

# Glossary for *Time and Time Again*.

Tom Johnston and Randy Weis.

Revised October 20, 2008.

## **Table of Contents.**

Glossary for <i>Time and Time Again</i> .....	1
Table of Contents.....	1
On Reading This Glossary.....	4
Glossary of Technical Terms for the Asserted Versioning Approach to Bi-Temporal Data Management.....	5
12/31/9999 .....	5
adjacent versions.....	5
asserted version.....	5
asserted version table .....	6
asserted versioning.....	6
assertion .....	6
assertion begin date.....	6
assertion end date .....	6
assertion time .....	7
begin date .....	7
bi-temporal.....	7
bi-temporal date .....	7
bi-temporal table .....	8
business key .....	8
child episode .....	8
child row .....	8
child table.....	9
clock tick.....	9
closed-open .....	9
conventional table .....	10
current episode .....	10
current transaction.....	10
current version .....	11
currently asserted version .....	11
currently effective version .....	11
deferred assertion .....	11
directly follows .....	12
earliest episode.....	12
earliest version .....	12
effective begin date .....	12
effective end date .....	12
effective time period .....	13

effective time .....	13
end date .....	14
episode .....	14
episode begin date.....	14
episode end date .....	14
event .....	15
expand .....	15
flush.....	15
future version .....	16
granularity .....	16
hypothetical state .....	16
hypothetical state set .....	16
internalized batch transaction file .....	17
latest version .....	17
no longer asserted version.....	17
not yet asserted version.....	17
object.....	18
object identifier .....	18
open assertion.....	18
open current transaction .....	19
open episode.....	19
open transaction .....	19
open version .....	19
open-ended time period.....	19
original delete.....	20
original insert .....	20
original transaction.....	20
original update .....	20
optional temporal referential integrity .....	20
overlapping .....	21
parent episode .....	21
parent row .....	21
parent table.....	22
past assertion.....	22
past episode.....	22
past version .....	22
queryable history.....	22
relational table.....	23
required temporal referential integrity .....	23
row insertion date.....	23
seamless access .....	23
semantics .....	24
shrink.....	24
snapshot.....	24
squeeze .....	24

supercede.....	25
surrogate key.....	25
taxonomy.....	25
temporal delete transaction.....	26
temporal dimensions.....	26
temporal foreign key.....	26
temporal insert transaction.....	26
temporal referential integrity.....	27
temporal transaction.....	27
temporal update transaction.....	27
temporally adjacent.....	28
terminate.....	28
time period.....	28
transaction table.....	28
transaction time.....	29
TSQL2.....	29
until further notice.....	29
update in place.....	29
valid time.....	30
version.....	30
versioned row.....	30
versioned table.....	30
wholly contains.....	30
withdrawn.....	30
Thesaurus.....	31

### ***On Reading This Glossary.***

1. Grammatical variations of the same glossary term will not usually be distinguished. Thus both “version” and “versions” are used in the glossary, but only the former is a glossary entry. “supercede” is a glossary entry, but “superceded” and “superceding” are not.
2. Thesaurus entries are formulated as a binary (unordered) infix triple, with the semantic relationship coming between the two glossary entries being related. Each thesaurus entry relates two glossary entries as either synonyms or antonyms. Thus, the following entry means that the two terms are synonymous: effective date / syn / effective begin date.
3. Glossary entries used in the definition of an entry will be listed as “Components” under the entry they occur in.
4. Throughout this glossary and thesaurus, and the “Time and Time Again” series, when we refer to effective or assertion time period dates, we could also be talking about datetimes or timestamps. See the entry “clock tick” for additional information.
5. For the most part, discussions are about versions that are currently asserted. So to keep the verbiage to a minimum, we will leave out the “currently asserted” qualifier unless it seems needed for clarity. So, for example, instead of “currently asserted version”, we will normally use “version”; instead of “adjacent currently asserted versions”, we will normally use “adjacent versions”.

## ***Glossary of Technical Terms for the Asserted Versioning Approach to Bi-Temporal Data Management.***

### **12/31/9999**

A value whose semantics are that of until further notice, but which is treated by the DBMS as a valid date.

- 12/31/9999 is a valid value for an end date.
- 12/31/9999 is not a valid value for a begin date.

Components: semantics, until further notice, end date, begin date.

### **adjacent versions**

Two versions of the same object between whose effective time periods there is no clock tick.

- A closed-open representation of effective time periods is used. Thus, equivalently, a pair of versions such that the effective end date of one is equal to the effective begin date of the other.
- All versions in the same episode are adjacent versions.
- No versions in distinct episodes are adjacent versions.

Components: version, object, effective time period, clock tick, closed-open, effective end date, effective begin date.

