ST USM: Bridging the Semantic Gap with a Spatio-Temporal Conceptual Model

Vijay Khatri, Sudha Ram and Richard T. Snodgrass

November 14, 2001

TR-64

A TIMECENTER Technical Report

Title ST USM: Bridging the Semantic Gap
with a Spatio-Temporal Conceptual Model

Copyright © 2001 Vijay Khatri, Sudha Ram and Richard T. Snodgrass. All rights reserved.

Author(s) Vijay Khatri, Sudha Ram and Richard T. Snodgrass

Publication History November 2001. A TIMECENTER Technical Report.

TIMECENTER Participants

Aalborg University, Denmark
Christian S. Jensen (codirector), Michael H. Böhlen, Heidi Gregersen, Dieter Pfoser,
Simonas Šaltenis, Janne Skyt, Giedrius Slivinskas, Kristian Torp

University of Arizona, USA
Richard T. Snodgrass (codirector), Dengfeng Gao, Vijay Khatri, Bongki Moon, Sudha Ram

Individual participants
Curtis E. Dyreson, Washington State University, USA
Fabio Grandi, University of Bologna, Italy
Nick Kline, Microsoft, USA
Gerhard Knolmayer, Universty of Bern, Switzerland
Thomas Myrach, Universty of Bern, Switzerland
Kwang W. Nam, Chungbuk National University, Korea
Mario A. Nascimento, University of Alberta, Canada
John F. Roddick, University of South Australia, Australia
Keun H. Ryu, Chungbuk National University, Korea
Michael D. Soo, amazon.com, USA
Andreas Steiner, TimeConsult, Switzerland
Vassilis Tsotras, University of California, Riverside, USA
Jef Wijsen, University of Mons-Hainaut, Belgium
Carlo Zaniolo, University of California, Los Angeles, USA

For additional information, see The TIMECENTER Homepage:
URL: <http://www.cs.auc.dk/TimeCenter>

Any software made available via TIMECENTER *is provided "as is" and without any express or implied warranties, including, without limitation, the implied warranty of merchantability and fitness for a particular purpose.*

The TIMECENTER icon on the cover combines two "arrows." These "arrows" are letters in the so-called *Rune* alphabet used one millennium ago by the Vikings, as well as by their precedessors and successors. The Rune alphabet (second phase) has 16 letters, all of which have angular shapes and lack horizontal lines because the primary storage medium was wood. Runes may also be found on jewelry, tools, and weapons and were perceived by many as having magic, hidden powers.

The two Rune arrows in the icon denote "T" and "C," respectively.

Abstract

The representation of geospatial phenomena in databases is one of the key issues in applications like geo-marketing, environmental modeling, transportation planning and geology. For these geospatial applications, there is a need for abstract concepts that would bridge the conceptual gap between the real world and its spatio-temporal representation in the computer systems. To capture the semantics related to space and time in a conceptual schema, we propose an annotation-based approach that allows a database designer to focus first on the non-temporal and non-spatial aspects of the application, and subsequently augment the schema with spatio-temporal annotations. In this paper, we apply our annotation-based approach to the Unifying Semantic Model (USM) to propose the Spatio-Temporal Unifying Semantic Model (ST USM). ST USM is an upward-compatible, snapshot reducible, annotation-based spatio-temporal conceptual model that can comprehensively capture the semantics related to space and time without adding any new spatio-temporal constructs. We provide the formal semantics of ST USM via a mapping to conventional USM and constraints, from which the logical schema can be derived. We illustrate practical aspects related to our spatio-temporal design methodology with a hydrogeologic application at the United States Geological Survey (USGS) and a web-based design tool, thereby demonstrating simplicity and comprehensiveness to spatio-temporal conceptual modeling.

1 Introduction

Geographic Information Systems (GIS) are increasingly employed in a wide array of applications including social, environmental and economic studies [85], e.g., land information systems, environmental modeling, resource management, transportation planning, geo-marketing, geology and archaeology [41]. Underlying all these applications is temporal, spatial and time-varying spatial data, collectively referred to as *spatio-temporal* data. Consequently, many DBMS vendors are incorporating capabilities to manage spatial (e.g., Oracle Spatial [47] and Informix Geodetic DataBlade [67]) and temporal (e.g., Oracle Time Series Cartridge [49], Informix TimeSeries DataBlade [71]) data. Conceptual database design is widely recognized as an important step in the development of database applications [4, 16, 70] including geospatial applications described above. During conceptual database design, a conceptual model provides a notation and formalism that can be used to construct a high-level description of the real world. In this paper, we integrate conceptual modeling and spatio-temporal concepts to realize an annotation-based spatio-temporal conceptual model, the Spatio-Temporal Unifying Semantic Model (ST USM).

Geographic information is defined as having a theme (i.e., the phenomenon or object being observed), the *location* of the phenomenon and the *time* related to the phenomenon [72]. Conventional conceptual models (e.g., [11, 16, 31, 56]) do not provide a mechanism to explicitly capture the semantics related to space and time; as a result, the users' spatio-temporal data requirements cannot be adequately captured during conceptual design. Consequently, many researchers [10, 20, 22, 23, 42, 46] cite the need for abstract concepts to describe geographic data objects and their operations in databases. These concepts would bridge the conceptual gap between an aspect of the real world—sometimes referred to as the *miniworld* [16]—and its spatio-temporal representation in the computer systems. In this paper, we propose an annotation-based approach that divides spatio-temporal conceptual design into two steps: (i) capture the *current* reality of an application using a conventional conceptual model *without considering the spatial and temporal aspects* and only then (ii) annotate the schema with the spatio-temporal semantics of the application. Rather than creating any new constructs in a spatio-temporal conceptual model, we use annotations to capture the spatio-temporal aspects of the application. Our annotation-based approach is generic and can be applied to any conventional conceptual model to transform it into a spatio-temporal conceptual model. In this paper, we apply our annotation-based approach to the Unifying Semantic Model (USM) [60]—an extended version of the Entity-Relationship (ER) Model [11]—to propose ST USM. ST USM is graphical, annotation-based, *snapshot reducible* [73] and *upward compatible* [7] with the conventional USM, and does not introduce any special spatio-temporal constructs.

Our work makes several contributions. First, we have integrated and extended prior research related to spatio-temporal conceptual modeling. We propose a small intuitive ontology-based grammar for annotation, which comprehensively captures the semantics related to space and time. We show how to represent the spatio-temporal semantics related to, e.g., *event* and *state* [32], *valid time* and *transaction time* [76], *existence time* [32], *temporal granularities* [5, 6, 14], *valid time indeterminacy* [15], *shape* and *position* [13], *spatial resolution* and *imprecision* [86, 87], and *change in position* and/or *shape* over time [58, 82]; we have described the semantics related to granularities and indeterminacy in detail elsewhere [35]. With our approach, we have integrated the semantics of space and time into a traditional conceptual model. Since the various types of conceptual modeling abstractions are orthogonal to space and time, our annotations are generic and applicable to all types of conceptual modeling abstractions.

Second, our annotation-based approach naturally extends the conventional conceptual model without increasing complexity from the perspective of users, data analysts and CASE (Computer-Aided Software/Systems Engineering) tool vendors. Over the last two decades, conceptual models based on the ER Model [11] have been well accepted with users, data analysts and CASE tool vendors; additionally, there is a natural and well-understood method of translating ER schemas to subsequent relational schemas [16, 70]. However, many of the extant spatio-temporal conceptual models drift away from the core ER Models, and add distinct and unique constructs to capture spatio-temporal requirements. These new constructs require that the data analyst be cognizant of the new proposed formalism. Since such a clean-slate approach does not build on the widely adopted and researched conventional conceptual models (e.g., [4, 11, 16]), the CASE tool vendors need to develop their database design tools from scratch. With our annotation-based approach, the tool vendors can enhance their existing design tools to support spatiality and temporality with minimum changes. Additionally, the data analysts do not need to know about any additional spatio-temporal constructs. With our annotation-based approach, we claim to have achieved both simplicity and comprehensiveness in spatio-temporal conceptual modeling.

Third, our two-step approach takes into account how humans store and use geographic knowledge. One of the deficiencies of the existing conceptual models that can represent geographic phenomena is their inability to "represent information in way that is more natural to humans" [45, 57]. Mennis et al. [45] cite that humans cognitively store *what*, *where* and *when* data in separate knowledge structures. Our two-step approach that first focuses on facts (*what*) and then on the context (*where* and *when*) related to the facts corresponds to Anderson and Bower's Human Associative Memory Model [3], which segregates assertions into facts and context related to facts. Thus, our annotation-based approach integrates human cognition of geographic phenomena with the traditional conceptual modeling formalism.

Fourth, to underscore the practical focus of our approach, we describe an on-going hydrogeologic application at the United States Geological Survey (USGS) and a web-based modeling tool [61]. This tool allows the data analyst to: (i) develop a non-spatial and non-temporal schema; (ii) augment the schema with spatio-temporal annotation; and (iii) convert the *annotated schema* (in this paper, an *ST USM schema*) to a *translated schema* with explicit representation of spatio-temporal concepts, thus explicating the semantics associated with the annotations. While the ST USM schema developed at the end of step (ii) is used for capturing the user requirements and forms the basis for communication with the user, the schema at the end of step (iii) is used for subsequent translation to a logical schema. Thus, a schema developed at the end of steps (ii) and (iii) represent the equivalent semantics, for humans and computers, respectively.

Fifth, our annotation-based approach is straightforward to extend. While we posit that the spatio-temporal annotation presented in this paper is comprehensive, it is impossible to assert completeness with conceptual modeling because any formalism is motivated in part by pragmatic rather than purely theoretical reasons. It is possible that the formalism presented in this paper may need to be extended for a database application. In such a case, the annotations presented in this paper can be easily extended. We also allow some aspects of the annotations to be user-defined. For example, users have the flexibility to define their own temporal granularity; however, they must specify the semantics of the user-defined granularity. Since spatio-temporal annotation is orthogonal to the conceptual modeling abstractions, our annotation-based approach is not only generic but also straightforward to extend.

In summary, our annotation-based approach is based on ontological concepts related to space and time, is intuitive and straightforward for both users and data analysts, integrates with the existing conventional conceptual and relational models, provides a mechanism to capture the semantics related to user-defined granularities and indeterminacy, is practical to implement, and neatly dovetails into the existing database design paradigm.

The rest of the paper is organized as follows. In Section 2, we motivate the need for a spatio-temporal conceptual model using an application at the USGS. We then summarize the spatio-temporal requirements based on an ontology related to temporal and spatial concepts. In Section 4, we describe key design issues associated with our annotation-based spatio-temporal conceptual design methodology, which first focuses on "what" is important for an application in the real world and then associates "what" with "when/where." Consequently, Section 5 and Section 6 describe a mechanism to capture the "what" and "when/where" semantics. We summarize a conventional conceptual model, USM, which provides various abstractions to capture "what" is important for an application. We apply our annotation-based approach to USM realizing ST USM, which captures the semantics related to "when/where." We present the syntax and the associated semantics of ST USM in terms of USM and constraints specified in first-order logic. In Section 7, we apply our annotation-based approach to the hydrogeologic application, described in Section 2. Continuing with the hydrogeologic application, we employ an example in Section 8 to outline the mappings from ST USM to a relational model that supports spatial abstractions. We round out the paper with a comparison with extant spatio-temporal conceptual models and describe our future research directions.

2 Motivation

The representation of geographic phenomena in databases is one of the central issues in GIS [45]. Conceptual modeling [11, 16, 31, 56] takes the user requirements as an input and transforms them into a high-level conceptual schema. A good conceptual schema acts as glue between users, data analysts and the database implementation. While designing a database, a data analyst may use the conceptual schema as a communication tool with the user. The schema represents the structure of the data manipulated by the application and serves as the system metadata. Since conceptual design is not dependent on the implementation environment, the conceptual schema is relatively static compared to the current dynamic technology environment. Thus, the conceptual schema may be utilized in the event of technology upgrades and transfer. A conceptual model provides a formalism to develop this conceptual schema. A precursor to designing, developing and evaluating a spatio-temporal conceptual model is identifying the problems that need to be solved. In this section, we motivate the requirements associated with a spatio-temporal conceptual model using a ground-water application at USGS.

We are working with a group of researchers who are developing a ground-water flow model [12] for the Death Valley region. Beneath the earth's surface, there is a zone where all interstices are filled with water referred to as ground water. In arid regions like the Death Valley, ground water provides a large percentage of water for domestic, industrial and agricultural uses. Ground water is stored in voids, spaces, and cracks between particles of soil, sand, gravel, rock, or other earth materials. The objective of the ground-water flow model is to characterize regional 3D ground-water flow paths so that policy-makers can make decisions related to radio-nuclide contaminant transport, and the impact of ground water pumping on national parks and local communities in the region. However, the quality of the model output and the predictions based on these models are dependent on the data that forms an input to these models.

