U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology

FIPS

FIPS PUB 193

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

SQL ENVIRONMENTS

CATEGORY: SOFTWARE STANDARD SUBCATEGORY: DATABASE

1995 FEBRUARY 3

FIPS PUB 193

FIPS PUB 193

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

SQL ENVIRONMENTS

CATEGORY: SOFTWARE STANDARD SUBCATEGORY: DATABASE

Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-0001

Issued February 3, 1995

U.S. Department of Commerce
Ronald H. Brown, Secretary

Technology Administration
Mary L. Good, Under Secretary for Technology

National Institute of Standards
and Technology
Arati Prabhakar, Director

Foreword

The Federal Information Processing Standards Publication Series of the National Institute of Standards and Technology (NIST) is the official publication relating to standards and guidelines adopted and promulgated under the provisions of Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235. These mandates have given the Secretary of Commerce and NIST important responsibilities for improving the utilization and management of computer and related telecommunications systems in the Federal Government. The NIST, through its Computer Systems Laboratory, provides leadership, technical guidance, and coordination of Government efforts in the development of standards and guidelines in these areas.

Comments concerning Federal Information Processing Standards Publications are welcomed and should be addressed to the Director, Computer Systems Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899.

James H. Burrows, Director
Computer Systems Laboratory

Abstract

An SQL environment is an integrated data processing environment in which heterogeneous products, all supporting some aspect of the FIPS SQL standard (FIPS PUB 127), are able to communicate with one another and provide shared access to data and data operations and methods under appropriate security, integrity, and access control mechanisms. Some components in an SQL environment will be full-function SQL implementations that conform to an entire level of FIPS SQL and support all of its required clauses for schema definition, data manipulation, transaction management, integrity constraints, access control, and schema information. Other components in an SQL environment may be specialized data repositories, legacy databases, or graphical user interfaces and report writers, all of which support selected portions of the SQL standard and thereby provide a degree of integration between themselves and other products in the same SQL environment. This FIPS PUB is the beginning of a continuing effort to define appropriate conformance profiles that can be used by both vendors and users to specify exact requirements for how various products fit into an SQL environment. The emphasis in this first publication is to specify general purpose, SQL external repository interface (SQL/ERI) server profiles for non-SQL data repositories. Two major SQL/ERI Server Profiles are specified: read-only and read-write. To make it easier to specify integration among heterogeneous, non-SQL data models, this specification defines a new minimal level of the SQL language that can be supported by various non-SQL implementations.

Key words: CLI; client; conformance; database; ERI; Federal Information Processing Standard (FIPS); interface; Internet; ISP; multimedia; object; profile; PSM; RDA; relational; repository; server; standard; SQL; testing.

National Institute of Standards and Technology
FIPS PUB 193
72 pages (Feb. 3, 1995)
CODEN: FIPPAT

U.S. Government Printing Office
Washington: 1995

For sale by the National Technical Information Service
U.S. Department of Commerce
Springfield, VA 22161

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION 193

February 3, 1995

Announcing the Standard for SQL Environments

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology after approval by the Secretary of Commerce pursuant to Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235.

1. Name of Standard. SQL Environments (FIPS PUB 193).

2. Category of Standard. Software Standard, Database.

3. Explanation. An SQL environment is an integrated data processing environment in which heterogeneous products, all supporting some aspect of the FIPS SQL standard (FIPS PUB 127), are able to communicate with one another and provide shared access to data and data operations and methods under appropriate security, integrity, and access control mechanisms. Some components in an SQL environment will be full-function SQL implementations that conform to an entire level of FIPS SQL and support all of its required clauses for schema definition, data manipulation, transaction management, integrity constraints, access control, and schema information. Other components in an SQL environment may be specialized data repositories, legacy databases, or graphical user interfaces and report writers, all of which support selected portions of the SQL standard and thereby provide a degree of integration between themselves and other products in the same SQL environment.

This FIPS PUB is the beginning of a continuing effort to define appropriate conformance profiles that can be used by both vendors and users to specify exact requirements for how various products fit into an SQL environment. The emphasis in this first FIPS for SQL Environments is to specify general purpose, SQL external repository interface (SQL/ERI) profiles for non-SQL data repositories. These profiles specify how a subset of the SQL standard can be used to provide limited SQL access to legacy databases, or to support SQL gateways to specialized data managers such as Geographic Information Systems (GIS), full-text document management systems, or object database management systems. All of the profiles specified herein are for server-side products, that is, products that control persistent data and provide an interface for user access to that data. Subsequent versions of this FIPS PUB may specify SQL environment profiles for client-side products, that is, products that access data and then present that data in graphical or report-writer style to an end user, or process the data in some other way on behalf of the end user.

4. Approving Authority.

Secretary of Commerce.

5. Maintenance Agency.

Department of Commerce
National Institute of Standards and Technology
(Computer Systems Laboratory)

6. Cross Index.

- *Federal Information Resources Management Regulations (FIRMR)* subpart 201.20.303, Standards, and subpart 201.39.1002, Federal Standards, April 1992.

- FIPS PUB 127-2, *Federal Information Processing Standards Publication - Database Language SQL*, adoption of ANSI SQL (ANSI X3.135-1992) and ISO SQL (ISO/IEC 9075:1992) for Federal use, U.S. Department of Commerce, National Institute of Standards and Technology, June 2, 1993.

- ANSI/ISO/IEC 9579, *International Standard for Remote Database Access (RDA), Part 1: Generic RDA and Part 2: SQL Specialization*, ISO/IEC 9579-1:1993 and ISO/IEC 9579-2:1993, published December, 1993.

- ANSI/ISO/IEC DIS 9075-3, *(Draft) International Standard for Database Language SQL, Part 3: Call Level Interface (SQL / CLI)*, JTC1 Draft International Standard (DIS), document SC21 N9117, 13 October 1994.

- ANSI/ISO/IEC CD 9075-4, *(Draft) International Standard for Database Language SQL, Part 4: Persistent Stored Modules (SQL / PSM)*, JTC1 Committee Draft (CD), CD Ballot document SC21 N8897, August 1994.

7. Related Documents. SQL Environment specifications depend upon existing standards and stable specifications (see Cross Index above) and upon emerging SQL and SQL Multimedia standards. The following items identify formal ISO/IEC international standards projects for which preliminary specifications and base documents exist, but where the development effort has not yet reached a complete and stable stage (i.e. the Committe Draft (CD) stage). As these specifications mature and move through the standards processs, they can be referenced more reliably in procurement requirements.

(Working Draft) Database Language SQL (SQL3)
- Part 1: Framework
- Part 2: Foundation -- including Abstract Data Types and Object SQL
- Part 3: Call Level Interface -- extensions to ISO/IEC CD 9075-3 identified above.
- Part 4: Persistent Stored Modules -- extensions to ISO/IEC CD 9075-4 identified above.
- Part 5: Language Bindings -- extensions to the binding clauses of ISO/IEC 9075:1992.
- Part 6: SQL XA Interface Specialization -- to support X/Open XA-interface.

(Working Draft) SQL Multimedia (SQL/MM)

Part 1: Framework
Part 2: Full Text
Part 3: Spatial
Part 4: General Purpose Facilities
Other Parts: Reserved for other SQL/MM sub-projects with no current base document (e.g., images, photographs, motion pictures, sound, music, video, etc.).

For information on the current status of the above Working Drafts, contact NIST personnel working on SQL Standardization at 301-975-3251. For document references to the above and for additional related documents, see the References section of the SQL/ERI Server Profiles specification (attached).

8. Objective.

The primary objective of this FIPS PUB for SQL Environments is to specify SQL profiles that can be used by Federal departments and agencies to support integration of legacy databases and other non-SQL data repositories into an SQL environment. The intent is to provide a high level of control over a diverse collection of legacy or specialized data resources. An SQL environment allows an organization to obtain many of the advantages of SQL without requiring a large, complex, and error-prone conversion effort; instead, the organization can evolve, in a controlled manner, to a new integrated environment.

9. Applicability

This standard is applicable in any situation where it is desirable to integrate a client-side productivity tool or a server-side data repository into an SQL environment. It is a non-mandatory standard that may be invoked on a case-by-case basis subject to the integration objectives of the procuring department or agency. It is particularly suitable for specifying limited SQL interfaces to legacy databases or to specialized data repositories not under the control of a full-function SQL database management system. It can be used along with other procurement information to specify SQL interface requirements for a wide range of data management procurements.

One special area of application envisioned for this standard is Electronic Commerce, a National Challenge Application area of the National Information Infrastructure. The primary objective of Electronic Commerce is to integrate communications, data management, and security services in a distributed processing environment, thereby allowing business applications within different organizations to interoperate and exchange information without human intervention. At the data management level, electronic commerce requires a logically integrated database of diverse data stored in geographically separated data banks under the management and control of heterogeneous database management systems. An over-riding requirement is that these diverse data managers be able to communicate with one another and provide shared access to data and data operations and methods under appropriate security, integrity, and access control mechanisms. FIPS SQL provides a powerful database language for data definition, data manipulation, and integrity management to satisfy many of these requirements. It is unrealistic to expect that every data manager involved in electronic commerce will conform to even the Entry SQL level of the FIPS SQL standard; however, it is not unrealistic to require that they support a limited SQL interface, even a read-only interface, provided by one of the SQL/ERI Server profiles specified herein. New procurements to add components to the National Information Infrastructure, or to upgrade existing components, can define the necessary SQL schemas and point to appropriate SQL/ERI Server profiles as procurement requirements.

This standard may also be applicable, on a case-by-case basis, in many of the following areas:

Legacy databases
Full-Text document databases
Geographic Information Systems
Bibliographic information retrieval
Object database interfaces
Federal data distribution
Operating system file interface
Open system directory interface
Electronic mail repositories
CASE tool repositories
XBase repositories
C++ sequence class repositories
Object Request Broker interface repository
Real-time database interface
Internet file repositories

Further detail on each of these potential application areas can be found in Section 8, "Applicability", of the FIPS specification for SQL Environments.

10. Specifications. See the Specifications for SQL Environments - SQL External Repository Interface (SQL/ERI) - Server Profiles (attached).

11. Implementation. Implementation of this standard involves four areas of consideration: the effective date, acquisition of conforming implementations, interpretation, and validation.

11.1 Effective date. This publication is effective beginning February 3, 1995. Since it is a non-mandatory specification, based on the established FIPS SQL standard, and used at the discretion of individual Federal procurements, no transitional period or delayed effective date is necessary.

11.2 Acquisition. All conforming implementations of a specific SQL/ERI profile will support some aspects of the FIPS SQL standard. However, such implementations will not normally be full function database management systems and conformance will often be dependent upon SQL schema definitions and other requirements provided as part of each individual procurement. In most cases, a procurement will not be able to simply point to an SQL/ERI profile and demand conformance to it. Instead, successful procurements will normally use an appropriate SQL/ERI profile, together with an application-specific schema definition, as one aspect of overall procurement requirements. In many cases, vendors of products that provide a limited SQL interface will define their interfaces in terms of a fixed SQL schema definition. In those cases, procurements can point to the vendor-provided schema definition and to an appropriate SQL/ERI profile as a procurement requirement. In some cases, especially in those situations where schema definitions and requirements are not known in advance, a request for a proposal (RFP) may require that an SQL schema, and adherence to one of the SQL/ERI Server profiles, be presented as part of the response proposal.

11.3 Interpretation. NIST provides for the resolution of questions regarding specifications and requirements of the FIPS for SQL Environments, and issues official interpretations as needed. Procedures for interpretations are specified in FIPS PUB 29-3. All questions about the interpretation of FIPS SQL Environments should be addressed to:

Director
Computer Systems Laboratory
ATTN: SQL Environments
National Institute of Standards and Technology
Gaithersburg, MD 20899
Telephone: (301) 975-2833

11.4 Validation. Implementations of the FIPS for SQL Environments may be validated in accordance with NIST Computer Systems Laboratory (CSL) validation procedures for FIPS SQL (FIPS PUB 127). Recommended procurement terminology for validation of FIPS SQL is contained in the U.S. General Services Administration publication Federal ADP & Telecommunications Standards Index, Chapter 4 Part 2. This GSA publication provides terminology for three validation options: Delayed Validation, Prior Validation Testing, and Prior Validation. The agency may select the appropriate validation option and may specify appropriate time frames for validation and correction of nonconformities.

Implementations may be evaluated using the NIST SQL Test Suite, a suite of automated validation tests for SQL implementations. Although this test suite was designed to test conformance of full-function SQL database management systems, it can be modified to accommodate testing of SQL/ERI Server implementations. The results of validation testing by the SQL Testing Service are published on a quarterly basis in the Validated Products List, available from the National Technical Information Service (NTIS).

Current information about the NIST SQL Validation Service and the status of validation testing for SQL Environments is available from:

National Institute of Standards and Technology
Computer Systems Laboratory
Software Standards Validation Group
Building 225, Room A266
Gaithersburg, Maryland 20899
(301) 975-2490

12. Where to Obtain Copies. Copies of this publication are for sale by the National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161, telephone 703-487-4650. When ordering, refer to Federal Information Processing Standards Publication 193 (FIPSPUB193), SQL Environments. Payment may be made by check, money order, or deposit account.

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION 193

Specifications for

SQL Environments

SQL External Repository Interface (SQL/ERI)

Server Profiles

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Computer Systems Laboratory
Information Systems Engineering Division
Gaithersburg, MD 20899

Table of Contents

FIPS PUBLICATION 193

Specification for SQL Environments - SQL External Repository Interface (SQL/ERI) - Server Profiles

Abstract

An SQL environment is an integrated data processing environment in which heterogeneous products, all supporting some aspect of the FIPS SQL standard (FIPS PUB 127), are able to communicate with one another and provide shared access to data and data operations and methods under appropriate security, integrity, and access control mechanisms. Some components in an SQL environment will be full-function SQL implementations that conform to an entire level of FIPS SQL and support all of its required clauses for schema definition, data manipulation, transaction management, integrity constraints, access control, and schema information. Other components in an SQL environment may be specialized data repositories, legacy databases, or graphical user interfaces and report writers, all of which support selected portions of the SQL standard and thereby provide a degree of integration between themselves and other products in the same SQL environment. This FIPS PUB is the beginning of a continuing effort to define appropriate conformance profiles that can be used by both vendors and users to specify exact requirements for how various products fit into an SQL environment. The emphasis in this first publication is to specify general purpose, SQL external repository interface (SQL/ERI) server profiles for non-SQL data repositories. The SQL/ERI interface supports integration of heterogeneous, non-SQL data repositories into an SQL environment while retaining full use of the SQL language for user applications. All of the profiles specified herein are for server-side products, that is, products that control persistent data and provide a standard interface for accessing that data. Subsequent versions of this FIPS PUB may specify profiles for client-side products in an SQL environment, that is, products that access data and then present that data in graphical or report-writer style to an end user, or process the data in some other way on behalf of the end user. To make it easier to specify integration among heterogeneous, non-SQL data models, this specification defines a new minimal level of the SQL language that can be supported by various non-SQL implementations. Non-SQL data repositories, such as Geographic Information Systems (GIS), full-text document management systems, or object database management systems, may use this minimal level, or one of the other levels specified in FIPS SQL, to describe their capabilities as SQL/ERI Servers. Two major SQL/ERI Server profiles are specified: read-only and read-write. This specification may also be used as a starting point for defining International Standardized Profiles (ISPs) for SQL language access to non-SQL data repositories.

Keywords: (CLI; client; conformance; database; ERI; FIPS; interface; Internet; ISP; multimedia; object; profile; PSM; RDA; relational; repository; server; standard; SQL; testing)

Electronic Availability: An electronic version of this specification is available using Internet anonymous FTP protocols.

Internet Node:	speckle.ncsl.nist.gov	
User name:	ftp	
Password:	<YourName>@<YourInternetAddress>	
Change Directory to:	isowg3/FIPSdocs	
Get File:	fips193.ps	-- Postscript version

An ASCII text version of this document is also available in the same directory as above, but with file name "fips193.txt".

You will receive some sign-on messages. If these messages confuse your FTP client, you can turn them off when you sign-on again by preceding your password with a hyphen (-).

1. Introduction

An SQL environment is an integrated data processing environment in which heterogeneous products, all supporting some aspect of the FIPS SQL standard (FIPS PUB 127), are able to communicate with one another and provide shared access to data and data operations and methods under appropriate security, integrity, and access control mechanisms. Some components in an SQL environment will be full-function SQL implementations that conform to an entire level of FIPS SQL and support all of its required clauses for schema definition, data manipulation, transaction management, integrity constraints, access control, and schema information. Other components in an SQL environment may be specialized data repositories, legacy databases, or graphical user interfaces and report writers, all of which support selected portions of the SQL standard and thereby provide a degree of integration between themselves and other products in the same SQL environment. The intent is to provide a high level of control over a diverse collection of legacy or specialized data resources. An SQL environment allows an organization to obtain many of the advantages of SQL without requiring a large, complex, and error-prone conversion effort; instead, the organization can evolve, in a controlled manner, to a new integrated environment.

This FIPS PUB is the beginning of a continuing effort to define appropriate conformance profiles that can be used by both vendors and users to specify exact requirements for how various products fit into an SQL environment. The emphasis in this first specification is to specify general purpose, SQL external repository interface (SQL/ERI) profiles for non-SQL data repositories. These profiles specify how a subset of the SQL standard can be used to provide limited SQL access to legacy databases, or to support SQL gateways to specialized data managers such as Geographic Information Systems (GIS), full-text document management systems, or object database management systems. All of the profiles specified herein are for server-side products, that is, products that control persistent data and provide an interface for user access to that data. Subsequent versions of this FIPS PUB may specify SQL environment profiles for client-side products, that is, products that access data and then present that data in graphical or report-writer style to an end user, or process the data in some other way on behalf of the end user.

1.1 Database Language SQL

Database Language SQL is enjoying success as an effective International Standard for customers and implementors of full-function, SQL-compliant database management systems that support the relational data model. Many vendors have implemented an entire level of the SQL'92 standard [8] and offer products that conform to all of its clauses for schema definition, data manipulation, transaction management, integrity constraints, access control, and schema information. Other vendors have implemented selected portions of the SQL standard, most often read-only data retrieval or very restricted data manipulation, in order to provide SQL access to legacy databases or to support SQL gateways to specialized data managers.

The first SQL standard, in 1986, provided basic language constructs for defining and manipulating tables of data; a revision in 1989 added language extensions for referential integrity and generalized integrity constraints; and the most recent revision in 1992 provides new facilities for schema manipulation and data administration, as well as substantial enhancements for data definition and data manipulation. A companion standard for Remote Database Access (RDA) [9], completed in 1993, provides the basic services and protocols for SQL interoperability in a distributed, wide area client/server environment. A companion standard for an SQL Call Level Interface (SQL/CLI) [10], registered as a draft international standard (DIS) in October 1994, provides a language binding appropriate for third-party software developers in a local client/server environment. An extension to SQL for definition and invocation of persistent stored procedures and for SQL flow of control statements, named Persistent SQL Modules (SQL/PSM) [11] and registered as a draft standard in March 1994, permits user definition of procedural program blocks that can then be optimized at multiple SQL servers and invoked as needed, thereby reducing both processing time and communications volume. Features of the SQL'92 standard are discussed in References [1], [2], and [15]. Proposed features of the next SQL revision, often called SQL3, are discussed in [4], [5], and [23].

SQL is particularly appropriate for the definition and management of data that is structured into repeated occurrences having common data structure definitions. SQL provides a high-level query and update language for set-at-a-time retrieval and update operations, as well as required database management functions for schema and view definition, integrity constraints, schema manipulation, and access control. SQL provides a data manipulation language that is mathematically sound and based on a first-order predicate calculus. SQL is self-describing in the sense that all schema information is queryable through a set of catalog tables called the Information Schema.

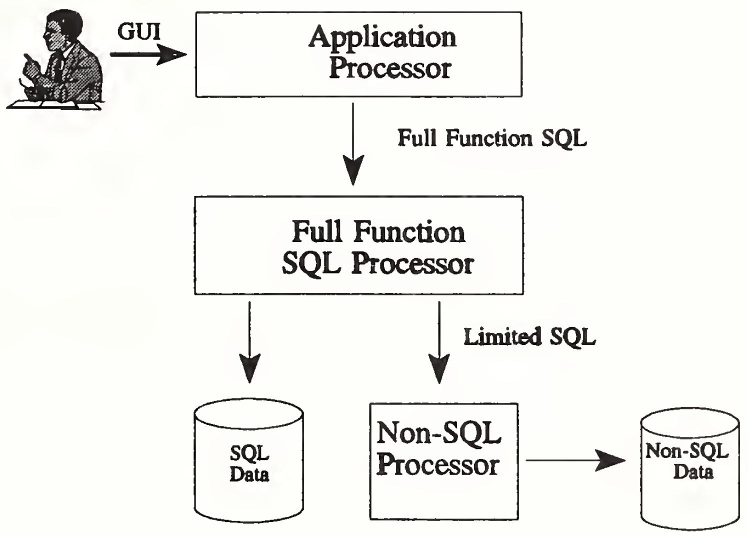
SQL is becoming the language of choice for many user productivity tools -- such tools communicate with a human user through a graphical user interface and then formulate SQL queries to communicate with underlying persistent data repositories. Formal language profiles for partial SQL conformance are necessary because the user productivity tools and the underlying data managers may be purchased at different times from different vendors and are unlikely to even know of one another's existence. A recognized profile specification will allow limited portability and interoperability, even in otherwise non-homogeneous environments.

