1979

Rational Data Base Standards: An Examination of the 1978 CODASYL DDLC Report

Eric K. Clemons
University of Pennsylvania

Follow this and additional works at: https://repository.upenn.edu/mgmt_papers

Part of the Management Sciences and Quantitative Methods Commons

Recommended Citation

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/mgmt_papers/253
For more information, please contact repository@pobox.upenn.edu.
Rational Data Base Standards: An Examination of the 1978 CODASYL DDLC Report

Abstract
The CODASYL Data Description Language committee’s 1978 Report incorporates numerous enhancements and language changes made since the earlier 1971 and 1973 reports. Unfortunately, the major design limitations associated with these earlier specifications, in particular a schema facility too closely related to machine rather than enterprise requirements and an extremely limited subschema facility, are retained.

After examination of these limitations, we suggest that the recent CODASYL specifications remain inappropriate as either an instance of an ANSI/SPARC three-schema architecture or as a candidate for a national data base system standard. A long term strategy for the development of a more rational proposal for standardization is suggested. And a short term strategy is offered, one that permits rational planning for and implementation of data base conversions to occur today, without concern that subsequently developed standards might render obsolete the conversion effort and data base management system selected.

Disciplines
Management Sciences and Quantitative Methods

This journal article is available at ScholarlyCommons: https://repository.upenn.edu/mgmt_papers/253
RATIONAL DATA BASE STANDARDS:
AN EXAMINATION OF THE 1978 CODASYL DDLC REPORT

ABSTRACT

The CODASYL Data Description Language Committee's 1978 Report incorporates numerous enhancements and language changes made since the earlier 1971 and 1973 reports. Unfortunately, the major design limitations associated with these earlier specifications, in particular a schema facility too closely related to machine rather than enterprise requirements and an extremely limited subschema facility, are retained.

After examination of these limitations, we suggest that the recent CODASYL specifications remain inappropriate as either an instance of an ANSI/SPARC three-schema architecture or as a candidate for a national data base system standard. A long term strategy for the development of a more rational proposal for standardization is suggested. And a short term strategy is offered, one that permits rational planning for and implementation of data base conversions to occur today, without concern that subsequently developed standards might render obsolete the conversion effort and data base management system selected.
I. INTRODUCTION

We are addressing two related questions:

1. What is the suitability of the CODASYL 1978 DDL specifications [13] as a candidate for adoption as a national data base system standard?


I think that many arguments in favor of rapid agreement on a data base standard are clear. Every organization has a large investment in data and data processing software; there is pressure on management to convert to a data base architecture, converting existing data and programs to realize the savings and additional benefits believed to accrue from an integrated data base management system; and it is crucial that the considerable expense associated with this conversion not be wasted by subsequent agreement on a standard that renders obsolete the data base system chosen [4]. Likewise, as users wish to avoid the expenses of unnecessary data base conversions, so too do implementors and vendors of data base systems wish to avoid unnecessary modifications and alterations of their products. Indeed, since the 1978 CODASYL specifications differ significantly from earlier specifications [19], there is a certain reluctance on the part of some implementors to modify their systems to meet these new specifications, because there is no guarantee that they will remain fixed for a period sufficient to recover conversion costs.

Systems conforming to CODASYL specifications have been chosen by many corporate users; likewise, CODASYL is the only model with sufficient vendor support to be considered as a serious candidate for a standard. In fact, the CODASYL specifications are rapidly emerging as a de facto American data base system standard. I feel very strongly that this is unfortunate; the CODASYL model, in its present form, is largely inappropriate.

Fortunately, there exists an alternative to the premature adoption of a standard: It is only necessary to decide on a "kernel" of a standard, a component of the programmer interface that will be supported in any future data base standard. Here, the CODASYL model fares somewhat better. It is in widespread use, making it a logical choice. And the ANSI/SPARC proposals which will no doubt have a major influence on future data base management system technology permit great flexibility in any subsequently adopted standards; thus the kernel may be only one of several, dramatically different interfaces supported. Also,
the low level of the CODASYL data manipulation language and
the limited inter-schema mapping facilities supported should
make inclusion of a CODASYL interface relatively easy and
inexpensive.

II. SHORTCOMINGS OF CODASYL SPECIFICATIONS

My principal objection to the CODASYL system is its
lack of concern for and support of the programming user.
This is not an objection to the design, level, or syntax of
the current DML -- if so it would be only a superficial
objection -- rather, it is an objection to the form of
subschema provided.

