Influencing Database Language Standards

Leonard Gallagher
Information Systems Engineering Division
National Institute of Standards and Technology
Gaithersburg, MD 20899 USA

LGallagher@nist.gov

In this first article of the regular column on data base standardization activities, I give an overview of topic areas under active development in the formal national and international standardization bodies. I solicit contributions on these active topics so that standardizers and researchers can cooperate in the near term, before irreversible decisions are made, to produce the most useful and highest quality database standards.

1. Committee structure

First, for those who are interested, I give a short introduction to the committee structure for producing internationally approved standards. This structure is important because many procurement agencies, like governments, require that international standards adopted by relevant standardization bodies be given first priority in the procurement process. Two important organizations in international information technology standardization are the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). These are private organizations that have built up prestige and recognition over many years for developing standards in widely dispersed technical areas. Other organizations, like the International Telecommunication Union (ITU), have a legal mandate, based on international treaties, for developing standards in a specific area. Computer language standardization has been conducted for many years under the ISO umbrella, and database standardization first began as a programming language standard, so there is nearly universal recognition of database standards produced by ISO.

Several years ago, ISO and IEC merged their standardization efforts in the area of Information Technology and formed a new Joint Technical Committee (JTC1) under a common set of procedures. Standards approved by JTC1 are recognized by both ISO and IEC as if they had been formally adopted by each organization separately. Voting within JTC1 is done by member body on a country basis with one vote per country and with recognized domestic standardization bodies acting as the member body for each country. For example, the United States is represented by the American National Standards Institute (ANSI). JTC1 is divided into a number of subcommittees, each administered by a member body representative, with responsibility in a specific topic area. Each subcommittee (SC) develops a Draft International Standard (DIS) that is then approved or disapproved as an International Standard (IS) by JTC1 member body ballot.

The largest subcommittee in JTC1 is SC21, administered by ANSI and operating under the title "Open Systems Interconnection, Data Management, and Open Distributed Processing". SC21 is divided into 5 active working groups; database standardization is done in WG3, titled "Database", administered by the Standards Council of Canada (SCC), with Bruce Catley of the Canadian Government Telecommunications Agency as convenor. WG3 is itself informally divided into several rapporteur groups, including Database Languages (DBL), Information Resource Dictionary System (IRDS), Remote Database Access (RDA), SQL Multimedia (SQL/MM), and Reference Model of Data Management (RMDM). I chair the DBL rapporteur group, the group responsible for Database Language SQL. Kohji Shibano of Japan chairs the SQL/MM rapporteur group, which is defining an SQL class library for multimedia applications; it is based upon the emerging abstract data type (ADT) facility in SQL. Each rapporteur group meets two times per year for approximately 3 to 7 days duration. The three groups DBL, RDA, and SQL/MM often meet together, either in parallel or contiguously, since they are closely related and co-dependent. One of these twice yearly meetings is always scheduled together with all of the other WGs in SC21 in order to accommodate issues that cut across working groups.

According to the procedures, a working group produces a

base document that is then processed by member body ballot in its parent subcommittee. The steps in the process are Base Document, Working Draft (WD), Committee Draft (CD), and Draft International Standard (DIS). The real work on a document is done at the Base Document and Working Draft stages, and this is when contributions have the greatest impact. When a specification reaches Committee Draft stage it is registered by JTC1 and becomes a formal candidate for standardization. Supposedly, a specification is both complete and stable before it is registered as a Committee Draft, so major modifications become more difficult at each subsequent step in the process.

Contributions to the international process come from the member bodies rather than from individuals. Individuals are able to participate in their domestic organizations and contributions approved domestically are then considered internationally. Occasionally an international organization, like X/Open or the Internet Society, is able to achieve Liaison Organization status with JTC1, thereby allowing it to make contributions directly to the Working Groups and Subcommittees, but voting on document progression is always conducted on a member body basis with a 2/3-rds majority required for progression to the next step.

2. Active standardization topics

Assume that an active standardization topic is one introduced in a Base Document that has not yet reached the Draft International Standard (DIS) stage. Usually a base document remains in this state for approximately two to three years before progressing to DIS and IS. At the present time SQL, RDA, and SQL/MM all have Base Documents at or just preceding the Working Draft stage. I report on the status of each and identify some active topics that have technical issues to be resolved before further progression.

