The Asserted Versioning Glossary

This Glossary contains approximately three-hundred definitions, nearly all of which are specific to Asserted Versioning. Most expressions have both a *Mechanics* entry and a *Semantics* entry. A Mechanics entry describes how the defined concept is implemented in the "machinery" of Asserted Versioning. A Semantics entry describes what that concept means. We can also think of a Mechanics entry as telling us what a component of Asserted Versioning is or what it does, and a Semantics entry as telling us why it is important.

In linguistics, the usual contrast to semantics is syntax. But syntax is only the "parts list" of Asserted Versioning. The Asserted Versioning Framework, or any other implementation of Asserted Versioning, has an intricately interconnected set of parts, which correspond to the syntax of a language. But when it is turned on, it is a software engine which translates metadata and data models into the database schemas it uses to do its work, transforms the data instances it manages from one state to another state, augments or diminishes the totality of the representation of the objects its data corresponds to, and facilitates the ultimate purpose of this wealth of activity, which is to provide meaningful information about the time-varying state of the world an enterprise is a part of and needs to remain cognizant of.

Grammar

Grammatical variations of the same glossary term will not usually be distinguished. Thus both “version” and “versions” are in this book, but only the former is a Glossary entry. "Currently asserted" is listed as a component of one or more definitions, but the corresponding Glossary entry is "current assertion".

Dates and Times

All references to points in time in this Glossary, unless otherwise noted, refer to them using the word "date". This is done for the same reason that all examples of points in time in the text, unless otherwise noted, are dates. This reason is simply convenience. Periods of time in either of the two bi-temporal dimensions are delimited by their starting point in time and ending point in time. These points in time may be timestamps, dates, or any other point in time recognizable by the DBMS. As defined in this Glossary, they are clock ticks.

Components

Components of a definition are other Glossary entries used in the definition. Listing the components of every definition separately makes it easier to pick them out and follow cross-reference trails.

The Components section of these definitions are also working notes towards a formal ontology of temporal data. If we assume first-order predicate logic as an initial formalization, we can think of the components of a Glossary definition, together with a set of primitive (formally undefined) terms, as the predicates with which the Mechanics and Semantics sections of those definitions can be expressed as statements in first-order predicate logic.

Thus formalized, automated inferencing and theorem proving mechanisms can then be used to discover new theorems. And the point of that activity, of course, is that it can make us aware of the deductive implications of things we already know, of statements we already recognize as true statements. These deductive implications are other true statements. But until we are aware of them, they are not part of our knowledge about the world. These mechanisms can also be used to prove or disprove conjectures about temporal data, thus adding some of them to the totality of that knowledge, and adding, for the rest of them, the knowledge that they are wrong.

Of particular note are those few Glossary entries whose list of components is empty (indicated by "N/A"). In an ontology, the collection of undefined terms is called a controlled vocabulary, and these Glossary entries with empty component lists are part of the controlled vocabulary for a formal ontology of Asserted Versioning.

Non-Standard Glossary Definitions

Broadly speaking, the semantics entry of a Glossary definition describes a concept, while the mechanics entry describes its implementation. However, in some cases, there doesn't seem to be a need for both kinds of entry, and so those definitions will have just a mechanics section, or just a semantics section. And in other cases, it seems more appropriate to provide a general description rather than to attempt a precise definition.

But the heart of this Glossary are the definitions which have both a semantics and a mechanics section. Together, the collection of their semantics entries is a summary statement of Asserted Versioning as a theory of bi-temporal data management, while the collection of their mechanics entries is a summary statement of the implementation of the theory in Asserted Versioning practice.

Allen Relationships

The original Allen relationships are leaf nodes in our Allen relationship taxonomy. Most of the Allen relationships, as well as our taxonomic groupings which are OR'd collections of those relationships, have an inverse. The inverse of an Allen relationship or relationship group, between two time periods which do not both begin and end on the same clock tick, is the relationship in which the two time periods are reversed. Following Allen's original notation, we use a superscript suffix (x^{-1}) to denote the inverse relationship. Inverse relationships exist in all cases where one of the two time periods is shorter than the other and/or begins on an earlier clock tick than the other. Consequently, all the Allen relationship except [equals], have an inverse.

"Trivial" Definitions

Some Glossary definitions may appear to be "trivial", in the sense that we can reliably infer what those expressions mean from the expressions themselves. For example, "end date" is defined as "an assertion end date or an effective end date".

Definitions like these exist because these expressions are used in the definitions of other expressions. So they are a kind of shorthand. But in addition, our ultimate objective, with this Glossary, is to formalize it as an ontology expressed in predicate logic. For that purpose, apparently trivial entries such as "end date" are needed as predicates formal definitions of, for example, the expression "assertion end date".

Glossary Entries

include

See *Allen relationship [fills^{-1}]*.

"assert" cognates

Mechanics: the cognate terms "accept", "agree", "assent", "believe", "claim", "know", "say" and "think".

Semantics: terms which, for purposes of the discussions in this book, may be taken as synonymous with "assert" as that word is defined in this book.

Comments:

- There are important differences among these terms, in the fields of epistemology and semantics. For example, some designate what philosophers call "speech acts", while others designate what philosophers call "propositional attitudes".

12/31/9999

Mechanics: the latest date which can be represented by the SQL Server DBMS.

Semantics: a value for an end date which means that the end of the time period it delimits is unknown but assumed to be later than Now().

Comments:

- For other DBMSs, the value used should similarly be the latest date which can be represented by that DBMS.

Components: end date, Now(), time period.

9999

Mechanics: a DBMS-agnostic representation of the latest date which can be represented by a specific DBMS.

Semantics: a DBMS-agnostic representation of a value for an end date which means that the end of the time period it delimits is unknown but assumed to be later than Now().

Components: end date, Now(), time period.

actionable

Description: data which is good enough for its intended purposes.

Comments:

- As a kind of shorthand, we say that the assertion time period of a row is the period of time during which we assert that it is true. And if we discover that a row is incorrect, and does not make a true statement, we do end its assertion time period.
- But some true statements are not actionable. For example, a currently effective row in a one-hundred column table may have ten of its columns filled with accurate data, and the other ninety columns empty. So that row makes a true statement "as far as it goes", but because it is so incomplete, it is probably not a statement that provides enough information to act on.
- And some actionable statements are not even true. Financial forecasts, for example, may be actionable. But because they are about the future, what they describe hasn't happened yet, and so they are statements which are neither true nor false.[1]

Components: currently asserted.

ad hoc query

Description: a query which is not embedded in an application program, and which is not run as part of the IT production schedule.

Comments:

- These queries are usually written by business researchers and analysts, and are often run only a few times before they are discarded. Thus the cost of writing them is amortized over only a few occasions on which they are used, and so it is important to keep the query-writing costs as low as possible. This is why we recommend that, as far as possible, ad hoc queries should be written against views.
- See also: *production query*.

Allen relationship taxonomy

Description: a taxonomy of Allen relationships, developed by the authors and presented in Chapter 3.

Comments:

- Our *Mechanics* definitions of the Allen relationships will express time periods as date pairs, using the closed-open convention. The two time periods will be designated P_1 and P_2, and the begin and end dates, respectively, $eff\_beg\_dt_1$ and $eff\_end\_dt_1$, and $eff\_beg\_dt_2$ and $eff\_end\_dt_2$.

[1] This, at least, is the standard interpretation of Aristotle's position on what are called "future contingents", as expressed in his work *De Interpretatione*.

- These definitions assume that the begin date value for a time period is less than the end date value for that time period. This assumption excludes non-sensical time periods that end before they begin. It also excludes *empty time periods*.
- Our *Semantics* definitions of the Allen relationships will be stated in terms of clock ticks contained or not contained in time periods, and so these definitions are independent of the convention chosen for using pairs of dates to delimit time periods. In particular, "begin", "end", "earlier", "later" and other terms refer to relationships in time, not to comparisons of begin and/or end dates to other begin and/or end dates.
- Boolean operators (AND, OR, NOT) are capitalized.

Allen relationship, [aligns]

Mechanics: P_1 and P_2 [align] if and only if
((eff_beg_dt_1 = eff_beg_dt_2) AND (eff_end_dt_1 < eff_end_dt_2))
OR ((eff_beg_dt_1 > eff_beg_dt_2) AND (eff_end_dt_1 = eff_end_dt_2))
AND NOT((eff_beg_dt_1 = eff_beg_dt_2) AND (eff_end_dt_1 = eff_end_dt_2)).

Semantics: P_1 and P_2 [align] if and only if they either start or end on the same clock tick, but not both.

Allen relationship, [before]

Mechanics: P_1 is [before] P_2 if and only if (eff_end_dt_1 < eff_beg_dt_2).

Semantics: P_1 is [before] P_2 if and only if the next clock tick after P_1 is earlier than the first clock tick in P_2.

Allen relationship, [before^{-1}]

Mechanics: P_1 is [before^{-1}] P_2 if and only if (eff_beg_dt_1 > eff_end_dt_2).

Semantics: P_1 is [before^{-1}] P_2 if and only if the first clock tick in P_1 is later than the next clock tick after P_2.

Allen relationship, [during]

Mechanics: P_1 is [during] P_2 if and only if (eff_beg_dt_1 > eff_beg_dt_2) AND (eff_end_dt_1 < eff_end_dt_2).

Semantics: P_1 is [during] P_2 if and only if the first clock tick in P_1 is later than the first clock tick P_2, and the last clock tick in P_1 is earlier than the last clock tick in P_2.

Allen relationship, [during^{-1}]

Mechanics: P_1 is [during^{-1}] P_2 if and only if ($eff\_beg\_dt_1 < eff\_beg\_dt_2$) AND ($eff\_end\_dt_1 > eff\_end\_dt_2$).

Semantics: P_1 is [during^{-1}] P_2 if and only if the first clock tick in P_1 is earlier than the first clock tick in P_2, and the last clock tick in P_1 is later than the last clock tick in P_2.

Allen relationship, [*equals*]

Mechanics: P_1 [equals] P_2 if and only if ($eff\_beg\_dt_1 = eff\_beg\_dt_2$) AND ($eff\_end\_dt_1 = eff\_end\_dt_2$).

Semantics: P_1 [equals] P_2 if and only if they both start and end on the same clock tick.

Allen relationship, [excludes]

Mechanics: P_1 [excludes] P_2 if and only if ($eff\_end\_dt_1 <= eff\_beg\_dt_2$).

Semantics: P_1 [excludes] P_2 if and only if the next clock tick after P_1 is no later than the first clock tick in P_2.

Allen relationship, [excludes^{-1}]

Mechanics: P_1 [excludes^{-1}] P_2 if and only if ($eff\_beg\_dt_1 >= eff\_end\_dt_2$).

Semantics: P_1 [excludes^{-1}] P_2 if and only if the first clock tick in P_1 is no earlier than the next clock tick after P_2.

Allen relationship, [fills]

Mechanics: P_1 [fills] P_2 if and only if ($eff\_beg\_dt_1 >= eff\_beg\_dt_2$) AND ($eff\_end\_dt_1 <= eff\_end\_dt_2$).

Semantics: P_1 [fills] P_2 if and only if the first clock tick in P_1 is no earlier than the first clock tick in P_2, and the last clock tick in P_1 is no later than the last clock tick in P_2.

Allen relationship, [fills^{-1}]

Mechanics: P_1 [fills] P_2 if and only if ($eff\_beg\_dt_1 <= eff\_beg\_dt_2$) AND ($eff\_end\_dt_1 >= eff\_end\_dt_2$).

Semantics: P_1 [fills] P_2 if and only if the first clock tick in P_1 is no later than the first clock tick in P_2, and the last clock tick in P_1 is no earlier than the last clock tick in P_2.

Allen relationship, [finishes]

Mechanics: P_1 [finishes] P_2 if and only if (eff_beg_dt$_1$ > eff_beg_dt$_2$) AND (eff_end_dt$_1$ = eff_end_dt$_2$).

Semantics: P_1 [finishes] P_2 if and only if the first clock tick in P_1 is later than the first clock tick in P_2, and the two time periods end on the same clock tick.

Allen relationship, [finishes^{-1}]

Mechanics: P_1 [finishes^{-1}] P_2 if and only if (eff_beg_dt$_1$ < eff_beg_dt$_2$) AND (eff_end_dt$_1$ = eff_end_dt$_2$).

Semantics: P_1 [finishes^{-1}] P_2 if and only if the first clock tick in P_1 is earlier than the first clock tick in P_2, and the two time periods end on the same clock tick.

Allen relationship, [intersects]

Mechanics: P_1 [intersects] P_2 if and only if (eff_beg_dt$_1$ <= eff_beg_dt$_2$) AND (eff_end_dt$_1$ > eff_beg_dt$_2$).

Semantics: P_1 [intersects] P_2 if and only if the first clock tick in P_1 is no later than the first clock tick in P_2, and the next clock tick after P_1 is later than the first clock tick in P_2.

Allen relationship, [intersects^{-1}]

Mechanics: P_1 [intersects^{-1}] P_2 if and only if (eff_beg_dt$_1$ >= eff_beg_dt$_2$) AND (eff_beg_dt$_1$ < eff_end_dt$_2$).

Semantics: P_1 [intersects^{-1}] P_2 if and only if the first clock tick in P_1 is no earlier than the first clock tick in P_2, and the first clock tick in P_1 is earlier than the next clock tick after P_2.

Allen relationship, [meets]

Mechanics: P_1 [meets] P_2 if and only if (eff_end_dt$_1$ = eff_beg_dt$_2$).

Semantics: P_1 [meets] P_2 if and only if the next clock tick after P_1 is the same as the first clock tick in P_2.

Allen relationship, [meets^{-1}]

Mechanics: P_1 [meets^{-1}] P_2 if and only if (eff_beg_dt$_1$ = eff_end_dt$_2$).

Semantics: P_1 [meets^{-1}] P_2 if and only if the first clock tick in P_1 is the same as the next clock tick after P_2.

Allen relationship, [occupies]

Mechanics: P_1 [occupies] P_2 if and only if
((eff_beg_dt$_1$ >= eff_beg_dt$_2$) AND (eff_end_dt$_1$ <= eff_end_dt$_2$)) AND
NOT((eff_beg_dt$_1$ = eff_beg_dt$_2$) AND (eff_end_dt$_1$ = eff_end_dt$_2$)).