A large part of the input data for this model is spatial and/or temporal in nature. For example, two key objects of interest in the application are spring-water sites and borehole sites. Both of these objects need to be spatially referenced to the Earth. A spring-water site is represented as a point whose location on the surface of the earth is given by geographic *x* and *y* co-ordinates, with a spatial granularity of degree. Spring-water sites are the points where spring discharge is measured. Geographically spring-water sites exist within a spring and there can be many spring-water sites within a spring (i.e., many points where discharge is measured). Similar to springs, wells (or boreholes) are access points to the ground water system. Information of their construction and condition are important for monitoring ground water supply and remediation. A borehole site refers to a part of the well whose 3D location is given by *x*- and *y*- co-ordinates on the earth’s surface with a spatial granularity of degree, and depth below land surface in spatial granularity of foot; there can be different borehole sites at different depths at the same surface location. Physically, a well is composed of hole-intervals with different diameters. A well can also be thought to be composed of a set of casing and opening. A casing is a section of a well with concrete, steel or plastic installed on the well. An opening is a section of a well that is open to allow water flow. Additionally, a borehole site may have access tubes associated with them that give access to a section of the borehole. Casing, opening, hole interval and access tubes define the characteristics of a well, and the water-level measurements taken at the borehole site are influenced by these characteristics.

The primary input data for the ground-water flow model includes discharge (in cubic feet per second) at the spring-water site and water depth (in feet below land surface) at the borehole site, which is collected by a source agency. Discharge and water depth need to be associated with the time of measurement. Additionally, the time associated with discharge and water depth data needs to be captured to the temporal granularity of minute. The researchers evaluate the collected water level and discharge measurements to decide which of them will be included as an input for the ground-water flow model. The measurements used as an input to the model are often referred to as io-water-level (i.e., input-output water level) and io-discharge (i.e., input-output discharge), respectively. There are various hydraulic tests conducted at the borehole site; the results of these tests need to be coupled with the time (in minute) when the tests were conducted. Additionally, borehole sites may have a pumplift that removes water from the borehole site and this can affect other data collected at the borehole site.

Capturing requirements related to, e.g., spring-water site, borehole site, well, casing, water level, source agency, has many associated issues. The existing conceptual models provide limited modeling support to capture the spatio-temporal data requirements. For example, there is a need for a mechanism to represent temporal user-defined composites (e.g., io-water-level), spatial aggregates (e.g., well is composed of casing and opening), and spatial and temporal constraints (e.g., borehole site is located within a well). Additionally, there does not exist a mechanism to capture the semantics related to temporal and spatial granularities (e.g., degree and second). A conceptual schema is used by users, data analysts and database implementation; thus, it should be formally defined so that it is interpretable by machines, be straightforward to be understood by the users and integrate with the existing database design methodologies for data analysts.

Overall, a proposed spatio-temporal conceptual model should: (i) contain minimal constructs that comprehensively capture the semantics associated with spatiality and temporality; (ii) provide a framework for expressing the structure of spatio-temporal data which is easily understood and communicated to the users; (iii) be easily translated into implementation-dependent logical models; (iv) be upward-compatible with the existing non-spatial and non-temporal conceptual models so that it does not invalidate the existing conventional schemas; (v) include a mechanism to represent multiple granularities in a conceptual schema; (vi) provide a formalism related to indeterminacy associated with the spatial and temporal data; and (vii) allow the data analyst to model non-spatial, non-temporal, spatial, temporal and time-varying spatial aspects of the application in a straightforward manner.

3 Desiderata

Geographic applications require data referenced by geographic co-ordinates [79] with time sometimes being referred to as the fourth dimension. We summarize key temporal, spatial and time-varying spatial terminology in this section.

In this paper, temporal, spatial and time-varying spatial aspects are collectively referred to as *spatio-temporal* aspects of the application. The definition of time domain, temporal granularity and granularity relationships are based on existing time glossaries [5, 32].

3.1 Temporal Ontology

A *time domain* is denoted by the pair (T, ≤), where T is a nonempty set of *time instants* and "≤" is the total order on T. We can assume the time domain is either *discrete* or *dense*. An *instant* is a time point on the time line. For example, ($\mathbf{Z}$, ≤) represents a discrete time domain where instants are isomorphic to integers implying that every instant has a unique successor. The time between two instants is referred to as a *time period*. An unanchored contiguous portion of the time line is called a *time interval*, e.g., one day. An interval is relative while an instant is absolute [75]. A finite union of non-overlapping time periods are referred as a *temporal element*. A non-decomposable time interval of fixed minimal duration is referred to as a *chronon*. Relationships between temporal intervals can be described with thirteen Allen's [2] predicates: *before*, $before^{-1}$, *meets*, $meets^{-1}$, *overlaps*, $overlaps^{-1}$, *finished*, $finished^{-1}$, *during*, $during^{-1}$, *starts*, $starts^{-1}$ and *equals*.

Temporal granularity is an integral part of temporal data and is defined as a mapping TG from *index* i to subsets of the time domain [14]. Thus, temporal granularity defines a countable set of non-decomposable granules $TG(i)$ that can be composed of a set of contiguous instants or non-contiguous instants; additionally, a special granule called the *origin*, $TG(0)$ is non-empty. Some examples of temporal granularities are Gregorian-day, business-day and business-week. While Gregorian-day is a temporal granularity with contiguous granules of day, business-day is not. Each non-empty granule may have a textual representation termed a *label* (e.g., "2001-10-5"), which can be mapped to the index integer with a mapping called the *label mapping*. The earliest time domain element in the origin is referred to as an *anchor* with respect to the time domain. The union of granules is called an *image* of a temporal granularity. The smallest interval of the time domain that contains the image of a temporal granularity is called the *extent* of that granularity. *Scale* and *cast* are operations that convert time values between granularities [14]. The function $scale(g, TH)$ takes an instant g (i.e., $l_{TG} \sim u_{TG}$) in a granularity TG and returns the smallest instant $l_{TH} \sim u_{TH}$ in a granularity TH such that $l_{TG} \sim u_{TG} \subseteq l_{TH} \sim u_{TH}$, where u_{TG} (*upper support*) and l_{TG} (*lower support*) are indexes that refer to the minimum and maximum granule of the granularity TG within which an *indeterminate instant* is located. The event *might* have occurred after l_{TG} and *definitely* occurred by u_{TG}; l_{TG} and u_{TG}, thus, correspond to the Lipski bounds [43]. Similarly, the function $cast(g, TH)$ scales the instant and then returns the index corresponding to the first granule. Thus, the cast operation returns a determinate instant when applied to a determinate instant. For example, $scale(2000_{\text{year}}, \text{month}) = 2000\text{-}01_{\text{month}} \sim 2000\text{-}12_{\text{month}}$ and $cast(2000_{\text{year}}, \text{month}) = 2000\text{-}01_{\text{month}}$.

Facts can interact with time in two orthogonal ways resulting in *transaction time* and *valid time* [76]. Valid time denotes when the fact is true in the real world and implies the storage of histories related to facts. On the other hand, transaction time links an object to the time it is current in the database and implies the storage of versions of a database object. The versions of entity instances (or their attributes) are referred to as *extension versioning* and versioning of the definitions of the entities is termed as *schema versioning* [74]. *Existence time*, which applies to an object, is the valid time when the object exists [25]; it is also referred to as a *lifespan* [32] of the object. While the temporal granularity can be specified for existence time and valid time, that for transaction time is system-defined. The transaction time has duration from insertion to (logical) deletion [34] and can include granules only up to the current time granule in the real world. A transaction timestamp includes *Until Changed* (*UC*), which is a special transaction time marker. If the transaction time includes *UC*, the associated fact is current in the database. An uninterpreted time value is referred to as *user-defined time* [34]. Time-varying data may be modeled as an *event* or a *state* [33]. An event occurs at a point of time, i.e., an event has no duration. In other words, an event is instantaneous and its occurrence results in a fact becoming true. A state has duration, e.g., a storm occurred from 5:07 PM to 5:46 PM. An event and a state are associated with a temporal granularity.

There are three kinds of temporal statements: *current*, *temporal sequenced* and *temporal non-sequenced*, which can be associated with different kinds of time (i.e., valid-time and transaction time) and different types of temporal data types (i.e., instant, interval and period) [75]. Current statements refer to *now* (e.g., the *current query*: "What is the *current* water-depth at the specified borehole?" or the *current constraint*: "Two different borehole sites cannot be associated with the same pumplift now"). Temporal sequenced statements refer to each point of time (represented by a granularity index); e.g., the *temporal sequenced query*: "What has been the water-depth at the specified borehole each day since 2001-01-03?" or the *temporal sequenced constraint*: "Every day within the lifespan of a borehole is associated with a test for leakance and diffusivity." Temporal non-sequenced statements do not treat time specially or reference all points of time; e.g., the *temporal non-sequenced query*: "At the specified borehole, when was the water-depth greater than the maximum depth recorded on 2000-03-04?" or the *temporal non-sequenced constraint*: "A pumplift that is assigned to a borehole site cannot be reassigned to another pumplift."

3.2 Spatial Ontology

Any data that can be associated to a location on the Earth is referred to as geographic data [13]. Geographic space based on Euclidean geometry is the basis for most GISs [44]. In this case, the location can be expressed by a set of co-ordinates, e.g., latitude and longitude. We briefly describe concepts related to space, spatial granularity and spatial relationships. The definitions of spatial granularity and indeterminacy—based on [86, 87]—are described in detail in [35].

The space domain may be represented as a set (e.g., $\mathbf{R}^3$, $\mathbf{R}^2$, $\mathbf{N}^3$, $\mathbf{N}^2$) with elements referred to as *points*. For geographic applications, horizontal space is segregated from vertical space; correspondingly, we define *horizontal* and *vertical spatial granularities*. Intuitively, the horizontal space domain corresponds to the Earth's surface while vertical space domain corresponds to the depth/height below/above sea level. We define horizontal spatial granularity as a mapping from integers to any partition of horizontal space; the partition may arise from pixellation of space and may be a regular square or any other shape like triangular irregular network (TIN) or even irregular shapes (e.g., county).

Examples of horizontal spatial granularities are dms-deg, dms-min and county. For granularities like dms-deg, space is partitioned along two perpendicular directions and the granularity is construed to be dms-deg along both the dimensions. County is an example of an irregular horizontal spatial granularity.

A spatial object is associated with *geometry* and *position*. Geometry represents the shape and size of an object [13]. The position in space is based on co-ordinates in a mathematically-defined reference system, e.g., latitude and longitude. Geometry of the spatial object may be 0-, 1- or 2- dimensional corresponding to a *point*, a *line* and a *region*. A point is "a zero-dimensional spatial object with co-ordinates", a line is "a sequence of ordered points, where the beginning of the line may have a special start node and the end a special end node" and a region or polygon consists of "one outer and zero or more inner rings" [84]. A line is a one-dimensional geometric primitive and can be classified as a *line segment*, a *line string* or a *line arc* [13]. A line described by two sets of co-ordinates and the shortest connection between them is referred to as a line segment. An ordered sequence of sets of co-ordinates and the shortest connection between them is called a line string. A line arc is an ordered sequence of co-ordinates and connections that are defined by a set of mathematical functions. David et al. [13] define an area as "a bounded continuous two-dimensional geometric primitive delimited by one outer non-intersecting boundary and zero or more nested non-intersecting inner boundaries." They differentiate between a line and a region—the line itself is "the carrier" of the information while in a region, the area is of primary importance and the "boundary is secondary…to limit the area." A point is an object whose location and not extent is of relevance, a line describes a facility for moving through space and a region is an abstraction where both the location and extent are important [19].

Similar to temporal sequenced and non-sequenced statements [75], there are *spatial sequenced* and *spatial non-sequenced statements*. Spatial sequenced statements refer to each point of space (represented by spatial granularity index) within the specified geometry; e.g., the *spatial sequenced query*: "For a specified borehole, what are the lithologies (description of rock composition and texture) along its depth, from the top to the bottom of the borehole?" or the *spatial sequenced constraint*: "Each point within the depth of a borehole is associated with a lithology." Spatial non-sequenced statements do not treat space specially or reference all points of space within geometry; e.g., the *spatial non-sequenced query*: "Is sandstone (a type of lithology) associated within a specified borehole at any depth?" and *spatial non-sequenced constraint*: "If a borehole has lithology *x* then it cannot have lithology *y* under it."

3.3 Time-Varying Spatial Ontology

Geography is different from geometry in the following sense: in geography, space is indivisibly coupled with time [55]. Lately, there has been much interest in adding a temporal aspect to geographic databases [24] since time integrates human activity, orders events and separates cause from effect [85]. Three types of interaction between an object and space-time are possible [58, 82]: (i) moving objects, i.e., its position changes continuously but the shape does not (e.g., a car moving on a road network); (ii) objects whose spatial characteristics and position change with time discretely, i.e., changing shape (e.g., change of the shape of land parcels in a cadastral application); and,

(iii) integration of the above two behaviors, i.e., continuous moving and changing phenomena (e.g., modeling a storm). Similarly, Erwig et al. [19] state that spatio-temporal databases are essentially "about moving objects." Three types of abstractions of spatial objects include a point, a line and a region; while a point and a line may "*move* over time", a region can change its location (i.e., move) and change its shape.

Having summarized the temporal, spatial and time-varying spatial semantics that need to be captured in a conceptual model, we describe the design issues for capturing these spatio-temporal semantics during conceptual modeling.

4 Design Issues in Spatio-temporal Conceptual Modeling

Our spatio-temporal conceptual design methodology employs annotations, a generic methodology that can be applied to any conceptual model to translate it to a spatio-temporal conceptual model. We discuss key design issues related to our annotation-based approach to conceptual modeling: upward compatibility, snapshot reducibility, treatment of space vs. time, annotation syntax, orthogonality and cognition.