### **asserted version**

A row in a bi-temporal table whose two temporal dimensions are effective time and assertion time.

- Effective time is equivalent to Jensen's valid time.
- Assertion time includes Jensen's transaction time as the special case in which the assertion begin date is also the row insertion date, but also permits a row to be inserted with an assertion begin date in the future.

Components: bi-temporal table, temporal dimension, effective time, assertion time, valid time, transaction time, assertion begin date, row insertion date.

### **asserted version table**

A bi-temporal table whose two temporal dimensions are effective time and assertion time.

Components: bi-temporal table, effective time, assertion time.

### **asserted versioning**

The implementation of bi-temporality developed by Johnston and Weis.

Components: bi-temporal.

### **assertion**

The claim that a row in a relational table is a true description of the object uniquely identified by its business key.

- When the table is an asserted version table, the claim is that the description of the identified object is true during the effective time period of that row.

Components: relational table, object, business key, asserted version table, effective time period.

### **assertion begin date**

The first date on which a version is asserted to be true.

- Every version makes an assertion. The assertion it makes is that, during the effective time of that row, its description of the object is correct.
- A version can never be inserted with an assertion begin date in the past, because an assertion is made by inserting a version. Thus, inserting a version with an assertion begin date in the past would be to make a false statement about what had or had not been asserted in that past period of time.
- A version can be inserted with an assertion begin date in the future.
- The closed/open convention for using pairs of dates to indicate time periods is used.

Components: version, assertion, effective time, object, effective begin date, versioned row, closed/open.

### **assertion end date**

The last date on which a version is asserted to be the truth.

- An assertion end date is always set to 12/31/9999 when its row is inserted. It retains that value unless and until that assertion is withdrawn.
- An assertion is withdrawn when and only when a new version is created with an overlapping (or identical) effective time period.
- The closed/open convention for using pairs of dates to indicate time periods is used. Thus, the assertion end date is actually one clock tick later than the clock tick on which the assertion is asserted to be the truth.

Components: version, 12/31/9999, withdrawn, overlapping, effective time period, closed/open, clock tick.

### **assertion time**

The temporal dimension which interprets a time period on a row as indicating when that row uniquely represented that object as it existed during the indicated effective time period.

- We do *not* say “The temporal dimension which interprets a time period on a row as indicating when that row was the unique row validly representing that object” because that suggests that this temporal dimension is a property of rows. It is not. It is a property of objects represented by rows.
- But assertion time periods *are* properties of rows or, more precisely, of the existentially quantified statements (assertions) made by those rows,

Components: temporal dimension, time period, object, effective time period.

### **begin date**

A date which is either an effective begin date or an assertion begin date.

Components: effective begin date, assertion begin date.

### **bi-temporal**

The association of a row in a relational table with both an effective time period and an assertion time period.

Components: effective time period, assertion time period.

### **bi-temporal date**

A date which is either a begin date or an end date.

Components: begin date, end date.

### **bi-temporal table**

A table whose rows are temporally delimited in two orthogonal temporal dimensions.

- Computer scientists call these two dimensions “valid time” and “transaction time”. See Jensen’s definition.
- We call the two dimensions of asserted versioning effective time and assertion time. Effective time is the same as valid time, but assertion time is not the same as transaction time.

Components: temporal dimension, valid time, transaction time, effective time, assertion time.

### **business key**

A key which uniquely identifies an object by means of one or more values which the business uses to refer to or uniquely describe each object of interest.

Components: object.

### **child episode**

An episode in an asserted version table Y is a child to an episode in asserted version table X if and only if the episode in Y has a temporal foreign key whose value is identical to the value of the object identifier of that episode in X, and the effective time period of that episode in X wholly contains the effective time period of that episode in Y. An episode in Y is a child to a row in conventional table X if and only if the episode in Y has a foreign key that points to a row in X that was inserted prior to the effective begin date of the episode in Y, and that was deleted, if indeed that occurred, after the episode end date of that episode in Y.

Components: episode, asserted version table, temporal foreign key, object identifier, effective time period, conventional table, foreign key, effective begin date, episode end date.

### **child row**

A row in a child table. A row in Y is a child to a row in conventional table X if and only if the row in Y has a foreign key whose value is identical to the primary key value of that row in X.

- Rows in asserted versioned tables are never child rows. Children, in asserted versioned tables are always episodes.

Components: child table, conventional table, foreign key, asserted version table, episode.

### **child table**

Y is a child table to a conventional table X if and only if there is a foreign key dependency from Y to X. Y is a child table to an asserted version table X if and only if there is a temporal foreign key dependency from Y to X.

