# Time Domain

Angelo Montanari* and Jan Chomicki+

*Dipartimento di Matematica e Informatica,
Università degli Studi di Udine, Udine, Italy

+ Department of Computer Science and Engineering
University at Buffalo, SUNY, USA

## Synonyms

Temporal domain, temporal structure

## Definition

In its full generality, a time domain can be defined as a set of *temporal individuals* connected by a set of *temporal relations*. Different choices for the temporal individuals and/or the temporal relations give rise to different temporal ontologies.

In the database context, the most common temporal ontology takes *time instants* (equivalently, points or moments) as the temporal individuals and a *linear order* over them as the (unique) temporal relation [5]. In addition, one may distinguish between discrete and dense, possibly continuous, time domains and between bounded and unbounded time domains. In the discrete case, one may further consider whether the time domain is finite or infinite and, in the case of unbounded domains, one can differentiate between left-bounded, right-bounded, and totally unbounded domains. Moreover, besides linear time, one may consider *branching time*, where the linear order is replaced with a partial one (a tree or even a directed acyclic graph), or circular time, which can be used to represent temporal periodicity.

As for temporal individuals, time instants can be replaced with *time intervals* (equivalently, periods or anchored stretches of time) connected by (a subset of) Allen's relations `before`, `meets`, `overlaps`, `starts`, `during`, `equal`, and `finishes`, and their inverses or suitable combinations [7]. As in the case of instant-based domains, one may distinguish between discrete and dense domains, bounded and unbounded domains, linear, branching, and circular domains, and so on.

Finally, as most temporal database applications deal with both qualitative and quantitative temporal aspects, instant-based time domains are usually assumed to be isomorphic to specific numerical structures, such as those of natural, integer, rational, and real numbers, or to fragments of them, while interval-based ones are obtained as suitable intervallic constructions over them. In such a way, time domains are endowed with *metrical features*.

## Historical background

The nature of time and the choice between time instants and time intervals as the primary objects of a temporal ontology have been a subject of active philosophical debate since the times of Zeno and Aristotle. In the twentieth century, major contributions to the investigation of time came from a number of disciplines. A prominent role was played by Prior who extensively studied various aspects of time, including axiomatic systems of tense logic based on different time domains.

Nowadays, besides physics, philosophy, and linguistics, there is a considerable interest in temporal structures in mathematics (theories of linear and branching orders), artificial intelligence (theories of action and change, representation of and reasoning with temporal constraints, planning), and theoretical computer science (specification and verification of concurrent and distributed systems, formal analysis of hybrid temporal systems that feature both discrete and continuous components). A comprehensive study and logical analysis of instant-based and interval-based temporal ontologies, languages, and logical systems can be found in [2].

As for *temporal databases*, the choice of the time domain over which temporal components take their value is at the core of any application. In most cases, a discrete, finite, and linearly ordered (instant-based) time domain is assumed. This is the case, for instance, with SQL standards [9]. However, there is no a single way to represent time in a database, as witnessed by the literature in the field. To model when something happened, time instants are commonly used; validity of a fact over time is naturally represented by the (convex) set of time instants at which the fact holds, the time *period of validity* in the temporal database terminology; finally, to capture processes as well as some kinds of temporal aggregation, time intervals are needed.

## Scientific fundamentals

**Basics.** The choice between time instants and time intervals as the basic time constituents is a fundamental decision step that all temporal systems have in common. In mathematics, the choice of *time instants*, that is, points in time without duration, is prevalent. Even though quite abstract, such an solution turned out extremely fruitful and relatively easy to deal with in practice. In computer science, additional motivations for this choice come from the natu-

ral description of computations as possibly infinite sequences of instantaneous steps.

The alternative option of taking *time intervals*, that is, anchored stretches of time with duration, as temporal individuals seems to better adhere to the concrete experience of people. Physical phenomena as well as natural language expressions involving time can be more easily described in terms of time intervals instead of time instants. Nevertheless, the complexity of any systematic treatment of time intervals prevents many systems from the adoption of an interval-based ontology.

The instant and the interval ontologies are systematically investigated and compared in [2]. The author identifies the conditions an instant-based (resp., interval-based) structure must satisfy to be considered as an adequate model of time. Then, through an axiomatic encoding of such conditions in an appropriate language, he provides a number of (first-order and higher order) logical theories of both instant-based and interval-based discrete, dense, and continuous structures. Finally, he illustrates the strong connections that link the two time ontologies. In particular, he shows how interval-based temporal structures can be obtained from instant-based ones through the standard process of interval formation and how instant-based temporal structures can be derived from instant-based ones by a (non-trivial) limiting construction.

A metric of time is often introduced to allow one to deal with time distance and/or duration. In particular, a time metric is needed to define calendar times, such as those based on the commonly used Gregorian calendar.

