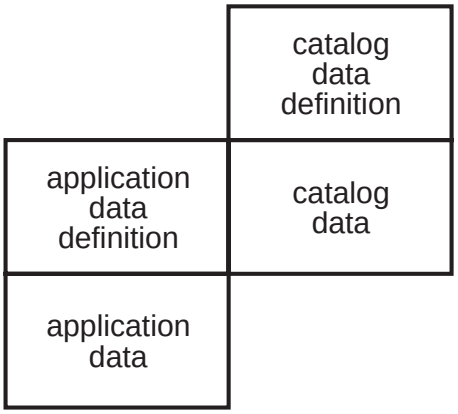


Figure 4-2 Level Pairs

The application data definition level defines the way in which the application data is to be stored. In SQL terms, it describes the tables, columns, and so on, used to hold the data. A collection of related descriptions or definitions of data is called a schema in SQL. A schema has an owner, who has exclusive control over the use of the objects in the schema. A schema also has a separate name space; that is, the name of every object of the same type in a schema must be unique, but since the name of the schema is also always associated with an object, objects in different schemata may have the same name. This description (schema) is also accessible data within the context of SQL and is contained in the ''Information Schema''. This Information Schema forms the catalog data definition level but since it is also, at least conceptually, a set of SQL tables (views) it is folded back onto the catalog data, which is said therefore to be self-describing.

Not all the catalog data is metadata for the application data. A database management system must also deal with some form of administration data. This usually includes definitions of users and their privileges, and so on, and this is held in the catalog as data, but at a different level.

4.2 Client/Server

Client/server is a general concept whereby one process, called the server, provides services to another process, called the client.

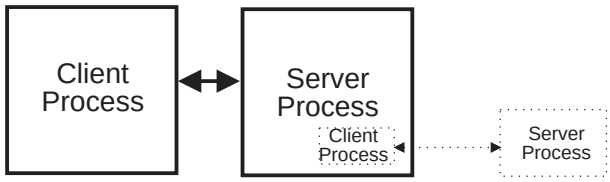


Figure 4-3 The Client/Server Model

Any server may itself request service from another process and thereby play the role of a client. A server can service multiple clients, either synchronously or simultaneously. The denomination of a process as client or server is thus a relative, and not an absolute, determination. Naturally, the ultimate client is a human being.

In computing, the client/server concept is most usually used to describe the relationship between processes running on physically distinct machines connected by a network or other communications mechanisms. In the X/Open Model the component parts can all be considered to have a client side and a server side. Unfortunately, in certain of the X/Open specifications some components are given names which include the words client and server. For example, RDA client and SQL server. It should be noted that these names are given purely from the perspective of the individual specification in which they occur and do not imply a permanent role. An RDA client can be a server for SQL client services and an RDA server a client of SQL server services. To allow easy reference to the specifications these slightly misleading names have been used in this document. In reading the document readers should remember that sometimes the role of a component may not match that implied by its name.

Chapter 5

The Model

This chapter gives an abstract description of the X/Open Model of an environment for information systems processing.

5.1 Components

The X/Open Model covers three component specifications. These are:

- the X/Open **SQL** specification
- the X/Open **RDA** specification
- the X/Open **CLI** specification.

The X/Open **SQL** specification forms the basis of the model. It provides a language for the definition and manipulation of data and metadata. It also provides the specification of an interface, the embedded SQL interface, to an SQL database management system from the programming languages C and COBOL. The X/Open **CLI** specification provides an alternative interface to an SQL database management system from these languages using a functional rather than an embedded style. The X/Open **RDA** specification defines the preferred way of connecting to one or more SQL databases on remote systems.

This guide references Version 2 of the X/Open **SQL** specification, currently an X/Open Preliminary Specification. The current CAE version is listed in **Referenced Documents**.

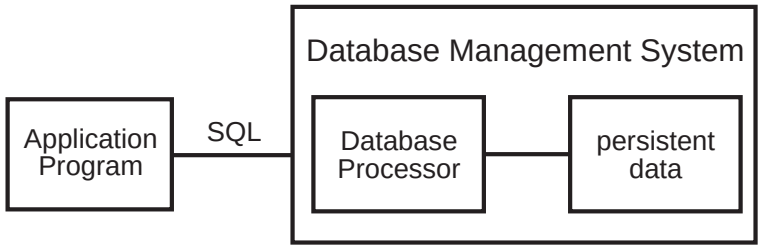


Figure 5-1 Data Management Overview

Figure 5-1 gives an overview of the X/Open Data Management architecture. SQL is shown as the language interface between the AP and the database management system which contains the database processor and persistent data. The SQL Call Level Interface and the SQL Remote Database Access should be considered as being internal to the database processor box. Associated processors such as those of Distributed TP are not shown.

The application program implements the desired function of the end-user enterprise. Each application program specifies a sequence of database operations using a standard language, SQL. This language is described by the X/Open **SQL** specification.

The database processor processes the SQL commands and performs the physical operations on the persistent data necessary to achieve the results specified by the application program.

The model of the database processor can be seen in Figure 5-2 on page 14.

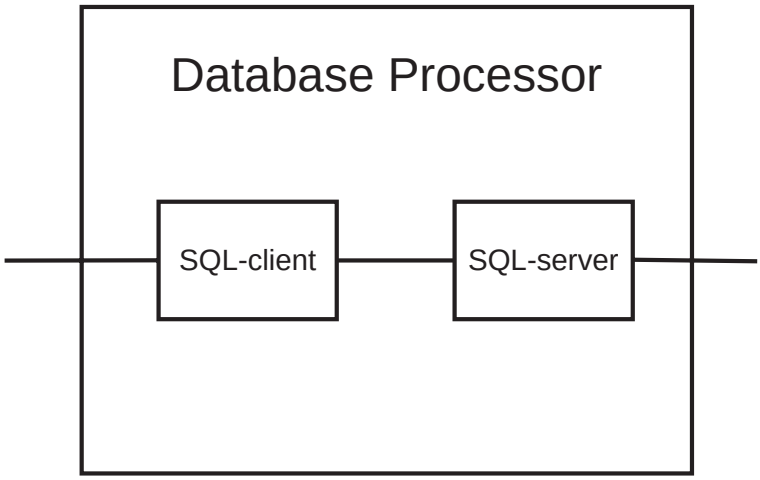


Figure 5-2 Database Processor

An application program communicates with an SQL client using an interface that accepts SQL language. One of the possible interfaces is defined by the X/Open **CLI** specification. Another possible interface is the embedded SQL interface defined in the X/Open **SQL** specification.

An SQL client communicates with one or more SQL servers using some communications interface and protocol. One of the options is provided by the X/Open **RDA** specification. This allows SQL clients and SQL servers to communicate using an open protocol.

5.2 Process Interfaces

The database management processes can be divided between three major components:

- the SQL client
- the SQL server
- Remote Database Access.

5.2.1 The SQL Client

The SQL client is that part of the database management system that interacts with application programs. The SQL client is responsible for intercepting SQL statements and determining where these statements need to be processed. The SQL client holds error and diagnostic information for those application programs that are connected to it and processes requests for such information. In addition, the SQL client takes the primary role in processing SQL connection statements.

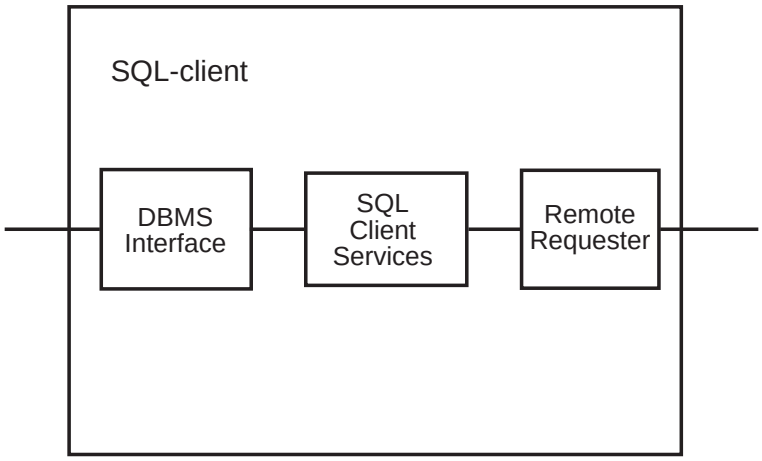


Figure 5-3 SQL Client

The SQL client itself can be broken down into three sub-components. These are shown in Figure 5-3.

The X/Open Model provides two alternative DBMS Interfaces. These are shown in Figure 5-4 on page 16.

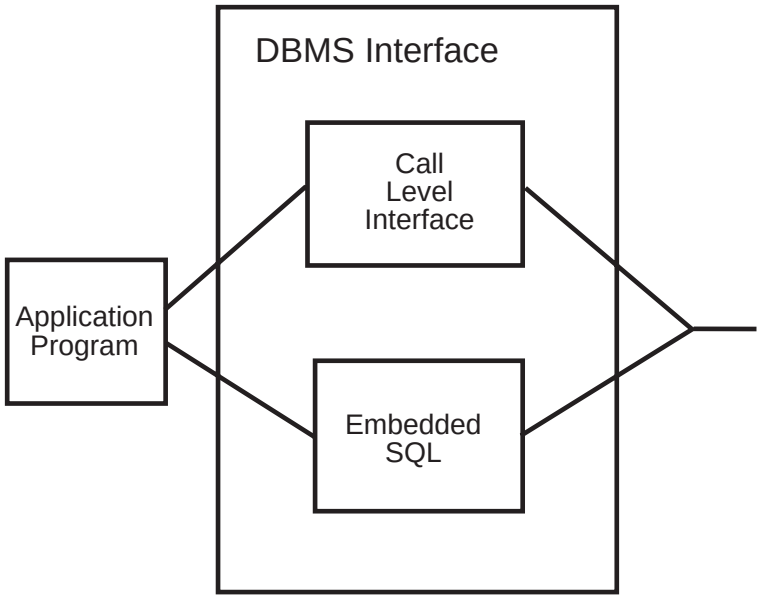


Figure 5-4 DBMS Interface

The first of the two interfaces is the embedded SQL interface for use from the programming languages ISO C and ISO COBOL. This is described in the X/Open **SQL** specification.

The second of the interfaces is the SQL Call Level Interface (CLI), which is separately specified.

