SQL Standardization: The Next Steps

Andrew Eisenberg
Progress, Bedford, MA 01730
andrew.eisenberg@progress.com

Jim Melton
Oracle, Sandy, UT 84093
jim.melton@acm.org

SQL Standards To Date

In the last few months, we have spent the majority of this column reviewing the meaning of the word "standard", updating you on the status of the long-awaited third-generation of the SQL standard (formerly known as SQL3 and now known as SQL:1999), and introducing you to two of the three parts of the SQLJ specifications.

This month, we're going to look a bit further into the future by surveying some of the new components of the SQL standard that are currently under development.

Before we do that, however, one last word on SQL:1999's availability: In March, 1999, we told you (see reference [1]) the titles of the parts of the SQL standard that would be published in 1999 and gave you the addresses of organizations from which you could purchase the documents. Sadly, there was a minor, but possibly important, typographical error in each of the titles, so we correct that error in references [2] through [6]. Furthermore, we now have more information about how you can acquire copies of the documents.

Many of you have no doubt discovered in the past that standards were available for purchase only if you were willing to go through the tedium of navigating the bureaucracies owning them and to pay the seemingly exorbitant prices being charged. For example, the SQL-92 standard can still be purchased (in hardcopy form) for a mere US$220! That probably seems like a lot of money, particularly if you're a student or just individually interested in the subject.

Well, you'll be happy to learn that you can now *download* the five volumes of the SQL:1999 standard from ANSI's Electronic Standards Store (Web Reference [1]) for US$20/volume and also from NCITS' Electronic Store (Web Reference [2]) for US$20/volume. Those prices are an order of magnitude better than the cost of hardcopy would have been…but note that only electronic (PDF) format is available. (Hardcopy is available—from NCITS, but not from ANSI—at a "slightly higher price"…about US$540 for the five volumes!)

One more note: ANSI approved adoption of those ISO/IEC documents in late 1999 and they have been given ANSI/ISO/IEC designations, too.

Coming Attractions

SQL standardization has been under way for about 15 or 16 years. In that period, three major editions of the SQL standard (and one minor edition, plus two incremental parts) were published: SQL-86, SQL-89, SQL-92, CLI-95, PSM-96, and now SQL:1999.

The emergence of incremental parts (such as References [3] and [4]) signaled a sea change in the way that further standardization of SQL language would be conducted. Instead of feeling forced to periodically republish the entire (monolithic) SQL standard every few years, we are now able to enhance the language in smaller increments and on more reasonable schedules. Naturally, that doesn't mean that the entire standard won't be revised and republished, but it does mean that there are alternatives to doing so when specific new capabilities can be isolated and standardized.

As we write this column, incremental parts are being developed that cover several widely divergent facilities for the SQL language. The parts that are closest to completion and publication are Part 9, SQL/MED (Management of External Data) and Part 10, SQL/OLB (Object Language Bindings).

In addition to incremental parts, it is possible to *amend* a published standard to add new facilities to it. The text of an amendment reads very much like the text of an incremental part, but there is a presumption that an amendment will be folded into the text of the document (or documents) that it amends when it is (or they are) next published. (By contrast, the expectation for incremental parts is that they will remain separate documents under predicted circumstances.) The first amendment for SQL:1999 is currently in preparation and is called SQL/OLAP (On-Line Analytical Processing).

Let's take a look at each of these three documents in turn.

Object Language Bindings

By the time you read these words, work will be complete on Part 10, SQL/OLB (Object Language

Bindings). SQL/OLB corresponds to SQLJ Part 0, about which we wrote in late 1998 (Reference [7]). A version of this specification aligned with SQL-92 was adopted as a new incremental part of the ANSI SQL standard (Reference [8]) in late 1998. More or less simultaneously, a Final Committee Draft ballot was initiated internationally in pursuit of a new incremental part of the ISO SQL standard. That FCD ballot closed in early 1999 with a substantial number of comments, the great majority of which essentially requested that the document be aligned instead with SQL:1999 as well as with JDBC™ 2.0, whose publication was then imminent.

