A Consensus Glossary of Temporal Database Concepts*

Christian S. Jensen James Clifford Ramez Elmasri
Shashi K. Gadia Pat Hayes Sushil Jajodia (editors)
Curtis Dyreson Fabio Grandi Wolfgang Käfer Nick Kline Nikos Lorentzos
Yannis Mitsopoulos 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 Gio Wiederhold

Abstract

This document contains definitions of a wide range of concepts specific to and widely used within temporal databases. In addition to providing definitions, the document also includes separate explanations of many of the defined concepts. Two sets of criteria are included. First, all included concepts were required to satisfy four relevance criteria, and, second, the naming of the concepts was resolved using a set of evaluation criteria. The concepts are grouped into three categories: concepts of general database interest, of temporal database interest, and of specialized interest. This document is a digest of a full version of the glossary[1]*. In addition to the material included here, the full version includes substantial discussions of the naming of the concepts.*

The consensus effort that lead to this glossary was initiated in Early 1992. 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. The present glossary subsumes all the previous documents. It was most recently discussed at the "ARPA/NSF International Workshop on an Infrastructure for Temporal Databases," in Arlington, TX, June 1993, and is recommended by a significant part of the temporal database community. The glossary meets a need for creating a higher degree of consensus on the definition and naming of temporal database concepts.

1 Introduction

A technical language is an important infrastructural component of any scientific community. To be effective, such a language should be well-defined, intuitive, and agreed-upon. This document contains recommended definitions and names for a wide 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.

This document is a digest of a full version of the glossary. The full version does not contain any additional definitions, but includes discussions of the particular choices of names for the concepts. Thus, when several different names were previously used for a concept, the alternatives are enumerated and discussed.

The history of this document (and its full version) 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, from Benjamin/Cummings Publishers. That glossary also appeared in the September 1992 issue of the ACM SIGMOD Record. The effort continued, independently of the book, and the community was invited to submit proposals to an open mailing list. As results, status documents appeared in December 1992 and in March 1993. In June 1993, a complete document of the 100 glossary entries proposed to date was discussed among 44 temporal database researchers at the "ARPA/NSF International Workshop on an Infrastructure for Temporal Databases," in Arlington, TX, with the goal of obtaining a widely agreed upon glossary. An editorial board has since supervised a revision of the glossary based on the input from the workshop. The present glossary is a digest of the result. Each individual who contributed significantly to the glossary effort is a coauthor of this document.

The document is organized as follows. The next section first lists four relevance criteria for concepts, then lists nine evaluation criteria for the naming of concepts. Finally, the structure of a glossary entry for a concept is explained. The next three sections constitute the main body of the glossary and contain glossary entries for concepts. The first includes entries for concepts that are expected to be of interest to researchers within the general database area. The second covers concepts that are expected to be of general interest

*Correspondence may be directed to the electronic mail distribution, `tdbglossary@cs.arizona.edu`, or to the editor, C. S. Jensen, at Aalborg University, Datalogi, Fr. Bajers Vej 7E, DK–9220 Aalborg Ø, Denmark, `csj@iesd.auc.dk`. The affiliations and e-mail addresses of the many contributors may be found in a separate section at the end of the document.

[1] The full version of the glossary is available via anonymous ftp from `cs.arizona.edu` where it may be found in the `tsql` directory along with other material related to the ongoing glossary initiative. It is also available as Technical Report R 93–2035 (November 1993) from Aalborg University.

within temporal databases only. The third covers the remaining concepts of more specialized interest. Finally, the affiliations and e-mail addresses of the authors are listed, and an index is included on the last page.

2 Relevance and Evaluation Criteria for the Glossary

2.1 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.

2.2 Evaluation Criteria for Naming Concepts

Below is a list of criteria for what is a good name. Contributors 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 the 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.

2.3 Structure of the Glossary

The following template is used for presenting the concepts of the glossary.

Name—the chosen name of the concept is used as the heading.

Definition—the definition of the concept.

Explanation—further exploration of the definition and its consequences, including exemplification; this section is optional.

The full version extends the template with these items.

Previously Used Names—list of previously used names.

Discussion of Naming—reasons for the particular choice of name (and concept) and reasons for not selecting previously used names (and concepts).

Names of concepts that are defined in the glossary are typeset with a special font. For example, valid time and transaction time have entries in the glossary. The special font is only used for the first occurrence of a name in a subsection of a glossary entry, and only if the entry of the name may be found in the current section or in an earlier section. To locate a particular glossary entry, use the index at the end of the document.