Semantics: P_1 [occupies] P_2 if and only if the first clock tick in P_1 is no earlier than the first clock tick in P_2, and the last clock tick in P_1 is no later than the last clock tick in P_2, and P_1 and P_2 do not both begin and end on the same clock tick.

Allen relationship, [occupies^{-1}]

Mechanics: P_1 [occupies^{-1}] P_2 if and only if
((eff_beg_dt$_1$ <= eff_beg_dt$_2$) AND (eff_end_dt$_1$ >= eff_end_dt$_2$)) AND
NOT((eff_beg_dt$_1$ = eff_beg_dt$_2$) AND (eff_end_dt$_1$ = eff_end_dt$_2$)).

Semantics: P_1 [occupies^{-1}] P_2 if and only if the first clock tick in P_1 is no later than the first clock tick in P_2, and the last clock tick in P_1 is no earlier than the last clock tick in P_2, and P_1 and P_2 do not both begin and end on the same clock tick.

Allen relationship, [overlaps]

Mechanics: P_1 [overlaps] P_2 if and only if
(eff_beg_dt$_1$ < eff_beg_dt$_2$) AND
(eff_end_dt$_1$ > eff_beg_dt$_2$) AND
(eff_end_dt$_1$ < eff_end_dt$_2$).

Semantics: P_1 [overlaps] P_2 if and only if the first clock tick in P_1 is earlier than the first clock tick in P_2, and the next clock tick after P_1 is later than the first clock tick in P_2, and the last clock tick in P_1 is earlier than the last clock tick in P_2.

Allen relationship, [overlaps^{-1}]

Mechanics: P_1 [overlaps] P_2 if and only if
(eff_beg_dt$_1$ > eff_beg_dt$_2$) AND
(eff_beg_dt1 < eff_end_dt$_2$) AND
(eff_end_dt$_1$ > eff_end_dt$_2$).

Semantics: P_1 [overlaps^{-1}] P_2 if and only if the first clock tick in P_1 is later than the first clock tick in P_2, and the first clock tick in P_1 is earlier than the next clock tick after the end of P_2, and the last clock tick in P_1 is later than the last clock tick in P_2.

Allen relationship, [starts]

Mechanics: P_1 [starts] P_2 if and only if ($eff\_beg\_dt_1 = eff\_beg\_dt_2$) AND ($eff\_end\_dt_1 < eff\_end\_dt_2$).

Semantics: P_1 [starts] P_2 if and only if the two time periods start on the same clock tick, and the last clock tick in P_1 is earlier than the last clock tick in P_2.

Allen relationship, [starts^{-1}]

Mechanics: P_1 [starts^{-1}] P_2 if and only if ($eff\_beg\_dt_1 = eff\_beg\_dt_2$) AND ($eff\_end\_dt_1 > eff\_end\_dt_2$).
Semantics: P_1 [starts^{-1}] P_2 if and only if the two time periods start on the same clock tick, and the last clock tick in P_1 is later than the last clock tick in P_2.

Allen relationships

Mechanics: the set of thirteen positional relationships between two time periods, a time period and a point in time, or two points in time, as first defined in James F. Allen's 1983 article *Maintaining Knowledge about Temporal Intervals*.

Semantics: the set of all possible positional relationships between two time periods, a time period and a point in time, or two points in time, defined along a common timeline.

Comments:
- The Allen relationships are mutually exclusive and jointly exhaustive.
- Good discussions of the Allen relationships can also be found in (Snodgrass, 2000), From Chapter 4, and (Date, Darwen and Lorentzos, 2002), From Chapter 6.

Components: time period, point in time.

approval transaction

Mechanics: a transaction that changes the assertion begin date on the assertions in a deferred assertion group to an earlier date.

Semantics: a transaction that moves assertions in an deferred assertion group from far future assertion time to near future assertion time.

Comments:
- The transaction by which deferred assertions are moved close enough to Now() that the business is willing to let them become current by means of the passage of time. See *fall into currency*.

Components: assertion, assertion begin date, deferred assertion group, far future assertion time, near future assertion time.

as-is

Mechanics: data whose assertion begin date is earlier than Now() and whose assertion end date is later than Now().

Semantics: data whose assertion time period is current.

Comments:

- See *as-was*. The as-is vs. as-was distinction is often confused with the distinction between current and past versions. Many best practices implementations of versioning do not distinguish between the two, and therefore introduce ambiguities into their temporal semantics.

Components: assertion begin date, assertion end date, assertion time period, Now().

assert

Mechanics: to place a row in an asserted version table in current assertion time.

Semantics: to claim that a row in an asserted version table makes a true and/or actionable statement.

Components: actionable, asserted version table, current assertion, statement.

asserted version table

Mechanics: a bi-temporal table in which each row can exist in past, present or future assertion time, and also in past, present or future effective time.

Semantics: a table each of whose rows indicates when the object it represents is as its business data describes it, and when that row is claimed to make a true and/or actionable statement about that object.

Comments:

- In contrast, rows in bi-temporal tables of the standard temporal model cannot exist in future assertion time.
- Also, a table whose structure conforms to the schema presented in Chapter 6. See *bi-temporal data canonical form*.

Components: actionable, assertion time, bi-temporal table, business data, effective time, object, statement.

Asserted Versioning database

Mechanics: a database that contains at least one asserted version table.

Components: asserted version table.

Asserted Versioning Framework

Mechanics: software which (i) generates asserted version tables from logical data models and associated metadata; (ii) enforces temporal entity integrity and temporal referential integrity constraints as asserted version tables are maintained; (iii) translates temporal insert, update and delete transactions into the physical transactions which maintain an asserted version table; and (iv) internalizes pipeline datasets.

Comments:

- The Asserted Versioning Framework is software developed by the authors which implements Asserted Versioning.

Components: asserted version table, internalization of pipeline datasets, physical transaction, temporal data management {the Asserted Versioning temporal model}, temporal entity integrity, temporal referential integrity, temporal transaction.

assertion

Mechanics: the temporally delimited claim that a row in an asserted version table makes a true and/or actionable statement about what the object it represents is like during the time period designated by that version of that object.

Semantics: the claim that a statement is true and/or actionable.

Components: actionable, asserted version table, object, statement, time period, version.

assertion approval date

Mechanics: the new assertion begin date which an approval transaction specifies for the assertions in a deferred assertion group.

Semantics: the near future assertion time date to which all assertions in a deferred assertion group are to be retrograde moved.

Components: approval transaction, assertion, assertion begin date, deferred assertion group, near future assertion time, retrograde movement.

assertion begin date

Mechanics: the begin date of the assertion time period of a row in an asserted version table.

Semantics: the date indicating when a version begins to be asserted as a true and/or actionable statement of what its object is like during its indicated period of effective time.

Comments.

- A row can never be inserted with an assertion begin date in the past, because an assertion cannot exist prior to the row whose truth it asserts. See *temporalized extension of the Closed World Assumption*.
- But a row can be inserted with an assertion begin date in the future because when that future date comes to pass, the row will already exist. See *deferred assertion*.

Components: actionable, assert, assertion time period, asserted version table, begin date, effective time period, object, version.

assertion end date

Mechanics: the date on which the assertion time period of a row in an asserted version table ends, or a date indicating that the end of the assertion time period is unknown but presumed to be later than Now().

Semantics: the date indicating when a version stops being asserted as a true and/or actionable statement of what its object is like during its indicated period of effective time, or indicating that the end of the assertion time period is unknown but presumed to be later than Now().

Comments.

- An assertion end date is always set to 9999 when its row is inserted. It retains that value unless and until that assertion is withdrawn.

Components: actionable, assertion time period, asserted version table, effective time period, end date, Now(), object, statement, version.

assertion group

Mechanics: a group of one or more deferred assertions, sharing the same assertion begin date.

Semantics: a group of one or more assertions sharing the same future assertion time period.

Components: assertion, assertion begin date, deferred assertion, future assertion time period.

assertion group date

Mechanics: the assertion begin date on a group of one or more deferred assertions.

Semantics: the date which indicates when a group of deferred assertions will become currently asserted.

Comments:

- This date is also the unique identifier of an assertion group.

Components: assertion begin date, currently asserted, deferred assertion.

assertion table

Mechanics: a uni-temporal table whose explicitly represented time is assertion time.

Semantics: a uni-temporal table each of whose rows is a temporally-delimited assertion about what its object is like Now().

Components: uni-temporal, assertion, assertion time, Now(), object.

assertion time

Mechanics: a series of clock ticks, extending from the earliest to the latest clock ticks which the DBMS can recognize, within which assertion begin and end dates are located.

Semantics: the temporal dimension which interprets a time period associated with a row as indicating when that row is asserted to be true.

Components: assertion begin date, assertion end date, clock tick, temporal dimension, time period.

assertion time period

Mechanics: a time period in assertion time associated with a specific row in an asserted version table.

Semantics: the period of time during which a row in an asserted version table is claimed to make a true and/or actionable statement.

Components: actionable, asserted version table, assertion time, statement, time period.

as-was

Mechanics: data whose assertion end date is earlier than Now().

Semantics: data whose assertion time period is past.

Comments:

- See *as-is*. The as-was vs. as-is distinction is an assertion time distinction, but in supporting temporal data management in their databases, IT professionals often confuse this distinction with the effective time distinction between past and current versions.

Components: assertion end date, assertion time period, Now().

atomic clock tick

Mechanics: the smallest unit of time kept by a computer's clock that can be recognized by a specific DBMS.

Semantics: a unit of time that is indivisible for purposes of temporal data management.

Comments:

- See *clock tick*.

Components: N/A.

AVF

See *Asserted Versioning Framework*.

basic temporal transaction

Mechanics: a temporal transaction which does not specify any temporal parameters.

Semantics: a temporal transaction which accepts the default values for its temporal parameters, those being an effective begin date of Now(), an effective end date of 9999, an assertion begin date of Now() and an assertion end date of 9999.

Comments:

- Assertion end dates are the one temporal parameter than cannot be specified on temporal transactions. All temporal transactions, including basic ones, create asserted version rows with an assertion end date of 9999.

Components: assertion begin date, assertion end date, effective begin date, effective end date, Now(), temporal parameter, temporal transaction.

basic versioning

Mechanics: a form of versioning in which a version date is added to the primary key of an otherwise non-temporal table.

Semantics: a form of versioning in which all versions of the same object are contiguous.

Comments:

- Basic versioning is not part of Asserted Versioning. It is a form of best practices versioning. See Chapter 4.
- See also: *logical delete versioning, temporal gap versioning, effective time versioning*.

Components: contiguous, object, non-temporal table, version.

begin date

Mechanics: an assertion begin date or an effective begin date.

Semantics: a date which marks the start of an assertion or an effective time period.

Components: assertion begin date, assertion time period, effective begin date, effective time period.

bi-temporal data canonical form

Mechanics: the schema common to all asserted version tables.

Semantics: a single schema which can express the full range of bi-temporal semantics.

Comments:

- Any history table, logfile, or version table can be transformed into an asserted version table without loss of content.

Components: asserted version table, bi-temporal.

bi-temporal database

Mechanics: a database containing at least one bi-temporal table.

Components: bi-temporal table.

bi-temporal envelope

Semantics: a specified effective time period, included within a specified assertion time period.

Comments:

- The temporal scope of every temporal transaction is delimited by the bi-temporal envelope specified on the transaction.
- Every row in an asserted version table exists in a bi-temporal envelope.

Components: assertion time period, effective time period, include.

bi-temporal table

Mechanics: a table whose rows contain one pair of dates which define an epistemological time period, and a second pair of dates which define an ontological time period.

Semantics: a table whose rows contain data about both the past, the present and the future of things, and also about the past, the present and the future of our beliefs about those things.

Comments.

- See *epistemological time*, *ontological time*.

Components: "assert" cognate (belief), epistemological time, ontological time, thing, time period.

business data

Mechanics: all columns of an asserted version table other than those columns which implement Asserted Versioning.

Semantics: the columns of data which record the properties or relationships of objects during one or more periods of effective time.

Components: asserted version table, effective time, object.

business key

Mechanics: the primary key of the entity in the logical data model from which an asserted version table is generated.

Semantics: the unique identifier for an object as represented in a non-temporal table.

Comments:

- If a surrogate key is used in the logical data model, this surrogate key is used as the business key in an asserted version table.

Components: asserted version table, non-temporal table, object.

child managed object

Mechanics: a version in a TRI relationship.

Semantics: a managed object which represents a child object.

Components: child object, TRI, version.

child object

Semantics: an object, represented by a managed object, which is existence-dependent on another object, also represented by a managed object.

Components: existence dependency, managed object, object, represent.

child row

Mechanics: a row in an asserted version table which contains a non-null temporal foreign key.

Semantics: a version which represents an object which is existence-dependent on some other object.

Comments.

- The various "parent" and "child" expressions also apply to conventional tables, of course, in which case the relationship involved is referential integrity, not temporal referential integrity. But in this Glossary, we are explicitly defining these expressions as they apply to asserted version tables.

Components: asserted version table, existence dependency, object, temporal foreign key.

child table

Mechanics: a table which contains at least one temporal foreign key.

Semantics: a table whose rows represent child objects.

Components: child object, temporal foreign key.

child version

Mechanics: a version in an asserted version table X is a child to an episode in asserted version table Y if and only if the version in X has a temporal foreign key whose value is identical to the value of the object identifier of that episode in Y, and the effective time period of that episode in Y [fills^{-1}] the effective time period of that version in X.

Semantics: a version in an asserted version table X is a child to an episode in asserted version table Y if and only if the object for that version in X is existence dependent on the object for that episode in Y, and the effective time period of that episode in Y [fills^{-1}]the effective time period of that version in X.

Components: Allen relationship [fills^{-1}], asserted version table, effective time period, episode, existence dependency, object, object identifier, temporal foreign key, version.

chronon

Semantics: the term used in the computer science community for what Asserted Versioning calls an atomic clock tick.

Comments:

- See the 1992 entry in the bibliography for the standard computer science glossary of bi-temporal concepts.

Components: atomic clock tick.

circa flag

Mechanics: a flag used by the Asserted Versioning Framework as a component of one or more indexes on asserted version tables, in order to improve the performance of queries which reference asserted version tables, and also of updates to those tables.

Semantics: a flag which distinguishes between rows which are definitely known to be in the assertion time past from all other rows. (From Chapter 15).

Components: Asserted Versioning Framework, asserted version table, assertion time.

clock tick

Mechanics: the unit of time used for effective begin and end dates, assertion begin and end dates, episode begin dates and row create dates, in an asserted version table.

