The TSQL2 Data Model for Time

September 23, 1994

A TSQL2 Commentary

The TSQL2 Language Design Committee

Title	The TSQL2 Data Model for Time
Primary Author(s)	Curtis E. Dyreson, Michael D. Soo and Richard T. Snodgrass
Publication History	September 1994. TSQL2 Commentary.

Abstract

A glib characterization of the TSQL2 model of time is that it is a "don't ask, don't tell" model. The TSQL2 model of time refrains from deciding whether time is ultimately continuous, dense, or discrete, nor can a user ask a question that will differentiate among the alternatives. Instead, the model accommodates all three alternatives by assuming that an *instant* on a time-line is much smaller than a *chronon*, which is the smallest entity that a timestamp can represent exactly. Consequently, an instant can only be approximately represented. A discrete image of the represented times emerges at run-time as timestamps are *scaled* to user-specified (or default) granularities and as operations on those timestamps are performed to the given scale.

1 Introduction

TSQL2 adds comprehensive support for *time* to SQL-92. This support rests on three temporal dimensions: user-defined time, valid time, and transaction time. A single model of time, however, is the foundation for each of these separate dimensions. In this commentary, we present TSQL2's model of time. We focus on the concepts of an instant and a period, discuss how each is modeled in TSQL2, and give an overview of the semantics of operations on the modeled entities.

A word of caution, the experienced temporal database researcher will likely find this commentary to be somewhat dry as the TSQL2 model of time encompasses the basic models found in most temporal database papers.

2 The TSQL2 Model of the Time

In the temporal database community, three basic time models have been proposed: the *continuous model*, in which time is viewed as being isomorphic to the real numbers, with each real number corresponding to a "point" in time, the *dense model*, in which time is viewed as being isomorphic to the rationals, and the *discrete model*, in which time is viewed as being isomorphic to the integers [Clifford & Tansel 1985]. Science and metaphysics have yet to determine which model best fits reality; good arguments can be made for each of the three models. While some temporal database ontologies choose amongst these three alternatives, in the TSQL2 ontology, we view this choice as largely unimportant; time can be either continuous, dense, or discrete. However, since we are modeling time on a discrete computing device, our "view" of time and the uses to which we put that view must necessarily be discrete.

In all three models, a point on the underlying time-line is called an *instant*, and the time between two instants is known as a *time period* (period for short) [1]. A period is uniquely described by two instants; the instant that starts the period and the instant that terminates the period. An instant is a special kind of period since it is an "period" with the same starting and terminating instants. One very important feature of an instant is that it is instantaneous, i.e., has *zero* duration.

[1]The consensus glossary terms these values *intervals* [Jensen et al. 1994], but that term was already used in SQL-92 for an unachored duration of time. We utilize the SQL-92 terminology in TSQL2 to ensure compliance.

2.1 Time is Bounded

In TSQL2, time is modeled as a segment of the discrete, dense, or continuous time-line. Conceptually, while time may extend into the infinite future or the infinite past, in our model, time is bounded at both ends. This is not a crucial feature of TSQL2's model, which could just as easily be unbounded, rather it is a pragmatic choice. Times are stored in fixed-sized data structures called *timestamps*. An unbounded range of times would make such a storage scheme impractical since timestamps would also need to be unbounded in size. The choice is also consistent with a popular cosmology which asserts that time begin at the "Big Bang," 14 ± 4 billion years ago, and will end at the "Big Crunch," sometime in the future [Dyreson & Snodgrass 1993A].

2.2 A Discrete Image of Time

The segment of time can be partitioned into a finite number of discrete, smaller segments known as *granules*. The partitioning is called *scaling* and is dynamic, occurring during execution of a query, rather than static [Dyreson & Snodgrass 1993B]. For example, a scale of "days" partitions the time-line into day-sized granules. Every set of granules is well-ordered. Two special values, *beginning* and *forever*, are the least and greatest values, respectively, in the ordering. Beginning and forever are not granules; rather, they are instants just outside the closed period of time and have special operational semantics.

We do not assume that a TSQL2 implementation will support every conceivable scale; only certain scales will be supported. The smallest supported scale is that of *chronons* [Jensen et al. 1994]. A chronon is the smallest representable granule. The TSQL2 model of time does not mandate a specific chronon size; a chronon may be of any duration (e.g., nanoseconds, years, Chinese imperial dynasties). We believe that specifying the minimum scale should be left to the implementation rather than fixed in the data model.

Scaling creates a discrete image, in terms of granules, of a (possibly) continuous time-line. By choosing an appropriate scale, a user can view the time-line as a discrete set of seconds, or days, or years. The image is created during execution of a query and then discarded, although the act of viewing an instant at a particular scale sometimes changes the information we retain about that instant. We discuss below how instants are modeled at various scales.

2.3 Modeling Instants

To model instants, we introduce an instant timestamp. The smallest amount of time that an instant timestamp can represent is a chronon. But a chronon is a segment of a time-line whereas an instant is a point on a time-line. What then is the relationship between chronons and instants?

We could either assume that chronons are the same size as instants or we could assume that chronons are much bigger than instants, that is, that every chronon contains a large (possibly infinite) number of instants. If we assume that chronons and instants are the same, then we must also use the discrete model of time. If the model of time were continuous or dense then there would have to be an infinite number of chronons because, in either of these two models, there are an infinite number of instants in any non-zero length segment of the time-line. Since the theme of our model is that time is (possibly) continuous, we must assume that chronons are much bigger than instants.