The committee responsible for SQL standardization (ISO/IEC JTC1/SC32/WG3) has held three Editing Meetings to resolve those comments, the final one in January, 2000 in Santa Fe, NM, USA. At that meeting, the last comments were successfully resolved and the group recommended that the document be progressed to Final Draft International Standard ballot. That FDIS ballot should complete in June or July of this year, with the resulting new Part being published as ISO/IEC 9075-10:2000.

Reference [7] detailed the technical content of the specification that became the ANSI SQL/OLB standard. The principle technical differences between that document and the ISO OLB:2000 standard derive entirely from new SQL:1999 capabilities and JDBC2.0 features. Principle among these are:

- Support for scrollable and holdable cursors
- Support for user-defined types
- Support for updatable result sets and batch updates
- Enhanced connection and transaction facilities

In addition to those technical enhancements, considerable improvements were made in the editorial quality of the document. If you're interested, you can acquire a copy of this (non-copyrighted, not-quite-finished) document at:

ftp://jerry.ece.umassd.edu

in directory:

/SC32/WG3/Progression_Documents/FCD/

under filename:

fcdi2-olb-1999-11.pdf (or .ps or .txt).

Management of External Data

Over the last few years, it has become increasingly apparent that application developers are no longer able to focus all of their efforts on building new applications that use SQL database systems to manage their data. They are increasingly being required to integrate all that SQL data with their legacy data (which some observers suggest may actually contain an order of magnitude more data than SQL databases!).

There is an enormous amount of data that exists in ordinary files, on archive media such as magnetic tapes, in non-relational databases (hierarchical, like IMS, and CODASYL or other "network" models), and even real-time, non-stored data like that returned by sensors. And applications must be able to access all of this data, using it together to make appropriate business decisions.

The cost of accessing all that non-SQL data in conjunction with the SQL data is awfully high, and it is made worse by the requirement for programmers to use different interfaces for different sorts of data. Various SQL vendors have learned that they can help their customers (and, not incidentally, earn additional revenue) by providing solutions that allow applications to use the SQL language to access all that non-SQL data.

Oracle, for example, offers its Open Transparent Gateway and the heterogeneous access services it provides; Sybase has its OmniConnect that supports SQL access to many different data sources; and IBM provides DataJoiner to allow access to traditional and nontraditional data. Other vendors, including both major database vendors and smaller niche-market players, offer analogous products.

In late 1998, a (non-final) Committee Draft ballot was initiated in ISO on a new part of the SQL standard to address this market requirement. This new part 9 is named SQL/MED (Management of External Data) and provides an API between an SQL-server (that is, some "local" SQL database management system) and another entity called a *foreign-data wrapper*. The local SQL-server and the foreign-data wrapper exchange information that allows the local SQL-server to retrieve—and probably insert, update, and delete—data that is actually controlled by a *foreign server*. The foreign-data wrapper's responsibility is to allow the local SQL-server to treat the data managed by the foreign server as though it were tabular data…whether or not it actually is in tabular form. Consequently, the local SQL-server deals with *foreign tables*.

The data managed by a foreign server can be non-SQL data, such as flat files or IMS database, but it can also be SQL data managed by other vendors' products. The local SQL-server might choose to decompose an SQL statement given to it by a client application and cause various aspects of the statement to be executed by one or more foreign-data wrappers, while executing some aspects of the statement locally. By providing such seamless (or as nearly so as possible) access to various data sources, a standard for heterogeneous federated database management can finally be provided!

The specific goal of SQL/MED is to specify an open interface that permits anybody, whether an

existing vendor of SQL systems or some entrepreneur who understands the characteristics of some data source required by applications, to write foreign-data wrappers that can be sold in "shrink-wrapped" form so that they work with any SQL database system (on a given hardware and operating system platform, of course) and provide an SQL interface to another data source. By creating a marketplace for such interfaces, we believe that the ability for applications to access data stored in any format will be vastly increased at a much lower cost than applications builders encounter today.