**Temporal models and query languages.** The choice of the time domain has an impact on various components of temporal databases. In particular, it influences temporal data models and temporal query languages.

As for *temporal data models*, almost all of them adopt an instant-based time ontology. Moreover, most of them assume the domain to be linear, discrete and finite. However, many variants of this basic structure have been taken into consideration [8]. Right-unbounded domains have been used to record information about the future. Dense and continuous domains have been considered in the context of temporal constraint databases, that allow one to represent large, or even infinite, sets of values, including time values, in a compact way. Branching time has been exploited in applications where several alternatives have to be considered in the future and/or past evolution of temporal data.

Many data models distinguish between absolute (anchored) and relative (unanchored) time values. Absolute time values denote specific temporal individuals. In general, they are associated with a time metric, such as that of calendar times. As an example, the 14th of September 2007 is an absolute time value that denotes a specific element of the domain of days in the Gregorian calendar. Relative time values specify the distances between pairs of time instants or the durations of time intervals. Absolute and relative time values can also be used in combination. As an example, the expression 7 days after the 14th of September 2007 denotes the 21st of September 2007.

As for *temporal query languages*, they typically assume that time is isomorphic to natural numbers. This is in agreement with the most common, *linear-time* dialect of temporal logic. In temporal constraint databases, however, the use of classical query languages like relational calculus or algebra accommodates a variety of time domains, including dense and continuous ones.

**Time domain and granularity.** Despite its apparent simplicity, the addition of the notion of time domain to temporal databases presents various subtleties. The main ones concern the nature of the elements of the domain. As soon as calendar times come into play, indeed, the abstract notion of instant-based time domain must be contextualized with respect to a specific *granularity* [3, 6]. Any given granularity can be viewed as a suitable abstraction of the real time line that partitions it into a denumerable sequence of homogeneous stretches of time. The elements of the partition, granules in the temporal database terminology, become the individuals (non-decomposable time units) of a discrete time domain. With respect to the considered granularity, these temporal individuals can be assimilated to time instants. Obviously, if a shift to a finer granularity takes place, e.g., if one moves from the domain of months to the domain of days, a single granule must be replaced with a set of granules. In such a way, being instantaneous is not more an intrinsic property of a temporal individual, but it depends on the time granularity one refers to. A detailed analysis of the limitations of the temporal database management of instant-based time domains can be found in [9].

**The association of time with data.** The association of the elements of the time domain with data is done by *timestamping*. A timestamp is a time value associated with a data object. In the relational setting, one distinguishes between attribute-timestamped data models, where timestamps are associated with attribute values, and tuple-timestamped data models, where timestamps are associated with tuples of values. As a third possibility, a timestamp can be associated with an entire relation/database.

Timestamps can be single elements as well as sets of elements of the time domain. Time instants are usually associated with relevant events, e.g., they can be used to record the day of the hiring or of the dismissal of an employee. (Convex) sets of time instants are associated with facts that hold over time. As an example, if a person E works for a company C from the 1st of February 2007 to the 31st of May 2007, one keeps track of the fact that every day in between the 1st of February 2007 and the 31st of May 2007, endpoints included, E is an employee of C.

Time intervals are needed to deal with situations where validity over an interval cannot be reduced to validity over its subintervals (including point subintervals) [10]. This is the case with processes that relate to an interval as a whole, meaning that if a process consumes a certain interval it cannot possibly transpire during any proper subinterval thereof. Examples are the processes of baking a cake or of flying from Venice to Montreal. This is also the case

when validity of a fact at/over consecutive instants/intervals does not imply its validity over the whole interval. As an example, two consecutive phone calls with the same values are different from a single phone call over the whole period. This is also the case for some kinds of temporal aggregation [4]. Finally, the use of time intervals is common in several areas of AI, including knowledge representation and qualitative reasoning, e.g., [1].

It is important to avoid any confusion between this latter use of intervals as timestamps and their use as compact representations of sets of time points (time periods in the temporal database literature). Time intervals are indeed often used to obtain succinct representations of (convex) sets of time instants. In such a case, validity over a time period is interpreted as validity at every time instant belonging to it. As an example, the fact that a person E worked for a company C from the 1st of February 2007 to the 31st of May 2007 can be represented by the tuple (E, C, [2007/02/01, 2007/05/31]) meaning that E worked for C every day in the closed interval [2007/02/01, 2007/05/31] .

## Key applications

As already pointed out, the time domain is an essential component of any temporal data model, and thus its addition to SQL standards does not come as a surprise.