4.1 Upward Compatibility

Upward compatibility [7] implies the ability to render conventional conceptual schemas spatio-temporal without affecting the legacy schemas. The objective of upward compatibility is to be able to develop spatio-temporal schemas without invalidating extant legacy schemas, thus, protecting the investments in existing schemas. It also implies that both the legacy schemas and the spatio-temporal schemas can co-exist. If the spatio-temporal extension is a strict superset provided by adding non-mandatory semantics, it would ensure that the spatio-temporal extension is upward compatible with conventional conceptual model. Upward compatibility requires that the syntax and semantics of traditional conceptual model remain unaltered. Unlike some of the proposed temporal conceptual models (e.g., [17, 18]) that change the semantics of existing constructs, our approach preserves the semantics of the conventional conceptual model. Additionally, our annotation syntax (Appendix B) includes ϵ implying that annotations are non-mandatory.

4.2 Snapshot Reducibility

Snapshot reducibility [7, 73] implies "natural" generalization of the syntax and semantics of extant conventional conceptual models for incorporating the spatio-temporal extension. Snapshot reducibility ensures that the semantics of spatio-temporal model are understandable in terms of the semantics of conventional conceptual model. Here, the overall objective is to help ensure minimum additional investment in data analyst/user training. As we assume that the data analysts will be conversant with conventional conceptual models, an extension using annotations should require minimum additional training costs, fewer errors and no significant drop in productivity.

With annotations, our approach naturally extends the semantics of a conventional conceptual model (with implicit snapshot semantics), thereby, inducing the spatio-temporal sequenced semantics; removing the annotations renders the schema with traditional snapshot semantics. For example, in a conventional conceptual model a *key attribute* [16] uniquely identifies an entity (at a point in time). A *temporal key* [75] implies uniqueness *at each point in time*. As may be evident the semantics of a temporal key are implied by the semantics of a key in a conventional conceptual model.

4.3 Space vs. Time

Objects, their properties and relationships between the objects are predominantly embedded in time; we may or may not choose to capture the associated temporality in an application. On the other hand, some objects, e.g., LITHOLOGY, are not inherently geo-referenced. For example, the notion of sandstone—a type of lithology—is independent of where it is found. As with Abraham and Roddick [2], we construe the time-varying spatial aspect as a lineage of the spatial aspects.

4.4 Annotation Syntax

Our spatio-temporal design methodology uses annotations to capture the temporal, spatial and time-varying spatial aspects of the application; the annotations are based on ontology described in section 3.

Our annotation syntax was influenced by that of spatio-temporal queries [83], which in turn was based on notations for queuing systems [37]. As shown in Appendix B, the overall structure of an *annotation phrase* is:

⟨temporal annotation⟩ // ⟨spatial annotation⟩ // ⟨time-varying spatial annotation⟩

The temporal annotations, spatial annotations and time-varying spatial annotations are each separated by a double forward slash (//).

The temporal annotation first specifies the existence time (or valid time) followed by the transaction time. The temporal annotation for existence time and transaction time is segregated by a forward slash (/). Any of these aspects can be specified as not being relevant to the associated conceptual construct by using "-". The valid time or existence time can be modeled as an event (E) or a state (S) and has an associated temporal granularity. For example, an annotation phrase "S (day) / - //" associated with a class PUMPLIFT succinctly denotes that a pumplift has an associated existence time with a temporal granularity of day (day) and the lifespan of entities is modeled as states (S); no transaction time is recorded for entities in this class. Similarly, "S (min) / T //" denotes that entities of the class DISCHARGE exist in a bitemporal space. The temporal granularity of the states (S) is minute (min). Additionally, we also need to capture transaction time (T) associated with DISCHARGE. In this second example, the granularity associated with transaction time is not specified as it is system-defined.

The spatial annotation includes the geometry and position in *x*-, *y*- and *z*-dimension; each dimension is segregated by a forward slash (/). For example, "// P(deg) / P(deg) / -" for SPRING_SITE describes a spatial entity with a geometry of points in the *x-y* plane. The associated horizontal spatial granularity is degree.

The interaction between an object and space-time can result in change in the shape and/or change in the position of an object. A time-varying spatial annotation can be specified only if spatial and temporal annotation have already been specified. For example, "E(sec)/-// P(deg)/ P(deg)/-//Pos@xy" denotes that entities in the entity class have only time-varying position while the shape is time-invariant. The geometry of the entities is a point (P) in an *x-y* plane with a spatial granularity of degree. The position of the entity changes in the *x-y* plane (Pos@xy) over time and each geometry is valid for time granules (E) measured in second. Our annotation also includes formalism to model indeterminacy; e.g., an indeterminate state with probability distribution function [15] is designated as S~. Often the probability distribution is not known and a user makes a simplified assumption of a uniform distribution; in that case indeterminate state is represented as S+-. Details related to modeling spatio-temporal granularities and indeterminacy can be found elsewhere [35].

In specifying the annotation syntax, we had to make certain choices related to what to include in the annotation syntax. These choices are not peculiar to our annotation syntax. Even conventional conceptual models do not include every possible constraint in the schema; e.g., typically *uniqueness constraints* on an attribute are not shown in the schema and may be specified in the *data dictionary*, an organized listing of data elements [59, 88]. Similarly, non-sequenced constraints are not represented in the schema primarily for pragmatic reasons; the schema will cease to be easy to use (readable). For example, a *lifetime key constraint* (a type of temporal non-sequenced constraint) may require uniqueness over the entire lifespan of the entity; note that this constraint is orthogonal to the temporal sequenced key constraint. Non-sequenced constraints may be defined; however, non-sequenced constraints are not explicitly shown in the schema and may be included in, e.g., a data dictionary.

4.5 Orthogonality: "What", "When" and "Where"

Our annotation-based approach integrates the semantics related to space and time into a traditional conceptual model, e.g., [4, 11, 16], without adding any special constructs. Since various types of conceptual modeling abstractions (e.g., *entity*, *attribute*, *relationship* and *key*) are orthogonal to space and time, the annotations are generic and applicable to all types of conceptual modeling abstractions. In other words, our annotations (Appendix B) are not specific to any construct in a conventional conceptual model; they apply to all the constructs. For example, an annotation phrase "S(day)/-//" can apply to an entity class, an attribute (including key, composite and multi-valued) or a relationship.

4.6 Human Cognition of Spatio-temporal Data

Since a conceptual schema acts as a communication device between users, data analysts and the database implementation, the formalism provided by a conceptual model for developing a schema should be comprehensible and straightforward to use for the users and data analysts. Many prior studies have compared conventional conceptual models for comprehension (e.g., [1, 9, 69]) and perceived ease of use (e.g., [30, 36, 68]); these studies reinforce the fact that many significant underlying requirements related to a conceptual modeling formalism are user-related. Further, one of the problems with the extant spatio-temporal conceptual models is reported to be their inability to "represent information in way that is more natural to humans" [45, 57]. Therefore, it is imperative to propose a spatio-temporal modeling formalism that takes into account human cognition, this would ensure that the proposed model is comprehensible and straightforward to use.

Some researchers, e.g., [3, 65], posit that all human knowledge is stored as abstract conceptual propositions. The *propositions* are assertions about the real world. As shown in Figure 1, Anderson and Bower's [3] Human Associative Model (HAM) is based on propositions; in their model, a proposition is composed of a *fact* and *context* associated with the fact. The *subject* and *predicate* correspond to a topic and a comment about the topic; this corresponds to information representation as object-property or property-value pairs. For some applications, the *context* in which the fact is true can be the key to reasoning about the mini-world. This context in turn is composed of *time* and *location* associated with the fact.

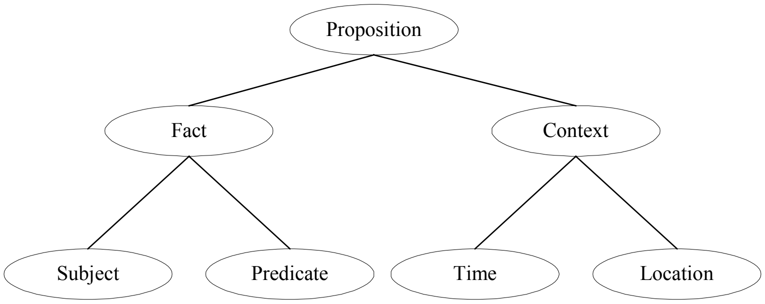


Figure 1: Human Associative Memory Model [3]

Our two-step approach to spatio-temporal database design that first focuses on facts (*what*) and then on the context (*where* and *when*) related to the facts corresponds to Anderson and Bower's Human Associative Memory Model [3]. Thus, we expect our annotation-based approach will lead to schemas that are comprehensible and straightforward to use.

Having outlined the underlying philosophy of our spatio-temporal conceptual modeling approach, we apply our methodology to a conventional conceptual model, USM, to propose a spatio-temporal conceptual model called ST USM. We have used USM as the base model because it provides a formalism to comprehensively model "what" is important in the real world. However, our annotation-based approach is not specific to USM and can be applied to any conventional conceptual model. In the next two sections, we describe the two steps of our spatio-temporal conceptual modeling methodology via USM and ST USM.

5 USM: Representing “what”

The various types of abstractions supported by typical conceptual models, e.g., [4, 11, 16, 60, 70], include classification, association, aggregation and generalization/specialization. The underlying principle of these abstractions is selective emphasis of detail. A conventional conceptual model represents “what” is important in the real world; USM [60] is an extended version of the Entity Relationship model [10] that provides various abstractions to capture the semantics of “what” is important for an application. We summarize below the key terms and terminology related to conventional conceptual modeling, specifically USM. Figure 2 illustrates a USM schema that represents “what” is important for the hydrogeologic application described in Section 2. A summary of the graphical notations used is provided in Appendix A.

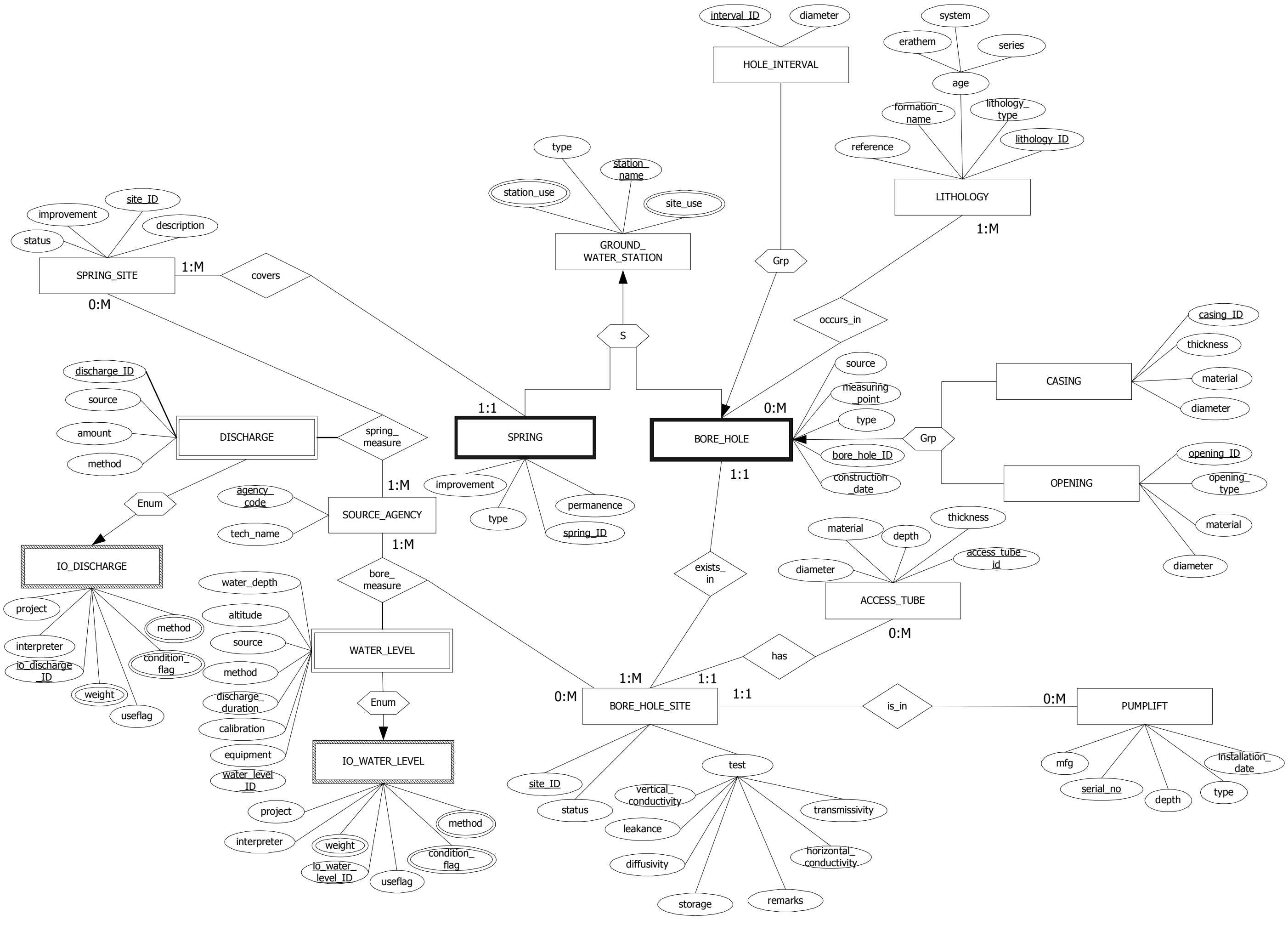


Figure 2: The USM schema for ground-water flow model without considering temporal and spatial aspects

5.1 Class and Attribute

All real world objects are referred to by the term *entity*. A collection of entities for which common characteristics are to be modeled is called an *entity class*. Characteristics or properties of entities are called *attributes* (A_i, where $i = 1,\ldots, n$) of the class. Each attribute defined for an entity class has associated with it an *attribute domain* ($dom(A_i)$), which is the set of values that an entity can take for the attribute. Thus, an *entity class* may be defined as $E = \cup_i (A_i, dom(A_i))$. The set of instantiations of an entity class is referred to as an *entity set*. In other words, an entity e of an entity class E may be designated as $e(E)$ and a set of entities of an entity class is represented as $S(E)$ where $e(E) \in S(E)$. For example in Figure 2, PUMPLIFT is an entity class which has attributes like serial number (serial_no), manufacturer (mfg), type and installation date (installation_date).