1.2 SQL environment

Many applications require a logically integrated database of diverse data (e.g. documents, graphics, spatial data, alphanumeric records, complex objects, images, voice, video) stored in geographically separated data banks under the management and control of heterogeneous data management systems. An over-riding requirement is that these various data managers be able to communicate with one another and provide shared access to data and data operations and methods under appropriate security. integrity, and access control mechanisms. Much of this source data may be stored in simple file systems, legacy data management systems, or very specialized data repositories that satisfy only a small percentage of these data management requirements. The objective of an SQL environment is to logically integrate these diverse data repositories "as if" they were under the control of a single SQL data manager. User presentation tools, such as graphical user interfaces or report writers, can then use this SQL interface to collect data from various sources, merge it together under *ad hoc* join conditions, and present it to the user in a pleasing graphical format.

A properly functioning SQL environment will use the SQL language to describe this data using standardized facilities, integrate it into a single federated collection, enforce any integrity or access control constraints, and make it available as a logical whole to sophisticated user productivity or presentation tools. These client-side tools can then use the full power and flexibility of SQL for data retrieval and manipulation. The underlying data managers may implement non-relational data models and thus may have difficulty supporting SQL requirements for nested subqueries, multi-table joins, derived columns in a select list, referential integrity, or other relational model features. On the other hand, they may offer advanced features of other data models that are rarely supported by relational implementations. With emerging features in the SQL language for user-defined abstract data types (ADTs), stored procedures, encapsulation, polymorphism, and other object management facilities, these diverse data repositories can be described as specialized SQL repositories and accessed using already standardized SQL binding alternatives identified in Section 6. With this approach, SQL may prove to be as successful as an integrator of heterogeneous data repositories as it has been as a language interface to the relational data model. The SQL language can meet these integration objectives if non-SQL implementations provide a "simple" SQL interface to their data and services, and if full-function SQL implementations use that simple interface to provide full-function services to end user tools. This specification defines standard profiles for such "simple" SQL interfaces, thereby making it easier to specify and support the desired integration.

An SQL environment depends upon the data integration architecture presented in Section 2. A simplification of this architecture is given in the figure below. Components in the architecture consist of Application Processors, Full-Funtion SQL Processors, and Non-SQL Processors. The Application Processors represent client-side products that desire the ability to use the full power and flexibility of the SQL language when accessing data from a database. The Non-SQL Processors are server-side products representing data managers that control much of the data that is to be integrated and made available to the Application Processors. The Full-Function SQL Processors serve a dual role, both as server-side data managers and as "integrators" that make it possible for Application Processors to access data managed by the Non-SQL Processors in a standard manner. The interface between an Application Processor and an SQL Processor is a full-function SQL interface, whereas the interface between an SQL Processor and a Non-SQL Processor is one of the more limited SQL/ERI interfaces described in Section 3. All interfaces use one of the binding styles identified in Section 6. It is the integrator's role to provide access to all data as if it were managed by a full-function SQL processor.



SQL Environment

Section 4 describes existing conformance levels of the SQL language and then defines a new, minimal SQL language level that can be used to define conformance alternatives for SQL/ERI Servers. Section 5 identifies various higher level SQL features and data types that an SQL/ERI Server may support. In this way, an SQL/ERI Server can present the features of a different data model to an SQL application by describing them as SQL abstract data types, methods, procedures, or other callable routines. Section 7 specifies two major SQL/ERI Server profiles -- a read-only profile for static data repositories, and a read-write profile that allows SQL Update, Insert, and Delete statements. The read-write profile also provides an option that allows creation of SQL tables and views. Section 8 identifies a number of application areas for which SQL/ERI Server profiles may be applicable. Section 9 describes how the NIST SQL Test Suite serves as the basis for conformance testing of SQL/ERI Servers and Section 10 identifies some procurement considerations for users that intend to use this FIPS PUB for SQL Evironments to aide in the specification of procurement requirements.

The SQL/ERI profiles specified herein may be used by customers and vendors of non-SQL processors to validate claims of conformance for partial support of the SQL language. If these SQL/ERI Server profiles prove to be helpful for integrating non-SQL data repositories into SQL environments, then later versions of this FIPS PUB may specify profiles for SQL/ERI Clients as enhancements to full-function SQL implementations and profiles for other client-side products in an SQL environment. This specification may also be used as a starting point for defining International Standardized Profiles (ISPs) [14] for SQL language access to non-SQL data repositories.

2. Data Integration Architecture

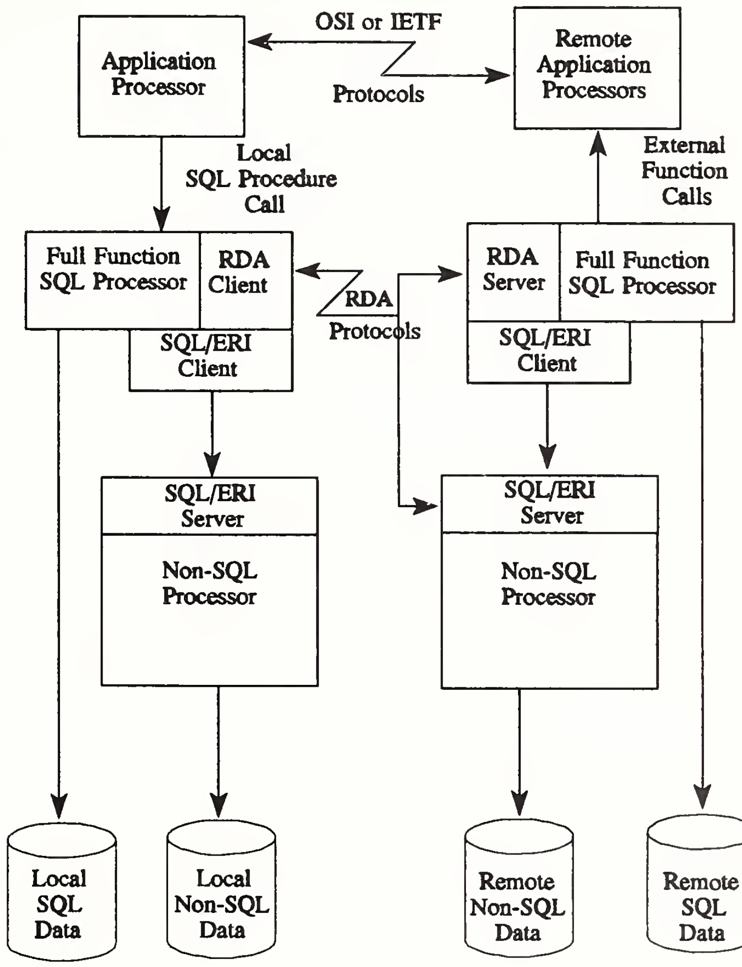
This FIPS for SQL Environments envisions an integrated data processing environment in which SQL and non-SQL processors are able to comunicate with each other and provide shared access to data and data operations and methods under appropriate security, integrity, and access control mechanisms. Application processors will then have protected access to all data using the full power and flexibility of Database Language SQL.

Standard communication among cooperating systems is possible at the present time using either OSI protocols [16] or Internet Society protocols [7]. Efforts are underway within both of these arenas to provide cross-protocol mappings for interoperability. Application services in both protocol environments provide for association control, file transfer, virtual terminal, and electronic mail. Future versions will contain extensions of these facilities as well as enhancements for remote database access (RDA), document management, and electronic data interchange (EDI). Near-term extensions to these protocols should make it possible for user-defined objects at various remote sites to communicate their existence and provide access to their methods to application processors. Objects at remote sites may be able to "show themselves" to users at local workstations by using emerging specifications and standards for graphical user interfaces.

The RDA component of both OSI and Internet Society communication protocols provides the basis of distributed access to remote data repositories and "standard" access to the data they manage. With implementation of an External Repository Interface (ERI), discussed in Section 3 below, it is possible for non-SQL data repositories to be "self-describing" in terms of SQL facilities so that they can be accessed and manipulated by all other sites using standard SQL language and RDA protocols. With longer-term emerging data management standards that support object-oriented and knowledge-based features, an ERI interface can evolve into a "seamless" integration of complex, structured data and supporting application services.

Begin with an Application Processor that wishes to communicate with and access data at a number of different data repositories, some local and some remote. The Application Processor could use existing communication protocols to connect to external processes or transfer files, but it would prefer not to have to manage its own communications links or worry about integrity, access control, remote transactions, or any number of different data manipulation functions; instead, it would rather communicate with a single, "familiar" data manager for both schema data and actual data occurrences. The "familiar" data manager could then connect itself to remote sites and access the desired data and data definitions, returning them to the accessing processor in a standard format. A remote object would still be able to use windowing protocols to "show itself" to the accessing process or use file transfer protocols to transfer objects or object definitions not under the control of the communicating data managers.

This architecture assumes the existence of any number of heterogeneous data repositories, some at the local site and some at distributed sites. It also assumes a full-function SQL processor at all sites, but not necessarily as the manager of the most important data. The non-SQL processors may control the maps, documents, graphics images, or complex engineering structures that the Application Processor wishes to access. The local SQL processor conforms to Database Language SQL and has two integrated client components, one conforming to the RDA/SQL Specialization and one conforming to the SQL/ERI interface proposed in Section 3 of this specification. Communications among the three SQL components are likely to be proprietary. The local site may have any number of non-SQL data repositories each controlled by a non-SQL Processor having a component that conforms to the SQL/ERI interface. Communications among the internal components of the non-SQL Processor are also proprietary. The local site has a proprietary local procedure calling mechanism and a proprietary local inter-process communications capability. Using these proprietary mechanisms and one of the standard local binding styles identified in Section 6 (e.g. SQL/CLI), the Application Processor issues standard SQL calls to the local full-function SQL processor, and the SQL/ERI Client component of the SQL processor is able to communicate, using an ERI specified subset of standard SQL, with the SQL/ERI Server of the non-SQL Processor.



Data Integration Architecture

The local site is connected to one or more remote sites via a standard OSI or Internet communications network ([16], [7]) that allows "messages" or "calls" to be exchanged among processes. Some messages may be sent directly from the Application Processor to processes or file stores at the remote site, but ideally, some local repository manager makes a connection and sends messages on behalf of the Application Processor. The Generic RDA and RDA/SQL Specialization standards [9] specify protocols that allow the RDA Client component of the local SQL processor to send SQL statements to the RDA Server component of a remote SQL processor, or the SQL/ERI Server component of any Non-SQL processor, and receive data in return. All protocols and data formats are defined in the RDA standards and are transmitted as ASN.1 (ISO 8825) packages. If the Application Processor is operating, interactively, on behalf of a human user, then any of the data repositories may use a local graphical user interface (GUI), or non-local windowing protocols, to present status information or a "menu of choices" to the human user. In this way an interactive "browsing" or "navigational" capability is provided to the human user without losing the standard RDA/SQL communications used by the non-human processors.

At the remote site there exists a full-function SQL Processor as well as any number of non-SQL Processors. Components of the SQL Processor conform to the SQL and RDA standards, and satisfy the proposed SQL/ERI Client requirements. Each non-SQL Processor has a component that conforms to the SQL/ERI Server specification. The remote site handles internal communications and procedure calls in the same proprietary manner as does the local site.

At the present time the RDA standards specify interchange protocols for transmitting records of data from a server site to a client site, provided that the data items in the records are either numbers or character strings. Near term RDA follow-on specifications will extend the data types handled to all of those specified in the SQL'92 standard [8], i.e. fixed and variable length character strings, fixed and variable length bit strings, fixed and floating point numerics, dates, times, timestamps, and intervals. Later RDA follow-on specifications will provide interchange mechanisms, in terms of ASN.1 elements, for the user defined abstract data types (ADTs) specified in the emerging SQL3 working draft [12]. RDA protocols do not by themselves provide interchange mechanisms for other data objects, so interchange standards for images, motion pictures, maps, topologies, or other complex objects will remain critical for transmitting object instances among various sites.

SQL and RDA provide the basis for standardized communication. An SQL external repository interface (SQL/ERI) makes it possible for non-SQL data repositories to share their data with user applications. With emerging SQL3 enhancements for object-oriented and knowledge-based data management and emerging RDA extensions for distributed database, the ERI can evolve to support "seamless" data integration.

3. SQL External Repository Interface (SQL/ERI)

Applications require access to multiple data repositories, many of which are managed by non-SQL processors. It is not unusual for applications to require data from the operating system, from graphics repositories, from CD-ROM's, from CAD/CAM databases, or from libraries of cataloged data. From a user's perspective, it is unrealistic to expect every data repository to be able to handle even the lowest "Entry SQL" queries. For example, who would expect an electronic mail system to handle SQL joins and subqueries over its message headers? Yet, every e-mail system is a data repository with information that applications sometimes require. What is needed is an interface specification that enables a non-SQL data repository to make certain external views available to SQL processors and for those SQL processors to, in turn, allow the full power and flexibility of the SQL query language over those views to the end user. It makes sense to specify a "client" and a "server" interface to external repositories so that non-SQL systems can act as servers to SQL requests for data. It makes sense to develop the conformance requirements needed for non-SQL systems to provide SQL views of their data and for SQL systems to provide full function SQL operations over that data to SQL users.

This interface is defined to be the SQL External Repository Interface (SQL/ERI). It consists of a "Server" part and a "Client" part. Non-SQL systems may claim conformance as SQL/ERI Servers and full-function SQL systems may claim conformance as SQL/ERI Clients. This first FIPS PUB for SQL Environments only addresses conformance criteria for SQL/ERI Servers; subsequent versions may address conformance criteria for SQL/ERI Clients. A wide range of non-SQL products and services might be able to claim conformance as SQL/ERI Servers. They could provide high level abstract data types with application-specific methods and operations. They would be required to evaluate "simple" SQL queries over individual tables defined in the schema. The exact meaning of "simple" is specified in the SQL/ERI profile specifications at different levels of service. The SQL processor can then think of the external repository as an SQL-environment that can be connected to, but that can only respond to whatever SQL statements are specified for that level of service.

If an SQL system claims conformance as an SQL/ERI Client, then it agrees to provide SQL functionality, at whatever level of the SQL standard it conforms to, over any table provided by an SQL/ERI Server. This may require that the SQL system automatically create a temporary table whenever the external view is referenced in a query, and then populate that table using the limited capabilities provided by the "server" interface so that it can guarantee the ability to perform nested queries, or searched updates and deletes, or recursive queries, or whatever is requested by its application.

With the SQL/ERI "client" and "server" definitions, non-SQL systems would be able to provide services to SQL-based applications even though they might not be able to provide the expected query flexibility, access control, concurrency control, or updatability required of a full-function SQL data manager. Full-function SQL processors could provide these expected data management facilities and, in addition, provide user access to data repositories not otherwise accessible via the SQL language. Section 2 describes how SQL/ERI profiles might be used to provide uniform and integrated application access to both SQL and non-SQL data at local and remote sites.

The SQL/ERI profile specifications provide several different conformance levels for non-SQL systems. A conforming SQL/ERI server is required to be "self-describing" as if it were a separate SQL-environment. It is required to supply an SQL Information Schema describing all available tables and the equivalent SQL data types for its columns. If the ERI Server provides new abstract data types not defined in the SQL standard, then it is also required to provide an SQL ADT interface definition as specified in the emerging SQL3 standard [12].

What is needed to make the above scenario feasible is an SQL/ERI Server profile, so that these non-SQL data repositories can provide a simple, external interface, accessible from full-function SQL systems. Sophisticated applications can then be built without the need to "understand" the non-standard data access methods unique to each repository. Instead, full-function SQL systems could be used as intermediaries. The SQL "client" could connect itself to the non-SQL "server" using the standard SQL/ERI interface; the application could then use the full power and flexibility of the SQL data manipulation language, as well as the system provided special access methods, to select and mange the data as if it were maintained in an SQL database. Section 7 of this FIPS PUB provides the necessary SQL/ERI Server profiles to get this integration scenario started.

4. SQL/ERI Leveling Rules

The SQL'92 standard [8] specifies three levels of conformance for SQL language and SQL implementations: Entry SQL, Intermediate SQL, and Full SQL. In addition, FIPS SQL [3] defines a fourth level of conformance, called Transitional SQL, approximately halfway between Entry SQL and Intermediate SQL. FIPS Transitional SQL is intended to provide a common, near-term goal for SQL implementations that already have a number of features beyond Entry SQL. It is intended for use in U.S. federal government procurements in the interim period before Intermediate SQL implementations are readily available. All of these existing SQL conformance levels require the facilities of a full-function SQL processor, i.e. schema definition, data manipulation, transaction management, and access control.

New conformance alternatives are needed for non-SQL processors that wish to claim conformance to only a portion of the SQL language. Such processors may be able to provide very sophisticated data retrieval capabilities, but may not be able to allow update of data instances or creation of new schema objects. Since existing SQL levels cut across both the schema definition and data manipulation facilities in the SQL standard, it is necessary to consider each SQL level separately as applied to schema definition or data manipulation.

Consider the SQL leveling rules separately for schema definition and data manipulation. Use the term Schema Definition Language (SDL) to identify SQL language features defined in Clause 11, "Schema definition and manipulation", in the SQL'92 standard, and use the term Data Manipulation Language (DML) to identify SQL language features defined in Clause 13, "Data Manipulation". One is then able to discuss the following alternatives for partial support of the SQL language:

Entry DML	Entry SDL
Transitional DML	Transitional SDL
Intermediate DML	Intermediate SDL
Full DML	Full SDL

There is an additional requirement to specify new Minimal DML and Minimal SDL levels to be used exclusively in the definition of SQL/ERI Server profiles. These Minimal definitions are intended for use only by non-SQL processors and cannot be used to claim conformance to the SQL standard as an SQL processor. Minimal DML will support SQL operations on a single table, with no joins and no subqueries, and with severe limitations on derived columns and set functions. Minimal SDL will support specification of only the simplest views and the simplest SQL tables, using only character string, integer, decimal, and real data types, with no table constraints, with only very limited column constraints, and possibly no support for null values.

Levels of conformance in the SQL standard are specified by Leveling Rules in each clause of the specification. Using the style of the SQL standard, the following subsections specify restrictions that apply for Minimal SDL and Minimal DML in addition to any restrictions for Entry SQL. All Clause and Subclause references, and all syntactic terms delimited by angle brackets (i.e. <...>) are from SQL'92 [8].

4.1 Minimal Schema Definition Language

1. A <schema element> contained in a <schema definition> shall be a <table definition> or a <view definition>.

2. A <table element> contained in a <table definition> shall be a <column definition>.

3. A <column constraint> shall not be a <unique specification>, a <references specification>, or a <check constraint definition>; thus a <column constraint> may only specify NOT NULL.

4. In some cases, an SQL/ERI Server implementation at the Minimal SDL level or below may choose not to provide support for SQL null values; if every column of every accessible table is constrained to be NOT NULL, then the implementation may require that every <column definition> in a new <table definition> have an explicit or implicit NOT NULL constraint.

5. The <data type> of a <column definition> shall not specify NUMERIC, FLOAT, or DOUBLE PRECISION; thus a <column definition> may only specify DECIMAL, REAL, INTEGER, SMALLINT, and fixed length CHARACTER string <data type>s.

6. A <view definition> shall not specify WITH CHECK OPTION.

7. The <query expression> contained in a <view definition> shall satisfy the restrictions specified by the Minimal Data Manipulation Language leveling rules below.

4.2 Minimal Data Manipulation Language

1. A <query expression> shall be a <query specification>.

2. A <derived column> in the <select list> of a <query specification> shall be a <value expression primary> that is either a <column reference> or a <set function specification>, and the <derived column> shall not contain an <as clause>.

3. A <set function specification> that is a <derived column> in the <select list> of a <query specification> shall be either COUNT(*) or a <general set function> whose directly contained <value expression> is a <column reference>.

4. A <table expression> shall not contain a <group by clause> or a <having clause>.

5. The <from clause> contained in a <table expression> shall contain exactly one <table reference>, and that <table reference> shall be a single <table name> without an associated <correlation name>. A <table name> may be qualified to include a <schema name>.

6. A <search condition> contained in an <SQL data statement> shall not contain any <subquery>. Any <predicate> contained in a <search condition> shall be a <comparison predicate> without subqueries, a <between predicate>, a <like predicate>, a <null predicate>, or an <in predicate> whose <in predicate value> is a parenthesized list of <value specification>s.

7. A <row value constructor> contained in any <predicate> shall have exactly one <row value constructor element> that is a <value expression>.

8. A <value expression> in a <search condition> shall be either a <numeric value expression> or a <string value expression> that is a <character primary>.