The CODASYL system is not appropriate as an instance of
the ANSI/SPARC three-schema architecture. It pre-dates the
ANSI/SPARC proposal and does not successfully capture its
philosophy. While the 1978 DDL specifications include a
proposal for a new data storage description language (DSDL)
and thus include three schemas, they are not the correct
three schemas: The DDL schema is not purely conceptual, but
contains constructs better placed in the internal schema as
they deal primarily with access efficiency [10]. The
subschema facility is even farther from an external schema
facility, including both conceptual and internal level
constructs. The resulting design is not clean and does not
provide adequate separation of functions; this is
significant, not because ANSI/SPARC proposal represents an
absolute standard that must be closely followed, but because
the limitations of the selected CODASYL design have
unfortunate implications for programming ease and programmer
productivity, data independence, and distributed processing.

Likewise, I feel that the CODASYL system is not
appropriate for adoption as a national data base standard,
again because of limitations of the subschema facility and
the programming interface. In order to understand the
orientation and limitations of the system, it is necessary
to remember the period -- late 1960s -- in which its
original design and specification were prepared. The
principal concerns of the Data Base Task Group were to
provide a limited increase in flexibility and generality of
data base systems without incurring substantial penalties in
reduced machine efficiency. Thus, networks of associated
records provide greater generality than simple hierarchies;
by freezing the supported associations to be those
explicitly declared in sets, flexibility is limited but
efficient access is assured. Similarly, by limiting maps
between schema and subschemas to a few simple forms,
efficient operation is preserved. Unfortunately, the
resulting design, while efficient, is too limited; in
several ways it is inappropriate for the technology and
demands of contemporary data processing, a decade later and
in the future.

These limitations stem, principally, from the fact that an subschema follows the schema too closely in form. Individual records in the schema map to single records in the subschema, and data associations remain by set membership. In general, networks exist in a data base not because any single user requires so general a structure, but because the collection of hierarchical associations required by each user are incompatible [7]. Thus, if one user wants a hierarchical association between courses he taught and all student grades for the courses:

```
COURSE-REC:  COURSE-ID  CREDITS  FACULTY-NAME

STUDENT-REC:  STUDENT-NAME  GRADE
```

while another user wants a hierarchical association between a student and all course grades received:

```
STUDENT-REC:  STUDENT-NAME

COURSE-REC:  COURSE-ID  CREDITS  GRADE  TERM
```

this will probably be captured at the conceptual level with a network of the following form:
At the external or subschema level users should not see networks but rather the hierarchies required for their individual applications. In fact, where possible the details of the conceptual schema, its record types and set associations, should be hidden from the user. Navigation, data association made using DML statements exploiting set membership, is only slightly removed from manipulation using record keys or device addresses. Such navigation should not be necessary. Rather, subschema records should be in direct correspondence, not with schema records, but with the cognitive structures used by programmers in the solving of problems and the design of algorithms. Thus a STUDENT-TRANSCRIPT subschema record would be a single record comprising student name and a repeating group containing course, grade, and term data; the user would request this record with a single DML statement, although it may correspond to dozens of schema records, of four record types, linked by membership in three sets.

The design limitations of the CODASYL subschema facility have undeniable implications for the process of application program development, maintenance, and execution.

1. Because the subschema structures are in close correspondence, not with user cognitive structures, but with structures provided for the complete enterprise data model, considerable user navigation is required to make necessary data associations and to construct the relevant information objects. This process is difficult, slow, and prone to error; obviously programmer productivity is affected.

2. In the CODASYL model, changes or extensions to the set of supported applications may well result in major structural changes to the schema; e.g., addition of a new application may change a schema
hierarchy to a confluency. Because of the close correspondence between schema and subschema records, the application programs are not buffered from this change, and thus may require major redesign and reprogramming effort. Moreover, the semantics of existing data associations, made by uMl accesses and host language iteration and qualification, are very difficult to determine from the programs. Redesign will not be an easy, automated process; rather it will be manual and difficult. Obviously, data independence is affected [21].

3. Again, because of the level of CODASYL DML and the close relationship between schema and subschema, a number of data selection procedures (e.g., ignore records with the following data values) and data reduction procedures (e.g., return only average balances, grouped by class and status of account) are performed by the application programs. Specified in the schema to subschema map, these procedures could be performed by a "data base machine" supporting the DBMS, rather than by the user program, substantially reducing the volume of data actually returned to the user program. Thus, channel traffic and communications expenses in a distributed environment are affected.