Regardless of the interface used, the SQL commands will be processed by the SQL client services processor which will determine where the command should be processed. Requests for diagnostic information will be processed within the SQL client, and SQL connection statements will be intercepted and interpreted and the desired connection negotiated with the appropriate SQL server. On behalf of a single application program, an SQL client may connect to many SQL servers and indeed may connect more than once to the same server. These connections may be open concurrently or serially. The SQL client services processor may in some cases need to make uses of external services. For instance, user identification and authentication are not defined in SQL and so these services must be provided by some external processor. Similarly, SQL does not possess a directory services function. Location information needed to support the distribution of data is not specified by the SQL specification and may need to be provided by some external directory service.

The X/Open Model, unlike the ISO RMDM, does not have an explicit distribution controller. Distribution in the X/Open Model is controlled either by the AP through its use of the connection service or transparently by the SQL client. Distribution using RDA is limited to a single RM as RDA supports only one-phase commit. Distribution over more than one RM is permitted by SQL and CLI but is implementation-defined.

The X/Open Model provides two alternative ways for an SQL client to communicate with an SQL server. These are shown in Figure 5-5 on page 17.

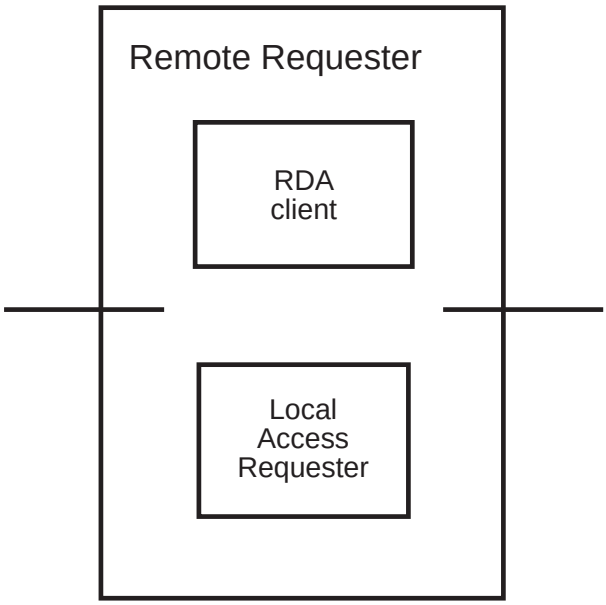


Figure 5-5 Remote Requester

In order to connect to an SQL server, the SQL client must choose one of two routes. The first route is that of the RDA service and protocols. The second is a direct interface, shown in the figure as Local Access Requester. This is the appropriate choice for the situation where both the SQL client and SQL server are situated on the same physical computer system but may also be used where the SQL client and SQL server are from the same vendor in order to provide a proprietary remote interface. In the more open environment of distributed computing, the preferred choice is the RDA service.

5.2.2 The SQL Server

The SQL server is that part of the Database Management System that operates upon and retrieves the persistent data from the computer systems storage media. The SQL server is responsible for accepting SQL statements from an SQL client and processing them. This processing can involve the retrieval, insertion, deletion or modification of data and/or metadata. The SQL server returns retrieved data and error and diagnostic information to the requesting SQL client.

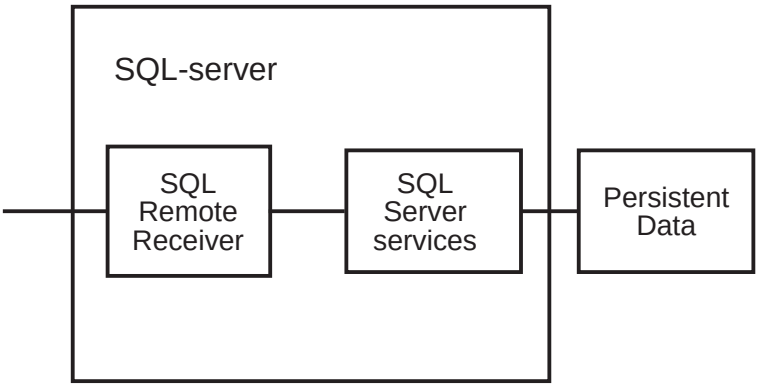


Figure 5-6 SQL Server

The SQL server itself can be broken down into three sub-components. These are shown in Figure 5-6.

In the previous section we have seen that the X/Open Model provides two alternative ways for an SQL client to communicate with an SQL server, thus the SQL server has two corresponding ways in which it, too, communicates.

These are shown in Figure 5-7.

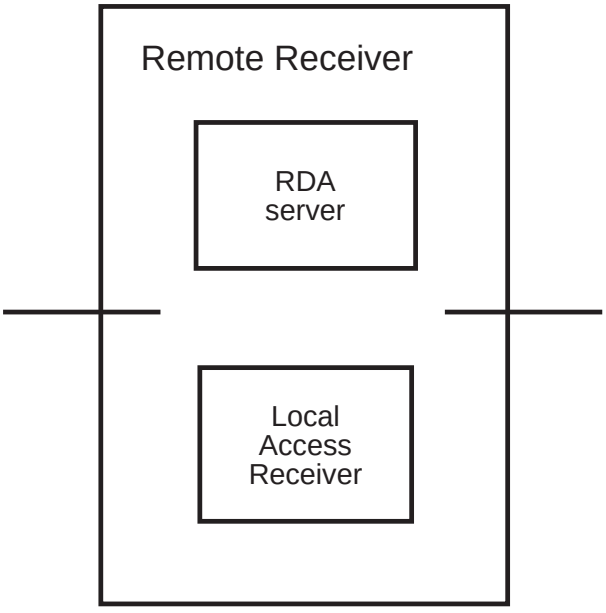


Figure 5-7 Remote Receiver

Depending on the route chosen by the communicating SQL client either the RDA server or the Local Access Receiver will be activated. As previously stated, the preferred choice for remote connections is RDA and, for the situation where both the SQL client and SQL server are situated on the same physical computer system, the Local Access Requester/Receiver interface is more appropriate.

It is conceivable that an SQL server may itself require data or metadata from another SQL server that is unknown to the SQL client which it is servicing. In this case, the SQL server takes on the role of SQL client and accesses the necessary SQL server as if it were an SQL client.

5.2.3 Remote Data Access

RDA consists of three sub-components:

- the RDA client
- the RDA server
- the RDA service-provider.

These are shown in Figure 5-8.

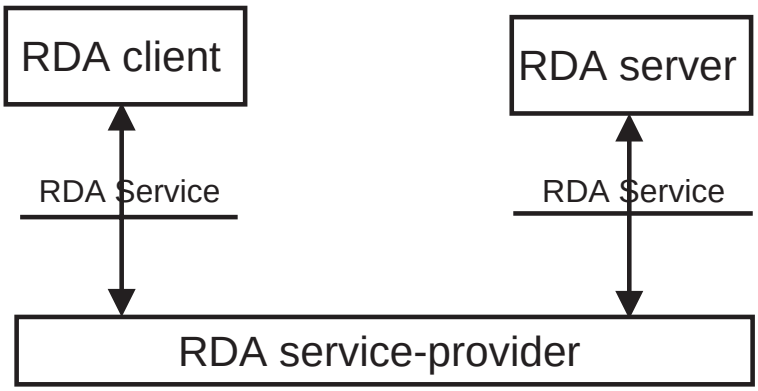


Figure 5-8 RDA Components

The RDA client is a user of the RDA service-provider. It initiates, on behalf of its own client (which may be an application program, but which is normally the SQL client component of an SQL database management system), a dialogue with an RDA server.

The RDA server is also a user of the RDA service-provider. It provides database services to RDA clients. These database services are not provided directly by the RDA server but by an SQL server co-located with the RDA server. The RDA server acts as an agent for the RDA client with regard to requests for services from the SQL server.

The RDA service-provider provides interworking between an RDA client and an RDA server. The RDA service-provider is equivalent to the Communications Resource Manager in the X/Open Distributed TP Model.

5.3 Data

The persistent data in the X/Open Model includes the application data (also called SQL data) and its metadata together with the necessary administration data. Persistent data is held in SQL tables and accessed by using various SQL statements. The SQL data is held in various SQL tables. The metadata for this data is, conceptually, also held in SQL tables as is the administration data. The metadata and the administration data together are collectively referred to as the schema for the data. The schema for the tables that hold the schema is called the ‘‘Information Schema’’.

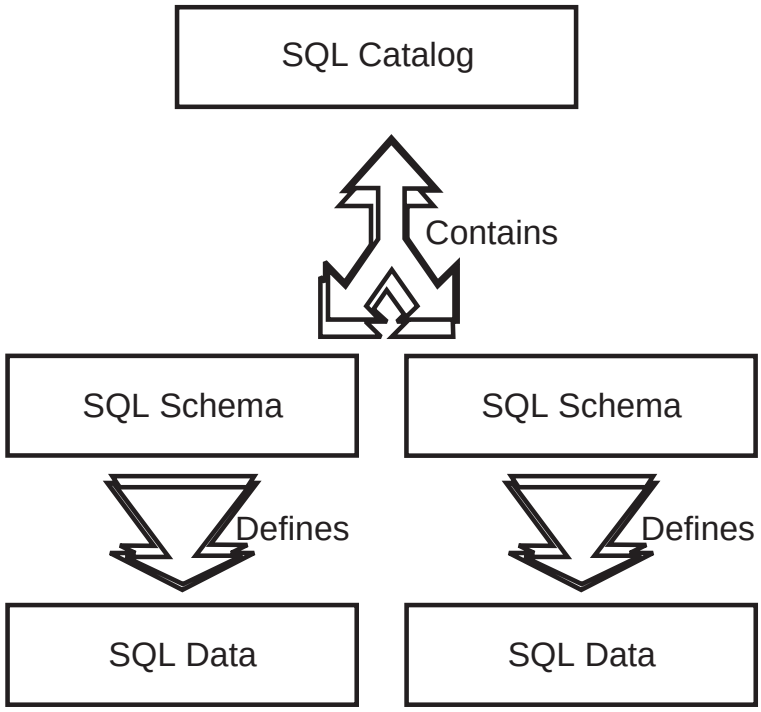


Figure 5-9 Catalog

A set of SQL schemata are gathered together in the X/Open Model into a group called a catalog. A catalog has no defined operational significance but forms part of the naming convention for all metadata objects. It may be used in a number of different ways, grouping schemata that all belong to a single business area, or that define data under the control of a single product, instance of a product, or a computer system. Alternatively, it may be used to distinguish between the test and production versions of schemata. The relationship between SQL data, SQL schemata and SQL catalogs is shown in Figure 5-9.