9. A <value expression primary> in a <search condition> shall be either a <column reference> or an <unsigned value specification>; thus it may not be a <set function specification> or a <scalar subquery>.

10. A <numeric primary> shall not be a <numeric value function>.

11. A <character primary> shall not be a <character value function>.

12. A <sort key> in a <declare cursor> shall be a <column name>; thus it may not be an <unsigned integer>.

Note: Leveling Rule 2a of Subclause 13.8, "<insert statement>", is incorrect in that it should also allow a <null specification>. This is corrected in SQL Technical Corrigendum 1 [20].

5. Optional Extensions

An SQL/ERI Server will often support additional data types and SQL language facilities beyond those specified for the given level of service. This section identifies features in the SQL'92 standard, and emerging features in the SQL3 and SQL/MM specifications, that an SQL/ERI Server may support. Each of these items is an optional indication that must be explicitly declared by an SQL/ERI Server implementation before an application program can rely on its existence.

An SQL/ERI Server that supports a Read-Only interface will support only the read-only aspects of each feature. Thus a non-SQL implementation may be able to define itself to an SQL client using very complex abstract data types (ADTs) and methods specified in SQL3, even though it does not allow creation or modification of such types. The SQL/ERI server will declare the accessing "signature" of such facilities in the Information Schema, so that an application can use the SQL/PSM routine and procedure calling mechanisms to access the data. This is how the very specialized data managers such as document management systems, geographic information systems, or CAD/CAM systems may make their specialized features available to SQL applications.

5.1 SQL'92 features

The SQL conformance levels defined in Section 4 of this specification identify a broad level of capability for conforming SQL/ERI servers. In addition, it is sometimes desirable to identify other features specified in the SQL'92 standard [8] as either offered by a product or required by a specific procurement. Section 14 of FIPS SQL [3] identifies 83 features of the SQL'92 standard beyond the Entry SQL requirements. An SQL/ERI Server could identify which features are supported beyond its declared level of service by implementing the SQL_FEATURES table as specified in Section 15 of FIPS SQL. Implementation of the SQL_FEATURES table is a requirement for all SQL/ERI Servers that claim a base level of SQL data manipulation language support at the Intermediate DML level or above. A procurement could also use this list to identify, unambiguously, those SQL features beyond the identified conformance level that are either required or desirable for that procurement.

FIPS SQL also identifies default minimum requirements for the precision, size, or number of occurrences of database constructs (see section 16.6 of [3]). Unless otherwise specified in a procurement, the Entry Value sizing limits apply to all Entry SQL or Transitional SQL features and the Intermediate Value sizing limits apply to all Intermediate SQL or Full SQL features. An SQL/ERI Server could identify its own sizing limits by implementing the SQL_SIZING table as specified in Section 15 of FIPS SQL. Implementation of the SQL_SIZING table is a requirement for all SQL/ERI Servers that claim a base level of SQL data manipulation language support at the Intermediate DML level or above. A procurement is responsible for identifying its own sizing limits on all required features, but in the absence of an explicit declaration, the default minimum limits apply for that procurement.

5.2 Stored procedures and callable routines (SQL/PSM)

An emerging new part of the SQL standard for Persistent Stored Modules (SQL/PSM) was registered as an ISO/IEC Committee Draft (CD) in March 1994 (see [11]). Although this specification will not reach formal standardization until at least late 1995 or early 1996, it should be sufficiently complete and stable to justify its careful use in procurements before that date. The intent of SQL/PSM is to make SQL a computationally complete programming language with variables, procedures, functions, and flow-of-control statements. In particular, packages of SQL procedures (i.e., modules) may be stored at server nodes in a communications network with only a procedure call needed to invoke a desired action. The advantage is that modules may be stored persistently in the database, subject to SQL access control and integrity management, thereby allowing a reduction of comunications overhead along with optimization and other performance efficiencies at the server site.

A major advantage of the SQL/PSM is that non-SQL data managers will be able to present their special features to applications in an SQL environment as callable SQL functions or procedures. The only requirement is that the calling syntax be standard SQL syntax as far as the calling application is concerned and that the parameters be defined as SQL data types. The content body of the functions and procedures may not be visible to the end user and thus may be implementation-dependent. The SQL/ERI Server can use the SQL Information Schema catalog tables to make known to the users exactly which functions and procedures are available for their use. The "signature" of such routines will also be available from the Information Schema.

5.3 SQL multimedia class library (SQL/MM)

A new ISO/IEC international standardization project for development of an SQL class library for multimedia applications was approved in early 1993. This new standardization activity, named SQL Multimedia (SQL/MM), will specify packages of SQL abstract data type (ADT) definitions using the facilities for ADT specification and invocation provided in the emerging SQL3 specification [12]. SQL/MM intends to standardize class libraries for science and engineering, full-text and document processing, and methods for the management of multimedia objects such as image, sound, animation, music, and video. It will likely provide an SQL language binding for multimedia objects defined by other JTC1 standardization bodies (e.g. SC18 for documents, SC24 for images, and SC29 for photographs and motion pictures).

The project plan for SQL/MM indicates that it will be a multi-part standard consisting of an evolving number of parts. Part 1 will be a Framework that specifies how the other parts are to be constructed. Each of the other parts will be devoted to a specific SQL application package. Even though this project is just getting started, initial base documents exist for Part 1: Framework, Part 2: Full Text, Part 3: Spatial, and Part 4: General Purpose Facilities (see [13]). As the different components of the SQL/MM specification reach CD and DIS stability, an SQL/ERI Server could claim support for specific features.

5.4 Abstract data types and methods

The emerging SQL3 specification contains a number of data abstraction facilities, including user-defined data types and methods. For example, see Clauses 4.11, "Abstract data types", 11.47, "<abstract data type definition>", and 11.48, "<abstract data type body>", of the August, 1994, version of [12]. If data abstraction is an inherent requirement of an SQL/ERI Server, then it could define its Abstract Data Types and make them available to SQL applications using these definitional mechanisms. As the data abstraction facilities of the SQL3 specification reach CD and DIS stability, an SQL/ERI Server could use them with more confidence to permanently define its abstract data types and methods.

A major advantage of these ADT features combined with the SQL/PSM identified above is that non-SQL data managers will be able to present their application-specific ADTs to applications in an SQL environment. The signature of such ADTs would be available in the Information Schema provided by each SQL/ERI Server, and the special methods on each ADT would be callable as SQL functions or procedures. The only requirement is that the calling syntax be standard SQL syntax as far as the calling application is concerned and that the parameters be SQL data types or known ADT types provided by the SQL/ERI Server. The abstract data type body of usable ADTs may not be visible to the end user and thus may be implementation-dependent. The SQL/ERI Server can use the SQL Information Schema catalog tables to make known to the users exactly which ADTs and associated methods are available for their use.

5.5 Object data management

The emerging SQL3 specification contains facilities for defining and referencing object identifiers. For example, see Clauses 4.10 "Object identifier", 11.48 "<abstract data type body>" WITH OID option, and 6.12 "<OID value function>", of the August, 1994, version of [12]. If object identity is an inherent requirement of an SQL/ERI Server, then it could define its Abstract Data Types with Object Identifiers (OID) and make them available to SQL applications using these definitional mechanisms. In this manner, object database management systems and specialized object repositories could make their features and facilities available to an SQL environment. As the different OID facilities of the SQL3 specification reach CD and DIS stability, an object-oriented SQL/ERI Server could use them with more confidence to permanently define its objects and their methods.

5.6 Encompassing transactions

Database Language SQL [8] already supports the notion of an "encompassing transaction", that is, a transaction that may involve resource managers other than a single SQL database system (including possibly non-SQL resource managers) and controlled by some entity other than the SQL database system. Communication between that other entity (usually called a transaction manager) and the resource managers requires a standardized interface, because such products are often required to participate in the same global transaction even though they are purchased from different vendors at different times and may be operating at different nodes in a communications network. In this more global transaction processing environment, the encompassing transaction will be initiated and terminated by that other agent, which interacts with the SQL environment via an interface that may be different from SQL COMMIT and ROLLBACK statements.

The X/Open Company, Ltd., has defined an application program interface (API), called the "XA interface", to coordinate the activities of resource managers (RM) and transaction managers (TM). This interface has two components, called the xa_component and the ax_component. The xa_component is to be used by TMs to influence the actions of RMs, whereas the ax_component is to be used by RMs to communicate with the TMs. The XA-interface, as defined by X/Open, specifies an API for the ISO Distributed Transaction Processing (DTP) standard (ISO/IEC 10026:1992), which itself only specifies services and protocols. ISO has not yet published any APIs for this standard, or for other related OSI standards, although there is a study group in ISO/IEC JTC1/SC21 to examine the API question. The XA-interface was published by the X/Open Company, Ltd., in 1991 and 1992 as a "Common Application Environment" specification, which is intended to be completely stable. X/Open has continued work on their DTP specifications and a follow-on enhancement, informally called XA2, is in development; however, XA2 is expected to be completely compatible with the existing XA specification, while adding additional facilities.

A number of SQL resource managers are also beginning to claim conformance to the xa_component of the XA-interface. In recognition of this trend, both ANSI and ISO/IEC approved new SQL standardization projects in 1994 that will lead to formal standardization of an SQL Specialization of the X/Open XA-interface as an ANSI/ISO/IEC International Standard. These projects are just getting started, but the base document is derived from and completely compatible with the existing X/Open specification, and the project description requires that it retain this compatibility during development. The following functions are defined as part of the xa-component:

xa_close, xa_commit, xa_complete, xa_end, xa_forget, xa_open, xa_prepare, xa_recover, xa_rollback, and xa_start. The resulting International Standard for the SQL/XA Interface will be a specialization of these functions for SQL resource managers that retains existing syntax and intended semantics, but may refine the details of how SQL resource managers shall respond to xa-routines in terms of existing SQL transaction semantics.

The XA-interface is intended only for calls from a global transaction manager to various resource managers. It is not intended for normal use by end-user application programs. Instead, application programs should use interfaces designed for communications between the application program and the global transaction manager. For this reason, procurements should be very careful how they specify requirements for encompassing transactions; it may be a requirement that SQL/ERI Servers be able to process two-phase commit requests, and thereby participate as resource managers in a global transaction under the direction of a global transaction manager, but it may not be a requirement that an end-user application program be able to communicate directly with such a resource manager using xa-routines.

6. SQL Binding Alternatives

The SQL'92 standard [8] specifies three different binding styles: Module, Embedded SQL, and Direct Invocation. The RDA'93 standard [9] specifies protocol interfaces for RDA clients and RDA servers, and an emerging standard for the SQL call level interface (SQL/CLI) is under a rapid development path in ISO/IEC with final approval as a new International Standard expected sometime during calendar year 1995. The following subsections describe how an SQL/ERI Server may claim conformance to an SQL/ERI profile using one of these interface alternatives.

6.1 SQL Module processor

An SQL/ERI Server may provide a Module binding style to application programs. If a user creates a <module> according to the Format and Syntax Rules of Clause 12, "Module", of the SQL'92 standard, and if the <module> satisfies the restrictions of a given level of SQL for a given programming language, then the SQL/ERI Server shall process that <module> as an input text file and shall produce a binary output file that can be linked to the compiled output of any application program written in the programming language identified by the <language clause> of the <module>. The <module> output file shall abide by whatever restrictions are required for cross-language procedure calls by the operating system and processing platform for which Module binding support is claimed.

6.2 Embedded SQL preprocessor

An SQL/ERI Server may provide an Embedded SQL binding style to application programs. If a user creates an <embedded SQL host program> according to the Format and Syntax Rules of Clause 13, "Embedded SQL", of the SQL'92 standard, and if the <embedded SQL host program> satisfies the restrictions of a given level of SQL for a given programming language, then the SQL/ERI Server shall process that <embedded SQL host program> according to the General Rules and other requirements specified in the SQL'92 standard. An SQL/ERI Server may compile the entire <embedded SQL host program> to produce an executable file, or it may produce a conforming program, P, written in the language identified by the <language clause> of the <embedded SQL

host program> and an implicit (maybe not actual) module, M, both as specified by Syntax Rules 13 through 15 of Subclause 19.1, "<embedded SQL host program>", of the SQL'92 standard. If the user compiles program P with a standard conforming programming language compiler designed for the operating system and processing platform environment for which Embedded SQL support is claimed, then the compiled version of P and an implementor-dependent version of M shall be linkable in that processing environment to produce an executable file that executes correctly according to the SQL'92 standard.

6.3 Direct invocation of SQL statements

An SQL/ERI Server may provide a Direct Invocation style of binding according to the requirements of Clause 20, "Direct invocation of SQL", of the SQL'92 standard. This binding style is very difficult to write conformance tests for because there is no "standard" way to capture data returned as the result of a query. Instead, conformance to this binding style requires a subjective evaluation of the results by a human user. For this reason, among others, the FIPS SQL standard [3] does not recognize Direct Invocation as the sole conformance alternative of an SQL implementation. Instead, it allows Interactive Direct SQL as a conformance option in addition to a Module or Embedded binding style.

For an SQL/ERI Server it makes more sense to recognize the Direct Invocation binding style as a viable conformance alternative. There are many situations, e.g. electronic bulletin boards, where a user may desire to send an SQL statement to an SQL/ERI Server and have the data results displayed on a screen or dumped into a human readable text file. For these reasons, the SQL/ERI Server profiles specified in Section 7 below do recognize Direct Invocation as a valid conforming interface style.

If a user creates a <direct SQL statement> according to the Format and Syntax Rules of Clause 20, "Direct invocation of SQL", of the SQL'92 standard, and if the <direct SQL statement> satisfies the restrictions of a given level of SQL, then the SQL/ERI Server shall process that <direct SQL statement> as input text and shall display the results, if any, in a human readable form on some sort of display device. For SQL/ERI Servers providing access to multimedia data, the display device may include a sound system, motion picture display, or even some form of virtual reality. The only real requirement is that a reasonable conformance testing authority be able to decide, subjectively, whether or not the <direct SQL statement> was properly executed.

If a <direct SQL statement> is a <direct select statement: multiple rows> that returns only character string or numeric data in the result rows and columns, then the SQL/ERI Server shall provide a user option to redirect the output of the query into a human readable text file. In this context, human readable means formatted so that a reasonable conformance testing authority can readily distinguish rows and columns and easily read the data. All numeric data returned as text shall be in the form of a valid SQL <signed numeric literal>, unless some explicit user action results in its being cast into some other form, e.g. Money with currency symbols attached. The SQL/ERI Server may use General Rules 5a, 5b, 6a, and 6b, of Subclause 6.10, "<cast specification>", of the SQL'92 standard, for additional guidance in casting numeric values into numeric literals.

Other requirements for the Direct Invocation binding style are as follows: if a statement raises an exception condition, then the SQL/ERI Server shall display a message indicating that the statement failed, giving a textual description of the failure; if a statement raises a completion condition that is a "warning" or "no data", then the SQL/ERI Server shall display a message indicating that the statement completed, giving a textual description of the "warning" or "no data"; an SQL/ERI Server that supports null values shall provide some implementation-defined symbol for displaying null values and, for character string values, this symbol must be distinguishable from a value of all <space>s. Preferably, the SQL/ERI Server will provide an implementor-defined method for a user to specify how null values shall be displayed, e.g. SET NULL AS '*'; however, this SET feature is not really needed if, instead, the SQL/ERI Server supports the NULL alternative in the <cast operand> of the <cast specification>.

6.4 SQL call level interface (SQL/CLI)

An SQL/ERI Server may provide an SQL Call Level Interface binding style according to the requirements of the emerging standard for SQL/CLI [10]. We expect the SQL/CLI specification to be formally approved as an ANSI/ISO/IEC standard sometime during calendar year 1995, in time for any future NIST testing of SQL/ERI Server profiles.

The call level interface is a requirement for third-party software developers who produce "shrink-wrapped" software for use on personal computers and workstations. They do not wish to use a Module processor or an Embedded SQL preprocessor binding style because they do not wish to distribute any source code with the products they sell to individual users. Instead they desire a services call interface to SQL data repositories that can be invoked from the calling environment provided by the host operating system. The calls to the SQL data repository can then be embedded in the object code just like calls to any other system service.

The Call Level Interface is an alternative mechanism for executing SQL statements. Reference [10] states that the SQL/CLI consists of a number of routines that:

-- Allocate and deallocate resources.

-- Control connections to SQL-servers.

-- Execute SQL statements using mechanisms similar to Dynamic SQL.

-- Obtain diagnostic information.

-- Control transaction termination.

-- Obtain information about the implementation.

The AllocHandle routine allocates resources to manage an SQL-environment, an SQL-connection, a CLI descriptor area, or SQL-statement processing. An SQL-connection is allocated in the context of an allocated SQL-environment, and a CLI descriptor descriptor area and an SQL-statement are allocated in the context of an allocated SQL-connection. The FreeHandle routine deallocates a specified resource. The ReleaseEnv routine is used to deallocate all the allocated SQL-connections within a specified allocated SQL-environment.

Each allocated SQL-environment has an attribute that determines whether output character strings are null terminated by the implementation. The application can set the value of this attribute by using the routine SetEnvAttr and can retrieve the current value of the attribute by using the routine GetEnvAttr.

The Connect routine establishes an SQL-connection. The Disconnect routine terminates an established SQL-connection. Switching between established SQL connections occurs automatically whenever the application switches processing to a dormant SQL-connection. The ExecDirect routine is used for a one-time execution of an SQL-statement. The Prepare routine is used to prepare an SQL-statement for subsequent execution using the Execute routine. In each case, the executed SQL-statement may contain dynamic parameters.

The interface for a description of dynamic parameters, dynamic parameter values, the resultant columns of a dynamic select statement, and the target specifications for the resultant columns is a CLI descriptor area. A CLI descriptor area for each type of interface is automatically allocated when an SQL-statement is allocated. The application may allocate additional CLI descriptor areas and nominate them for use as the interface for the description of dynamic parameter values or the description of target specifications by using the routine SetStmtAttr. The application can determine the handle value of the CLI descriptor area currently being used for a specific interface by using the routine GetStmtAttr. The GetDescField and GetDescRec routines enable information to be retrieved from a CLI descriptor area. The CopyDesc routine enables the contents of a CLI descriptor area to be copied to another CLI descriptor area.

When a dynamic select statement is prepared or executed immediately, a description of the resultant columns is automatically provided in the applicable CLI descriptor area. In this case, the application may additionally retrieve information by using the DescribeCol routine to obtain a description of a single resultant column and by using the NumResultCols routine to obtain a count of the number of resultant columns. The application sets values in the CLI descriptor area for the description of the corresponding target specifications either explicitly using the routines SetDescField and SetDescRec or implicitly using the routine BindCol.

When an SQL-statement is prepared or executed immediately, a description of the dynamic parameters is automatically provided in the applicable CLI descriptor area if this facility is supported by the current SQL-connection. An attribute associated with the allocated SQL-connection indicates whether this facility is supported. The value of the attribute may be retrieved using the routine GetConnectAttr. The application sets values in the CLI descriptor area for the description of dynamic parameter values and, regardless of whether automatic population is supported, in the CLI descriptor area for the description of dynamic parameters either explicitly using the routines SetDescField and SetDescRec or implicitly using the routine BindParam.

When a dynamic select statement is executed, a cursor is implicitly declared and opened. The cursor name can be supplied by the application by using the routine SetCursorName. If a cursor name is not supplied by the application, an implementation-dependent cursor name is generated. The same cursor name is used for each implicit cursor within a single allocated SQL-statement. The cursor name can be retrieved by using the GetCursorName routine.

The Fetch routine is used to position an open cursor on the next row and retrieve the values of bound columns for that row. A bound column is one whose target specification in the specified CLI descriptor area defines a location for the target value. Values for unbound columns can be individually retrieved by using the GetCol routine. The GetCol routine also enables the values of character string columns to be retrieved piece by piece. The current row of a cursor can be deleted or updated by executing a preparable dynamic delete statement or a preparable dynamic update statement, respectively, for that cursor under a different allocated SQL-statement to the one under which the cursor was opened. The CloseCursor routine enables a cursor to be closed.

The GetDiagField and GetDiagRec routines obtain diagnostic information about the most recent routine operating on a particular resource. The Error routine is used to obtain diagnostic information about the execution of the most recent routine. The Error routine may be used instead of the resource specific diagnostic routines GetDiagField and GetDiagRec. Information on the number of rows affected by the last executed SQL-statement can be obtained by using the RowCount or GetDiagField routine. The GetFunctions, GetInfo, and GetTypeInfo routines are used to obtain other information about the implementation.

An SQL-transaction is terminated by using the EndTran routine. The Cancel routine is used to cancel the execution of a concurrently executing SQL/CLI routine or to terminate the processing of deferred parameter values and the execution of the associated SQL-statement.

The CLI ExecDirect routine and the CLI Prepare routine each support an input character string parameter identified as StatementText. If P is the value of StatementText, then P shall satisfy the following restrictions:

1. P shall conform to the Format, Syntax Rules, and Access Rules for a <preparable statement> as specified in Subclause 17.6, "<prepare statement>", of the SQL'92 standard.

