# Proposed Temporal Database Concepts—May 1993*

Christian S. Jensen (editor) James Clifford Curtis Dyreson Shashi K. Gadia
Fabio Grandi Sushil Jajodia Nick Kline Angelo Montanari
Daniel Nonen Elisa Peressi Barbara Pernici John F. Roddick
Nandlal L. Sarda Maria Rita Scalas Arie Segev Richard T. Snodgrass
Mike D. Soo Abdullah Tansel Paolo Tiberio

## Abstract

*This document contains the complete set of glossary entries proposed by members of the temporal database community from Spring 1992 until May 1993. It is part of an initiative aimed at establishing an infrastructure for temporal databases. As such, the proposed concepts will be discussed during "International Workshop on an Infrastructure for Temporal Databases," in Arlington, TX, June 1993, with the specific purpose of defining a consensus glossary of temporal database concepts and names.*

*Earlier status documents appeared in March 1993 and December 1992 and included terms proposed after an initial glossary appeared in SIGMOD Record in September 1992. This document subsumes all the previous documents. Additional information related to the initiative may be found at* `cs.arizona.edu` *in the* `tsql` *directory, accessible via anonymous ftp.*

*Correspondence may be directed to the TSQL electronic mail distribution, `tsql@cs.arizona.edu`, or to the editor at Aalborg University, Datalogi, Fr. Bajers Vej 7E, DK–9220 Aalborg Ø, Denmark, `csj@iesd.auc.dk`. Affiliations and e-mail addresses of the authors follow. J. Clifford, Information Systems Dept., New York University, `jcliffor@is-4.stern.nyu.edu`; C. Dyreson, Computer Science Dept., University of Arizona `curtis@cs.arizona.edu`; S. K. Gadia, Computer Science Dept., Iowa State University, `gadia@cs.iastate.edu`; F. Grandi, University of Bologna, Italy, `fabio@deis64.cineca.it`; N. Kline, Computer Science Dept., University of Arizona, `kline@cs.arizona.edu`; A. Montanari, Dip. di Matematica e Informatica, Università di Udine, Italy, `montanari@uduniv.cineca.it`; D. Nonen, Computer Science Dept., Concordia University, Canada, `daniel@cs.concordia.ca`; E. Peressi, Dip. di Matematica e Informatica, Università di Udine, Italy, `peressi@udmi5400.cineca.it`; B. Pernici, Dip. di Matematica e Informatica, Università di Udine, Italy, `pernici@ipmel2.polimi.it`; J. F. Roddick, School of Computer and Information Science, University of South Australia `roddick@unisa.edu.au`; A. Segev, School of Business Adm. and Computer Science Research Dept., University of California, `segev@csr.lbl.gov`; N. L. Sarda, Computer Science and Eng. Dept., Indian Institute of Technology, Bombay, India, `nls@cse.iitb.ernet.in`; M. R. Scalas, University of Bologna, Italy, `rita@deis64.cineca.it`; R. T. Snodgrass, Computer Science Dept., University of Arizona, `rts@cs.arizona.edu`; A. Tansel, Bernard M. Baruch College, City University of New York `UZTBB@CUNYVM.CUNY.EDU`; P. Tiberio, University of Bologna, Italy, `tiberio@deis64.cineca.it`.

## 1 Introduction

A technical language is an important infra-structural component of any scientific community. To be effective, such a language should be well-defined, intuitive, and agreed-upon.

This document contains proposals for definitions and names of a range of concepts specific to temporal databases that are well-defined, well understood, and widely used. The proposal meets a need for creating a higher degree of consensus on the definition and naming of central concepts from within the field. The use of inconsistent terminology adversely affects the accessibility of the literature—to members of the community as well as others—and has an adverse effect on progress.

Being a proposal, simply stating definitions and names would be counter-productive and against the intentions. Consequently, the proposals in the document generally include alternatives and discussions of why specific decisions were made. When several alternative names for concepts were considered, the document not only states the decisions, but it also presents the alternatives and discusses why the decisions were made.

The history of this document may be described as follows. An initial glossary of temporal database concepts arose from e-mail discussions when appropriate terminology was considered for the book *Temporal Databases: Theory, Design, and Implementation,* edited by A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, Benjamin/Cummings Publishers. That glossary also appeared in the September 1992 issue of the ACM SIGMOD Record. The efforts continued, independently of the book, and the community was invited to submit proposals to the



This paper appeared in Proceedings of the International Workshop on an Infrastructure for Temporal Databases, Arlington, TX, June 14-16, 1993, pp. A1-A24.

mailing list tsql@cs.arizona.edu. As results, status documents appeared in December 1992 and in March 1993. All these previous documents are subsumed by the present document.

With the goal of obtaining a consensus glossary, the proposed concepts and names will be discussed during "International Workshop on an Infrastructure for Temporal Databases," in Arlington, TX, June 1993. The objective of this workshop is to define and establish a common infrastructure of temporal databases and to develop a consensus base document that will provide a foundation for implementation and standardization as well as for further research.

The document is organized as follows. The following section constitutes the main part of the paper. It contains proposals for 68 concepts. A small section of 15 proposals follows. These proposals were submitted relatively late, and the community has not yet had the opportunity to fully discuss them. Two appendices follow which outline relevance and evaluation criteria for glossary entries. These criteria are referenced throughout the document. Finally, an index is included on the last page.

# 2 Complete Listing of Proposed Glossary Entries

## 2.1 Valid Time

### Definition

The *valid time* of a fact is the time when the fact is true in the modeled reality. A fact may have associated any number of events and intervals, with single events and intervals being important special cases.

### Alternative Names

Real-world time, intrinsic time, logical time, data time.

### Discussion

Valid time is widely accepted already (+E3); it is short and easily spelled and pronounced (+E2). Most importantly, it is intuitive (+E8).

The name "real-world time" derives from the common identification of the modeled reality (opposed to the reality of the model) as the real world (+E8). This name has no apparent advantages to valid time, and it is less frequently used and longer (−E3, −E2).

"Intrinsic time" is the opposite of extrinsic time. Choosing intrinsic time for valid time would require us to choose extrinsic time for transaction time. The names are appropriate: The time when a fact is true is intrinsic to the fact; when it happened to be stored in a database is clearly an extrinsic property. Still, "intrinsic" is rarely used (−E3) and is longer and harder to spell than "valid" (−E2). As we shall see, transaction time is preferred over "extrinsic time" as well. Also, should a third concept of time be invented, there will be no obvious name for that concept (−E4).

"Logical time" has been used for valid time in conjunction with "physical time" for transaction time. As the discussion of intrinsic time had to include extrinsic time, discussing logical time requires us to also consider physical time. Both names are more rarely used than valid and transaction time (−E3), and they do not posses clear advantages over these.

The name "data time" is probably the most rarely used alternative (−E3). While it is clearly brief and easily spelled and pronounced, it is not intuitively clear that the data time of a fact refers to the valid time as defined above (+E2,−E8).

## 2.2 Transaction Time

### Definition

A database fact is stored in a database at some point in time, and after it is stored, it may be retrieved. The *transaction time* of a database fact is the time when the fact is stored in the database. Transaction times are consistent with the serialization order of the transactions. Transaction time values cannot be after the current time. Also, as it is impossible to change the past, transaction times cannot be changed. Transaction times may be implemented using transaction commit times.

### Alternative Names

Registration time, extrinsic time, physical time.

### Discussion

Transaction time has the advantage of being almost universally accepted (+E3), and it has no conflicts with valid time (+E1, +E4, +E7).

Registration time seems to be more straight forward. However, often a time of a particular type is denoted by $t_x$ where $x$ is the first letter of the type. As $r$ is commonly used for denoting a relation, adopting registration time creates a conflict (−E2).

Extrinsic time is rarely used (−E3) and has the same disadvantages as intrinsic time.

Finally, physical time is used infrequently (−E3) and seems vague (−E8).

## 2.3 User-defined Time

### Definition

*User-defined time* is an uninterpreted attribute domain of date and time. User-defined time is parallel to domains such as "money" and integer—unlike transaction time and valid time, it has no special query language support. It may be used for attributes such as "birth day" and "hiring date."

### Discussion

Conventional database management systems generally support a time and/or date attribute domain. The SQL2 standard has explicit support for user-defined time in its `datetime` and `interval` types.

## 2.4 Valid-Time Relation

### Definition

A *valid-time relation* is a relation with exactly one system supported valid time. In agreement with the definition of valid time, there are no restrictions on how valid times may be associated with the tuples (e.g., attribute value time stamping may be employed).

### Alternative Names

Historical relation.

### Discussion

While historical relation is used currently by most authors (+E3), two problems have been pointed out. First, the qualifyer "historical" is too generic (−E5). Second, "historical," being a reference to the past, is misleading because a valid-time relation may also contain facts valid in the future (−E8, −E9).

"Valid-time relation" is straight forward and avoids these problems. Also, it is consistent with "transaction time relation," to be discussed next (+E1).

## 2.5 Transaction-Time Relation

### Definition

A *transaction-time relation* is a relation with exactly one system supported transaction time. As for valid-time relations, there are no restrictions as to how transaction times may be associated with the tuples.