2.1 Database language SQL

Since 1986, Database Language SQL has been enjoying success as an effective International Standard for the definition and management of relationally structured data. The most recent SQL adoption in 1992 (see [3]) provides new facilities for schema manipulation and data administration, as well as substantial enhancements for data definition and data manipulation. Recent textbooks describing SQL-92 include [1], [2], and [5].

Early in 1991, technical committees for SQL standardization committed to enhancing SQL into a computationally complete language for the definition and management of persistent, complex objects. This includes the specification of abstract data types (ADT's), object identifiers, methods, inheritance, polymorphism, encapsulation, and all of the other facilities normally associated with object data management. Preliminary specifications for these facilities are contained in the most recent SQL3 Working Draft [8]. In the following paragraphs, I list some of the topics and issues under consideration for adoption.

Object identity

Object identity is that aspect of an object that never changes and that distinguishes the object from all other objects. It is a unique identification of an object that is independent of the state of that object, and which persists over time. An ADT definition allows several alternatives for object identifier (OID) specification: WITH OID VISIBLE, WITH OID NOT VISIBLE, or WITHOUT OID. There is a continuing debate in the SQL standardization committees as to whether SQL should support all three of the above options, or if every new ADT definition should be assumed to carry a unique object identifier. The outcome of this debate will not affect the functionality of the new language, but it may influence its appearance and style. Other issues involved with object identity include how OID's are represented externally or passed to programming languages. Another issue in object identity is how to handle equality. Should equality be based on object ID equality or can equality of attributes imply equality of objects? If equality of attributes is considered, then how are private attributes kept encapsulated?

Encapsulation

Each attribute of an ADT has an encapsulation level specified as either PUBLIC, PRIVATE, or PROTECTED. Public components form the interface of the ADT and are visible to all authorized users of the ADT. Private components are totally encapsulated, and are visible only within the definition of the ADT that contains them. Protected components are partially encapsulated, being visible both within their own ADT and within the definitions of all subtypes of that ADT. Since the SQL query language is value based, a continuing issue is how to support traditional relational query processing without violating encapsulation.

Object management

Since SQL is a "table-based" language, SQL designers have to address issues concerning whether or not SQL object instances may exist outside of table occurrences. If SQL objects are allowed to exist outside of tables, then new syntax to manipulate them and new structures to hold collections of them must become part of the language. Although these issues are still subject to debate and modification, the current status is to require that object manipulation be achieved through table operations and all object instances are visible as column values in a table.

Methods and functions

An abstract data type includes not only a collection of attributes but also the methods that define the behavior of the ADT. A continuing issue is the "impedance mismatch" between SQL data types and programming language data types. How does one pass an ADT instance to a standard programming language, operate on it using methods defined completely in that programming language, and then return a modified instance to the database? The actual mapping from the <formal parameter list> in the external function declaration to the parameter list of the programming language routine can become quite complex, so techniques for better management of cross-language calls are under active consideration. More sophisticated data type correspondence, especially to object programming languages such as C++, will likely be the subject of near-term considerations.

Subtypes

Specification of "UNDER ADT-name" in the subtype clause of an ADT definition permits a new ADT to be defined as a subtype of an existing ADT. A type can have more than one subtype and more than one supertype. A supertype shall not have itself as a proper subtype and a subtype family shall have exactly one maximal supertype. The SQL implementation of a type hierarchy requires that an instance of a subtype is also an instance of all of its supertypes. Every instance is associated with a "most specific type" that corresponds to the lowest subtype assigned to the instance. At any given time, an instance must have exactly one most specific type. A continuing issue is whether or not this restriction can be relaxed to allow an instance to share multiple types simultaneously.