Semantics: the transition from one point in effective time or assertion time to the next point in effective time or assertion time, according to the chosen granularity which defines those two points in time as contiguous.

Comments:

- Note that chronons are atomic clock ticks, not clock ticks.
- A one month per tick clock represents a situation in which a database is updated at most once a month. By the same token, a one week or one day clock would record updates that take place at most once a week or once daily, respectively.

Components: asserted version table, assertion begin date, assertion end date, contiguous, effective begin date, effective end date, granularity, episode begin date, point in time, row create date.

closed assertion

Mechanics: a row in an asserted version table whose assertion end date is not 9999.

Semantics: a row in an asserted version table with a known assertion end date.

Components: asserted version table, assertion end date, 9999.

closed assertion time

Mechanics: an assertion time period whose end date is not 9999.

Semantics: an assertion time period whose end date is known.

Components: 9999, assertion end date, assertion time period.

closed effective time

Mechanics: a effective time period whose end date is not 9999.

Semantics: an effective time period whose end date is known.

Components: 9999, effective end date, effective time period.

closed episode

Mechanics: an episode whose effective end date is not 9999.

Semantics: an episode whose effective end date is known.

Components: 9999, effective end date, episode.

closed-closed

Mechanics: a convention for using a pair of clock ticks to designate an effective or assertion time period, in which the earlier clock tick is the first clock tick in the time period, and in which the later clock tick is the last clock tick in the time period.

Comments:

- Using this convention, two time periods [meet] if and only if the begin date of the later one is one clock tick after the end date of the earlier one, at whatever level of granularity is used to designate the clock ticks.

Components: assertion time period, clock tick, effective time period.

closed-open

Mechanics: a convention for using a pair of clock ticks to designate an effective or assertion time period, in which the earlier clock tick is the first clock tick in the time period, and in which the later clock tick is the first clock tick after the last clock tick in the time period.

Comments:

- Using this convention, two time periods [meet] if and only if the begin date of the later one is the same clock tick as the end date of the earlier one, at whatever level of granularity is used to designate the clock ticks.

Components: assertion time period, clock tick, effective time period.

conditional temporal transaction

Mechanics: a temporal transaction qualified by one or more WHERE clause business data predicates.

Components: business data, temporal transaction.

contiguous

Mechanics: time period or point in time X is contiguous with time period or point in time Y if and only if either X [meets] Y or X [meet^{-1}] Y.

Components: Allen relationship [meet], Allen relationship [meet^{-1}], point in time, time period.

conventional data

Mechanics: data in a conventional table.

Semantics: data which represents currently asserted current versions of persistent objects, but which lacks assertion time periods and effective time periods.

Comments:

- More accurately, conventional data is data which lacks *explicitly expressed* assertion and effective time periods. For in fact, conventional data is asserted, and its assertion time period is co-extensive with its physically presence in the database. And conventional data is also versioned, and the effective time of the one version of an object thus represented is always from Now() until further notice.

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

conventional database

Mechanics: a database none of whose tables are temporal.

Semantics: a table whose rows describe the current state of the objects they represent.

Comments:

- In the early part of the book, used as synonymous with "non-temporal database". But starting in Chapter 15, a distinction is drawn in which a conventional database may contain temporal data about persistent objects, but not in the form of bi-temporal tables.

Components: object, represent, state.

conventional table

Mechanics: a table whose rows have no assertion or effective time periods.

Semantics: a table whose rows contain data describing what we currently claim things are currently like.

Components: assertion time period, effective time period, thing.

conventional transaction

Mechanics: an insert, update or delete against a conventional table.

Semantics: a request to create, modify or remove a row in a conventional table.

Comments:

- Conventional transactions are SQL insert, update or delete statements.

Components: conventional table.

current assertion

See *currently asserted*.

current episode

Mechanics: an episode whose episode begin date is earlier than Now() and whose episode end date is later than Now().

Semantics: an episode for an object which includes a current version of that object.

Comments:

- An object may have at most one current episode.
- A past episode is not a current episode because its effective time period is past.
- A no longer asserted episode is not a current episode because its assertion time period is past.
- An episode all of whose assertions are deferred is not a current episode because it is not yet asserted.
- An episode all of whose versions have an effective begin date in the future is not a current episode because it has no current version.

Components: episode begin date, episode end date, Now(), object, version.

current transaction

Mechanics: a temporal transaction which becomes currently effective as soon as it is applied to the database, and which also becomes currently asserted as soon as it is applied to the database.

Semantics: a temporal transaction which accepts the date the transaction is submitted as the begin date of the assertion period within which its transformations will be contained, and also as the begin date of the effective period within which its transformations will be contained.

Comments:

- See *deferred transaction, proactive transaction, retroactive transaction*.

Components: assertion time period, begin date, currently asserted, currently effective, effective time period, temporal transaction.

current version

See *currently effective*.

currently asserted

Mechanics: a row in an asserted version table whose assertion time period includes Now().

Semantics: a statement which we currently claim is true and/or actionable.

Components: actionable, asserted version table, assertion time period, include, Now(), statement.

currently asserted current version

Mechanics: a row in an asserted version table whose assertion time period includes Now(), and whose effective time period includes Now().

Semantics: a row in an asserted version table which represents our current belief that the statement made by the business data in that row correctly describes what its object is currently like.

Comments:

- Rows in conventional tables are currently asserted current versions of the objects they represent.

Components: "assert" cognate (belief), asserted version table, assertion time period, business data, effective time period, include, Now(), object, statement.

currently effective

Mechanics: a row in an asserted version table whose effective time period includes Now().

Semantics: a statement which describes what the object it represents is currently like.

Components: asserted version table, effective time period, include, Now(), object, represent, statement.

dataset

Mechanics: a named collection of data that the operating system, or the DBMS, or the AVF, can recognize and manage as a single object.

Semantics: a managed object which represents a type, and which contains multiple managed objects each of which represent an instance of that type.

Comments:

- See also the Wikipedia definition.

Components: AVF, instance, managed object, object, type.

de-dupped

Mechanics: a conventional table from which multiple rows representing the same object have been eliminated and/or consolidated, leaving at most one row to represent each object.

Semantics: a table from which row-level synonyms have been eliminated.

Comments:

- If a table needs to be de-dupped, it is because its business keys are not reliable. In a world of pure theory, rows with unreliable business keys would not be allowed into a table. But business requirements for such data, however unreliable, frequently outweigh theoretical considerations.
- See also: *dirty data, row-level homonym, row-level synonym*.

Components: conventional table, object, represent, row-level synonym.

deferred assertion

Mechanics: a row in an asserted version table whose assertion begin date is greater than Now().

Semantics: an assertion which will not be made until some future date.

Components: asserted version table, assertion begin date, Now().

deferred assertion group

Mechanics: a collection of one or more rows in an asserted version table which all have the same future assertion begin date.

Components: assertion begin date, asserted version table.

deferred transaction

Mechanics: a temporal transaction which uses a future date as its assertion begin date.

Semantics: a temporal transaction which creates a deferred assertion.

Components: assertion begin date, deferred assertion, temporal transaction.

deMorgan's equivalences

Mechanics: NOT(X OR Y) is truth-functionally equivalent to (NOT-X AND NOT-Y).
(ii) (NOT-X OR NOT-Y) is truth-functionally equivalent to NOT(X AND Y).

Semantics: (i) if it's false that either of two statements is true, then it's true that both of them are false. (ii) If either of two statements is false, then it's false that they are both true.

Comments:

- Along with the truth-functional equivalence of (P IMPLIES Q) with (NOT-P OR Q), the deMorgan's equivalences allow any statement in propositional logic to be broken down into a series of ORs or a series of ANDs (including, in both cases, the NOT operand). In these simplified forms, computer software can carry out logic proofs, discovering contradictions when they exist, and presenting the logical implications of a set of assertions, many of which may turn out to be a surprise to those who accepted the original set of assertions.
- Software which does this is called an inference engine. While relational databases are the principal way that software helps us reason about instances, inference engines are the principal way that software helps us reason about types. Reasoning about types is what formal ontology is about. It is not often recognized that formal ontology and relational databases are complementary in this way, as means of reasoning about, respectively, types and instances.

Components: N/A.

design encapsulation

Mechanics: hiding all temporal design issues so that the design of a temporal database is a matter of (i) creating a conventional logical data model, (ii) designating those entities in the model which are to be physically generated as temporal tables, and (iii) describing all their temporal features in metadata tables.

Semantics: the ability to declaratively express all temporal design requirements for a data model.

Comments:

- Recent consulting experience by the authors has demonstrated the very significant cost savings that result from the ability to exclude all temporal design considerations from the process of creating the logical data model for a database. Discussions of how to best implement project-specific temporal requirements are often difficult, lengthy, contentious and inconclusive – in short, costly. Those costs are incurred over and over again, for every data model in which even a single table must be given temporal capabilities. And almost every time, the result is a slightly different solution, whose maintenance and querying are never quite the same as for any other temporalized tables. Asserted Versioning's design encapsulation feature eliminates these costs, and guarantees a uniform implementation of temporal semantics across all tables in all databases in the enterprise. Moreover, firmly grounded in computer science research, Asserted Versioning guarantees that its single enterprise solution is a complete and correct implementation of bi-temporal semantics.

Components: temporal database.

directly queryable data

Description: data which doesn't need to be transformed before queries can be written against it.

Comments:

- Much of the temporal data in an enterprise is not directly queryable.
- Even if one table of temporal data is directly queryable, the needs of any specific query may require access to multiple temporal tables, and often that *combination* of tables is not directly queryable. For example, a query may need both last year's customer data from an enterprise data warehouse, and last month's customer data from an Operational Data Store (ODS) history table. But it is unlikely that the schemas in the two databases are identical, and therefore unlikely that the combination of those tables is directly queryable.
- With directly queryable data, nothing needs to be done to get the data ready to be queried, whether by native SQL or via query tools.

Components: N/A.

dirty data

Mechanics: a collection of data whose instances do not all have unique identifiers.

Semantics: a collection of data in which row-level homonyms and/or row-level synonyms may exist.

Comments:

- Among IT professionals, the term "dirty data" often has a broader meaning, and covers a wide range of problems with data. In this narrower sense, dirty data is the result of unreliable business keys.
- See also: *de-dupped*, *row-level homonym, row-level synonym*.

Components: row-level homonym, row-level synonym.

effective begin date

Mechanics: using the closed-open convention, the first date in an effective time period.

Semantics: the date indicating when an effective time period begins.

Comments:

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

Components: closed-open, effective time period.

effective end date

Mechanics: using the closed-open convention, the first date after the last date in an effective time period.

Semantics: the date indicating when an effective time period ends.

Comments:

- The effective end date of an episode is the effective end date of its latest version.

Components: closed-open, effective time period.

effective time

Mechanics: the temporal dimension along which effective time periods are located.

Semantics: 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.

Comments:

- 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, effective time period, object, represent.

effective time period

Mechanics: the period of time of a version or an episode starting on its effective begin date and ending one clock tick prior to its effective end date.

Semantics: the period of time during which an object exists, as asserted by a row in an asserted version table.

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

effective time versioning

Mechanics: a form of versioning similar to temporal gap versioning, but in which a row create date is added to each version, in addition to a version begin date and a version end date.

Semantics: a form of versioning in which versions of the same object may or may not be contiguous, in which no version is physically deleted, in which the version dates delimit an effective time period, and in which the date the row was physically created is also provided.

Comments:

- Effective time versioning is not part of Asserted Versioning. See From Chapter 4.
- See also: *basic versioning, logical delete versioning, temporal gap versioning*.

Components: contiguous, effective time period, object, row create date, temporal gap versioning, version, version begin date, version end date.

empty assertion time

Mechanics: an assertion time period whose begin and end dates have the same value.

Semantics: an assertion time period which includes no clock ticks.

Comments:

- Deferred assertions which are deleted before they become currently asserted are moved into empty assertion time, i.e. are given empty assertion time periods.

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

end date

Mechanics: an assertion end date or an effective end date.

Semantics: a date which marks the end of an assertion or an effective time period.

Components: assertion end date, assertion time period, effective end date, effective time period.

enterprise contextualization

Mechanics: the expression of the Asserted Versioning Framework in a single unit of source code.

Semantics: a implementation of a software framework to provide seamless access to queryable bi-temporal state data about persistent objects, consisting of: one canonical set of schemas, across all tables and all databases; one set of transactions that update bi-temporal data and enforce temporal entity integrity and temporal referential integrity across all tables and all databases; a standard way to retrieve bi-temporal data; and a way to remove all temporal logic from application programs, isolate it in a separate layer of code, and invoke it declaratively.

Components: Asserted Versioning Framework, persistent object, seamless access, temporal data management taxonomy (queryable temporal data) / (state temporal data) / (bi-temporal data), temporal entity integrity, temporal referential integrity.

episode

Mechanics: within shared assertion time, a series or one or more rows representing the same object in which, in effective time, each non-initial row [meets] the next one, in which the initial row is [before^{-1}] any earlier row of the same object, and in which the latest row is [before] any later version of the same object.

Semantics: within one period of assertion time, a set of one or more effective-time contiguous asserted version rows representing the same object which are preceded and followed by at least one effective-time clock tick in which that object is not represented.

Comments:

- This is one of the most important concepts in asserted versioning.
- Episodes are a series of versions of the same object that are contiguous in effective time *within* a period of shared assertion time. They represent what we believe, during that period of assertion time, the life history of that object was/is/will be like, across those contiguous periods of effective time. (From Chapter 5.)

Components: Allen relationship [before], Allen relationship [before^{-1}], Allen relationship [meets], assertion time, clock tick, contiguous, effective time, object, represent, shared assertion time, version.

episode begin date

Mechanics: the effective begin date of the earliest version of an episode.

Semantics: the date on which the episode begins to be in effect.

Components: effective begin date, episode, version.

episode end date

Mechanics: the effective end date the latest version of an episode.

Semantics: the date on which the episode ceases to be in effect.

Components: effective end date, episode, version.

epistemological time

Semantics: the epistemological time of a row in a bi-temporal table is the period of time during which we claim that the statement made by that row is true and/or actionable.

Comments:

- A neutral term referring to either the standard temporal model's transaction time or to Asserted Versioning's assertion time.
- See "*assert" cognates*.

Components: actionable, "assert" cognates (claim), bi-temporal table, statement, time period.

event

Semantics: a point in time or a period of time during which one or more objects come into existence, change from one state to another state, or go out of existence.

