TSQL2: A Design Approach

September 16, 1994

A TSQL2 Commentary

The TSQL2 Language Design Committee

Title	TSQL2: A Design Approach
Primary Author(s)	Richard T. Snodgrass
Publication History	April 1992. White Paper. September 1994. TSQL2 Commentary.

TSQL2 Language Design Committee

Richard T. Snodgrass, Chair	University of Arizona
rts@cs.arizona.edu	Tucson, AZ
Ilsoo Ahn	AT&T Bell Laboratories
ahn@cbnmva.att.com	Columbus, OH
Gad Ariav	Tel Aviv University
ariavg@ccmail.gsm.uci.edu	Tel Aviv, Israel
Don Batory	University of Texas
dsb@cs.utexas.edu	Austin, TX
James Clifford	New York University
jcliffor@is-4.stern.nyu.edu	New York, NY
Curtis E. Dyreson	University of Arizona
curtis@cs.arizona.edu	Tucson, AZ
Ramez Elmasri	University of Texas
elmasri@cse.uta.edu	Arlington, TX
Fabio Grandi	Universitá di Bologna
fabio@deis64.cineca.it	Bologna, Italy
Christian S. Jensen	Aalborg University
csj@iesd.auc.dk	Aalborg, Denmark
Wolfgang Käfer	Daimler Benz
kaefer%fuzi.uucp@germany.eu.net	Ulm, Germany
Nick Kline	University of Arizona
kline@cs.arizona.edu	Tucson, AZ
Krishna Kulkarni	Tandem Computers
kulkarni_krishna@tandem.com	Cupertino, CA
T. Y. Cliff Leung	Data Base Technology Institute, IBM
cleung@vnet.ibm.com	San Jose, CA
Nikos Lorentzos	Agricultural University of Athens
eliop@isosun.ariadne-t.gr	Athens, Greece
John F. Roddick	University of South Australia
roddick@unisa.edu.au	The Levels, South Australia
Arie Segev	University of California
segev@csr.lbl.gov	Berkeley, CA
Michael D. Soo	University of Arizona
soo@cs.arizona.edu	Tucson, AZ
Suryanarayana M. Sripada	European Computer-Industry Research Centre
sripada@ecrc.de	Munich, Germany

Contents

List of Figures

1 Motivation

Many within the temporal database research community perceive that the time has come to consolidate approaches to temporal data models and calculus-based query languages, to achieve a consensus query language and associated data model upon which future research can be based. While some two dozen query language proposals exist, with a diversity of language and modeling constructs, common themes keep resurfacing. However, the community is quite fragmented, with each research project being based on a particular and different set of assumptions and approaches. Often these assumptions are not germane to the research *per se*, but are made simply because the research required a data model or query language with certain characteristics, with the particular one chosen rather arbitrarily. It would be better in such circumstances for research projects to choose the *same* language. Unfortunately, no existing language has attracted a following large enough to become the one of choice.

As a result of an ARPA/NSF workshop on an infrastructure of temporal databases, a language design committee has been formed to develop a specification for a consensus extension to SQL-92 that could form a common core for future research. This extension is termed the *Temporal Structured Query Language*, or TSQL2. This commentary outlines the process by which a design for TSQL2 could be produced by the research community.

2 Scope

The scope of the TSQL2 language design should be restricted so that a coherent design is possible. This section lists aspects that should be included, and perhaps more importantly, those that should *not* be included.

- **TSQL2 is to be a *relational* query language.**

  Given that SQL is "intergalactic dataspeak" (Mike Stonebraker's term), TSQL2 should whenever possible be consistent with standard SQL, specifically, SQL-92. It simply doesn't make sense to base TSQL2 on competing (and arguably better) query languages such as Quel, Datalog, or Daplex. While it is certainly the case that interesting research is possible and even desirable in extending the other languages to include temporal support, such extensions are necessarily outside of the scope of TSQL2.