- child tables are also sometimes referred to as “dependent tables” or “RI-dependent tables”.

Components: conventional table, foreign key, asserted version table.

### **clock tick**

The transition from one point in time to the next point in time, according to the clock which defines the granularity for a set of bi-temporal dates.

- The clock tick granularity used throughout these articles is a date. Thus, one clock tick is the transition from one day to the next day. In many real-world situations, the clock tick will be an hour, a minute, a second or even a full timestamp.
- A calendar day clock represents a situation in which a database is updated from a transactional batch file, once a night. So it isn't at all an unrealistic example of the temporal granularity used in many real-world applications.

Components: granularity, bi-temporal date.

### **closed-open**

A convention for using a pair of dates to define a time period, in which the begin date is included in the time period, and the end date is the first clock tick after the end of the time period.

- This definition applies to other granularities of time as well, such as date-times and timestamps.

Components: time period, begin date, end date, clock tick, granularity.

## **conventional table**

A table whose rows represent objects.

- A table in which each instance of the type represented by the table is represented by one and only one row in that table.
- In an OLTP database, assuming that there are no versioned tables in the database, object tables are all the tables which are not transaction tables.
- Roughly speaking, object tables are the tables which are the concern of Master Data Management.
- For example, a Client table in which each client is represented by one and only one row, is a conventional table. A table of customer status codes is a conventional table.

Components: versioned table, object table, transaction table.

## **current episode**

An episode whose current version has begin dates in the past, and end dates in the future. (Both effective dates and assertion dates are included in this definition.)

- A past episode is not a current episode because its effective period is past.
- A no longer asserted episode is not a current episode because its assertion period is past.
- There are no future episodes, although there may be any number of future versions. The intuition here is that an episode records the evolution of an object over time, and objects do not evolve in the future.
- The latest version of a current episode is the current version for that object.
- An object can have at most one current episode, and thus at most one current version.

Components: episode, current version, begin date, end date, effective date, assertion date, past episode, no longer asserted, future episode, future version, latest version, object.

## **current transaction**

An original transaction which accepts the date the transaction is submitted as the effective begin date of the version it will create.

Components: original transaction, effective begin date, version.

### **current version**

A version whose effective begin date is in the past, and whose effective end date is either 12/31/9999 or is a future date, and also whose assertion begin date is in the past, and whose assertion end date is 12/31/9999.

- This term should always be understood, unless otherwise indicated, as shorthand for "currently effective and currently asserted version".
- Current versions are always the latest versions of current episodes.

Components: version, effective begin date, effective end date, 12/31/9999, assertion begin date, assertion end date, current episode.

### **currently asserted version**

A version which has an assertion begin date in the past, and an assertion end date of 12/31/9999.

- assertion end dates are initially assigned the value 12/31/9999, and the only time that value is changed is when it is updated in place and set to the date on which the assertion is terminated.
- The only reason for terminating an assertion is when it is superceded by a later assertion. There are two reasons for superceding an assertion.
  - One is because a versioned change has occurred. Thus, the original assertion which is being superceded asserted, when created, that it would be asserted until further notice, i.e. until 12/31/9999. Well, creating a superceding version *is* "further notice".
  - The other is because a version is discovered to be in error.

Components: version, assertion begin date, assertion end date, 12/31/9999, update in place, terminate, supercede, versioned change, until further notice.

### **currently effective version**

A version which has an effective begin date in the past, and an effective end date of 12/31/9999 or a future date.

- effective end dates are initially assigned either the value 12/31/9999, or a future date.

Components: version, effective begin date, effective end date, 12/31/9999.

### **deferred assertion**

A row in an asserted version table whose assertion begin date is in the future.

Components: asserted version table, assertion begin date.

### **directly follows**

One version of an object directly follows another version of that object if and only if there is no version of that object whose effective begin date is greater than the effective begin date of the latter version and less than the effective begin date of the former version.

- Note that if version Y directly follows version X, the two versions may or may not be adjacent.

Components: version, object, effective begin date, adjacent.

### **earliest episode**

The episode of an object with the chronologically earliest episode begin date of all episodes of that object.

Components: episode, object, episode begin date.

### **earliest version**

The version of an episode of an object with the chronologically earliest effective begin date of all versions of that episode.

Components: version, episode, object, effective begin date.

### **effective begin date**

The begin date of the effective time period of a version or of an episode.

- The effective begin date of an episode is the effective begin date of its earliest version.

Components: effective time period, version, episode, earliest version.

### **effective end date**

A date one clock tick past the end date of the effective time period of a version or an episode.

- An effective end date must always be later than its corresponding effective begin date.
- The effective end date of an episode is the effective end date of its latest version.