3 Concepts of General Database Interest

3.1 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 instants and time intervals, with single instants and intervals being important special cases. Valid times are usually supplied by the user.

3.2 Transaction Time

Definition

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

3.3 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."

3.4 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.

3.5 Valid-time Relation

Definition

A *valid-time relation* is a relation with exactly one system supported valid time. There are no restrictions on how valid times may be incorporated into the tuples; e.g., the valid-times may be incorporated by including one or more additional valid-time attributes in the relation schema, or by including the valid-times as a component of the values of the application-specific attributes.

3.6 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 incorporated into the tuples.

3.7 Snapshot Relation

Definition

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

3.8 Bitemporal Relation

Definition

A *bitemporal relation* is a relation with exactly one system supported valid time and exactly one system-supported transaction time. As for valid-time relations and transaction-time relations, there are no restrictions as to how either of these temporal dimensions may be incorporated into the tuples.

Explanation

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.

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, see Section 3.9. 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).

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.

3.9 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.

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.

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. Nouns specific to temporal databases, such as instant, chronon, interval, element, and span, may be modified by "valid-time," "transaction-time," and "bitemporal."

3.10 Temporal as Modifier

Definition

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

3.11 Temporal Database

Definition

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

3.12 Instant

Definition

An *instant* is a time point on an underlying time axis.

Explanation

Various models of time have been proposed in the philosophical and logical literature of time. These view time, among other things, as discrete, dense, or continuous. Intuitively, the instants in a discrete model of time are isomorphic to the natural numbers, i.e., there is the notion that every instant has a unique successor. Instants in the dense model of time are isomorphic to (either) the real or rational numbers: between any two instants 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."

In a data model that supports a time line using chronons (isomorphic to the natural numbers or a subset thereof), an instant is represented by a chronon. A single chronon may therefore represent multiple instants.

3.13 Chronon

Definition

In a data model, a one-dimensional *chronon* is a non-decomposable time interval of some fixed, minimal duration. 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.

Explanation

Data models may represent a time line by a sequence of non-decomposable, consecutive time intervals of identical duration. These intervals are termed chronons. A data model will typically leave the particular chronon duration unspecified, to be fixed later by the individual applications, within the restrictions posed by the implementation of the data model.

3.14 Time Interval

Definition

A *time interval* is the time between two instants. In a system that supports a time line composed of chronons, an interval may be represented by a set of contiguous chronons.

3.15 Temporal Element

Definition

A *temporal element* is a finite union of n-dimensional time 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.

Explanation

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.

3.16 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 instants. 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.

3.17 Timestamp

Definition

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

3.18 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.

3.19 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.

Explanation

Calendars are most often 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.

3.20 Transaction-timeslice Operator

Definition

The *transaction timeslice operator* may be applied to any relation with transaction time timestamps. It takes as one argument the relation and as a second argument a transaction-time element whose greatest value must not exceed the current transaction time. It returns the argument relation reduced in the transaction-time dimension to just those times specified by the transaction-time argument.

Explanation

Several types of transaction-timeslice operators are possible. Some may restrict the type of the time argument to intervals or instants. Some operators may, given an instant as time argument, return a snapshot relation or a valid-time relation when applied to a transaction-time or a bitemporal relation, respectively; other operators may always return a result relation of the same type as the argument relation.

3.21 Valid-timeslice Operator

Definition

The *valid timeslice operator* may be applied to any relation with valid time timestamps. It takes as one argument the relation and as a second argument a valid-time element. It returns the argument relation reduced in the valid-time dimension to just those times specified by the valid-time argument.

Explanation

Several types of valid-timeslice operators are possible. Some may restrict the type of the time argument to intervals or instants. Some operators may, given an instant as time argument, return a snapshot relation or a transaction-time relation when applied to a valid-time or a bitemporal relation, respectively; other operators may always return a result relation of the same type as the argument relation.

3.22 Schema Evolution

Definition

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

3.23 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.

Explanation

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.

3.24 Event

Definition

An *event* is an instantaneous fact, i.e., something occurring at an instant. An event is said to occur at a chronon t if it occurs at any instant during t.

3.25 Event Occurrence Time

Definition

The *event occurrence time* of an event is the valid-time instant at which the event occurs in the real-world.

3.26 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.

