DBL:YOW-031/X3H2-95-161

20 May, 1995

ISO

International Organization for Standardization

Organisation International de Normalisation

ISO/IEC JTC1 SC21 WG3

Database Languages

Title: Accomodating SQL3 and ODMG

Source: U.S.A.

Author: Jim Melton

Project: ISO/IEC JTC1.21.3.3

References:

1) ANSI X3.135-1992, *American National Standard for Information Systems — Database Language — SQL*, American National Standards Institute, Inc., approved 16 October, 1992; and ISO/IEC 9075:1992, *Information technology — Database languages — SQL*, International Organization for Standardization, 1 November, 1992.

2) X3H2-95-110/DBL:YOW-016, *(Draft) Amendment 2 to Database Language SQL, ANSI X3.135-1992*; and *(Draft) SQL Technical Corrigendum 2 (TC #2)*, Stephen Cannan (ed.), March, 1995.

3) X3H2-95-083/DBL:YOW-003, *(ISO Working Draft) Framework for SQL (SQL/Framework)*, Jim Melton (ed.), March, 1995

4) X3H2-95-084/DBL:YOW-004, *(ISO-ANSI Working Draft) Database Language SQL (SQL3)*, Jim Melton (ed.), March, 1995

5) X3H2-95-085/DBL:YOW-005, *(ISO Working Draft) SQL Call-Level Interface (SQL/CLI)*, Jim Melton (ed.), March, 1995

6) X3H2-95-086/DBL:YOW-006, *(ISO Working Draft) SQL Persistent Stored Modules (SQL/PSM)*, Jim Melton (ed.), March, 1995

7) X3H2-95-087/DBL:YOW-007, *(ISO Working Draft) SQL Host Language Bindings (SQL/Bindings)*, Jim Melton (ed.), March, 1995

8) X3H2-95-088/DBL:YOW-008, *(ISO-ANSI Working Draft) SQL Global Transaction Interface (SQL/Transaction)*, Jim Melton (ed.), March, 1995

9) *The Object Database Standard: ODMG-93*, Release 1.1, R. G. G. Cattell (ed.), Morgan Kaufmann Publishers, 1994

Discussion Paper

1 Discussion

It came to pass in the Ides of March, 1995, that I learned that Sybase was to hold a customer meeting in Versailles, France, during the first week of April, 1995. April in Paris is not an uncomely time. Hence I availed myself of the opportunity to attend this meeting. Having prevailed upon my lord and master ("Hi, Don!"), I acquired the necessary travel documents and got me hence to the City of Lights.

Once I secured permission to go to Versailles, it occurred to me that O_2 Technology has its headquarters in Versailles. I contacted François Bancilhon and asked if he would be able to meet with me to discuss SQL3/OQL accomodation (as proposed at the December, 1994, joint meeting between X3H2 members, X3H7 members, and ODMG members).

François was able to meet with me for 3 days; he was also able to arrange for Guy Ferran and Sophie Cluetto join us for portions of those days. I excused myself from the customer meeting (save for one necessary showing of the flag) and spent the rest of the time speaking (and, not at all incidentally, eating) French. Of course, not all was sweetness and light (nor fruit and cheese). Instead, I found myself involved in an intense series of technical discussions from which I emerged not a little dazed.

The present paper's purpose is to update the SQL3 community regarding those three days and to outline our conclusions regarding possible accomodation between SQL3 and OQL (more properly between the SQL3 data and object models and the ODMG data and object models).

1.1 What I did on my Spring vacation

Before launching into the technical issues, I want to reassure you all that April in Paris is, indeed, very lovely. The first couple of days weren't too much fun, as the weather was a bit dreary and I wasn't at the top of my form.

However, the sun soon came out and the flowers really began to bloom. The Chateau at Versailles was truly gorgeous, what with the trees beginning to bud and the flower beds starting to show off. Of course, France being France, the museum staff was mostly on strike, so the tours through the more interesting rooms of the Chateau were not running. Still, a walk through the "public rooms" and through the gardens was quite enjoyable.

On a different day, wandering through Paris was a great way to depressurize from the *trois jours* of discussing object technology. I ended up a little footsore, but it was classical Paris. People were out in multitudes, as though this was the first really beautiful day of the Spring. Many of the monuments and so forth were crowded, as the museums were all still on strike, but there were quite a few areas of town not yet overwhelmed by tourists.

