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 explanations of concepts as well as discussions of the adopted names.

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 the present glossary 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.

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.

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 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.

The glossary includes discussions of the particular choices that were made. Thus, when several different names were previously used for a concept, the document not only states the chosen name, but it also presents the alternatives and discusses why the decision was 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 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 100 glossary entries proposed to date was discussed among 40 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 result is the present glossary. Each individual who has contributed significantly to

*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.

the glossary effort is a coauthor of this document.

The document is organized as follows. The next section first lists four relevance for concepts, then lists nine evaluation criteria for the naming of concepts. These criteria are referenced throughout the document. 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 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 along with acknowledgements. Finally, 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. As an example of this use, the occurrence of "+E1" in a glossary entry indicates that the name in question satisfies criterion E1. Reversely, "−E1" indicates that E1 is not satisfied. 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.

2.3 Structure of the Glossary

In the following, we will name concepts selected by applying the four above principles. For most of the concepts being named, we will employ the same template:

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.

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.

Previously Used Names

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

Discussion of Naming

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).

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.

Previously Used Names

Registration time, extrinsic time, physical time, transaction commit time.

Discussion of Naming

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. Physical time is used infrequently (−E3) and seems vague (−E8).

Transaction commit time is lengthy (−E2), but more importantly, the name appears to indicate that the transaction time associated with a fact must be identical to the time when that fact is committed to the database, which is an unnecessary restriction (−E8). It is also imprecise (−E9) because the transaction time of a fact in general is a transaction-time element, not a single time instant as implied.

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."

Discussion of Naming

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.

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.

Previously Used Names

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

Discussion of Naming

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. The shorter term "temporal data type" is expected to be sufficiently descriptive.

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.

Previously Used Names

Historical relation.

Discussion of Naming

While historical relation is used currently by most authors (+E3), two problems have been pointed out. First, the qualifier "historical" is too generic (−E5). Second, "historical" might be construed as referring only to the past, which could be misleading because a valid-time relation may also contain facts predicted to be valid in the future (−E8, −E9).

"Valid-time relation" is straightforward and avoids these problems. Also, it is consistent with the name transaction-time relation (+E1).

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.

Previously Used Names

Rollback relation.

Discussion of Naming

"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, by analogy, 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)?

3.7 Snapshot Relation

Definition

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

Previously Used Names

Relation, conventional relation, static relation.

Discussion of Naming

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).

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 (+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 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.

Previously Used Names

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

Discussion of Naming

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.

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 (+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. 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.

Previously Used Names

Time-oriented.

Discussion of Naming

"Temporal" is already being used in the sense defined here. In addition, some researchers have used it 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."

3.11 Temporal Database

Definition

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

Previously Used Names

Time-oriented database, historical database.

Discussion of Naming

The concept of a temporal database is defined separately due to its importance. The discussion in Section 3.10 applies here.

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.

Previously Used Names

Event, moment.

Discussion of Naming

Event is already used widely within temporal databases, but is often given a different meaning (+E3, −E5), while "moment" may be confused with the distinct term "chronon" (−E7).

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.

Previously Used Names

Instant, moment, time quantum, time unit.

Discussion of Naming

"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).

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.

Previously Used Names

Time period.

Discussion of Naming

The name "time interval" ("interval" when "time" is clear from the context) is widely accepted (+E3). The name "period" often implies a cyclic or recurrent phenomenon (−E8, −E9).

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.

Previously Used Names

Time period set.

Discussion of Naming

The concept of a valid-time element was previously named 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 (+E1, +E9). The name "time period set" is an early term for a temporal element, but the adopted name has been used much more frequently (+E3).

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.

Previously Used Names

Duration, interval, time distance.

Discussion of Naming

It is already accepted that "(time) 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).

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.

Previously Used Names

Timestamp, temporal element, temporal domain.

Discussion of Naming

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

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.

Discussion of Naming

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).

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.

Previously Used Names

Rollback operator, timeslice operator, state query.

Discussion of Naming

The name "rollback operator" has procedural connotations, which in itself is inappropriate (−E8). The name indicates that the operator is computed by moving backwards in time, presumeably from the current transaction time. However, the operator could equally well be computed by a forward motion in time, from the time when the argument relation was created. This makes "rollforward operator" an equally acceptable term. 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.