2. P shall not be a <commit statement> or a <rollback statement>.

3. P shall abide by the Leveling Rules of the level of SQL support claimed by the SQL/ERI Server.

The SQL/CLI specification is intended to support CLI routines embedded into both pointer-based programming languages and non-pointer-based programming languages. In particular, the Syntax Rules of <CLI routine> specified in Subclause 5.1 of [10] indiate that CLI routines may be embedded into any one of the following standard programming languages: Ada, C, COBOL, Fortran, MUMPS, Pascal, and PL/I. An SQL/ERI Server will indicate which of those languages it supports.

6.5 RDA/SQL-Server interface

An SQL/ERI Server may provide an RDA/SQL-Server protocol interface according to the protocols defined in the RDA'93 standard [9]. The RDA protocols allow communication and interoperability among conforming RDA processors in an OSI communications network. Many vendors are also supporting the RDA protocols in a TCP/IP communications network using agreements specified by the NIST Opens Systems Environment Implementors Workshop (OIW) for RDA accessibility using Internet RFC 1006 for upper layer OSI interface to Internet protocols. This SQL/ERI profile specification allows an SQL/ERI Server to claim conformance to the RDA/SQL-Server interface over an arbitrary communications network. If an application program, acting as an RDA client, is able to form an association with an SQL/ERI Server and communicate thereafter using RDA protocols subject to the implementor agreements specified by the Open Systems Environment Implementor's Workshop (e.g. in [16]), then the SQL/ERI Server may claim conformance to the RDA/SQL-Server interface style.

Reference [9] describes the services of the RDA standard in terms of an RDA Client, an RDA Server, and an RDA Service as follows:

> An RDA client is an application-process, within an open system, that requests database services from another application-process called a database server. A database server is an application-process, within the same or another open system, that supplies database storage facilities and provides, through OSI communication, database services to RDA clients. An RDA client and a database server communicate by means of the RDA Service, supported by an RDA service-provider. The part of the database server that uses the RDA service-provider to communicate with an RDA client is called an RDA server. The RDA client has the ability to initiate RDA service requests, while the RDA server can only issue RDA service responses to reply to such requests.
>
> A data resource is a named collection of data and/or capabilities on the database server known to both the RDA client and the RDA server. The meaning of the data content and capabilities of a data resource depend upon the application of RDA, which is determined by each RDA specialization standard (e.g. the SQL specialization). The RDA client opens a data resource in order to gain access to the data content or capabilities of that resource through Database Language services (e.g. SQL).

An RDA transaction is a logically complete unit of processing as determined by the RDA client. Execution during an RDA transaction of a sequence of database access services that change data resources enables the set of changes to be handled as an atomic unit. When the RDA transaction is terminated, either the whole set of changes is applied to the data resources or no changes are applied. The RDA client requests termination of an RDA transaction by requesting the RDA server either to commit or to roll back the complete set of changes of that transaction. Changes made to the data content of data resources during an RDA transaction are not made available to other RDA clients until that RDA transaction is terminated at the RDA server. RDA provides a choice of two application-contexts for managing RDA transactions: 1) a basic application-context for one-phase commitment, and 2) a TP application-context for two-phase commitment. The RDA protocol for the basic application-context is completely specified in the RDA standard, whereas the protocol for the TP application context is dependent upon the ISO/IEC Distributed Transaction Processing standard (ISO/IEC 10026).

An RDA operation models a request by an RDA client that is transferred to an RDA server for processing. RDA operations enable an RDA client to request any of five types of RDA services:

a) RDA Dialogue Management services, to start and end RDA dialogues;

b) RDA Transaction Management services, to start and end RDA transactions;

c) RDA Control services, to report the status or cancel existing operations;

d) Resource Handing services, to enable or disable access by RDA clients to data resources;

e) Database Language services, to access and modify data resources.

An RDA client may request RDA operations without waiting for the results of previously requested RDA operations. Thus an RDA server may have several RDA operations outstanding for a particular RDA dialogue.

An RDA dialogue is a cooperative relationship between and RDA client and an RDA server. The RDA client initilizes the RDA dialogue and requests RDA operations that are to be performed by the RDA server. An RDA dialogue is uniquely identified within the scope of the OSI environment, and all RDA operations occur within the bounds of an RDA dialogue. An RDA dialogue can exist only in the context of an established application-association, and ceases to exist if the association is released. A failed RDA dialogue cannot be recovered; the process of recovery after a failure is a local matter beyond the scope of the RDA 1993 standard, and recovery actions outside the RDA service-provider may be necessary. In the event of dialogue failure, it is a requirement that all changes made to data resources by any RDA transaction that is not already terminating when RDA dialogue failure occurs be rolled back by the database server during its recovery process. If an RDA dialogue is terminating when RDA dialogue failure occurs, then it may either be committed or rolled back.

The NIST OSE Implementor's Workshop (OIW) has specified implementation agreements for the Basic Application Context of the RDA'93 standard [9], with profiles for: Immediate Execution, Stored Execution, Status, and Cancel. Future work is in progress by the OIW to specify corresponding profiles for the Transaction Processing (TP) Application Context of the RDA'93 standard. For the purpose of the SQL/ERI Server profiles specified in this document, support for the RDA/SQL-Server interface requires support as an RDA Server for the Immediate Execution profile of the Basic Application Context as specified in [16], with the ability to respond to SQL statements at the level of support for SQL language claimed by the SQL/ERI Server. The other profiles of the Basic Application Context defined in [16], and the TP Application Context, are optional enhancements to this basic requirement as follows:

RDA Stored Execution. Support for the basic requirements specified above and, in addition, support for the RDA Stored Execution Functional Unit as specified in the RDA'93 standard and with implementor agreements for the Stored Execution profile as specified in [16].

RDA Status. Support for the basic requirements specified above and, in addition, support for the RDA Status Functional Unit as specified in the RDA'93 standard and with implementor agreements for the Status profile as specified in [16].

RDA Cancel. Support for the basic requirements specified above and, in addition, support for the RDA Cancel Functional Unit as specified in the RDA'93 standard and with implementor agreements for the Cancel profile as specified in [16].

RDA TP Application Context. Support for the basic requirements specified above and, in addition, support for the RDA SQL TP Application Context as specified in the RDA'93 standard, and dependent upon ISO/IEC 10026 (Distributed Transaction Processing), and with implementor agreements for Distributed Transaction Processing as specified in [16].

7. SQL/ERI Server Profiles

This section specifies two general-purpose functional profiles for partial SQL language support that an implementation may claim conformance to, as follows:

- SQL/ERI Read-Only Server
- SQL/ERI Read-Write Server

Each general-purpose profile has a number of "level of service" alternatives for data manipulation, schema definiton, transaction management, and binding style. If an implementation conforms to any one of these profiles, then it may claim to be a FIPS conforming SQL/ERI Server. Because many of the alternatives in these profiles identify a proper subset of full-function SQL requirements, conformance to any one of them does not imply conformance to the standard for Database Language SQL [8]. These profiles are intended for use by customers and vendors of products that claim only partial support of an SQL language interface to their data repository.

Any implementation claiming conformace to one of the SQL/ERI Server profiles shall provide a written public statement responding to the ten profile items identified in the following paragraphs. The implementation requirements of each response are given in Subsection 7.1 for Read-Only Servers and in Subsection 7.2 for Read-Write Servers. Syntax for deriving specific SQL-related object identifiers is given in Section 7.3, and popular profiles for CLI and RDA binding alternatives are defined in Sections 7.4 and 7.5, respectively.

An SQL/ERI Server profile shall specify:

1. A base level of SQL data manipulation language (DML) support, by choosing exactly one of the following DML alternatives.

    Minimal DML
    Entry DML
    Transitional DML
    Intermediate DML
    Full DML

2. A base level of SQL schema definition language (SDL) support, by choosing exactly one of the following SDL alternatives.

    No SDL
    Minimal SDL
    Entry SDL
    Transitional SDL
    Intermediate SDL
    Full SDL

3. A base level of SQL transaction management support, by choosing exactly one of the following transaction management alternatives.

    No Transactions
    Commit-Rollback
    Transaction Mode
    Transaction Isolation
    Transaction Diagnostics
    Constraints

Note: The alternatives for SQL transaction management support are nested. Support for any one of them implies support for all those listed above it. The three alternatives for Transaction Mode, Transaction Isolation, and Transaction Diagnostics support <transaction mode>, <isolation level>, and <diagnostics size>, respectively, in the SQL'92 <set transaction statement>, and the Constraints alternative supports the SQL'92 <set constraints mode statement>.

4. A default isolation level for SQL transaction management, by choosing exactly one of the following default isolation level alternatives.

    Read Uncommitted
    Read Committed
    Repeatable Read
    Serializable

Note: If the default isolation level is anything other than Serializable, and if other concurrent users are able to update the database, then read statements may be subject to "dirty read", "non-repeatable read", or "phantom" rows (see Subclause 4.28, "SQL-transactions", of the SQL'92 standard). Even a read-only profile is subject to these phenomena if other concurrent users (not through that profile) can update the database.

5. Which binding styles are supported, by choosing one or more of the following binding style alternatives.

   Module
   Embedded SQL
   Direct Invocation
   SQL/CLI
   RDA/SQL

Note: It is expected that the SQL/CLI binding style will be the most popular choice for SQL/ERI products within a single local client/server environment and that the Direct Invocation or RDA/SQL binding styles will be the most popular when the server data repository is an isolated node in a wide area client/server environment.

6. For each of the Module, Embedded SQL, or SQL/CLI binding styles chosen, which programming language interfaces are supported, by choosing one or more of the following programming language alternatives.

   Ada
   C
   COBOL
   Fortran
   MUMPS
   Pascal
   PL/I
   SAMeDL (via module, embedded, or effect)

Note: The Direct Invocation and RDA/SQL binding styles do not require or provide a programming language interface. The preferred language interfaces for SQL/CLI are C and/or COBOL. SAMeDL is an alternative only for the Module binding style (see [21]).

7. Which SQL session management facilities are supported, by specifying one or more of the following session management features.

   No Session
   Set Catalog
   Set Schema
   Set Names
   Set Session Authorization
   Set Time Zone
   All Session

Note: SQL session management is defined in Clause 16, "Session management", of SQL'92. If the base level of SQL DML support specified above is Intermediate DML or above, then implementations shall support both Set Session Authorization and Set Time Zone, because they are required Intermediate SQL features. Because of its importance for users, support for "SET SCHEMA <unqualified schema name>" is required if SQL DML support is Transitional DML or above.

8. Which optional extensions are supported, by choosing <u>one or more</u> of the following optional extensions.

    No Extensions
    SQL Features
    Executable SQL/PSM
    Definable SQL/PSM
    SQL/MM: FullText
    SQL/MM: Spatial
    SQL/MM: General
    ADTs and methods
    Object data management
    SQL/XA

Note: If SQL Features is chosen, then the implementation shall support the SQL_FEATURES table as specified in Section 15 of FIPS SQL (see [3]); if Executable SQL/PSM is chosen, then the implementation shall support <routine invocation> and the ROUTINES base table as specified in the SQL/PSM specification (see [11]); if Definable SQL/PSM is chosen, then the implementation shall support all requirements of the SQL/PSM specification (see [11]); if SQL/MM: Full Text, SQL/MM: Spatial, or SQL/MM: General is chosen, then the implementation shall point to the then current SQL/MM specification (see Section 5.3 above and [13]) and explicitly indicate which Parts, and which conformance alternatives within each Part, are supported; if ADTs and methods is chosen, then the implementation shall support the appropriate Abstract Data Type clauses in the then current SQL3 specification (see Section 5.4 above and [12]); if Object data management is chosen, then the implementation shall support the appropriate object management clauses in the then current SQL3 specification (see Section 5.5 above and [12]); if SQL/XA is chosen, then the implementation shall support the SQL specializaion of the X/Open XA interface specification (see Section 5.6 above and [12]).

9. If the RDA/SQL binding style is chosen, then which underlying communications protocols are supported, by choosing <u>one or more</u> of the following alternatives.

Minimal OSI (MOSI)	-- see new OIW 1994 agreements
Full Stack OSI	-- see [16] for 1992 OIW stable agreements
Internet RFC 1006	-- see unpublished NIST RDA TestBed Agreements
Other Transport	-- give name & version of transport mechanism used

Note: All of the above depend upon the International Standard for Remote Database Access (RDA) [9] in their upper layers; however, they may differ in their directory services and in their services for making an association at the application layer and in how that association is propagated through to the transport and physical layers. It is expected that the Internet RFC 1006 alternative will be the most popular in the near term (because the Internet is so pervasive) for *ad hoc* associations among RDA clients and servers in a wide area network.

10. If the RDA/SQL binding style is chosen, then which RDA options are supported, by choosing <u>one or more</u> of the following (see Section 6.5 above).

    None
    RDA Stored Execution
    RDA Status
    RDA Cancel
    RDA TP Application Context

7.1 SQL/ERI Read-Only Server

This profile specifies a read-only interface to a data repository. It does not include support for any of the Schema Definition or Schema Manipulation SQL language elements specified in Clause 11 of the SQL'92 standard, or for any of the SQL data change statements, i.e. Insert, Update, or Delete, specified in Clause 13. It is most likely that the level of support specified for SQL schema definition language will be "No SDL". Depending upon the various base level attributes specified, this profile may have Information Schema requirements that differ from those specified in SQL'92 [8] or FIPS SQL [3].

Schema Definition Rules

1. The SQL/ERI Read-Only Server profile assumes that all schema objects are owned by a user different from the user accessing the repository through this profile, and that appropriate privileges have been granted to all accessing users. If the SQL/CLI binding style is identified, then users are made known to the system using the Connect routine specified in Subclause 6.10, "Connect", of [10]. If the RDA/SQL binding style is identified, then users are made known to the system using the NIST OIW RDA Testbed implementor agreements. Otherwise, as with the SQL'92 standard, the particular method by which users are made known to the system is implementation-defined.

2. If the level of SQL data manipulation language support claimed for the SQL/ERI Read-Only Server profile is Minimal DML, Entry DML, or Transitional DML, then the implicit schema definition may contain some <table constraint>s, or various <schema element>s, that are not visible to the user but whose existence may affect the semantics of certain statements.

3. If the level of SQL data manipulation language support claimed for the SQL/ERI Read-Only Server profile is Minimal DML or Entry DML, and if a table with table name TN is visible in the Information Schema TABLES view for a user with user name UN, then one of the following <grant statement>s, executed by the owner of table TN, is implicit:

    GRANT SELECT ON TABLE TN TO UN, or
    GRANT SELECT ON TABLE TN TO PUBLIC

    It doesn't make any difference to a read-only user which of these statements is implicit, so the choice is implementation-dependent.

4. Information about schema objects, privileges, and constraints are made visible to potential users through the Information Schema views, subject to the Information Schema Rules specified below.

Data Manipulation Rules

1. If the Module, Embedded SQL, or RDA/SQL binding styles are specified, then the SQL/ERI Read-Only Server profile requires support for the following SQL statements, as specified in Clause 13, "Data Manipulation", in the SQL'92 standard, with any restrictions specified by the given level of SQL data manipulation language support and subject to other rules specified in this profile.

    <declare cursor>
    <open statement>
    <fetch statement>
    <close statement>
    <select statement: single row>

2. If the Direct Invocation binding style is specified, then the SQL/ERI Read-Only Server profile requires support for the following <direct SQL statement>s listed in Clause 20, "Direct invocation of SQL", in the SQL'92 standard, with any restrictions specified by the given level of SQL data manipulation language support and subject to other rules specified in this profile:

   <direct select statement: multiple rows>

3. If the SQL/CLI binding style is specified, then the SQL/ERI Read-Only Server profile requires support for the following <preparable statement>s listed in Subclause 17.6 of the SQL'92 standard, with any restrictions specified by the given level of SQL data manipulation language support and subject to other rules specified in this profile:

   <dynamic single row select statement>
   <dynamic select statement>

4. If an SQL/ERI Server implementation at the Minimal SDL level or below chooses not to provide support for null values (see item 4 of Section 4.1), then it may raise an implementation-defined exception in any SQL statement that attempts to process null values.

Transaction Management Rules

1. If the level of SQL transaction management support is "No Transactions", then SQL transaction management is not supported for any binding style. Otherwise, Commit and Rollback transaction management is supported depending on the specified binding style as follows.

   Case:

   a. If the Module, Embedded SQL, or Direct Invocation binding style is specified, then the requirements of the SQL <commit statement> and the SQL <rollback statement> from Clause 14, "Transaction management", of the SQL'92 standard [8] apply to this profile.

   b. If the SQL/CLI binding style is specified, then the requirements of the routines for transaction management (e.g. EndTran and Cancel) as specified in the SQL/CLI specification [10] apply to this profile.

   c. If the RDA/SQL binding style is specified, then the requirements for transaction management in the RDA Basic Application Context, as specified in the RDA standard [9], with implementor agreements specified in [16], apply to this profile.

   d. If the RDA option for TP Application Context is specified, then the requirements for the TP Application Context, as specified in the RDA standard [9], with implementor agreements for Distributed Transaction Processing as specified in [16], apply to this profile.

   Note: The purpose of requiring support for SQL Commit and Rollback in Read-Only profiles is to give the user a standard way to signal to the system that a read-only transaction has completed. This has semantic implications only if other concurrent users (not through this profile) are able to update the database.

2. If the level of SQL transaction management support is "No Transactions", and if the default isolation level is XXX, then the <set transaction statement>

SET TRANSACTION READ ONLY, ISOLATION LEVEL XXX

is implicit for the single implicit transaction of any SQL-session through this profile.

3. If the level of SQL transaction management support is Commit-Rollback, and if the default isolation level is XXX, then the <set transaction statement>

   SET TRANSACTION READ ONLY, ISOLATION LEVEL XXX

   is implicit for every transaction of any SQL-session through this profile.

4. If the level of SQL transaction management support is Transaction Mode or above, then this profile includes support for the <transaction access mode> alternative of the SQL <set transaction statement> as specified in Subclause 14.1 of the SQL'92 standard; however, the <transaction access mode> shall always be READ ONLY.

5. If the level of SQL transaction management support is Transaction Isolation or above, then this profile includes support for the <isolation level> alternative of the SQL <set transaction statement> as specified in Subclause 14.1 of the SQL'92 standard. If the default isolation level is XXX, and if an explicit <set transaction statement> with an explicit <isolation level> is not specified for a transaction of any SQL-session through this profile, then the <set transaction statement>

   SET TRANSACTION READ ONLY, ISOLATION LEVEL XXX

   is implicit for that transaction.

6. If the level of SQL transaction management support is Transaction Diagnostics or above, then this profile includes support for the <diagnostics size> alternative of the SQL <set transaction statement> as specified in Subclause 14.1 of the SQL'92 standard.

7. If the level of SQL transaction management support is Constraints, then this profile includes support for the <set constraints mode statement> as specified in Subclause 14.2 of the SQL'92 standard.

8. If the optional extension for SQL/XA is specified, then the implementation shall support the SQL specializaion of the X/Open XA interface specification (see Section 5.6 above and [12]).

Connection Management Rules

1. If the Module, Embedded SQL, or Direct Invocation binding styles are specified, and if the level of SQL data manipulation language support is Full DML, then the SQL/ERI Read-Only Server profile requires support for <SQL connection statement>s as specified in Clause 15, "Connection management", of the SQL'92 standard. If the level of SQL data manipulation language support is anything other than Full SQL, then there is no requirement to support any <SQL connection statement> for these binding styles.

2. If the SQL/CLI binding style is specified, then the requirements of the routines for connection management (i.e. Connect, Disconnect) as specified in the SQL/CLI specification [10] apply to this profile.

3. If the RDA/SQL binding style is specified, then the requirements of RDA Dialogue Management and RDA Resource Handling as specified in the RDA standard [9], with implementor agreements specified in [16], apply to this profile.

Session Management Rules

1. If the indicated SQL session management support is "No Session", then SQL session management is not supported for any binding style. Otherwise, for each feature identified, this profile supports the requirements of that feature as identified in Clause 16, "Session management", of the SQL'92 standard. If the indicated SQL session management support is "All Session", then all features of Clause 16 are supported in this profile.

2. If SQL data manipulation language support specifies Transitional DML or above, then support for "SET SCHEMA <unqualified schema name>" is implicit in this profile.

3. If SQL data manipulation language support specifies Intermediate DML or above, then support for "SET SCHEMA <unqualified schema name>", SET SESSION AUTHORIZATION, and SET TIME ZONE is implicit in this profile.

4. If SQL data manipulation language support specifies Full DML, then support for "All Session" is implicit in this profile.

Dynamic SQL and Diagnostics Management Rules

1. If SQL data manipulation language support specifies Minimal DML or Entry SQL, then support for SQL statements in Clause 17, "Dynamic SQL", and Clause 18, "Diagnostics management", is not required.

2. If SQL data manipulation language support specifies Transitional DML or above, then support for all read-only provisions of Clause 17, "Dynamic SQL", and Clause 18, "Diagnostics management", with any restrictions identified in the Leveling Rules for higher levels of DML, is required for all implementations that claim that level of DML support.