To make concrete the terms and objections stated, we consider as an example a data base again containing student course information. In the schema we have student records related to grade, course, and section grades as follows:

```
COURSE-REC:
  COURSE-ID
  CREDITS

SECTION-REC:
  SECTION-ID
  FACULTY-NAME
  TERM

STUDENT-REC:
  STUDENT-NAME

GRADE-REC:
  GRADE
```

From this we want to construct a summary transcript, with student name, average grade point, and average grade point
For each term:

01 SUMMARY-TRANSCRIPT.
02 STUDENT-NAME ...
02 GRADE-POINT ...
02 TERM-ENTRY OCCURS ...
03 TERM-ID ...
03 TERM-AVERAGE ...

With an external schema facility, retrieval of this transcript is requested with a single READ; changes to the conceptual schema structure that change record types and associations alter inter-schema mapping functions but not application programs; and in a distributed environment the data base machine can transmit the desired summaries, rather than the grade and course credit and term information needed to compute these summaries. Also, we note that employing one current DML to compute these summaries, the user must:

1. FIND all GRADE records for a student
2. for each GRADE, FIND and GET the owner SECTION record
3. sort SECTION records in ascending order by term
4. make each SECTION record current, in order by term
5. for each SECTION record, as it becomes current, FIND and GET the owner COURSE record to get credit information. Also, for each current SECTION and the desired student, the member GRADE record must again have a FIND and GET to get the actual grade received.
6. with the information obtained in the preceding step, host language arithmetic statements are used to compute the desired averages.

Clearly, obtaining the information with a single READ is preferable.

III. AN ALTERNATIVE EXTERNAL SCHEMA FACILITY

It is of limited usefulness to criticize a system design, without proposing an alternative. As an alternative, I offer a greatly enhanced subschema facility, one that in effect offers each user a virtual data base with simple structure corresponding to the specific needs of each application program.
Such a facility has three basic requirements. To construct schema to subschema maps it is necessary to specify:

1. access information
2. restructuring information
3. data item definition

Access information specifies from which records data are to be obtained, what data values are necessary for qualification, and which set membership or other access paths are to be employed to make the necessary associations. Restructuring information controls repetition (e.g., the inclusion of all term summaries in a single summary transcript in the example of section II), grouping (e.g., grouping of grade information by the term that the course was taken), and whether complete content or summary only data are to be included (e.g., include only summary over-all average and term averages, but no individual course grades). Data item definition includes specifying the source of data items actually present in the schema, as well as rules for preparing virtual computed items and structured items. A detailed description of such a general external schema facility for a relational environment is available [7]; language enhancements for a CODASYL system are in preparation [11]. Such a facility will greatly simplify the programmer's interaction with data base systems, while leaving concern for enterprise support and machine efficiency to other schema levels, as is appropriate.

IV. A CANDIDATE FOR STANDARDIZATION?

I do not propose that any current research on external schema facilities be given serious study as a candidate for standardization at this time. Several technical problems remain, requiring technical study; likewise, several questions concerning human factors design and performance remain unanswered. An efficient implementation of a general external schema facility appears difficult; naive approaches suffer from explosive growth of required secondary storage and machine processing time. Equally important, the problem of data base update in a multi-schema environment remains unsolved: surprisingly few maps from conceptual schema to external schema are invertible, implying that for most user updates to data at the level of the user's virtual data base, corresponding changes to the stored data base cannot be determined [5, 8].

Perhaps the most important consideration in any language, interface, or architecture design is their effect on programmer performance, in particular programmer
Rational Data Base Standards

Productivity and program correctness and ease of maintenance. There has been some interest in human factors study and some guidelines have been given [20]; some interesting experiments have been performed [16, 17, 22] but there has been no conclusive work produced.

I estimate that resolution of technical design problems and human factors questions is two or three years in the future; preparation of potential standards, based on this work, will require still more time.

V. WHAT DO WE DO NOW?

It is apparent that we cannot wait three to five years for the adoption of national standards, but must act now. Perhaps it is more accurate to say that if we do not act rapidly, we will have lost the potential for rational choice: sheer volume of existing implementations and in-progress conversions based on systems currently commercially available will dictate a standard.