Multiple inheritance

Real world examples require that we have some method for an object instance to maintain multiple types simultaneously. To handle these situations, SQL provides "multiple inheritance", i.e. a subtype can have more than one direct supertype. In this way an instance will satisfy the requirement to always have a "most specific type". Multiple inheritance could lead to ambiguous inheritance of components from its supertypes, so SQL provides some disambiguity rules. The exact detail of these rules is the subject of continuing debate. These rules, and other related issues, are subject to improvement and evolution as the SQL ADT facility stabilizes over the next two or three years.

Polymorphic functions

Polymorphism is the ability to invoke an operation on any of several different objects and have that object determine what to do at execution time. Support for polymorphism involves technical decisions concerning early or late binding among objects and the procedures that invoke their methods. The algorithms for function resolution are under continuing discussion and modification as we discuss the best way to support the above features while retaining, as much as possible, compile time binding.

Control Structures

SQL computational completeness requires the introduction of various "control" statements into the language. The first of these, to support procedure calls, include ASSIGNMENT, CALL, and RETURN. The obvious next step was to consider if more control statements and other "programming language" facilities should be added to SQL. These include: a CASE statement to allow selection of an execution path based on alternative choices, an IF statement with THEN, ELSE, and ELSEIF alternatives to allow selection of an execution path based on the truth value of one or more conditions, a LOOP statement, with a WHILE clause, to allow repeated execution of a block of SQL statements based on the continued true result of a search condition in the WHILE clause, and a LEAVE statement to provide a graceful exit from a block or loop statement. All of these facilities are currently included in the SQL/PSM part of SQL3, which has already been registered as a Committee Draft (CD).

Compound statement

A compound statement is a statement that allows a collection of SQL statements to be grouped together into a "block". A compound statement may declare its own local variables and specify exception handling for an exception that occurs during execution of any statement in the group. Issues include consideration of restrictions on where such statements may occur in the language.

Exception handling

An exception declaration establishes a one-to-one correspondence between an SQLSTATE error condition and a user-defined exception name. The exception handling mechanism under consideration for SQL3 is based very strongly on the mechanism defined in Ada. Each compound statement is assumed to have an exception handler; if one is not explicitly defined, then a default handler is provided by the system. When the execution of a statement results in an active exception condition, then the containing exception handler is immediately given control. If an exception condition occurs in the exception handler itself, then the compound statement is terminated and that exception condition becomes the "active" exception condition.

Parameterized Types

A parameterized type is really a "type family" with a new data type generated for each value of an input parameter. For example, an ADT definition for VECTOR(TYPE, N) can be thought of as a family of data types, one for each instance of a data type like Real or Complex, and one for each positive integer value of N. The keyword TYPE indicates that a parameter is a data type name rather than a data type value. The rules for matching a parameterized type reference to a parameterized type definition are the same as the rules for matching polymorphic functions.

Constructor types

A constructor type is a special parameterized type supported in the SQL language itself, rather than defined by a user. Examples of constructor types are LIST, SET, and ARRAY. At the present time SQL3 provides limited support for LIST and SET, but no support yet for ARRAY. Issues include the relationship between the existing table construct and list, set, and array instances. Should table operations be used on lists and arrays? Should some constructors be defined as subtypes of others? Should there be a generic "collection" type? Should there be implicit casting functions between and among these data types?

Stored procedures

In the existing SQL-92 standard, a module is a persistent object created by the module language. It is a named package of procedures that can be called from an application program, where each procedure consists of exactly one SQL statement. However, there is no requirement that an implementation be able to execute module language (the alternative is embedded SQL) and the resulting persistent module is not stored as part of the SQL schema, is not reflected in the information schema tables, and cannot be passed across an RDA connection to a remote site. In the emerging SQL3 specification, standardization committees have recognized the requirement for some "standard" capability to define persistent modules that "live" in the SQL schema and whose procedures may be called from any SQL statement in the same processing environment. In SQL3 the CREATE MODULE statement has the same status as any other schema definition statement. Module definitions are reflected in the Information Schema just like any other schema object and they are subject to ownership and access control declarations. A module definition consists of collection of procedures. Each procedure consists of an SQL parameter list and a single SQL statement, which may be a compound statement (see above). An SQL CALL statement can access any of the procedures, either locally or remotely, and pass parameters to it. The primary benefit of persistent, stored modules is that implementations are able to optimize groups of statements rather than just individual statements. Specifications for persistent, stored modules are currently included in the SQL/PSM part of SQL3, which has already been registered as a Committee Draft (CD).