In short, Paris in April is a nice restorative. But...on with the main purpose of this paper:

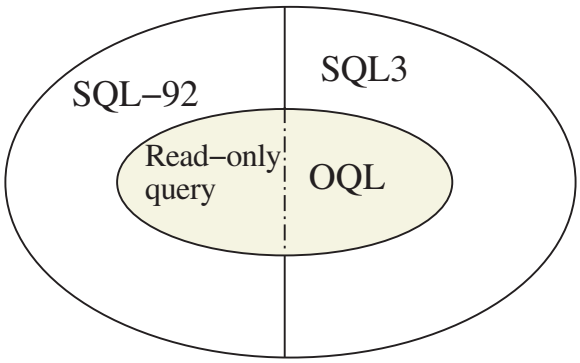
1.2 A quick review

Readers will undoubtedly recall (either through presence there or from reports) the two "joint meetings" between various X3H2 members, a few X3H7 members, and many of the ODMG members. At those meetings, the attendees discussed the desirability of having a

single object database query language to avoid fragmenting the marketplace and overloading vendors wanting to pursue both "extended relational" markets and "pure OODBMS markets".

One of the most important conclusions of those joint meetings is that separation of the data/object models and they query languages would be quite difficult, if not actually impossible. Pursuant to this, Andrew Eisenberg and Doug Barry agreed to produce a comparison of the models of the two languages/groups. Of course, Andrew found himself in the throes of corporate turmoil; as a result, his part of the action item was delayed until fairly recently (roughly early March). Andrew circulated to the members of the "working group" of the second joint meeting his description of the SQL3 data and object models. Unfortunately, I have not seen any evidence that Doug Barry contributed with a description of the ODMG models; nonetheless, this information is available in reference [9]. More unfortunately, no real comparison of the models was ever produced (though Amelia Carlson did produce an informal sketch based on Andrew's paper). (Nonetheless, Andrew's description of the SQL3 models has proved quite valuable in understanding the differences between SQL and OQL vis-à-vis objects.)

Readers may also remember the Venn diagram produced by that second joint meeting, illustrating the generally-agreed goal of the accomodation process:



It is generally agreed that this picture means that the participants intend for OQL to eventually be a subset of SQL3 and that some read-only subset of SQL-92 be a part of OQL.

Of course, life is rarely as simple as simply stating this. There are difficulties with the syntax of OQL as it overlaps with that of SQL-92, and there are subtle problems with the data models of the two languages. One of the most prominent difficulties is illustrated by the following query:

```
SELECT x.name FROM persons x
```

In SQL-92, this query returns a virtual table having a single column whose data type is (probably...we can't be certain without seeing the schema) CHARACTER or CHARACTER VARYING of some (maximum) length.

By contrast, in OQL, this query will return a `bag` (*ugh*, what an ugly word in this context) of `struct` containing a single element whose data type is (again, probably) `string` of some length.

It is this example that begins to make people nervous about the chances of really "merging" the two languages and the two models.

1.3 The negotiations in Versailles

As we began to discuss the direction we had been given at the joint meeting, a couple of other interesting queries were posed:

```
SELECT STRUCT(name:x.name) FROM persons x
```

Of course, this query doesn't actually exist in SQL3, but has a rough analog since the ROW constructor has been added. It wouldn't be terribly difficult, we opined, to add this to SQL3 more fully with the same semantics that it has in OQL. The other query was:

```
SELECT x FROM persons x
```

In OQL, this returns a `set` of objects (assuming that "persons" is a collection of objects!), complete with their OIDs (in fact, probably only the OIDs...*but*—and this is a critical point—it should be completely invisible to the application programmer whether one "has" an OID or something else comprising the actual object...whatever that means). Of course, in SQL3 as it is defined today, this query isn't meaningful at all.

These queries have proved so problematic to resolve without harming SQL-92 compatibility that François proposed that OQL should do the following: replace OQL's existing `SELECT` syntax with a new "`SELECT_OBJECT`" statement. This new statement would essentially use all of the existing OQL `SELECT` syntax, but munge the keywords so that there is no syntactic overlap with SQL-92.

In order to get SQL-92's "read-only" subset into OQL, François proposes that the OQL specification "point to" the SQL-92 standard (reference [1]) and simply identify a BNF "entry point" (such as `<query specification>`). This, we concluded, would give OQL the ability to have retrieval of objects and values with its existing semantics plus the ability to retrieve data with SQL-92 semantics *using SQL-92 syntax*.

Another sticky point in making it possible for OQL programs to access SQL databases is the existence of nulls in SQL. (Chris Date, no doubt, would be amused.) Happily, it didn't take much to persuade the O_2 folks that there was simply no option other than adding to OQL the ability to handle nulls. Guy Ferran was tasked with this job and he accepted it with a certain dread over the amount of work it was going to require of him.