### Alternative Names

Rollback relation.

### Discussion

"Transaction-time relation" is already used by several authors, but other authors use the name "rollback relation." The motive for adopting transaction-time relation is identical for the motive for adopting valid-time relation. The motive for adopting rollback relation is that this type of relation supports a special rollback operation (+E7). But then, for reasons of parallelity, should not a valid-time relation be named for the special operation on valid-time relations corresponding to the rollback operation, namely transaction timeslice (−E4)?

## 2.6 Snapshot Relation

### Definition

Relations of a conventional relational database system incorporating neither valid-time nor transaction-time timestamps are *snapshot relations*.

### Alternative Names

Relation, conventional relation, static relation.

### Discussion

With several types of relations, simply using "relation" to denote one type is often inconvenient. The modifier "snapshot" is widely used (+E3). In addition, it is easy to use and seems precise and intuitive (+E2,9,8). The alternative "conventional" is longer and used more infrequently. Further, "conventional" is a moving target—as technologies evolve, it changes meaning. This makes it less precise. Finally, "static" is less frequently used than "snapshot," and it begs for the definition of the opposite concept of a dynamic relation, which will not be defined (−E3, −E1).

## 2.7 Bitemporal Relation

### Definition

A *bitemporal relation* is a relation with exactly one system supported valid time and exactly one system-supported transaction time.

### Alternative Names

Temporal relation, fully temporal relation, valid-time and transaction-time relation, valid-time transaction-time relation.

### Discussion

We first discuss the concept; then we discuss the name.

In the adopted definition, "bi" refers to the existence of exactly two times. An alternative definition states that a bitemporal relation has one or more system supported valid times and one or more system supported transaction times. In this definition, "bi" refers to the existence of exactly two types of times.

Most relations involving both valid and transaction time are bitemporal according to both definitions. Being the most restrictive, the adopted definition is the most desirable: It is the tightest fit, giving the most precise characterization (+E9).

The definition of bitemporal is used as the basis for applying bitemporal as a modifier to other concepts such as "query language." This adds more important reasons for preferring the adopted definition.

Independently of the precise definition of bitemporal, a query language is bitemporal if and only if it supports any bitemporal relation (+E1), see Section 2.8. With the adopted definition, most query languages involving both valid and transaction time may be characterized as bitemporal. With the alternative definition, query languages that are bitemporal under the adopted definition are no longer bitemporal. This is a serious drawback of the alternative definition. It excludes the possibility of naming languages that may be precisely named using the adopted definition. With the alternative definition, those query languages have no (precise) name. What we get is a concept and name (bitemporal query language) for which there is currently little or no use.

Also, note that a query language that is bitemporal with the alternative definition is also bitemporal with regard to the adopted definition (but the adopted definition does not provide a precise characterization of this query language). Thus, the restrictive definition of a bitemporal relation results in a non-restrictive definition of bitemporal query language (and vice-versa).

The name "temporal relation" is commonly used. However, it is also used in a generic and less strict sense, simply meaning any relation with some time aspect. It will not be possible to change the generic use of the term (−E7), and since using it with two meanings causes ambiguity (−E9), it is rejected as a name for bitemporal relations. In this respect "temporal relation" is similar to "historical relation."

Next, the term "fully temporal relation" was proposed because a bitemporal relation is capable of modeling both the intrinsic and the extrinsic time aspects of facts, thus providing the "full story." However, caution dictates that we avoid names that are absolute (−E6). What are we going to name a relation more general than a temporal relation?

The name "valid-time and transaction-time relation" is precise and consistent with the other names, but it is too cumbersome to be practical (−E2). Also, it may cause ambiguity. For example, the sentence "the topic of this paper is valid-time and transaction-time relations" is ambiguous.

We choose to name relations as opposed to databases because a database may contain several types of relations. Thus, naming relations is a more general approach.

## 2.8 Snapshot, Valid- and Transaction-Time, and Bitemporal as Modifiers

The definitions of how "snapshot," "valid-time," "transaction-time," and "bitemporal" apply to relations provide the basis for applying these modifiers to a range of other concepts. Let $x$ be one of snapshot, valid-time, transaction-time, and bitemporal. Twenty derived concepts are defined as follows (+E1).

**relational database** An $x$ relational database contains one or more $x$ relations.

**relational algebra** An $x$ relational algebra has relations of type $x$ as basic objects.

**relational query language** An $x$ relational query language manipulates any possible $x$ relation. Had we used "some" instead of "any" in this definition, the defined concept would be very imprecise (−E9).

**data model** An $x$ data model has an $x$ query language and supports the specification of constraints on any $x$ relation.

**DBMS** An $x$ DBMS supports an $x$ data model.

The two model-independent terms, data model and DBMS, may be replaced by more specific terms. For example, "data model" may be replaced by "relational data model" in "bitemporal data model."

The nouns that have been modified above are not specific to temporal databases. The nouns chronon and event are specific to temporal databases and may be modified by "valid-time," "transaction-time," and "bitemporal."

## 2.9 Temporal as Modifier

### Definition

The modifier *temporal* is used to indicate that the modified concept concerns some aspect of time.

### Alternative Names

Time-oriented.

**Discussion**

"Temporal" is already being used in the sense defined here. In addition, some researchers have used in a more specific sense (i.e., supports both transaction time and valid time). This practice was awkward: Using "temporal" with the general definition in the beginning of a paper and then adopting the more specific meaning later in the paper created confusion. It also lead to the use of "time-oriented" instead of temporal in the generic sense.

Realizing that the use of the generic meaning of "temporal" cannot be changed prompted the adoption of "bitemporal' for the specific meaning.

Being only the name of a generic concept, "temporal" may now be used instead of the more cumbersome "time-oriented." It may be applied generically as a modifier for "database," "algebra," "query language," "data model," and "DBMS."

## 2.10 Temporal Database

**Definition**

A *temporal* database supports some aspect of time, not counting user-defined time.

**Alternative Names**

Time-oriented database, historical database.

**Discussion**

The concept of a temporal database is defined separately due to its importance. The discussion in Section 2.9 applies here.

## 2.11 Transaction Timeslice Operator

**Definition**

The *transaction timeslice operator* may be applied to any relation with a transaction time. It also takes as argument a time value not exceeding the current time, *NOW*. It returns the state of the argument relation that was current at the time specified by the time argument.

**Alternative Names**

Rollback operator, timeslice operator, state query.

**Discussion**

The name "rollback operator" has procedural connotations, which in itself is inappropriate (−E8). Why not use "rollforward operator?" The choice between one of them is rather arbitrary. Further, the transaction timeslice operator may be computed using both rollback (decremental computation) and rollforward (incremental computation).

"State query" seems less precise than transaction timeslice operator (−E9). It is equally applicable as a name for the valid timeslice operator (−E8). Further, "state operator" is better than "state query."

The name "transaction timeslice" may be abbreviated to timeslice when the meaning is clear from the context.

## 2.12 Valid Timeslice Operator

**Definition**

The *valid timeslice operator* may be applied to any relation with a valid time. It takes as argument a time value. It returns the state of the argument relation that was valid at the time of the time argument.

**Alternative Names**

Timeslice operator.

**Discussion**

"Valid timeslice operator" is consistent with transaction timeslice operator (+E1). "Timeslice" is appropriate only in a disambiguating context (+E2).

## 2.13 Temporal Element

**Definition**

A *temporal element* is a finite union of $n$-dimensional intervals. Special cases of temporal elements include *valid-time elements*, *transaction time elements*, and *bitemporal elements*. They are finite unions of valid-time intervals, transaction-time intervals, and bitemporal intervals, respectively.

**Alternative Names**

Temporal element.

**Discussion**

Observe that temporal elements are closed under the set theoretic operations of union, intersection and complementation. Temporal elements are often used as timestamps. A temporal element may be represented by a set of chronons.

A valid time element was previously termed a temporal element. However, for the naming to be consistent with the remainder of the glossary, "temporal" is reserved as a generic modifier, and more specific modifiers are adopted.

## 2.14 Chronon

### Definition

In one dimension, a *chronon* is the shortest duration of time supported by a temporal DBMS, i.e., a nondecomposable unit of time. A particular one-dimensional chronon is a subinterval of fixed duration on a time-line. An $n$-dimensional chronon is a non-decomposable region in $n$-dimensional time. Important special types of chronons include valid-time, transaction-time, and bitemporal chronons.

### Alternative Names

Instant, moment, time quantum, time unit.

### Discussion

Various models of time have been proposed in the philosophical and logical literature of time (e.g., van Benthem). These view time, among other things, as discrete, dense, or continuous. Intuitively, discrete models of time are isomorphic to the natural numbers, i.e., there is the notion that every moment of time has a unique successor. Dense models of time are isomorphic to (either) the real or rational numbers: between any two moments of time there is always another. Continuous models of time are isomorphic to the real numbers, i.e., both dense and also, unlike the rational numbers, with no "gaps."