Therefore, my suggestion made originally in section I appears reasonable: We should agree that any future standard for data base architecture must include the current CODASYL DML and subschema facility in its programmer interface, permitting data base conversions to be planned and performed now. We should also agree that, after five years, the facilities for CODASYL schema, subschema, and WDL schema will be re-evaluated, based on advances in the areas of external, conceptual, and internal schema research. Perhaps, as a result of these advances, CODASYL specifications will have only limited resemblance to current specifications. Or, perhaps, future standards will preserve nothing of the current CODASYL specifications beyond that which is explicitly included in the kernel.

I believe that much additional research in the area of conceptual schema is required. Recent work by Bachman and Daya [3], Chen [6], and Gerritsen and Lee [15] indicate the potential for representing data base semantics as well as structure in the schema. Work on external schema facilities, based on my own research cited earlier and the implementation results of the IBM System R group [2] must continue, and must be subjected to human factors study and valuation. Work by CODASYL at the internal schema level will continue. It is to be hoped that the results of these separate efforts can be combined, within the framework of an ANSI/SPARC three-schema architecture, to produce a data base architecture appropriate to the needs of business and government in the decade ahead.
ACKNOWLEDGEMENTS

I would like to acknowledge my debt to my colleagues at the University of Pennsylvania, Rob Gerritsen and Frank Uermano, to my co-panelists at the Fourth Very Large Data Base Conference, John Berg, Frank Manola, and Diane C. P. Smith; and to my co-participants at the N.B.S. Three-Schema Feasibility Workshops, particularly Don Chamberlin, Henry Lefkowitz, and Carlo Zaniolo. Their thinking has influenced my thinking. Obviously, biases, errors, and omissions in this paper are my own.

REFERENCES


<table>
<thead>
<tr>
<th><strong>DISTRIBUTION LIST</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Department of the Navy - Office of Naval Research</td>
<td></td>
</tr>
<tr>
<td>Data Base Management Systems Project</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Defense Documentation Center</strong></td>
<td><strong>Office of Naval Research</strong></td>
</tr>
<tr>
<td>(12 copies)</td>
<td>(2 copies)</td>
</tr>
<tr>
<td>Cameron Station</td>
<td>Information Systems Program</td>
</tr>
<tr>
<td>Alexandria, VA 22314</td>
<td>Code 437</td>
</tr>
<tr>
<td>Arlington, VA 22217</td>
<td>Arlington, VA 22217</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Office of Naval Research</strong></td>
<td><strong>Office of Naval Research</strong></td>
</tr>
<tr>
<td>Code 102IP</td>
<td>Branch Office</td>
</tr>
<tr>
<td>Arlington, Virginia 22217</td>
<td>493 Summer Street</td>
</tr>
<tr>
<td></td>
<td>Boston, MA 02210</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Office of Naval Research</strong></td>
<td><strong>Office of Naval Research</strong></td>
</tr>
<tr>
<td>Branch Office, Chicago</td>
<td>Branch Office, Pasadena</td>
</tr>
<tr>
<td>536 South Clark Street</td>
<td>1030 East Green Street</td>
</tr>
<tr>
<td>Chicago, IL 60605</td>
<td>Pasadena, CA 91103</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>New York Area Office</strong></td>
<td>Naval Research Laboratory</td>
</tr>
<tr>
<td>715 Broadway - 5th Floor</td>
<td>(6 copies)</td>
</tr>
<tr>
<td>New York, NY 10003</td>
<td>Technical Information Division</td>
</tr>
<tr>
<td></td>
<td>Code 2627</td>
</tr>
<tr>
<td></td>
<td>Washington, DC 20375</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Dr. A. L. Slaflkosky</strong></td>
<td><strong>Office of Naval Research</strong></td>
</tr>
<tr>
<td>Scientific Advisor</td>
<td>Code 455</td>
</tr>
<tr>
<td>Commandant of the Marine Corps</td>
<td>Arlington, VA 22217</td>
</tr>
<tr>
<td>(Code RD-1)</td>
<td></td>
</tr>
<tr>
<td>Washington, DC 20380</td>
<td>Navy Electronics Laboratory Center</td>
</tr>
<tr>
<td></td>
<td>Advanced Software Technology Division</td>
</tr>
<tr>
<td></td>
<td>Code 5200</td>
</tr>
<tr>
<td></td>
<td>San Diego, CA 92152</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Office of Naval Research</strong></td>
<td></td>
</tr>
<tr>
<td>Code 456</td>
<td></td>
</tr>
<tr>
<td>Arlington, VA 22217</td>
<td></td>
</tr>
</tbody>
</table>