Attributes are created by *property relationships*. Associations between or among members of entity classes are called *class relationships*. If an attribute is created by a property relationship, its attribute domain is a set of values drawn from a *domain class*. A domain class is a pool of values from which one or more attributes draw their values. An attribute created by a class relationship refers to members in some other entity class. In this case, the domain of the attribute is a set of entities belonging to the other entity class. USM distinguishes between *simple* and *constructed* entity classes. In simple entity classes all attributes are created by property relationships, whereas in constructed classes there are one or more attributes that are created by class relationships. Constructed classes are used to model entities built from other entities of the database application.

In the following subsections, we describe several class relationships: interaction, generalization/specialization, composite and grouping.

5.2 Interaction Relationship

An *interaction relationship* refers members of one entity class to members of one or more entity classes. Formally, let R be an interaction relationship and $E_1, E_2,\ldots, E_n$ be classes that participate in the relationship. A relationship may be considered to be a subset of the Cartesian product $S(E_1) \times S(E_2) \times\ldots\times S(E_n)$: a relationship instance r_i consists of exactly one entity from each participating entity set. An instance of an *interaction class* associated with an interaction relationship may be described as $(e_1, e_2,\ldots, e_n, attr_1, attr_2,\ldots, attr_m)$ where $e_i \in S(E_i)$ $(i = 1\ldots n)$ and $attr_1, attr_2,\ldots, attr_m$ are attributes that describe the relationship. For example in Figure 2, is_in is an interaction relationship between BORE_HOLE_SITE and PUMPLIFT. Each instance of is_in includes an entity from S(BORE_HOLE_SITE) and an entity from S(PUMPLIFT); is_in does not include any interaction attributes and does not have an associated interaction class. The interaction relationship spring_measure includes interaction attributes source, amount and method, which are represented as attributes of an interaction class WATER_LEVEL.

5.3 Generalization/Specialization Relationship

Generalization is a form of abstraction in which similar objects are related to a higher-level generic object; the constituent objects may be considered as the *specialization* of the generic object [8]. A generalization proceeds from the recognition that a number of entity classes share some common features. Based on their commonality, a generalization synthesizes these entity classes into a single, higher-level entity class, the generalization class. The crucial property of higher and lower level entities created by specialization and generalization is *attribute inheritance*; the attributes of higher-level entity classes are said to be *inherited* by the lower-level entity classes [70]. Subclasses may be *attribute-defined* or *roster-defined*. In the former, an entity in the superclass has an attribute whose value determines its subclass. For a roster-defined subclass, a membership in the subclass is decided by the user on examining the roster.

The generalization/specialization relationship is a subset of the Cartesian product $S(S_1) \times \ldots \times S(S_m) \times S(P_1) \times \ldots \times S(P_n)$ over subclasses $S_1, S_2, \ldots, S_m$ and superclasses $P_1, P_2, \ldots, P_n$. Here, S_i is both a *subset* and *subtype* of P_j, where $1 \leq i \leq m$ and $1 \leq j \leq n$. The notion of subset is based on set membership, i.e., S_i is a subset of P_j if every member of S_i also belongs to P_j. The notion of subtype is extended from its conventional definition [66]; class S_i is a subtype of class P_j if and only if:

1. every attribute defined for P_j is also defined for S_i,
2. the domain of every attribute of S_i is a subset of the domain of the corresponding attribute in P_j and
3. all members of S_i share common characteristics, such as having a common value for one or more attributes of P_j, or satisfying a user-specified criteria, so that a member of P_j having such characteristics must also belong to S_i, whereas a member of P_j without such characteristics must not belong to S_i.

For example in Figure 2, a GROUND_WATER_STATION is a superclass with SPRING and BORE_HOLE as its subclasses. Certain common attributes in a GROUND_WATER_STATION apply to both SPRING and BORE_HOLE: station_name, site_use, station_use and type. Attributes like permanence and improvement are specific to a SPRING, and construction_date and measuring_point are attributes specific to BORE_HOLE. The subclasses SPRING and BORE_HOLE are based on the value of the attribute, type, of superclass GROUND_WATER_STATION, i.e., these subclasses are attribute-defined.

5.4 Composite Relationship

A composite relationship defines a new class called the *composite class* that has another class (or subsets of a class) as its members. A composite relationship is similar to "power set grouping" in [66], in that they both represent a set whose members are subsets of the base class, E; in other words, a composite relationship may be represented as $\mathbf{P}(S(E))$.

Each member of a composite class, referred as a *composite*, is a subclass of some other class called the *base class*. The USM definition of a composite class requires that each member of it must also be a subtype of the base class. When a composite class is defined, it always has at least one attribute called Contents, which refers to base class members of the composite class. Thus, Contents is a multi-valued attribute; however it may or may not be unique. Additional attributes may be included to describe each composite [60]. A composite can be defined as (Contents, $attr_1$, $attr_2$,…, $attr_m$), where values the of attribute Contents are subclasses E_1, E_2,…, E_n of class E. For example, IO_DISCHARGE is a composite class with DISCHARGE as its component class. Each input-output discharge (IO_DISCHARGE) used in the ground-water flow model is a composite of DISCHARGE at the spring; i.e., an input-output discharge is composed of possibly several spring discharges. In this example, the component class DISCHARGE is both a subclass and a subtype of the composite class IO_DISCHARGE.

5.5 Grouping Relationship

A grouping relationship defines a new class, called a *grouping class*, whose members are physically or logically made up of members or sets of members from some other entity class(es), called *component classes*. The grouping establishes a "part-of" or "property-of" relationship. USM refers to each member of the grouping class as a *group*.

When a grouping class is defined, it always has at least one attribute called Contents, which is an aggregate of members from each of the component classes. A grouping relationship based on component classes E_1, E_2, …, E_n is denoted by $\mathbf{P}(S(E_1)) \times \mathbf{P}(S(E_2)) \times \ldots \times \mathbf{P}(S(E_n))$ and a group may be defined as (Contents, $attr_1$, $attr_2$, …, $attr_m$) where $attr_1$, $attr_2$, …, $attr_m$ are additional attributes that describe the properties of each member of the grouping class. For example, BORE_HOLE is a grouping class with CASING and OPENING as its component classes. This implies that a set of casing and a set of opening that is "grouped together" forms a borehole. Note that BORE_HOLE is not of the same type as CASING or OPENING.

Table 1 below summarizes the various types of abstractions described in this section.

Level of abstraction	**Description**	**Domain**
Simple Class	A collection of entities that have common properties ($attr_1$, $attr_2$,…, $attr_m$)	Cartesian product of attribute domains: $D_E = Dom_1 \times \ldots \times Dom_m$
Interaction Class	A collection of associations among members of classes E_1,…, E_n and described by the interaction attributes $attr_1$, $attr_2$,…, $attr_m$	Cartesian product of domains of the participating classes and interaction attributes: $D_R = D_{E1} \times \ldots \times D_{En} \times Dom_1 \times \ldots \times Dom_m$
Subclass	A subtype and a subset of each of its super-classes P_1,…, P_n; described by all attributes of its super-classes and its own attributes $attr_1$, $attr_2$,…, $attr_m$, if there are any	Cartesian product of the intersection of domains of its super-classes and domains of subclass attributes $D_S = (D_{P1} \cap \ldots \cap D_{Pn}) \times Dom_1 \times \ldots \times Dom_m$
Composite class	Described by attributes (Contents, $attr_1$, $attr_2$,…, $attr_m$); each member of the composite class is both a subset and a subtype of its base class E	Cartesian product of the powerset of the underlying class and domains of composite attributes: $D_{E^*}=\mathbf{P}(D_E) \times Dom_1 \times \ldots \times Dom_m$
Grouping Class	Described by attributes (Contents, $attr_1$, $attr_2$,…, $attr_m$); the projection of each member of the grouping class G on component class is a subset but not a subtype of the component class(es)	Cartesian product of the power sets of domains of component classes and domains of the grouping attributes: $D_G = \mathbf{P}(D_{E1}) \times \ldots \times \mathbf{P}(D_{En}) \times Dom_1 \times \ldots \times Dom_m$

Table 1: Summary of Various Types of Abstractions

In summary, Figure 2 illustrates the USM schema for "what" is important for the hydrogeologic application textually described in Section 2. Some of the pertinent entity classes for this application are BORE_HOLE_SITE and SPRING_SITE. A BORE_HOLE_SITE exists within a BORE_HOLE. A HOLE_INTERVAL can be grouped (Grp) to form a

BORE_HOLE. Another perspective of BORE_HOLE is that is a grouping (Grp) of CASING and OPENING. DISCHARGE and WATER_LEVEL are measured at SPRING_SITE and BORE_HOLE_SITE, respectively. The users specify (or, enumerate: Enum) the composite of WATER_LEVEL and DISCHARGE resulting in IO_WATER_LEVEL and IO_DISCHARGE that is used as an input for the flow model. Figure 2 also shows constraints like the *cardinality constraint* [16] and the key constraint. The cardinality constraint specifies the number of relationship instances that an entity can participate in; e.g., each pumplift can be associated with exactly one (1:1) borehole site in the is_in relationship. Each borehole site can have a minimum of 0 and a maximum of many (i.e., 0:M) associated pumplift in the relationship is_in. A key attribute is an attribute whose values are used to identify the entity uniquely; key values are distinct for each entity in the entity set. For example, serial_no is a key attribute of PUMPLIFT; it is denoted by underlining the attribute name in the schema. Note that at this point of modeling "what" is pertinent for the hydrogeologic application, spatiality may "creeps in" in the schema. For example, a borehole is associated with at least one and a maximum of many (0:M) lithology (over its depth). While we recommend that the data analyst and the user not think about space or time at this stage of modeling the requirements; at the same time they need not worry about the space or time creeping in the schema, as these will be taken care of in the subsequent step.

In this section, we described briefly the semantics associated with conventional conceptual model, USM, which the base model of ST USM. We next explicate the semantics of annotation-based ST USM using USM and constraints in first-order logic.

6 ST USM: Representing "when" and "where"

A spatio-temporal conceptual model provides a formalism to explicitly capture the temporal, spatial and time-varying spatial semantics of the application. In this section, we present the syntax and semantics of our spatio-temporal annotation. For each construct introduced in Section 5, we show how that construct can be annotated as temporal, spatial and time-varying spatial, and then provide the associated semantics of the annotated constructs.

6.1 Entity Class

An application may require capturing the lifespan or transaction time of an entity, the shape and position of an entity in space, or a change in the shape and/or position of an entity over its lifespan. We describe the semantics of a temporal and spatial entity class with an example.

6.1.1 Temporal Entity Class

A *temporal entity class* refers to entities with associated existence time and/or transaction time. A temporal entity class implies that the membership of an entity in the entity set is time-varying. We assume that a temporal entity class (as contrasted with entities of that class) exists during the entire modeled time. Thus, the existence time represents the lifespan of an entity and defines the time when facts associated with an entity can be true in the miniworld. Similarly, we can capture the transaction time associated with an entity, which may be important for applications requiring traceability. In the real world, all objects are temporal. However, an entity class need not be modeled as temporal if the user is not interested in the lifespan of an entity or if the lifespan is not known.