"Instant" and "moment" invite confusion between a *point* in the continuous model and a nondecomposable *unit* in the discrete model (−E8). Clocking instruments invariably report the occurrence of events in terms of time intervals, not time "points." Hence, events, even so-called "instantaneous" events, can best be measured as having occurred during an interval (−E9). "Time quantum" is precise, but is longer and more technical than "chronon" (−E2). "Time unit" is perhaps less precise (−E9).

## 2.15 Timestamp

### Definition

A *timestamp* is a time value associated with some time-stamped object, e.g., an attribute value or a tuple. The concept may be specialized to valid timestamp, transaction timestamp, interval timestamp, event timestamp, bitemporal element timestamp, etc.

## 2.16 Lifespan

### Definition

The *lifespan* of a database object is the time over which it is defined. The valid-time lifespan of a database object refers to the time when the corresponding object exists in the modeled reality, whereas the transaction-time lifespan refers to the time when the database object is current in the database.

If the object (attribute, tuple, relation) has an associated timestamp then the lifespan of that object is the value of the timestamp. If components of an object are timestamped, then the lifespan of the object is determined by the particular data model being employed.

### Alternative Names

Timestamp, temporal element, temporal domain.

### Discussion

Lifespan is widely accepted already (+E3); it is short and easily spelled and pronounced (+E2). Most importantly, it is intuitive (+E8).

## 2.17 Temporally Homogeneous

### Definition

A temporal tuple is *temporally homogeneous* if the lifespan of all attribute values within it are identical. A temporal relation is said to be temporally homogeneous if its tuples are temporally homogeneous. A temporal database is said to be temporally homogeneous if all its relations are temporally homogeneous. In addition to being specific to a type of object (tuple, relation, database), homogeneity is also specific to some time dimension, as in "temporally homogeneous in the valid-time dimension" or "temporally homogeneous in the transaction-time dimension."

### Alternative Names

Homogeneous.

### Discussion

The motivation for homogeneity arises from the fact that no timeslices of a homogeneous relation produce null values. Therefore a homogeneous relational model is the temporal counterpart of the snapshot relational model without nulls. Certain data models assume temporal homogeneity. Models that employ tuple timestamping rather than attribute value timestamping are necessarily temporally homogeneous—only temporally homogeneous relations are possible.

In general, using simply "homogeneous" without "temporal" as qualifier may cause ambiguity because the unrelated notion of homogeneity exists also in distributed databases (−E5).

## 2.18 Event

### Definition

An *event* is an isolated instant in time. An event is said to occur at time $t$ if it occurs at any time during the chronon represented by $t$.

### Alternative Names

Instant, moment.

### Discussion

Both "instant" and "moment" may be confused with the distinct term "chronon" (−E5, −E7).

## 2.19 Interval

### Definition

An *interval* is the time between two events. It may be represented by a set of contiguous chronons.

### Alternative Names

Time period.

### Discussion

The name "interval" is widely accepted (+E3). The name "period" often implies a cyclic or recurrent phenomenon (−E8, −E9). In addition, "time period" is longer (−E2).

## 2.20 Span

### Definition

A *span* is a directed duration of time. A duration is an amount of time with known length, but no specific starting or ending chronons. For example, the duration "one week" is known to have a length of seven days, but can refer to any block of seven consecutive days. A span is either positive, denoting forward motion of time, or negative, denoting backwards motion in time.

### Alternative Names

Duration, interval, time distance.

### Discussion

It is already accepted that "interval" denotes an anchored span (−E7). A "duration" is generally considered to be non-directional, i.e., always positive (−E7). The term "time distance" is precise, but is longer (−E2).

## 2.21 Temporal Expression

### Definition

A *temporal expression* is a syntactic construct used in a query that evaluates to a temporal value, i.e., an event, an interval, a span, or a temporal element.

In snapshot databases, expressions evaluate to relations and therefore they may be called relational expressions to differentiate them from temporal expressions.

### Discussion

All approaches to temporal databases allow relational expressions. Some only allow relational expressions, and thus they are unisorted. Some allow relational expressions, temporal expressions and also possibly boolean expressions. Such expressions may defined through mutual recursion.

## 2.22 Time-invariant Attribute

### Definition

A *time-invariant attribute* is an attribute whose value is constrained to not change over time. In functional terms, it is a constant-valued function over time.

## 2.23 Time-varying Attribute

### Definition

A *time-varying attribute* is an attribute whose value is not constrained to be constant over time. In other words, it may or may not change over time.

## 2.24 Temporal Data Type

### Definition

The user-defined temporal data type is a time representation specially designed to meet the specific needs of the user. For example, the designers of a database used for class scheduling in a school might be based on a "Year:Term:Day:Period" format. Terms belonging to a user-defined temporal data type get the same query language support as do terms belonging to built-in temporal data types such as the DATE data type.

### Alternative Names

User-defined temporal data type, auxiliary temporal data type.

### Discussion

The phrase "user-defined temporal data type" is uncomfortably similar to the phrase "user-defined time", which is an orthogonal concept. Nevertheless, it is an appropriate description for the intended usage and we have used in our work. If the notion of providing special purpose temporal terms becomes more popular, I suspect the shorter term "Temporal Data Type" will be sufficiently descriptive.

## 2.25 Schema Evolution

### Definition

A database system supports *schema evolution* if it permits modification of the database schema without the loss of extant data. No historical support for previous schemas is required.

### Alternative Names

Schema versioning, data evolution.

### Discussion

While support for "schema evolution" indicates that an evolving schema may be supported, the term "schema versioning" indicates that previous versions of an evolving schema are also supported. Therefore, "schema versioning" is appropriate for a more restrictive concept.

The name "data evolution" is inappropriate because "data" refers to the schema contents, i.e., the extension rather than the intension. Data evolution is supported by conventional update operators.

While some confusion exists as to its exact definition, "schema evolution" is an accepted name and is widely used already.

## 2.26 Schema Versioning

### Definition

A database system accommodates *schema versioning* if it allows the querying of all data, both retrospectively and prospectively, through user-definable version interfaces. While support for schema versioning implies the support for schema evolution, the reverse is not true.

Support for schema versioning requires that a history of changes be maintained to enable the retention of past schema definitions.

### Alternative Names

Schema evolution, data evolution.

### Discussion

The name "schema evolution" does not indicate that previously current versions of the evolving schema are also supported. It is thus less precise that "schema versioning." As schema evolution, schema versioning is an intensional concept; "data evolution" has extensional connotations and is inappropriate.

## 2.27 Snapshot Equivalent

### Definition

Informally, two tuples are *snapshot equivalent* if the snapshots of the tuples at all times are identical.

Let temporal relation schema $R$ have $n$ time dimensions, $D_i$, $i = 1, \ldots, n$, and let $\tau^i$, $i = 1, \ldots, n$ be corresponding timeslice operators, e.g., the valid timeslice and transaction timeslice operators. Then, formally, tuples $x$ and $y$ are snapshot equivalent if

$$\forall t_1 \in D_1 \ldots \forall t_n \in D_n ( \\ \tau^n_{t_n}(\ldots(\tau^1_{t_1}(x))\ldots) = \tau^n_{t_n}(\ldots(\tau^1_{t_1}(y))\ldots)) \; .$$

Similarly, two relations are *snapshot equivalent* if at every time their snapshots are equal. *Snapshot equivalence* is a binary relation that can be applied to tuples and to relations.

### Alternative Names

Weakly equal, temporally weakly equal, weak equivalence.

### Discussion

Weak equivalence has been used by Ullman to relate two algebraic expressions (Ullman, Principles of Database Systems, Second Edition, page 309). Hence, "temporally weakly equal" is preferable to "weakly equal" (+E7).

In comparing "temporally weakly equal" with "snapshot equivalent," the former term is longer and more wordy, and is somewhat awkward, in that it contains two adverbs (−E2). "Temporally weak" is not intuitive—in what way is it weak? Snapshot equivalent explicitly identifies the source of the equivalence (+E8).

## 2.28 Snapshot-Equivalence Preserving Operator

### Definition

A unary operator $F$ is *snapshot-equivalence preserving* if relation $r$ is snapshot equivalent to $r'$ implies $F(r)$ is snapshot equivalent to $F(r')$. This definition may be extended to operators that accept two or more argument relation instances.

### Alternative Names

Weakly invariant operator, is invariant under weak binding of belongs to.

### Discussion

This definition does not rely on the term "weak binding" (+E7).

## 2.29 Snapshot Equivalence Class

### Definition

A *snapshot equivalence class* is a set of relation instances that are all snapshot equivalent to each other.

### Alternative Names

Weak relation.

### Discussion

"Weak relation" is not intuitive, as the concept identifies a set of relation instances, not a single instance (−E8).

## 2.30 Value Equivalence

### Definition

Informally, two tuples on the same (temporal) relation schema are *value equivalent* if they have identical non-timestamp attribute values.

