Standards In Practice

Andrew Eisenberg
Sybase, Burlington, MA 01803
andrew.eisenberg@sybase.com

Jim Melton
Sybase, Sandy, UT 84093
jim.melton@sybase.com

Introductions

Longtime readers of the SIGMOD record will know that this column was most recently written by Len Gallagher of the USA's National Institute of Standards and Technology (NIST). The nature of Len's job at NIST changed some months ago and his considerable talents are being applied in other areas than *de jure* standardization activities. His moving on has left rather large shoes to fill — which is part of the reason that you see two of us on the masthead.

Jim Melton has participated in database standardization activities as his primary job for rather more than a decade. In that time, he has represented both Digital Equipment Corporation and Sybase, Inc. to ANSI Technical Committee X3H2 (now known as NCITS H2), represented the USA to the corresponding ISO committee, and has been the Editor for SQL-92, CLI-95, PSM-96, and all parts of the emerging next generation of the SQL standard, SQL3. You may have seen Jim's regular column, "SQL Update" in *Database Programming & Design* or one of his books: *Understanding the New SQL: A Complete Guide* and *Understanding SQL's Stored Procedures: A Complete Guide to SQL/PSM*.

Andrew Eisenberg has been active in database standardization for even longer. In addition to representing Digital and Sybase on ANSI X3H2, Andrew has also attended on behalf of CCA and Oracle. Andrew participates in several other standards organizations, including the Object Management Group and the Transaction Processing Performance Council. Andrew's writings have appeared in these pages before when, in December, 1996, he wrote "New Standard for Stored Procedures in SQL" [1] while Len Gallagher was the owner of this column.

We are shocked to realize that, together, we represent well over a quarter-century of standardization representation and activity. While this realization startles us, it allows us to offer you a long-term perspective about standardization and how it relates to the database industry.

Our goal is to use this space to keep you informed about the status of standardization for various database–related or –influenced standards — *de facto* as well as *de jure* — while adding value through analysis of the processes used to develop various standards in and surrounding our industry. We'll undoubtedly write many of the columns ourselves, but we will also call upon the talents of others in the database standards arena to share their talents and views with you.

In this column, we're going to examine the subject of information technology standards in a broader sense, look at some recent notable failures and successes, and attempt to predict how this is likely to shake out in the next few years. We'll also survey a number of relevant standards bodies and the standards they're building to see not only their status but how they're changing their approach to standardization.

What Is A Standard?

Most of us use the word "standard" fairly often and in varying ways. So that you'll have a better idea of what our column is going to discuss over the months ahead, we're going to give you our take on what the word means in today's information technology (IT) environment. In other words: what is a standard, anyway?

First of all, there are "*de jure* standards". These are standards that are published by recognized formal standards organizations. The most prominent such organization in the United States is the American National Standards Institute, better known as ANSI. While ANSI cooperates with other organizations that develop standards in the USA, it has the official stamp as the country's principle developer of standards, including information technology standards. In fact, ANSI does not actually develop standards itself; instead it accredits other bodies to develop standards while operating under ANSI's auspices and rules.

ANSI has published information technology standards published by several different accredited SDOs (Standards Developing Organizations), including the IEEE and a group called NISO (National Information Standards Organization). However, the majority of IT-related standards published by ANSI are developed by a group formerly known as X3 (which doesn't stand for anything). Now known as NCITS — National Committee for Information Technology

Standardization — this group charters Technical Committees to develop specific standards arising from projects assigned to them. For example, Technical Committee (TC) H2 — formerly known as X3H2 — has a series of projects in the area of database technology, including the SQL standard and related standards like RDA (Remote Database Access) and SQL/MM (SQL Multimedia and Application Packages).

Internationally, the principle standards-setting body is the International Organization for Standardization, or ISO. Like ANSI, ISO cooperates with other bodies as well. One primary such body is IEC, the International Electrotechnical Commission. A few years ago, ISO recognized that it and IEC were each developing standards in the IT area, so a combined effort was established in the form of a Joint Technical Committee, JTC1.