All the data and metadata held at a server are referred to in the X/Open Model as a database. The model leaves open the question of whether or not a schema or a catalog may span servers.

5.4 The Complete Model

Having decomposed a database management system into its component parts we can now put the parts back together again at look at the complete model. This is shown in Figure 5-10.

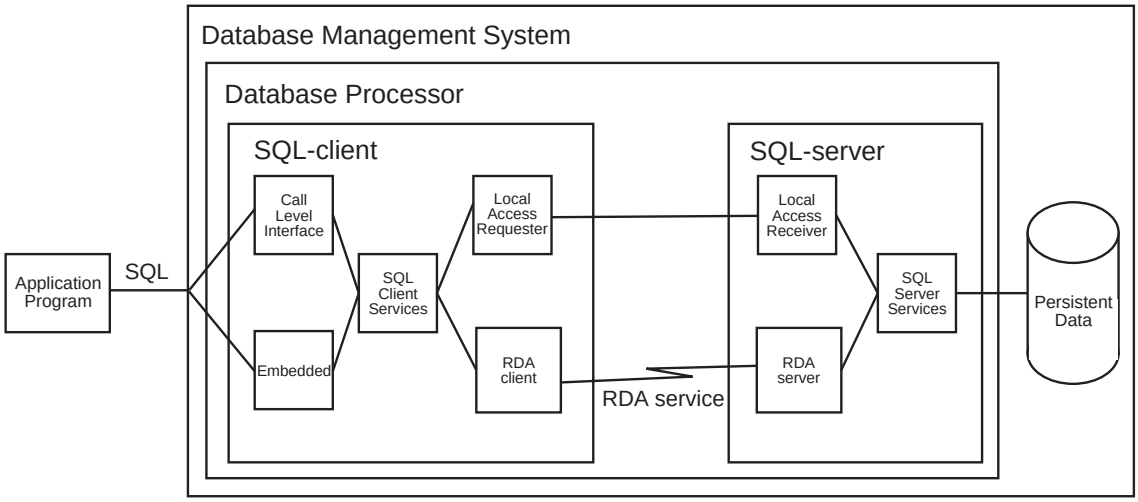


Figure 5-10 The X/Open Model

The figure does not show objects or interfaces outside of the scope of the X/Open Model. Thus the interfaces with Distributed TP and with other services such as authentication, authorization and directory are not shown. These are discussed elsewhere in this document.

5.5 Relationship to the X/Open Distributed TP Model

An SQL database management system is a resource manager (RM) in the terms of the X/Open Distributed TP Model. Thus an instance of an SQL DBMS fits into the X/Open Distributed TP Model in the place reserved for an RM in that model.

This relationship is shown in Figure 5-11.

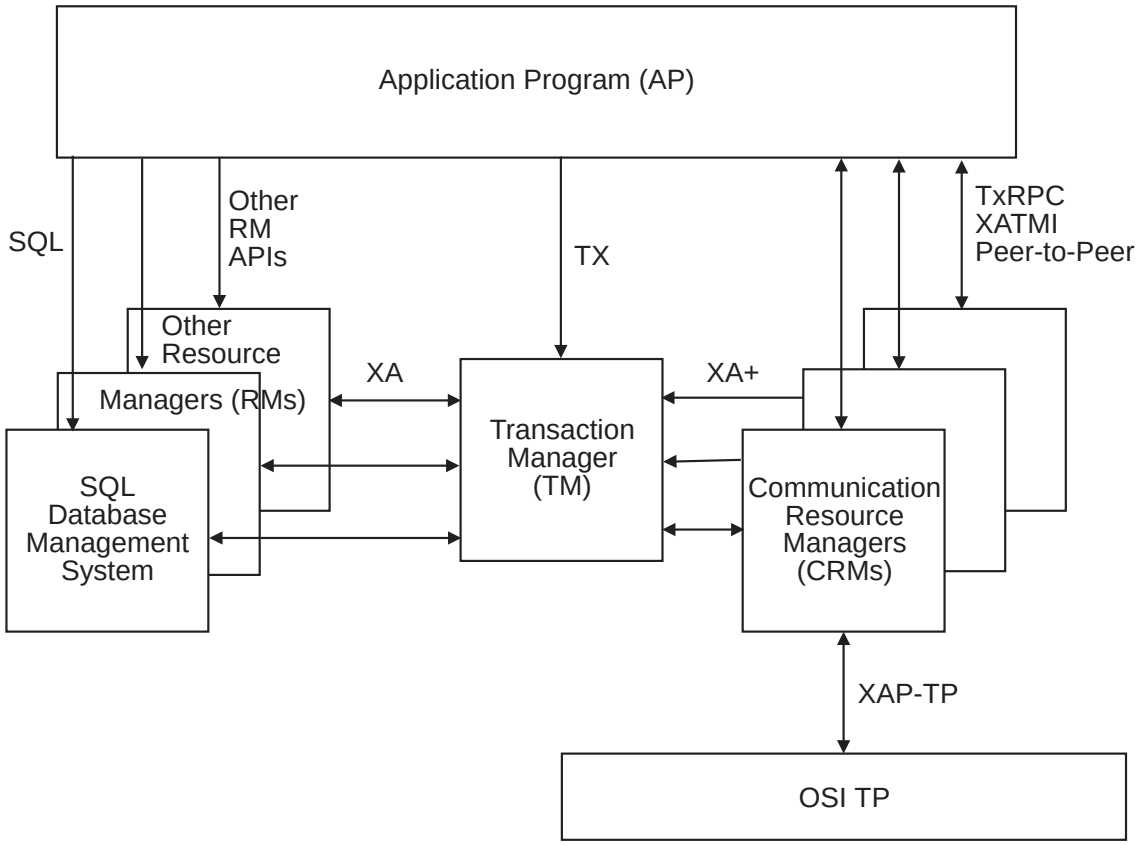


Figure 5-11 SQL and Distributed TP

In the diagram it can be seen that the native RM API for an SQL database management system is SQL. Other RMs may use their own native APIs. When working in the TP context the SQL API is restricted through the exclusion of the transaction control statements (see Section 6.1.2 on page 27). These statements are no longer needed as the application program uses the TX interface to perform the same transaction control functions through the transaction manager (TM).

The TM and the SQL database management system communicate with each other through the XA interface.

As shown in Figure 5-11, the SQL database management system RM may also communicate directly with a CRM; that is, assuming the role of AP as shown in the X/Open Distributed TP Model. This variable role concept is explicitly permitted by the X/Open Distributed TP specifications.

The current specifications for X/Open Database Management do not permit complete integration with the X/Open Distributed TP Model in the heterogeneous multiple TM domain

situation. In this situation, Distributed TP requires use of OSI-TP, but the current X/Open **RDA** specification excludes the TP context specified by ISO RDA. Integration in a single TM domain or homogeneous multiple TM domains is possible as is integration in the heterogeneous multiple TM domain situation when a proprietary (non-RDA) SQL client to SQL server interface is used.

5.6 Relationship to International Standards

This section highlights the differences between the components of the X/Open Model and the corresponding ISO standards.

5.6.1 SQL

The SQL language defined by X/Open contains on the one hand only a subset of the SQL language defined by ISO SQL, and on the other hand it adds some extensions to ISO SQL. The relationship of the X/Open **SQL** specification to the various levels of ISO SQL is shown in Figure 5-12.

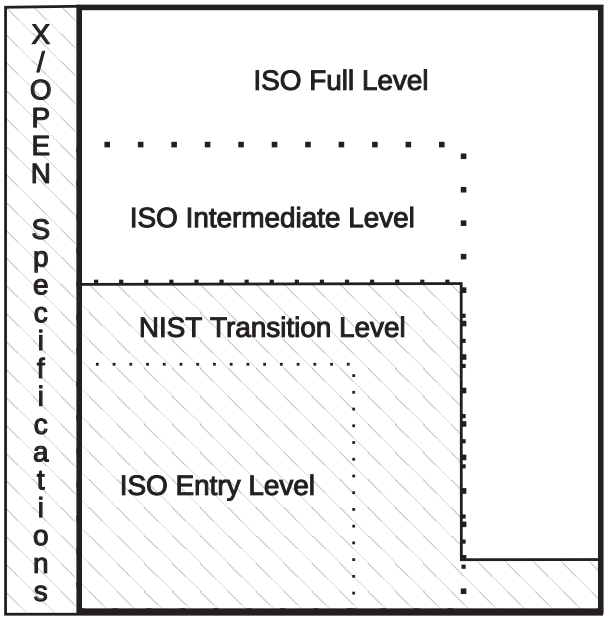


Figure 5-12 Relationship of X/Open SQL to ISO Conformance Levels

X/Open SQL subsumes both ISO SQL Entry level and Transition level defined by NIST FIPS 127-2. NIST's Transitional level is roughly midway between ISO's Entry level and Intermediate level. In addition to the Transitional level features, X/Open SQL also includes the SQL connection statements that only occur in ISO's Full level.

X/Open SQL includes some extensions that are not part of ISO SQL.

These extensions are:

- the requirement to be able to support multiple servers in a single transaction
- the storage schema (INDEX) statements and associated catalog information
- extensions to the Information schema to include a SERVER_INFO view and a REMARKS column in some views
- a vendor escape clause to allow optional specification of implementation-dependent syntax
- extensions to COBOL embedded host variables
- a WHENEVER SQLWARNING statement.

In addition to the Intermediate features not in the Transition level and the majority of the Full level features, ISO SQL also provides embedded language bindings for FORTRAN, Pascal, Ada, MUMPS and PL/I. X/Open does not currently support the bindings for FORTRAN, Pascal or Ada despite these being existing X/Open specifications.

ISO SQL also supports a module language binding and a Direct SQL interface. The Direct SQL interface is not relevant in the CAE and the module language is not specified.

5.6.2 RDA

Similar to the X/Open **SQL** specification, the X/Open **RDA** specification both subsets the capabilities provided by the ISO RDA standards (ISO RDA and ISO RDA SQL Specialization) and provides capabilities not found in those standards. The relationship of X/Open RDA to ISO RDA and ISO RDA SQL Specialization is shown in Figure 5-13.