An instant timestamp records that an instant is located sometime *during* a particular chronon or granule (e.g., a day, a year, a month). Without loss of generality, we assume that an instant timestamp represents any instant during a granule. Hence, at a very abstract level, the *exact* instant modeled by an instant timestamp is never precisely known. At best, only the granule during which it is located is known. Two instants that are represented by the same granule may still model different instants. For example, assume that two different instants are positioned during the same hour-long granule. The two instants are represented by identical instant timestamps, each with a scale of an hour.

Alternatively, we might assume that when we represent an instant, we are actually representing the very first instant in each granule. However, the first assumption, that an instant is sometime within a granule better describes actual clock measurements. When we glance at a wristwatch and report that the time is currently "3:45," we typically do not mean the very first instant in the 45th minute after three o'clock, rather, we mean that we simply do not know more than that it is sometime within the minute described by "3:45."

All (current) TSQL2 operations on timestamps support both of these (conflicting) assumptions, and a user will be unable to detect any difference between these two assumptions by testing various operations. Stated differently, if we assume that an instant may occur anywhere within a granule, we may just as easily assume that it is always the first instant within that granule because no timestamp operation uses the location of an instant within a granule. So while the distinction between the two assumptions has no practical import, the first assumption better models the process by which we derive temporal information.

Yet another alternative is to assume that the instant is really the entire chronon, or even an entire granule. However, this is not what is meant by an instant. An instant is instantaneous (of no duration) rather than being a chronon, an hour, a day, or even a year in duration. Nor would such an assumption model the reality of clock measurements. We when glance at our wristwatch and report the time is currently "3:45," we typically do not mean that it is the entire chronon (assume nanoseconds) starting with "3:45."

In summary, an instant is modeled by a timestamp coupled with an associated scale (e.g., day, year, month). The instant modeled by the timestamp is some instant within the granule selected by the associated scale. So an instant timestamp of "June 1, 1993" at the scale of a day models some instant during the 1st day of June in the year 1993. It is important to note that we cannot ask which instant, that is, there is no TSQL2 operation which permits us to ask which instant. Furthermore, there is no TSQL2 operation which will require us to know more about the instant than that it is sometime during that day. For example, the operation that adds the above instant timestamp to a span of "3 days" is akin to asking, "In terms of days, what is some instant during June 1, 1993 displaced by 3 days?" The answer is some instant in "June 4, 1993".

2.4 Modeling Periods

To model periods, we introduce a period timestamp. The period timestamp has an associated scale (e.g., day, year, month, etc.). The timestamp is the composition of two instant timestamps and a constraint. The constraint is that the instant timestamp that starts the period equals or precedes (in the given scale) the instant timestamp that terminates the period. Both the starting and terminating instants have the same scale. For example, the period timestamp for "June 1, 1993 — June 1, 1993" means that the period is all within the same day, but we aren't sure when it begins or ends during that day (note that "June 1, 1993" equals "June 1, 1993" at the scale of days).

2.5 Impact of the Model on the Semantics of Timestamp Operations

As stated earlier, scaling creates a discrete image of a (possibly) continuous underlying time-line. Operations on timestamps are defined with respect to this discrete view of time. For example, a comparison operation at the scale of days compares days, an addition operation at the scale of years adds years, and a duration operation that asks for the duration of an period at the scale of seconds computes that duration in terms of seconds. It is very important to note that a scale of *instants* is not supported; the scale of chronons is the smallest possible scale. Consequently, timestamp operations that are performed at the level of instants do not exist (i.e., we cannot ask if one instant precedes, in an image of instants, another instant). In fact, such a question is not only unaskable, but unanswerable in our model of time since an instant timestamp does not record the exact location of an instant.

3 Summary

The theme for the TSQL2 model of time is that users manipulate a discrete image of a time-line that is possibly continuous, dense, or discrete. The image emerges at run-time when timestamps are scaled to the user-specified granularity. Instant timestamps model durationless temporal values, instants, that are located sometime during a particular granule. Period timestamps model temporal values with duration, periods, that may span one or more granules. Timestamp operations are defined with respect to the discrete image and are performed to the scale of the operands.

Acknowledgements

This work was supported in part by NSF grant ISI-8902707 and ISI-9302244 IBM contract #1124, and the AT&T Foundation. We wish to thank Nick Kline and Christian Jensen for their very insightful comments and contributions.

References

[Clifford & Tansel 1985] Clifford, J. and A. U. Tansel. "On an Algebra for Historical Relational Databases: Two Views," in *Proceedings of ACM SIGMOD International Conference on Management of Data*. Ed. S. Navathe. Association for Computing Machinery. Austin, TX: May 1985, pp. 247–265.

[Dyreson & Snodgrass 1993A] Dyreson, C. E. and R. T. Snodgrass. "The TSQL2 Baseline Clock." Commentary. TSQL2 Design Committee. 1993.

[Dyreson & Snodgrass 1993B] Dyreson, C. E. and R. T. Snodgrass. "Precision Specification in TSQL2." Commentary. TSQL2 Design Committee. 1993.

[Jensen et al. 1994] Jensen, C. S., J. Clifford, R. Elmasri, S. K. Gadia, P. Hayes and S. Jajodia [eds]. "A Glossary of Temporal Database Concepts." *ACM SIGMOD Record*, 23, No. 1, Mar. 1994, pp. 52–64.