Components: clock tick, end date, effective time period, version, episode.

### **effective time period**

The period of time for which an asserted version is believed to be the truth about an object.

- An effective time period is defined by an effective begin date and an effective end date, begins on its effective begin date and ends one day (one clock tick) prior to its effective end date.
- For a version, the period of time for which we believe that version is the truth about an object. Note that “for which”, in this definition, and “during which” in the definition of assertion time period do not mean the same thing. “for which” describes the asserted applicability of the version to the object it represents. “During which” describes the period of time of the assertion itself.
- For an episode of an object, the period of time during which our database will report that the object exists. Equivalently, the period of time which begins on the effective begin date of the earliest version of that episode, and ends one clock tick before the effective end date of the latest version of that episode.

Components: asserted version, object, effective begin date, effective end date, clock tick, version, assertion time period, assertion, episode, latest version.

### **effective time**

The temporal dimension which interprets a time period on a row as indicating when the object represented by that row existed such that the row was the unique row validly representing that object.

- We do *not* say “The temporal dimension which interprets a time period on a row as indicating when that row was the unique row validly representing that object” because that suggests that this temporal dimension is a property of rows. It is not. It is a property of objects represented by rows.
- But assertion time periods *are* properties of rows or, more precisely, of the existentially quantified statements (assertions) made by those rows,

Components: temporal dimension, time period, object, assertion time period.

## **end date**

A date on an asserted version row which marks the end of either an effective time period or an assertion time period.

- In both cases, the end of the time period is one clock tick prior to the end date.

Components: asserted version, effective time period, assertion time period, clock tick, end date.

## **episode**

A series or one or more adjacent versions of an object in which the earliest version is either the first version of that object, or a version which is not temporally adjacent to the version it directly follows.

- In the first case, it is the earliest version of the earliest episode of the object.
- In the second case, it is the earliest version of a non-earliest episode.
- See “supercede”.
- Thus, the first version of an object creates the earliest episode of that object. That episode remains the current episode until it is terminated by means of a temporal delete transaction, or until the passage of time reaches the effective end date of its latest version.
- If, when a version Y, which is not a future version, is inserted into a table, and the latest version of that object has an effective end date in the past, then Y begins a non-earliest episode of that object. That new episode remains the current episode until a temporal delete is again applied to that object, or until the passage of time reaches its effective end date.

Components: adjacent versions, object, earliest version, first version, temporally adjacent, earliest episode, supercede, current episode, terminate, temporal delete transaction, effective end date, latest version, future version.

## **episode begin date**

The effective begin date of the episode, which is the same as the effective begin date of its first version.

Components: effective begin date, episode, first version.

## **episode end date**

The effective end date of the episode, which is the same as the effective end date of its last version.

Components: effective end date, episode, last version.

## **event**

A point or duration of time at or during which one or more objects were affected so as to change one or more of their properties or relationships.

Components: point in time, duration of time, object.

## **expand**

To increase the effective time period of an episode or a version.

For an episode, via adding a version:

- This can be done by inserting an adjacent version whose effective end date is equal to the effective begin date of the earliest version of that episode.
- This can be done by inserting an adjacent version whose effective begin date is equal to the effective end date of the latest version of that episode.

For an episode, via superceding a version:

- This can be done by either moving the effective begin date back, or by moving the effective end date forward.
- Normally, an effective begin date cannot be altered. One exception is to correct an erroneous effective begin date. Another exception is to "make room" for a new version whose effective time period would otherwise overlap with the version.
- See "shrink".
- A version with a 12/31/9999 effective end date is always the most recent version of its episode. So if a version with a 12/31/9999 effective end date is superceded by a version with a non-12/31/9999 effective end date, we do not know if that supercession expanded the effective time period of the episode affected from its (real but unknown) effective end date, or shrank the episode's effective time period. But the only implications are for temporal referential integrity, so if we always trigger temporal referential integrity checking when a 12/31/9999 date is superceded, we can guarantee to avoid temporal referential integrity violations.

Components: effective time period, episode, version, adjacent version, effective end date, effective begin date, earliest version, latest version, supercede, shrink, 12/31/9999, most recent version, temporal referential integrity.

## **flush**

To delete all future versions of an object as part of deleting a current episode.

- To the user, this will seem to be removing future transactions against an object from the virtual batch transaction file.
- An original delete of a versioned object terminates the current episode of that object, and is the temporal equivalent of deleting the object from a conventional table. Whenever this happens, all future versions of that object must also be deleted because otherwise, the mere passage of time could make those objects re-appear in the database.

Components: future version, object, delete, current version, future transaction, virtual batch transaction file, original delete, terminate, current episode, conventional table.