Information Schema Rules

1. If the level of SQL data manipulation language support claimed for the SQL/ERI Read-Only Server profile is Minimal DML or Entry DML, then support for the following Information Schema views, as specified in Clause 21, "Information Schema and Definition Schema", of the SQL'92 standard, is required:

   TABLES
   COLUMNS

   Note: See FIPS 127-2 Errata for handling "long" names.

2. If an SQL/ERI Server implementation at the Minimal SDL level or below chooses not to provide support for null values (see item 4 of Section 4.1), then it shall provide an implementation-defined conversion of would-be null values in Information Schema tables to an appropriate non-null value.

3. If the level of SQL data manipulation language support claimed for the SQL/ERI Read-Only Server profile is Transitional DML, then support for the following Information Schema views, as specified in Clause 21, "Information Schema and Definition Schema", of the SQL'92 standard, is required:

   TABLES
   VIEWS
   COLUMNS
   TABLE_PRIVILEGES
   COLUMN_PRIVILEGES
   USAGE_PRIVILEGES

   Note: See FIPS 127-2 Errata for handling "long" names.

4. If the level of SQL data manipulation language support claimed for the SQL/ERI Read-Only Server profile is Intermediate DML or Full DML, then an implementation conforming to this profile shall provide all of the Information Schema views required by the SQL'92 standard for Intermediate SQL or Full SQL, respectively. In many cases some of these tables may be empty, or trivial, but a conforming SQL/ERI Server at these SQL data manipulation language levels is required to support them to reflect an accurate picture of the implicit schema definition.

7.2 SQL/ERI Read-Write Server

This profile specifies a read-write interface to a data repository. It requires support for the SQL data change statements, i.e. Insert, Update, and Delete, specified in Clause 13, "data manipulation", of the SQL'92 standard. Depending upon the level of SQL schema definition language support specified, it may or may not require support for SQL schema definition or schema manipulation statements. Depending upon the various base level attributes specified, this profile may have Information Schema requirements that differ from those specified in SQL'92 [8] or FIPS SQL [3].

Schema Definition Rules

1. The SQL/ERI Read-Write Server profile assumes that some schema objects are owned by a user different from the user accessing the repository through this profile, and that appropriate privileges have been granted to all accessing users. If the SQL/CLI binding style is identified, then users are made known to the system using the Connect routine specified in Subclause 6.10, "Connect", of [10]. If the RDA/SQL binding style is identified, then users are made known to the system using the NIST OIW RDA Testbed implementor agreements. Otherwise, as with the SQL'92 standard, the particular method by which users are made known to the system is implementation-defined.

2. If the level of SQL data manipulation language support claimed for the SQL/ERI Read-Write Server profile is Minimal DML, Entry DML, or Transitional DML, then the implicit schema definition may contain some <table constraint>s, or various <schema element>s, that are not visible to the user but whose existence may affect the semantics of certain statements.

3. Information about schema objects, privileges, and constraints are made visible to potential users through the Information Schema views, subject to the Information Schema Rules specified below.

4. If the level of SQL schema definition language support is different from "No SDL", then

   Case:

   a. If the Module, Embedded SQL, or RDA/SQL binding styles are specified, then the SQL/ERI Read-Write Server profile requires support for all of the SQL data definition and manipulation statements, as specified in Clause 11, "Schema definition and manipulation", of the SQL'92 standard, with any restrictions specified by the given level of SQL schema definition language support and subject to other rules specified in this profile.

   b. If the Direct Invocation binding style is specified, then the SQL/ERI Read-Write Server profile requires support for the following <direct SQL statement>s listed in Clause 20, "Direct invocation of SQL", in the SQL'92 standard, with any restrictions specified by the given level of SQL schema definition language support and subject to other rules specified in this profile:

      <SQL schema statement>

   c. If the SQL/CLI binding style is specified, then the SQL/ERI Read-Write Server profile requires support for the following <preparable statement>s listed in Subclause 17.6 of the SQL'92 standard, with any restrictions specified by the given level of SQL schema defintion language support and subject to other rules specified in this profile:

      <preparable SQL schema statement>

Data Manipulation Rules

1. If the Module, Embedded SQL, or RDA/SQL binding styles are specified, then the SQL/ERI Read-Write Server profile requires support for the following SQL statements, as specified in Clause 13, "Data Manipulation", in the SQL'92 standard, with any restrictions specified by the given level of SQL data manipulation language support and subject to other rules specified in this profile.

   <declare cursor>
   <open statement>
   <fetch statement>
   <close statement>
   <select statement: single row>
   <delete statement: positioned>
   <delete statement: searched>
   <insert statement>
   <update statement: positioned>
   <update statement: searched>
   <temporary table declaration> -- Full DML only

2. If the Direct Invocation binding style is specified, then the SQL/ERI Read-Only Server profile requires support for the following <direct SQL statement>s listed in Clause 20, "Direct invocation of SQL", in the SQL'92 standard, with any restrictions specified by the given level of SQL data manipulation language support and subject to other rules specified in this profile:

   <direct select statement: multiple rows>
   <delete statement: searched>

<insert statement>
<update statement: searched>
<temporary table declaration> -- Full DML only

3. If the SQL/CLI binding style is specified, then the SQL/ERI Read-Write Server profile requires support for the following <preparable statement>s listed in Subclause 17.6 of the SQL'92 standard, with any restrictions specified by the given level of SQL data manipulation language support and subject to other rules specified in this profile:

   <dynamic single row select statement>
   <dynamic select statement>
   <delete statement: searched>
   <insert statement>
   <update statement: searched>
   <preparable dynamic delete statement: positioned>
   <preparable dynamic update statement: positioned>

4. If an SQL/ERI Server implementation at the Minimal SDL level or below chooses not to provide support for null values (see item 4 of Section 4.1), then it may raise an implementation-defined exception in any SQL statement that attempts to process null values.

Transaction Management Rules

1. The level of SQL transaction management support shall not be "No Transactions". In all cases, SQL Commit and Rollback transaction management is supported as defined by the specified binding style.

   Case:

   a. If the Module, Embedded SQL, or Direct Invocation binding style is specified, then the requirements of the SQL <commit statement> and the SQL <rollback statement> from Clause 14, "Transaction management", of the SQL'92 standard [8] apply to this profile.

   b. If the SQL/CLI binding style is specified, then the requirements of the routines for transaction management (e.g. EndTran and Cancel) as specified in the SQL/CLI specification [10] apply to this profile.

   c. If the RDA/SQL binding style is specified, then the requirements for transaction management in the RDA Basic Application Context, as specified in the RDA specification [9], with implementor agreements specified in [16], apply to this profile.

   d. If the RDA option for TP Application Context is specified, then the requirements for the TP Application Context, as specified in the RDA standard [9], with implementor agreements for Distributed Transaction Processing as specified in [16], apply to this profile.

2. If the level of SQL transaction management support is Commit-Rollback, then the <set transaction statement>

   SET TRANSACTION READ WRITE, ISOLATION LEVEL SERIALIZABLE

   is implicit for every transaction of any SQL-session through this profile.

3. If the level of SQL transaction management support is Transaction Mode or above, then this profile includes support for the <transaction access mode> alternative of the SQL <set transaction statement> as specified in Subclause 14.1 of the SQL'92 standard.

   Case:

   a. If an explicit <set transaction statement> with an explicit <tranaction access mode> is not specifed for a transaction of any SQL-session through this profile, then the <set transaction statement>

      SET TRANSACTION READ WRITE, ISOLATION LEVEL SERIALIZABLE

      is implicit for that transaction.

   b. If an explicit <set transaction statement> with a <tranaction access mode> of READ ONLY is specifed for a transaction of any SQL-session through this profile, and if the default isolation level is XXX, then the <set transaction statement>

      SET TRANSACTION READ ONLY, ISOLATION LEVEL XXX

      is implicit for that transaction.

4. If the level of SQL transaction management support is Transaction Isolation or above, then this profile includes support for the <isolation level> alternative of the SQL <set transaction statement> as specified in Subclause 14.1 of the SQL'92 standard. If an explicit <set transaction statement> with a <tranaction access mode> of READ ONLY is specifed, and if an explicit <set transaction statement> with an explicit <isolation level> is not specified for a transaction of any SQL-session through this profile, and if the default isolation level is XXX, then the <set transaction statement>

   SET TRANSACTION READ ONLY, ISOLATION LEVEL XXX

   is implicit for that transaction.

5. If the level of SQL transaction management support is Transaction Diagnostics or above, then this profile includes support for the <diagnostics size> alternative of the SQL <set transaction statement> as specified in Subclause 14.1 of the SQL'92 standard.

6. If the level of SQL transaction management support is Constraints, then this profile includes support for the <set constraints mode statement> as specified in Subclause 14.2 of the SQL'92 standard.

7. If the optional extension for SQL/XA is specified, then the implementation shall support the SQL specializaion of the X/Open XA interface specification (see Section 5.6 above and [12]).

Connection Management Rules

1. If the Module, Embedded SQL, or Direct Invocation binding styles are specified, and if the level of SQL data manipulation language support is Full DML, then the SQL/ERI Read-Write Server profile requires support for <SQL connection statement>s as specified in Clause 15, "Connection management", of the SQL'92 standard. If the level of SQL data manipulation language support is anything other than Full SQL, then there is no requirement to support any <SQL connection statement> for these binding styles.

2. If the SQL/CLI binding style is specified, then the requirements of the routines for connection management (i.e. Connect, Disconnect) as specified in the SQL/CLI specification [10] apply to this profile.

3. If the RDA/SQL binding style is specified, then the requirements of RDA Dialogue Management and RDA Resource Handling as specified in the RDA standard [9], with implementor agreements specified in [16], apply to this profile.

Session Management Rules

1. If the indicated SQL session management support is "No Session", then SQL session management is not supported for any binding style. Otherwise, for each feature identified, this profile supports the requirements of that feature as identified in Clause 16, "Session management", of the SQL'92 standard. If the indicated SQL session management support is "All Session", then all features of Clause 16 are supported in this profile.

2. If SQL data manipulation language support specifies Transitional DML or above, then support for "SET SCHEMA <unqualified schema name>" is implicit in this profile.

3. If SQL data manipulation language support specifies Intermediate DML or above, then support for "SET SCHEMA <unqualified schema name>", SET SESSION AUTHORIZATION, and SET TIME ZONE is implicit in this profile.

4. If SQL data manipulation language support specifies Full DML, then support for "All Session" is implicit in this profile.

Dynamic SQL and Diagnostics Management Rules

1. If SQL data manipulation language support specifies Minimal DML or Entry SQL, then support for SQL statements in Clause 17, "Dynamic SQL", and Clause 18, "Diagnostics management", is not required.

2. If SQL data manipulation language support specifies Transitional DML or above, then support for all provisions of Clause 17, "Dynamic SQL", and Clause 18, "Diagnostics management", with any restrictions identified in the Leveling Rules for higher levels of DML, is required for all implementations that claim that level of DML support.

Information Schema Rules

1. If the level of SQL data manipulation language support claimed for the SQL/ERI Read-Write Server profile is Minimal DML or Entry DML, then support for the following Information Schema views, as specified in Clause 21, "Information Schema and Definition Schema", of the SQL'92 standard, is required:

   TABLES
   VIEWS
   COLUMNS
   TABLE_PRIVILEGES
   COLUMN_PRIVILEGES

   Note: See FIPS 127-2 Errata for handling "long" names.

2. If an SQL/ERI Server implementation at the Minimal SDL level or below chooses not to provide support for null values (see item 4 of Section 4.1), then it shall provide an implementation-defined conversion of would-be null values in Information Schema tables to an appropriate non-null value.

3. If the level of SQL data manipulation language support claimed for the SQL/ERI Read-Write Server profile is Transitional DML, then support for the following Information Schema views, as specified in Clause 21, "Information Schema and Definition Schema", of the SQL'92 standard, is required:

   TABLES
   VIEWS
   COLUMNS
   TABLE_PRIVILEGES
   COLUMN_PRIVILEGES
   USAGE_PRIVILEGES

   Note: See FIPS 127-2 Errata for handling "long" names.

4. If the level of SQL data manipulation language support claimed for the SQL/ERI Read-Write Server profile is Intermediate DML or Full DML, then an implementation conforming to this profile shall provide all of the Information Schema views required by the SQL'92 standard for Intermediate SQL or Full SQL, respectively. In many cases some of these tables may be empty, or trivial, but a conforming SQL/ERI Server at these SQL data manipulation language levels is required to support them to reflect an accurate picture of the implicit schema definition.

7.3 Object Identifiers for SQL/ERI Server profiles

The National Institute of Standards and Technology is the registration authority for the following node in the joint ISO/IEC and CCITT branch of the international object-identifier hierarchical name tree (see CCITT X.660 or ISO/IEC 9834-1):

{ joint-iso-ccitt (2) country (16) us (840) gov (101) sql (4) }

It is NIST's intention to use this node to register objects derived from NIST publications related to Database Language SQL, including profiles defined in this FIPS for SQL Environments. Other registered objects will include FIPS SQL profiles, FIPS SQL test suite reports, FIPS SQL certificates, client-side profiles in an SQL environment, SQL/ERI Client profiles, a register of SQL environments, and possibly others. For information on the current state of this register, including its on-line availability through an SQL/ERI Read-Only RDA Server interface, contact FIPS SQL registration authority personnel at telephone +1-301-975-3251 or send e-mail to LGallagher@nist.gov.

The following syntactic productions yield object identifiers for SQL-related objects. See Clause 3.2, "Notation", of ISO/IEC 9075:1992 for definition of the syntactic notation used.

```
<nist sql-related object identifier>  ::=  <nist sql-related provenance> <nist sql-related object>

<nist sql-related provenance>  ::=  <joint-iso-ccitt> <country> <us> <gov> <sql>

<joint-iso-ccitt>  ::=  2  |  joint-iso-ccitt  |  joint-iso-ccitt <left paren> 2 <right paren>
```

```
<country> ::= 16 | country | country <left paren> 16 <right paren>
<us> ::= 840 | us | us <left paren> 840 <right paren>
<gov> ::= 101 | gov | gov <left paren> 101 <right paren>
<sql> ::= 4 | sql | sql <left paren> 4 <right paren>

<nist sql-related object> ::=
      <fips 127 profile>
   | <fips 127 certificate>
   | <fips 193 client-side profile>
   | <fips 193 sql-eri-client profile>
   | <fips 193 sql-eri-server profile>
   | <nist sql test report>
   | <nist sql environment>

<fips 127 profile> ::= <fips 127 profile designator> <fips 127 profile identifier>
<fips 127 profile designator> ::= 1 | sql-profile <left paren> 1 <right paren>
<fips 127 profile identifier> ::= <SQL variant>
```

Note: <SQL variant> is defined in Clause 3.4, "Object identifier for SQL", of Reference [8], with the following extension to accommodate the FIPS SQL definition of Transitional SQL.

```
<SQL conformance> ::= <low> | <transitional> | <intermediate> | <high>
<transitional> ::= 3 | Transitional <left paren> 3 <right paren>

<fips 127 certificate> ::= <fips 127 certificate register> [ <fips 127 certificate identifier> ]
<fips 127 certificate register> ::= 2 | nist-sql-cert <left paren> 2 <right paren>
<fips 127 certificate identifier> ::= <years value> <certificate number>
<years value> ::= !!Defined in ISO/IEC 9075 to be an <unsigned integer>
<certificate number> ::= <unsigned integer>

<fips 193 client-side profile> ::= <fips 193 client profile designator> <fips 193 client profile identifier>
<fips 193 client profile designator> ::= 3 | env-client <left paren> 3 <right paren>
<fips 193 client profile identifier> ::= !! To be defined in a future NIST publication.

<fips 193 sql-eri-client profile> ::= <eri-client profile designator> <eri-client profile identifier>
<eri-client profile designator> ::= 4 | eri-client <left paren> 4 <right paren>
<eri-client profile identifier> ::= !! To be defined in a future NIST publication.

<fips 193 sql-eri-server profile> ::=
      <eri-server profile designator>
      <eri-server profile identifier>
      [ <fips 193 options list> ]

<eri-server profile designator> ::= 5 | eri-server <left paren> 5 <right paren>
<eri-server profile identifier> ::= !! See below.
<fips 193 options list> ::= !! See below.

<nist sql test report> ::= <nist sql test report register> [ <nist vsr identifier> ]
<nist sql test report register> ::= 6 | nist-vsr <left paren> 6 <right paren>
<nist vsr identifier> ::= <years value> <vsr number>
```

```
<years value> ::= !!Defined in ISO/IEC 9075 to be an <unsigned integer>
<vsr number> ::= <unsigned integer>
```

Note: If a NIST validation summary report (VSR) results in the award of a certificate for conformance to FIPS SQL, then the <years value> and the <certificate number> of an entry in the <fips 127 certificate> register will match the <years value> and the <vsr number>, respectively, of an entry in the <nist sql test report> register.

```
<nist sql environment> ::= <nist sql environment register> [ <sql environment identifier> ]
<nist sql environment register> ::=  7 | sql-env <left paren> 7 <right paren>
<sql environment identifier> ::= !! To be defined in a future NIST publication.

<eri-server profile identifier> ::=
    <eri module-ro server>
  | <eri module-rw server>
  | <eri embedded-ro server>
  | <eri embedded-rw server>
  | <eri direct-ro server>
  | <eri direct-rw server>
  | <eri cli-ro server>
  | <eri cli-rw server>
  | <eri rda-ro server>
  | <eri rda-rw server>
  | <eri cli-ro1 server>
  | <eri cli-rw1 server>
  | <eri rda-ro1 server>
  | <eri rda-rw1 server>
  | <eri cli-ro2 server>
  | <eri cli-rw2 server>
  | <eri rda-ro2 server>
  | <eri rda-rw2 server>
  | <eri cli-ro3 server>
  | <eri cli-rw3 server>
  | <eri rda-ro3 server>
  | <eri rda-rw3 server>
  | <eri cli-ro4 server>
  | <eri cli-rw4 server>
  | <eri rda-ro4 server>
  | <eri rda-rw4 server>

<eri module-ro server> ::= 1 | mod-ro <left paren> 1 <right paren>
<eri module-rw server> ::= 2 | mod-rw <left paren> 2 <right paren>
<eri embedded-ro server> ::= 3 | emb-ro <left paren> 3 <right paren>
<eri embedded-rw server> ::= 4 | emb-rw <left paren> 4 <right paren>
<eri direct-ro server> ::= 5 | dir-ro <left paren> 5 <right paren>
<eri direct-rw server> ::= 6 | dir-rw <left paren> 6 <right paren>
<eri cli-ro server> ::= 7 | cli-ro <left paren> 7 <right paren>
<eri cli-rw server> ::= 8 | cli-rw <left paren> 8 <right paren>
<eri rda-ro server> ::= 9 | rda-ro <left paren> 9 <right paren>
<eri rda-rw server> ::= 10 | rda-rw <left paren> 10 <right paren>
```

```
<eri cli-ro1 server> ::= 11 | cli-ro1 <left paren> 11 <right paren>
<eri cli-rw1 server> ::= 12 | cli-rw1 <left paren> 12 <right paren>
<eri rda-ro1 server> ::= 13 | rda-ro1 <left paren> 13 <right paren>
<eri rda-rw1 server> ::= 14 | rda-rw1 <left paren> 14 <right paren>
<eri cli-ro2 server> ::= 15 | cli-ro2 <left paren> 15 <right paren>
<eri cli-rw2 server> ::= 16 | cli-rw2 <left paren> 16 <right paren>
<eri rda-ro2 server> ::= 17 | rda-ro2 <left paren> 17 <right paren>
<eri rda-rw2 server> ::= 18 | rda-rw2 <left paren> 18 <right paren>
<eri cli-ro3 server> ::= 19 | cli-ro3 <left paren> 19 <right paren>
<eri cli-rw3 server> ::= 20 | cli-rw3 <left paren> 20 <right paren>
<eri rda-ro3 server> ::= 21 | rda-ro3 <left paren> 21 <right paren>
<eri rda-rw3 server> ::= 22 | rda-rw3 <left paren> 22 <right paren>
<eri cli-ro4 server> ::= 23 | cli-ro4 <left paren> 23 <right paren>
<eri cli-rw4 server> ::= 24 | cli-rw4 <left paren> 24 <right paren>
<eri rda-ro4 server> ::= 25 | rda-ro4 <left paren> 25 <right paren>
<eri rda-rw4 server> ::= 26 | rda-rw4 <left paren> 26 <right paren>
```