JTC1 manages the development of IT standards under ISO's rules. It creates subcommittees (SCs) for specific areas of IT standardization; until about a year ago, SC21 was responsible for standards related to the upper layers (including the applications layer) of Open Systems Interconnection, or OSI. However, OSI was widely viewed as a market failure, so SC21 was dissolved by JTC1 and its more mainstream projects, including SQL, were assigned to other SCs, some new and some already in existence. SQL was assigned to a new SC32, named Data Management and Interchange, along with RDA, SQL/MM, and several other projects.

Not the Only Game In Town

However, OSI's marketplace failure provides clear demonstration that standards developed by a *de jure* standards organization — even technically superior standards — are not necessarily meaningful to the IT industry.

Instead, industry tends to select standards that are genuinely useful to business. "Useful" has many components, but we believe the principle ones (roughly in order of importance) to be:

- Relevant — only those specifications solving problems that need to be solved by some market segment are likely to be adopted.
- Timely — meaning that the specification is available when business needs it and neither significantly earlier nor too late.
- Good enough — a specification doesn't have to be perfect or even necessarily complete in order to be used by industry.

The standards landscape is littered with standards that solve problems that nobody realized were problems, that arrived before the problem was recognized, or that were completed long after industry had selected another solution (often in pursuit of perfection).

Happily, neither ISO nor ANSI (and its analogs in other countries, like BSI in Great Britain, AFNOR in France, and JSA in Japan) have a monopoly on creating standards. Real standards come from a variety of sources — *de facto* standards are those actually in use by the marketplace, whether or not they are sanctioned by some formal body.

The term "*de facto* standard" is often reserved for specifications published by some consortium (a group of organizations formally working together towards a common goal). However, the phrase more accurately means "in reality", so we prefer to use "consortium standard" for those *de facto* standard published by a consortium.

Examples of consortium standards are very easy to find today: the Object Management Group (OMG) has published a number of specifications that have been adopted by large segments of the IT business; a collection of specifications for distributed transaction management published by X/Open (now part of The Open Group) are widely implemented; and the Unicode Consortium's Unicode character set has been generally accepted as the best character set for internationalization purposes.

But what about other specifications that are widely used or implemented, even though they may come from a single corporation? Microsoft's Windows 95 product is often said to be a "standard operating system", for example. We mildly disagree with calling Windows 95 a "standard", since we reserve that word for specifications instead of products, but it's not unreasonable for the term to be applied in spite of our reservations. After all, any product that's used so widely suggests that a wide variety of users have standardized on its use.

The point is this: a real standard is a specification (or perhaps a product) that is widely accepted at least by some identifiable market segment. *De jure* standards aren't necessarily *de facto* standards, and the converse is definitely true as well.

Standardization's Changing Landscape

Formal standards organizations like ISO and ANSI have long been criticized for operating at what seems like a glacial pace. The very nature of their processes appears to inhibit rapid adoption of anything.

In fact, these organizations have adopted rules that endeavor to guarantee "due process" — to ensure that every interested party has the opportunity to be heard and even to participate if they wish. The theory is that providing this level of access results,

not only in fewer disgruntled losers and fewer lawsuits, but also in standards of higher quality — more complete, more accurate, and more reflective of broad industry requirements instead of special interests only.

In other words, the processes are designed to be slow and tedious, but thorough.

Unfortunately, the entire IT industry — not to mention its clients' businesses — seems to run on "web time" these days. Slow-moving processes are often unable to provide the specifications needed by businesses in the timeframes they need them. The market needs specifications that are "good enough" much more quickly...more than it needs excellent standards "eventually". This means that ANSI and ISO cannot be depended on to deliver standards that take years to develop, cautiously resolving every objection from every dissentor, carefully dotting every "i" and crossing every "t".

But what are the alternatives? Common wisdom says that consortia can move much faster and deliver specifications that are good enough and do so much faster. However, our experience is that consortia only *appear* to move faster, because they often start their work with a nearly-complete specification provided by one or more members — and the original developers of the specification may have spent several years working on it privately. In other words, the result often takes about as long as ANSI or ISO would have taken, but the work is done in private by special interests without the participation of others (which would result in a need for unpleasant consensus-building).

Nonetheless, this process is often very useful for a reason that may be surprising. While individual companies (or, indeed, individuals) develop specifications in private, sometimes for years, the consortium process tends to progress only those ideas that have the backing of some market segment, frequently vendors of related products, but often part of the user community. In other words, the consortia are formed to progress useful ideas, not those without market support.