- **TSQL2 need not be consistent with existing standards.**

  In general, TSQL2 need *not* be consistent with SQL-92, which is now an ANSI and IOS standard, nor with SQL3, which is currently being designed. SQL-92 contains severe flaws in its (minimal) handling of time-stamps, and SQL3 is a moving target which, in its present state, is regarded by many as a baroque design with a bewildering array of features. Consistency with the non-temporal aspects of existing standards for SQL, including SQL89, SQL-92, and SQL3, is desirable if such consistency does not conflict with other goals.

- **TSQL2 will not be another standard.**

  While the goal is a fully elaborated language design, there is no expectation that this design will be made into a standard. Of course, one hopes that our results would be acceptable to the standards bodies; at a minimum, our design should be communicated to these bodies. However, it is important to keep in focus the objective of the TSQL2 design: to provide a basis for future *research* in temporal databases. It also must be emphasized that TSQL2 should in no way limit or constrain future research in temporal databases, which should be free to adopt or propose whatever linguistic constructs are appropriate.

- **TSQL2 will not be an object-oriented query language.**

  While temporal object-oriented query languages are being actively investigated, it would be distracting and counter-productive at this stage to attempt to merge the rather disparate approaches of object-oriented and relational languages while also addressing the temporal processing needs. Those involved in object-oriented language design are encouraged to produce, in parallel with this effort, a temporal object-oriented extension to SQL. At a later date, the two extensions could be merged.

- **TSQL2 should be comprehensive.**

  TSQL2 should have constructs, extended in a natural fashion, that support all of the functionality of SQL, including update, aggregates, and schema specification and evolution. Consistent with the modifier "temporal", TSQL2 should support both valid and transaction time.

- **The language design should include a formal semantics.**

  Fortunately, there is a tradition of rigor in the temporal database community. The recent publication in TODS of a straightforward semantics of SQL will also help here.

- **The language will have an associated algebra.**

  Such an algebra would demonstrate the existence of an executable equivalent to the declarative constructs in the language, and would suggest implementation strategies.

- **TSQL2 will be a *language* design.**

  The TSQL2 design should not attempt to define storage structures, indexing structures, access methods, fourth-generation interfaces, support for distributed systems or heterogeneous databases, or optimization techniques. Such aspects, while important, are more properly the target of the research efforts that will utilize TSQL2 as a common substrate.

- **TSQL2 should reflect areas of convergence.**

  The design of TSQL2 should avoid active areas of research where new results are generated frequently. Such areas include support for recursion and temporal database design.

3 Language Design Process

It is in everyone's best interest to have as many participants in the design as possible. It would be wonderful to tap the extensive expertise available in the research community. On the other hand, the process must balance the desirability for input with the necessity of a design by a small number of designers, to avoid "design by committee" and all the difficulties such a design necessarily brings upon itself. Fortunately, there is a natural limiting mechanism available. Simply put, the design should be done by those researchers willing to expend the (considerable) effort to produce initial proposals and/or to modify designs in response to comments from a much larger community of evaluators.

Language designers will be self-selected persons who are willing to write *commentaries* on some specific aspect of the design. commentaries will include a survey of relevant research and a concrete proposal for some component of TSQL2. Generally the proposal will include a formal syntax of the suggested constructs, an informal semantics (in prose) of these constructs, and, ideally, a formal semantics. These commentaries should explicitly state the rationale behind important design decisions, to enable concrete discussion of the proposals.

Evaluators will be self-selected persons willing to comment in writing on a commentary. The comments will be collected, and addressed either by the author(s) of the initial commentary or by other designers willing to produce a new draft of the commentary.

The process will be iterative, and will converge when everyone is satisfied (or exhausted). The committee will be open to anyone who volunteers, and will consist of both language designers and evaluators.

4 Tasks

Here a series of tasks are listed, culminating in a fully elaborated language specification. Each task has as its goal the production of and agreement on a commentary that addresses the indicated portion of the language.

4.1 Terminology

An agreed-upon set of concepts must be the first order of business. Fortunately, the previously mentioned workshop has produced a consensus glossary of temporal database concepts.

