Proposed Glossary Entries—December 1992

Christian S. Jensen, editor*

Abstract

This document describes the current status, as of December 15, 1992, of an initiative aimed at creating a consensus glossary of temporal database concepts and names. It contains the set of currently proposed, complete glossary entries. Existing terms and criteria for evaluation of glossary entries are contained in appendices.

The document is intended to help future contributors of glossary entries. Proposed glossary entries should be sent to `tsql@cs.arizona.edu`. Other information related to the initiative may be found at `cs.arizona.edu` in the `tsql` directory, accessible via anynomous ftp.

1 Introduction

This document is a structured presentation of the current status of an initiative aimed at creating a consensus glossary of temporal database terms. It contains the list of complete proposals for temporal database concepts and names which have so far been submitted to the mailing list `tsql@cs.arizona.edu`. The purpose of the document is to give potential contributors an overview of the terms proposed so far.

In order to obtain a consensus glossary, the proposed concepts and names are intended to be discussed during the first workshop on temporal databases ("International Workshop on an Infrastructure for Temporal Databases"), scheduled to be held in Arlington, TX, June 14-16, 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.

During the preparation of a forthcoming book on temporal databases (*Temporal Databases: Theory, Design, and Implementation,* edited by A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A.d Segev, and R. Snodgrass, Benjamin/Cummings Publishers, Database Systems and Applications Series), a glossary on temporal database concepts and terms was developed. The glossary also appears in the September 1992 issue of the SIGMOD Record. The terms and concepts from this glossary are included here as an appendix in a dictionary-like format. Evaluation criteria, to be used when proposing new glossary entries, are also included in the appendix.

*These definitions were proposed by Jim Clifford, Curtis Dyreson, Shashi Gadia, Sushil Jajodia, Christian S. Jensen, Nick Kline, Daniel Nonen, John F. Roddick, Arie Segev, Richard T. Snodgrass, Mike D. Soo, and Abdullah Tansel. 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`.

2 New, Proposed Glossary Entries

2.1 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.

2.2 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.

2.3 Schema Versioning

Definition

A database system accomodates *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.

2.4 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 τ^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).

2.5 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).

2.6 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).

2.7 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 τ^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

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

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.

Alternative Names

None.

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-endend (+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.

2.8 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).

2.9 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).

2.10 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).

2.11 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.

2.12 Time-line Clock Granularity

Definition

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

Alternative Names

None.

Discussion

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

2.13 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).

2.14 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).

2.15 Initiation

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).

2.16 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.

Alternative Names

None.

Discussion

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

2.17 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).

2.18 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.

2.19 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).

2.20 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).

2.21 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.

2.22 Retroactive Bitemporal Relation

Definition

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

Alternative Names

None.

Discussion

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

2.23 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).

2.24 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.

Alternative Names

None.

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.

2.25 Tick

Definition

Same as definition of "chronon".

Alternative Names

Chronon, instant, atomic time unit, time unit.

Discussion

Tick is concise, intuitive, and unpretentious.

A Relevance Criteria for Concepts

It must be 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. These criteria should be referenced when proposing a glossary entry. 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 We have avoided creating homographs and homonyms. Names with an already accepted meaning, e.g., an informal meaning, should not be given an additional meaning.

E6 We have striven to be conservative when naming concepts. 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.

C Overview of Existing Terms

The following list of temporal database terms appeared as complete glossary entries in "Jensen, C. S., J. Clifford, S. K. Gadia, A. Segev, and R. T. Snodgrass: *A Glossary of Temporal Database Concepts, ACM SIGMOD Record*, Vol. 21, No. 3, September 1992, pp. 35–43.

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

chronon A *chronon* is the shortest duration of time supported by a temporal DBMS—it is a non-decomposable unit of time. A particular chronon is a subinterval of fixed duration on time-line. 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."

event 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.

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

lifespan 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.

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

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."

span 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.

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

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

temporal element A *temporal element* is a finite union of n-dimensional time boxes. Temporal elements are closed under the set theoretic operations of union, intersection and complementation. Temporal elements may be used as timestamps. Special cases of temporal elements occur as timestamps in valid-time relations, transaction-time relations, and bitemporal relations. These special cases are termed *valid-time elements*, *transaction time elements*, and *bitemporal elements*. They are defined as finite unions of valid-time intervals, transaction-time intervals, and bitemporal rectangles, respectively.

temporal expression 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. 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.

temporally homogeneous 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."

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.

time-invariant attribute 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.

timestamp 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.

transaction time 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.

transaction-time relation 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.

transaction timeslice operator 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.

user-defined time *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."

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.

valid time 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.

valid-time relation 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).

valid timeslice operator 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.