The development of SQL/MED has profited greatly from a new spirit of cooperation among the SQL vendors, particularly in the United States. While we are all still competing vigorously with one another, we are doing something heretofore unusual in database standardization—working together to develop new SQL language capabilities that satisfy the requirements of all vendors and therefore (we fervently hope) a broader range of the user community. This sort of cooperation appears to resulting in standardized capabilities that will actually be implemented by most vendors instead of the mish-mash of features, many of which are interesting to only a single vendor, that seemed to result in earlier years.

The CD ballot on SQL/MED resulted in a very large number of comments, which were resolved in a series of Editing Meetings held during 1999. The proposals to resolve those comments created a revised SQL/MED document that was submitted for a Final Committee Draft ballot at the beginning of 2000. Significant additional comments are anticipated on this FCD ballot and Editing Meetings will be held later in 2000 to resolve them. The goal is for SQL/MED to be published as an International Standard some time in 2001 as ISO/IEC 9075-9:2001. Interested parties can acquire a copy of this (non-copyrighted and definitely not complete) document at:

ftp://jerry.ece.umassd.edu

in directory:

/SC32/WG3/Progression_Documents/FCD/

under the filename:

fcd1-med-1999-11.pdf (or .ps or .txt).

On-Line Analytical Processing

Standards, and their revisions, often take several years to develop, ballot, complete, and publish. SQL:1999 followed SQL-92 by seven years—which we all recognize as much too long. Sometimes, even a more reasonable cycle of republication, such as three years, is too long for a new facility with urgent market demand.

In 1999, several SQL vendors recognized that there was strong and specific demand for analytical tools in the SQL engines—tools that are currently being supplied only through after-market packages. Conversations with the companies offering those after-market packages revealed that they too would prefer to have basic analytical tools built into the database engines so their packages could focus on real value-added analysis capabilities.

With this realization and agreement, the SQL standards community was offered an opportunity to standardize a selection of features—commonly called "OLAP" features—if they could do so more quickly than by including it in the subsequent generation of the standard or even by creating a new incremental part to the standard.

The solution chosen to address this requirement for rapid progression was to establish a project to develop an *amendment* to SQL:1999. The format of an amendment, as suggested earlier in this column, is quite similar to that of an incremental part, but an amendment will be merged into the next generation of the documents it amends automatically, thus minimizing any future difficulties associated with maintenance or enhancement of the specification.

The closer cooperation between SQL vendors on which we commented in our SQL/MED discussion seems to apply even more to the SQL/OLAP work. IBM and Oracle have been particularly aggressive about developing change proposals that satisfy the broadest range of analytical and statistical tool requirements, and other vendors—such as Informix and Microsoft—have been actively participating in proposal development and review. The result is an elegant specification of new SQL language syntax and semantics that has come together remarkably fast.

In fact, a Final Proposed Draft Amendment (FPDAM) ballot on the SQL/OLAP amendment was initiated at the start of 2000. Under the expectation that the relatively high-quality ballot document will need no more than a single Editing Meeting to resolve all comments, a subsequent Final Draft Amendment (FDAM) ballot should result in publication of SQL/OLAP late in 2000 as ISO/IEC 9075-1/AMD1:2000. The title is not yet final, but should be something along the lines of *Amendment 1 to ISO/IEC 9075-1:1999, On-Line Analytical Processing (SQL/OLAP)*. Incidentally, this document amends several parts of SQL:1999, so it was thought appropriate to characterize it as an amendment to part 1, SQL/Framework.