To formally define the concept, let temporal relation schema $R$ have $n$ time dimensions, $D_i$, $i = 1, \ldots, n$, and let $\tau^i$, $i = 1, \ldots, n$ be corresponding timeslice operators, e.g., the valid timeslice and transaction timeslice operators. Then tuples $x$ and $y$ are value equivalent if

$$
\begin{array}{c}
\exists t_1 \in D_1 \ldots \exists t_n \in D_n (\tau^n_{t_n}(\ldots(\tau^1_{t_1}(x))\ldots) \neq \emptyset) \wedge \\
\exists s_1 \in D_1 \ldots \exists s_n \in D_n (\tau^n_{s_n}(\ldots(\tau^1_{s_1}(y))\ldots) \neq \emptyset) \\
\Rightarrow \\
\bigcup_{\forall t_1 \in D_1 \ldots \forall t_n \in D_n} \tau^n_{t_n}(\ldots(\tau^1_{t_1}(x))\ldots) = \\
\bigcup_{\forall s_1 \in D_1 \ldots \forall s_n \in D_n} \tau^n_{s_n}(\ldots(\tau^1_{s_1}(y))\ldots) \ .
\end{array}
$$

Thus the set of tuples in snapshots of $x$ and the set of tuples in snapshots of $y$ are required to be identical. This is required only when each tuple has some non-empty snapshot.

### Discussion

The concept of value equivalent tuples has been shaped to be convenient when addressing concepts such as coalescing, normal forms, etc. The concept is distinct from related notions of the normal form SG1NF and *mergeable* tuples.

Phrases such as "having the same visible attribute values" and "having duplicate values" have been used previously.

The orthogonality criterion (+E1) is satisfied. Further, the concept is a straight-forward generalization of identity of tuples in the snapshot-relational model. There are no competing names (+E3), the name seems open-ended (+E4) and does not appear to have other meanings (+E5). Further, the name is consistent with existing terminology (+E7) and does not violate other criteria.

## 2.31 Fixed Span

### Definition

The duration of a span is either context-dependent or context-independent. A *fixed span* has a context-independent duration. For example, the span `one hour` has a duration of 60 minutes and is therefore a fixed span.

### Alternative Names

Constant span.

### Discussion

Fixed span is short (+E2), precise (+E9), and has no conflicting meanings (+E5).

"Constant" appears more precise (+E8) and intuitive (+E9), but it is also used as a keyword in several programming languages (−E5).

## 2.32 Variable Span

### Definition

A span that is not fixed is *variable*—the value of the span is dependent on the context in which it appears. For example, the span `one month` represents a duration of between twenty-eight and thirty-one days depending on the context in which it is used.

### Alternative Names

Moving span.

### Discussion

Variable span is intuitive (+E9), and precise (+E9).

"Moving span" is unintuitive (−E9) and has informal spatial connotations (−E5).

## 2.33 Physical Clock

### Definition

A *physical clock* is a physical process coupled with a method of measuring that process. Although the underlying physical process is continuous, the physical clock measurements are discrete, hence a physical clock is discrete.

### Alternative Names

Clock.

### Discussion

A physical clock by itself does not measure time; it only measures the process. For instance, the rotation of the earth measured in solar days is a physical clock. Most physical clocks are based on cyclic physical processes (such as the rotation of the earth). The modifier "physical" is used to distinguish this kind of clock from other kinds of clocks, e.g., the time-line clock (+E9). It is also descriptive in so far as physical clocks are based on recurring natural or man-made phenomena (+E8).

## 2.34 Time-line Clock

### Definition

In the discrete model of time, a *time-line clock* is a set of physical clocks coupled with some specification of when each physical clock is authoritative. Each chronon in a time-line clock is a chronon (or a regular division of a chronon) in an identified, underlying physical clock. The time-line clock switches from one physical clock to the next at a synchronization point. A synchronization point correlates two, distinct physical clock measurements. The time-line clock must be anchored at some chronon to a unique physical state of the universe.

### Alternative Names

Base-line clock, time-segment clock.

### Discussion

A time-line clock glues together a sequence of physical clocks to provide a consistent, clear semantics for a discrete time-line. A time-line clock provides a clear, consistent semantics for a discrete time-line by gluing together a sequence of physical clocks. Since the range of most physical clocks is limited, a time-line clock is usually composed of many physical clocks. For instance, a tree-ring clock can only be used to date past events, and the atomic clock can only be used to date events since the 1950s. The term "time-line" has a well-understood informal meaning, as does "clock," which we coopt for this definition (+E5). This concept currently has no name (+E7)(−E3), but it is used for every timestamp (e.g., SQL2 uses the mean solar day clock—the basis of the Gregorian calendar—as its time-line clock). The modifier "time-line" distinguishes this clock from other kinds of clocks (+E1). Time-line is more intuitive than "base-line" (+E8), but less precise (mathematically) than "time-segment," since the time-line clock usually describes a segment rather than a line (−E9). We prefer time-line clock to time-segment clock because the former term is more general (+E4) and is intuitively appealing.

## 2.35 Time-line Clock Granularity

### Definition

The *time-line clock granularity* is the uniform size of each chronon in the time-line clock.

### Discussion

The modifier "time-line" distinguishes this kind of granularity from other kinds of granularity (+E1) and describes precisely where this granularity applies (+E9).

## 2.36 Beginning

### Definition

The time-line supported by any temporal DBMS is, by necessity, finite and therefore has a smallest and largest representable chronon. The distinguished value *beginning* is a special valid-time event preceding the smallest chronon on the valid-time line. Beginning has no transaction-time semantics.

### Alternative Names

Start, begin, commencement, origin, negative infinity.

### Discussion

Beginning has the advantage of being intuitive (+E8), and does not have conflicting meanings (+E5).

"Begin" appears to be more straight-forward (+E8) but suffers from conflicting meanings since it is a common programming language keyword (−E5).

"Start," "commencement," and "origin" are awkward to use, e.g., "Start precedes the event," "Commencement precedes the event," and "Origin precedes the event." (−E8). Furthermore, choosing start would require us to choose "end" for the opposite concept, and end is a common programming language keyword (−E5). Origin also has a conflicting meaning relative to calendars (−E5).

Lastly, "negative infinity" is longer (−E2) and slightly misleading since it implies that time is infinite (−E9). This may or may not be true depending on theories about the creation of the universe. Also, negative infinity has a well-established mathematical meaning (−E5).

## 2.37 Forever

### Definition

The distinguished value *forever* is a special valid-time event following the largest chronon on the valid-time line. Forever has no transaction-time semantics.

### Alternative Names

Infinity, positive infinity.

**Discussion**

Forever has the advantage of being intuitive (+E8) and does not have conflicting meanings (+E5).

"Infinity" and "positive infinity" both appear to be more straightforward but have conflicting mathematical meanings (−E5). Furthermore, positive infinity is longer and would require us to choose "negative infinity" for its opposite (−E2).

## 2.38 Initiation

**Definition**

The distinguished value *initiation* denotes the transaction time when the database was created, i.e., the chronon during which the first update to the database occurred. Initiation has no valid-time semantics.

**Alternative Names**

Start, begin, commencement, origin, negative infinity, beginning.

**Discussion**

The arguments against "start," "begin," "commencement," "origin," and "negative infinity" are as in the discussion of beginning.

Initiation is preferred over beginning since transaction time is distinct from valid time. Using different terms for the two concepts avoids conflicting meanings (+E5).

## 2.39 Timestamp Interpretation

**Definition**

In the discrete model of time, the *timestamp interpretation* gives the meaning of each timestamp bit pattern in terms of some time-line clock chronon (or group of chronons), that is, the time to which each bit pattern corresponds. The timestamp interpretation is a many-to-one function from time-line clock chronons to timestamp bit patterns.

**Discussion**

Timestamp interpretation is a concise (+E2), intuitive (+E8), precise (+E9) term for a widely-used but currently undefined concept (+E7).

## 2.40 Timestamp Granularity

**Definition**

In the discrete model of time, the *timestamp granularity* is the size of each chronon in a timestamp interpretation. For instance, if the timestamp granularity is one second, then the size of each chronon in the timestamp interpretation is one second (and vice-versa).

**Alternative Names**

Time granularity.

**Discussion**

Timestamp granularity is not an issue in the continuous model of time. The adjective "timestamp" is used to distinguish this kind of granularity from other kinds of granularity, such as the granularity of non-timestamp attributes (+E9,+E1). "Time granularity" is much too vague a term since there is a different granularity associated with temporal constants, timestamps, physical clocks, and the time-line clock although all these concepts are time-related. Each time dimension has a separate timestamp granularity. A time, stored in a database, must be stored in the timestamp granularity regardless of the granularity of that time (e.g., the valid-time date January 1st, 1990 stored in a database with a valid-time timestamp granularity of a second must be stored as a particular second during that day, perhaps midnight January 1st, 1990). If the context is clear, the modifier "timestamp" may be omitted, for example, "valid-time timestamp granularity" is equivalent to "valid-time granularity" (+E2).

## 2.41 Temporal Specialization

**Definition**

*Temporal specialization* denotes the restriction of the interrelationship between otherwise independent (implicit or explicit) timestamps in relations. An example is a relation where facts are always inserted after they were valid in reality. In such a relation, the transaction time would always be after the valid time. Temporal specialization may be applied to relation schemas, relation instances, and individual tuples.