### **future version**

A version of an object whose effective begin date is in the future.

Components: version, object, effective begin date.

### **granularity**

note the difference with Jensen, who says granularity is what the DBMS can represent. We say that it is, among all the granularities the DBMS can represent, the one actually being used. We leave the complexities of translating between different granularities to a future commercial implementation of bi-temporality. For our implementation, we will use a timestamp for all asserted version tables.

Components: xxx.

### **hypothetical state**

A situation in the world which we pretend is true for the sake of examining the consequences that would ensue if it really were true.

- Thus, a state of affairs which we hypothesize is true.

Components: this term has no components which are other Glossary entries. In terms of formalizing this Glossary, it is a primitive predicate.

### **hypothetical state set**

A set of one or more rows in an asserted version table all of which have the same assertion begin date and assertion end date and, therefore, the same assertion time period.

- Multiple hypothetical state sets may have overlapping assertion time periods.
- 

Components: asserted version table, assertion begin date, assertion end date, assertion time period.

### **internalized batch transaction file**

A conceptual queue into which future-dated transactions are placed and from which those transactions appear to be automatically applied to the database on their assertion begin date.

- As of October, 2008 this replaces the term "virtual batch transaction file".
- Note that future-dated transactions are those with a future assertion begin date, regardless of the effective begin and/or end dates on the transaction.
- Rows in an asserted version table with an effective begin date in the future are rows describing the future state of the object.
- Rows in an asserted version table with an assertion begin date in the future are the physical implementation of not-yet-applied transactions.

Components: assertion begin date, effective begin date, effective end date.

### **latest version**

The version of an episode with the chronologically latest effective end date.

- Every episode has a latest version.
- There may be future versions with effectivity dates chronologically later than any episode's latest version.

Components: version, episode, effective end date.

### **no longer asserted version**

A version whose assertion end date is in the past.

Components: version, assertion end date.

### **not yet asserted version**

A version whose assertion begin date is in the future.

Components: version, assertion begin date.

## **object**

Something which exists over time and can change over time.

- Persistent objects may be abstract or concrete.
- Events, whether points in time or durations in time, are not persistent objects, because events, by definition, do not change.
- Persistent objects change by participating in events.
- All versions are timeslices of persistent objects.
- All versioned tables represent types of persistent objects.
- All conventional tables, other than transaction tables, represent types of persistent objects.
- Examples of persistent objects include vendors, customers, employees, regulatory agencies, products, services, bills of material, invoices, purchase orders, claims, certifications, etc.
- In an OLAP, star-schema database, dimension tables are tables of persistent objects. Indeed, they are precisely those objects which change over time because of their participation in the events which change them, events each of which is represented by a transaction in the star-schema fact table.

Components: event, point in time, duration in time, version, timeslice, persistent object, versioned table, conventional table, transaction table.

## **object identifier**

The unique identifier of an object.

- Note that the unique identifier of an asserted version is the concatenation of (a) an object identifier; (b) the version's effective begin date; and (c) the version's assertion begin date.

Components: object, asserted version, effective begin date, assertion begin date.

## **open assertion**

A version whose assertion end date is 12/31/9999.

Components: version, assertion end date, 12/31/9999.

### **open current transaction**

An original transaction which, if valid, will create a version whose effective begin date is the date current when the transaction is applied, and whose effective end date is 12/31/9999. A transaction which is both a current transaction and an open transaction.

Components: original transaction, version, effective begin date, effective end date, 12/31/9999, current transaction, open transaction.

### **open episode**

An episode whose effective end date is 12/31/9999.

- Note that an effective time period with an end date in the future is not an open-ended time period, unless that future effective end date is 12/31/9999.

Components: episode, effective end date, effective time period, 12/31/9999, open-ended.

### **open transaction**

An original transaction which will create a version whose effective end date is 12/31/9999.

Components: version, effective end date, 12/31/9999, effective time period, open-ended.

### **open version**

A version whose effective end date is 12/31/9999.

- Note that an effective time period with an end date in the future is not an open-ended time period, unless that future effective end date is 12/31/9999.

Components: version, effective end date, 12/31/9999, effective time period, open-ended.

### **open-ended time period**

A time period whose end date is 12/31/9999.

- Note that an effective time period with an end date in the future is not an open-ended time period, unless that future effective end date is 12/31/9999.

Components: time period, end date, 12/31/9999.

### **original delete**

- A delete transaction against an episode, but written as though its target is a conventional table, not an asserted versioned table.

Components: delete transaction, episode, conventional table, asserted version table.

### **original insert**

An insert transaction for an episode, but written as though its target is a conventional table, not an asserted versioned table.

Components: episode, conventional table, asserted version table.