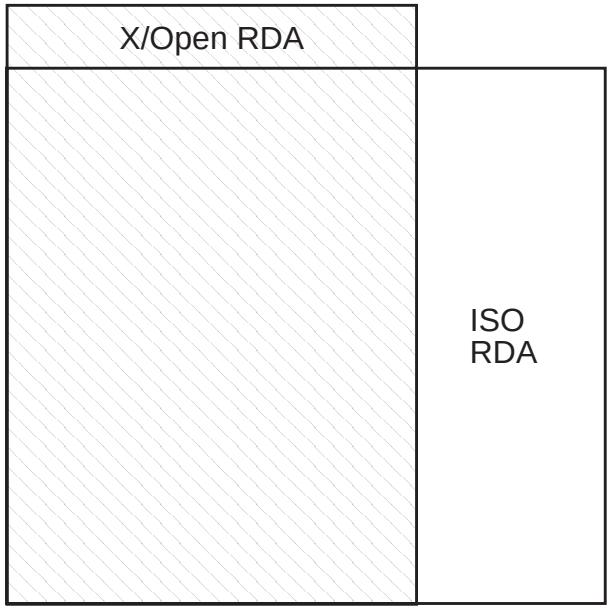


Figure 5-13 Relationship of X/Open RDA to ISO RDA

X/Open RDA subsets the capabilities of the ISO RDA standards by not providing the following features:

- the Stored Execution DBL functions, R-DefineDBL, R-InvokeDBL and R-DropDBL
- control services on another dialogue; that is, control services requested by one RDA dialogue to be performed on another RDA dialogue
- the RDA SQL TP application-context; that is, X/Open RDA does not provide two-phase commitment.

This subsetting preserves compatibility with the ISO RDA standards, since it pertains only to the use of optional capabilities.

X/Open RDA also constrains the ISO RDA standards by specifying:

- implementation agreements that specify limits on the values and use of the parameters of the ISO RDA protocol

- requirements on and implementation agreements for the use of the OSI layers below RDA.

These implementation agreements specify a profile of the ISO RDA standards, and therefore are compatible with the ISO RDA standards. (A profile defines a combination of base standards that collectively perform a specific, well-defined function, and standardises the use of options and other variations in those standards.)

X/Open RDA extends the ISO RDA standards by providing:

- diagnostic information and the diagnostics management statement
- the character varying data type
- dynamic SQL statements
- connection management statements
- certain data definition statements.

These extensions are required to support those features of ISO SQL beyond Entry level that are included in the X/Open **SQL** specification, Version 1 and are the only extensions to the ISO RDA standards.

5.6.3 SQL CLI

In addition to the features and facilities of ISO CLI, X/Open CLI includes the additional functions listed in Section 6.3.7 on page 38 and Section 6.3.11 on page 39, In addition, ISO CLI defines a number of functions: *AllocConnect*( ), *AllocEnv*( ), *AllocStmt*( ), *FreeConnect*( ), *FreeEnv*( ), *FreeStmt*( ), *ColAttribute*( ), *RowCount*( ) and *Error*( ) which, whilst included, have been deprecated in the CLI.

5.6.4 ISO RMDM

This X/Open Model is fully compatible with the ISO RMDM. The ISO RMDM covers more aspects of data management; for example, Import/Export and IRDS, than the X/Open Model which does not include those aspects. The ISO RMDM is, furthermore, a much more generic model than this X/Open Model which is concerned with providing a practical model.

Chapter 6

Detailed Services Provided

This chapter provides a fairly detailed description of the functions provided by three basic components of the X/Open Model.

6.1 SQL Services

SQL services are invoked by means of SQL statements.

The following are the main classes of SQL-statements:

- connection control
- transaction control
- data definition
- data access definition
- data manipulation
- dynamic SQL
- diagnostics
- embedded declaration.

6.1.1 Connection Control

These statements provide the connection management or client/server relationship management aspects of SQL:

CONNECT
DISCONNECT
SET CONNECTION

The statements provide for making, breaking and selecting between currently established connections. The establishment of a connection allows an application program to use the data resources of other SQL database servers than the, probably local, default server.

6.1.2 Transaction Control

These statements control the transaction management aspects of SQL where this control is not provided by an external source:

COMMIT
ROLLBACK
SET TRANSACTION

In situations where there is no Transaction Manager present, the SQL language provides simple transaction control through the above statements.

6.1.3 Data Definition

These statements have a persistent effect on catalogs:

ALTER TABLE
CREATE SCHEMA
CREATE TABLE
CREATE VIEW
DROP SCHEMA
DROP TABLE
DROP VIEW
GRANT
REVOKE

These are all concerned with defining the structure of the persistent data. This enables a single definition of the characteristics of the data to be held, independent of the application programs which will create or use the data. In addition to defining the basic characteristics of the data, such as the data type and length, these statements also allow the definition of constraints and relationships on or between data items. These constraints and relationships are enforced by the SQL database management system independently of the application programs. The GRANT and REVOKE statements provide the ability to specify control of the access to the defined data.

6.1.4 Data Access Definition

These statements have a persistent effect on catalogs and define physical access paths to rows of tables:

CREATE INDEX
DROP INDEX

The statements enable a database administrator to define access paths to data, independent of the logical data definition, and to change these access paths without needing to modify application programs.

6.1.5 Data Manipulation

These statements, sometimes in combinations, either retrieve SQL data or have a persistent effect on SQL data:

OPEN
FETCH
Positioned UPDATE
Positioned DELETE
CLOSE
SELECT INTO
INSERT
Searched UPDATE
Searched DELETE

These statements provide the basic manipulation and retrieval facilities of the SQL language. The first five are used to manipulate a cursor. They can be used whenever the operations on the database can be determined before the application program which contains them is built. They permit the creation, deletion and update of data using either a searched (set)-oriented approach, or a positioned (single row, navigated) approach. These statements are logically independent of the physical storage or access paths defined and available, although performance will indeed be affected by such aspects. Figure 6-1 on page 29 shows a typical control flow for processing SQL statements through a cursor. This is one example of a possible flow; other, more complex flows

are also possible.

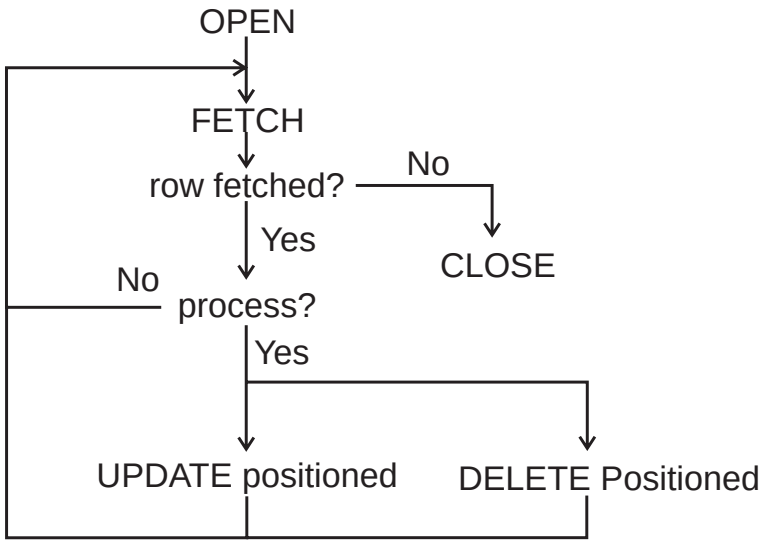


Figure 6-1 Basic Control Flow for Cursors

An OPEN statement opens a cursor, that has been previously specified with a DECLARE CURSOR declaration. At the time of the OPEN any parameter values are substituted and the derived table that is the cursor is completely defined. If necessary, the rows of the derived table are ordered. The cursor is positioned before the first row.

A FETCH statement retrieves a specified row from the cursor, if it exists. By default this is the next row, but other options are provided.

Once a row has been read it may be processed and this may involve updating or deleting the row. These actions are undertaken using the UPDATE Positioned and the DELETE Positioned statements respectively.

When a cursor has been completely read, a CLOSE statement is used to release the cursor and any resources associated with it.

6.1.6 Dynamic SQL

These statements provide for the dynamic execution of other SQL statements:

ALLOCATE DESCRIPTOR
DEALLOCATE DESCRIPTOR
DESCRIBE
Dynamic FETCH
Dynamic OPEN
EXECUTE
EXECUTE IMMEDIATE
GET DESCRIPTOR
PREPARE
SET DESCRIPTOR

These statements allow for the construction and execution of SQL statements which are not known at the time the application program is built. The statements which can be built and executed include all the data definition statements, data access definition statements, and the

data manipulation statements above, with the exception of the CLOSE, FETCH and OPEN. Other statements, in addition to these, may be accepted by specific implementations.

Figure 6-2 shows a typical control flow for processing Dynamic SQL statements. This is one example of a possible flow; other, more complex flows are also possible.

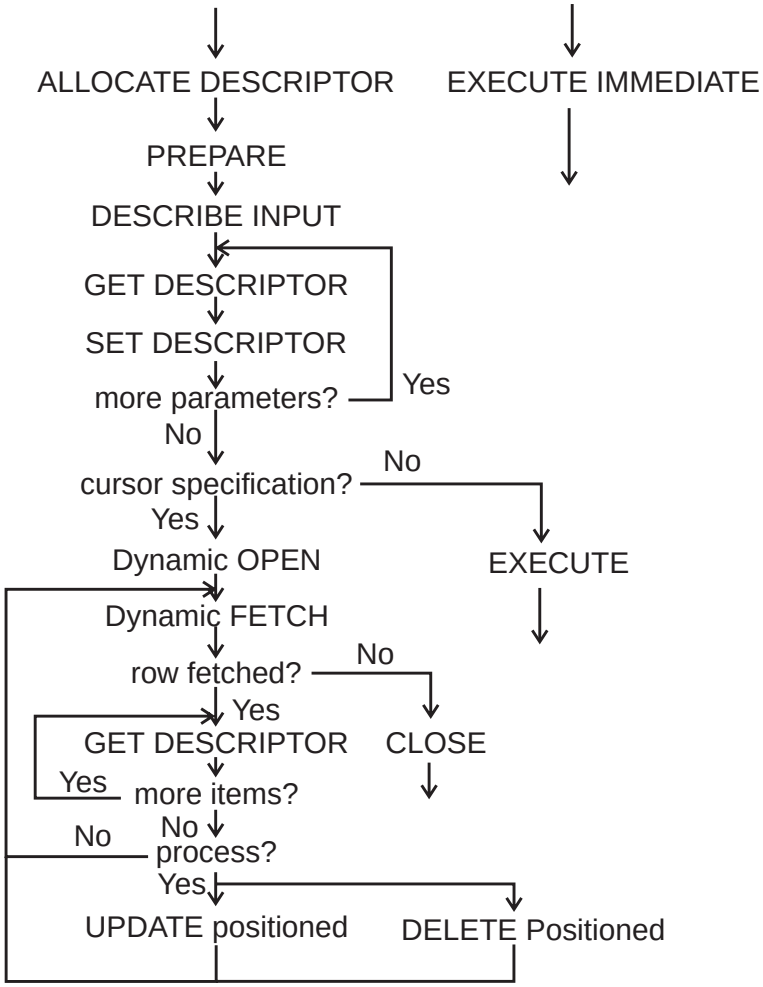


Figure 6-2 Basic Control Flow for Dynamic SQL

Dynamic SQL provides two major alternatives; one for SQL statements which return results and/or require parameters, and one for those which do not. The latter case is simply handled by the EXECUTE IMMEDIATE statement which treats its character as itself an SQL statement and performs it. The former requires slightly more complex handling.