**Alternative Names**

Temporal restriction.

**Discussion**

Data models exist where relations are required to be specialized, and temporal specializations often constitute important semantics about temporal relations that may be utilized for, e.g., query optimization and processing purposes.

The chosen name is more widely used than the alternative name (+E3). The chosen name is new (+E5) and indicates that specialization is done with respect to the temporal aspects of facts (+E8). Temporal specialization seems to be open-ended (+E4). Thus, an opposite concept, temporal generalization, has been defined. "Temporal restriction" has no obvious opposite name (−E4).

## 2.42 Specialized Bitemporal Relationship

**Definition**

A temporal relation schema exhibits a *specialized bitemporal relationship* if all instances obey some given specialized relationship between the (implicit or explicit) valid and transaction times of the stored facts. Individual instances and tuples may also exhibit specialized bitemporal relationships. As the transaction times of tuples depend on when relations are updated, updates may also be characterized by specialized bitemporal relationships.

**Alternative Names**

Restricted bitemporal relationship.

**Discussion**

The primary reason for the choice of name is consistency with the naming of temporal specialization (+E1). For additional discussions, see temporal specialization.

## 2.43 Retroactive Temporal Relation

**Definition**

A temporal relation schema including at least valid time is *retroactive* if each stored fact of any instance is always valid in the past. The concept may be applied to temporal relation instances, individual tuples, and to updates.

**Discussion**

The name is motivated by the observation that a retroactive bitemporal relation contains only information concerning the past (+E8).

## 2.44 Predictive Temporal Relation

**Definition**

A temporal relation schema including at least valid time is *predictive* if each fact of any relation instance is valid in the future when it is being stored in the relation. The concept may be applied to temporal relation instances, individual tuples, and to updates.

**Alternative Names**

Proactive bitemporal relation.

**Discussion**

Note that the concept is applicable only to relations which support valid time, as facts valid in the future cannot be stored otherwise.

The choice of "predictive" over "proactive" is due to the more frequent every-day use of "predictive," making it a more intuitive name (+E8). In fact, "proactive" is absent from many dictionaries. Tuples inserted into a predictive bitemporal relation instance are, in effect, predictions about the future of the modeled reality. Still, "proactive" is orthogonal to "retroactive" (−E1).

## 2.45 Degenerate Bitemporal Relation

**Definition**

A bitemporal relation schema is *degenerate* if updates to it's relation instances are made immediately when something changes in reality, with the result that the values of the valid and transaction times are identical. The concept may be applied to bitemporal relation instances, individual tuples, and to updates.

**Discussion**

"Degenerate bitemporal relation" names a previously unnamed concept that is frequently used. A degenerate bitemporal relation resembles a transaction-time relation in that only one timestamp is necessary. Unlike a transaction-time relation, however, it is possible to pose both valid-time and transaction-time queries on a degenerate bitemporal relation.

The use of "degenerate" is intended to reflect that the two time dimensions may be represented as one, with the resulting limited capabilities.

## 2.46 Valid-time Interval

**Definition**

A *valid-time interval* is an interval along the valid time-line. It identifies when some fact was true in reality.

**Discussion**

A valid-time interval can be represented with a contiguous, non-empty set of valid-time chronons.

## 2.47 Transaction-time Interval

**Definition**

A *transaction-time interval* is an interval along the transaction time-line. It identifies when a fact was logically in the database, from the time it was inserted until the time it was logically deleted.

**Discussion**
A transaction-time interval can be represented with a non-empty set of contiguous transaction-time chronons.

## 2.48 Bitemporal Interval

**Definition**
A *bitemporal interval* is a region in two-space of valid time and transaction time, with sides parallel to the axes. It identifies when a fact, recording that something was true in reality during the specified interval of valid time, was logically in the database during the specified interval of transaction time.

**Discussion**
A bitemporal interval can be represented with a non-empty set of bitemporal chronons.

## 2.49 Spatiotemporal as Modifier

**Definition**
The modifier *spatiotemporal* is used to indicate that the modified concept concerns simultaneous support of some aspect of time and some aspect of space, in one or more dimensions.

**Alternative Names**
Spatio-temporal, temporal-spatial, space-time-oriented.

**Discussion**
This term is already in use, interchangeably with "spatio-temporal," in the geographic information systems community (+E3) (hence, the preference over "temporal-spatial"), and is consistent with the "temporal" modifier (+E7). Avoiding the hyphen makes it easier to type (+E2), another reason to prefer it over "temporal-spatial". It may be applied generically as a modifier for "database," "algebra," "query language," "data model," and "DBMS.'

## 2.50 Spatial Quantum

**Definition**
A *spatial quantum* (or simply *quantum*, when the sense is clear) is the shortest distance (or area or volume) of space supported by a spatial DBMS—it is a nondecomposable region of space. It can be associated with one or more dimensions. A particular unidimensional quantum is an interval of fixed length along a single spatial dimension. A particular three-dimensional quantum is a fixed-sized, located cubic volume of space.

**Alternative Name**
Spatial unit.

**Discussion**
"Spatial quantum" is preferred over "spatial unit" because spatial distances and volumes are usually given as measurements of some unit (such as meters), but the "unit of measurement" is not the same as the "spatial quantum." The former term ("spatial quantum") is more precise (+E9), in part, because it avoids this possible confusion.

## 2.51 Spatiotemporal Quantum

**Definition**
A *spatiotemporal quantum* (or simply *quantum*, when the sense is clear) is a non-decomposable region in two, three, or four-space, where one or more of the dimensions are spatial and the rest, at least one, are temporal.

**Alternative Name**
Spatiotemporal unit, spatiotemporal chronon.

**Discussion**
This term is a generalization of chronon and spatial quantum. "Unit" is perhaps less precise (−E9). "Chronon" specifically relates to time, and thus is inconsistent with the adjective "spatiotemporal."

## 2.52 Spatiotemporal Interval

A *spatiotemporal interval* is a region in $n$-space, where at least one of the axes is a spatial dimension and the remaining axes are temporal dimensions, with the region having sides that are parallel to all axes. It identifies when and where a fact was true.

**Discussion**
A spatiotemporal interval can be represented by a non-empty set of spatiotemporal quanta.

## 2.53 Spatiotemporal Element

**Definition**
A *spatiotemporal element* is a finite set of spatiotemporal intervals. Spatiotemporal elements are closed under the set theoretic operations of union, intersection and complementation.

**Discussion**
This is the natural generalization of "temporal element." It can be represented with a set of spatiotemporal quanta.

## 2.54 Temporal Selection

### Definition

Facts are extracted from a temporal database by means of *temporal selection* when the selection predicate involves the times associated with the facts.

The generic concept of temporal selection may be specialized to include *valid-time selection, transaction-time selection,* and *bitemporal selection*. For example, in valid-time selection, facts are selected based on the values of their associated valid times.

### Discussion

Query languages supporting, e.g., valid-time data, generally provide special facilities for valid-time selection which are built into the languages.

The name has already been used extensively in the literature by a wide range of authors (+E3), it is consistent with the unmodified notion of selection in (non-temporal) databases (+E1, +E7), and it appears intuitive and precise (+E8, +E9).

## 2.55 Temporal Projection

### Definition

In a query or update statement, *temporal projection* pairs the computed facts with their associated times, usually derived from the associated times of the underlying facts.

The generic notion of temporal projection may be applied to various specific time dimensions. For example, *valid-time projection* associates with derived facts the times at which they are valid, usually based on the valid times of the underlying facts.

### Alternative Names

Temporal assignment.

### Discussion

While almost all temporal query languages support temporal projection, the flexibility of that support varies greatly.

In some languages, temporal projection is implicit and is based the intersection of the times of the underlying facts. Other languages have special constructs to specify temporal projection.

The name has already been used extensively in the literature (+E3). It derives from the `retrieve` clause in Quel as well as the `SELECT` clause in SQL, which both serve the purpose of the relational algebra operator projection, in addition to allowing the specification of derived attribute values.

A related concept, denoted a temporal assignment, is roughly speaking a function that maps a set of time values to a set of values of an attribute. One purpose of a temporal assignment would be to indicate when different values of the attribute are valid.

## 2.56 Temporal Dependency

### Definition

Let $X$ and $Y$ be sets of explicit attributes of a temporal relation schema, $R$. A *temporal functional dependency*, denoted $X \stackrel{\mathrm{T}}{\rightarrow} Y$, exists on $R$ if, for all instances $r$ of $R$, all snapshots of $r$ satisfy the functional dependency $X \rightarrow Y$.

Note that more specific notions of temporal functional dependency exist for valid-time, transaction-time, bitemporal, and spatiotemporal relations. Also observe that using the template for temporal functional dependencies, temporal multivalued dependencies may be defined in a straight-forward manner.

Finally, the notions of temporal keys (super, candidate, primary) follow from the notion of temporal functional dependency.

### Alternative Names

Independence, dependence.

### Discussion

Temporal functional dependencies are generalizations of conventional functional dependencies. In the definition of a temporal functional dependency, a temporal relation is perceived as a collection of snapshot relations. Each such snapshot of any extension must satisfy the corresponding functional dependency.