A temporal entity class, e.g., PUMPLIFT, with existence time is associated with an existence predicate $\varphi_{\mathsf{PUMPLIFT},et}$ that defines the lifespan of a pumplift in terms of an existence time granularity $TG_{\mathsf{PUMPLIFT},et}$ (e.g., day).

$$\varphi_{\mathsf{PUMPLIFT},et}: S(\mathsf{PUMPLIFT}) \times \mathbf{Z} \rightarrow \mathbf{B}$$

This predicate takes a particular pumplift entity and a particular granule (denoted by an integer, here a specific day) of the granularity and evaluates to a Boolean that is true if that entity exists in the modeled reality at that granule (day). There are two constraints on the existence predicate $\varphi_{\mathsf{PUMPLIFT},et}$:

(i) $\forall\ e \in S(\mathsf{PUMPLIFT}),\ \varphi_{\mathsf{PUMPLIFT},et}(e, i) \Rightarrow (TG_{\mathsf{PUMPLIFT},et}(i) \subseteq Image(TG_{\mathsf{PUMPLIFT},et}))$

(ii) $\forall\ e \in S(\mathsf{PUMPLIFT}),\ \exists\ i \in \mathbf{Z},\ \varphi_{\mathsf{PUMPLIFT},et}\ (e, i)$

The first constraint states that a pumplift can exist only within the defined image of a granularity. Second, every entity exists at some granule (e.g., "2001-7-01") within the image of the granularity. Intuitively, if a pumplift with an associated lifespan does not exist during any granule within the image, it is meaningless to store it in the database. We can define the *existence temporal projection* operator (π^{et}) as a function that returns the temporal element of an entity. Formally, $\pi^{et}: S(\mathsf{PUMPLIFT}) \rightarrow \{\ i \mid e \in S(\mathsf{PUMPLIFT}) \wedge \varphi_{\mathsf{PUMPLIFT},et}(e, i)\}$. For example, the lifespan of a pumplift may be 1, 2,…, 10 corresponding to say "2001-7-02",…, "2001-7-11."

Similarly, a temporal entity class PUMPLIFT with transaction time is associated with a transaction time predicate $\varphi_{\mathsf{PUMPLIFT},\text{tt}}$ that defines the transaction time of a pumplift in terms of a transaction time granularity TG_{tt} (e.g., second). The transaction time granularity is only defined for all points less than *now*. If a transaction timestamp includes *Until Changed* (*UC*), a special transaction time marker, it denotes that the associated fact is current in the database. Unlike the existence time granularity, which can be specified by users, the transaction time granularity is system-defined. Intuitively, the transaction time predicate maintains transaction history in the information system.

$$\varphi_{\mathsf{PUMPLIFT},\text{tt}}: S(\mathsf{PUMPLIFT}) \times \{\mathbf{Z} \cup UC\} \rightarrow \mathbf{B}$$

The constraints on the transaction time predicate are similar to those on the existence time predicate. Similar to the existence temporal projection operator, we can define the *transaction temporal projection* operator (π^{tt}). A bitemporal entity class PUMPLIFT is associated with $\varphi_{\mathsf{PUMPLIFT},et}$ and $\varphi_{\mathsf{PUMPLIFT},\text{tt}}$ defined in terms of an existence time granularity $TG_{\mathsf{PUMPLIFT},et}$ and a transaction time granularity TG_{tt}.

Having defined temporal entity class abstractly, we next describe the semantics of a simple temporal entity class continuing with an example of PUMPLIFT. When an entity class is defined as temporal, it implies that the application would have queries like "What is the average monthly power consumption by all pumplifts over their installed existence?" (an example of a sequenced query) and "What are the pumplifts that were installed before 1995 and are operational now?" (an example of a current query).

Figure 3 illustrates the representation of existence time expressed as state (S) with day (Gregorian day) as the temporal granularity name. Based on the users' requirements, the data analyst simply annotates PUMPLIFT with "S(day)/-//" and does not need to contend with the complexity of the underlying semantics or the associated temporal constraints.

Figure 3 also shows the semantics of a temporal entity class in ST USM via a mapping using the concepts of a conventional conceptual model; we refer to it as a *translated USM schema*. This mapping from ST USM to (translated) USM are *snapshot equivalent* [33], i.e., the two schemas (ST USM and translated USM) represent the same information content over snapshots taken at all times. In order to express the semantics of a temporal entity class, we need to specify a TEMPORAL_GRANULARITY in which the evolution of a temporal object is embedded. The relationship PUMPLIFT_has_ET associates an entity with a corresponding TEMPORAL_GRANULARITY. Each TEMPORAL_GRANULARITY is uniquely identified by a granularity_name, shown by the underlined attribute. An extent is the smallest time interval that includes the image of a granularity and is expressed by two indexes, minimum and maximum. Each anchor_gran is a recursive relationship (i.e., a relationship where an entity from the same entity class can play different roles) such that each participating granularity optionally has an anchor (0:1) and each granularity is an anchor for 0 to many (0:M) other granularities.

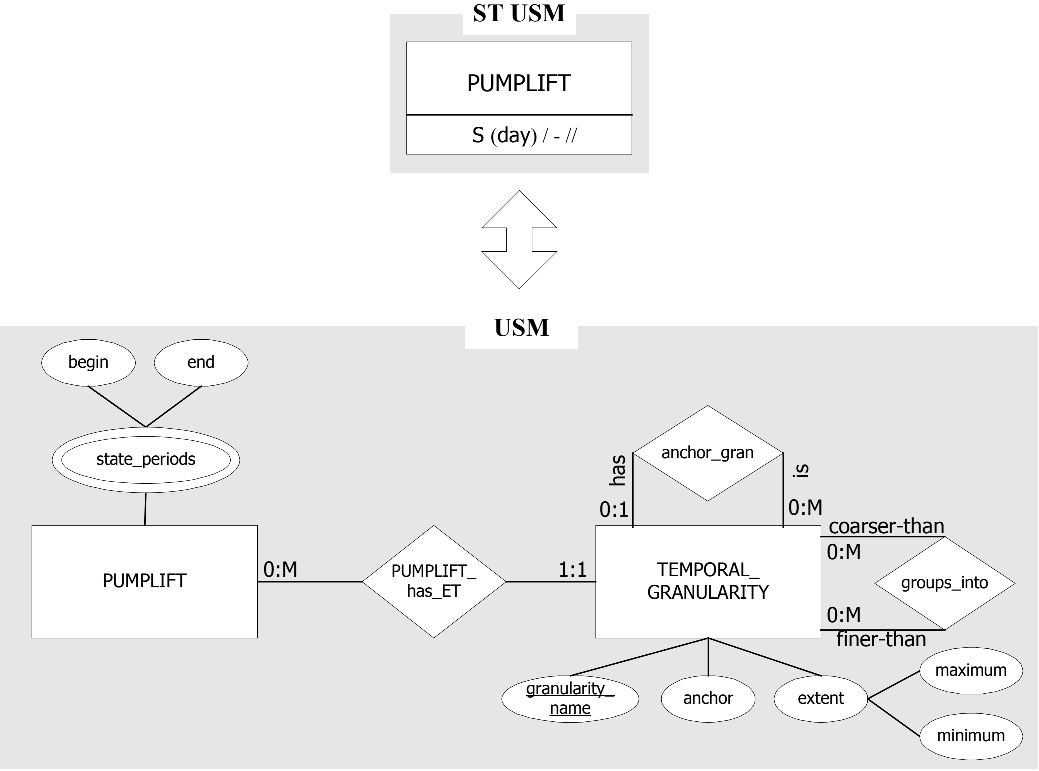


Figure 3: Temporal Entity Class in ST USM and its semantics in USM

The anchor of a granularity *TG* is the first index of a strictly finer granularity that corresponds to the origin of this granularity, i.e., *TG*(0). All granularities except the bottom granularity have an associated anchor. A finer-than and a coarser-than relationship between granularities are denoted by a recursive relationship groups_into, where one entity plays the role of finer-than and the other the role of coarser-than. The relationships anchor_gran together with groups_into helps create a granularity graph [14], which can help a user choose the level of detail associated with facts. Details related to granularities and indeterminacy is presented in [35].

A temporal entity with existence time may have a set of event_instants or state_periods associated with it depending on whether a temporal entity is represented as an event or a state. A time period of PUMPLIFT is represented with indexes begin and end of state_periods. A double-lined ellipse in USM denotes a multi-valued attribute. For example, state_periods is represented as multi-valued attribute and represents a set of state periods (i.e., a temporal element) associated with an entity. Figure 3 illustrates the complete semantics of a temporal entity class; in the subsequent sections, we will not show the complete semantics associated with temporal granularities. This is done primarily to help the reader focus on the additional semantics not explicated in previous sections.

We now describe the constraints on temporal entities of PUMPLIFT. These constraints are implicit in the ST USM schema but are explicit in the translated USM schema. Constraints 6.1.1 and 6.1.2 are based on our definition of a temporal entity; i.e., a temporal entity has an associated temporal granularity and has an associated temporal element. Constraints 6.1.3–6.1.5 are based the definition of temporal granularity. Constraints 6.1.6–6.1.8 are based on the definition of a temporal element and are examples of non-sequenced constraints. In these definitions, we assume a *closed-open representation* [75], i.e., the begin index is contained in the period while the index corresponding to the end is not. For example, an instant for a temporal element may be represented by [17, 18). In this example, begin index (i.e., 17) is inclusive in the instant while end index (i.e., 18) is not.

Constraint 6.1.1: The existence time for all the entities of PUMPLIFT have the same associated granularity; in this case, day.

$\forall\, e \in S(\text{PUMPLIFT}),\ e.\text{PUMPLIFT\_has\_ET.TEMPORAL\_GRANULARITY (granularity\_name)} = \text{day}$

Constraint 6.1.2: Every entity of PUMPLIFT has an associated temporal element with well-formed periods.

$\forall\, e \in S(\text{PUMPLIFT}),\ \exists p \in e.\text{state\_periods},\ p.\text{begin} < p.\text{end}$

Constraint 6.1.3: Each TEMPORAL_GRANULARITY has a lower and an upper bound referred to as minimum and maximum; these bounds are well-formed.

$\forall\, e \in S(\text{TEMPORAL\_GRANULARITY}),\ e(\text{extent.minimum}) < e(\text{extent.maximum})$

Constraint 6.1.4: All the granularities, except one, have an anchor. The bottom granularity is allowed not to have an anchor.

$\forall\, e \in S(\text{TEMPORAL\_GRANULARITY}),\ \neg\, \text{has}(e.\text{anchor\_gran}) \Rightarrow$
$\neg\, (\exists\, e_2 \in \text{TEMPORAL\_GRANULARITY} \wedge e \neq e_2 \wedge \neg\, \text{has}(e_2.\text{anchor\_gran}))$

Constraint 6.1.5: For a temporal granularity, if an anchor does not exist then that is the bottom granularity that does not have any granularity finer than it; in other words, it cannot take the role of coarser-than in the relationship groups-into.

$\forall\, e \in S(\text{TEMPORAL\_GRANULARITY}),\ \neg\, \text{has}(e.\text{anchor\_gran}) \Rightarrow \neg\, \text{coarser-than}(e.\text{groups\_into})$

Constraint 6.1.6: State periods of an entity of PUMPLIFT are well-formed.

$\forall\, e \in S(\text{PUMPLIFT}),\ \forall p \in e.\text{state\_periods},\ p.\text{begin} < p.\text{end}$

Constraint 6.1.7: Temporal elements are well-formed. A temporal element is defined as a union of non-overlapping time intervals.

$\forall\, e \in S(\text{PUMPLIFT}),\ \forall\, p_1, p_2 \in e.\text{state\_periods},\ p_1.\text{begin} < p_2.\text{begin} \Rightarrow p_1.\text{end} \leq p_2.\text{begin}$

Constraint 6.1.8: The extent of a temporal granularity defines the upper and lower bounds for any temporal element. In other words, a temporal element cannot include an index that is larger than the corresponding extent.maximum or smaller than the corresponding extent.minimum.

$\forall\, e \in S(\text{PUMPLIFT}),\ \forall\, p \in e.\text{state\_periods},\ e.\text{PUMPLIFT\_has\_ET.TEMPORAL\_GRANULARITY (extent.minimum)} \leq p.\text{begin} <$
$p.\text{end} \leq e.\text{PUMPLIFT\_has\_ET.TEMPORAL\_GRANULARITY (extent.maximum)}$

We next describe the semantics associated with a bitemporal entity class using an example of PUMPLIFT. Specifying PUMPLIFT as bitemporal allows an application to include queries like "What is the average monthly history of power consumption by PUMPLIFT during their installed existence as best known on January 27, 2001?" As shown in Figure 4, a bitemporal entity class PUMPLIFT in ST USM includes both existence time and transaction time annotation. In Figure 4, there are two relationships between PUMPLIFT and TEMPORAL_GRANULARITY: PUMPLIFT_has_ET and PUMPLIFT_has_TT. While the former defines an association between an entity *e* and its existence granularity (i.e., granularity_name = day), the latter defines the association between the entity *e* and its (system-defined) transaction time granularity.