### **original transaction**

A transaction for an episode, but written as though its target is a conventional table, not an asserted versioned table. An original insert, update or delete.

Components: original insert, original update, original delete.

### **original update**

An update transaction against an episode, but written as though its target is a conventional table, not an asserted version table.

Components: episode, conventional table, asserted version table.

### **optional temporal referential integrity**

A relationship from a child table to a parent table which is implemented by a nullable temporal foreign key.

Components: child table, parent table, temporal foreign key.

## **overlapping**

Two time periods, along the same temporal dimension, are overlapping if and only if they are on different asserted version rows for the same object, and have at least one clock tick in common.

- Note that two time periods, along the same temporal dimension, where the end date of one of them has the same value as the begin date of the other one, are not overlapping. Because of the closed-open convention, they are adjacent.

Components: time period, temporal dimension, asserted version rows, object, clock tick, end date, begin date, closed-open, adjacent.

## **parent episode**

An episode in X is a parent to an episode in asserted version table Y if and only if the episode in Y has a TFK whose value is identical to the object identifier of that episode in X, and the effective time period of that episode in X wholly contains the effective time period of that episode in Y. An episode in X is a parent to a row in conventional table Y if and only if the row in Y has a TFK whose value is identical to the object identifier of that episode in X, the effective begin date of that episode in X is earlier than the date of insertion of that row in Y, and the effective end date of that episode occurs, if at all, only after the deletion of that row in Y occurs, if at all.

Components: episode, parent, asserted version table, TFK, object identifier, effective time period, wholly contains, conventional table, effective begin date, effective end date.

## **parent row**

A row in a conventional parent table. A row in X is a parent to a row in conventional table Y if and only if the row in Y has a foreign key whose value is identical to the primary key value of that row in X. A row in X is a parent to an episode in asserted version table Y if and only if the episode in Y has a foreign key whose value is identical to the primary key value of that row in X, that row in X was inserted prior to the effective begin date of that episode, and is not deleted, if ever, until after the effective end date of that episode.

Components: conventional table, parent table, foreign key, primary key, episode, asserted version table, effective begin date, effective end date.

## **parent table**

A conventional table X is a parent to a table Y if and only if there is a foreign key dependency from Y to X. An asserted version table X is a parent to a table Y if and only if there is a TFK dependency from Y to X.

Components: conventional table, foreign key dependency, asserted version table, TFK dependency.

## **past assertion**

An assertion of a version whose assertion end date is not 12/31/9999.

- The semantics of versioning do not permit assertion end dates to be future dates. An assertion is initially created with a 12/31/9999 end date and, if changed, can only be changed from that date to the date current at the time of the change.

Components: assertion, version, assertion end date, 12/31/9999, semantics, end date.

## **past episode**

An episode of an object whose latest version has an effective end date which is in the past.

Components: episode, object, latest version, effective end date.

## **past version**

A version of an object whose effective end date is in the past.

Components: version, object, effective end date.

## **queryable history**

Data about an object which was valid at some time in the past, which is no longer currently valid, but which is as easily and rapidly accessible as current data about that object.

- “As easily and rapidly accessible as current data” means what it says. Our way of providing this access is to use asserted version tables. In such tables, production queries against current data (the most common kind of query) can be used to retrieve historical data simply by adding a date to a pair of BETWEEN clauses of a SQL statement (one for the effective time period and one for the assertion time

- period that would, without that addition, retrieve the corresponding current data from a conventional table (or from that asserted version table).
- Providing queryable history, in this manner, can significantly lower the development and operations cost of accessing historical data, and significantly improve the currency of the historical data retrieved.

Components: object, asserted version table, effective time period, assertion time period, conventional table.

### **relational table**

A table in a relational database whose rows have business keys that are guaranteed to be unique identifiers of instances of the type of object represented by the table.

- Multi-sets, also called “bags”, are not relational tables.
- Tables which have a surrogate key unique identifier, but no business key, are not relational tables. Note that this definition defines “relational table” in terms of semantics, regardless of the presence or absence of a primary key.
- Tables whose business keys cannot be guaranteed to be free of homonyms or synonyms are not relational tables.

Components: business key, unique identifier, object, surrogate key, semantics, homonym, synonym.

### **required temporal referential integrity**

A relationship from a child table to a parent table which is implemented by a non-nullable temporal foreign key.

Components: child table, parent table, temporal foreign key.

### **row insertion date**

The date on which a row was physically inserted into a table.

Components: none.

### **seamless access**

Something that can change over time.

Components: none.

## **semantics**

Components: xxxxxx

### **shrink**

To decrease the effective time period of a version.