Other (conflicting) notions of of temporal dependencies and keys have been defined, but none are as closely paralleled by snapshot dependencies and keys as the above. The naming of the concepts is orthogonal with respect to existing snapshot concepts, and the new names are mutually consistent (+E1, +E7).

Related notions of independent and dependent attributes exist. Using temporal as a prefix distinguishes the concept from conventional dependencies and points to the specific nature of the dependency. Thus ambiguity is avoided (+E5), and precision is enhanced (+E9)—at the expense of brevity (−E2).

"Temporal dependency" has also been used in a non-generic sense, to denote a different concept. The term "temporal" is often used in a generic sense, so ambiguity results when it is also used in a specific sense. Thus "temporal" is used here only in a generic sense.

## 2.57 Temporal Normal Form

### Definition

A pair $(R, F)$ of a temporal relation schema $R$ and a set of associated temporal functional dependencies $F$ is in *temporal Boyce-Codd normal form* (TBCNF) if

$$\forall X \stackrel{\mathrm{T}}{\rightarrow} Y \in F^{+} (Y \subseteq X \vee X \stackrel{\mathrm{T}}{\rightarrow} R)$$

where $F^{+}$ denotes the closure of $F$ and $X$ and $Y$ are sets of attributes of $R$.

Similarly, $(R, F)$ is in *temporal third normal form* (T3NF) if for all non-trivial temporal functional dependencies $X \stackrel{\mathrm{T}}{\rightarrow} Y$ in $F^{+}$, $X$ is a temporal superkey for $R$ or each attribute of $Y$ is part of a minimal temporal key of $R$.

The definition of *temporal fourth normal form* (T4NF) is similar to that of TBCNF, but also uses temporal multivalued dependencies.

### Alternative Names

Time normal form, P normal form, Q normal form, first temporal normal form.

### Discussion

The three temporal normal forms mentioned in the definition are not a complete account of temporal normal forms. Indeed, the alternative names refer to different and complementing notions of temporal normal forms.

The naming of the concepts is orthogonal with respect to existing snapshot concepts, and the new names are mutually consistent (+E1, +E7).

## 2.58 Calendar

### Definition

A *calendar* provides a human interpretation of time. As such, calendars ascribe meaning to temporal values where the particular meaning or interpretation is relevant to the user. In particular, calendars determine the mapping between human-meaningful time values and an underlying time-line.

### Discussion

Calendars are generally cyclic, allowing human-meaningful time values to be expressed succinctly. For example, dates in the common Gregorian calendar may be expressed in the form <*month day, year*> where each of the fields month, day, and year cycle as time passes.

The concept of calendar defined here subsumes commonly used calendars such as the Gregorian calendar, the Hebrew calendar, and the Lunar calendar, though the given definition is much more general. This usage is consistent with the conventional English meaning of the word (+E3). It is also intuitive for the same reason (+E8).

## 2.59 Gregorian Calendar

### Definition

The *Gregorian calendar* is composed of 12 months, named in order, January, February, March, April, May, June, July, August, September, October, November, and December. The 12 months form a year. A year is either 365 or 366 days in length, where the extra day is used on "leap years." Leap years are defined as years evenly divisible by 4, with centesimal years being excluded, unless that year is divisible by 400. Each month has a fixed number of days, except for February, the length of which varies by a day depending on whether or not the particular year is a leap year.

### Discussion

The Gregorian calendar is widely used and accepted (+E3,+E7). This term is defined and used elsewhere (−R1), but is in such common use in temporal databases that it should be defined.

## 2.60 Calendric System

### Definition

A calendric system is a collection of calendars. Each calendar in a calendric system is defined over contiguous and non-overlapping intervals of an underlying time-line. Calendric systems define the human interpretation of time for a particular locale as different calendars may be employed during different intervals.

### Discussion

A calendric system is the abstraction of time available at the conceptual (query language) level. The term "calendric system" has been used to describe the calculation of events within a single calendar—it therefore has a conflicting meaning (−E7). Our definition generalizes this usage to multiple calendars in a very natural way, however. Furthermore, our meaning is intuitive in that the calendric system interprets time values at the conceptual level (+E8).

## 2.61 Temporal Natural Join

### Definition

A temporal natural join is a binary operator that generalizes the snapshot natural join to incorporate one or more time dimensions. Tuples in a temporal natural join are merged if their explicit join attribute values match, and they are temporally coincident in the given time dimensions. As in the snapshot natural join, the relation schema resulting from a temporal natural join is the union of the explicit attribute values present in both operand schemas, along with one or more timestamps. The value of a result timestamp is the temporal intersection of the input timestamps, that is, the chronons contained in both.

**Alternative Names**

Natural time-join, time-equijoin.

**Discussion**

The snapshot natural join can be generalized to incorporate valid time (the *valid-time natural join*), transaction time (the *transaction-time natural join*), or both (the *bitemporal natural join*). In each case, the schema resulting from the join is identical to that of the snapshot natural join appended with the timestamp(s) of the input relations.

"Temporal natural join" directly generalizes the snapshot term "natural join" in that "temporal" is used as a modifier consistent with its previously proposed glossary definition (+E7). "Natural time-join" is less precise since it is unclear what is natural, i.e., is the join over "natural time" or is the time-join "natural" (−E7, −E9). "Time-equijoin" is also less precise since, in the snapshot model, the natural join includes a projection while the equijoin does not (−E7, −E9).

## 2.62 Upper Support Chronon

**Definition**

In the discrete model of time, the *upper support chronon* is the latest chronon during which an indeterminate event might have occurred.

**Alternative Names**

Upper bound.

**Discussion**

The upper support chronon is an upper bound on the possible times when an indeterminate event might have occurred. The noun "support" is preferred to "bound" because the use of the former term is consistent with probability theory (+E9). For an indeterminate event, a probability mass function gives the probability that the event occurred during each chronon. The probability that the event occurred sometime after the upper support chronon is zero.

## 2.63 Lower Support Chronon

**Definition**

In the discrete model of time, the *lower support chronon* is the earliest chronon during which an indeterminate event might have occurred.

**Alternative Names**

Lower bound.

**Discussion**

The lower support chronon is a lower bound on the possible times when an indeterminate event might have occurred. The noun "support" is preferred to "bound" because the use of the former term is consistent with probability theory (+E9). For an indeterminate event, a probability mass function gives the probability that the event occurred during each chronon. The probability that the event occurred sometime before the lower support chronon is zero.

## 2.64 Valid-time Partitioning

**Definition**

*Valid-time partitioning* is the partitioning (in the mathematical sense) of the valid time-line into *valid-time elements*. For each valid-time element, we associate an interval of the valid time-line on which a cumulative aggregate may then be applied.

**Alternative Names**

Valid-time grouping.

**Discussion**

To compute the aggregate, first partition the time-line into valid-time elements, then associate an interval with each valid-time element, assemble the tuples valid over each interval, and finally compute the aggregate over each of these sets. The value at any event is the value computed over the partitioning element that contains that event.

The reason for the *associated* interval with each temporal element is that we wish to perform a *partition* of the valid time-line, and not exclude certain queries. If we exclude computing the aggregate on overlapping intervals, we exclude queries such as "Find the average salary paid for one year before each hire." Such queries would be excluded because the one-year intervals before each hire might overlap.

Partitioning the time-line is a useful capability for aggregates in temporal databases (+R1,+R3).

Grouping is inappropriate because the valid-time elements form a true partition; they do not overlap and must cover the time line. However the associated intervals may be defined in any way.

One example of valid-time partitioning is to divide the time-line into years, based on the Gregorian calendar. Then for each year, compute the count of the tuples which overlap that year.

There is no existing term for this concept. There is no partitioning attribute in valid-time partitioning, since the partitioning does not depend on attribute values, but instead on valid-times.

Valid-time partitioning may occur before or after value partitioning.

## 2.65 Dynamic Valid-time Partitioning

### Definition

In *dynamic valid-time partitioning* the valid-time elements used in the partitioning are determined solely from the timestamps of the relation.

### Alternative Names

Moving window.

### Discussion

The term dynamic is appropriate (as opposed to static) because if the information in the database changes, the partitioning intervals may change. The intervals are determined from intrinsic information.

One example of dynamic valid-time partitioning would be to compute the average value of an attribute in a relation (say the salary attribute), for the previous year before the stop-time of each tuple. A technique which could be used to compute this query would be for each tuple, find all tuples valid in the previous year before the stop-time of the tuple in question, and combine these tuples into a set. Finally, compute the average of the salary attribute values in each set.

It may seem inappropriate to use valid-time elements instead of intervals, however there is no reason to exclude valid-time elements. Perhaps the elements are the intervals during which the relation is constant.

The existing term for this concept does not have an opposing term suitable to refer to static valid-time partitioning, and can not distinguish between the two types of valid-time partitioning (−E3, +E9). Various temporal query languages have used both dynamic and static valid-time partitioning, but have not always been clear about which type of partitioning they support (+E1). Utilization of these terms will remove this ambiguity from future discussions.