Note: Each of the first ten profile identifiers above identifies a family of profiles, with support for the specified binding alternative and the specified read-only or read-write alternative; an explicit profile is given by adding the optional <fips 193 options list>. In each case, an implementation of the profile is only required to satisfy the minimal conformance options specified herein, plus the explicitly specified options. The remaining profile identifiers identify explicit profiles as defined in Sections 7.4 and 7.5 below; they may also include the optional <fips 193 options list> to identify support for SQL features above the base requirements of that profile. Other profile identifiers and profiles may be added later, as appropriate. Requests to add new profiles to this register may be addressed to the NIST Computer Systems Laboratory, attention: SQL Environment profiles.

```
<fips 193 options list> ::= { <fips 193 option> } ...

<fips 193 option> ::=
    <dml option>
  | <sdl option>
  | <transaction option>
  | <isolation option>
  | <additional binding option>
  | <language option>
  | <session option>
  | <extension option>
  | <protocol option>
  | <rda option>

<dml option> ::=
    <minimal dml>
  | <entry dml>
  | <transitional dml>
  | <intermediate dml>
  | <full dml>

<minimal dml> ::= 110 | dml-min <left paren> 110 <right paren>
<entry dml> ::= 120 | dml-ent <left paren> 120 <right paren>
```

```
<transitional dml>  ::=  130  |  dml-tran <left paren> 130 <right paren>
<intermediate dml>  ::=  140  |  dml-int <left paren> 140 <right paren>
<full dml>  ::=  150  |  dml-full <left paren> 150 <right paren>

<sdl option>  ::=
      <no sdl>
  |   <minimal sdl>
  |   <entry sdl>
  |   <transitional sdl>
  |   <intermediate sdl>
  |   <full sdl>

<no sdl>  ::=  200  |  sdl-none <leftparen> 200 <right paren>
<minimal sdl>  ::=  210  |  sdl-min <left paren> 210 <right paren>
<entry sdl>  ::=  220  |  sdl-ent <left paren> 220 <right paren>
<transitional sdl>  ::=  230  |  sdl-tran <left paren> 230 <right paren>
<intermediate sdl>  ::=  240  |  sdl-int <left paren> 240 <right paren>
<full sdl>  ::=  250  |  sdl-full <left paren> 250 <right paren>

<transaction option>  ::=
      <no transaction>
  |   <commit-rollback>
  |   <transaction mode>
  |   <transaction isolation>
  |   <transaction diagnostics>
  |   <transaction constraints>

<no transaction>  ::=  300  |  tx-none <leftparen> 300 <right paren>
<commit-rollback>  ::=  310  |  tx-cr <left paren> 310 <right paren>
<transaction mode>  ::=  320  |  tx-tm <left paren> 320 <right paren>
<transaction isolation>  ::=  330  |  tx-ti <left paren> 330 <right paren>
<transaction diagnostics>  ::=  340  |  tx-td <left paren> 340 <right paren>
<transaction constraints>  ::=  350  |  tx-tc <left paren> 350 <right paren>

<isolation option>  ::=
      <read uncommitted>
  |   <read committed>
  |   <repeatable read>
  |   <serializable>

<read uncommitted>  ::=  410  |  isol-ru <left paren> 410 <right paren>
<read committed>  ::=  420  |  isol-rc <left paren> 420 <right paren>
<repeatable read>  ::=  430  |  isol-rr <left paren> 430 <right paren>
<serializable>  ::=  440  |  isol-sr <left paren> 440 <right paren>

<additional binding option>  ::=
      <no additions>
  |   <plus module>
  |   <plus embedded>
  |   <plus direct>
```

```
 | <plus cli>
 | <plus rda>

<no additions> ::= 500 | bind-none <left paren> 500 <right paren>
<plus module> ::= 510 | bind-mod <left paren> 510 <right paren>
<plus embedded> ::= 520 | bind-emb <left paren> 520 <right paren>
<plus direct> ::= 530 | bind-dir <left paren> 530 <right paren>
<plus cli> ::= 540 | bind-cli <left paren> 540 <right paren>
<plus rda> ::= 550 | bind-rda <left paren> 550 <right paren>

<language option> ::=
   <language not applicable>
 | <embedded ada>
 | <embedded c>
 | <embedded cobol>
 | <embedded fortran>
 | <embedded mumps>
 | <embedded pascal>
 | <embedded pl1>
 | <module ada>
 | <module c>
 | <module cobol>
 | <module fortran>
 | <module mumps>
 | <module pascal>
 | <module pl1>
 | <cli ada>
 | <cli c>
 | <cli cobol>
 | <cli fortran>
 | <cli mumps>
 | <cli pascal>
 | <cli pl1>
 | <samedl via module mapping>
 | <samedl via embedded mapping>
 | <samedl via effects>

<language not applicable> ::= 600 | lang-none | lang-none <left paren> 600 <right paren>
<embedded ada> ::= 621 | emb-ada <left paren> 621 <right paren>
<embedded c> ::= 622 | emb-c <left paren> 622 <right paren>
<embedded cobol> ::= 623 | emb-cob <left paren> 623 <right paren>
<embedded fortran> ::= 624 | emb-for <left paren> 624 <right paren>
<embedded mumps> ::= 625 | emb-mum <left paren> 625 <right paren>
<embedded pascal> ::= 626 | emb-pas <left paren> 626 <right paren>
<embedded pl1> ::= 627 | emb-pl1 <left paren> 627 <right paren>
<module ada> ::= 641 | mod-ada <left paren> 641 <right paren>
<module c> ::= 642 | mod-c <left paren> 642 <right paren>
<module cobol> ::= 643 | mod-cob <left paren> 643 <right paren>
<module fortran> ::= 644 | mod-for <left paren> 644 <right paren>
<module mumps> ::= 645 | mod-mum <left paren> 645 <right paren>
```

```
<module pascal>  ::=  646  |  mod-pas <left paren> 646 <right paren>
<module pl1>  ::=  647  |  mod-pl1 <left paren> 647 <right paren>
<cli ada>  ::=  661  |  cli-ada <left paren> 661 <right paren>
<cli c>  ::=  662  |  cli-c <left paren> 662 <right paren>
<cli cobol>  ::=  663  |  cli-cob <left paren> 663 <right paren>
<cli fortran>  ::=  664  |  cli-for <left paren> 664 <right paren>
<cli mumps>  ::=  665  |  cli-mum <left paren> 665 <right paren>
<cli pascal>  ::=  666  |  cli-pas <left paren> 666 <right paren>
<cli pl1>  ::=  667  |  cli-pl1 <left paren> 667 <right paren>
<samedl via module mapping>  ::=  610  |  samedl-mod <left paren> 610 <right paren>
<samedl via embedded mapping>  ::=  611  |  samedl-emb <left paren> 611 <right paren>
<samedl via effects>  ::=  612  |  samedl-eff <left paren> 612 <right paren>

<session option>  ::=
      <no session>
   |  <set catalog>
   |  <set schema>
   |  <set names>
   |  <set session authorization>
   |  <set time zone>
   |  <all session>

<no session>  ::=  700  |  sess-none <leftparen> 700 <right paren>
<set catalog>  ::=  710  |  sess-cat <left paren> 710 <right paren>
<set schema>  ::=  720  |  ses-schema <left paren> 720 <right paren>
<set names>  ::=  730  |  sess-names <left paren> 730 <right paren>
<set session authorization>  ::=  740  |  sess-auth <left paren> 740 <right paren>
<set time zone>  ::=  750  |  sess-tz <left paren> 750 <right paren>
<all session>  ::=  760  |  sess-all <left paren> 760 <right paren>

<extension option>  ::=
      <no extensions>
   |  <sql features table>
   |  <executable sql psm>
   |  <definable sql psm>
   |  <sql multimedia full text>
   |  <sql multimedia spatial>
   |  <sql multimedia general>
   |  <complex number extension>
   |  <vector extension>
   |  <numerics extension>
   |  <boolean extension>
   |  <abstract data type extension>
   |  <object data management>
   |  <sql xa-interface routines>

<no extensions>  ::=  800  |  ext-none <left paren> 800 <right paren>
<sql features table>  ::=  810  |  ext-features <left paren> 810 <right paren>
<executable sql psm>  ::=  820  |  psm-call <left paren> 820 <right paren>
<definable sql psm>  ::=  830  |  psm-all <left paren> 830 <right paren>
```

```
<sql multimedia: full text>  ::=  840  |  ext-fulltext <left paren> 840 <right paren>
<sql multimedia: spatial>  ::=  850  |  ext-spatial <left paren> 850 <right paren>
<sql multimedia: general>  ::=  860  |  ext-general <left paren> 860 <right paren>
<complex number extension>  ::=  861  |  ext-complex <left paren> 861 <right paren>
<vector extension>  ::=  862  |  ext-vector <left paren> 862 <right paren>
<numerics extension>  ::=  863  |  ext-numerics <left paren> 863 <right paren>
<boolean extension>  ::=  864  |  ext-boolean <left paren> 864 <right paren>
<abstract data type extension>  ::=  870  |  ext-adt <left paren> 840 <right paren>
<object data management extensions>  ::=  880  |  ext-odm <left paren> 850 <right paren>
<sql xa-interface routines>  ::=  890  |  sql-xa <left paren> 850 <right paren>

<protocol option>  ::=
    <protocol not applicable>
  | <minimal osi>
  | <full stack osi>
  | <ietf rfc 1006>
  | <other transport>

<protocol not applicable>  ::=  900  |  rda-none  |  rda-none <left paren> 900 <right paren>
<minimal osi>  ::=  910  |  prot-mosi <left paren> 910 <right paren>
<full stack osi>  ::=  920  |  prot-osi <left paren> 920 <right paren>
<internet rfc 1006>  ::= 930  |  prot-1006 <left paren> 930 <right paren>
<other transport>  ::=  940  |  prot-other <left paren> 940 <right paren>

<rda option>  ::=
  | <rda stored execution>
  | <rda status>
  | <rda cancel>
  | <rda tp context>

<rda stored execution>  ::=  960  |  rda-stored <left paren> 960 <right paren>
<rda status>  ::=  970  |  rda-status <left paren> 970 <right paren>
<rda cancel>  ::=  980  |  rda-cancel <left paren> 980 <right paren>
<rda tp context>  ::=  990  |  rda-tp <left paren> 990 <right paren>
```

7.4 Specific SQL/ERI CLI Server profiles

The SQL/CLI binding style is expected to be very popular in local area networks consisting of client applications and server data repositories. The following object identifiers, whose syntax is defined in Section 7.3 above, identify specific SQL/ERI CLI Server profiles that will be popular for Federal procurements over the next several years.

SQL/ERI Read-Only CLI Server Profile - Level 1

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) cli-ro1(11) }

Profile: 1) Minimal DML 2) No SDL 3) Commit-Rollback 4) Read Committed 5) SQL/CLI 6) C 7) No Session 8) No Extensions 9) Not Applicable 10) Not Applicable

SQL/ERI Read-Only CLI Server Profile - Level 2

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) cli-ro2(15) }

Profile: 1) Transitional DML 2) No SDL 3) Commit-Rollback 4) Read Committed 5) SQL/CLI 6) C 7) Set Schema 8) SQL Features 9) Not Applicable 10) Not Applicable

SQL/ERI Read-Only CLI Server Profile - Level 3

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) cli-ro3(19) }

Profile: 1) Intermediate DML 2) No SDL 3) Commit-Rollback 4) Read Committed 5) SQL/CLI 6) C 7) {Set Schema, Set Session Authorization, Set Time Zone} 8) SQL Features 9) Not Applicable 10) Not Applicable

SQL/ERI Read-Only CLI Server Profile - Level 4

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) cli-ro4(23) }

Profile: 1) Intermediate DML 2) No SDL 3) Commit-Rollback 4) Read Committed 5) SQL/CLI 6) C 7) All Session 8) {SQL Features, Executable SQL/PSM} 9) Not Applicable 10) Not Applicable

SQL/ERI Read-Write CLI Server Profile - Level 1

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) cli-rw1(12) }

Profile: 1) Minimal DML 2) No SDL 3) Commit-Rollback 4) Serializable 5) SQL/CLI 6) C 7) No Session 8) No Extensions 9) Not Applicable 10) Not Applicable

SQL/ERI Read-Write CLI Server Profile - Level 2

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) cli-rw2(16) }

Profile: 1) Transitional DML 2) No SDL 3) Transaction Isolation 4) Serializable 5) SQL/CLI 6) C 7) Set Schema 8) SQL Features 9) Not Applicable 10) Not Applicable

SQL/ERI Read-Write CLI Server Profile - Level 3

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) cli-rw3(20) }

Profile: 1) Intermediate DML 2) No SDL 3) Transaction Isolation 4) Serializable 5) SQL/CLI 6) C 7) {Set Schema, Set Session Authorization, Set Time Zone} 8) SQL Features 9) Not Applicable 10) Not Applicable

SQL/ERI Read-Write CLI Server Profile - Level 4

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) cli-rw4(24) }

Profile: 1) Intermediate DML 2) Minimal SDL 3) Transaction Isolation 4) Serializable 5) SQL/CLI 6) C 7) All Session 8) {SQL Features, Executable SQL/PSM} 9) Not Applicable 10) Not Applicable

7.5 Specific SQL/ERI RDA Server profiles

A very beneficial use of SQL-related object identifiers is in "level of service" negotiations that may take place when one open system opens a dialogue with another open system. The following object identifiers, whose syntax is defined in Section 7.3 above, identify specific SQL/ERI RDA/SQL-Server profiles that will be popular for Federal procurements over the next several years.

SQL/ERI Read-Only RDA Server Profile - Level 1

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) rda-ro1(13) }

Profile: 1) Minimal DML 2) No SDL 3) Commit-Rollback 4) Read Committed 5) RDA/SQL 6) No Language 7) No Session 8) No Extensions 9) Internet RFC 1006 10) No RDA Options

SQL/ERI Read-Only RDA Server Profile - Level 2

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) rda-ro2(17) }

Profile: 1) Transitional DML 2) No SDL 3) Commit-Rollback 4) Read Committed 5) RDA/SQL 6) No Language 7) Set Schema 8) SQL Features 9) Internet RFC 1006 10) {RDA Status, RDA Cancel}

SQL/ERI Read-Only RDA Server Profile - Level 3

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) rda-ro3(21) }

Profile: 1) Intermediate DML 2) No SDL 3) Commit-Rollback 4) Read Committed 5) RDA/SQL 6) No Language 7) {Set Schema, Set Session Authorization, Set Time Zone} 8) SQL Features 9) Internet RFC 1006 10) {RDA status, RDA cancel}

SQL/ERI Read-Only RDA Server Profile - Level 4

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) rda-ro4(25) }

Profile: 1) Intermediate DML 2) No SDL 3) Commit-Rollback 4) Read Committed 5) RDA/SQL 6) No Language 7) All Session 8) {SQL Features, Executable SQL/PSM} 9) Internet RFC 1006 10) {RDA Status, RDA Cancel, RDA Stored Execution}

SQL/ERI Read-Write RDA Server Profile - Level 1

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) rda-rw1(14) }

Profile: 1) Minimal DML 2) No SDL 3) Commit-Rollback 4) Serializable 5) RDA/SQL 6) No Language 7) No Session 8) No Extensions 9) Internet RFC 1006 10) No RDA Options

SQL/ERI Read-Write RDA Server Profile - Level 2

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) rda-rw2(18) }

Profile: 1) Transitional DML 2) No SDL 3) Transaction Isolation 4) Serializable 5) RDA/SQL 6) No Language 7) Set Schema 8) SQL Features 9) Internet RFC 1006 10) {RDA Status, RDA Cancel}

SQL/ERI Read-Write RDA Server Profile - Level 3

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) rda-rw3(22) }

Profile: 1) Intermediate DML 2) No SDL 3) Transaction Isolation 4) Serializable 5) RDA/SQL 6) No Language 7) {Set Schema, Set Session Authorization, Set Time Zone} 8) SQL Features 9) Internet RFC 1006 10) {RDA Status, RDA Cancel}

SQL/ERI Read-Write RDA Server Profile - Level 4

Object Identifier: { joint-iso-ccitt(2) country(16) us(840) gov(101) sql(4) eri-server(5) rda-rw4(26) }

Profile: 1) Intermediate DML 2) Minimal SDL 3) Transaction Isolation 4) Serializable 5) RDA/SQL 6) No Language 7) {Set Schema, Set Session Authorization, Set Time Zone} 8) {SQL Features, Executable SQL/PSM 9) Internet RFC 1006 10) {RDA Status, RDA Cancel, RDA Stored Execution}

8. Applicability

This standard is applicable in any situation where it is desirable to integrate a client-side productivity tool or a server-side data repository into an SQL environment. It is a non-mandatory standard that may be invoked on a case-by-case basis subject to the integration objectives of the procuring department or agency. It is particularly suitable for specifying limited SQL interfaces to legacy databases or to specialized data repositories not under the control of a full-function SQL database management system. It can be used along with other procurement information to specify SQL interface requirements for a wide range of data management procurements.

One special area of application envisioned for this standard is Electronic Commerce, a National Challenge Application area of the National Information Infrastructure. The primary objective of Electronic Commerce is to integrate communications, data management, and security services in a distributed processing environment, thereby allowing business applications within different organizations to interoperate and exchange information without human intervention. At the data management level, electronic commerce requires a logically integrated database of diverse data stored in geographically separated data banks under the management and control of heterogeneous database management systems. An over-riding requirement is that these diverse data managers be able to communicate with one another and provide shared access to data and data operations and methods under appropriate security, integrity, and access control mechanisms. FIPS SQL provides a powerful database language for data definition, data manipulation, and integrity management to satisfy many of these requirements. It is unrealistic to expect that every data manager involved in electronic commerce will conform to even the Entry SQL level of the FIPS SQL standard; however, it is not unrealistic to require that they support a limited SQL interface, even a read-only interface, provided by one of the SQL/ERI Server profiles. New procurements to add components to the National Information Infrastructure, or to upgrade existing components, can define the necessary SQL schemas and point to appropriate SQL/ERI Server profiles as procurement requirements.

This standard may also be applicable, on a case-by-case basis, in many of the following areas.

8.1 Legacy databases. A legacy database is an already installed database managed by a non-standard database management system (DBMS). This may include hierarchical and network databases popular in the 1970 to 1990 timeframe, non-standard relational databases, other non-standard data repositories, or even home-grown databases developed by Federal departments and agencies over the past three decades as enhancements to proprietary file access mechanisms. A Federal procurement might solicit development of a new interface to a legacy database that supports one of the SQL/ERI Server profiles. Modern applications could then access the legacy data using standard SQL statements.

8.2 Full-Text document databases. A document database is a database specialized to optimize the handling of text and text operations. Traditional SQL systems have been weak in this area because the SQL 1986 and 1989 standards had very minimal requirements for text management. The SQL 1992 standard has enhanced requirements for character sets and character string operations, but still falls short of the text handling requirements of text-intensive applications. An emerging standard for SQL/MM Part 2: Full-Text [13], expected sometime after 1996, is addressing more sophisticated user requirements for SQL management of text. In the meantime, some SQL vendors are offering Full-Text extensions and some Full-Text document database vendors are offering limited SQL interfaces. A Federal procurement for a Full-Text document database management system may stipulate, in addition to its requirements for text management, conformance to one of the SQL/ERI Server profiles as either a mandatory or a desirable requirement of that procurement.

8.3 Geographic Information Systems. A geographic information database is a database structured to optimize the handling of spatial data and spatial operations, especially traditional earth science information such as maps or physical topography, as well as any social, economic, or demographic data that is spatially referenced. Traditional SQL systems have been weak in this area because the existing SQL 1992 standard does not require support for constructor data types such as lists, sets, and arrays, often required as the basis of definition for more complex spatial data structures. An emerging standard for SQL/MM Part 3: Spatial [13], expected sometime after 1996, is addressing more sophisticated user requirements for SQL definition and management of spatial data types and spatial operations. In the meantime, some SQL vendors are offering Spatial extensions and some Geographic Information System (GIS) vendors are offering limited SQL interfaces. Since many Federal applications require integration of traditional government data (e.g. Census, Labor statistics, Economic Indicators, Meteorological, Health Care) with Geographic Information Systems, a Federal procurement for a Geographic Information System (GIS) may stipulate requirements for two-way integration capabilities, both to read data from an SQL database into the GIS for incorporation into spatial objects and to provide SQL application access to the value-added result. The first of these requirements is discussed in Section 10.1 below, the second could be addressed in a GIS procurement by specifying mandatory support for application access via one of the SQL/ERI Server profiles specified herein.

8.4 Bibliographic information retrieval. A bibliographic database is a database that supports the requirements of American National Standard Z39.50-1992, a standard for information retrieval developed by the National Information Standards Organization (NISO). Appendix C of that standard specifies a list of approximately 75 attributes for each database of documents. Each attribute assumes an ASCII character string value. The 1992 version of Z39.50 specifies application layer protocols for read-only queries over one or more databases each supporting the listed attributes. The results of a query are returned to the user using a NISO Record Syntax specified by other NISO standards. Annex E of Z39.50 identifies 19 possible formats for Record Syntax. If it is desirable to integrate such bibliographic databases into an SQL environment, then a procurement might solicit development of a new interface to the bibliographic database that supports one of the SQL/ERI Read-Only Server profiles, probably with an RDA/SQL binding style to take advantage of the underlying communications protocols already specified by Z39.50. Each NISO database would map to an SQL table and each bibliographic attribute would map to a column of the table. Each NISO supported Record Syntax would also map to an SQL Table with each field of the record equivalent to a column of the table. SQL applications could then access the bibliographic data using standard SQL statements.