4.2 Physical Time Line and Time-stamp Representation

Current DBMS's assume a time line starting at 1 A.D. or later and consisting of days or seconds, up to 9999 A.D.. One difficulty is that there are several definitions of *second* and of *day*. Another difficulty is that such a limited time line is of little use to many potential users of a temporal database, such as geologists, archæologists, anthropologists, and astronomers. Such a time line doesn't even include all of recorded history, and so doesn't fully support historians. Expanding the time line back to the creation of the universe (approximately 15 billion years ago), raises other definitional questions. For example, a solar year in the time of the dinosaurs was 400 days long. A year is difficult to define more than 6 billion years ago, before the earth was formed.

What is needed is an application-independent identification of one or more physical clocks that cover all of past time (15 billion years) and all of the foreseeable future. This definition of a physical time line should be convertible to other definitions that might be useful. A representation as a time-stamp data structure is also needed, with a precise semantics, i.e., a correspondence with a particular time of this physical clock for each valid bit pattern. Decisions need to be made about treating events as infinitely small points in time or as chronons of finite but nondecomposable length, closed or open representations for intervals, granularity, discrete versus continuous time, bounded versus unbounded time, and linear versus branching time.

4.3 User-defined Time Domain

In conventional as well as time-oriented databases, individual attributes can be associated with a temporal domain, termed *user-defined time*. Such a domain is supported by the DBMS in similar ways to other specialized domains, such as money, e.g., conversion to and from a string representation and the availability of comparison predicates. While SQL-92 and DB2's SQL include two time-oriented attribute domains, *datetimes* and *intervals*, these language variants are limited to a single calendar, the Gregorian calendar, offer little support for anchored intervals, do not support languages other than English, and exhibit many problems with the semantics of arithmetic and boolean expressions. A proposal is needed that addresses these problems, while providing appropriate constructs for schema definition, time value input and output, predicates, arithmetic manipulation, and temporal functions.

4.4 Underlying Valid-time Relational Data Model

Determining the correct data model underlying TSQL2 will probably be the most difficult of all the tasks. Unfortunately, and not coincidentally, this task is a central one, on which most of the other tasks are predicated. To focus the design, I advocate that time be added to the data model in two separate steps, with the first to add valid time and the second to later add transaction time.

A proposal is needed that confronts the controversies currently raging in the research community, including 1NF versus ¬1NF, temporally grouped versus temporally ungrouped, tuple time-stamped versus attribute value time-stamped, homogeneous versus non-homogeneous, events versus intervals, interpolated versus stepwise constant data, recurrent events, and whether keys should be required.

4.5 Benchmark Queries

A basis is needed on which to compare language proposals. This task involves informally defining an example schema containing several relations, populating this schema with example relation instances, listing in English prose interesting queries on this schema, and displaying the results of these queries on the example instances.

4.6 Valid-time Selection and Projection

Valid-time selection is the analogue of conventional selection: the identification of tuples that satisfy some specified predicate, in this case a predicate on the time(s) the data elements (attribute values or tuples) were valid. One design issue is whether the `where` clause in SQL should be extended, or whether a new clause is preferred.

Valid-time projection is an analogue of conventional projection, where component(s) of tuples are retained, in this case, components of the time(s) the data elements were valid. One fundamental question is whether the derived intervals must be subsets of the underlying intervals. A second design issue is whether the target list in SQL should be extended, or whether a new clause is preferred. The subject of temporal joins, such as time intersection, time union, and temporal outer joins, also needs to be addressed here.

4.7 Aggregates

Extension of the current SQL aggregates is required, along with the definition of new time-oriented aggregates (e.g., `first`), of temporal analogues of aggregate variants such as `unique` (e.g., moving window), and of order-dependent predicates that operate on groups of tuples.

4.8 Schema Specification and Evolution

SQL has a create table statement. This will need to be extended to allow specification of time-varying relations in addition to conventional relations. Other meta-data, such as the nature of interpolation to be imposed on continuous data represented discretely, must be included in the schema. Also, the schema of a relation may need to be changed to indicate a conversion from a time-varying relation, or vice versa. This will probably be a particularly easy extension to design.