Comments:

- Events are the occasions on which changes happen to persistent objects. As events, they have two important features: (i) they occur at a point in time, or sometimes last for a limited period of time; and (ii) in either case, they do not change. An event happens, and then it's over. Once it's over, that's it; it is frozen in time. (From Chapter 2.)

Components: point in time, period of time, object, persistent object, state.

existence dependency

Mechanics: a version of object X is existence dependent on an episode of object Y if and only if there can be no clock tick in assertion time during which there is a clock tick in effective time occupied by X but not occupied by Y.

Semantics: an object X is existence dependent on an object Y if and only if there can be no point in time at which X exists but Y does not exist.

Comments:

- Existence dependency is expressed by a foreign key if and only if the objects are represented in conventional tables.
- Existence dependency is expressed by a temporal foreign key if and only if the objects are represented in asserted version tables.

- Note the "can be". If "is" were in its place, the statements would express a correlation, but not a requirement.

Components: assertion time, clock tick, effective time, episode, object, occupy, version.

explicitly temporal data

Mechanics: a row of data which contains an assertion time period and/or an effective time period.

Semantics: a row of data whose assertion time and/or effective time is expressed by means of one or more columns of data.

Comments:
- See *implicitly temporal data*.

Components: assertion time, effective time.

external pipeline dataset

Mechanics: a dataset whose destination or origin is one or more production tables, and which is a distinct managed object to the operating system and/or the DBMS.

Semantics: a dataset whose contents are production data.

Comments:
- See the Wikipedia definition.
- Tabular data which will become part of the production database are transactions acquired or generated by a company's OLTP systems. They are either immediately and directly applied to the production database, or are augmented, corrected or otherwise transformed as they are moved along an "inflow data pipeline" leading into the production database.
- Tabular data which has been a part of the production database are the persisted result sets of SQL queries or equivalent processes. They are either end state result sets, i.e. immediately delivered to internal business users or exported to outside users, or are augmented as they move along an "outflow data pipeline" leading to a final state in which they are delivered to internal business users or outside users.
- The various kinds of external pipeline datasets do not form a partitioning. Most of these names are in fairly widespread usage, but no standard definition of them exists. Therefore, in this Glossary, we will provide a description of them, but cannot provide a definition.
- The distinction between inflow pipeline datasets and outflow pipeline datasets is a matter of perspective. Any physical dataset may be used to update a production table, in which case it is an inflow dataset. And any physical dataset other than those created by means of manual data entry or automated data collection from

instruments, for example, contain data from production tables and so are outflow datasets.

Components: dataset, managed object, production data, production table.

external pipeline dataset, batch file

Description: this term is generally used to refer to a file or a table of insert, update and/or delete transactions whose target is a production table. It is a dataset that exists at the start of an inflow pipeline.

external pipeline dataset, data extract

Description: this term is generally used to refer to the results of a query which are stored as a physical dataset which will be moved to some other location before being made available to end users. It is a dataset that exists along an outflow pipeline.

external pipeline dataset, data feed

Description: this term is generally used to refer to a dataset which is being used to populate a production table. It is a dataset that exists at the end of an inflow pipeline to its target.

Comments:

- Of course, the same physical dataset which was an extract may, with or without going through additional modifications, also be a feed.

external pipeline dataset, data staging area

Description: this term is generally used to refer to a physical dataset of production data that is being worked on until it can be moved into its target production table.

Comments:

- When applied to a production table, the contents of a data staging area may or may not overlay rows already in that table.
- The contents of a data staging area may have originated as a copy of rows in a production table, or simply be a collection of transactions each of which requires data from multiple sources, and so must be built up over time. If it originated as a copy of production rows, it is both an outflow pipeline dataset and, later on, an inflow pipeline dataset.
- The purpose of a staging area is to move the row or rows representing an object into a state where they are not available to normal queries. The reason for doing this is usually to withdraw those rows into an area where a series of updates can be made to them, only after which are those rows returned to production data status.

external pipeline dataset, history table

Description: this term is generally used to refer to a table of data which contains the before-image copies of production rows which are about to be updated. It is a dataset that exists at the end of a (very short) outflow pipeline.

external pipeline dataset, logfile table

Mechanics: this term is generally used to refer to a table of data which contains the before-image copies of production rows which are about to be inserted, updated or deleted. It is a dataset that exists at the end of a (very short) outflow pipeline.

external pipeline dataset, query result set

Mechanics: this term is always used to refer to the results of an SQL query. It is a dataset that exists at the start of an outflow pipeline.

external pipeline dataset, report

Description: this term is generally used to refer to a dataset at the end of an outflow pipeline, at which point the data can be directly viewed.

external pipeline dataset, screen

Mechanics: this term is generally used to refer to a dataset at the end of an outflow pipeline, at which point the data can be directly viewed.

Comments:

- Aside from the difference in media (video display vs. hardcopy), screens differ from reports in that reports usually contain data representing many objects, while screens usually contain data representing one object or a few objects.

fall into currency

Mechanics: to become a current assertion and/or a current version when an assertion and/or effective begin date becomes a date in the past.

Semantics: to become a current assertion and/or a currently version because of the passage of time.

Comments:

- Once an assertion and/or a version falls into currency, it remains current until its end date becomes a date in the past.

Components: assertion begin date, current assertion, effective begin date, current version, passage of time.

fall out of currency

Mechanics: to become a past assertion and/or a past version when an assertion and/or effective end date becomes a date in the past.

Semantics: to become a past assertion and/or a past version because of the passage of time.

Components: assertion end date, effective end date, passage of time, past assertion, past version.

far future assertion time

Mechanics: the assertion time location of deferred assertions whose begin dates are far in the future.

Semantics: the assertion time location of deferred assertions that would be obsolete before the passage of time made them current.

Comments:

- See *near future assertion time*.
- A typical far future assertion begin date would be hundreds or even thousands of years in the future. In business databases, there is little risk of such assertions falling into currency by the mere passage of time.
- The intent, with far future deferred assertions, is that they exist in a "temporal sandbox" within a production table. They can be used for forecasting, for "what if" analyses, or for building up or otherwise working on one or more assertions until those assertions are ready to become visible in the production table that physically contains them. When they are ready, an approval transaction will move them to near future assertion time, where the passage of time will quickly make them current assertions.

Components: assertion begin date, assertion time, current assertion, deferred assertion, passage of time.

*f*CTD function

Mechanics: a function that converts an integer into that integer number of clock ticks of the correct granularity.

Comments:

- "CTD" stands for "clock tick duration". (From Chapter 14.)

Components: clock tick, granularity.

*f*CUT function

Mechanics: a function that splits a row in an asserted version table into two contiguous versions in order to [align] version boundaries in a target table to effective time boundaries on a temporal transaction.

Comments:

- A temporal update or delete transaction will affect only clock ticks within the effective time period specified by the transaction.
- If the first clock tick in the transaction's effective time period is a non-initial clock tick in a version of the object referenced by the transaction, then that version must be split into a contiguous pair of otherwise identical versions.
- If the last clock tick in the transaction's effective time period is a non-final clock tick in a version of the object referenced by the transaction, then that version must be split into a contiguous pair of otherwise identical versions.
- The result is that the temporal transaction can be carried out by updating or deleting complete versions.
- See *match*.

Components: Allen relationship [align], asserted version table, contiguous, effective time, target table, temporal transaction, version.

from now on

Mechanics: a time period of [Now() – 9999], where Now() is the clock tick current when the time period was created.

Semantics: a time period which is current from the moment it is created until further notice.

Comments:

- That current assertion time starts Now(), i.e. when the transaction is processed, and continues on until further notice. Every temporal transaction that accepts the default values for effective time, creates a version that describes what its object looks like *from now on*. Every non-deferred temporal transaction creates an assertion that, *from now on*, claims that its version makes a true statement. (From Chapter 9.)

Components: 9999, clock tick, Now(), time period, until further notice.

*f*TRI function

Mechanics: a function that evaluates to True if and only if a valid TRI relationship holds between the episode and the version specified in the function.

Components: episode, TRI, version.

future assertion

See *deferred assertion*.

future version

Mechanics: a row in an asserted version table whose effective begin date is later than Now().

Semantics: a row in an asserted version table which describes what the object it represents will be like during a specified future period of time.

Components: asserted version table, effective begin date, Now(), object, represent, time period.

granularity

Mechanics: the size of the unit of time used to delineate effective time periods and assertion time periods in an asserted version table.

Comments:

- More generally, the granularity of a measurement is the size of the units in which the measurement is expressed, a smaller size referred to as a "finer" granularity. For example, inches are a finer granularity of linear measurement than yards, and ounces are a finer granularity of the measurement of weight than pounds.

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

hand-over clock tick

Semantics: the point in near future assertion time to which an approval transaction sets the assertion begin date of one or more deferred assertions, and also the assertion end date of any assertions which they replace or supercede.

Components: approval transaction, assertion begin date, assertion end date, deferred assertion, near future assertion time, replace, supercede.

historical data

Mechanics: rows in asserted version tables whose effective end date is earlier than Now().

Semantics: data which describes the past state or states of a persistent object.

Comments:

- Note that this term does not refer to data which is historical, i.e. no longer current date, but rather to data which is about history, i.e. about the past states of persistent objects.
- For the term which does refer to data which is itself historical, see *as-was data*.
- Note that, in the special sense used here, historical data is data about persistent objects. Thus, fact/dimension data marts do not provide historical data because their history is a history of events, not of objects.

<u>Components</u>: asserted version table, effective end date, Now(), persistent object, state.

implicitly temporal data

<u>Mechanics</u>: a row in a non-temporal table whose assertion time and/or effective time is co-extensive with its physical presence in its table.

<u>Semantics</u>: a row of data whose assertion time and/or effective time is not expressed by means of one or more columns of data.

<u>Comments</u>:
- Thus, rows in conventional tables are implicitly temporal data. No columns of those tables indicate assertion or effective time periods. Each row is asserted for as long as it is present in its table, and is in effect for as long as it is present in its table.

<u>Components</u>: assertion time, effective time, non-temporal table.

incommensurable

<u>Mechanics</u>: two asserted version rows are incommensurable if and only if their assertion time periods do not [<u>intersect</u>].

<u>Semantics</u>: unable to be meaningfully compared.

<u>Comments</u>:
- Rows which share no clock ticks in assertion time are semantically and truth-functionally isolated from one another. They are what philosophers call *incommensurable*. (From Chapter 6.)
- Incommensurability restricts TEI and TRI relationships to shared assertion time.

<u>Components</u>: Allen relationship [<u>intersect</u>], asserted version table, assertion time period.

inflow pipeline dataset

<u>Mechanics</u>: a dataset whose destination is one or more production tables.

<u>Comments</u>:

- Inflow pipeline datasets are tabular data which will become part of the production database. They originate with transactions acquired or generated by a company's OLTP systems. They are either immediately and directly applied to the production database, or are augmented, corrected or otherwise transformed as they are moved along an "inflow data pipeline" leading into the production database.

Components: dataset, production table.

instance

Semantics: a thing of a particular type.

Comments:
- See *type*.
- The concepts of types and instances has long history. A related distinction is that between particulars and universals.

Components: thing, type.

internalized pipeline dataset, Current Data

Mechanics: all those rows in asserted version tables which lie in the assertion time present and also in the effective time present. (From Chapter 13.)

Semantics: a record of what we currently believe things are currently like.

Components: asserted version table, assertion time, effective time.

internalized pipeline dataset, Current History

Mechanics: all those rows in asserted version tables which lie in the assertion time present but in the effective time past. (From Chapter 13.)

Semantics: a record of what we currently believe things used to be like.

Components: asserted version table, assertion time, effective time.

internalized pipeline dataset, Current Projections

Mechanics: all those rows in asserted version tables which lie in the assertion time present but in the effective time future. (From Chapter 13.)

Semantics: a record of what we currently believe things may eventually be like.

Components: asserted version table, assertion time, effective time.

internalized pipeline dataset, Pending History

Mechanics: all those rows in asserted version tables which lie in the assertion time future but in the effective time past. (From Chapter 13.)

Semantics: a record of what we may come to believe things used to be like.

Components: asserted version table, assertion time, effective time.

internalized pipeline dataset, Pending Projections

Mechanics: all those rows in asserted version tables which lie in both the assertion time future and in the effective time future. (From Chapter 13.)

Semantics: a record of what we may come to believe things may eventually be like.

Components: asserted version table, assertion time, effective time.

internalized pipeline dataset, Pending Updates

Mechanics: all those rows in asserted version tables which lie in the assertion time future but in the effective time present. (From Chapter 13.)

Semantics: a record of what we may come to believe things are currently like.

Components: asserted version table, assertion time, effective time.

internalized pipeline dataset, Posted History

Mechanics: all those rows in asserted version tables which lie in both the assertion time past and also in the effective time past. (From Chapter 13).

Semantics: a record of what we used to believe things used to be like.

Components: asserted version table, assertion time, effective time.

internalized pipeline dataset, Posted Projections

Mechanics: all those rows in an asserted version table which lie in the assertion time past but in the effective time future. (From Chapter 13.)

Semantics: a record of what we used to believe things may eventually be like.

Components: asserted version table, assertion time, effective time.

internalized pipeline dataset, Posted Updates

Mechanics: all those rows in asserted version tables which lie in the assertion time past but in the effective time present. (From Chapter 13)

Semantics: a record of what we used to believe things are currently like.

Components: asserted version table, assertion time, effective time.

lock

Mechanics: to lock a row in an asserted version table is to set its assertion end date to a non-9999 value which is later than Now().

Semantics: to lock an asserted version row is to prevent it from being updated or deleted without moving it into past assertion time.

Comments:

- See *withdraw*.
- A deferred transaction locks a row by setting its assertion end date to the assertion begin date of the deferred assertion it creates. Rows that are locked by means of deferred assertions remain currently asserted until their assertion end dates fall into the past.

Components: 9999, asserted version table, assertion end date, Now(), past assertion.

logical delete versioning

Mechanics: a form of versioning similar to basic versioning, but in which delete transactions are carried out as logical deletions, not as physical deletions.

Semantics: a form of versioning in which all versions of the same object are contiguous, and in which no version is physically deleted.

Comments:

- Logical delete versioning is not part of Asserted Versioning. See Chapter 4.
- See also: *basic versioning, temporal gap versioning, effective time versioning*.