Miscellaneous features

The features discussed above are not the only SQL enhancements specified in the SQL3 Working Draft. Some of the following features offer desirable functional extensions not directly related to object data management. These features have "preliminary" syntax and semantics specified in SQL3; however, all SQL3 specifications are subject to substantial evolution or reconsideration before adoption in any future SQL standard, so user requirements and improved specifications are always welcome.

Dynamic assertions. Support for integrity constraints that are triggered by specific database actions, such as after update or before insertion. Assertions are "dynamic" in that they may reference before and after images of the database.

Dynamic triggers. Support for triggering a sequence of database actions based on a specific database action, such as after delete. Assertions and Triggers make it possible for object self-management to be fully specified in a database schema.

Recursive expressions. Support for SQL expressions of indefinite, recursive depth, such as those arising out of "bill-of-materials" part's hierarchies.

Multiple null states. A facility that allows user definitions for an arbitrary number of application specific Null values, such as "Unknown", "Missing", "Not Applicable", "Pending", etc. Each such Null value would have a different representation in the database so that they can be distinguished by query expressions during retrieval or update.

Roles and data security. An enhanced facility for database security management that builds upon the existing Grant and Revoke definitions. It extends the security model to include named "roles" in addition to schema objects, actions, and users.

Savepoints and subtransactions. A subtransaction is a portion of a transaction that is marked for potential rollback without affecting the other parts of the transaction. By setting and releasing savepoints, an application programmer is able to recover more easily from failed subtransactions, thereby leading to more efficient code.

2.2 Remote database access (RDA)

The SQL standard does not address communication protocols for interoperation between heterogeneous systems in an open systems environment. The just published Remote Database Access (RDA) standard, RDA-93 [4], meets this need and provides the basic services and protocols for SQL interoperability in a client/server architecture.

The RDA-93 standard was developed against the SQL-89 specification and thus only supports the Entry SQL level of the SQL-92 standard. A follow-on RDA standard, with work just getting started [6], will address interoperability issues for full SQL-92 functionality. Some of these issues are identified below.

Client/Server harmonization

The SQL-92 standard introduces the terms client and server and specifies Connection management statements for client connection to an SQL-Server; however, it does not define conformance requirements to guarantee interoperation of clients and servers from different vendors. The RDA-93 standard defines an RDA-Client and an RDA-Server and specifies protocols to form an association between different open systems, open a data resource, begin a transaction, and begin executing SQL data statements; however, it does not link these protocols to SQL Connection management statements. Since there are alternative ways to map SQL Connection management facilities to RDA protocols, further harmonization is needed in this area is needed to ensure that all SC21/WG3 standards will work smoothly together.

Dynamic SQL descriptor areas

The Dynamic SQL facility in SQL-92 allows SQL implementations to process SQL statements that are generated during program execution. The text string of a statement can first be prepared, then a describe statement will return information about the data types of columns or parameters contained in the prepared statement. All such information is maintained by the system in a descriptor area. In a client/server environment the SQL standard does not specify how this descriptor information is returned from the server to the client. There are several alternatives for doing this, e.g. all at once or piecewise, and each has some advantages and disadvantages. This issue will be discussed and resolved during processing of the follow-on RDA specification.

Character set harmonization

The SQL-92 standard provides facilities for defining and naming new character sets and choosing collations on those character sets; however, it does not standardize any specific character sets or collations. If two SQL implementations are interoperating using RDA, then RDA must provide some method for the client and the server to communicate character set and collation information and choose a common basis for subsequent processing.

2.3 SQL multimedia (SQL/MM)

A new ISO/IEC project for development of an SQL class library for multimedia applications was approved in early 1993. This new standardization activity, named SQL/MM, will specify packages of SQL abstract data type (ADT) definitions using the facilities for ADT definition provided in the emerging SQL3 specification.