4.9 Add Transaction Time to the Data Model

Since transaction time is orthogonal to valid time, the design process will be simplified if these two aspects are attacked separately. The hope is that once the impact of adding valid time to the language has been adequately considered, the incorporation of transaction time will be easier. Some feel that transaction time can be handled identically or almost identically to valid time; clearly if this is possible it will simplify this task considerably.

4.10 Schema Versioning

When schema evolution and support for transaction time are both present, a database may contain multiple versions of the schema, each in effect for disjoint intervals of transaction time. This aspect, while difficult to implement, probably has little impact on the language design.

4.11 Transaction Time Selection and Projection

At a minimum, these constructs should support rollback. A design decision is whether the valid-time selection and projection constructs should be extended, or whether different constructs are needed.

4.12 Incorporate All SQL Constructs

To arrive at a comprehensive language definition, the interaction between proposed language constructs concerning time and all existing constructs, including modules, embedded SQL, views, and protection, needs to be examined in a systematic fashion.

4.13 Core Algebra

This task involves the design of representations for temporal relations (the objects in the algebra) and operators on these objects to support valid-time selection and projection. Issues including uni-sorted versus multi-sorted, algebraic equivalences, closure, snapshot reducibility, and update semantics should be considered.

4.14 Add Aggregates to the Algebra

Clearly the algebra should support all the aggregate variants present in the TSQL2 design.

4.15 Add Transaction Time to the Algebra

If schema versioning is to be supported in the algebra, this task must consider how algebra expressions are to be type checked in the presence of multiple schemas active at various transaction times.

5 Prerequisites

Dependencies between the tasks result in a partial order on their completion, as shown in Figure 1. These dependencies take the following considerations into account.

- The design of the time-stamp representation and of the user-defined time domain are independent of extensions of the underlying data model to incorporate valid or transaction time.
- The constructs for valid-time selection should be consistent with those used in expressions involving user-defined time.
- The constructs for valid-time selection and projection, unlike those for user-defined time, require a specified data model.
- While constructs for schema specification do not require the operations of valid-time selection and projection to be elaborated, the constructs for schema evolution will require new tuples to be generated that are consistent with a modified schema.
- Schema versioning is only possible if transaction time is supported.

While commentaries that address only one task are best, those that address several tasks in concert will certainly still be welcome. Also, these dependencies are only suggestive; work can certainly proceed on multiple tasks concurrently.

6 History

Temporal databases have been an active research topic for at least fifteen years. During this time, several dozen temporal query languages have been proposed. In April, 1992 Richard Snodgrass circulated a white paper, which was the initial version of this commentary, proposing that a temporal extension to SQL be produced by the research community. In parallel, the temporal database community organized the "ARPA/NSF International Workshop on an Infrastructure for Temporal Databases," which was held in Arlington, TX, in June, 1993. Discussions at that workshop indicated that there was substantial interest in a temporal extension to SQL-92. A general invitation was sent to the community, and about a dozen people volunteered to develop a language specification. Several people later joined the committee. The group corresponded via electronic mail from early July, 1993, submitting, debating, and refining proposals for the various portions of the language. In September, 1993, the first draft specification, accompanied by thirteen commentaries, was distributed to the committee. In December, 1993 a much enlarged draft, accompanied by some twenty-four commentaries, was distributed to the committee. A preliminary language specification appeared in the March, 1994 issue of *ACM SIGMOD Record*, and twenty-three commentaries were made available via anonymous FTP at `FTP.cs.arizona.edu`. A tutorial of the language appeared in the September, 1994 issue of *ACM SIGMOD Record*, and the final language specification and 28 commentaries were also made available via anonymous FTP that month. The design of the language and the writing of the commentaries followed the process shown in Figure 1; all the tasks identified in the figure have been accomplished in the commentaries and in the language design.

Acknowledgements

The comments of many people who read a previous draft of this initiative, especially those of James Clifford, Shashi Gadia, Sushil Jajodia, Christian S. Jensen, Sham Navathe, Arie Segev, and Abdullah Tansel, are appreciated. This work was supported in part by NSF grants ISI-8902707 and ISI-9302244, IBM contract #1124 and the AT&T Foundation.