8.5 Object database interfaces. An object database is a database managed by an object database management system (ODBMS). Object database management systems may implement non-relational data models and thus may have difficulty supporting full-function SQL requirements for nested subqueries, multi-table joins, Group-By set functions, derived columns in a Select list, value-based referential integrity, or other relational model features. On the other hand, ODBMS's may offer advanced features of object models that are rarely supported by relational implementations. These features might include user-defined abstract data types (ADTs), object identifiers, methods, inheritance, polymorphism, encapsulation, and other object-oriented enhancements. Because of their close relationship with an object-oriented programming language (e.g., C++ or Smalltalk), ODBMS's often make it easy to integrate user-written routines into database operations. Often object DBMS's are used for specialized applications with complex data structures and application-specific methods on those structures. The next version of the SQL standard, expected sometime after 1996 (see [12]), will likely incorporate many of these object database features into the SQL language. In the meantime, some SQL vendors are enhancing their products to support user-defined ADT's and other object capabilities and some object database vendors are supporting robust SQL interfaces to their products. The ODMG-93 specification sponsored by the Object Database Management Group, an informal consortium of approximately ten object database vendors, points to an Object Query Language that has many of the same features as SQL. Thus it makes sense to support an SQL interface, at least a Read Only SQL interface, to these products. The specialized methods appropriate to each object could be viewed as callable SQL functions and procedures. If it is desirable to integrate an object database into an SQL environment, then an original procurement for ODBMS software, or a follow-on new procurement for an SQL interface, could point to one of the SQL/ERI Server profiles as a mandatory or desirable procurement requirement. The vendor of an object database product may automatically support SQL views of its collection types, or a procurement could specify an exact mapping from object collection types to tables that must be supported as a procurement requirement. In either case, the result is that an SQL application would have access to objects and methods using standard SQL syntax and a standard SQL binding style.

8.6 Federal data distribution. A number of Federal agencies support public access to federally maintained data either by maintaining a public database of information or by distributing data on floppy disks, CD-ROMs, or magnetic tape. All Federal departments and Agencies, but especially those with responsibility for providing public data (e.g. Agriculture, Health and Human Services, Census, NASA, NOAA, BLS), will have increasing requests for convenient public access to data. Even individuals will be requesting additional electronic access to their individual data maintained by various Federal agencies (e.g. IRS, Social Security, Medicare). An existing SQL database could provide public access by supporting a Remote Database Access (RDA) or Direct Invocation user interface with appropriate access control restrictions; a non-SQL legacy database could support one of the SQL/ERI Read Only Server profiles for RDA or Direct Invocation in addition to the ERI profiles supported for internal goverment use (see 8.1 above). When goverment data is distributed on disk or CD-ROM, it is often accompanied by software, executable on different workstations, to provide convenient views of the data using a graphical user interface (GUI). At the present time, the government must provide software executable on a number of different workstation platforms with different GUI capabilities and requirements. In the future, the government might reduce its software development efforts by providing an appropriate SQL/ERI front-end, usually with the SQL Call Level Interface option, for each workstation family (e.g. DOS, Macintosh, Unix). Such software should be available commercially in the near term. End users could then use their favorite client-side SQL environment tool to browse the data and present it using report-writer, graphical viewer, or hypermedia presentation techniques.

8.7 Operating system file interface. Sometimes an SQL application desires access to the file characteristics of files stored in an operating system's file store. For example, the POSIX standard (FIPS PUB 151) requires that the following file characteristics be maintained for each file in the persistent file directory: file mode, file serial number, id of device containing file, number of links, id of file owner, id of file group, file size in bytes, time of last access, time of last data modification, and time of last file status change. If it is desirable that this information, as well as file name, file extension, and file usage characteristics, be accessible to SQL applications in the same operating system environment, then an original procurement for a POSIX compliant operating system, or a follow-on new procurement for SQL query access, could point to one of the SQL/ERI Read Only Server profiles supporting an SQL schema description of the desired attributes as a procurement requirement.

8.8 Open system directory interface. Sometimes an SQL application desires access to the directory information related to all of the workstations accessible in a local area network (LAN), or a wide area network (WAN). For example, the X.500 (Directory) standards supported by the International Telecommunications Union (ITU) require that certain explicit directory information be accessible at each site, and part of the IEEE POSIX standard specifies an application program interface (API) for these directory services. In addition to supporting the required X.500 communications protocols and the POSIX API, a number of implementations are also making this information available as an SQL database. If it is desirable that this information be accessible to SQL applications in the same open systems environment, then an original procurement for X.500 compatble communications products, or a follow-on new procurement for SQL query access to directory information, could point to one of the SQL/ERI Read Only Server profiles supporting an SQL schema description of the desired directory information as a procurement requirement.

8.9 Electronic mail repositories. Most electronic mail systems use the host file system to maintain e-mail documents in a user's file space and to maintain a log of e-mail activity in its own file space. Sometimes an electronic mail usability tool, procured separately from the e-mail system itself, will require SQL query access to these files, either relative to a specific individual user or to a group of users. If such SQL access is desirable, then the original procurement for the e-mail system, or a new procurement just for a follow-on SQL interface, might specify an SQL schema of metadata that must be maintained and an appropriate SQL/ERI profile for limited SQL access. Under this scenario, there might exist an underlying base table with column attributes such as: mail-id, owner, title, subject, to/from-address, linked-to-mail-id, timestamp, length, confirmation, content. Each e-mail user might own a table view that identifies all instances from the base table of e-mail sent or received by that user. Users would then be able to grant Select or Delete permissions on their own views to other users, thereby maintaining privacy in the database while supporting flexible multi-user access for the usability tools.

8.10 CASE tool repositories. Many computer-aided software engineering (CASE) tools operate similarly to the e-mail usability tools described above. Each CASE tool may use the underlying file system to maintain persistent data pertinent to its application domain. It is unrealistic to expect every such data repository to handle full function SQL statements. Instead, such repositories can be integrated into an SQL environment if they provide even the simplest SQL/ERI minimal SQL interface. The CASE tool can make simple external SQL views available to more powerful SQL processors, and those processors can, in turn, provide the full power and flexibility of the SQL language to end user applications. If integration of CASE tools into an SQL environment is desirable, then the original CASE tool procurement, or a new follow-on SQL interface procurement, can require implementation of an appropriate SQL/ERI Server profile over an SQL schema specified by that procurement.

8.11 XBase repositories. XBase is an emerging ANSI standard specification for a computer database and graphical presentation language popular on personal computers. The project description for this proposed new language standard specifies the desirability of an SQL interface so that XBase users can have access to standard conforming SQL databases. In this situation, XBase may be regarded as a client-side product in an SQL environment. The XBase language also provides data definition and data management capabilities for persistent tables of data in personal computer environments. With the advent of computer networks, it is often desirable to consider data on individual personal computers or workstations as stand-alone data repositoires in an integrated data processing environment. In this situation, XBase implementations may be regarded as a server-side products in an SQL environment. If it is desirable to integrate data from individual XBase repositories into a distributed SQL processing environment, then the original XBase procurement, or a follow-on SQL interface procurement, could specify an appropriate SQL/ERI Read-Only Server or SQL/ERI Read-Write Server profile for convenient access from other applications in the SQL environment. All XBase tables and columns would map directly to SQL tables and columns, with either XBase data types or SQL data types for columns, as appropriate. If XBase data types are used, then the SQL view would present those columns as SQL Domains and would provide special callable functions for XBase operations on those data types. All applications could then depend upon a single, standard language for access to persistent data, and the SQL Call Level Interface (SQL/CLI) and/or SQL Remote Database Access (SQL/RDA) could be used for interoperability.

8.12 C++ sequence class repositories. The emerging ANSI standard for the C++ programming language specifies search capabilities for Sequence classes in its Standard Template Library. Such sequence classes may sometimes be considered as persistent, data repositories. Often a small, or isolated, database application will use C++ Sequence classes for data management. Since the template library for Sequence classes provides search capabilities analogous to simple SQL predicates, it may be possible to integrate C++ Sequence class repositories into an SQL environment with a minimum of development effort. If such integration is desirable, then either in-house development, or inexpensive commercial software, may provide the appropriate SQL/ERI interface. All applications in the SQL environment would then have homogeneous access to the C++ data repository using standard SQL language statements.

8.13 Object Request Broker repositories. An Object Request Broker (ORB) provides user access to a collection of objects that have public interface definitions. In the X/Open and Object Management Group's architecture for a common object request broker (CORBA), these interface definitions are maintained in a database, called the Interface Repository, that is analogous in intent to an SQL Information Schema. If integration of such interface repositories, or integration of the public interfaces themselves, into an SQL environment is desirable, then the original ORB procurement, or a new follow-on SQL interface procurement, can require implementation of an appropriate SQL/ERI Server profile over an SQL schema specified by that procurement.

8.14 Real-Time database interface. A real-time database is a database optimized for access speed and specialized for handling data structures prevalent in radar systems, aircraft guidance, and satellite transmission. Often real-time databases are specially developed in a systems programming language for performance efficiency. Application-specific data structures are then stored in collection types, usually analogous to the C++ sequence classes mentioned above, with SQL cursor-like operations (e.g., Fetch, Insert) for moving structure instances to and from the database. The Real-Time Object Manager (RTOM) supporting Navy command and control systems is an example of such a system. If it is desirable to integrate real-time databases into an SQL environment, then either the original real-time database procurement, or an explicit follow-on for an SQL interface, could point to an appropriate SQL/ERI Server profile supporting an application-specific schema as a procurement requirement.

8.15 Internet file repositories. The Internet Engineering Task Force (IETF), the engineering group responsible for developing Internet Society applications, has published file access specifications for Wide Area Information System (WAIS) and World Wide Web (WWW). Files stored in these repositories in standard text formats, or in multimedia formats for hypertext, images, audio, motion pictures, or music, are then accessible on the Internet using File Transfer Protocols (FTP) or other client-side file access and presentation tools such as GOPHER and MOSAIC. Since the metadata for these files is very similar to the metadata of a file system (see 8.7 above) and the text content is often subject to bibliographic information retrieval (see 8.4 above), it may be possible to integrate these repositories into an SQL environment with a minimum of development effort, possibly even using tools already available for integrating file systems and bibliographic retrieval systems. If it is desirable to integrate WAIS and Web repositories into an SQL environment, then either ad hoc academic development, or inexpensive commercial software, may provide the appropriate SQL/ERI Server interface. Use of SQL/ERI profiles may provide needed access controls and integrity constraints on the repository side, as well as homogeneous access from SQL applications in remote client-side sites.

9. Conformance Testing

The National Institute of Standards and Technology offers a formal testing service for SQL implementations in support of its federally mandated program of Federal Information Processing Standards (FIPS). The NIST SQL Test Suite was first developed in 1988 to support testing of FIPS PUB 127, the standard for Database Language SQL. This test suite has evolved over the years to support new interfaces and other enhancements to the SQL standard as they are adopted by national and international SQL standardization groups. Version 4.0 of the NIST SQL Test Suite, available since June 1993, contains tests for the Entry SQL level of the 1992 standard; future versions will test other levels of conformance as well as new interface standards such as the Call Level Interface (CLI) and Remote Database Access (RDA). It is expected that this test suite will be modified as needed to enable conformance testing of the various SQL/ERI Server profiles specified herein and that validation reports for tested SQL/ERI products will be published quarterly in the NIST Validated Products List. The following sections discuss features of the NIST SQL Test Suite and how it might be modified for future validation of SQL/ERI products.

9.1 NIST SQL Test Suite

The purpose of the NIST SQL Test Suite is to help evaluate conformance of SQL implementations to mandatory requirements of FIPS PUB 127. This test suite is used as part of the formal testing service for SQL that issues Certificates of Validation for tested products passing all required tests. A Validation Summary Report is issued for all implementations tested. A Validation Summary Report documents, to the extent tested, the implementation's conformance to FIPS PUB 127-2. NIST publishes a quarterly register, Validated Products List, showing SQL implementations which hold current Certificates of Validation and registered Validation Summary Reports.

The NIST SQL Test Suite was first made available to the public in August 1988 as Version 1.1, and included tests for three programming languages: COBOL, FORTRAN, and C. In May 1989 the test suite was enlarged and released as Version 1.2, and included tests for additional SQL features, as well as tests for Embedded SQL Pascal and a Pascal interface to Module Language SQL. Version 2.0, completed about a year later, contained additional tests as well as the support system (software utilities) to administer the validation process. Continuing standardization work for SQL resulted in a revised SQL standard, ANSI X3.135-1989, published December 1989. This revised standard contained integrity enhancements for SQL, including referential integrity, default values for columns, and check clauses. FIPS PUB 127 was revised to specify these new integrity features as an optional module which federal agencies could either require or (by default) not require in a procurement. Version 2.0 of the test suite also contained a set of tests to validate conformance to this optional module. In the same time frame, ANSI X3.168-1989 standardized the embedding of SQL in programming languages (Ada, C, COBOL, FORTRAN, Pascal and PL/I). The first release of the NIST SQL Test Suite contained tests for Embedded SQL, in anticipation of this standard. Since numerous implementations of Embedded SQL already existed, prior to standardization, NIST hoped that the early availability of tests for that interface would hasten the conformance of implementations to the revised FIPS PUB 127-1. Version 3.0 provided test suites for Ada bindings to SQL and also tests for the errata in the SQL Information Bulletin SQLIB-1. ANSI X3.135-1992, the 1992 revision of the SQL standard, represents a major enhancement in SQL functionality. Conformance to FIPS PUB 127-2, Entry SQL, requires additional capabilities from an SQL implementation beyond those required for minimal conformance to FIPS PUB 127-1. The Integrity Enhancement Feature is now mandatory. Support for the following additional features is now required: SQLSTATE status codes, delimited identifiers, renaming columns, commas in parameter lists, SQL Errata against ANSI X3.135- 1989 (approved after publication of SQLIB-1).

Version 4.0 of the NIST SQL Test Suite provides tests for all the features in Entry SQL. Version 2.0 was used in the formal testing service offered by NIST which opened in April 1990. Version 3.0 became the official version of the test suite in July 1992, and Version 4.0 became the official version in January 1994.

The NIST SQL Test Suite provides ten programming language test suite types: Embedded (preprocessor) SQL Ada, Embedded SQL C, Embedded SQL COBOL, Embedded SQL FORTRAN, Embedded SQL Pascal, Module Language Ada, Module Language C, Module Language COBOL, Module Language FORTRAN, and Module Language Pascal. NIST also provides an Interactive Direct SQL test suite type to test interactive invocation of SQL statements as defined in FIPS 127-2. The original test programs were developed in Embedded (preprocessor) SQL for the C language. The design objective for the test programs was to provide a simple test for every general rule in the standard and to cover fully all SQL syntax. Ada, COBOL, FORTRAN, and Pascal test routines, as well as module language test routines, were generated by software (written by NIST) from the original Embedded SQL C language. The original Embedded SQL C Language tests are very simple, using only a carefully restricted subset of the C Language. Otherwise, it would be technically infeasible to translate these tests into the other programming languages. The Interactive Direct SQL test files were created by extracting SQL statements from the Embedded SQL C programs. Test cases were reworked to avoid reference to cursors and host variables. The resulting text files were annotated with comments describing the test and the expected results required for a "pass."

Each test is designed to be short and simple, exercising as little of the host language as possible. The host language compiler should be validated separately to ensure that it conforms to the applicable standards. The use of complex host language code in SQL conformance programs would make tests difficult to understand and would make it more difficult to resolve questions of interpretation of the SQL standard. Most of the tests involve 3 small tables containing a total of 23 rows. The data types of columns in these tables are either character string or integer, so the tests will work across all these programming languages. Other tables are used to test approximate numeric and scaled exact numeric data types. Additional tests have been written to cover the data type variables specific to each language. Each program contains one or more tests. Although allowing only one test per program would simplify the evaluation of implementations with a high degree of nonconformity, it would impose additional overhead on implementations with a high degree of conformity. The tests within a program are intended to be independent so any one test may be removed without affecting the remaining tests.

Each test is self-evaluating; i.e., each test is written with knowledge of the data in the database and the correct response for a specific SQL statement. Each test checks for correct execution of the SQL statement and then inserts into the reporting table, TESTREPORT, a "pass" or "fail" value for that test. After all the test programs have executed, a summary of test results is produced automatically by another program which reads TESTREPORT. As each test is executed, a description of the test is printed on standard output (the screen) along with appropriate data values and the test result. This output should be considered as a "log" of the test programs. It is intended to assist in debugging and in analyzing nonconformities. This output is not needed to produce the automated conformance analysis of the SQL test suite.

These tests are not designed to debug DBMS software; however, they may help identify problem areas. The use of small tables does not challenge the buffer-management strategy of an implementation. In addition, the frequent use of ROLLBACK (after tests which modify tables), to restore the base data to its original state (and thus simplify testing), limits testing of the COMMIT path. Since the SQL standard does not address physical database design, it is likely that schema definition and DML tests will be run in the simplest manner possible, without optimization.

The test suite includes a few tests for the "SQL Flagger" option specified in Section 10.d of FIPS PUB 127-2. These tests contain extensions to the SQL standard. In general, if an SQL implementation supports these extensions, it must be able to flag the extensions with warning messages. These tests are to be run with the flagging turned off and then, if successful, rerun with the flagging turned on. Test evaluation for the SQL Flagger is subjective, based upon examination of any warnings which are printed (or displayed on the screen) when extensions to SQL are used. The "SQL Flagger" tests are very limited. They are intended to demonstrate the existence and style of monitoring provided by a vendor. They do not systematically attempt to detect SQL extensions which are not flagged. For Entry SQL, standard features which are required only by higher levels (beyond Entry) should all be flagged along with nonstandard features. It is desirable, but not required, that the flagging message indicate the exact status (Transitional SQL, Intermediate SQL, Full SQL, nonstandard extension) of the flagged feature.

The test suite has a set of programs to test the specifications in FIPS PUB 127-2, Section 16.6, "Sizing for database constructs." These minimum specifications for the precision, size, or number of occurrences of database constructs are contained (by default) in procurements which do not provide alternate specifications. Reporting of the FIPS sizing tests is separate from reporting on other tests. FIPS sizing tests are not technically considered conformance tests, and passing these tests is not required for a Certificate of Validation for FIPS 127-2. Utility programs are included to make global and program-specific changes in a controlled and systematic manner and to document those changes in the automated report.

The test suite contains additional tests to help evaluate conformance to: (1) the minimum sizing parameters for database constructs specified in FIPS PUB 127-2, Section 16.6, (2) the flagging of extensions, specified in FIPS PUB 127-2, Section 10.d, SQL Flagger, and (3) Interactive Direct SQL, as specified in FIPS PUB 127-2 Section 16.5.

The test suite contains ten different programming language test suite types. An SQL implementation claiming conformance to FIPS PUB 127-2 for a particular SQL interface; for example, Embedded SQL COBOL, should be tested with the appropriate test suite type. The programming language compiler used for testing should conform to the FIPS standard for that language and should be listed in the Validated Products List, which is published quarterly by NIST.

The intention of NIST is that this test suite should be used to help evaluate compliance of implementations of SQL to FIPS PUB 127- 2. A correct implementation of FIPS 127-2 requires the incorporation of the SQL standard document, ANSI X3.135-1992 (or ISO/IEC 9075:1992), into the design specifications for the SQL implementation. The SQL test suite then confirms that the standard has been interpreted and implemented correctly by the SQL supplier. The test suite is intended to be used, in conjunction with the SQL supplier's own independently-developed regression tests, to ensure a robust and internally consistent product. A quality SQL implementation is not achievable by simply "fixing the product" until it passes the tests.

It is important to recognize the limitations of this test suite and of any test suite. In particular, it would be incorrect for implementations to claim conformance to FIPS PUB 127-2 simply by virtue of correct performance of these tests. It is reasonable, however, for purposes of procurement, to substantiate claims of conformance to FIPS PUB 127-2 by demonstrating correct execution of these tests. Performance is recognized as a critical selection factor in many DBMS procurements. However, performance is not an issue for standards validation testing and is not measured by this test suite.

9.2 Testing SQL/ERI implementations

The NIST SQL Test Suite tests an entire level of the FIPS SQL standard and includes tests for schema definition, data manipulation, transaction management, and programming language interface at each level. Each of these areas are tested separately, so with a moderate effort it will be possible to modify the test programs to test the various levels of SDL and DML, the transaction mangement alternatives, and the different binding styles of the SQL/ERI Read-Write profiles. At the present time, the test suite does not distinguish between the Minimal SQL and Entry SQL levels, so more extensive modification will be needed to accommodate Minimal DML testing without losing completeness of coverage. The timescale and scope of this work is dependent upon available funding for NIST SQL Test Suite development.