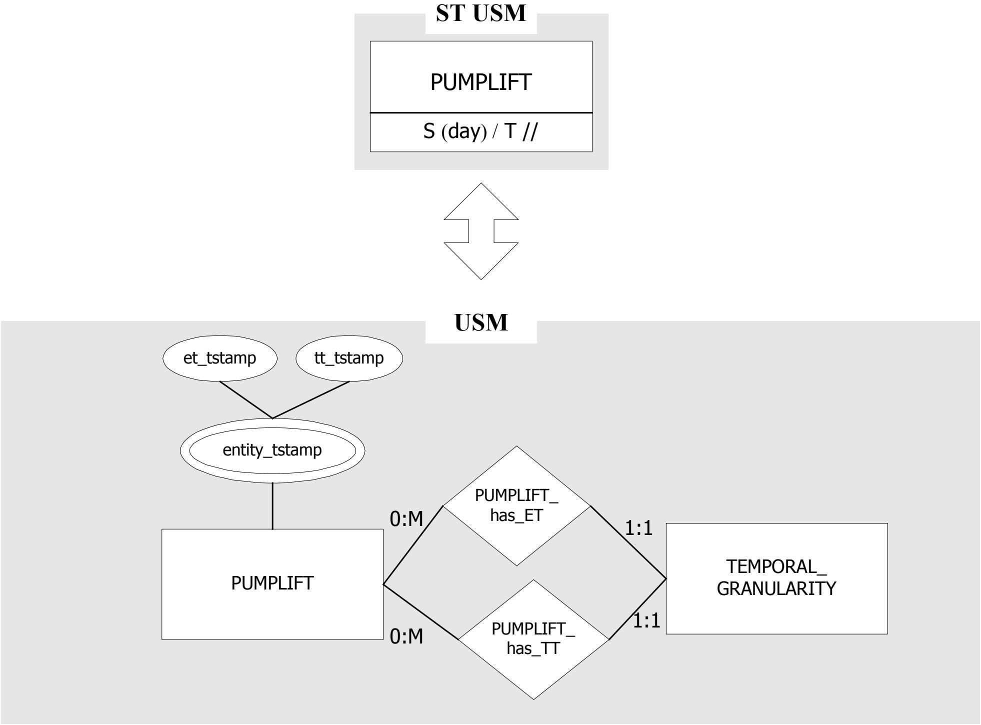


Figure 4: Bitemporal entity class in ST USM and its semantics in USM

Besides the definition-based constraints 6.1.1, 6.1.3, 6.1.4 and 6.1.5, there are additional (transaction time related) constraints on entities of a bitemporal entity class.

Constraint 6.1.9: Every entity has an associated bitemporal granule within specified extent.

$\forall e \in S$(PUMPLIFT), $\exists p \in e$.entity_tstamp,
(*e*.PUMPLIFT_has_ET.TEMPORAL_GRANULARITY (extent.minimum) $\leq$ *p*.et_tstamp $\leq$
(*e*.PUMPLIFT_has_ET.TEMPORAL.GRANULARITY (extent.maximum)) $\wedge$
(*e*.PUMPLIFT_has_TT.TEMPORAL_GRANULARITY (extent.minimum)) $\leq$ *p*.tt_tstamp $\leq$
e.PUMPLIFT_has_TT.TEMPORAL_GRANULARITY (extent.maximum))

Constraint 6.1.10: For transaction time, we do not need to specify the granularity, since it is system-defined. For the entire schema, there is single transaction time granularity; here, we use sec.

$\forall e \in S$(PUMPLIFT), *e*.PUMPLIFT_has_TT.TEMPORAL_GRANULARITY (granularity_name) = sec

Constraint 6.1.11: The bitemporal timestamp is such that the existence and transaction timestamps are within the extent of their respective granularities.

$\forall e \in S(\text{PUMPLIFT}), \forall p \in e.\text{entity\_tstamp},$
$(e.\text{PUMPLIFT\_has\_ET.TEMPORAL\_GRANULARITY (extent.minimum)} \leq p.\text{et\_tstamp} \leq e.\text{PUMPLIFT\_has\_ET.TEMPORAL.GRANULARITY (extent.maximum)}) \wedge$
$(e.\text{PUMPLIFT\_has\_TT.TEMPORAL\_GRANULARITY (extent.minimum)} \leq p.\text{tt\_tstamp} \leq e.\text{PUMPLIFT\_has\_TT.TEMPORAL\_GRANULARITY (extent.maximum)})$

6.1.2 Spatial Entity Class

A *spatial entity class* refers to geo-referenced entities with an associated shape and position, which is used to locate them in a two- or three-dimensional space. In this subsection, we first define a spatial entity in terms of a spatial granularity and then describe the associated semantics of a spatial entity class in ST USM using examples of SPRING_SITE and BORE_HOLE_SITE.

A spatial entity in a horizontal space domain, e.g., SPRING_SITE, is associated with a *horizontal geometry predicate* $\psi_{\text{SPRING\_SITE},xy}$ that defines the location of a spring site in terms of horizontal spatial granularity.

$$\psi_{\text{SPRING\_SITE},xy}: S(\text{SPRING\_SITE}) \times \mathbf{Z} \rightarrow \mathbf{B}$$

Spatial partitions may be formed in 2-dimensional or 3-dimensional space; however, these partitions are represented by integers.

There are two constraints on the horizontal geometry predicate.

(i) $\forall e \in S(\text{SPRING\_SITE}), \psi_{\text{SPRING\_SITE},xy}(e, i) \Rightarrow (SG_{\text{SPRING\_SITE},xy}(i) \subseteq Image(SG_{\text{SPRING\_SITE},xy}))$

(ii) $\forall e \in S(\text{SPRING\_SITE}), \exists i \in \mathbf{Z}, \psi_{\text{SPRING\_SITE},xy}(e, i)$

The first constraint implies that any partition of the horizontal space domain denoted by an index *i* lies within the image of the granularity. The second constraint states that a spatial entity must exist somewhere within the defined image; i.e., each spatial entity has an associated geometry. We can define a *horizontal spatial projection* operator (π^{xy}) that takes a spatial entity and returns its geometry (e.g., point, line or region). Formally,

$$\pi^{xy}: S(\text{SPRING\_SITE}) \rightarrow \{ i \mid e \in S(\text{SPRING\_SITE}) \wedge \psi_{\text{SPRING\_SITE},xy}(e, i)\}$$

In this example above, π^{xy} is constrained to be a point on the horizontal surface.

A spatial entity in 3-dimensional space, e.g., BORE_HOLE_SITE, is associated with $\psi_{\text{BORE\_HOLE\_SITE},xy}$ and $\psi_{\text{BORE\_HOLE\_SITE},z}$ that define the location of an entity in terms of horizontal and vertical spatial granularities, i.e., $SG_{\text{BORE\_HOLE\_SITE},xy}$ and $SG_{\text{BORE\_HOLE\_SITE},z}$. For an entity in 3-dimensional space, we define a vertical projection operator, i.e., π^{z}. Formally,

$$\pi^{z}: S(\text{BORE\_HOLE\_SITE}) \rightarrow \{ j \mid (e \in S(\text{BORE\_HOLE\_SITE}) \wedge \psi_{\text{BORE\_HOLE\_SITE},z}(e, j)) \}$$

In the example above π^{xy} is constrained to be a point and π^{z} is constrained to be a line. Additionally, there exists a relationship between horizontal and vertical geometry predicate: $\forall i, \psi_{\text{BORE\_HOLE\_SITE}, xy}(e, i) \Rightarrow \exists j, \psi_{\text{BORE\_HOLE\_SITE}, z}(e, j)$.

Let us now describe the related spatial annotation. The spatial annotation follows a double forward slash (//). The geometry associated with a spatial entity can be a point (P), a line (L), or a region (R) in 2- or 3-dimensional space. The annotation for *x*-dimension is followed by the *y*-dimension, which is followed by the *z*-dimension; each dimension is separated by a forward slash (/). Additionally, the user can specify the associated horizontal and vertical granularities.

Example 1: "// P(deg) / P(deg) / -" describes a spatial entity with a geometry of points in the *x*-*y* plane. The associated horizontal spatial granularity is degree.

Example 2: "// P(deg) / P(deg) / L(ft)" defines a geometry that is a point in the *x*-*y* plane with a horizontal spatial granularity of degree and a line in the *z*-dimension with a vertical spatial granularity of foot.

We now describe the semantics of a spatial entity class in ST USM using USM and the constraints. Figure 5 shows a spatial entity class SPRING_SITE, which includes geometry of points with a horizontal spatial granularity as degree.

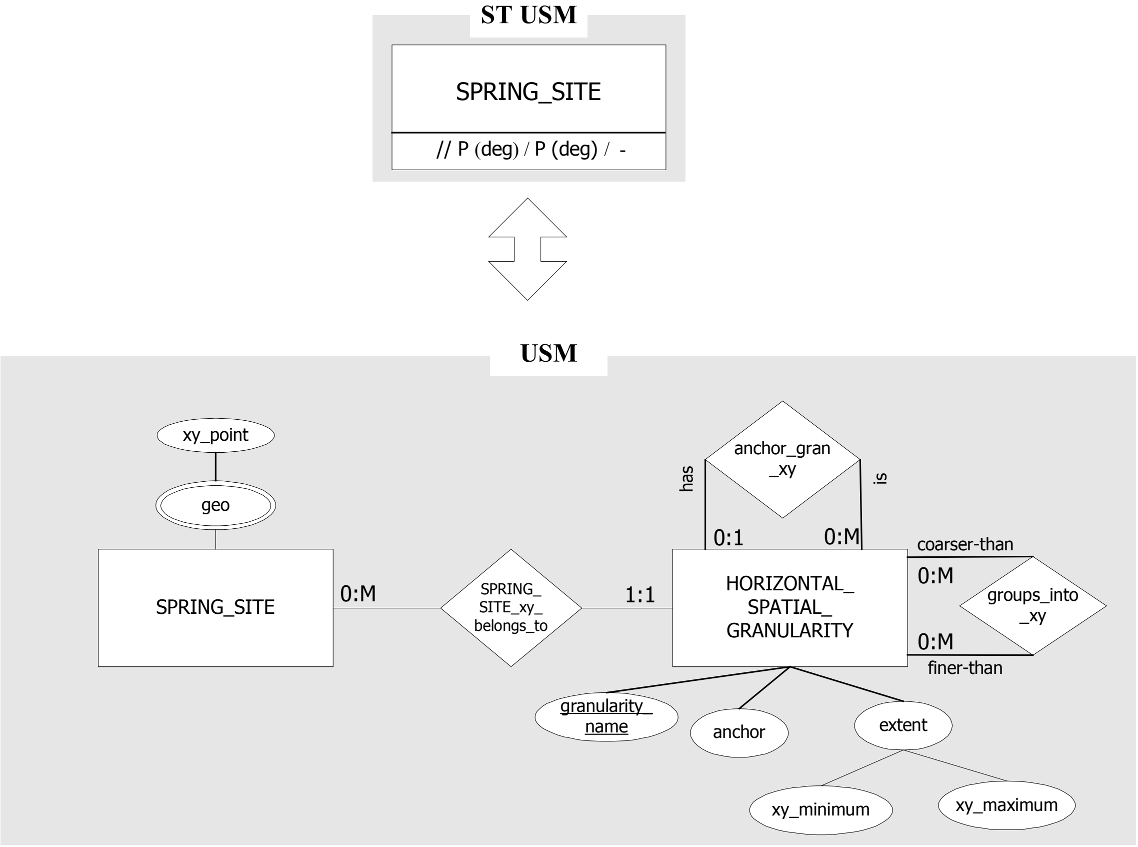


Figure 5: A spatial entity class in ST USM in horizontal space and its semantics in USM

In order to specify the semantics of a spatial entity class, we need to define the HORIZONTAL_SPATIAL_GRANULARITY in which the geometry of a spatial entity is embedded. A HORIZONTAL_SPATIAL_GRANULARITY is uniquely specified by granularity_name. The extent is the minimum-bounding rectangle that includes the image of the granularity. The recursive relationships groups_into_xy and anchor_gran_xy are similar to those in the temporal entity class. The relationship SPRING_SITE_xy_belongs_to relates a spatial entity with a corresponding horizontal spatial granularity. We next describe the associated constraints. Constraints 6.1.12 and 6.1.13 are based on our definition of a spatial entity, constraints 6.1.14–6.1.17 are based on the definition of spatial granularity, and constraints 6.1.18–6.1.20 are based on geometry of spatial entities.

Constraint 6.1.12: All entities in SPRING_SITE must have the same horizontal spatial granularity; in this case degree.

$\forall e \in S(\text{SPRING\_SITE}),\ e.\text{SPRING\_SITE\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (granularity\_name)} = \text{deg}$

Constraint 6.1.13: Every entity of SPRING_SITE has an associated geometry (e.g., a point in horizontal space for SPRING_SITE) within the specified extent.

$\forall e \in S(\text{SPRING\_SITE}), \exists g \in e.\text{geo},$
$e.\text{SPRING\_SITE\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (extent.xy\_minimum)} \leq g.\text{xy\_point} \leq$
$e.\text{SPRING\_SITE\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (extent.xy\_maximum)}$

Constraint 6.1.14: The indexes corresponding to the geometry of a spatial object lie within xy_minimum and xy_maximum.

$\forall e \in S(\text{SPRING\_SITE}), \forall g \in e.\text{geo},$
$e.\text{SPRING\_SITE\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (extent.xy\_minimum)} \leq g.\text{xy\_point} \leq$
$e.\text{SPRING\_SITE\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (extent.xy\_maximum)}$

Constraint 6.1.15: Extent is well-formed.

$\forall e \in S(\text{HORIZONTAL\_SPATIAL\_GRANULARITY}),\ e(\text{extent.xy\_minimum}) < e(\text{extent.xy\_maximum})$

Constraint 6.1.16: All granularities except one (i.e., the bottom granularity) have an anchor.