Components: basic versioning, contiguous, object, version.

maintenance encapsulation

Mechanics: hiding the complexity of temporal insert, update and delete transactions so that a temporal transaction needs, in addition to the data supplied in a corresponding conventional transaction, either no additional data, or else one, two or three dates representing, respectively, the effective begin date of a version, the effective end date of a version or the assertion begin date of an assertion.

Semantics: the ability to express all temporal parameters on temporal transactions declaratively.

Comments:

- Maintenance encapsulation means that inserts, updates and deletes to bi-temporal tables, and queries against them, are simple enough that anyone who could write them against non-temporal tables could also write them against these tables. (From *Preface*.)

Components: assertion, assertion begin date, conventional transaction, effective begin date, effective end date, temporal transaction, version.

managed object

Semantics: a named data item or collection of data that is manipulable by the operating system, the DBMS or the AVF, and which references a persistent object.

Comments:

- For example, tables, rows, columns, versions and episodes are all managed objects. Individual customers, clients or policies, while examples of objects, are not examples of managed objects.
- In the phrase "managed object", the word "object", by itself, has no meaning. In particular, it has no connection with the technical term "object".
- For example, tables, rows, columns, versions and episodes are all managed objects. Individual customers, clients or policies, while examples of objects, are not examples of managed objects.
- Managed objects are data which transformations and constraints treat as a single unit. (From Chapter 5.)

Components: reference, persistent object.

match

Mechanics: to apply the *f*CUT function to any non-locked version in the target table of a temporal update or delete transaction whose effective time period [overlaps] that specified on the transaction.

Semantics: to modify the target table for a temporal update or delete transaction so that there is no non-locked version for the object specified on the transaction whose effective time period [overlaps] the effective time period specified on the transaction.

Components: Allen relationship [overlaps], effective time period, fCUT, lock, object, target table, temporal delete transaction, temporal update transaction, version.

near future assertion time

Mechanics: the assertion time location of deferred assertions whose begin dates are about to fall into currency.

Semantics: the assertion time location of deferred assertions that the passage of time will make current soon enough to satisfy business requirements.

Comments:

- See *far future assertion time*.
- Deferred assertions located in the near future will become current assertions as soon as enough time has passed. In a real-time update situation, a near future deferred assertion might be one with an assertion begin date just a few seconds from now. In a batch update situation, a near future deferred assertion might be one that does not become currently asserted until midnight, or perhaps even for another several days. What near future deferred assertions have in common is that, in all cases, the business is willing to wait for these assertions to *fall into currency,* i.e. to become current not because of some explicit action, but rather when the passage of time reaches their begin dates. (From Chapter 12.)

Components: assertion begin date, assertion time, current assertion, deferred assertion, fall into currency, passage of time.

non-contiguous

Mechanics: time period or point in time X is non-contiguous with time period or point in time Y if and only if either X is [before] Y or X is [before^{-1}] Y.

Components: Allen relationship [before], Allen relationship [before^{-1}], point in time, time period.

non-temporal data

See *conventional data*.

non-temporal database

See *conventional database*.

non-temporal table

See *conventional table.*

Now()

Mechanics: a DBMS-agnostic representation of a function which always returns the current clock tick.

Semantics: a variable representing the current point in time.

Comments:

- SQL Server may use *getdate()*, and DB2 may use *Current Timestamp* or *Current Date*. (From Chapter 3.)
- Now() stands for a function, not a value. However, we will often use *Now()* to designate a specific point in time. For example, we may say that a time period starts at Now() and continues on until 9999. This is a shorthand way of emphasizing that, whenever that time period was created, it was given as its begin date the value returned by Now() at that moment. (From Chapter 3.)

Components: clock tick, point in time.

object

Mechanics: what is represented by the object identifier (oid) in an asserted version table.

Semantics: an instance of a type of thing which exists over time, has properties and relationships, and can change over time.

Comments:

- See also: *events*. Events, whether points in time or durations in time, are not objects, because events, by definition, do not change.
- Examples of objects include vendors, customers, employees, regulatory agencies, products, services, bills of material, invoices, purchase orders, claims, certifications, etc.

Components: asserted version table, instance, object identifier, oid, represent, type, thing.

object identifier

Mechanics: the unique identifier of the persistent object represented by a row in an asserted version table, used as part of the primary key of that row.

Comments:

- The unique identifier of a row in an asserted version table is the concatenation of an object identifier, an effective begin date, and an assertion begin date.

Components: asserted version table, persistent object.

occupied

Mechanics: a series of one or more clock ticks is occupied by an object if and only if those clock ticks are all included within the effective time period of a version of that object.

Semantics: a time period is occupied by an object just in the object is represented in every clock tick in that time period.

Components: clock tick, effective time period, include, object, represent, version.

oid

See *object identifier*.

ontological time

Semantics: the ontological time of a row in a bi-temporal table is the period of time during which its referenced object exists.

Comments:

- A neutral term referring to either the standard temporal model's valid time or to Asserted Versioning's effective time.

Components: bi-temporal table, object, referent, time period.

open episode

Mechanics: An episode whose effective end date is 9999.

Semantics: an episode whose effective end date is not known.

Comments:

- The effective end date of an episode is the effective end date of its latest version.

Components: 9999, effective end date, episode.

open version

Mechanics: a version whose effective end date is 9999.

Semantics: a version whose effective end date is unknown.

Components: 9999, effective end date, version.

open-closed

Mechanics: a convention for using a pair of clock ticks to designate an effective or assertion time period, in which the earlier clock tick is the last clock tick before the first clock tick in the time period, and in which the later clock tick is the last clock tick in the time period.

Comments:

- Using this convention, two time periods [meet] if and only if the begin date of the later one is the same clock tick as the end date of the earlier one, at whatever level of granularity is used to designate the clock ticks.

Components: assertion time period, clock tick, effective time period.

open-open

Mechanics: a convention for using a pair of clock ticks to designate an effective or assertion time period, in which the earlier clock tick is the last clock tick before the first clock tick in the time period, and in which the later clock tick is the first clock tick after the last clock tick in the time period.

Comments:

- Using this convention, two time periods [meet] if and only if the begin date of the later one is one clock tick before the end date of the earlier one, at whatever level of granularity is used to designate the clock ticks.

Components: assertion time period, clock tick, effective time period.

outflow pipeline dataset

Mechanics: a dataset whose origin is one or more production tables.

Comments:

- Outflow pipeline datasets are tabular data which has been a part of the production database; they are the persisted result sets of SQL queries or equivalent processes. They are either end state result sets, i.e. immediately delivered to internal business users or exported to outside users, or are augmented as they move along an "outflow data pipeline" leading to a final state in which they are delivered to internal business users or outside users.
- The termination points of outflow pipelines may be either internal to the organization, or external to it; and we may think of the data that flows along these pipelines to be the result sets of queries applied to those production tables. (From Chapter 12.)

Components: dataset, production table.

override

Mechanics: to set the assertion end date of a row to the same value as its assertion begin date.

Semantics: to withdraw a row into empty assertion time.

Comments:

- An assertion is overridden only when an approval transaction retrograde moves a matching version to an earlier assertion period than the assertion period of the assertion being overridden.

Components: assertion begin date, assertion end date, empty assertion time.

parent episode

Mechanics: an episode in an asserted version table X is a parent to a version in asserted version table Y if and only if the version 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 includes ([fills^{-1}]) the effective time period of that version in Y.

Semantics: a, episode in an asserted version table X is a parent to a version in asserted version table Y if and only if the object for that version in Y is existence dependent on the object for that episode in X, and the effective time period of that episode in X includes ([fills^{-1}]) the effective time period of that version in Y.

Components: Allen relationship [fills^{-1}], asserted version table, effective time period, episode, existence dependency, include, object, object identifier, temporal foreign key, version.

parent managed object

Mechanics: an episode in a TRI relationship.

Semantics: a managed object which represents a parent object.

Components: episode, parent object, TRI.

parent object

Semantics: an object, represented by a managed object, on which another object, also represented by a managed object, is existence dependent.

Components: existence dependency, managed object, object.

parent table

Mechanics: X is a parent table if and only if there is a table, not necessarily distinct, which contains a temporal foreign key which references X.

Semantics: X is a parent table if and only if its rows represent parent objects.

Components: temporal foreign key, parent object.

passage of time

Semantics: the means by which asserted versions may move from future to current, and from current to past time, in either or both temporal dimensions.

Comments:

- Creating future versions and/or deferred assertions is a way of managing a large volume of transactions so that the result of those transactions will all become current on exactly the same clock tick. An example would be a corporate acquisition in which the entire set of customers, policies, accounts and other objects managed by the acquired company need to become part of the acquiring company's production databases – and thus available to the maintenance processes, queries and reporting processes of the acquiring company – all at the same time, on precisely the same clock tick.

Components: asserted version, temporal dimension.

past assertion

Mechanics: a row whose assertion end date is earlier than Now().

Semantics: a row which represents a statement we are no longer willing to claim is true and/or actionable.

Components: actionable, assertion end date, Now(), represent, statement.

past episode

Mechanics: an episode of an object whose latest version has an effective end date which is earlier than Now().

Semantics: the representation of an object in a period of past effective time which is either [before] or [before-^{1}] all other representations of the same object.

Components: Allen relationship [before], Allen relationship [before^{-1}], episode, effective end date, effective time, Now(), object, represent, version.

past version

Mechanics: a version of an object whose effective end date is earlier than Now().

Semantics: the representation of an object in a period of past effective time which [meets] and/or [meets-^{1}] another representation of the same object.

Components: Allen relationship [meets], Allen relationship [meets^{-1}], effective end date, effective time, Now(), object, represent, version.

pending transaction

Description: an insert, update or delete statement that has been written but not yet submitted to the applications that maintain the production database. Sometimes pending transactions are collected outside the target database, in batch transaction files. More commonly, they are collected inside the target database, in batch transaction tables. (From *Preface*.)

Comments:

- Pending transactions are collected in batch transaction files. See *external pipeline dataset, batch transaction file*.
- As internalized by Asserted Versioning, they are those semantic collections of asserted version rows called Pending History, Pending Updates and Pending Projections.

PERIOD datatype

Mechanics: the representation of a time period as a datatype.

Semantics: the representation of a time period by a single column of data, a well-defined set or range of values, and a well-defined set of operations on those values.

Comments:

- Several DBMS vendors, including Oracle and Teradata, have defined PERIOD datatypes, but we do not know whether or not their definitions are equivalent.
- We would regard any PERIOD datatype as inadequate unless it could express a time period with an unknown starting point or an unknown ending point. We would regard DBMS support for any PERIOD datatype as inadequate unless a unique index could be defined on any column with a PERIOD datatype that would treat any two time periods as duplicates if they shared even a single clock tick.

Components: N/A.

persistent object

See *object*.

physical logfile

Mechanics: the ability of the AVF to recreate the state of an asserted version table as of any past point in time, using the row create date.

Comments:

- See *semantic logfile*.
- Deferred assertions which have been retrograde moved from far future to near future assertion time are the one exception to this ability to recreate any past physical state of an asserted version table. Currently, Asserted Versioning does not preserve information about the far future assertion time these assertions originally existed in. (From Chapter 16)

Components: asserted version table, AVF, row create date.

physical transaction

Description: a SQL insert, update or delete transaction submitted to the DBMS.

Comments:

- The AVF translates each temporal transaction into the one or more physical transactions that, when processed, carry out the intentions expressed by the user who submitted the temporal transaction.

pipeline dataset

Mechanics: a dataset whose destination or origin is one or more production tables.

Comments:

- Pipeline production datasets (*pipeline datasets*, for short) are points at which data comes to rest along the inflow pipelines whose termination points are production tables, or along the outflow pipelines whose points of origin are those same tables. (From Chapter 12.)

Components: dataset, production table.

pipeline dataset, internalization of

Mechanics: the representation of the contents of external pipeline datasets as rows in asserted version production tables which exist in non-current assertion time and/or non-current effective time.

Components: asserted version table, assertion time, current assertion, current version, effective time, external pipeline dataset, production table, represent.

pipeline dataset, re-presentation of

Mechanics: the ability to recreate the contents of any external pipeline dataset from internal pipeline datasets by means of a query.

Components: external pipeline dataset, internalized pipeline dataset.

point in time

Mechanics: a time period whose begin date value is one clock tick before its end date value.

Semantics: a time period consisting of a single clock tick.

Comments:

- For purposes of temporal data management, a point in time is considered indivisible.
- Note that in this book, in which we use a month as our level of temporal granularity, that one month is considered indivisible. For example, if a transaction is applied, it is assumed that its results will remain unchanged until the next month.

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

posted transaction

Description: copies of data about to be inserted, and before-images of data about to be updated or deleted. The contents of various forms of logfiles. (From *Preface*.)

Comments:

- Posted transactions are collected in logfiles. See *external pipeline dataset, logfile table*.
- As internalized by Asserted Versioning, they are those semantic collections of asserted version rows called Posted History, Posted Updates and Posted Projections.

proactive delete

Semantics: a temporal delete transaction that removes the representation of an object from one or more clock ticks in future effective time.

Components: clock tick, effective time, object, represent, temporal transaction.

proactive insert

Semantics: a temporal insert transaction that adds the representation of an object to one or more clock ticks in future effective time.

Components: clock tick, effective time, object, represent, temporal transaction.

proactive transaction

Mechanics: a temporal transaction that specifies a non-9999 effective end date that is later than Now().

Semantics: a temporal transaction which anticipates the effective-time future.

Comments:

- See *retroactive transaction*.

Components: temporal transaction, effective begin date.

proactive update

Semantics: a temporal update transaction that changes the business data representing an object in one or more clock ticks in future effective time.

Components: business data, clock tick, effective time, object, represent, temporal transaction.

production data

Semantics: business data that describes the objects and events of interest to the business.

Components: business data, object, event.

production database

Mechanics: a database that contains production data.

Semantics: the logical collection of databases whose currently asserted contents are the company's official statements describing the objects and events represented by those statements.

Comments:

- Production databases are the collections of production datasets which the business recognizes as the official repositories of that data. Production databases consist of production tables. (From Chapter 12.)

Components: currently asserted, event, object, production data, represent, statement.

production dataset

Description: a dataset that contains production data.

production query

Description: a query which is usually embedded in an application program, and which is run as part of the IT production schedule.

Comments:

- See also: *ad hoc query*. (From Chapter 5.)

production row

Mechanics: a row in a production table.

Semantics: : a row which describes an object or event of interest to the business.