SQL/OLAP introduces several tools widely used in data analysis. First (but not most importantly), it introduces a number of new numeric

functions, such as: LN (natural logarithm of the argument), EXP (raises *e* to the power of the argument), POWER (raises one argument to the power of the other argument), SQRT (takes the square root of the argument), FLOOR and CEILING (returns the largest integer less than or equal to, or the smallest integer greater than or equal to, the argument), ranking functions (returns the ranking of an argument among a multiset of values), and percentile functions (returns the ranking of an argument as its percentile value among a multiset of values).

More complex functions are also provided, including functions that compute standard deviations, covariances, correlations, slopes and intercepts of trend lines, and even several sorts of averages.

While those functions are necessary, they are made much more powerful by the introduction of the notion of *windows* to SQL. A window is a selection of rows in a (virtual) table determined either by grouping together all rows of that table that share values in a specified column or set of columns or by grouping rows based on their proximity to an identified row (either a specified number of rows preceding and following the identified row, or a specified number of groups of rows related by column values). For analytical purposes, it is often desirable to exclude the identified row, or even to exclude all rows that share with the identified row the values in identified columns, from the window, so SQL/OLAP provides syntax for such exclusions.

An incidental benefit of SQL/OLAP affects ordinary SQL cursor operations. Long-time users of SQL will be aware that the ORDER BY clause of an SQL cursor sorts rows based on the values of columns or expressions provided to that clause, and that rows for which the columns or expressions evaluate to the null value are all sorted either at the beginning of the result or at the end of the result—and that different implementations make different choices between beginning and end.

As a result of a need to control the treatment of null values for OLAP purposes, SQL/OLAP introduced syntax that allows the application to specify where nulls are sorted: NULLS FIRST and NULLS LAST. This new syntax has been added to ordinary cursor specifications as well, so applications can control that aspect of their cursor ordering.

You can get a copy of this (non-copyrighted and not complete) document at:

ftp://jerry.ece.umassd.edu

in directory:

/SC32/WG3/Progression_Documents/FPDAM/

under the filename:

fpdam-olap-1999-11.pdf (or .ps or .txt).

It's About Time

Get used to seeing that pun, because it'll show up in a future column!

Several years ago, work was initiated on another incremental part of SQL, part 7, called SQL/Temporal. For those of you not used to the terminology, temporal data is what allows you to telephone your bank to complain about missing your checking account statement from July, 1998, and actually get your missing statement in the mail a few days later. In other words, it permits you to do "time travel" in your database—unfortunately, you can only travel backwards in time…we don't have the technology to allow you to do database queries to find out what the stock market "did" in 2010!

Temporal data is generally managed in terms of *transaction time*—that is, the time during which the database system believes a particular piece of information to be valid—and *valid time*—the time during which the information actually is valid…at least according to the information we have available to us "today". Not all applications require that both transaction and valid time be captured, but a surprising number of applications benefit greatly from having one or the other (or both) available.

We also want to be sure to distinguish *temporal* data from *time series* data. The latter is more commonly available with today's database products and is used for analyzing trends in data recorded at different points over some period of time—such as stock market data collected daily for several months. By contrast, temporal data management is not yet widely implemented, and we find that most temporal data is managed by code written into applications instead of in database management systems!

Work on SQL/Temporal stalled about three years ago for three reasons. First, there was not a lot of enthusiasm from SQL vendors—or, indeed, from their customer base—for adding temporal support to database engines. Second, the SQL standards folks were forced to concentrate on completing SQL3 (in part because it was taking much longer and was much more difficult than expected) and had to avoid the distraction of working on a project for which there was seemingly little market demand. Finally, there is a fundamental disagreement between two camps about precisely what capabilities SQL/Temporal should provide and how those capabilities should be provided (that is, both syntax and semantics).

As SQL3 development drew to a close, resulting in publication of SQL:1999, participants became aware that the marketplace was showing an increased awareness of the benefits of temporal data support and at least some of the SQL vendors have

become interested in the issue. Therefore, development on SQL/Temporal has recently begun to revive and one may hope to see publication of this incremental part some time in the 2002 or 2003 timeframe.