An ALLOCATE DESCRIPTOR allocates resources which will permit the program to obtain information about input parameters and/or output column values.

A PREPARE statement takes a character string representing an SQL statement and performs what can best be compared to the compilation phase in conventional program generation. If the prepared statement contained parameter references, the number and types of these can be obtained with a DESCRIBE INPUT statement. This statement returns the information into an area allocated by the ALLOCATE DESCRIPTOR statement. Using GET and SET descriptor, information about individual parameters can be retrieved and the values of the parameters can be set.

Depending on the type of SQL statement prepared, one of two routes will then be followed.

If the prepared statement will either not return any rows, or at most one row, then an EXECUTE statement can be used. This completes the processing of the prepared statement using the supplied parameters.

If more than one row can be returned, then a dynamic form of cursor processing is performed. This processing has the same form as that for static SQL cursor processing, but the GET DESCRIPTOR statement may be used to obtain the values of items fetched through the cursor.

6.1.7 Diagnostics

This statement provides for the generation and return of error and diagnostic information:

GET DIAGNOSTICS

This statement allows an application program to retrieve not only more information about an error or warning condition raised as the result of executing a statement, but also allows the retrieval of multiple error conditions and warning together with the associated diagnostics from a single statement. For example, several constraints may have been violated by a single statement and all the relevant information is at once available, instead of just a single code.

6.1.8 Embedded Declaration

These statements declare objects or actions for an embedded SQL application program:

DECLARE CURSOR
Dynamic DECLARE CURSOR
WHENEVER

These statements provide declarations for exception handling and for the declaration of cursors for use with the data manipulation statements.

6.1.9 Overview of Embedded SQL Control Flow

Figure 6-3 on page 32 shows a typical control flow for processing SQL statements through Embedded SQL. This is one example of a possible flow; other, more complex flows are also possible.

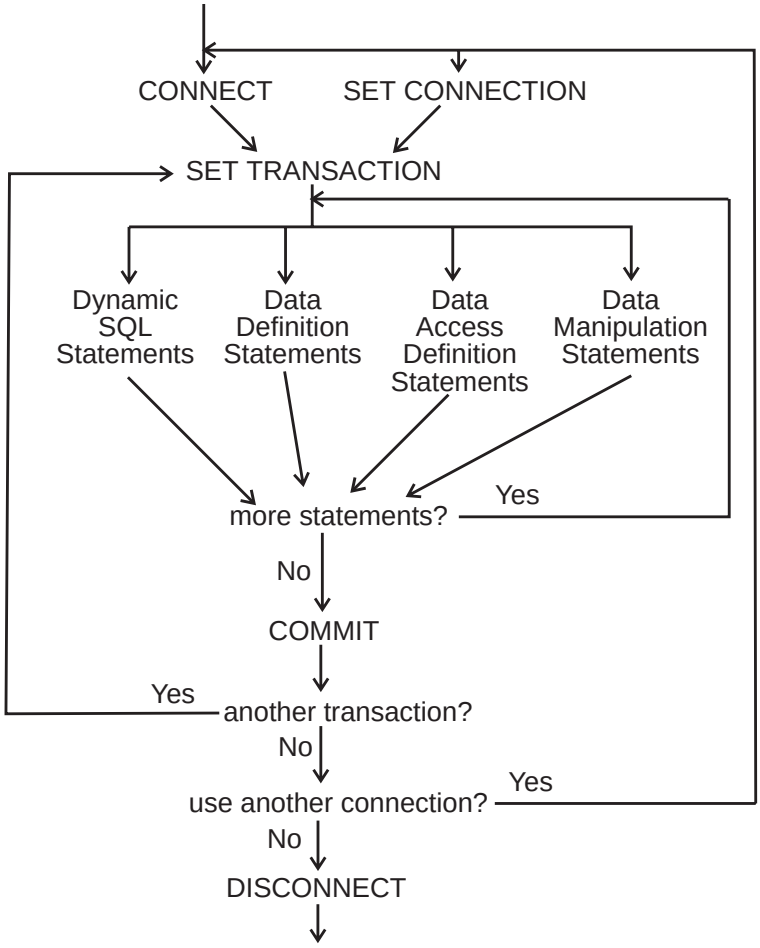


Figure 6-3 Embedded SQL Basic Control Flow

A CONNECT statement (either explicit or implicit) provides a connection with an SQL server. A SET TRANSACTION statement allows the setting of the characteristics of the following transaction; for example, whether the transaction is to be read-only or not.

One of the four classes of SQL statement shown may then be executed. A new transaction is started by the first SQL statement requiring that a transaction be active. The cursor and dynamic SQL statements have specific interrelationships; these are shown in the relevant section describing those statements.

If there are no more SQL statements to be executed in the current transaction, then a COMMIT requests that the transaction be terminated and data changes be made persistent. If data changes are to be undone, then a ROLLBACK is used instead.

If there are additional SQL statements to be executed for the current connection, thus requiring a new transaction, this is automatically started by the first SQL statement issued subsequently that requires a transaction to be active.

If there are no more SQL statements to be executed for the current connection, then either a DISCONNECT statement or a new CONNECT is done to establish a new connection for the subsequent transaction or a SET CONNECTION is done to reactivate a previous connection.

6.2 RDA Services

The RDA services can be classified as follows:

- Dialogue Management Services
- Transaction Management Services
- Control Services
- Resource Handling Services
- Database Language Services.

6.2.1 Dialogue Management Services

The RDA Dialogue Management services provide for the initialization and termination of an RDA dialogue. There must be an RDA dialogue active before any SQL operations can be processed by the RDA server. (There must also be an OSI association active before an RDA dialogue can be initialized.)

R-Initiate service — Initializes a new RDA dialogue between the RDA client and the RDA server. The RDA client can negotiate parameters such as the SQL level (conformance level) and any optional RDA services required.

R-Terminate service — Terminates an RDA dialogue in an orderly manner, without loss of information. An RDA dialogue cannot be terminated while there is an RDA transaction open.

These services support the SQL connection management statements. The SQL connection management statements are processed by the SQL client, and any RDA Dialogue Management services (or underlying association services) that are needed are then requested. Thus an SQL CONNECT statement may lead to an RDA R-Initialize, an SQL DISCONNECT statement may lead to an RDA R-Terminate, and an SQL SET CONNECTION statement may lead to an RDA R-Terminate followed by an RDA R-Initialize.

6.2.2 Transaction Management Services

The RDA Transaction Management services provide for the initiation and termination of transactions. These functions are used for one-phase commitment (RDA Basic application context), and enable distributed working without full distributed transaction processing (two-phase commitment). (X/Open RDA does not provide two-phase commitment.)

R-BeginTransaction service — Initiates a transaction.

R-Commit service — Terminates the current transaction and (if successful) makes persistent all changes made to data during this transaction. If the R-Commit is unsuccessful, the transaction is terminated but none of the changes made to data during this transaction are retained.

R-Rollback service — Terminates the current transaction and ensures that none of the changes made to data during this transaction are retained.

These services support the SQL transaction management statements. There is no explicit SQL statement to initiate a transaction; rather, execution of an SQL transaction-initiating statement when no SQL-transaction is currently active leads to an RDA R-BeginTransaction. Execution of an SQL COMMIT statement leads to an RDA R-Commit. Execution of an SQL ROLLBACK statement leads to an RDA R-Rollback.

6.2.3 Database Language Services

The RDA Database Language Services provide for the execution of database language (SQL) statements. (X/Open RDA has only one Database Language service because it does not support the RDA Stored Execution DBL services.

R-ExecuteDBL service

Requests the execution of a single SQL statement. The statement can be executed one or more times; if it is executed more than once then it can be either executed each time with the same set of parameters or executed each time with a different set of parameters.

6.2.4 Control Services

The RDA Control services provide for the control of outstanding RDA operations at the RDA server. Because execution of an RDA operation by an RDA server is not instantaneous and because of the asynchronous capability of the RDA Service, several RDA operations can be outstanding at any one time.

R-Cancel service — Requests cancellation of outstanding RDA operations. Whether or not those operations have actually been cancelled is indicated later by the response to those operations.

R-Status service — Queries the RDA server for the status of outstanding RDA operations.

6.2.5 Resource Handling Services

The RDA Resource Handling services control the availability of data resources.

R-Open service — Makes a data resource available for access in subsequent RDA service requests. Only one data resource may be open at a time. The RDA client provides data to authenticate the right to use the data resource (if required by the RDA server); and may specify parameters controlling its use, such as access mode (retrieval or update) and SQL level.

R-Close service — Terminates the availability of a data resource.

There are no SQL statements that correspond directly to RDA Resource Handling services. Rather, the RDA client must ensure that an R-Open request for the appropriate data resource has been issued before it accesses any SQL data in that data resource. Information identifying data resources is derived from explicit or implicit SQL CONNECT statements.

6.2.6 Overview of RDA Control Flow

Figure 6-4 on page 35 shows a typical control flow for processing SQL statements through RDA to a remote SQL server. This is one example of a possible flow; other, more complex flows are also possible.