$\forall e \in S(\text{SPATIAL\_GRANULARITY}), \neg \text{has}(e.\text{anchor\_gran\_xy}) \Rightarrow \neg (\exists e_2 \in \text{SPATIAL\_GRANULARITY} \wedge e_2 \neq e_2 \wedge$
$\neg\text{has}(e_2.\text{anchor\_gran\_xy})$

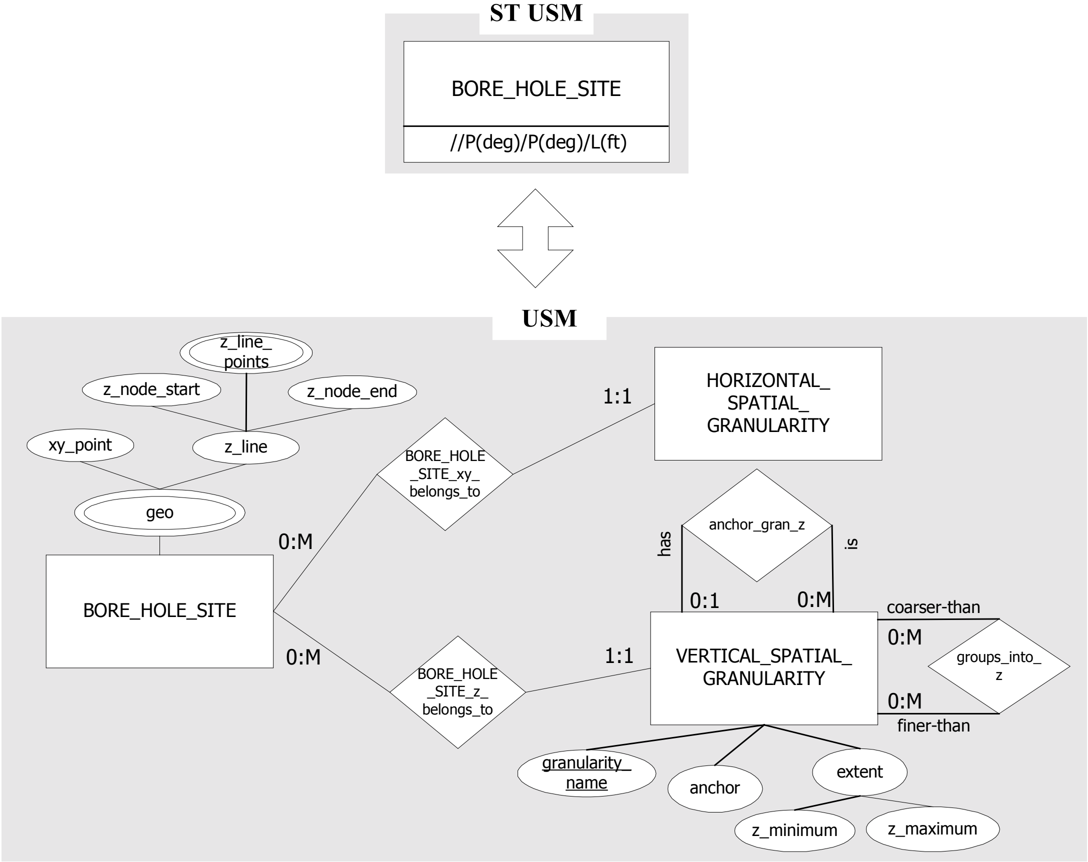


Figure 6: A spatial entity in three-dimensional space and its semantics in USM

Constraint 6.1.17: The bottom granularity does not have any granularity that is finer than it; in other words it cannot take the role of coarser-than in the relationship groups-into.

$\forall e \in S(\text{SPATIAL\_GRANULARITY}), \neg \text{has}(e.\text{anchor\_gran\_xy}) \Rightarrow \neg \text{coarser-than}(e.\text{groups\_into\_xy})$

Constraint 6.1.18: The geometry of spring-site is constrained to point.
$\forall\ e \in S$(SPRING_SITE), $\forall\ g \in e$.geo, g.xy_point $\in$ *point*

Similarly, a spatial entity embedded in a three-dimensional space is associated with HORIZONTAL_SPATIAL_GRANULARITY and VERTICAL_SPATIAL_GRANULARITY (Figure 6). A spatial object in a three-dimensional space has two associated relationships, BORE_HOLE_SITE_xy_belongs_to and BORE_HOLE_SITE_z_belongs_to corresponding to its horizontal and vertical spatial granularities. We have not shown the details associated with HORIZONTAL_SPATIAL_GRANULARITY as they have already been described in Figure 5. In Figure 6, a line starts with a node (i.e., z_node_start) and ends with a node (i.e., z_node_end) and has multiple points (i.e., z_line_points) between the start and end nodes [62]. The associated constraints related to horizontal granularity are identical to the ones already described. The constraints related to vertical spatial granularity are similarly defined as in constraints 6.1.12–6.1.17.

Constraint 6.1.19: The horizontal and vertical geometry of borehole site are represented as a point and a line, respectively.
$\forall\ e \in S$(BORE_HOLE_SITE), $\forall\ g \in e$.geo, g.xy_point $\in$ *point* $\wedge$ g.z_line $\in$ *line*

Constraint 6.1.20: The vertical geometry is associated with a horizontal geometry.
$\forall\ e \in S$(BORE_HOLE_SITE), $\forall g_1 \in e$.geo, $\exists g_2 \in e$.geo,
g_1.z_line.z_node_start = g_2.xy_point $\wedge$ g_1.z_line.z_node_end = g_2.xy_point

6.1.3 Time-Varying Spatial Entity Class

A time-varying spatial entity includes modeling two types of changes: (i) *non-spatial change*, an entity with a fixed associated position/geometry and a lifespan; and (ii) *spatial change*, an entity whose geometry varies over its lifespan. This corresponds to Tryfona and Jensen's [87] differentiation between "a spatial entity set in time" and "the position of a spatial entity set in time." The former models a spatial entity with an associated existence time (and/or transaction time). The latter captures change in shape and/or position over time (existence time and/or transaction time). For the second case, only the position of an entity may change continuously while the shape does not (e.g., a transportation application) or the shape may change discretely (e.g., a cadastral application) or both position and shape may change continuously (e.g., modeling a hurricane).

In case (i) above, a time-varying spatial entity class has an associated existence predicate (i.e., $\varphi_{E,\mathrm{et}}$), a geometry predicate (i.e., $\psi_{E,xy}$), and the spatial projection is not time-varying. Intuitively, this implies that the application needs to capture when and where entities exist. However, queries that involve evolving geometries over time like "During the year 2001, did the surface area of Lake Mesquite decrease by more than 10% of its area measured on 2000-08-25" are not required of the application. The annotation for this case is simply a combination of the spatial and temporal annotation already described in the previous two subsections; e.g., "S($\langle g_{et}\rangle$)/-//P($\langle g_{xy}\rangle$)/P($\langle g_{xy}\rangle$)/-", implying that the geometry of entities is a (non-time-varying) point (P) on the *x*-*y* plane with a spatial granularity of $\langle g_{xy}\rangle$. Additionally, the lifespan of entities is modeled as a state (S) with a temporal granularity of $\langle g_{et}\rangle$. The associated semantics are a combination of the semantics shown in Figure 3 and Figure 5.

In case (ii) above, the application needs to capture various shapes and/or positions of the entities over time. That is, a user may want to capture valid time or transaction time associated with the geometry. A 2-dimensional time-varying spatial entity class with a time-varying geometry has an associated time-varying geometry predicate that defines the position of entities (in terms of the horizontal spatial granularity $SG_{E,xy}$) at various time granularity indexes (as specified by the temporal granularity $TG_{E,et}$); $\forall j,\ \varphi_{E,et}(e, j) \Rightarrow \exists i,\ \psi_{E,xy}\ (e, i)$.

Our annotation syntax includes a formalism to specify whether or not position and/or shape is time-varying: Pos (or, Position) and Sh (or, Shape), respectively. Additionally, the user can also specify the dimension (e.g., xy and xyz) over which the position and/or shape changes over time. However, to include a time-varying spatial annotation, the temporal and spatial annotation should already have been specified. The time-varying spatial annotation is specified after the second double forward slash (//). To recap, the complete spatio-temporal annotation is specified as:

⟨temporal annotation⟩ // ⟨spatial annotation⟩ // ⟨time-varying spatial annotation⟩

Example 3: "E(sec)/-// P(deg)/ P(deg)/-//Pos@xy" denotes that entities in the entity class have only time-varying position while the shape is time-invariant. The associated shape of the entities is a point (P) in the *x-y* plane with a horizontal spatial granularity of degree. The position of entities changes in the *x-y* plane (Pos@xy) over time and the associated shapes are valid for time granules (E) measured in second.

Example 4: "S(min)/-// R(dms-sec)/ R(dms-sec)/-//Sh@xy" implies that the shape of entities is a region (R) in an *x-y* plane with a horizontal spatial granularity of dms-second. The shape of the entity changes over time in the *x-y* plane (Sh@xy); each shape is valid for a set of time granules (S = state) measured in minute. If the geometry changes from region to point (or even line), the geometry is still generically represented as region (R) only as a point (line) is a degenerate region.

Figure 7 illustrates a time-varying spatial entity class with a changing position in ST USM and its corresponding semantics in USM. A time-varying spatial entity class with time-varying shape/position implies that each geometry (i.e., multi-valued attribute geo) is associated with state_periods. Each entity from ⟨ENTITY_CLASS⟩ is related with entities from ⟨ENTITY_CLASS⟩_GEOMETRY_T, a weak entity class. A weak entity class implies that an entity in ⟨ENTITY_CLASS⟩_GEOMETRY_T cannot exist without a corresponding entity in ⟨ENTITY_CLASS⟩; ⟨ENTITY_CLASS⟩_GEOMETRY_T_ID is its partial key. Constraints 6.1.1, 6.1.3–6.1.5 related to temporal granularity and constraints 6.1.12, 6.1.14–6.1.17 related to spatial granularity are applicable here.

Constraint 6.1.21: cf. constraint 6.1.6

$\forall\ e \in S(\langle\text{ENTITY\_CLASS}\rangle\text{\_GEOMETRY\_T}),\ \forall p \in e.\text{state\_periods},\ p.\text{begin} < p.\text{end}$

Constraint 6.1.22: cf. 6.1.7

$\forall\ e \in S(\langle\text{ENTITY\_CLASS}\rangle),\ \forall p_1, p_2 \in e.\langle\text{ENTITY\_CLASS}\rangle\text{\_has\_G\_T}.\langle\text{ENTITY\_CLASS}\rangle\text{\_GEOMETRY\_T.state\_periods},$
$p_1.\text{begin} < p_2.\text{begin} \Rightarrow p_1.\text{end} < p_2.\text{begin}$

Constraint 6.1.22: Each entity has an associated geometry at all points in time.

$\forall\ e \in S(\langle\text{ENTITY\_CLASS}\rangle),\ \forall\ p \in e.\langle\text{ENTITY\_CLASS}\rangle\text{\_has\_G\_T}.\langle\text{ENTITY\_CLASS}\rangle\text{\_GEOMETRY\_T.state\_periods},$
$\forall k \in [p.\text{begin}, p.\text{end}],\ \exists g \in e.\langle\text{ENTITY\_CLASS}\rangle\text{\_has\_G\_T}.\langle\text{ENTITY\_CLASS}\rangle\text{\_GEOMETRY\_T.geo},$
$g.\text{xy\_point} \in \text{point}$