Paper Shuffling

Finally, just to be thorough, we have to note a change in the collection of parts comprising the SQL standard. References [2] through [6] make up SQL:1999. However, Part 5 (SQL/Bindings) is more closely tied to Part 2 (SQL/Foundation) than we thought it would be; as a result, adding any new features to the language—or, indeed, understanding the language as published—very often requires dealing simultaneously with both documents. In other words, we have realized…rather late…that the material in SQL/Bindings should really have been integrated into SQL/Foundation.

By contrast, the specification of the views and base tables of the Information Schema and Definition Schema, currently part of SQL/Foundation, are readily separated and rarely have to be considered simultaneously with understanding or adding some feature to the SQL language. In other words, that material could have easily been separated into another document.

For the next generation of the SQL standard (which we're calling "SQL:200n" and *not* "SQL4"!), Part 5 has been eliminated by merging its contents into SQL/Foundation, and a new Part 11, SQL/Schemata, has been created to hold the Information and Definition Schema specifications that were removed from SQL/Foundation.

Summary

The following table provides a summary of the anticipated delivery of these "Next Steps":

Part	Delivery Date
Part 10, SQL/OLB (Object Language Bindings)	Mid 2000
Amendment 1, SQL/OLAP (On-line Analytical Processing)	Late 2000
Part 9, SQL/MED (Management of External Data)	2001
Part 7, SQL/Temporal	2003?
SQL:200x	2003?

Very obviously, SQL standardization is not complete—not by a long shot. While SQL:1999 has loads of features that are already widely implemented (and not a few that now seem unlikely ever to be implemented by more than a single vendor, if that), the marketplace continues to put new and challenging demands on the vendors and their ability to deliver products. The vendors also continue to find new and exciting ways to compete with one another by offering useful new features to deliver to their customers. As the features being demanded, considered, and built prove their usefulness to a broad community, they will continue to be fodder for additional standardization work.

Will SQL ever be "finished"? Of course! Every language (even COBOL) eventually comes to an end to its usefulness. SQL will be no different. But there isn't anything on the horizon today—not even XML and its variants (like XML Query)—that solve the same problems that SQL solves and does so as well as SQL.

Until the needs change, or the technology available changes dramatically, we believe that SQL will continue to be (as Mike Stonebreaker called it) "Intergalactic Dataspeak". And…the SQL standard will probably continue to grow and be enhanced.

References

[1] "SQL:1999, formerly known as SQL3", Andrew Eisenberg and Jim Melton, SIGMOD Record, Vol. 28, No. 1, March 1999.

[2] ISO/IEC 9075-1:1999, *Information technology — Database language — SQL — Part 1: Framework (SQL/Framework)*, 1999.

[3] ISO/IEC 9075-2:1999, *Information technology — Database language — SQL — Part 2: Foundation (SQL/Foundation)*, 1999.

[4] ISO/IEC 9075-3:1999, *Information technology — Database language — SQL — Part 3: Call-Level Interface (SQL/CLI)*, 1999.

[5] ISO/IEC 9075-4:1999, *Information technology — Database language — SQL — Part 4: Persistent Stored Modules (SQL/PSM)*, 1999.

[6] ISO/IEC 9075-5:1999, *Information technology — Database language — SQL — Part 5: Host Language Bindings (SQL/Bindings)*, 1999.

[7] "SQL Part 0, No Known as SQL/OLB (Object Language Bindings), Andrew Eisenberg and Jim Melton, SIGMOD Record, Vol. 27, No. 4, December, 1998.

[8] ANSI X3.135.10-1998, *Information Systems — Database Languages — SQL — Part 10: Object Language Bindings (SQL/OLB)*, 1998.

Web References

[1] ANSI's Electronic Standards Store: http://webstore.ansi.org

[2] NCITS' Standards Store: http://www.cssinfo.com/ncits.html