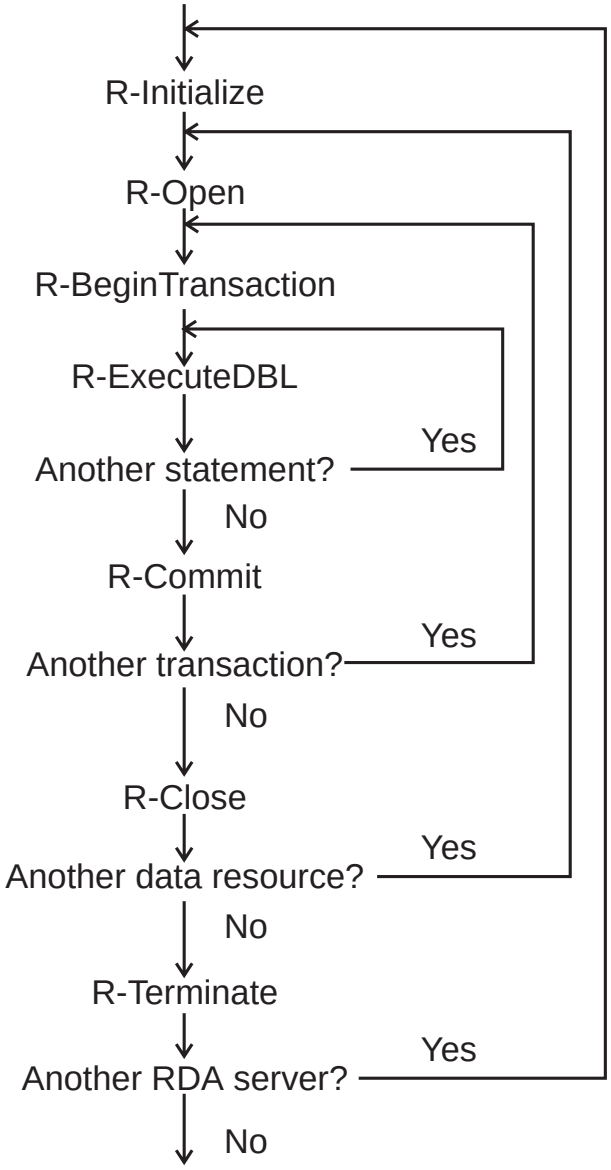


Figure 6-4 RDA Basic Control Flow

An R-Initialize creates an RDA dialogue. An R-Initialize results from either an (explicit or implicit) SQL CONNECT statement or an SQL SET CONNECTION statement referencing a previous SQL CONNECT statement.

An R-Open makes a data resource available. An R-Open results from an SQL CONNECT or SET CONNECTION statement, but is also required if a new data resource at the same RDA server is to be accessed.

An R-BeginTransaction opens a transaction. There is no explicit SQL statement to initiate a transaction; rather, an R-BeginTransaction results from execution of an SQL transaction-initiating statement when no SQL-transaction is currently active.

An R-ExecuteDBL causes execution of an SQL statement. Multiple R-ExecuteDBLs may be executed in a transaction.

If there are no more SQL statements to be executed in the current transaction, then an R-Commit requests that the transaction be terminated and data changes be made persistent. An R-Commit results from an SQL COMMIT statement. If data changes are to be undone, then an R-Rollback is used instead. An R-Rollback results from an SQL ROLLBACK statement.

If there are additional SQL statements to be executed for the current data resource, thus requiring a new transaction, then another R-BeginTransaction is issued.

If there are no more SQL statements to be executed for the current data resource, then an R-Close is issued. An R-Close results from either an SQL DISCONNECT statement or the end of execution of SQL statements, but is also required if a new data resource at the same RDA server is to be accessed.

If there are additional SQL statements to be executed for another data resource (but the same RDA server), then another R-Open is issued.

If there are no more SQL statements to be executed for this RDA server, then an R-Terminate is issued. An R-Terminate results from either an SQL DISCONNECT statement or the end of execution of SQL statements.

If there are additional SQL statements to be executed for another RDA server, then another R-Initialize is issued.

6.3 CLI Services

The SQL Call Level Interface services are provided by functions. There are two types of functions; the basic functions and the concise functions.

The concise functions are alternatives to the basic functions that provide a shorthand way of using certain functions or groups of functions with limited options. They are designed to simplify application coding and decrease the number of CLI functions in many typical cases.

Basic CLI functions may be divided into the following categories:

- connection control
- transaction control
- SQL statement execution
- descriptor access
- result acquisition
- diagnostics
- introspection
- resource control
- attribute manipulation
- function cancellation
- schema functions.

6.3.1 Connection Control

These functions provide control of the connections with SQL-servers:

Connect( )
Disconnect( )

These CLI functions have a direct one-to-one correspondence to the identically named SQL statements.

6.3.2 Transaction Control

This function provides transaction control in the absence of global Transaction Manager:

EndTran( )

This function supports SQL's native transaction management and enables distributed working without full Distributed TP support. It is functionally equivalent to the SQL COMMIT and ROLLBACK statements.

6.3.3 SQL Statement Execution

These functions provide for the preparation, parameter value specification and execution of SQL statements:

ExecDirect( )
Execute( )
GetCursorName( )
ParamData( )
Prepare( )
PutData( )
SetCursorName( )

These functions, together with those for Descriptor Access, provide much the same facilities as those of the Dynamic SQL statements.

6.3.4 Descriptor Access

These functions manage the values of descriptor fields:

CopyDesc( )
GetDescField( )
SetDescField( )

CLI descriptors are data structures containing information about either columns or dynamic parameters. They are analogous to SQL descriptor areas in SQL. The values of the descriptors may be queried, set and/or copied from one descriptor to another.

6.3.5 Result Acquisition

These functions control the access to data returned from an SQL statement execution:

CloseCursor( )
Fetch( )
FetchScroll( )
GetData( )

6.3.6 Diagnostics

This function provides error and diagnostic information:

GetDiagField( )

It is functionally equivalent to the SQL GET DIAGNOSTICS statement.

6.3.7 Introspection

These functions provide information about the characteristics and capabilities of both a CLI implementation and the servers enabled through that implementation:

DataSources( )
GetFunctions( )
GetInfo( )
GetTypeInfo( )

An application may use the introspection functions to find out the characteristics of both a CLI implementation and servers enabled through that implementation.

6.3.8 Resource Control

These functions control and manage resources used by the CLI:

AllocHandle( )
FreeHandle( )

Resources for the CLI are identified by handles. Associated with each handle is memory containing data structures which the implementation needs to manage the functions using the resource. The types of resources available are environment, connection, statement and descriptor.

6.3.9 Attribute Manipulation

These functions manage and control the attributes associated with resources:

GetConnectAttr( )
GetEnvAttr( )
GetStmtAttr( )
SetConnectAttr( )
SetEnvAttr( )
SetStmtAttr( )

Environments, connections and statements have attributes which may be queried and, in some cases, set to the required values.

6.3.10 Function Cancellation

This function provides for the cancellation of an outstanding CLI function such as a long running query:

Cancel( )

Execution of a CLI function is not instantaneous and, because the execution may be asynchronous, several queries may be outstanding at any one time. This function allows an outstanding query to be prematurely terminated.

6.3.11 Schema Functions

These functions provide a functional interface to a limited set of information taken from the Information Schema:

Columns( )
Languages( )
ServerInfo( )
SpecialColumns( )
Statistics( )
Tables( )

These functions provide an alternative way to retrieve frequently required information from the Information Schema to that of using normal SQL queries on the Information Schema tables themselves.

6.3.12 Overview of the CLI Control Flow

Figure 6-5 shows an overview of the basic control flow for an application program using the SQL CLI. This is merely one example of a possible flow; other, more complex, flows may be legitimately created.

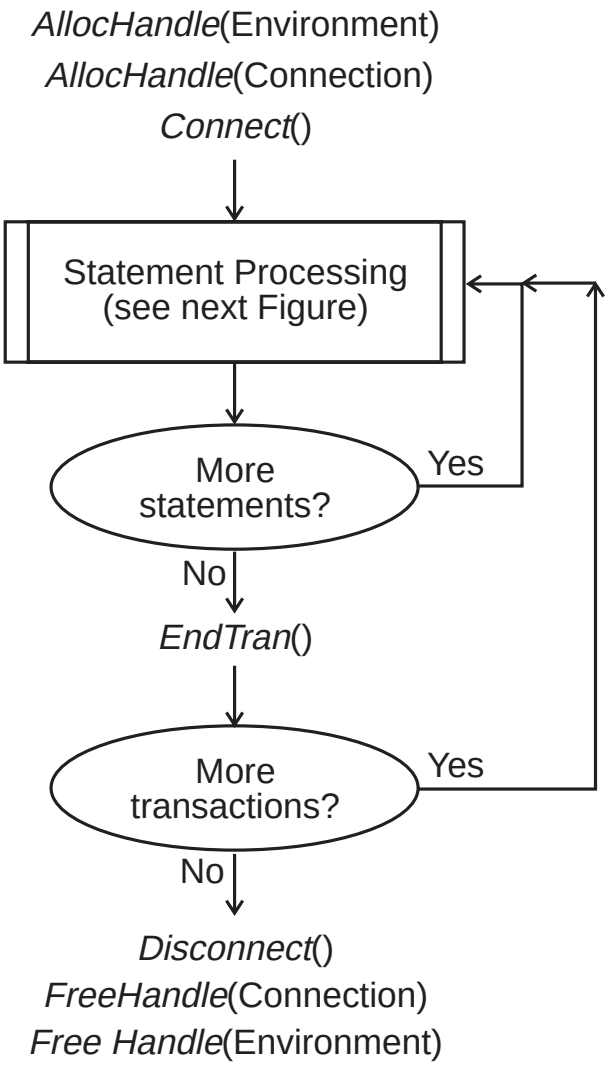


Figure 6-5 CLI Basic Control Flow — 1

The two *AllocHandle*( ) functions and the *Connect*( ) function establish an environment and a connection. Multiple environments may be permitted as may multiple connections within a single environment. Portable programs, however, should restrict themselves to a single environment.

Within a connection, multiple transactions may be processed, each terminated through the use of the *EndTran*( ) function. Naturally, within each transaction multiple statements can be executed.

Finally, the *Disconnect*( ) function and the two *FreeHandle*( ) functions disconnect the connection and release the resources associated with the connection and environment.

Figure 6-6 shows a typical control flow for processing SQL statements through the CLI. Again this is merely an example, other sequences are also valid.