Finally, and possibly most importantly, François and company recognized that OQL must recognize the concept of "table" (even if under a different name) in order that it be able to access SQL's table data. After a bit of discussion, Sophie Cluet posited that OQL should specify the keyword `table` to have the semantics of OQL's existing `bag(struct(...))`. The O_2 folks were delighted with this solution, as it meant very little additional specification or implementation and was thought to provide very nearly identical semantics. (Of course, they recognize that careful examination of the result is required to be sure that the semantics are precisely the same.)

There were several other areas that we discussed:

- Domains: SQL has them and OQL doesn't. We concluded that OQL could probably provide domain support via some sort of `typedef` syntax.

- Constraints: SQL has them and ODMG doesn't. While the O_2 folks recognized the necessity of providing constraint facilities in industrial-strength database systems, they were a bit nervous about how to provide them in an OO system. Quite a lot of discussion left us with the cautious conclusion that these can be supported by providing "C++ wrapper classes" (without any public `structs`) to encapsulate methods so that constraints can be checked automatically. This is promising, but not thought to be

completely satisfactory; more study is going to be needed here. (Assertions fell into the same category.)

- Virtual tables: SQL has views and unnamed transient virtual tables, while OQL has named (transient) queries. There is some commonality in these concepts and it seems likely that OQL can be enhanced to have (persistent) views; however, such queries should not *create* new objects, as that causes additional problems. It may also be desirable to reconsider adding named (sub)queries to SQL at some time in the future.

- Privileges: SQL has them and OQL doesn't. We didn't reach any conclusions on this issue, as we concluded that it's less a part of the data model and more a part of operational semantics. The intent is to visit this subject in the future.

- Data types: SQL has one set and OQL has a different set. Many types in each set have a corresponding type in the other set. In other cases, an easy mapping (using language-binding techniques near and dear to our hearts) can be used. Some care would have to be used to ensure that the collection type semantics match up, but the general feeling was that there isn't a real problem here. One possible exception is internationalization of character strings, which OQL hasn't addressed yet.

- Triggers: SQL has them and OQL does not. We concluded that triggers were highly desirable for OQL. We also agreed that it might be useful for SQL3 to consider adding a trigger mechanism for objects in addition to the existing table trigger mechanism.

- Multiple inheritance: We discovered that SQL3 and OQL use the same "renaming" rule when multiple supertypes have components of the same name that are not inherited from a common supertype. Therefore, there is no problem with attribute renaming and multiple inheritance.

- Function dispatch model: SQL3 uses a fully general dispatch model and OQL uses a classical model. Lengthy discussion uncovered the following facts:

  — C++ supports only the classical model (that is, functions defined on objects are polymorphic only on a single object argument, but other non-object arguments are also used to resolve polymorphism issues).

  — OMG's object model supports only the classical model, primarily for reasons of getting even *reasonable* performance in a distributed environment.

  — Smalltalk doesn't have the issue to resolve, as multiple inheritance isn't supported at all.

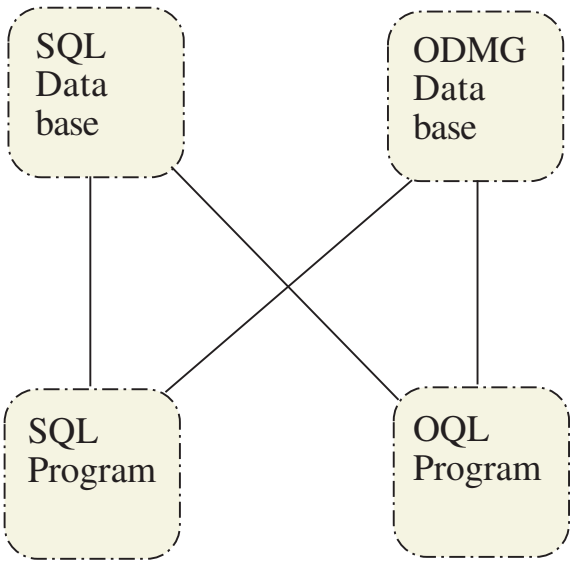
  — OQL currently supports only the classical model.

  However, François acknowledged that real applications often require the capabilities that generalized function dispatch offers. He proposed the following compromise: SQL3 should consider the possibility of offering generalized dispatch on those functions/methods that are defined *outside* of an ADT definition and restricting those functions defined as part of an ADT definition to the classic model. In return, OQL would add generalized dispatch for functions defined "stand-alone" and support a proposal to OMG to add the facility in an analogous form there.