Components: event, object, production table.

production table

Mechanics: a table in a production database.

Semantics: a table whose rows describe an object or event of interest to the business.

Comments:

- The term "production" indicates that these tables are in use by business processes, and contain "real" data. Regularly scheduled processes are being carried out to maintain these tables, and to keep their contents as accurate, secure and current as possible. Regularly scheduled processes, as well as non-scheduled ones, are being carried out to access this data to obtain needed information. So production tables are the tables that the business tries to keep accurate, current and secure, and from which it draws the information it needs to carry out its mission and meet its objectives. (From Chapter 3.)
- Production tables are production datasets whose data is designated as always reliable and always available for use. (From Chapter 12.)

Components: event, object, production database.

query encapsulation

Mechanics: hiding the complexity of many temporal queries so that (i) a query as of a past or future point in either or both of the data's two temporal dimensions can be written as if it were a query against a conventional table with the addition or one or two predicates to the WHERE clause of the query; and (ii) a query for data current in both its temporal dimensions can be written as a conventional query against a view generated from a temporal table.

Semantics: the ability to express most temporal query criteria with simple predicates added to the WHERE clause of an otherwise conventional query.

Comments:

- Query encapsulation means that queries against asserted version tables are simple enough that anyone who could write them against non-temporal tables could also write them against these tables. (From *Preface*.)

Components: conventional table, temporal dimension, temporal table.

queryable object

Semantics: a managed object that can be named in an SQL query.

Components: managed object.

referent

Mechanics: the persistent object identified by the object identifier of a row in an asserted version table.

Semantics: whatever is referred to and described by a managed object.

Components: asserted version table, managed object, object identifier, persistent object.

reliable business key

Mechanics: a business key which can be used to match data on a temporal transaction to one or more rows in the target table for that transaction.

Semantics: a business key which represents one and only one object.

Components: business key, object, represent, target table, temporal transaction.

replace

Mechanics: a row X replaces a row Y if and only if X and Y both represent the same object, X's effective time period [equals] Y's effective time period, X's business data is identical to Y's business data, and X's assertion time period [meets^{-1}] Y's assertion time period.

Semantics: a row X replaces a row Y if and only if X and Y both represent the same object, and X is a business-data identical assertion about what Y is like during the effective time period specified by Y.

Comments:

- See also: *withdraw*, *supercede*.
- A row X replaces a row Y if and only if X says the same thing about what the object Y represents is like, during all or part of the effective time period specified by Y.
- If a superceding version was also created as part of the temporal update transaction which created a replacement version, then this replacement version will [meet] that superceding version in effective time.
- A temporal update transaction whose effective time period [intersects] that of a target version, but does not [equal] it, requires the AVF to withdraw the target version and then to split that target version into one version that does *match* the transaction, and one (or two) versions that do not. This is done with the *f*CUT function. The resulting version or versions that do not match the transaction are replacements, with identical business data. The one version that does match the transaction is updated with the new business data, and supercedes the corresponding effective timespan of the withdrawn version.

Components: version, assertion, effective time, withdraw, supercede, temporal.

represent

Mechanics: a managed object represents an object in a series of one or more clock ticks if and only if those clock ticks are all included within the time period of that managed object.

Comments:

- See also *occupy*.

Components: managed object, clock tick, object, time period.

re-present

Description we use the hyphenated form "re-present" advisedly. We do mean that we will show how to *represent* those internalized datasets as queryable objects, in the ordinary sense of the word "represent". But we also wish to emphasize that we are re-presenting, i.e. presenting again, things whose presence we have removed.[2]

[2] We also wish to avoid confusion with our technical term *represent*, in which business data, we say, is represented in an effective time clock tick within an assertion time clock tick just in case

Those things are the external pipeline datasets which, in Chapter 12, we showed how to *internalize* within the production tables which are their destinations or points of origin. (From Chapter 13.)

retroactive delete

Mechanics: a temporal delete transaction that specifies an effective begin date that is earlier than Now().

Semantics: a temporal delete transaction that removes the representation of an object from one or more clock ticks in past effective time.

Comments:

- In a conventional table, the only mistake in data that can be corrected is a mistake in data values, and the correction is done "destructively", by overwriting the old data.
- But in an asserted version table, there are two other mistakes in data. One is to mistakenly claim that an object was represented during a past effective time period. The other is to mistakenly claim that an object was not represented during a past effective time period. A retroactive delete transaction is the means by which the former mistake is corrected. A retroactive insert transaction is the means by which the latter mistake is corrected.

Components: clock tick, effective begin date, Now(), object, represent, past version, temporal transaction.

retroactive insert

Mechanics: a temporal insert transaction that specifies an effective begin date that is earlier than Now().

Semantics: a temporal insert transaction that adds the representation of an object to one or more clock ticks in past effective time.

Comments:

- See *retroactive delete*.

Components: clock tick, effective begin date, Now(), object, represent, past version, temporal transaction.

---

that business data exists on an asserted version row whose assertion and effective time periods contain those clock tick pairs.

retroactive transaction

Mechanics: a temporal transaction that specifies an effective begin date that is earlier than Now().

Semantics: a temporal transaction which alters the effective-time past.

Comments:

- See *proactive transaction*.

Components: temporal transaction, effective begin date.

retroactive update

Mechanics: a temporal update transaction that specifies an effective begin date that is earlier than Now().

Semantics: a temporal update transaction that changes the business data representing an object in one or more clock ticks in past effective time.

Components: business data, clock tick, effective time, object, represent, temporal transaction.

retrograde movement

Mechanics: changing the assertion begin date on a deferred assertion to an earlier date.

Semantics: the movement of a deferred assertion from far future to near future assertion time.

Components: assertion begin date, deferred assertion, far future assertion time, near future assertion time.

row create date

Mechanics: the date on which a row in an asserted version table is physically inserted into that table.

Comments:

- The means by which a physical logfile can be re-presented as a queryable object.

row-level homonym

Mechanics: a row whose business key identifies two or more different objects.

Semantics: a row which represents two or more different objects.

Comments:

- A row-level homonym is eliminated by replacing it with multiple rows, one for each object represented by the row. For example, a row in a Client table which has been updated with data representing two or more different clients, is a homonym.
- See also: *de-dupped, dirty data, row-level synonym.*

Components: business key, object.

row-level synonym

Mechanics: two or more rows which cannot be distinguished by means of their business keys.

Semantics: two or more rows which represent the same object or, in a temporal context, represent the same object in at least one clock tick.

Comments:

- Row-level synonyms are eliminated by replacing them with one row that represents the one object that each of the synonyms references.. For example, multiple rows in a Client table which are discovered to represent the same client, are synonyms.
- See also: *de-dupped, dirty data, row-level homonym.*

Components: business key, clock tick, object.

seamless access

Description: the ability, in a query, to assemble result sets containing rows which exist in past, present or future time, in either or both of the two temporal dimensions, from the same set of tables that would be specified if the query were to retrieve current data only.

seamless access, performance aspect

Description: query performance against asserted version tables whose rows represent both non-current and current states of persistent objects must be nearly as good as query performance against non-temporal tables with an equivalent number of rows.

Comments:

- Queries which return temporal data, or a mix of temporal and current data, must return equivalent-sized results in an equivalent amount of elapsed time. Chapter 15 explains how we believe this is possible.
- Differences in maintenance performance will be greater than differences in query performance because one logical unit of work – the insertion, update or deletion

of business data about one object – will affect only one row in a conventional table, but one update or deletion to an asserted version table will usually affect several rows.

seamless access, usability aspect

Description: access to both current and non-current states of persistent objects which is just as easy for the data consumer to obtain as is access to only current states.

Comments:

- Temporal data must be available on-line, just as current data is. Transactions to maintain temporal data must be as easy to write as are transactions to maintain current data. Queries to retrieve temporal data, or a combination of temporal and current data, must be as easy to write as are queries to retrieve current data only. (From Chapter 1.)

semantic logfile

Semantics: the set of all past assertions and empty assertions in an asserted version table.

Comments:

- See *physical logfile*.
- The contents of a physical logfile of a particular table, as of point in time X, are all those rows physically present in the table as of that point in time. The contents of a semantic logfile of that table, as of that point in time, are all those rows asserted on or prior to that point in time. The difference is the set of all assertions which are deferred assertions as of that point in time.

Components: asserted version table, assertion, empty assertion.

shared assertion time

Mechanics: the shared assertion time of two or more versions are all those assertion time clock ticks included in both their assertion time periods.

Semantics: the shared assertion time of two or more versions is the assertion time period within which they are commensurable.

Components: assertion time, assertion time period, clock tick, (in)commensurable, version.

state

Semantics: the set of values in the business data columns of a row in an asserted version table which describes the properties and/or relationships which the object represented by that row has at a point in time or over a period of time.

Components: asserted version table, business data, object, point in time, represent, time period.

statement

Mechanics: a currently asserted row in an asserted version table.

Semantics: what is asserted, during a specified period of assertion time, is true of a referenced object during a specified period of effective time.

Comments:

- So in Asserted Versioning, a row in past assertion time is a record of a statement we once made, and a row in future assertion time is a record of a statement that we may make at some point in the future. Neither are statements.

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

successor

Mechanics: a row in an asserted version table that supercedes all or part of another row.

supercede

Mechanics: a row X supercedes a row Y if and only if X and Y both represent the same object, X's effective time period [intersects] Y's effective time period, X's business data is not identical to Y's business data, and X's assertion time period [meets^{-1}] Y's assertion time period.

Semantics: a row X supercedes a row Y if and only if X and Y both represent the same object, and X is a business-data different assertion about what Y is like during all of part of the effective time period specified by Y.

Comments:

- See also: *withdraw*, *replace*.
- A row X supercedes a row Y if and only if X says something new about what the object Y represents is like, during all or part of the effective time period specified by Y.
- If a replacement version was also created as part of the temporal update transaction which created this superceding version, then this superceding version will [meet^{-1}] that replacement version in effective time.

Components: Allen relationship [intersect], Allen relationship [meets^{-1}], assertion time period, business data, effective time period, object, represent.

tabular data

Mechanics: a collection of data structured as rows and columns.

Semantics: a collection of data in which the collection itself represents a type of object, and whose contents represent one or more properties and/or relationships of one or more instances of that type.

Comments:

- Besides DBMS tables, files and their records are tabular data, as are the rows and columns in spreadsheets.

Components: instance, object, type.

target episode

Mechanics: an episode that a temporal transaction will create, delete or modify.

Comments:

- There can be more than one target episode for a temporal update or delete. A temporal insert can insert only one episode.

Components: episode, temporal transaction.

target row

Mechanics: a row in an asserted version table that a temporal transaction will create, delete or modify.

Comments:

- There can be more than one target row for a temporal update or delete. A temporal insert can insert only one row.

Components: asserted version table, temporal transaction.

target span

Mechanics: the effective time period specified on a temporal transaction.

Semantics: the time period into which a temporal insert transaction will place a representation of an object, within which a temporal update transaction will modify existing representations of an object, or from which a temporal delete transaction will remove the representation of an object.

Components: effective time period, object, represent, time period, temporal delete transaction, temporal insert transaction, temporal transaction, temporal update transaction.

target table

Mechanics: the table specified on a temporal transaction.

Comments:

- Temporal transactions have one and only one target table, even though temporal delete transactions can modify multiple tables.

Components: temporal transaction.

taxonomy

Mechanics: an acyclic hierarchy, in which each child node is a KIND-OF its parent node, and in which the collection of child nodes under a common parent are jointly exhaustive and mutually exclusive. (From Chapter 2.)

Semantics: a partitioned semantic hierarchy.

Comments:

- We leave KIND-OF as formally undefined, i.e. as part of our controlled vocabulary of primitive terms. When X is a KIND-OF Y, it follows that every instance of X is also an instance of Y, and that this is so because of what "X" and "Y" mean.

Components: N/A.

TEI

See *temporal entity integrity*.

temporal container

Description: a spatial metaphor for the relationship of data to a time period, or for the relationship of effective time to assertion time.

temporal data

Semantics: data about the past, present and future states of objects, and/or about our past, present and future assertions that what that data says is true.

Comments:

- See *explicitly temporal data*, *implicitly temporal data*.

Components: assertion, object, state.

temporal data management taxonomy

Description: a taxonomy of methods for managing temporal data, developed by the authors and presented in From Chapter 2.

temporal data management taxonomy, (bi-temporal data)

Description: any method of managing state temporal data in two temporal dimensions.

Components: temporal data management taxonomy (state temporal data), temporal dimension.

temporal data management taxonomy, (event temporal data)

Description: any method of managing queryable temporal data that keeps track of changes to an object by recording the initial state of an object, and then by keeping a history of the events in which the object changed. (From Chapter 2.)

Comments:

- An event, once completed, cannot change. If data describing an event needs to be altered, it is because the data is incorrect, not because the event changed.

Components: event, object, state, temporal data management taxonomy (queryable temporal data), temporal dimension.

temporal data management taxonomy, (queryable temporal data)

Mechanics: any method of managing temporal data that does not require manipulation of the data before it can be queried. (From Chapter 2)

Components: temporal data.

temporal data management taxonomy, (reconstructable temporal data)

Description: any method of managing temporal data that requires manipulation of the data before it can be queried. (From Chapter 2.)

Components: temporal data.

temporal data management taxonomy, (state temporal data)

Description: any method of managing queryable temporal data that keeps track of the states of things as they change over time.

Comments:

- As an object changes from one state to the next, we store the before-image of the current state, and update a copy of that state, not the original. The update becomes the new current state of the object.
- When managing time using state data, what we record are not transactions, but rather the *results* of transactions, the rows resulting from inserts and (logical) deletes, and the rows representing both a before- and an after-image of every update. (From Chapter 2.)

Components: state, temporal data management taxonomy (queryable temporal data).

temporal data management taxonomy, (temporal data best practices)

Description: as described in Chapter 4, best practices in managing temporal data concern themselves with versioning, i.e. with keeping track of the changes to objects of interest by recording the states which those objects pass through as they change.

Components: object, state, temporal data, versioning.

temporal data management taxonomy, (temporal data management)

Description: any method of managing temporal data, at the table and row level, by means of explicit temporal schemas and constraints on the instances of those schemas.

Comments:

- Thus, for example, data warehouses and data marts are not part of this taxonomy because they are methods of managing temporal data at the database level.