3.27 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.

3.28 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.

4 Concepts of General Temporal Database Interest

4.1 Absolute Time

Definition

The modifier *absolute* indicates that a specific valid time at a given timestamp granularity is associated with a fact. Such a time depends neither on the valid time of another fact nor on the current time, *now*.

Explanation

Examples are: "Mary's salary was raised on March 30, 1993" and "Jack was killed on 10/12/1990."

Notice that absolute times are associated with chronologically definite statements only.

4.2 Relative Time

Definition

The modifier *relative* indicates that the valid time of a fact is related to either the valid time of another fact or the current time, *now*.

Explanation

The relationship between times can be qualitative (before, after, etc.) as well as quantitative (3 days before, 397 years after, etc.).

Examples are: "Mary's salary was raised yesterday," "it happened sometime last week," "it happened within 3 days of Easter," "the Jurassic is sometime after the Triassic," and "the French revolution occurred 397 years after the discovery of America."

Notice that both chronologically indefinite and definite statements can involve relative times.

4.3 Temporal Expression

Definition

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

Explanation

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 be defined through mutual recursion.

4.4 Fixed Span

Definition

A span is *fixed* if it possesses the special property that its duration is independent of the context.

Explanation

As an example of a fixed span, "one hour" always, independently of the context, has a duration of 60 minutes (discounting leap seconds). To see that not all spans are fixed, consider "one month," an example of a variable span in the Gregorian calendar. The duration of this span may be any of 28, 29, 30, and 31 days, depending on the context.

4.5 Variable Span

Definition

A span is *variable* if its duration is dependent on the context.

Explanation

Any span is either a fixed span or a variable span. An obvious example of a variable span is "one month," the duration of which may be any of 28, 29, 30, and 31 days, depending on the context. Disregarding the intricacies of leap seconds, the span "one hour" is fixed because it always, independently of the context, has a duration of 60 minutes.

4.6 Bitemporal Interval

Definition

A *bitemporal interval* is a region in two-space of valid time and transaction time, with sides parallel to the axes. When associated in the database with some fact, it identifies when that 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.

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

4.7 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. When associated in the database with some fact, it identifies when and where that fact was true.

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

4.8 Spatiotemporal Element

Definition

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

4.9 Snapshot Equivalent/ Weakly Equivalent

Definition

Informally, two tuples are *snapshot equivalent* or *weakly 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 or weakly equivalent if at every instant their snapshots are equal. Snapshot equivalence, or weak equivalence, is a binary relation that can be applied to tuples and to relations.

4.10 Snapshot-Equivalence Preserving Operator/Weakly Invariant Operator

Definition

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

4.11 Snapshot Equivalence Class/ Weak Relation

Definition

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

4.12 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

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

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.

Explanation

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.

4.13 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.

Explanation

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.

4.14 History

Definition

A *history* is the temporal representation of an "object" of the real world or of a database. Depending on the object, we can have *attribute histories, entity histories, relationship histories, schema histories, transaction histories,* etc.

Explanation

"History" is a general concept, intended in the sense of "train of events connected with a person or thing".

In the realm of temporal databases, the concept of history is intended to include multiple time dimensions as well as the data models. Thus we can have valid-time histories, transaction-time histories, bitemporal histories, user-defined time histories, etc. However, multi-dimensional histories can be defined from mono-dimensional ones (e.g. a bitemporal history can be seen as the transaction-time history of a valid-time history).

Formally or informally, the term "history" has been often used in many temporal database papers, also to explain other terms. For instance, salary history, object history, transaction history are all expressions used in this respect.

4.15 History-oriented

Definition

A temporal DBMS is said to be *history-oriented* if:

1. It supports history unique identification (e.g. via time-invariant keys, surrogates or OIDs);

2. The integrity of histories as first-class objects is inherent in the model, in the sense that history-related integrity constraints might be expressed and enforced, and the data manipulation language provides a mechanism (e.g., history variables and quantification) for direct reference to *object-histories*;

4.16 History Equivalent

Definition

Two objects are *history equivalent* if their histories are shapshot equivalent. *History equivalence* is a binary relation that can be applied to objects of any kind (of the real world or of a database).

Explanation

Unlike value equivalence which concerns only explicit-attribute values and completely disregards time, history equivalence implies a common evolution along with time (implicitly assumes equality of timestamps as well as explicit-attributes values).