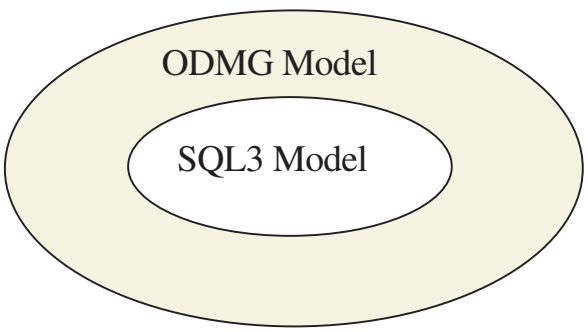
- Creating objects: SQL3's objects get "put" into the database by placing an object into a location specified as *`datatype`* `INSTANCE`. ODMG's objects get created by invocation of the constructor for the object. This is probably the most important problem that they see with SQL's object model. They urged me to try to persuade X3H2 and DBL RG to replace the `INSTANCE` facility with something more object-like, such as actual extents.

The belief is that accomplishment of elimination of these differences (including, obviously, all the niggling little details that will arise in the process) will permit OQL programmers to access SQL databases (at least in a read-only fashion) and will also permit SQL programmers to access ODMG databases! All that remains is a SMOP!

We used this picture to illustrate that ultimate, if elusive, goal:



With this goal in mind, we went a little further and posited a second, perhaps even more elusive goal: that the SQL3 data/object model would be a subset of the OQL data/object model. While this might initially be found controversial, we felt that it reflected the desire of the ODMG community to be fully compatible (in semantics if not in syntax) with SQL, and the SQL communities lack of desire to support every sort of object "location" that OQL supports (such as allowing individual objects to be persistent, named "things" in the schema). This was illustrated as:



Believe it or not, that filled the three days. We concluded our discussions with a table that outlined the different "stages" that could accomplish the goal of unification:

Phase	ODMG Changes	SQL3 Changes
1	Add nulls to the ODMG data model; add tables (table emulators) to the model; and add SQL-92-compliant syntax to OQL (includes changing OQL's existing SELECT to SELECT_OBJECT and pointing to SQL-92 to include SELECT)	Replace INSTANCE with extent tables
2	Add triggers, views, and constraints (possibly assertions) to the ODMG model; add UPDATE/INSERT/DELETE statements to OQL (with SQL-92 semantics); and add generic function dispatch to OQL	Enhance SQL's triggers to include ADT usage; add "object views" to SQL; specialize function dispatch so that the classic model is used for functions defined inside an ADT definition
3	Ensure that OIDs (that is, surrogates for objects) from an SQL3 database can be "stored" (without interpretation) in an OQL database	If and when interest justifies it, add "free-standing objects" and SELECT_OBJECT to SQL (probably SQL4, if ever); ensure that OIDs (that is, surrogates for objects) from an OQL database can be "stored" (without interpretation) in an SQL3 database
4	Fast-track or otherwise process OQL as an independent standard	As an alternative tack, propose to include the OQL language itself as part of ("as a Part of"?) the SQL standard

It also seemed reasonable to draw an analogy between the relationships between Ada and SQL and the relationships between OQL and SQL. Readers will recall that X3H2 and DBL RG defined a "traditional" (from SQL's point of view) binding between Ada and SQL. This style of binding was unacceptable to parts of the Ada community, so they initiated the SAMeDL project (recently completed and standardized) to "write SQL in Ada". Similarly, an enhanced OQL merged into SQL could be thought of as providing a "native C++" (or other OOPL) expression of database (relational plus object) semantics.

The reader may recall that OQL *per se* is only a query language and not a computationally-complete programming language. Instead, the ODMG model depends on OQL being embedded in an OOPS such as C++ or Smalltalk for its programming facilities. (In this way, the OOPS serves an analogous function to SQL/PSM or to SQL-92's language embeddings.)

1.4 Summary

The upshot of all this is that we may be closer than we originally thought to a possible rapproachment—or even unification—between SQL3 and ODMG/OQL. There appear to be only a few points (and that SMOP, of course) to be addressed in SQL3:

1) Maintain a clear and clean distinction between objects and values, while continuing to leverage as much common specification as feasible among them;

2) Specialize the function dispatch model as outlined above;

3) Enhance triggers and views to cover objects as well as tables;

4) Ensure that ODMG OIDs can be stored in SQL3 databases and vice versa (without interpretation, implying without actual interoperability)

5) Add a "SELECT_OBJECT" statement to permit "real" object access à la OQL.