It makes sense to standardize packages for science and engineering, full-text and document processing, or methods for the management of multimedia objects such as image, sound, animation, music, and video. This SQL/MM standard could 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 should be constructed. The Framework will require that all packages be specified using the ADT definitional mechanisms of the emerging SQL3 standard. The Framework may also specify General Purpose Facilities such as numeric functions, complex numbers, or data structures that are common to multiple other parts of the SQL/MM standard.

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 2: Full Text and Part 3: Spatial (see [7]). Users and researchers in these areas are encouraged to review the base documents and submit requirements or proposals for further development.

3. Availability of documents

Published international standards, as well as American National Standards, are available from the American National Standards Institute, 11 West 42nd Street, New York, NY 10036, telephone 212-642-4900. Base documents that reach the DIS stage are subject to copyright and must also be obtained from ANSI.

The rapporteur groups within SC21/WG3 process hundreds of proposals at each meeting. All proposals are subject to a 6-week rule and many are available electronically at least 6 weeks prior to the next meeting. To assist in the process of maintaining access to the justification and detailed explanations contained in

previous proposals, and to give rapid and wide access to current proposals, we maintain a database standardization archive of documents at NIST. The archive is implemented via an FTP Server on the Internet node "speckle.ncsl.nist.gov" in directory "isowg3".

If you are a researcher or a user willing to help in the standardization effort, then you are welcome to access the archive. Please sign on as user FTP and give your return e-mail address as the password. The "isowg3" directory has a readme.txt file that explains how to access documents in the archive and describes our conventions for determining file format. We maintain separate subdirectories for several X3 technical committees and for each of the WG3 rapporteur groups with documents grouped by meeting location. The minutes, agenda, and document register from each meeting are clearly identified to help locate desired papers.

4. Influencing the process

In the United States, two X3 technical committees are responsible for contributions and recommendations on JTC1 SC21/WG3 standardization projects. They are X3H2 for SQL, RDA, SQL/MM, and RMDM, and X3H4 for IRDS. Both of these committees meet between 4 and 6 times per year for approximately 3-4 days. Membership is individual, but only one individual from any single company or organization may vote. Membership fees range from $300 to $600 per year. Further information on membership can be obtained from CBEMA, X3 Secretariat, 1250 Eye Street NW, Suite 200, Washington, DC 20005-3922, telephone 202-737-8888.

Other countries have database committees similar to the above. Each domestic group submits approved proposals to JTC1/SC21/WG3 groups for further processing. Enforcement of the 6-week rule means that each member body can instruct its delegates to international meetings on each paper that will be considered.

The purpose of this column is to provide an additional forum, separate from the formal standardization committees, for researchers and users to discuss active database standardization issues and make contributions that might influence the process.

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] 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.

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

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

[6] RDA Revision. Proposed Draft Amendment #1 for Remote Database Access - Part 2: SQL Specialization, enhancements to support SQL-92 features, document SC21/WG3 N1633, September 1993.

[7] SQL/MM Base Documents. Part1: Framework, Part2: Full Text, Part3: Spatial, documents SC21/WG3 N1647, N1613, and N1614, September 1993.

[8] SQL Revision. ISO-ANSI Working Draft Database Language SQL (SQL3), Jim Melton - Editor, document ISO/IEC JTC1/SC21 N6931, American National Standards Institute, New York, NY 10036, July 1992. Later versions available electronically from the WG3 DBL document archive.

Leonard Gallagher is a computer scientist in the Information Systems Engineering Division at NIST. He is responsible for data models, database standardization, and integration of database technology with new approaches to information management such as knowledge and object-oriented systems, hypertext, and multimedia information systems. He has been a member of the ANSI/X3 technical committee on Database, X3H2, since 1979 and chairs the ISO/IEC JTC1/SC21/WG3 rapporteur group for follow-on enhancements to the ISO SQL standard. Dr. Gallagher received the B.A. degree in mathematics from St. John's University of Minnesota in 1965 and the Ph.D. in mathematics from the University of Colorado in 1972. He taught mathematics at the Catholic University of America for 6 years and has been involved with database research at NIST for the past 16 years.

Dr. Gallagher can be reached by telephone at +1-301-975-3251 or by electronic mail using the Internet address LGallagher@nist.gov.