In many cases, SQL/ERI profile testing will be accomplished by giving an implementation an SQL script of table definitions that will be used for DML testing. Any implementation claiming to support Minimal SDL or above will be able to read and implement the table definitions. If an implementation does not claim to support any schema definition, then they will be free to implement the schema definition in an implementation-dependent manner, so long as the result is "as if" the SQL script had been properly executed.

Most of the DML testing is accomplished over just three table definitions with a handful of rows in each table. If an SQL/ERI Server implementation supports Read-Write at the Minimal DML level or above, it should be able to load data into the three tables to satisfy the testing requirements of a large majority of the tests. If the SQL/ERI Server implementation is a Read-Only implementation, then it would be free to load the data in an implementation-dependent manner.

Since the test suite in the past did not make a distinction between Read-Write servers and Read-Only servers, a number of the tests have Update and Select statements mixed together in the same test. Often a test will update an existing column, or add a new row to a table, and then test to determine if a certain query condition is satisfied. For a Read-Only implementation, these tests will have to be substantially modified in order not to lose completeness of coverage. It is the intent of NIST to do this modification on a schedule consistent with the availability of funding and other resource requirements.

10. Procurement Considerations

This FIPS for SQL Environments may be used to assist in the procurement of any of the following types of products:

- -- Client-side user tools
- -- SQL/ERI Clients
- -- SQL/ERI Servers

This first specification is particularly oriented toward implementation profiles for SQL/ERI Servers, but may be of limited assistance in procurements for each of the other two types of products. The assumptions on each of these product types in an SQL Environment are given in Section 1.2, "SQL environment", Section 2, "Data Integration Architecture", and Section 3, "SQL External Repository Interface (SQL/ERI)". This specification cannot be used as the sole procurement instrument for any one of these product types. Instead, to be effective, it must be supplemented by other requirements and/or complementary schema information as indicated in other parts of this specification. The following subsections offer advice on how best to use this specification when procuring a product to be integrated into an SQL environment from any one of these product types.

10.1 Client-side products

A client-side product in an SQL environment is a product that uses the SQL language to access persistent data on behalf of some end user. The product could be a graphical user interface (GUI) or some other presentation tool interfacing with a human end user, or it could be a value-added, computer-aided software engineering (CASE) productivity tool that is accessed from some other end user tool. The following steps may be helpful:

1. State the functional requirements of the tool itself. This could vary considerably and is beyond the scope of the FIPS PUB for SQL Environments.

2. Case:

   a. If the tool interacts directly with a human end user, then state the requirements of the human to computer interface. This interface may depend upon "Human/Computer Interface Services" as discussed in Sections 3.3.2 and 4.8 of the NIST Application Portability Profile (APP) for Open Systems Environments (see [22]).

   b. If the tool provides a services interface to other software tools, then state the calling requirements and data types that must pass across this interface. This interface may depend upon the Common Object Request Broker Architecture (CORBA) published by OMG and X/Open, or upon emerging international standards for language independent procedure calling mechanisms.

3. Choose an SQL binding style to be used between the client-side tool and the SQL data repository. See Section 6, "SQL Binding Alternatives", for discussion of the available binding styles. It is expected that the SQL/CLI binding style will be the most popular choice for client-side products within a single local client/server environment and that the Direct Invocation or RDA/SQL binding styles will be the most popular choices when the client-side tool is accessing server data in a remote data repository.

4. Identify all of the SQL data types, and all of the SQL Abstract Data Type (ADT) instances, that may need to be imported into the application tool. Make sure that the functional requirements of the tool include the manipulation and presentation of these application-specific objects.

5. If the client-side tool is going to create and manage its own public persistent data, and thereby be an SQL/ERI Server for other products in the SQL Environment, then follow the steps in Section 10.3 below for procurement of an SQL/ERI Server.

10.2 SQL/ERI Clients

An SQL/ERI Client is a full-function, conforming FIPS SQL data manager that, in addition, supports the SQL/ERI interface described in Section 3. The functional requirements of the client side of this interface must be supplied by an individual procurement since they are beyond the scope of this first specification for SQL/ERI profiles. The following steps may be helpful:

1. If a full-function, conforming FIPS SQL data manager is not already available, then follow the Special Procurement Considerations given in Section 16 of FIPS PUB 127-2 [3] for procurement of an SQL Processor.

2. State the minimum profile of the SQL/ERI Server products that are to be integrated into the SQL Environment by this SQL/ERI Client. Use the profile items identified in Section 7 to determine this minimum profile.

3. Make sure that the SQL Processor supports all of the binding styles identified in the minimum profile for SQL/ERI Servers to be accessed, since a conforming FIPS SQL Processor need only support one such binding style. In particular, if the SQL/CLI or RDA binding styles are specified, make sure that the SQL Processor supports Connection Management statements (a Full SQL feature) and can map those statements to appropriate SQL/CLI service calls or appropriate RDA/SQL-Client protocols.

4. State the functional requirements for "schema federation", that is, the requirements for how the SQL processor is to make the SQL Schemas from an SQL/ERI Server visible as schema elements in some Catalog of the SQL Processor. The end user, be it human or software, should not be expected to have to do its own schema federation; a database administrator should be able to integrate the external schemas from the SQL/ERI Servers and make them appear as if they were part of the local SQL data. Most SQL implementations have the ability to do this, but it is not yet part of the *de jure* SQL standard, so the procuring authority cannot yet point to this capability in a formal standard and require conformance. This topic will be addressed further in profile specifications for SQL/ERI Clients, a follow-on objective of the FIPS for SQL Environments.

5. Make sure that the SQL Processor can translate full-function SQL DML statements into a module of lower level SQL DML statements (e.g. Minimal DML) that have the same effect. In order to do this, it may be necessary for the SQL Processor to build temporary tables, populate the temporary tables from data retrieved from external SQL/ERI Servers, and then further manipulate the data in the temporary tables before returning the correct result to the end user. For a further discussion of this point see Section 3, which discusses the assumed capabilities of an SQL/ERI Client. This topic will be addressed further in profile specifications for SQL/ERI Clients, a follow-on objective of the FIPS for SQL Environments.

6. If the profile for SQL/ERI Server products to be supported requires SQL/PSM, SQL/MM, ADTs and methods, or Object data management, then make sure that the SQL Processor is able to invoke SQL functions and ADT methods that are defined by an external server.

10.3 SQL/ERI Servers

An SQL/ERI Server is a server-side product in an SQL environment that controls the data that is to be made available to client-side tools. An SQL/ERI Server may be a legacy database, a specialized data manager such as a Geographic Information System or a Full-Text document management system, or an object database management system. With even partial support of the SQL language, such products are able to provide a degree of integration between themselves and other products in the SQL environment. The following steps may be helpful in the procurement of an SQL/ERI Server:

1. Determine if the SQL/ERI Server is to be a Read-Only Server or a Read-Write Server. A read-only server will have a much easier time meeting the conformance requirements specified in Section 7. Failure to specify either Read-Only or Read-Write means that, by default, the SQL/ERI Server is to be a Read-Only Server.

2. Identify any SQL schema definitions that the server shall support. See the discussions in Section 8 for examples of the kinds of schema definitions that might be specified. A procurement may require that

a specific schema definition be supported, or alternatively, it may simply require that a proposal in response to a procurement request include an SQL schema for the data that is to be made available. Implicitly, the schema definition determines whether or not a conforming implemenation shall support SQL null values (see item 4 of Section 4.1). Failure to identify any SQL schema definitions to be supported means that, by default, the supported SQL schema definitions are implementation-defined.

3. Specify a base level of SQL data manipulation language (DML) that shall be supported, by choosing exactly one of the following DML alternatives: Minimal DML, Entry DML, Transitional DML, Intermediate DML, or Full DML. See the discussion of each of these alternatives in Section 4 and the "Data Manipulation Rules" in Section 7.1 (Read-Only) or Section 7.2 (Read-Write). Failure to choose a base level of SQL data manipulation language means that, by default, the base level of SQL data manipulation language is Minimal DML.

4. Specify a base level of SQL schema definition language (SDL) that shall be supported, by choosing exactly one of the following SDL alternatives: No SDL, Minimal SDL, Entry SDL, Transitional SDL, Intermediate SDL, or Full SDL. See the discussion of each of these alternatives in Section 4 and the "Schema Definition Rules" in Section 7.1 (Read-Only) or Section 7.2 (Read-Write). Failure to choose a base level of SQL schema definition language means that, by default, the base level of SQL schema definition language is No SDL.

5. Specify a base level of SQL transaction management that shall be supported, by choosing exactly one of the following transaction management alternatives: No Transactions, Commit-Rollback, Transaction Mode, Transaction Isolation, Transaction Diagnostics, or Constraints. See the discussion of each of these alternatives in Section 7 and the "Transaction Management Rules" in Section 7.1 (Read-Only) or Section 7.2 (Read-Write). Failure to choose a base level of SQL transaction management means that, by default, the base level of SQL transaction management is Commit-Rollback.

6. Specify a default isolation level for SQL transaction management that shall be supported, by choosing exactly one of the following default isolation level alternatives: Read Uncommitted, Read Committed, Repeatable Read, or Serializable. See the discussion of each of these alternatives in Section 7 and the "Transaction Management Rules" in Section 7.1 (Read-Only) or Section 7.2 (Read-Write). Failure to choose a default isolation level means that, by default, the default isolation level is Read Committed.

7. Specify the binding styles that shall be supported, by choosing one or more of the following binding style alternatives: Module, Embedded SQL, Direct Invocation, SQL/CLI, or RDA/SQL. See the discussion of each of these alternatives in Section 6 and the effect of each alternative in the rules in Section 7.1 (Read-Only) or Section 7.2 (Read-Write). It is expected that the SQL/CLI binding style will be the most popular choice for SQL/ERI products within a single local client/server environment and that the Direct Invocation or RDA/SQL binding styles will be the most popular when the server data repository is an isolated node in a wide area client/server environment. Failure to choose a binding style means that, by default, an implementation may choose to support either the SQL/CLI binding style (see Section 6.4) or the SQL/RDA binding style (see Section 6.5).

8. For each of the Module, Embedded SQL, or SQL/CLI binding styles chosen above, specify the programming language interface that shall be supported, by choosing one or more of the following programming language alternatives: Ada, C, COBOL, Fortran, MUMPS, Pascal, PL/I, or SAMeDL. Failure to choose a programming language interface means that, by default, the interface for Programming Language C is the only requirement.

9. Specify which SQL Features beyond those required above are to be supported, by choosing "No Extensions" or by identifying features by feature number from FIPS PUB 127-2 (see Section 14 of [3]). See the discussion of SQL Features in Section 5.1. Identify which features are "required" and which are "desirable". Be very careful about requiring individual features (rather than a FIPS specified level of features) as that practice can easily lead to procurement protests. Be sure to declare how desirable features will be scored in the evaluation of responses to the procurement. Note that a Read-Only interface need only support the Read-Only aspects of each feature. Determine if it is a procurement requirement for the implementation to support the SQL_FEATURES table as specified in Section 15 of FIPS PUB 127-2 (required for Intermediate DML and above and also required if the "SQL Features" option is selected in the implementation profile declaration). Failure to specify any additional SQL Features to be supported means that none are required.

10. Specify whether or not the SQL/PSM optional extension shall be supported; if SQL/PSM support is required, then the implementation shall support the Executable SQL/PSM or the Definable SQL/PSM requirements of the SQL/PSM specification. See Section 5.2 and item (8) of Section 7 above for a discussion of SQL/PSM. Note that a Read-Only interface need only support the Read-Only aspects of SQL/PSM. If SQL/PSM is required and if the SQL/ERI Server is a Read-Only server, then specify exactly which functions and abstract data types (ADTs) are to be invokable by supplying the appropriate SQL3 schema definitions for the required functions, modules, and procedures. Failure to specify a requirement for SQL/PSM means that no support for any aspects of SQL/PSM is required.

11. Specify which, if any, conformance alternatives from SQL/MM shall be supported; if SQL/MM support is required, then point to the then current SQL/MM specification (see [13]) and explicitly indicate which Parts, and which conformance alternatives within each Part, are required. See Section 5.3 and item (8) of Section 7 above for a discussion of SQL/MM. Also see the identifiers for various features of SQL/MM in Section 7.3. Note that a Read-Only interface need only support the Read-Only aspects of SQL/MM. To be successful, a Read-Only procurement should include the desired SQL/MM features in the SQL schemas produced in item 2 above. Failure to specify a requirement for any of the SQL/MM Parts, or conformance alternatives within each Part, means that none are required. Keep in mind that at the time of publication of this FIPS for SQL Environments, the SQL/MM specification was very immature; it may change considerably before final adoption of any of its parts as International Standards.

12. Specify which SQL3 features dealing with abstract data types (ADTs), methods, and object data management are to be supported; if any of these SQL3 facilities are to be supported, then point to the appropriate ADT or object management clauses in the then current SQL3 specification (see Sections 5.4 and 5.5 above and [12]) and explicitly indicate which features are required. Note that a Read-Only interface need only support the Read-Only aspects of any indicated SQL3 features. To be successful, a Read-Only procurement should include the desired SQL3 features in the SQL schemas produced in item 2 above. Failure to specify a requirement for any of these SQL3 features means that none are required.

13. Specify whether or not the SQL/ERI Server shall support encompassing transactions or the emerging standard for SQL/XA (see Section 5.6 above and [12]). Note that it may or may not make any sense to require that a Read-Only SQL/ERI Server participate in read-write transactions managed by a global transaction manager. Failure to explicitly specify a requirement for SQL/XA in a procurement means that support for SQL/XA is not required.

14. Specify the minimum requirements for the precision, size, or number of occurrences of any required SQL data types or features. See the discussion of sizing in Section 5.1. Unless otherwise specified in a procurement, the Entry Value sizing limits from Section 16.6 of FIPS PUB 127-2 apply to all Entry SQL or Transitional SQL features and the Intermediate Value sizing limits apply to all Intermediate SQL or

Full SQL features. A procurement is responsible for identifying its own sizing limits on all required features, but in the absence of an explicit declaration, the default minimum limits apply for that procurement. Determine if it is a procurement requirement for the implementation to support the SQL_SIZING table as specified in Section 15 of FIPS PUB 127-2 (required for Intermediate DML and above).

15. Specify the minimum requirements for character set support. See the discussion of character sets in Section 16.7 of FIPS PUB 127-2. Failure to indicate explicit character set requirements for an SQL/ERI Server means that support for the representation of the 95-character graphic subset of ASCII (FIPS PUB 1-2), in an implementation defined collating sequence, is by default the minimum requirement for Minimal, Entry, and Transitional profiles. Profiles that require Intermediate DML or above must support the Intermediate SQL character set requirements.

16. Specify any SQL/ERI Server performance requirements. This standard is silent on the topic of performance. The NIST SQL test suite also makes no attempt to test the performance aspects of a conforming system. Whenever performance requirements are known in advance, they may be included as an integral part of the procurement specification.

17. Specify any SQL/ERI Server security requirements. Some environments require "trusted" database access beyond the GRANT and REVOKE privileges and the view definition capabilities specified by the SQL'92 standard. Procurements for systems that operate in these environments should include explicit additional requirements to be supported. For additional information, refer to *Trusted Database Management System Criteria* (NCSC-TG-021 Version 1), National Computer Security Center, April 1991, and *Security Issues in Database Language SQL*, NIST Special Publication 800-8, NIST, August 1993.

18. Read Sections 16.8, "DBMS procurement", and 16.11, "System integration", in FIPS PUB 127-2 to see if any of the discussions therein apply to this procurement for an SQL/ERI Server. Section 16.8 lists a number of emerging SQL3 features (see [12]) that might be listed as "desirable" features in an SQL/ERI Server procurement. Remember that a Read-Only interface need only support the Read-Only aspects of any specified features.

References

1. Cannan, S.J. and G.A.M. Otten. *SQL - The Standard Handbook*, McGraw-Hill Book Co, Berkshire SL6 2QL England, October 1992.

2. Date, C.J. with Hugh Darwen. *A Guide to the SQL Standard*, Addison-Wesley Publishing, Reading, MA 01867 USA, October 1992.

3. FIPS SQL, *Federal Information Processing Standard for Database Language SQL*, 2nd revision, FIPS PUB 127-2, U.S. Department of Commerce, National Institute of Standards and Technology, June 2, 1993.

4. Gallagher, Leonard. *Object SQL: Language Extensions for Object Data Management*, Proceedings of the First International Conference on Information and Knowledge Management (CIKM), Baltimore, MD, 9-12 November 1992, International Society of Mini and Micrcomputers (ISMM), pages 17-26.

5. Gallagher, Leonard and Joan Sullivan. *Database Language SQL: Integrator of CALS Data Repositories*, NIST technical report, NISTIR 4902, September 1992.

6. IEEE SFQL. IEEE Standards Committee on Optical Disk and Multimedia Platforms (SCODMP), SFQL Working Group, Institute of Electrical and Electronics Engineers, Inc., Washington, DC 20036-1903, USA.

7. IETF Netdata. Internet Engineering Task Force (IETF), Network Database Working Group (netdata), Corporation for National Research Initiatives, Reston, VA 22091, USA.

8. ISO/IEC 9075. *Database Language SQL*, International Standard ISO/IEC 9075:1992, American National Standard X3.135-1992, American National Standards Institute, New York, NY 10036, November 1992.

9. ISO/IEC 9579. *Open Systems Interconnection - Remote Database Access (RDA), Part 1: Generic Model and Part 2: SQL Specialization*, International Standard ISO/IEC 9579:1993, American National Standard ANSI/ISO/IEC 9579:1993, American National Standards Institute, New York, NY 10036, December 1993.

10. ANSI/ISO/IEC CD 9075-3, *(Draft) International Standard for Database Language SQL, Part 3: Call Level Interface (SQL/CLI)*, JTC1 Draft International Standard (DIS), document SC21 N9117, 13 October 1994.

11. ANSI/ISO/IEC CD 9075-4, *(Draft) International Standard for Database Language SQL, Part 4: Persistent Stored Modules (SQL/PSM)*, JTC1 Committee Draft (CD), CD Ballot document SC21 N8897, August 1994.

12. ISO/IEC SQL3. *(ISO-ANSI Working Draft) Database Language SQL (SQL3)*, document ISO/IEC JTC1/SC21 N6931, American National Standards Institute, July 1992. Later versions available from Working Group or Rapporteur Group documents as a six-part document, Part 1: Framework, Part 2: Foundation, Part 3: Call Level Interface, Part 4: Persistent Stored Modules, Part 5: Language Bindings, Part 6: SQL XA Specialization.

13. ISO/IEC SQL/MM. *SQL Multimedia and Application Packages* (SQL/MM), Project description in document ISO/IEC JTC1/SC21 N7179, American National Standards Institute, January 1993. Initial draft available as a four-part document, Part 1: Framework, Part 2: Full-Text, Part 3: Spatial, Part 4: General Purpose Facilities, with additional parts expected for other multimedia areas.

14. ISO/IEC TR 10000-3. *Information Technology - Framework and Taxonomy of International Standardized Profiles - Part 1: General Principles and Framework, Part 2: Principles and Taxonomy for OSI Profiles, Part 3: Principles and Taxonomy for Open System Environment Profiles*, all three Parts have been published by ISO/IEC and are available through the American National Standards Institute. The final text of Part 3 was distributed as document ISO/IEC JTC1/SGFS N1249, dated 3 January 1995.

15. Melton, Jim and Alan Simon. *Understanding the New SQL: A Complete Guide*, Morgan Kauffman Publishers, San Mateo, CA 94403, October 1992.

16. NIST OIW. *Stable Implementation Agreements for OSI Protocols*, Version 6, Edition 1, NIST Open Systems Environment Workshop, document NIST SP 500-206, December 1992.

17. NIST SQL Test Suite. *An automated suite of tests for evaluating conformance to FIPS SQL*, Version 4.0, NIST Computer Systems Laboratory, July 1993.

18. NIST SQL Validation Procedures. *Database Language SQL Validation Procedures*, unpublished NIST technical report, NIST Computer Systems Laboratory, 1993.

19. NIST VPL. *Validated Products List: Programming Languages, Database Language SQL, Graphics, GOSIP, POSIX, Security*; Judy B. Kailey, Editor, NISTIR 5585, issue No. 1, January 1995 (republished quarterly). Available by subscription from the National Technical Information Service (NTIS).

20. SQL Errata. *Database Language SQL - Technical Corrigendum 1*, ISO/IEC 9075:1992 TC 1, document ISO/IEC JTC1/SC21 N8574, dated June 1994, American National Standards Institute, published by ISO/IEC in October 1994.

21. SAMeDL. *SQL/Ada Module Description Language*, ISO/IEC DIS 12227, Draft International Standard, document ISO/IEC JTC1/SC22 N1385, Spring 1994.

22. NIST APP. *Application Portability Profile (APP) - Open System Environment Profile*, OSE/1 Version 2.0, NIST Special Publication 500-210, June 1993.

23. Melton, Jim. *Object Technology and SQL: Adding Objects to a Relational Language*, Data Engineering, December 1994, pages 15-26.

☆U.S. GOVERNMENT PRINTING OFFICE: 1995 - 386-627/37903