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.

Previously Used Names

Timeslice operator.

Discussion of Naming

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

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.

Previously Used Names

Schema versioning, data evolution.

Discussion of Naming

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.

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.

Previously Used Names

Schema evolution, data evolution.

Discussion of Naming

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.

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.

Previously Used Names

Event relation, instant relation.

Discussion of Naming

"Event relation" is not consistent with the distinction between "instant and "event" (−E7). "Instant relation" is longer than event (−E2).

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.

Previously Used Names

Event time.

Discussion of Naming

Event occurrence time is more precise than event time (+E9). Nevertheless, when the context is clear, the event occurrence time may be shorthend to the event time.

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.

Previously Used Names

Spatio-temporal, temporal-spatial, space-time-oriented.

Discussion of Naming

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."

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.

Alternative Name

Spatial unit.

Discussion of Naming

"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.

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.

Alternative Name

Spatiotemporal unit, spatiotemporal chronon.

Discussion of Naming

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."

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 xx/xx/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.

Discussion of Naming

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

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.

Previously Used Names

Constant span.

Discussion of Naming

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).

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.

Previously Used Names

Moving span.

Discussion of Naming

Variable span is intuitive (+E9), and precise (+E9). "Moving span" is unintuitive (−E9) and has informal spatial connotations (−E5).

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

Discussion of Naming

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

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.

Discussion of Naming

Both "snapshot equivalent" and "weakly equivalent" have strong support by members of the temporal database community, and it has not been possible to reach a consensus on which of these two names is preferable; we have therefore adopted both names, listed in alphabetical order, for this concept.

"Weak equivalence" was originally used by Ullman to relate two algebraic expressions (Ullman, Principles of Database Systems, Second Edition, page 309). We rely on the context to disambiguate this usage from the usage specific to temporal databases.

"Snapshot equivalent" explicitly identifies the source of the equivalence (+E8).

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.

Previously Used Names

Invariant under weak binding of belongs to.

Discussion of Naming

As for snapshot equivalent and weakly equivalent, both names have strong support by members of the temporal database community. Because it has not been possible to reach a consensus on which of these two names is preferable, both names have been adopted, listed alphabetically, for this concept.

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

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.

Discussion of Naming

As for snapshot equivalent and weakly equivalent, both names have strong support by members of the temporal database community. Because it has not been possible to reach a consensus on which of these two names is preferable, both have been adopted in alphabetical order.

"Weak relation" denotes a set of relation instances, not a single relation instance. It has therefore been argued that the name snapshot equivalence class is more intuitive. On the other hand, "weak relation" is shorter than "snapshot equivalence class."

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.

Discussion of Naming

The orthogonality criterion (+E1) is satisfied. Further, the concept is a straightforward 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.

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.

Previously Used Names

Merging.

Discussion of Naming

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).

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 (+R1). 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 (+R4), also to explain other terms. For instance, salary history, object history, transaction history are all expressions used in this respect.

Previously Used Names

Time sequence, time-series, temporal value, temporal evolution.

Discussion of Naming

Although "history" usually has to do with *past* events (−E5), its use for the future—as introduced by prophecies, science fiction, scientific forecasts—does not seem to present comprehension difficulties. (The adjective "historical" seems more problematic for some.) Talking about future history, requires the same extension of meaning as required by talking about future data.

The alternative term "temporal value" is less general, since it applies when "history" specializes into attribute history (value history). Moreover, "history" is a slightly more general concept than "time sequence": different time sequences (with different time granularities) could be extracted from the same history. Therefore the definition of "history" does not prevent defining "time sequence."

"History" is also preferred over alternative names because it allows a better definition of related terms. Since it implies the idea of time, "history" does not require further qualifications as "sequence" or "series" do (+E2). In particular, "history" well lends itself to be used as modifier (+E1), even though "time sequence" is an alternative consolidated term (−E3,−E6).

"History" is natural (+E8) and precise (+E9), whereas "temporal value" may recall a temporal element (e.g., timestamp value) and "time sequence" may recall a sequence of temporal elements.