- This can be done by either moving the effective begin date forward, or by moving the effective end date back.
- Semantically, changing a 12/31/9999 end date to some other date is the process of replacing an unknown end date with a known date. As a semantic change, this could expand the affected time period just as easily as shrink it. To the DBMS, however, since any date is earlier than 12/31/9999, the process is always one in which the time period is shrunk.
- TRI checks will always interpret a change from 12/31/9999 to another end date as potentially shrinking the time period of the affected object. The effect is that when this happens to a parent object in a temporal referential integrity relationship, this change will always trigger a temporal referential integrity check
- The (time) periods referred to here are either effective time periods or assertion time periods.
- See "expand".

Components: effective time period, current version, effective begin date, effective end date, semantics, 12/31/9999, end date, time period, temporal referential integrity, object, time period, assertion time period, expand.

### **snapshot**

Something that can change over time.

Components: none.

### **squeeze**

To create a version Z whose effective time period overlaps that of the currently asserted effective time period of an already asserted version X, or of a pair of already asserted adjacent versions X and Y.

- In the former case, the effective begin date of X must be moved forward to "make room" for Z, or else the effective end date of X must be moved back to make room for Z.
- In the latter case, the effective end date of X must be moved back, and the effective begin date of Y must be moved forward.

Components: version, effective time period, overlap, currently asserted, adjacent versions, effective begin date, effective end date.

## **supercede**

To replace a current version of an episode with a new current version of that same episode.

- Supersession is a *logical* function. Physically, supersession is done by inserting a new row in a versioned table.
- Deletion in a versioned table is always done via supersession.
- Temporal updates, in a versioned table, are always done via supersession.
- Creating the first version of an object does *not* involve supersession, as there is no current version of that object to supersede.
- Creating the first version of an episode of an object also does not involve supersession. Even if other versions of an object exist, the last version of every non-current episode is a version whose effective end date is in the past. Thus, when a new episode is started, the latest prior version of that object is not a current version. Thus, there is no current version of the object to supersede when that new episode begins.

Components: current version, episode, temporal update, object, version, effective end date.

## **surrogate key**

A system-generated key which, because it contains no component that the business uses to describe or identify an object, can be relied upon to never change.

Components: object.

## **taxonomy**

### **temporal delete transaction**

The result of translating an original delete into a physical transaction that supercedes the current version of the object with a terminated version, and that also flushes all transactions for that object from the virtual batch transaction file.

- Just as a conventional delete is an invalid transaction if the row representing the object in question is not on the database, a temporal delete is an invalid transaction if there is no current episode of the object in question on the database.

Components: original delete, supercede, current version, object, terminated version, flush, virtual batch transaction file, current episode.

### **temporal dimensions**

The semantics common to all instances of time periods of the same type.

- Thus, for example, there may be an instance of the time period 3/5/04 – 10/18/09 in both effective time and assertion time. These two dimensions determine the distinct semantics of these two instances.

Components: semantics, time period.

### **temporal foreign key**

A column which contains the object identifier used by one or more rows in an asserted version table.

Components: object identifier, asserted version table.

### **temporal insert transaction**

The result of translating an original insert into a physical transaction against an asserted version table.

- Just as a normal insert is an invalid transaction if the row representing the object in question is already on the database, a temporal insert is an invalid transaction if a currently asserted episode of the object in question, with an effective time period which overlaps that of the transaction, is already on the database.
- These temporal inserts are always physical inserts of a new (single-) version episode of the object in question.
- Note that temporal inserts may be of past, present or future episodes.

Components: original insert, asserted version table, object, currently asserted episode, effective time period, overlaps, past episode, present episode, future episode.

### **temporal referential integrity**

If there is a temporal referential integrity dependency from asserted version table Y to asserted version table X (not necessarily distinct), then no exposed state of the database is valid in which any episode in Y is not linked by a TFK to an episode in X whose effective time period wholly contains the effective time period of the episode in Y. If there is a temporal referential integrity dependency from conventional table Y to asserted version table X, then no exposed state of the database is valid in which any row in Y is not linked by a TFK to an episode in X whose effective begin date is not later than the date that row was inserted into Y, and whose effective end date is not earlier than the date that row was deleted from Y.

Components: asserted version table, episode, TFK, effective time period, wholly contains, conventional table, effective end date.

### **temporal transaction**

The result of translating an original transaction into one or more physical transactions against a versioned table.

- The identified types of temporal transactions are temporal insert, temporal update, temporal delete and temporal upsert.

Components: original transaction, versioned table.

### **temporal update transaction**

The result of translating an original update transaction into one or more transactions against an asserted version table.