While the *de jure* groups are supposed to have process steps to rule out those proposals without adequate industry support, the need to be inclusive frequently results in approval of projects with support only from an isolated group having the political know-how to manipulate the rules. Of course, the same can happen with consortia, but the costs of forming, joining, and running a consortium are significantly higher than joining a standards committee. so companies are likely to be more selective in what they support.

The same can be said of single-company "standards", like Java (or Windows). Such specifications are not born full-grown, but are developed over sometimes lengthy periods. The public doesn't see their development period, but only becomes aware of the existence of the "standard" when it has been sufficiently completed and publicized. Not that this is bad: again, the good ideas are usually the ones that see the light of day and the bad ones wither away — although we're all aware of good ideas that withered because of a lack of marketing prowess...still, those aren't standards because they're not widely enough in use.

A word of caution, however: Standards developed by special interest groups — whether consortia or individual organizations — are often not very "open", meaning that they may satisfy primarily the needs of their creator instead of the industry as a whole. You may be aware that ISO approved JavaSoft as a submitter of "publicly-available specifications" (PAS) related to Java — more on this below. This was a major coup for Sun and JavaSoft because it was the first time that an individual company was so approved (previously, only consortia had gotten that privilege). In return, ISO member countries wanted an agreement that JavaSoft would agree to allow ISO to take responsibility for maintenance (read "enhancement") of any standard resulting from one of their submissions. This was supposed to ensure that future changes to Java would be determined in an open process rather than behind closed doors. Unfortunately, we have recently learned that Sun made some significant enhancements — related to two-dimensional class libraries — in the Java Development Kit at the request of a major customer, but without any public commentary process...much less an ISO process. While this might seem like a minor issue, it certainly reflects the proprietary nature of such privately-developed "standards" and the dangers of thinking of them as "open standards". This might not adversely affect you at all — you might be quite happy with the results. Still, what about the next set of changes...will they help or hurt your application development efforts?

Bottom line: *caveat emptor*. You gets what you pays for. (Well, you certainly don't get what you don't pay for!) Always examine the process by which a standard comes into existence and is maintained, not just the technical content.

Now, a quick word on publicly-available specifications and fast-track processing: Standards groups like ISO — *de jure* organizations — have recognized that they aren't the only viable source of standards. As a result, they have instituted processes that allow a published specification that is widely available and accepted (which they call a PAS, or publicly available specification) to be turned into a *de*

jure standard using a process that involves a single review-and-vote cycle. This process is being used to turn ANSI and other national standards into ISO standards, X/Open specifications into *de jure* standards, and now the Java specification into an ISO standard.

Status of Relevant Standards and Standards Groups

Now we'll segue into something of more immediate interest: a discussion of several standards organizations (*de jure* and consortia) whose work is relevant to the IT industry today. Perhaps we will devote an entire column to some of them in the future.

> *"The nice thing about standards is that there are so many of them to choose from."*
> -- sometimes attributed to Grace Hopper and sometimes to Andrew S. Tanenbaum

NCITS H2 and ISO/IEC JTC1/SC32/WG3

NCITS H2 is the US Technical Committee on Database. SC32 is the Data Management and Interchange Subcommittee of JTC1. WG3 is the Database Languages Working Group of JTC1/SC32. Together, these committees have produced several SQL Standards over the years. SQL-86 and SQL-89 have been superceded by SQL-92. These standards define the SQL data model, DDL and DML statements, embedding in host programming languages, and dynamic execution of statements.

SQL/92 CLI

In 1995 an addendum for a Call-level Interface, SQL-92/CLI (sometimes called CLI-95), was adopted. CLI-95 is a subset of the popular (*de facto* standard) ODBC interface from Microsoft and others; it functions as a callable interface to an SQL database system, providing a highly dynamic capability by contrast with the relatively static facility provided by embedded SQL. CLI (and ODBC, of course) is primarily used by *ad hoc* applications, like decision support applications, whereas embedded SQL is more likely to be used by "glass house" applications that are considerably less dynamic in their function.