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*;

Previously Used Names

Temporal value integrity, grouped, object-oriented.

Discussion of Naming

"History-oriented" is preferred over "having temporal value integrity" since its meaning seems to be more direct. Further, in a more general perspective, integrity constraints can be introduced as well in a history-oriented model (e.g. history uniqueness, entity history integrity, referential history integrity).

"History-oriented" is also preferred over "grouped" (+E7) in order to avoid confusion with other kinds of grouping (e.g., defined terms "[dynamic/static] valid-time grouping").

"History-oriented" is not a synonym for "object-oriented," even though a good temporal object-oriented model should also be history-oriented. In general, object-orientation requires more features that are inherited from snapshot O-O models (+E7). For instance, (attribute/tuple—point/interval-stamped) relational models can also be history-oriented, provided that suitable integrity constraints and algebraic operators are defined.

Once history has been defined, "history-oriented" is quite intuitive (+E8).

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).

Previously Used Names

Snapshot equivalent.

Discussion of Naming

As indicated by the definition, "history equivalent" is closely related to snapshot equivalent (−E3). History equivalence is a natural extension to histories of the basic notion of snapshot equivalence (+E8).

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.

Previously Used Names

Temporal derivation.

Discussion of Naming

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).

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.

Discussion of Naming

"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.

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.

Discussion of Naming

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). The present definition generalizes that usage to multiple calendars in a very natural way, however. Furthermore, the meaning is intuitive in that the calendric system interprets time values at the conceptual level (+E8).

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).

Previously Used Names

Clock.

Discussion of Naming

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).

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.

Previously Used Names

Base-line clock, time-segment clock.

Discussion of Naming

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.

4.22 Time-line Clock Granularity

Definition

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

Discussion of Naming

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

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.

Previously Used Names

Start, begin, commencement, origin, negative infinity.

Discussion of Naming

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

'Begin" appears to be more straightforward (+E8), but suffers from conflicting meanings because 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).

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.

Previously Used Names

Infinity, positive infinity.

Discussion of Naming

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).

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.

Previously Used Names

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

Discussion of Naming

The arguments against "start," "begin," "commencement," "origin," and "negative infinity" are as in the discussion of beginning. "Birth" is an alternative that has been used by some authors.

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

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.

Discussion of Naming

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

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."

Previously Used Names
Time granularity.

Discussion of Naming
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.

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.

Discussion of Naming
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).

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 (+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.

Previously Used Names
Temporal assignment.

Discussion of Naming
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.

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.

Previously Used Names
Natural time-join, time-equijoin.

Discussion of Naming
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).

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 straight-forward 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.

Previously Used Names

Independence, dependence.

Discussion of Naming

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.

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.

Previously Used Names

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

Discussion of Naming

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).

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.

Previously Used Names

Homogeneous.

Discussion of Naming

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

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.

Previously Used Names

Temporal restriction.

Discussion of Naming

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).

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.

Previously Used Names

Restricted bitemporal relationship.

Discussion of Naming

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

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.

Discussion of Naming

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

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.

Previously Used Names

Proactive bitemporal relation.

Discussion of Naming

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).

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.

Discussion of Naming

"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.

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.

Previously Used Names

Fuzzy time, time imprecision, time incompleteness.

Discussion of Naming

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.

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 (+R1,+R3).

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.

Previously Used Names

Valid-time grouping.

Discussion of Naming

"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.

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.

Previously Used Names

Moving window.

Discussion of Naming

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.

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.

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.

Previously Used Names

Moving window.

Discussion of Naming

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.

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.

Previously Used Names

Moving window.

Discussion of Naming

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).

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.

Discussion of Naming

The term *instantaneous* is appropriate because the aggregate is applied over every chronon, every instant. It suggests an interest in the aggregate value over a very small time interval, 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).

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).

Previously Used Names

Process.

Discussion of Naming

"Process" is an over-loaded term, that is, a term having quite different meanings in different contexts (−E9).

Examples of macro-events are baking a cake, having a dinner party, flying from Rome to Paris.

It is worth remarking the distinction between macro-events and interval relations. Saying that a macro-event relates to the structure of an interval as whole means that if it consumes a certain interval it cannot possibly transpire during any subinterval thereof.

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