Components: temporal data.

temporal data management taxonomy, (the alternative temporal model)

Description: a method of managing uni-temporal versioned tables at the column as well as at the row level, using transformations not specifiable with current SQL, that are based on various composition and decomposition operations defined in other publications by Dr. Nikos Lorentzos.

Comments:

- The temporal model described by C. J. Date, Hugh Darwen and Nikos Lorentzos in their book Temporal Data and the Relational Model. (Morgan-Kaufmann, 2002).

Components: uni-temporal, versioned table.

temporal data management taxonomy, (the Asserted Versioning temporal model)

Description: a method of managing bi-temporal data, using transformations specifiable with current SQL, that manages each row in a temporal table as the assertion of a version of an episode of an object.

Comments:

- The temporal model described in this book.
- Distinguished from the alternative temporal model in particular (i) in all the ways it is distinguished from the standard temporal model, and also (ii) by its recognition and treatment of assertion tables and bi-temporal tables and (iii) its decision to not manage temporal data at the column level.
- Distinguished from the standard temporal model in particular by providing design and maintenance encapsulation, managing data located in future assertion time, its reliance on episodes as managed objects, and its internalization of adjunct datasets.

Components: assertion, episode, object, temporal data management taxonomy (bi-temporal data), temporal table, version.

temporal data management taxonomy, (the standard temporal model)

Description: a method of managing bi-temporal data, using transformations specifiable with SQL available in 2000, that manages each row in a temporal table as a row in a conventional table which has been assigned a transaction time period, a valid time period, or both.

Comments:

- The temporal model described by Dr. Rick Snodgrass in his book Developing Time-Oriented Database Applications in SQL (Morgan-Kaufmann, 2000).

Components: conventional table, temporal data management taxonomy (bi-temporal data), temporal table, transaction time, valid time.

temporal data management taxonomy, (uni-temporal data)

Description: any method of managing state temporal data in a single temporal dimension.

Comments:

- Thus versioning, in any of its forms, is a method of managing uni-temporal data.

Components: temporal data management taxonomy (state temporal data), temporal dimension.

temporal database

Mechanics: a database that contains at least one table whose rows include one or more columns representing an assertion and/or effective time period.

Semantics: a database at least one of whose tables is explicitly temporal.

Components: assertion time period, effective time period.

temporal date

Mechanics: a date which is either a begin date or an end date.

Semantics: a date which delimits a bi-temporal time period.

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

temporal default values

Mechanics: the values for the assertion time period and effective time period which the AVF assigns to a temporal transaction unless those values are specified on the transaction itself.

Comments:

- Those values are Now() for the assertion and effective begin dates, and 9999 for the assertion and effective end dates.

Components: assertion time period, AVF, effective time period, temporal transaction.

temporal delete cascade

Mechanics: a temporal delete transaction which removes all dependent child data from the transaction timespan specified on the temporal delete transaction.

Semantics:

Comments:

- a temporal delete cascade will attempt to remove both the parent object, and all its dependent children, from the clock ticks specified in the transaction. (From Chapter 11.)

Components: temporal delete transaction, transaction timespan.

temporal delete transaction

Mechanics: a temporal transaction against an asserted version table which removes business data representing an object from one or more contiguous or non-contiguous effective-time clock ticks.

Comments:
- A temporal delete is like a temporal update except that it specifies that every version or part of a version of the designated object that falls, wholly or partially, within that target span will be, in current assertion time, removed from that target effective timespan. (From Chapter 9.)
- A temporal delete withdraws its target object from one or more effective time clock ticks. In the process, it may {erase} an entire episode from current assertion time, or {split} an episode or a version in two, or {shorten} an episode or a version either forwards or backwards, or do several of these things to one or more episodes and/or versions with one and the same transaction.

Components: asserted version table, business data, effective time, object, represent, temporal transaction.

temporal dimension

Semantics: a type of time within which points in time and/or periods of time are ordered.

Components: type, point in time, time period.

temporal entity integrity

Mechanics: (i) for episodes, the constraint that no two episodes of the same object, in the same period of assertion time, either [meet] or [intersect]; (ii) for versions within an episode, the constraint that each effective-time adjacent pair of versions [meet] but do not [intersect].

Semantics: the constraint that, in any clock tick of assertion time, no clock tick of effective time is occupied by more than one representation of an object.

Comments:
- One of the two constraints by means of which Asserted Versioning expresses the semantics of bi-temporal data.

Components: Allen relationship [meet], Allen relationship [intersect], assertion time, episode, object, temporally adjacent, version.

temporal extent

Semantics: the number of clock ticks in a time period.

Components: clock tick, time period.

temporal extent state transformation

Mechanics: a transformation to an asserted version table in which, within a given period of assertion time, the number of effective-time clock ticks which a given episode occupies is increased or decreased.

Semantics: a transformation altering the temporal extent of an episode's effective time period, within a given period of assertion time.

Comments:

- A temporal extent transformation alters the total number of effective-time clock ticks in which a given object is represented. But these transformations act on episodes, not directly on objects.
- In the definitions of the temporal extent state transformations, the phrase "in a given period of assertion time" will be present implicitly.

Components: asserted version table, assertion time, clock tick, effective time, episode, occupied.

temporal extent state transformation taxonomy

Description: a taxonomy of additions to or deletions from the set of clock ticks which contain a representation of an object in an asserted version table, developed by the authors and presented in From Chapter 9.

temporal extent state transformation taxonomy, {create}

Mechanics: the temporal extent state transformation that adds the representation of an object to an effective time period that is either [before] or [before-1] the effective time period of any episode of that object already present in the table.

Semantics: the temporal extent state transformation that adds a new episode to an asserted version table.

Components: asserted version table, effective time, episode, object, represent.

temporal extent state transformation taxonomy, {erase}.

Mechanics: the temporal extent state transformation that, in a given period of assertion time, removes the representation of an object from an effective time period that is either [before] or [before^{-1}] all other representations of that same object.

Semantics: the temporal extent state transformation that withdraws an entire episode from current assertion time.

Components: Allen relationship taxonomy [before], Allen relationship taxonomy [before^{-1}], assertion time, effective time, object, represent.

temporal extent state transformation taxonomy, {lengthen backwards}

Mechanics: the temporal extent state transformation that adds the representation of an object to an effective time period that [meets] the effective time period of an episode of that object already present in the table.

Semantics: the temporal extent state transformation that expands the effective time period of an episode into a contiguous, earlier time period.

Components: Allen relationship [meets], effective time period, episode, object, representation.

temporal extent state transformation taxonomy, {lengthen forwards}

Mechanics: the temporal extent state transformation that adds the representation of an object to an effective time period that [meets^{-1}] the effective time period of an episode of that object already present in the table.

Semantics: the temporal extent state transformation that expands the effective time period of an episode into a contiguous, later time period.

Components: Allen relationship [meets^{-1}], effective time period, episode, object, representation.

temporal extent state transformation taxonomy, {lengthen}

Mechanics: the temporal extent state transformation that increases the number of clock ticks within the effective time period of an episode.

Semantics: the temporal extent state transformation that enlarges the effective time period of an episode.

Components: clock tick, effective time period, episode, object.

temporal extent state transformation taxonomy, {merge}

Mechanics: the temporal extent state transformation that adds the representation of an object to an effective time period that [meets^{-1}] the effective time period of an earlier episode of that object, and that also [meets] the effective time period of a later episode of that object.

Semantics: the temporal extent state transformation that transforms two adjacent episodes of the same object into one episode.

Components: effective time period, episode, Allen relationship [meets], Allen relationship [meets^{-1}], object, representation, temporally adjacent.

temporal extent state transformation taxonomy, {modify}

Mechanics: those temporal extent state transformations that increase or decrease the number of clock ticks occupied by an object, but that neither increase nor decrease the number of episodes that represent that object.

Semantics: those temporal extent state transformations that add or remove clock ticks from one or more episodes.

Components: clock tick, episode, object, represent.

temporal extent state transformation taxonomy, {shorten backwards}

Mechanics: the temporal extent state transformation that removes the representation of an object from one or more contiguous clock ticks that were the latest clock ticks of an episode of that object.

Semantics: the temporal extent state transformation that removes the effective time period of an episode from a later time period.

Comments:

Components: clock tick, contiguous, effective time, episode, object, representation.

temporal extent state transformation taxonomy, {shorten forwards}

Mechanics: the temporal extent state transformation that removes the representation of an object from one or more contiguous clock ticks that were the earliest clock ticks of an episode of that object.

Semantics: the temporal extent state transformation that removes the effective time period of an episode from an earlier time period.

Comments:

Components: clock tick, effective time, episode, object, representation.

temporal extent state transformation taxonomy, {shorten}

Mechanics: the temporal extent state transformation that reduces the number of clock ticks within the effective time period of an episode of that object.

Semantics: the temporal extent state transformation that reduces the effective time period of an episode.

Components: clock tick, effective time period, episode, object.

temporal extent state transformation taxonomy, {split}

Mechanics: the temporal extent state transformation that removes the representation of an object from one or more contiguous clock ticks that were neither the earliest nor the latest clock ticks of an episode of that object already present in the table.

Semantics: the temporal extent state transformation that transforms one episode into two adjacent episodes of the same object.

Components: clock tick, contiguous, episode, object, representation, temporally adjacent.

temporal foreign key

Mechanics: a non-primary key column of an asserted version table which contains the unique identifier of an object on which the object represented by its own row in an asserted version table is existence dependent.

Semantics: a column which designates an object on which the object represented by the row which contains it is existence dependent.

Comments:

- At the schema level, a TFK points from one table to a table it is dependent on. But at the row level, it points from one row, which is a version, to a group of one or more rows which make up an episode of the object whose *oid* matches the oid value in that temporal foreign key.
- At the instance level, a TFK guarantees that there is an episode of the designated object in the referenced table, and that the effective time period of that episode in the referenced table includes, ([fills^{-1}]) the effective time period of the version which contains the referring TFK. (From Chapter 6.)
- Temporal foreign keys, like conventional foreign keys, are the managed object construct which represents the dependency of a child object on a parent object.

Components: asserted version table, existence dependency, object, object identifier, represent.

temporal gap

Mechanics: the existence of at least one unoccupied clock tick between two time periods for the same object.

Components: clock tick, object, occupy, time period.

temporal gap versioning

Mechanics: a form of versioning similar to logical delete versioning, but in which both a version begin date and a version end date are used to delimit the time period of the version.

Semantics: a form of versioning in which versions of the same object may or may not be contiguous, and in which no version is physically deleted.

Comments:

- Logical delete versioning is not part of Asserted Versioning. See Chapter 4.
- See also: *basic versioning, logical delete versioning, effective time versioning*.

Components: contiguous, logical delete versioning, object, time period, version, version begin date, version end date.

temporal insert

Mechanics: a temporal transaction against an asserted version table which creates a single-row episode of the object specified in the transaction.

Semantics: a temporal transaction against an asserted version table which places business data representing an object into an effective time period that is not contiguous with any other clock ticks in which the same object is represented.

Comments:

- A temporal inserts adds a representation of its specified object to a series of one or more contiguous effective time clock ticks. In the process, it may {create} an entire single-episode version, or {merge} two adjacent versions (thus merging their two episodes), or {lengthen} a version either forwards or backwards, or do several of these things with one and the same transaction.

Components: asserted version table, business data, clock tick, contiguous, effective time period, episode, object, represent, temporal transaction, temporally adjacent.

temporal parameter

Mechanics: one of the three asserted versioning dates that can be specified on a temporal transaction, those being the effective begin date, the effective end date and the assertion begin date.

Semantics: the means by which an assertion time period and an effective time period is defined on a temporal transaction.

Components: assertion begin date, assertion time period, effective begin date, effective end date, temporal transaction.

temporal primary key

Mechanics: the unique identifier of a row in an asserted version table, made up of (i) an object identifier (oid), (ii) an effective begin date, and an assertion begin date.

Semantics: the unique identifier of a row in a bi-temporal table, made up of (i) the unique identifier of a persistent object, (ii) a unique designation of an effective time period, and (iii) a unique designation of an assertion time period.

Components: asserted version table, bi-temporal, effective begin date, effective time period, assertion begin date, assertion time period, object identifier, oid, persistent object.

temporal referential integrity

Mechanics: the constraint that for every asserted version row which contains a temporal foreign key, there is an episode of the object which that temporal foreign key references such that, within shared assertion time, the effective time period of that episode includes ([fills^{-1}]) the effective time period of that asserted version row.

Semantics: the constraint that, in any clock tick of assertion time, every clock tick that is occupied by a representation of a child object is also occupied by one representation of each of its parent objects.

Comments:

- A TRI relationship between a child managed object and a parent managed object is based on an existence dependency between the objects which those managed objects represent. (From Chapter 11.)

Components: Allen relationship [fill^{-1}], asserted version table, assertion time, child object, clock tick, effective time period, episode, include, object, occupy, parent object, reference, shared assertion time, temporal foreign key.

temporal tag

Description: a metaphor for the association of a row of data with a time period. The time period is a temporal tag applied to the row.

temporal transaction

Mechanics: an insert, update or delete transaction whose target is an asserted version table.

Semantics: a insertion, update or deletion of a temporally delimited assertion of a statement describing an object as it exists during a specified period of effective time.

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

temporal update

Mechanics: a temporal transaction against an asserted version table which modifies business data representing an object in one or more contiguous or non-contiguous effective-time clock ticks.

Semantics: a temporal transaction against an asserted version table which changes one or more business data items describing that object in one or more clock ticks included in the transaction's specified period of effective time.

Comments:

- Note that a temporal update will change the business data for an object in occupied clock ticks, and will ignore unoccupied clock ticks. Thus the clock ticks that a temporal update affects are not necessarily contiguous clock ticks. And consequently, to be valid, it is not necessary that all clock ticks in the effective-time range of a temporal update be occupied by the specified object. It is only necessary that at least one of them be occupied.

Components: asserted version table, business data, clock tick, contiguous, effective time period, object, represent, temporal transaction.

temporalize

Mechanics: to temporalize a managed object is to associate an explicit assertion time period and/or an explicit effective time period with it.

Components: assertion time period, effective time period, managed object.

temporalized extension of the Closed World Assumption

Mechanics: the constraint that a temporal transaction cannot insert data into past assertion time, update data in past assertion time, or delete data from past assertion time.

Semantics: the assumption that if a statement was not represented in the database at time T_1, then at time T_1 we did not make that statement.