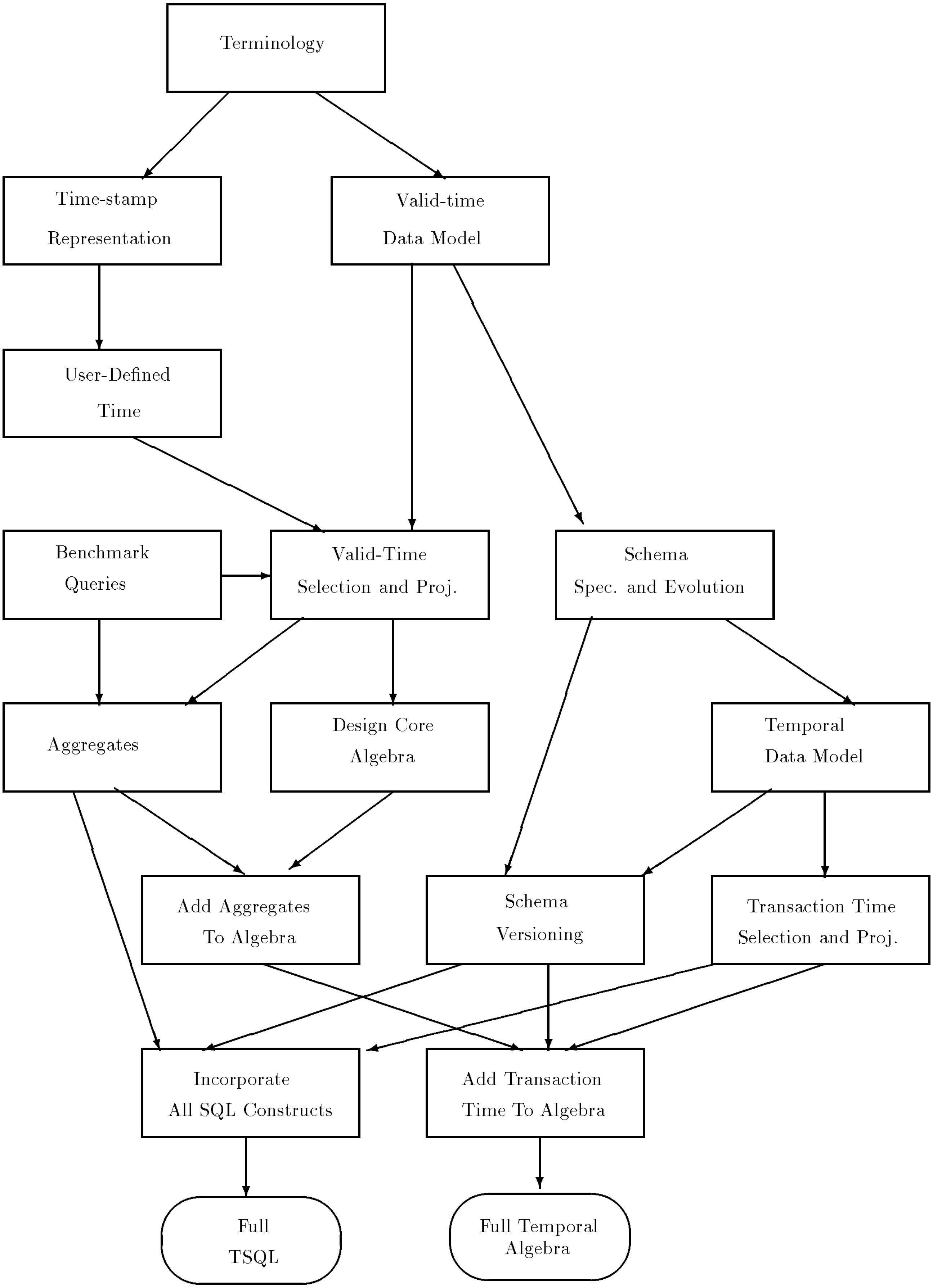

Figure 1: Task Dependencies