## 2.66 Static Valid-time Partitioning

### Definition

In *static valid-time partitioning* the valid-time elements used are determined solely from fixed points on a calendar, such as the start of each year.

### Alternative Names

Moving window.

### Discussion

This term further distinguishes existing terms (−E3, +E9). It is an obvious parallel to dynamic valid-time partitioning (+E1). Static is an appropriate term because the valid-time elements are determined from extrinsic information. The partitioning element would not change if the information in the database changed.

Computing the maximum salary of employees during each month is an example which requires using static valid-time partitioning. To compute this information, first divide the time-line into valid-time elements where each element represents a separate month on, say, the Gregorian calendar. Then, find the tuples valid over each valid-time element, and compute the maximum aggregate over the members of each set.

## 2.67 Valid-time Cumulative Aggregation

### Definition

In *cumulative aggregation*, for each valid-time element of the valid-time partitioning (produced by either dynamic or static valid-time partitioning), the aggregate is applied to all tuples associated with that valid-time element.

The value of the aggregate at any event is the value computed over the partitioning element that contains that event.

### Alternative Names

Moving window.

### Discussion

*Cumulative* is used because the interesting values are defined over a cumulative range of time (+E8). This term is more precise than the existing term (−E3, +E9). Instantaneous aggregation may be considered to be a degenerate case of cumulative aggregation where the partition is per chronon and the associated interval is that chronon.

One example of cumulative aggregation would be find the total number of employees who had worked at some point for a company. To compute this value at the end of each calendar year, then, for each year, define a valid-time element which is valid from the beginning of time up to the end of that year. For each valid-time element, find all tuples which overlap that element, and finally, count the number of tuples in each set.

### 2.68 Instantaneous Aggregation

**Definition**

In *instantaneous aggregation*, for each chronon on the valid time-line, the aggregate is applied to all tuples valid at that event.

**Discussion**

The term *instantaneous* is appropriate because the aggregate is applied over every chronon, every event. It suggests an interest in the aggregate value over a very small time interval, an instant, much as acceleration is defined in physics over an infinitesimally small time (+R3).

Many temporal query languages perform instantaneous aggregation, others use cumulative aggregation, while still others use a combination of the two. This term will be useful to distinguish between the various alternatives, and is already used by some researchers (+R4,+E3).

## 3 Unresolved Proposals

The following glossary entries were either submitted close to the deadline for contributions to this document or were affected by entries submited close to the deadline. Some entries were being discussed actively; other entries were proposed so late that there was no time for comments from the community. In comparison with the other proposed terms, these proposals are relatively unresolved.

### 3.1 Temporal Value Integrity

**Definition**

A temporal DBMS is said to have *temporal value integrity* if:

1. The integrity of temporal values as first-class objects is inherent in the model, in the sense that the language provides a mechanism (generally, variables and quantification) for direct reference to *value histories* as objects of discourse, and

2. Temporal values are considered to be *value equivalent* only if they are equal for all points in time over which they are defined.

**Discussion**

The concept of *temporal value integrity* provides a term for the characteristic distinguishing those models which represent *time* as just another attribute or set of attributes, from those which represent temporal values directly. The former models do not have a primitive notion of a temporal value. Instead, they have the primitive notions of time values and ordinary values, and they can represent associations between these two types of values, for example, they can represent the (non-temporal) *value of a SALARY at time t*. Those models with *temporal value integrity* have built in the primitive notion of a temporal value. In these models one can refer to a primitive temporal value like a *SALARY history*, as well as referring to the (non-temporal) *value of a SALARY history at time t*.

The orthogonality criterion (+E1) is satisfied, and there are no competing names in the literature (+E3), and the term does not appear to have other meanings (+E5). Further, the name is consistent with existing terminology (+E7) (and, indeed, clarifies the meaning of the term *value equivalence*), and does not violate other criteria.

### 3.2 Coalesce

**Definition**

The *coalesce* operation takes as argument a set of value-equivalent tuples and returns a single tuple which is snapshot equivalent with the argument set of tuples.

**Alternative Names**

Merging.

**Discussion**

Coalesce is an example of a snapshot-equivalence preserving operation which reduces the cardinality of a set of argument tuples.

The concept of coalescing has found widespread use in connection with data models where tuples are associated with interval-valued timestamps. In such models, two or more value-equivalent tuples with consecutive or overlapping timestamps typically are required to be or may be replaced by a single, value-equivalent tuple with an interval-valued timestamp which is the union of the timestamps of the original tuples.

There appears to be general consensus with respect to the name of this concept (+E3). The name "merging" is occasionally used when describing coalescing, but it has a less specific meaning and has not been proposed as a substitute for "coalescing" (−E3, −E9).

## 3.3 Period of Indeterminacy

**Definition**

The *period of indeterminacy* is either an anchored duration associated with an indeterminate event or a duration associated with an indeterminate span, that delimits the range of possible times represented by the event or span.

**Alternative Names**

Interval of indeterminacy, fuzzy interval.

**Discussion**

The period of indeterminacy associated with an indeterminate event is an anchored duration that delimits the range of possible times during which the event occurred. The event happened sometime during the period of indeterminacy but it is unknown exactly when. An anchored duration is usually referred to as an interval, however, in this context, we prefer to call it a period because the syntactic difference between an "indeterminate interval" and an "interval of indeterminacy" is slight, while the semantic difference is great. Hence, while using "interval of indeterminacy" might be more precise (+E9), it would also be more confusing (−E8). Using "fuzzy interval" would also be confusing due to the influence of fuzzy databases (+E5).

## 3.4 Admissibility Interval

**Definition**

Same as "period of indeterminacy."

**Alternative Names**

Period of indeterminacy.

**Discussion**

The name "admissibility interval" is more intuitive than "period of indeterminacy" (+E8) and was used in the TSOS system (+E7).

## 3.5 Chronologically Definite

**Definition**

The modifier *chronologically definite* indicates that a fact or an event has associated a valid time at a given timestamp granularity.

**Alternative Names**

Absolute time.

**Discussion**

A chronologically definite event or fact has associated a time (see also the discussion about "temporally indeterminate") and that this time does not depend on the time of other events or facts. For instance: Mary's salary was raised on March 30, 1993. The time associated to chronologically definite events has also been called absolute time in the literature.

## 3.6 Chronologically Indefinite

**Definition**

The modifier *chronologically indefinite* indicates that the time of a fact or an event is related to the occurrence of another event.

**Alternative Names**

Imprecise, relative.

**Discussion**

Example are: Mary's salary was raised yesterday. (here it depends on the utterance time for the sentence). Mary's salary was raised before Lucy's. The time associated to chronologically indefinite events has also been called relative time in the literature.

## 3.7 Time Indeterminacy

**Definition**

Information that is *time indeterminate* can be characterized as "don't know when" information, or more precisely, "don't know *exactly* when" information. The most common kind of time indeterminacy is valid-time indeterminacy or user-defined time indeterminacy. Transaction-time indeterminacy is rare because transaction times are always known exactly.

**Alternative Names**

Fuzzy time, time imprecision, time incompleteness.

**Discussion**

Often a user knows only approximately when an event happened, when an interval began and ended, or even the duration of a span. For instance, she may know that an event happened "between 2 PM and 4 PM," "on Friday," "sometime last week," or "around the middle of the month." She may know that a airplane left "on Friday" and arrived "on Saturday." Or perhaps, she has information that suggests that a graduate student takes "four to fifteen" years to write a dissertation. These are examples of time indeterminacy. The adjective "time" allows parallel kinds of indeterminacy to be defined, such as spatial indeterminacy (+E1). We prefer "time indeterminacy" to "fuzzy time" since fuzzy has a specific, and different, meaning in database contexts (+E8). There is a subtle difference between indeterminate and imprecise. In this context, indeterminate is a more general term than imprecise since precision is commonly associated with making measurements. Typically, a precise measurement is preferred to an imprecise one. Imprecise time measurements, however, are just one source of time indeterminate information (+E9). On the other hand, "time incompleteness" is too general. Time indeterminacy is a specific kind of time incomplete information.

## 3.8 Temporally Indeterminate

### Definition

The modifier *temporally indeterminate* indicates that a fact or event it is known to have occurred, but it is unknown precisely when.

### Alternative Names

Vague, imprecise.

### Discussion

There are (at least) two possible sources of indeterminacy: (i) a discrepancy between the granularities of the temporal qualification and the occurrence time; (ii) an underspecification of the occurrence time, when the granularities of the temporal qualification and the occurrence time coincide.

The proposed definition of *temporally-indeterminate* event is: "a *temporally-indeterminate* event is an event that is known to have occurred but precisely when is unknown". Reformulated in terms of statements it becomes: "a *temporally-indeterminate* statement is a statement that allows us to conclude that an event has occurred, but it does not tell us precisely when it has occurred."

Chronologically-indefinite statements are also temporally indeterminate, but not vice versa: temporally-indeterminate statements can be chronologically indefinite as well as chronologically definite.

The statements "Jack was killed on xx/xx/1990" and "Michelle was born yesterday" come within different categories with respect to the chronological definiteness/indefiniteness characterization, but they are both temporally indeterminate.