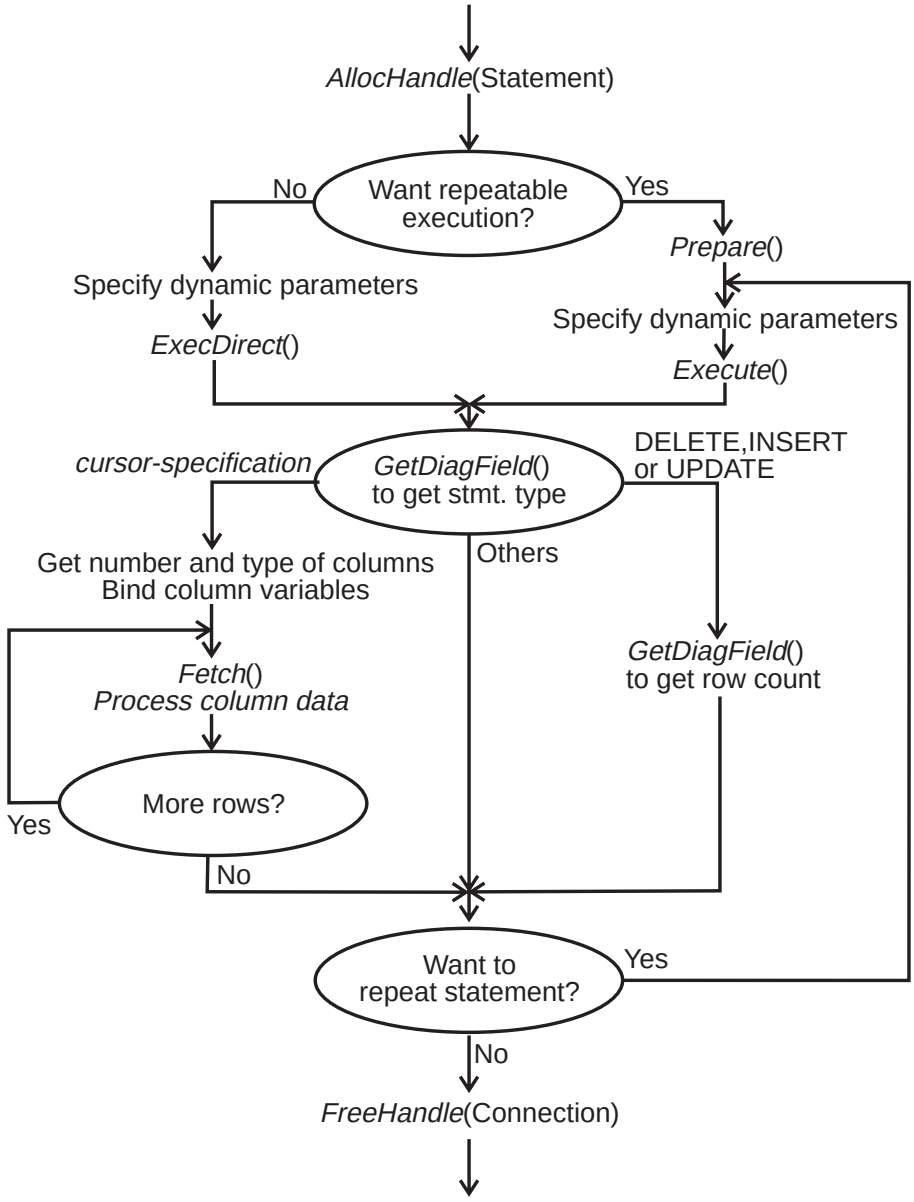


Figure 6-6 CLI Basic Control Flow — 2

The *AllocHandle*( ) function assigns resources for the execution of an SQL statement. The flow depends on whether a statement is to be repeatable or not. If not, then the *ExecDirect*( ) function may be used, otherwise the *Prepare*( ) and *Execute*( ) functions are used. Then, depending on the statement executed, information may be sought regarding the execution of the statement (for example, using the *GetDiagField*( )), or data may be retrieved using, for example, the *Fetch*( ) function.

When a statement is no longer needed its resources may be freed using the *FreeHandle*( ) function.

6.3.13 Concise CLI Functions

The concise functions and their underlying functions are shown in the following table.

Concise Function	Underlying Function
BindCol( )	*GetDescField*( )
BindParam( )	*GetDescField*( )
DescribeCol( )	*GetDescField*( )
GetDescRec( )	*GetDescField*( )
GetDiagRec( )	*GetDiagField*( )
NumResultCols( )	*GetDescField*( )
ReleaseEnv( )	*Disconnect*( ), *FreeHandle*( )
SetDescRec( )	*SetDescField*( )

Chapter 7

Instances of the X/Open Model

This chapter shows a number of different data management scenarios to illustrate the X/Open Model. These are by no means the only possibilities, but illustrate a number of common situations.

7.1 A Simple Local Database Management System

The configuration shown in Figure 7-1 is almost the simplest possible. Here we see an application program interfacing with an SQL client using both the Call Level Interface and the embedded SQL interface. This use of both the CLI and Embedded SQL is the only complication which makes this other than one of the simplest pictures possible. The SQL client communicates with a single SQL server that is located on the same Computer System as the client and so uses the Local Access Requester and Local Access Receiver.

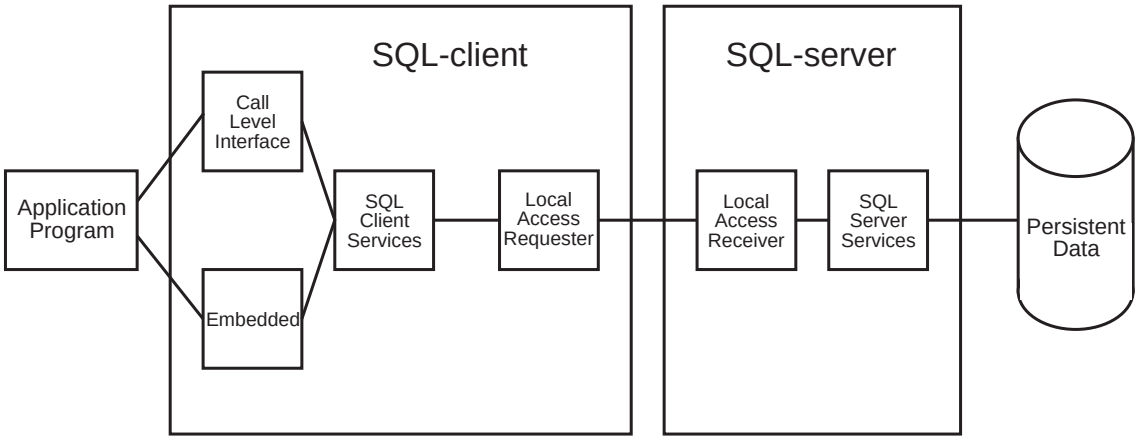


Figure 7-1 Local Database Configuration

7.2 A Simple Remote Database Management System

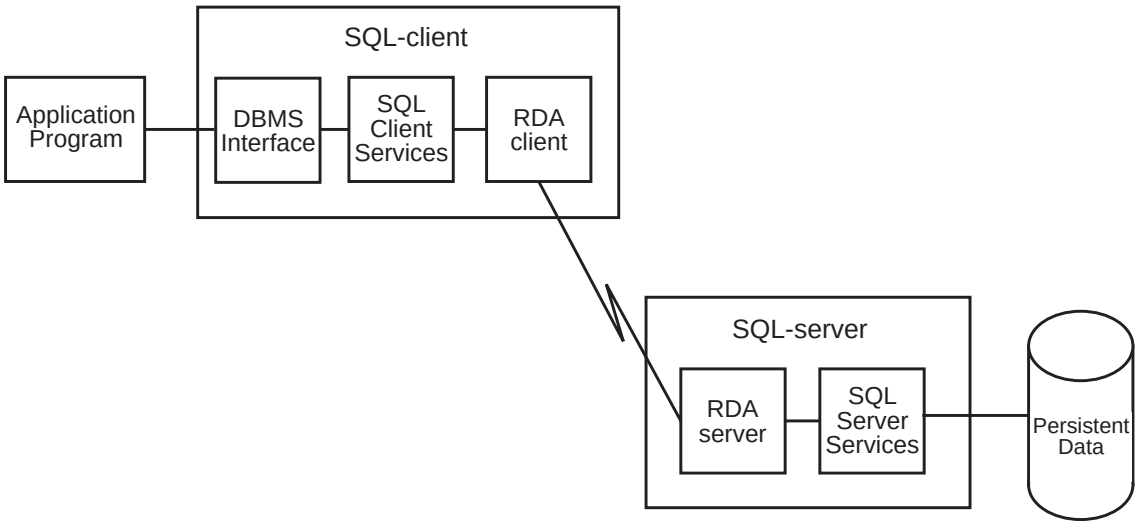


Figure 7-2 Simple Remote Database Configuration

The configuration shown in Figure 7-2 is the simplest possible remote configuration. Here we see an application program interfacing with an SQL client that communicates with a single SQL server which is located on a different Computer System to that of the client. Communication between the SQL client and SQL server is thus handled by an RDA client and RDA server pair. The DBMS Interface has not been differentiated. Thus the application program might be using either the Call Level Interface or Embedded SQL or both.

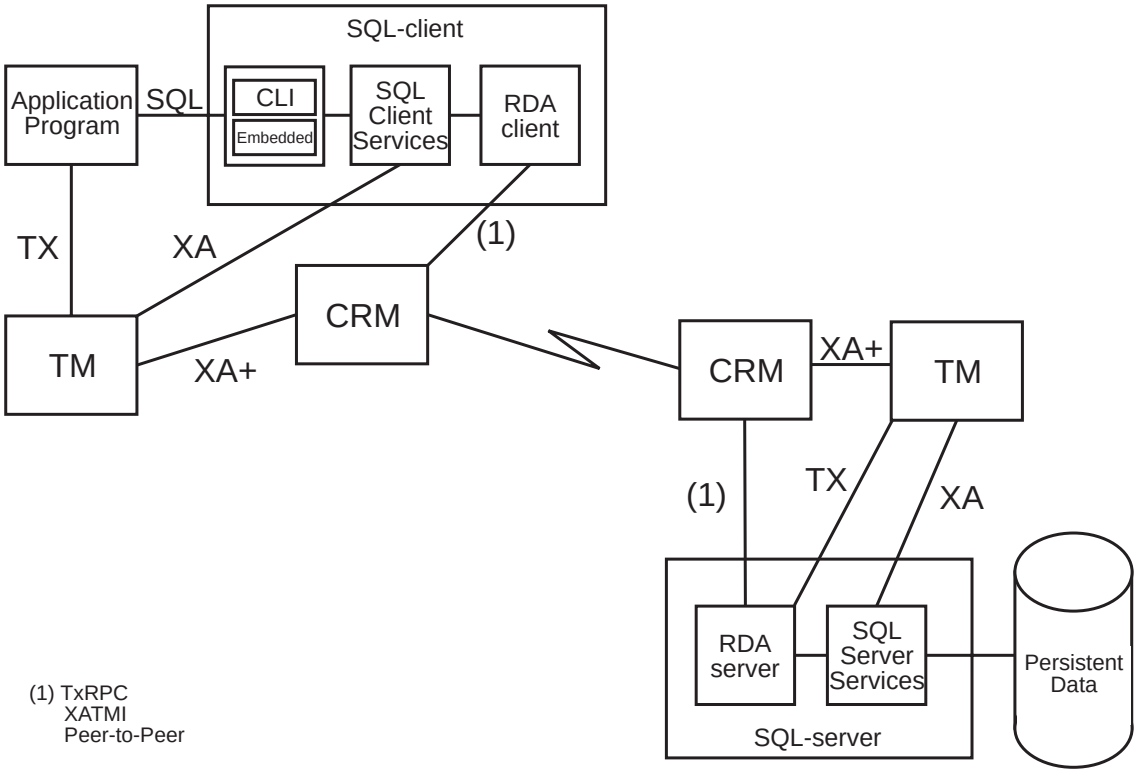


Figure 7-3 Simple Remote Database Configuration (with TP)