In SQL, time domains are encoded via *temporal data types* (they have been introduced in SQL-92 and preserved in SQL:1999). In SQL-92, five (anchored) time instant data types, three basic forms and two variations, are supported (DATE, TIME, TIMESTAMP, TIME WITH TIME ZONE, TIMESTAMP WITH TIME ZONE). In addition, SQL-92 features two (unanchored) data types that allow one to model positive (a shift from an instant to a future one) and negative (a shift from an instant to a past one) distances between instants. One can be used to specify distances in terms of years and months (the YEAR-MONTH INTERVAL type), the other to specify distances in terms of days, hours, minutes, seconds, and fractions of a second (the DAY-TIME INTERVAL type). As a matter of fact, the choice of using the word interval to designate a time distance instead of a temporal individual – in contrast with the standard use of this word in computer science – is unfortunate, because it confuses a derived element of the time domain (the interval) with a property of it (its duration). An additional (unanchored) temporal data type, called PERIOD, was included in the SQL/Temporal proposal for the SQL3 standard, which was eventually withdrawn. A period is a convex sets of time instants that can be succinctly represented as a pair of time instants, namely, the first and the last instants with respect to the given order.

SQL also provides *predicates*, *constructors*, and *functions* for the management of time values. General predicates, such as the equal-to and less-than predicates, can be used to compare pairs of comparable values of any given temporal type; moreover, the specific overlap predicate can be used to check whether two time periods overlap. Temporal constructors are expressions that return a

temporal value of a suitable type. We may distinguish datetime constructors, that return a time instant of one of the given data types, and interval constructors, that return a value of YEAR-MONTH INTERVAL or DAY-TIME INTERVAL types. As for functions, they include the datetime value functions, such as the CURRENT_DATE function, that return an instant of the appropriate type, the CAST functions, that convert a value belonging to a given (temporal or non temporal) source data type into a value of the target temporal data type, and the extraction functions, that can be used to access specific fields of instant or interval time values.

## Future directions

Despite the strong prevalence of instant-based data models in current temporal databases, a number of interesting problems, such as, for instance, that of temporal aggregation, motivate a systematic study and development of *interval-based data models*. Moreover, in both instant-based and interval-based data models intervals are defined as suitable sets of elements of an instant-based time domain. The possibility of assuming time intervals as the primitive temporal constituents of the temporal domain is still largely unexplored. We believe that such an alternative deserves a serious investigation.

## Cross references

Temporal Data Model, Temporal Query Languages, Temporal Granularity, Temporal Indeterminacy, Temporal Periodicity, Point-Stamped Temporal Models, Period-Stamped Temporal Models, Temporal Constraints, Temporal Algebras, Now in Temporal Databases

## References


[1] James Allen and G. Ferguson, Actions and Events in Interval Temporal Logic, *Journal of Logic and Computation*, 4(5):531-579, 1994.

[2] Johan van Benthem, *The Logic of Time. A Model-Theoretic Investigation into the Varieties of Temporal Ontology and Temporal Discourse* (second edition), Kluwer Academic Publisher, 1991.

[3] Claudio Bettini and Sushil Jajodia and X. Sean Wang, *Time Granularities in Databases, Data Mining, and Temporal Reasoning*, Springer, 2000.

[4] Michael H. Böhlen, Johann Gamper, and Christian S. Jensen, *How Would You Like to Aggregate Your Temporal Data?* In Proceedings of the 13th International Symposium on Temporal Representation and Reasoning (TIME), IEEE Comp. Society, 2006, pp. 121-136.

[5] Jan Chomicki and David Toman, *Temporal Databases*. Chapter 14 of the Handbook of Temporal Reasoning in Artificial Intelligence, M. Fisher, D. Gabbay, and L. Vila (Eds.), Elsevier B. V., 2005, pp. 429-467.

[6] Jerome Euzenat and Angelo Montanari, *Time Granularity*, Chapter 3 of the Handbook of Temporal Reasoning in Artificial Intelligence, M. Fisher, D. Gabbay, and L. Vila (Eds.), Elsevier B. V., 2005, pp. 59-118.

[7] Valentin Goranko, Angelo Montanari, and Guido Sciavicco, A Road Map of Interval Temporal Logics and Duration Calculi, *Journal of Applied Non-Classical Logics*, 14(1-2):9-54, 2004.

[8] Angelo Montanari and Barbara Pernici, *Temporal Reasoning*. Chapter 21 of Temporal Databases: Theory, Design and Implementation, A. Tansell, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass (Eds.), Database Systems and Applications Series, Benjamin/Cummings Pub. Co., Redwood City, CA, 1993, pp.534-562.

[9] Richard T. Snodgrass, *Developing Time-Oriented Database Applications in SQL*. Chapter 3: Instants and Intervals, Morgan Kauffman Publishers, 2000, pp. 24-87.

[10] Paolo Terenziani and Richard T. Snodgrass, Reconciling Point-Based and Interval-Based Semantics in Temporal Databases: A Treatment of the Telic/Atelic Distinction, *IEEE Transactions on Knowledge and Data Engineering*, 16(5):540–551, 2004