Comments:

- Note, by contrast, that a temporal transaction can insert data into past effective time, update data in past effective time, and delete data from past effective time.
- In chapter 12, we explained the reason that we can modify the statement content of past effective time but not of past assertion time. We said: ". . . . . a belief is expressed by the presence of a row in a table. No row, no belief. So if we write a transaction today that creates a row stating that we believed something yesterday, we are creating a row that states that we believed something at a time when there was no row to represent that belief. . . . . . (But) it would be a logical contradiction to state that we had such a belief at a point or period in time during which there was no row to represent that belief."

Components: past assertion time, statement, temporal transaction.

temporally adjacent

Mechanics: two rows in an asserted version table are temporally adjacent if and only if they have the same object identifier and, in shared assertion time, have no other rows with the same object identifier whose effective time period is later than that of the earlier row and earlier than that of the later row.

Semantics: temporally adjacent rows are rows representing the same object that, in shared assertion time, have no rows representing that same object between them in effective time.

Comments:

- If two rows in an asserted version table are adjacent, then either one is [before] the other, or one [meets] the other. This is because they would otherwise violate the temporal entity integrity constraint, which the AVF prevents.
- If one row in an asserted version table is adjacent to and [before] the other, then they belong to different episodes.
- If one row in an asserted version is adjacent to and [meets] the other, then they belong to the same episode.

Components: asserted version table, effective time period, object, object identifier, represent, shared assertion time.

temporally contiguous

Mechanics: two time periods occupied by the same object are temporally contiguous just in case they share no clock ticks and there are no clock ticks between them.

Components: clock tick, object, occupy, time period.

temporally delimited

Semantics: to be restricted to a time period.

Comments:

- A temporal transaction is temporally delimited to its effective time period within its assertion time period.
- Note that these time periods can, and often do, remain current until further notice, i.e. that the can, and often do, end in 9999.

Components: time period.

terminate

Mechanics: to withdraw the latest version of an episode and replace it with a version identical to it except that it has as an effective end date the date specified on a temporal delete transaction.

Semantics: to set an effective end date for an episode.

Comments:

- The termination of an episode withdraws that episode from some but not all of the effective-time clock ticks which it occupies. If the episode is withdrawn from *all* of its effective-time clock ticks, it is {erased} from current assertion time. See *temporal extent state transformation {erase}*.
- The termination of an episode should be thought of as the by-product of a temporal delete transaction, not as that transaction's semantic objective. The semantic objective of a temporal delete transaction is to remove the representation of an object from all clock ticks within the effective timespan specified on the transaction. If that effective timespan on the delete transaction includes the entire episode, then we say that the episode has been deleted, not that it has been terminated.

Components: effective end date, episode, replace, temporal delete transaction, version, withdraw.

TFK

See *temporal foreign key*.

thing

Semantics: things are what *exist* through time, and can change over time. (From Chapter 2.)

Comments:

- See *object, event*.

time period

Mechanics: a continuous length of either effective or assertion time, with a known begin date.

Semantics: a series of one or more contiguous clock ticks in either effective or assertion time, whose initial clock tick has a known value.

Comments:

- If the end date of a time period is not known, 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".

Components: assertion time, begin date, contiguous, clock tick, effective time.

timeslice

Mechanics: an object as it exists during a specified closed effective time period.

Semantics: a closed effective period of time of an object.

Comments:

- A timeslice of an object represented in an asserted version table does not have to align on episode or version boundaries. It is just a continuous period of time in the life history of an object (or in the projection of that life history into the future.

Components: closed effective time, object.

timespan

Mechanics: the period of time specified on a temporal transaction.

Semantics:

Comments:

Components:

transaction

Mechanics: (i) a row in a transaction table; or (ii) an insert, update or delete to a database.

Semantics: (i) data which is the record of an event; or (ii) the transformation of database.

Comments:

- The first sense designates a row of data that represents an event. For example, a customer purchase is an event, represented by a row in a sales table; the receipt of a shipment is an event, represented by a row in a receipts table. In this sense, transactions are what are collected in the fact tables of fact-dimension data marts.
- The second sense designates any insert, update or delete applied to a database. For example, it is an insert transaction that creates a new customer record, an update transaction that changes a customer's name, and a delete transaction that removes a customer from the database. (From Chapter 2.)
- (In any formalization of this Glossary, of course, this homonym would have to be resolved. In this book, we rely on context to do so.)

Components: event.

transaction begin date

Mechanics: in the standard temporal model, the date a row is physically inserted into a table.

Semantics: in the standard temporal model, the date which designates the start of the transaction time period of a row, using the closed-open convention.

Comments:

- Another one of the several homonyms of "transaction".

Components: closed-open, temporal data management taxonomy {the standard temporal model}, transaction time period.

transaction table

Semantics: a table whose rows represent events.

Comments:

- Transaction tables record the *events* that change the states of objects and, in particular, the relationships among them. (From Chapter 1.)
- Transaction tables are often used as the fact tables in fact-dimension data marts.
- There can be only one version of an event, since events do not persist and change over time. Multiple rows for the same event can only be multiple assertions about that event, presumably a series of corrections to the data.

Components: events.

transaction time

Description: "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." From [Jensen, 1992].

Comments:

- As defined by Jensen, the computer science term "transaction time" may be treated as co-extensive with our term "row creation time". But the two terms cannot be said to be *synonymous* because they are defined by means of two vocabularies between which semantic correlations have not been established. Also, while Jensen's definition, given here, indicates that transaction time is a point in time, Snodgrass' use of the term, in [Snodgrass, 2000] has it referring to a period of time, usually open-ended, but not necessarily so. In this second sense, the term is co-extensive with a proper subset of our term "assertion time period".

transaction time period

Mechanics: the assertion and effective time periods specified on a temporal transaction.

Semantics: the set of assertion time and effective time clock ticks outside of which a temporal transaction will have no effect on the database.

Comments:

- In this entry, "transaction" refers to "temporal transaction", not to any of the other homonyms of "transaction".

Components: assertion time period, clock tick, effective time period, temporal transaction.

transaction timespan

See *transaction time period*.

TRI

See *temporal referential integrity*.

TSQL2

Description: a temporal extension to the SQL-92 language standard, by Dr. Rick Snodgrass and others, first published in March 1994 in the *ACM SIGMOD Record*. A final version was published the following September.

Comments:

- This standard has not been adopted by the SQL standards committee.

type

Semantics: a kind of thing.

Comments:

- See *instance*.
- In a database, tables represent types of things, and rows represent instances of those types.

Components: thing.

uni-temporal data

Mechanics: data which has either an assertion time period or an effective time period, but not both.

Semantics: data which has one and only one explicitly expressed time period.

Comments:

- We say "explicitly expressed" to emphasize that all data is bi-temporal. In a conventional table, in which no assertion and/or time period columns are included, each row is nonetheless bi-temporal. Each row is a current assertion about what the object it represents is currently like. As a description of its object, each row's assertion time period and effective time period is the single clock tick Now(). As a claim that its object exists, each row's assertion and effective time periods are co-extensive with the row's physical presence in its table.

Components: assertion time period, effective time period, time period.

uni-temporal table

Mechanics: a uni-temporal assertion table or a uni-temporal version table.

Semantics: a table with a single explicitly expressed temporal dimension.

Comments:

- Uni-temporal version tables are common in business databases; Chapter 4 describes their major variants. But see *update in place* for a common misuse of this kind of table.
- Uni-temporal assertion tables are single-table logfiles. They are usually described as the history table companions to conventional tables. However, history tables are often designed with a primary key which is the same as the primary key of the table whose changes they track, but with the addition of a low-order date (or timestamp) to that primary key. In general, history tables like this are not

semantically complete uni-temporal assertion tables, being unable, for example, to distinguish entries which represent a delete from those which represent an update.

Components: assertion, uni-temporal, temporal dimension, version.

unreliable business key

Mechanics: a business key which cannot be used to match data on a temporal transaction to one or more rows in the target table for that transaction.

Semantics: a business key which may represent more than one object.

Comments:

- The distinction between reliable and unreliable business keys can be seen at work in the description of the match logic for Asserted Versioning's temporal transactions, in From Chapter 9.

Components: business key, object, represent, target table, temporal transaction.

until further notice

Mechanics: an assertion time period or an effective time period whose end date is 9999 (the highest temporal value the DBMS can represent).

Semantics: an assertion time period or an effective time period whose end date is unknown, but which is interpreted to be in the future.

Comments:

- In general, this term means "unknown but presumed valid". In a SQL Server asserted version table, a row is asserted until further notice if and only if its assertion end date is 12/31/9999, and is in effect until further notice if and only if its effective end date is 12/31/9999.
- Mechanically, a time period with a begin date in the past, and that is presumed to be current until further notice, will be interpreted by the DBMS in that way because in any query, Now() will *always* be greater than that begin date and less than that end date.
- Neither 12/31/9999, nor any other DBMS representation of 9999, is valid as an assertion begin date or effective begin date.
- If 12/31/9999 (or any other DBMS representation of 9999 is used in a business data column, of course, it has whatever semantics the business chooses to give to it.

Components: 9999, assertion time period, effective time period, end date.

update in place

Mechanics: to update data on a row by overwriting it.

Comments:

- In a conventional table, all updates are updates in place, and therefore all updates destroy historical information. An update which reflects a change in the object represented by a row of data has the effect of replacing a current version with a new current version, thus destroying effective-time history. An update which reflects a correction to a mistake made in the data has the effect of replacing a current assertion with a new current assertion, thus destroying assertion-time history.
- And so, in a uni-temporal table, either one or the other of those two types of updates must be done as an update in place, thus destroying one or the other of those two kinds of history, or else the two kinds of update are not distinguished and both result in the creation of a new row of data. But this second way of managing uni-temporal tables makes it impossible to distinguish true versions, i.e. rows which are part of the effective-time history of an object, from corrections to bad data. All too frequently, in business databases, this second way of managing uni-temporal tables is called versioning, and the rows in those tables are called versions. Interpreted in this way, these tables are themselves mistaken data, and provide incorrect information to their business consumers.

Components: N/A.

valid time

Description: the computer science term "valid time" may be treated as co-extensive with our term "effective time".

Comments:

- "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." From [Jensen, 1992]

version

Mechanics: a row in an asserted version table statement which makes a statement about what the object it represents is like during a specified effective time period.

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

Comments:

- Every row in an asserted version table is either a past, present or future version representing an object.

Components: asserted version table, effective time period, object, represent, statement.

version begin date

Description: the date on which a row in a best practices version table begins.

Comments:

- This expression does *not* apply to a row in an asserted version table. A row in an asserted version table is a version, and the begin date associated with that row, as a version, is its effective begin date. The version begin date of a row in any other kind of version table may represent either the physical date on which the row was created, or a logical date on which the version becomes effective.

version end date

Description: the date on which a row in a best practices version table ends.

Comments:

- This expression does *not* apply to a row in an asserted version table. A row in an asserted version table is a version, and the begin date associated with that row, as a version, is its effective begin date. The version end date of a row in any other kind of version table may represent either the physical date on which a delete transaction was applied to the row, or a logical date on which the version ceased to be in effect.

version split

Mechanics: a process in which the representation of an object is removed from one or more effective-time clock ticks that [fill] the effective time period of a version.

Semantics: a process in which one version is withdrawn, and replaced by two versions.

Comments:

- When a version is split, the earlier half becomes a new version which is located [before] the latter half. Because the two versions are not contiguous, the result of splitting a version is always to split an episode. See *temporal extent state transformation {split}*.
- Although a version split always results in an episode split, the reverse is not the case.

Components: clock tick, effective time, Allen relationship [fill], object, represent, version, withdraw.

version table

Mechanics: a uni-temporal table whose explicitly represented time is effective time.

Semantics: a uni-temporal table each of whose rows is a current assertion about what its object was, is or will be like during the specified period of effective time.

Comments:

- To be semantically valid, a uni-temporal version table *must* correct errors in data by overwriting those errors. Frequently, businesses have uni-temporal tables which they think are version tables, but in which every update results in a new row in the table. But this means that the begin, or begin and end dates, of those rows are not true version dates. They are just dates of physical database activity. If the update reflected a change in the object being represented, then the version date or dates have the semantics of effective dates. But if the update reflected a correction to bad data, then the version date or dates have the semantics of assertion dates. By mixing both kinds of updates, the semantics are destroyed, and we don't know what any row in the table is really telling us.
- An asserted version table is one kind of version table. IT best practices have given rise to many other kinds of version tables, which we grouped into four main types in From Chapter 4. What the standard temporal model calls a *valid-time table* is another kind of version table.

Components: effective time, object, uni-temporal.

version(ed) data

Description: data that describes what its object is like during a stated period of time.

Comments:

- What kind of period of time is involved depends on the kind of version table. In many best practice implementations of versioned tables, unfortunately, effective time and assertion time are not distinguished. An update that reflects a change to the object represented in the table results in a new version, but an update that reflects a correction to erroneous data also results in a new version. And in both cases, the version begin date is simply the date the new row was physically created. Nearly all best practice support for versioned data is semantically incomplete. But when effective time changes are not clearly distinguished from error correction changes, those implementations are semantically flawed, and the data thus supported is semantically ambiguous.

withdraw

Mechanics: to set the assertion end date of a row in an asserted version table to Now().

Semantics: to move an asserted version row into past assertion time.

Comments:

- See also *replace*, *supercede*.

- See also *lock*.
- Non-deferred update and delete transactions sets the end date of versions they will then replace and/or supercede to Now() which, upon the conclusion of the temporal transaction, immediately becomes a past date. This is a "clearing the decks" transformation which removes the representation of the object affected by the transaction from current assertion time.
- A temporal transaction cannot set a row's assertion end date to a past date because, if it did, it would create a contradiction. During the time period from that past date to the time the transaction took place, the database asserted that row. But after Now(), the record of that assertion is erased from the database. Just as we cannot retroactively begin an assertion, we cannot retroactively end one, either. This is another manifestation of the *temporalized extension of the Closed World Assumption*.
- If a temporal transaction sets a row's assertion end date to a future date, it locks that row, making it a closed version. If that row is the effective-time last row in an episode, it makes that episode a closed episode. Only deferred transactions can set a row's assertion end date to a future date.
- Thus, no temporal transaction can set a row's assertion end date to a past date. Deferred transactions set that date to a future date. Non-deferred transactions set that date to Now(), and then as soon as the transaction is complete, that row is in past assertion time.

Components: 9999, assertion end date, asserted version table, past assertion time.