As a first approximation, we can say that a statement is *temporally indeterminate* if the granularity of its temporal qualification (in the examples, the granularity of days) is coarser than the granularity of the time at which the denoted events (instantaneously) occur. Notice that temporal indeterminacy as well as chronological indefiniteness are mainly qualifications of statements rather than of the events they denote (better, temporal indeterminacy characterizes the relation between the granularities of the statement temporal qualification and of the event occurrence time). Notice also that it does not depend on the time at which the statement is evaluated. The crucial, and critical, point is clearly the determination of the time granularity of the event occurrence time.

Some problems could be avoided by adopting the following weaker notion of temporally indeterminacy: a statement whose temporal qualification has granularity G (to say, days) is temporally determinate with respect to every coarser granularity (e.g., months) and temporally indeterminate with respect to every finer granularity (e.g., seconds).

However, we do not like this solution, because it does not take into account information about the denoted events. In particular, for each event there exists a limit time granularity such that its occurrence time can be specified at such a granularity and all coarser ones, but not at finer ones. With respect to each finer granularity, the event as a whole does not make sense at all and it must decomposed into a set of components (if possible).

Let us go back to the proposed definition of temporal indeterminacy to discuss the following issue: does temporal indeterminacy always involve a discrepancy between temporal qualification (expressed as a valid time) and occurrence time granularities? Consider the sentence: "The shop remained open on a Sunday in April 1990 all the day long". Clearly, the truth value of the statement does not depend on its utterance time, that is, the statement is chronologically defined. Furthermore, day is the granularity of both the temporal qualification and the occurrence time. Nevertheless, we believe that this statement is *temporally indeterminate*, because the precise day in which the shop remained open is unknown (we only know that it belong to the set of Sunday days in April 1990).

These sources of indeterminacy are not exclusive and they can jointly contribute to make a statement temporally indeterminate. This is the case, for instance, in the sentence: "Jack was killed on a Friday night in 1990".

## 3.9 Temporally-indeterminate Event

### Definition

A *temporally-indeterminate event* (or just *indeterminate event*, when the context is clear) is an event that is known to have occurred but precisely when is unknown. The times when the event might have occurred must be contiguous; non-contiguous times can be modeled by an exclusive-or disjunction of indeterminate events.

**Alternative Names**

Temporally-incomplete event, temporally-fuzzy event, temporally-imprecise event.

**Discussion**

"Michelle was born yesterday" is a typical indeterminate event. An indeterminate event is composed of an event (e.g., "Michelle was born") and some indeterminate temporal information (e.g., "yesterday").

Note that an event with noncontiguous temporally-indeterminate information, such as "Jack was killed on a Friday night in 1990," is not an indeterminate event since the times when the event might have occurred are non-contiguous. The incomplete temporal information could be more substantial. For instance, an indeterminate event could have an associated probability mass function which gives the probability that the event occurred during each chronon on a time-line.

Currently, there is no name used in the literature to describe the incomplete temporal information associated with an event. The modifier "incomplete" is too vague (-E9), while "fuzzy" has unwanted connotations (i.e., with fuzzy sets) (-E9). "Indeterminate" is more general than "imprecise;" imprecise commonly refers to measurements, but imprecise clock measurements are only one source of indeterminate events.

## 3.10 Temporally-indeterminate Interval

**Definition**

A *temporally-indeterminate interval* (or just *indeterminate interval* when the context is clear) is an interval bounded by at least one temporally-indeterminate event. Since an interval cannot end before it starts, the possible times associated with the bounding events can overlap on only a single chronon.

**Alternative Names**

Temporally-incomplete interval, temporally-fuzzy interval, temporally-imprecise interval.

**Discussion**

Currently, there is no name used in the literature to describe the incomplete temporal information associated with an interval. The modifier "incomplete" is too vague (−E9), while "fuzzy" has unwanted connotations (i.e., with fuzzy sets) (−E9). "Indeterminate" is more general than "imprecise;" imprecise commonly refers to measurements, but imprecise clock measurements are only one source of indeterminate intervals.

## 3.11 Temporally Determinate

**Definition**

The modifier *temporally determinate* indicates that the occurrence time of an event or fact is known precisely.

**Alternative Names**

Precise.

**Discussion**

See the discussion of the term "temporally indeterminate."

## 3.12 Temporal Modality

**Definition**

*Temporal modality* concerns the way according to which a fact originally associated with a time point or interval at a given granularity distributes itself over the corresponding time points at finer granularities or within the interval at the same level of granularity. We distinguish two basic temporal modalities, namely *sometimes* and *always*.

The *sometimes temporal modality* states that the relevant fact is true in at least one of the corresponding time points at the finer granularity for time points, or in at least one of the time points of the interval in case an interval is given. For instance: "The light was on yesterday afternoon," meaning that it was on at least for one minute in the afternoon (assuming minute as temporal quantum).

The *always temporal modality* states that the relevant fact is true in each corresponding time point at the finer granularity. This is the case, for instance, of the sentence: "The shop remained open on a Sunday in April 1990 all the day long" with respect to the granularity of hour.

This issue is relate to attributes varying within their validity intervals.

## 3.13 Temporal Rule

**Definition**

A database rule is a *temporal rule* if either the condition part or the action part involve time points or temporal elements.

### Discussion

This definition is intended to distinguish between temporal and non-temporal rules in temporal active databases. A non-temporal rule can still cause changes over time in the case of retroactive or proactive (may change to predictive if term adopted) changes to temporal data. The concept has recently been used in many papers (+R4), it is well understood as defined above (+R3). There may be a problem with the preciseness (−E9).

## 3.14 Time Sequence

### Definition

A *time sequence* (TS) is a sequence (ordered by time) of pairs $< v, t >$ where $v$ is an arbitrary data object and $t$ are time points of a given granularity designating past and/or future times. A TS is identified by a surrogate (possibly a time-invariant key). If each $v$ is a single value, the TS is said to be *simple*, and if $v$ is a complex value (e.g., a set, a sequence, etc.), the TS is *complex*. A TS may have properties and/or constraints attached to it.

### Alternative Names

History, time-series.

### Discussion

The above definition is model-independent and can have different representations in different models. For example in the relational model where a relation is attribute-value timestamped (points), each point in the sequence will be a tuple. For tuple timestamping, $v$ will be a set of attribute values. Note that temporal elements are derivable from a time sequence.

The concept is specific to temporal databases (+R1) and is well defined and understood in the real world (+R2, +R3). It has been used and referred to in many works (+R4). The name is intuitive (+E8), it is not as widely used as "history" (−E3), but it describes the concept more accurately (+E9) than "history," i.e., the common use of history is in reference to the past, but a temporal database can have a time sequence that involves future times.

## 3.15 Temporal Interpolation

### Definition

Deriving a temporal value at a time point of a time sequence which is not stored explictly in the database, as a function of preceding and succeeding values, is referred to as *temporal interpolation*. *Temporal extrapolation* is defined similarly.

### Alternative Names

Temporal derivation.

### Discussion

This concept is important for large sequences (in particular, for continuous scientific data) where data is collected only for a subset of the time points in the time sequence, or all time points contain data, but interpolation is used as a form of compression. The alternative name of *temporal derivation* will apply if the definition is extended to encompass cases where the derivation is not based on interpolation, but on other computations or rules.

The concept is specific to temporal databases (+R1) and its essence—interpolation—is well-defined and understood in the real world (+R2, +R3). The name is intuitive (+E8).

# A Relevance Criteria for Concepts

It has been attempted to name only concepts that fulfill the following four requirements.

**R1** The concept must be specific to temporal databases. Thus, concepts used more generally are excluded.

**R2** The concept must be well-defined. Before attempting to name a concept, it is necessary to agree on the definition of the concept itself.

**R3** The concept must be well understood. We have attempted to not name a concept if a clear understanding of the appropriateness, consequences, and implications of the concept is missing. Thus, we avoid concepts from research areas that are currently being explored.

**R4** The concept must be widely used. We have avoided concepts used only sporadically within the field.

# B Evaluation Criteria for Naming Concepts

Below is a list of criteria for what is a good name. Contributors of glossary entries have been encouraged to reference these criteria when proposing glossary entries. The criteria are sometimes conflicting, making the choice of names a difficult and challenging task. While this list is comprehensive, it is not complete.

**E1** The naming of concepts should be orthogonal. Parallel concepts should have parallel names.

**E2** Names should be easy to write, i.e., they should be short or possess a short acronym, should be easily pronounced (the name or its acronym), and should be appropriate for use in subscripts and superscripts.

**E3** Already widely accepted names are preferred over new names.

**E4** Names should be open-ended in the sense that the name of a concept should not prohibit the invention of a parallel name if a parallel concept is defined.

**E5** The creation of homographs and homonyms should be avoided. Names with an already accepted meaning, e.g., an informal meaning, should not be given an additional meaning.

**E6** The naming of concepts should be conservative. No name is better than a bad name.

**E7** New names should be consistent with related and already existing and accepted names.

**E8** Names should be intuitive.

**E9** Names should be precise.

# Index