4.17 Temporal Interpolation

Definition

The derivation of the value of a history at a chronon for which a value is not explicitly stored in the database, is referred to as *temporal interpolation*. This derivation is typically expressed as a function of preceding and/or succeeding (in time) values of the history.

Explanation

This concept is important for large histories (in particular, for continuous scientific data) where data is collected only for a subset of the chronons in the history, or where all chronons 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.

4.18 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.

4.19 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.

4.20 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.

Explanation

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).

4.21 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.

Explanation

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.

4.22 Time-line Clock Granularity

Definition

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

4.23 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 instant preceding the smallest chronon on the valid-time line. Beginning has no transaction-time semantics.

4.24 Forever

Definition

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

4.25 Initiation

Definition

The distinguished value *initiation,* associated with a relation, denotes the time instant when a relation was created. "Initiation" is a value in the domain of transaction times and has no valid-time semantics.

4.26 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.

4.27 Timestamp Granularity

Definition

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

Explanation

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."

4.28 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.

4.29 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.

Explanation

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. 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.

4.30 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 instants contained in both.

4.31 Temporal Dependency

Definition

Let X and Y be sets of explicit attributes of a temporal relation schema, R. A *temporal functional dependency*, denoted $X \overset{\mathrm{T}}{\rightarrow} Y$, 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 straightforward manner.

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

Explanation

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.

4.32 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

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

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 $X \overset{\mathrm{T}}{\rightarrow} Y$ 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.

4.33 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.

4.34 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.

4.35 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."

Explanation

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.

4.36 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.

Explanation

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.

4.37 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.

4.38 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.

4.39 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.

4.40 Degenerate Bitemporal Relation

Definition

A bitemporal relation schema is *degenerate* if updates to its 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.

4.41 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.

Explanation

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.

5 Concepts of Specialized Interest

5.1 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.

Explanation

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 instant is the value computed over the partitioning element that contains that instant.

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.

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.

5.2 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.

Explanation

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.

5.3 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.

Explanation

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.

5.4 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 instant is the value computed over the partitioning element that contains that instant.

Explanation

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.

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.

5.5 Instantaneous Aggregation

Definition

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

5.6 Temporal Modality

Definition

Temporal modality concerns the way according to which a fact originally associated with a chronon or interval at a given granularity distributes itself over the corresponding chronons at finer granularities or within the interval at the same level of granularity.

Explanation

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 chronons at the finer granularity, or in at least one of the chronons 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 minutes as chronons).

The *always temporal modality* states that the relevant fact is true in each corresponding chronon 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 chronon granularity of hour.

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

5.7 Macro-event

Definition

A *macro-event* is a wholistic fact with duration, i.e., something occurring over an interval taken as a whole. A macro-event is said to occur over an interval I if it occurs over the set of contiguous chronons representing I (considered as a whole).

Contributors

An alphabetical listing of names, affiliations, and e-mail addresses of the contributors follows.

J. Clifford, Information Systems Dept., New York University, `jcliffor@is-4.stern.nyu.edu`; R. Elmasri, Computer Science Engineering Dept., University of Texas at Arlington `elmasri@cse.uta.edu`; S. K. Gadia, Computer Science Dept., Iowa State University, `gadia@cs.iastate.edu`; P. Hayes, Beckman Institute, `Phayes@cs.uiuc.edu`; S. Jajodia, Dept. of Information & Software, George Mason University, `jajodia@sitevax.gmu.edu`; C. Dyreson, Computer Science Dept., University of Arizona, `curtis@cs.arizona.edu`; F. Grandi, University of Bologna, Italy, `fabio@deis64.cineca.it`; W. Käfer, IBM Almaden Research Center, `kaefer@almaden.ibm.com` N. Kline, Computer Science Dept., University of Arizona, `kline@cs.arizona.edu`; N. Lorentzos, Informatics Laboratory, Agricultural University of Athens, `eliop@isosun.ariadne-t.gr`; Y. Mitsopoulos, Informatics Laboratory, Agricultural University of Athens; 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`; 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`; A. Segev, School of Business Adm. and Computer Science Research Dept., University of California, `segev@csr.lbl.gov`; R. T. Snodgrass, Computer Science Dept., University of Arizona, `rts@cs.arizona.edu`; M. D. Soo, Computer Science Dept., University of Arizona, `soo@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`; G. Wiederhold, ARPA/SISTO and Stanford University, `gio@DARPA.MIL`.

Index