SQL-92/PSM

In 1996 an addendum for Stored Procedures, SQL-92/PSM (Persistent Stored Modules, sometimes called PSM-96), was adopted. SQL-92/PSM added the following types of features:

- Multi-statement Procedures: groups of SQL statements can be executed together; flow-of-control statements, local variables, and condition handlers are provided
- Stored routines and modules: procedures, functions, and modules can be stored in an SQL-Server
- External routines: functions and procedures written in host programming languages can be invoked from SQL statement

SQL-92/PSM was discussed in this column in December 1996 [1].

SQL3

The two committees are currently working on SQL3, which will replace SQL-92 when it is adopted. SQL3 began its final CD ballot in October 1997. An editing meeting took place in March 1998. Additional editing meetings are scheduled for June 1998 and November 1998. If these meetings are successful, then SQL3 could be adopted in early 1999.

SQL3 extends the data types of SQL-92 significantly. It adds some predefined data types, like BOOLEAN, CHARACTER LARGE OBJECT, and ROW. It adds the collection type of ARRAY.

The single largest addition to SQL3 is user-defined data types (UDT's). Users will be able to define their own data types, each with a concrete representation, methods, and ordering properties. These UDT's can be used anywhere that a predefined data type can be used (as the data type of a column, for example).

A UDT can also be used in a new way. It can be associated with a base table, so that each attribute of the UDT maps to a column of the base table. A new data type, REF, can be used to refer to rows in such a table. Inheritance is supported for both base tables and UDT's. It remains to be seen whether it is single inheritance or multiple inheritance that is finally adopted.

Some of the other features that are provided by SQL3 are:

- Recursive Query – creates a result table from the traversal of rows that form a directed graph
- Similar Predicate – an extension of the LIKE predicate that allows regular expressions
- Roles – authorization may be granted to a role, and users may then take on different roles at different times
- Triggers – statements may be defined to execute each time insert, update, or delete statements are executed on a particular table. The statements may execute once for the statement, or individually for each row that is affected.

- Holdable Cursors – cursors may be defined to stay open after a transaction commit

SQL3 will have a set of features that are required for conformance to the standard; this is being termed Core SQL3. Additional packages of features will also be defined that require features in addition to those in Core SQL3.

It is likely that we will devote a column to SQL3 when it nears the end of its adoption process and has become more stable.

Object Data Management Group (ODMG)

ODMG is a non-profit consortium that was formed in 1991 to "develop and promote standards for object storage".

ODMG's latest publication is "Object Database Standard: ODMG 2.0" [2]. This specification contains an Object Model, an Object Definition Language (ODL), an Object Query Language (OQL), and language bindings to Java, C++, and Smalltalk. The Object Model is a superset of the OMG object model, adding relationships, extents, collection classes and concurrency control. The language bindings allow a programmer to do both application and database programming in the same environment.

ODMG 2.0 has extended previous versions of the standard with:

- a Java language binding
- a standard external form for both metadata and data

Transaction Performance Processing Council (TPC)

TPC is a non-profit corporation formed to "define transaction processing and database benchmarks and to disseminate objective, verifiable TPC performance data to the industry". TPC has published a number of benchmarks over time. Vendors run the benchmarks, certified auditors review the tests and results, and the results are submitted to TPC, after which they can then be published. There is a Technical Advisory Board (TAB) that reviews benchmark compliance challenges.

TPC-C

TPC-C is the current OLTP benchmark. It contains a mixture of read-only and update transactions to simulate a complex OLTP application environment. The metrics for TPC-C are transactions-per-minute-C (tpmC) and price-per-tpm-C ($/tpmC).

The current version of TPC-C is 3.3.2. V4.0, currently under development, might have the following changes:

- Increased cost for some of the transaction types
- Enforcement of referential integrity constraints

TPC-D

TPC-D is a benchmark for a complex decision support environment. The queries in the benchmark involve multi-table joins, sorting, and aggregation. TPC-D benchmarks may be run with one of a specified set of database sizes that range from 1GB to 10,000GB.

The current TPC-D benchmark is V1.3.1. A query stream contains 17 queries. The benchmark contains two metrics. The Power metric (QppD@size) is based on a single-stream Power test. The Throughput metric (QthD@size) may be based on an actual multi-stream run, or it may be calculated from the single-stream results.