The same configuration as shown in Figure 7-2 on page 44 is shown again in Figure 7-3 but in this version the relationship with Distributed TP is clearly shown. The application program (AP) uses the transaction manager (TM) to control its transactions. It communicates with the TM through the TX interface. The AP communicates directly with the SQL client part of the database management system using the SQL language and one or both of the interfaces (CLI and embedded). The SQL client communicates with the TM using the XA interface. The TM communicates with the CRM using the XA+ interface to support global transaction information flow, and the RDA client assumes the role of the AP to communicate with a remote SQL server which plays the part of the AP on the remote side of the connection.

7.3 A Complex Database Management System Environment

The configuration shown in Figure 7-4 is a very complex configuration using both local and remote SQL servers.

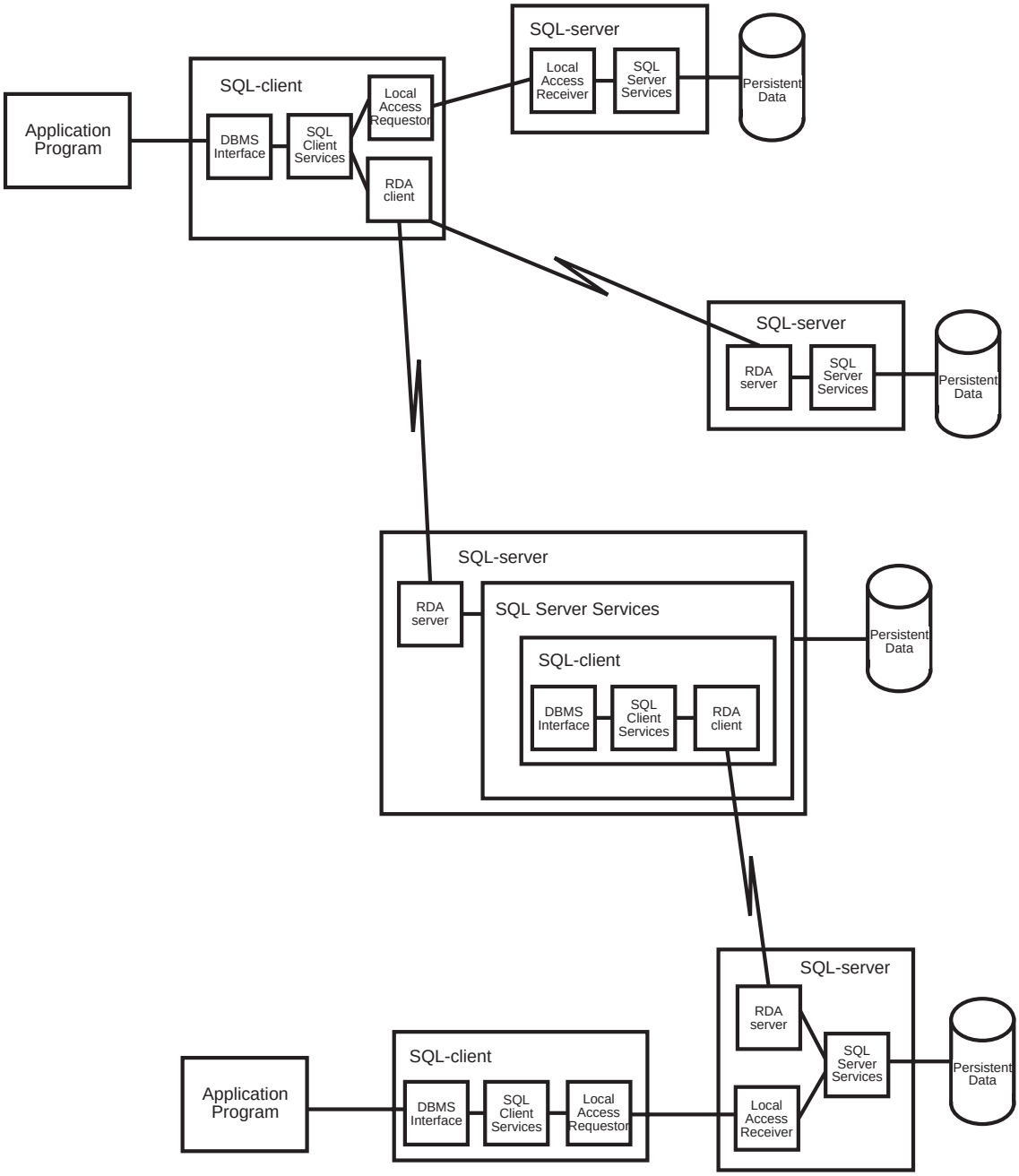


Figure 7-4 Complex Data Management Configuration

Here we see an application program interfacing with an SQL client that communicates with multiple SQL servers. Again the DBMS Interface has not been differentiated. The application program might be using either the Call Level Interface or Embedded SQL or both. One of the SQL servers is co-located with the SQL client; the other two are each located on a different

Computer System. Communication between the SQL client and the local SQL server is thus handled by the Local Access Requester/Local Access Receiver mechanism, whilst communication with the two remote SQL servers utilises a single RDA client and a pair of RDA servers.

To further complicate the picture, one of the remote servers itself plays the role of SQL client and accesses yet another SQL server at a still further remote location, also with the help of RDA. Thus, the box denoting the SQL Server Services encloses a complete SQL client box. The enclosed SQL client communicates with the final SQL server.

This final SQL server is shown as having its own local SQL client and application program.

Appendix A

RDA over OSI and Non-OSI Transport Providers

This appendix describes the X/Open **RDA over OSI and Non-OSI Transport Providers** specification. This document has been published as an X/Open Consortium Specification and is the result of a joint collaborative effort between the X/Open Data Management Working Group and the SQL Access Group.

The specification defines:

- Rules that permit X/Open RDA clients and servers to interoperate using other types of message transport providers than those specified by the OSI lower layer standards (the Transport Layer and below). Specifically, TCP/IP is supported.
- Recommendations for and guidance on the use of application programming interfaces (APIs) that can be employed in implementations of X/Open RDA.

The model of X/Open RDA for these capabilities is shown in Figure A-1. By making certain interfaces visible and standardisable, this architecture achieves the following objectives:

- It permits a single implementation of RDA to operate with either of two implementations of the OSI upper layers — full OSI or Minimal OSI.
- It permits these two implementations of the OSI upper layers to interoperate.
- It permits both of these two implementations of the OSI upper layers to operate with a variety of transport types.
- It permits a single implementation of RDA to operate with a variety of transport types.

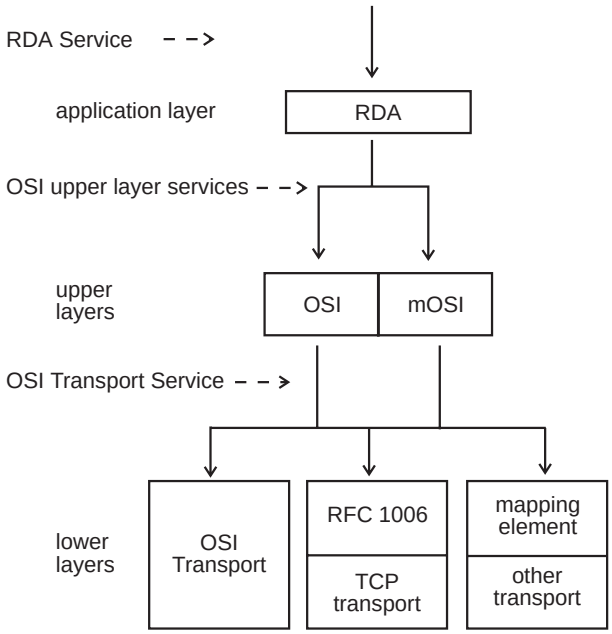


Figure A-1 Model of X/Open RDA

This model has three layers:

- the application layer, containing RDA
- the (OSI) upper layers, containing either the full OSI upper layer functionality or the mOSI subset (Miminal OSI) of the OSI upper layer functionality
- the lower layers, containing a transport provider.

In the model, the boundary between the upper layers and the lower layers is specified as the OSI Transport Service. Above this boundary, the standard OSI Application Layer and upper layers are required. Below this boundary, there is a different set of lower layers for each different type of transport provider. Each such transport provider must provide or emulate the services of the OSI Transport Service, and must specify the mapping from the OSI Transport Service to its particular transport service. Two such transport types are specified:

OSI transport — No mapping element is required, and the transport protocol is the OSI Transport Protocol.

TCP transport — The mapping element is the OSI Transport Service on top of the TCP (RFC 1006), and the transport protocol is the Transmission Control Protocol (TCP).

The model identifies two interfaces:

- The interface to an OSI upper layer services provider. Either of two APIs is recommended for this interface:
  - The X/Open **XAP** specification. This interface is recommended for vendors that wish to implement the full set of OSI upper layer services.
  - The X/Open **mOSI** specification. This interface is recommended for vendors that wish to implement only the minimum set of OSI upper layer services that is required for X/Open RDA.
- The interface to an OSI-compliant transport service-provider. The API recommended for this interface is the X/Open **Networking Services** specification. (See particularly its appendices for various transport providers.)

Index