- In these discussions, we have assumed that all updates to asserted version tables will cause a new version to be physically inserted. But in reality, the business is likely to want to track changes to only selected columns of a table. For the other columns, an original update can be applied as an update in place, and does not require creation of a superceding version.
- These temporal updates to versionable columns are always physical inserts of a new current version that supercedes the version current at the time the transaction began.

- In addition, if there are any future versions of that object, they are also updated in the same manner. Each one is superceded by a new version based on the version it is replacing, updated with the temporal update.
- There the effective end date of the superceded version will be changed from 12/31/9999 to the same value as the effective begin date of the superceding version.

Components: original update transaction, asserted version table, version, supercede, current version, future version, effective end date, 12/31/9999, effective begin date.

## **temporally adjacent**

Components: xxxxx.

## **terminate**

To end an episode by superceding its current version with a terminated version.

Components: episode, supercede, current version, terminated version.

## **time period**

A continuous length of time with a known begin date.

- If the end date of a time period is not known, 12/31/9999 is used as the end date. Because of its interpretation as a valid date by the DBMS, the effective semantics is "until further notice".
- In asserted versioning, there are two types of time periods: effective time periods and assertion time periods.
- Our convention is that a time period begins on its begin date, but ends one clock tick prior to its end date.

Components: begin date, end date, 12/31/9999, semantics, until further notice, asserted versioning, effective time period, assertion time period, clock tick.

## **transaction table**

Components: xxxxx.

## **transaction time**

A database fact is stored in a database at some point in time, and after it is stored, it may be retrieved. The transaction time of a database fact is the time when the fact is stored in the database. Transaction times are consistent with the serialization order of the transactions. Transaction time values cannot be after the current time. Also, as it is impossible to change the past, transaction times cannot be changed. Transaction times may be implemented using transaction commit times. [Jensen, 1992]

## **TSQL2**

Components: xxxxx.

## **until further notice**

In general, this means "unknown but presumed valid". It is the meaning of 12/31/9999, that the end date it appears in is unknown but assumed to be later than {now}, whatever value {now} has.

Components: 12/31/9999, end date, {now}.

## **update in place**

An update which overwrites the row it changes.

- With respect to business data columns, updates in place are used for all and only those columns that do not require versioning.
- With respect to columns implementing asserted versioning, there are only two such columns that may be updated in place. They are (a) the assertion end date, which is changed from 12/31/9999 to the assertion begin date of the version which is superceding its version, and at no other time; and (b) the (denormalized) episode begin date which is changed whenever an episode is extended backwards in effective time.

Components: asserted versioning, assertion end date, 12/31/9999, assertion begin date, version, supercede, episode begin date, episode, effective time.

**valid time**

The valid time of a fact is the time when the fact is true in the modeled reality. A fact may have associated any number of events and intervals, with single events and intervals being important special cases. [Jensen, 1992]

**version**

A row in a versioned table which represents the state of an object during a specified period of time.

Components: object, time period

**versioned row**

Components: xxxxx

**versioned table**

A table whose rows represent versions of objects.

- A version of an object is a timeslice of that object.
- A row in a versioned table describes that object as it was, is, will be, or might be, during its effective time period.

Components: version, object, timeslice, effective time period.

**wholly contains**

Time period 1 wholly contains time period 2 if and only if the begin date of time period 1 is earlier than or equal to the begin date of time period 2, the end date of time period 1 is later than or equal to the end date of time period 1, and there are no gaps between those dates.

Components: time period, begin date, end date, gap.

**withdrawn**

Components: xxxxx.

## **Thesaurus.**

This thesaurus is a list of synonyms and antonyms. A synonym X of a term Y may be substituted for Y in any context in which Y occurs.<sup>1</sup> The negation of an antonym Z of Y (NOT-Z) may also be substituted for Y in any context in which Y occurs.

A list of synonyms and antonyms helps we human beings understand what we mean when we use these expressions. But eventually, when this complete corpus of work about asserted versioning is completely formalized, it will help software inferencing engines derive new true assertions from existing ones, thus extending our understanding of what we mean by what we say.

child table	syn	dependent table
conventional table	syn	non-temporal table
conventional table	syn	object table
earliest version	syn	first version
object	syn	persistent object
temporal foreign key	syn	TFK
future transaction	syn	future assertion

---

<sup>1</sup> Actually, extensive work in philosophy of language has shown that synonymous terms cannot be freely substituted within "propositional attitude" contexts, with the guarantee that the truth value of the assertions in which the substitutions take place will not be altered. To begin research on this topic, one should start with Frege's discussion of the morning star and evening star example. For it may be true that someone believes that the evening star is Venus but false that the same person believes that the morning star is Venus, because that person may not know that the evening star is the morning star.