V2.0 may be approved in the beginning of 1999. Some of the changes being considered for V2.0 are:

- 6 new queries (including left outer join, use of the SUBSTRING function)
- A required multi-stream throughput test, with a minimum number of streams for each database size

TPC-W

TPC-W is a benchmark for a retail eCommerce environment on the web. It is currently under development, with possible approval in early 1999.

The benchmark models a storefront on the web. The benchmark may be run with one of a set of database sizes that range from 1K items to 1M items. The benchmark measures interactions seen by a browser, allowing for some user-interrupted transfers. The primary metrics will be Web Interactions Per Second (WIPS@size) and price-per-WIPS ($/WIPS@size).

SQLJ

SQLJ is an informal group of companies that has been investigating the ways that SQL and Java can be used together. This effort has spawned three documents that are about to be submitted to formal standards bodies for consideration.

SQLJ Part 0 - SQL Embedded in Java

SQL-92 defines the embedding of SQL statements in programming languages such as C or COBOL. This part of SQLJ defines the embedding of static SQL statements in a Java program. The expression of a

user's queries in SQLJ will usually be more compact and readable than their expression in JDBC.

The SQLJ statements are introduced in Java programs by "#sql", which is not a valid token in the Java language. Variables and expressions from the Java language can be used to provide values to SQL and to retrieve values from SQL. The variables and expressions are prefixed with ":" to distinguish them from SQL identifiers. The JDBC mapping of Java data types to SQL data types has been used by SQLJ.

A SQLJ program may then be passed to a SQLJ translator which can generate a pure Java program that might contain JDBC calls, or it might contain other Java statements that communicate with a SQL database.

The SQLJ translator is able to perform some validation of the SQL statements. The translator may be given the actual schema that will be used at runtime or an "exemplar" schema (one that is the same as the schema that will be used at runtime), in which case additional validation of the SQL statements can be done.

SQLJ Part 0 has just been submitted to NCITS H2 [3] for adoption as a new part of SQL, SQL/OLB (Object Language Bindings).

SQLJ Part 1 – Java Stored Procedures

SQLJ Part 1 [4] allows SQL users to invoke Java static methods. It defines the sqlj.install_jar procedure that allows a Jar file containing Java programs to be made known to SQL. SQL procedures and functions can then be defined that use the Java static methods contained in these programs as their bodies (this process may be automated by deployment descriptors in the Jar file). As in part 0, the JDBC mapping of Java data types and SQL data types has been used.

An SQL user can invoke these procedures and functions without the knowledge of whether the body contains SQL statements or a Java method.

SQLJ Part 2 – Java Data Types

SQLJ Part 2 [5] allows SQL3 User-defined Data Types (UDT's) to be defined that use Java classes for their definition. As we stated earlier, SQL3 UTD's may be used in variable, parameter, or column definitions.

As with Part 1, the sqlj.install_jar procedure makes the contents of a Jar file known to SQL. An SQL3 UDT may then be defined, where the UDT is mapped to a class, and the UDT attributes and methods are mapped to the class attributes and methods. The Java class must implement java.io.Serializable.

When a value is stored in such a UDT column the serialized form of the Java instance is physically stored. When the value is later used in a method invocation the stored value is deserialized and the method is then invoked.

Parts 1 and 2 of SQLJ will shortly be considered by NCITS, perhaps processed according to their Fast-Track rules.

References

[1] *New Standard for Stored Procedures in SQL,* Andrew Eisenberg, SIGMOD Record, Dec. 1996.

[2] *Object Database Standard: ODMG 2.0,* Rick Cattell and Douglas Barry, Morgan Kaufmann Publishers, Inc. 1997.

[3] *SQLJ: Embedded SQL for JAVA Part 0: Translator Specification,* Dan Coyle, H2-98-227, May 15, 1998.

[4] *SQLJ Part 1: Java Stored Procedures, Working Draft,* Phil Shaw, June 10, 1998.

[5] *SQLJ Part 2: Java Data Types, Working Draft,* Phil Shaw, June 10, 1998.

Web References

American National Standards Institute (ANSI)
http://www.ansi.org

National Committee for Information Technology Standards (NCITS)
http://www.ncits.org

JTC 1 Information technology
http://www.iso.ch/meme/JTC1.html

Object Data Management Group
http://www.odmg.org/

Transaction Processing Performance Council
http://www.tpc.org/