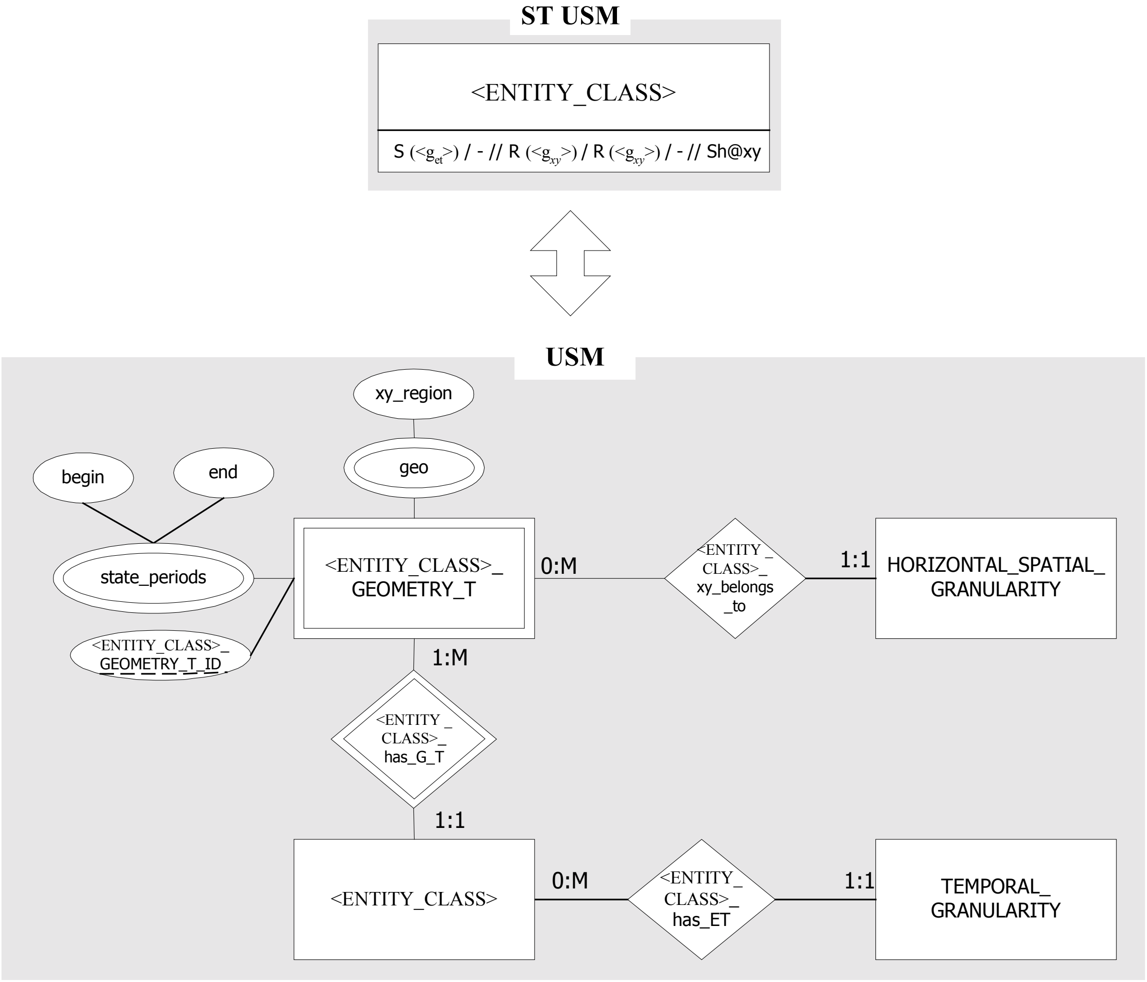

ST USM
<ENTITY_CLASS>
S (<g_{et}>) / - // R (<g_{xy}>) / R (<g_{xy}>) / - // Sh@xy
USM
xy_region
geo
begin
end
state_periods
<ENTITY_CLASS>_ GEOMETRY_T
<ENTITY_CLASS>_ GEOMETRY_T_ID
0:M
<ENTITY_CLASS>_ xy_belongs _to
1:1
HORIZONTAL_SPATIAL_ GRANULARITY
1:M
<ENTITY_CLASS>_ has_G_T
1:1
<ENTITY_CLASS>
0:M
<ENTITY_CLASS>_ has_ET
1:1
TEMPORAL_ GRANULARITY

Figure 7: Time-varying spatial entity class with changing shape in ST USM and its semantics in USM

6.2 Attribute

Entities have properties referred to as attributes that represent facts that need to be captured for a database application. An attribute of an entity class is associated with a value set or a domain [21]. An entity class can have two kinds of attributes, *descriptive attributes* and *spatial attributes*. If the user wants to capture the evolution of facts over time, the attribute is referred to as a *temporal attribute*; when the user wants to capture the evolution of the spatial facts over time, the attribute is referred to as a *time-varying spatial attribute*.

6.2.1 Temporal Attribute

Temporal attributes represent properties of an entity and can have both valid time and transaction time associated with them. The temporal annotation for an attribute is the same as that for a temporal entity class described in the previous section. Annotating an attribute renders it sequenced implying that the property is true for each point in time within the associated temporal element. In this subsection, we describe the associated semantics of a temporal attribute.

As shown in Table 2, a non-temporal or temporal attribute can be associated with a temporal entity or non-temporal entity class.

	Non-temporal Attribute	**Temporal Attribute**
Non-temporal Entity Class	Only the current properties of the currently relevant entities (*1*)	Maintain attribute value histories of the currently relevant entities (*2*)
Temporal Entity Class	Only the current properties over the lifespan of entities (*3*)	Maintain attribute value histories over the lifespan of entities (*4*)

Table 2: The semantics of temporal/non-temporal attribute/entity combinations

A temporal entity class implies that objects are important for a database application even when they are not current in the modeled reality. A non-temporal entity class implies that only the currently legitimate objects are important for the database application. Thus, if a non-temporal entity is no longer currently legitimate, one does not need to store the evolution of facts associated with this entity (cell *2* in Table 2). For example, a non-temporal entity class SOURCE_AGENCY with a temporal attribute tech_name would imply that histories of only currently relevant source agencies are pertinent for the application. A non-temporal attribute of a temporal entity (cell *3* in Table 2) indicates that: (i) the property does not vary with time; (ii) the user is only interested in the last recorded value of the attribute and not in its history; or (iii) the time associated with the attribute is unknown. This implies that the application will not include queries like "Give the history of tests that we have ever been conducted at a specified borehole site since 1998." A temporal attribute for a temporal entity (cell *4* in Table 2) necessitates that the valid time of the attribute is equal to the lifespan of the entity; this is a direct implication from the semantics of a conventional conceptual model where attributes are specified for an existing entity. In specifying the semantics of a temporal attribute, we generalize the conventional semantics of an attribute to define a temporal attribute, which is true for every point in time (represented by granularity indexes). The cell *1* in Table 2 represents a non-temporal entity class with non-temporal attribute in a conventional conceptual model.

In the following subsections, we describe time-varying single-valued, key, composite and multi-valued attributes.

6.2.1.1 Single-Valued/Multi-Valued Temporal Attribute

A single-valued temporal attribute is one where each entity has a maximum of one value for any time granularity index; however, it can have multiple values over the lifetime of the entity. A temporal attribute A of an entity class E with valid time is associated with an *attribute valid time* function $\varphi_{E,A,vt}$ that defines the attribute values, *dom*(A) at different time granularity indexes.

$\varphi_{E,A,vt}$: $S(E) \times \mathbf{Z} \rightarrow 2^{dom(A)}$

The constraints on the attribute valid time function are described below:

(i) $\forall\ e \in S(E),\ \varphi_{E,A,vt}(e, i) \in dom(A) \Rightarrow (TG_{E,A,vt}(i) \subseteq Image(TG_{E,A,vt}))$

(ii) $\forall\ e \in S(E),\ \forall\ i,\ \varphi_{E,et}(e, i) \Rightarrow \varphi_{E,A,vt}(e, i) \in dom(A)$

The first constraint states that the history of an attribute can be defined within the image of the granularity of the attribute. Constraint (ii) applies when the associated entity of a temporal entity is also temporal (cell *4* of Table 2). The second constraint states that the history of an attribute is defined within the lifespan of the associated entity. If no value is defined for a granularity index where an entity exists then the corresponding value of the temporal (optional) attribute for that index is assumed to be unknown; this is a direct implication from the semantics of a conventional conceptual model where unknown values of *optional attributes* [60] are represented by NULL. *Mandatory attributes* are required to have a non-null value; *temporal mandatory attributes* are required to have non-null values at each point in time, a sequenced constraint; this is natural generalization of the definition of a mandatory attribute. For a single-valued temporal attribute, $\varphi_{E,A,vt}(e,i)$ is constrained to be a *singleton* (i.e., a set with only one element); there is no such restriction for a multi-valued temporal attribute.

Similarly, a temporal attribute with transaction time is associated with an *attribute transaction function* $\varphi_{E,A,\mathrm{tt}}$ and a bitemporal attribute is associated with $\varphi_{E,A,\mathrm{tt}}$ and $\varphi_{E,A,vt}$.

Based on the users' requirements, a data analyst simply annotates an attribute as temporal. For example, in Figure 8 the user wants to capture the evolution of an attribute ⟨t-attrib⟩ as a set of state periods (S) with a temporal granularity, $\langle \mathrm{g}_{vt} \rangle$. The data analyst simply annotates ⟨t-attrib⟩ as "S($\langle \mathrm{g}_{vt} \rangle$)/-//." Notice that the annotation syntax is independent of the type of conceptual modeling abstraction; that is why the annotation syntax for an attribute is identical to that for an entity class in the previous section.

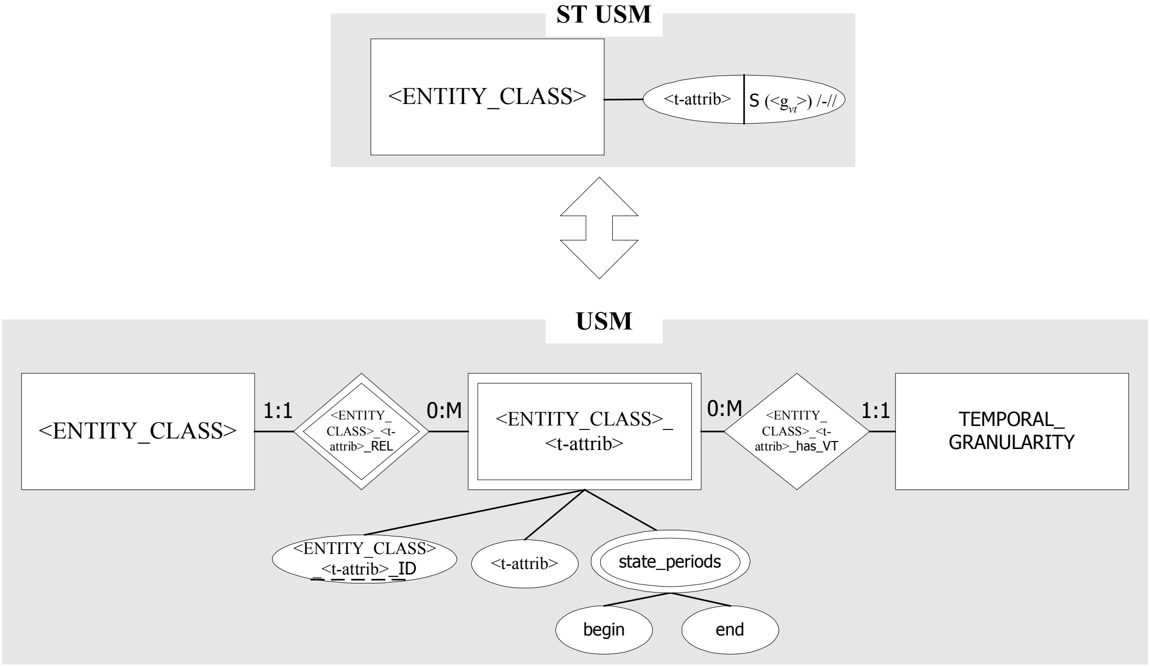


Figure 8: Temporal attribute in ST USM and its semantics in USM

The bottom part of Figure 8 shows the underlying semantics of a temporal attribute in ST USM via a mapping using the concepts of a conventional USM model. Each entity e of ⟨ENTITY_CLASS⟩ optionally has a corresponding ⟨ENTITY_CLASS⟩_⟨t-attrib⟩, a weak entity class corresponding to a temporal attribute of the ⟨ENTITY_CLASS⟩. ⟨ENTITY_CLASS⟩_⟨t-attrib⟩_ID is a partial key for ⟨ENTITY_CLASS⟩_⟨t-attrib⟩; it is represented by a dashed underline. Similar to a temporal entity class described in the previous section, the evolution of a temporal attribute needs to be embedded in a TEMPORAL_GRANULARITY. The relationship ⟨ENTITY_CLASS⟩_⟨t-attrib⟩_has_VT associates a temporal attribute with a corresponding TEMPORAL_GRANULARITY. A temporal attribute with event-instants has the semantics as shown in Figure 8 with state_periods replaced by event_instants.

We now describe the constraints on a temporal attribute. The constraints 6.2.1–6.2.4 (related to a temporal element) are similar to 6.1.1, 6.1.6–6.1.8, respectively. The granularity-based constraints 6.1.3–6.1.5 hold for a temporal attribute also. Constraint 6.2.5 is a sequenced consequent of non-temporal USM, where an attribute (e.g., *attr*) draws values from a domain (e.g., *dom*(*attr*) ∪ NULL); a temporal attribute draws values from the domain at each point in time during the lifespan of the entity.

Constraint 6.2.1: The valid time associated with an attribute for an entity class has the same associated granularity. The granularity associated with the valid time of an attribute is specified in ST USM as ⟨g_{vt}⟩ and corresponds to the granularity_name of a temporal granularity.

$\forall a \in S(\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle),\ a.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle\text{\_has\_VT.TEMPORAL\_GRANULARITY(granularity\_name)} = \langle g_{vt}\rangle$

Constraint 6.2.2: The state periods of a temporal attribute are well-formed.

$\forall a \in S(\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle),\ \forall p \in a.\text{state\_periods},\ p.\text{begin} < p.\text{end}$

Constraint 6.2.3: A temporal element is defined as a union of non-overlapping time intervals. The temporal element of a (temporal) attribute associated with every entity is well-formed. Assuming closed-open representation, this constraint is formally stated below.

$\forall e \in S(\langle\text{ENTITY\_CLASS}\rangle),\ \forall a \in e.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle\text{\_REL}.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle,\ \forall p_1, p_2 \in a.\text{state\_periods},$
$p_1.\text{begin} < p_2.\text{begin} \Rightarrow p_1.\text{end} \le p_2.\text{begin}$

Constraint 6.2.4: The extent of a temporal granularity defines the upper and the lower bounds for any temporal element. In other words, a temporal element cannot have a granule that is larger than extent.maximum or smaller than extent.minimum.

$\forall a \in S(\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle),\ \forall p \in a.\text{state\_periods}$
$a.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle\text{\_has\_VT.TEMPORAL\_GRANULARITY (extent.minimum)} \le p.\text{begin} < p.\text{end} \le$
$a.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle\text{\_has\_VT.TEMPORAL\_GRANULARITY (extent.maximum)}$

Constraint 6.2.5: If both the entity class and its attribute are temporal, the union of the temporal elements of an (temporal) attribute (*ap*) must be equal to the lifespan of the associated entity (*ep*).

$\forall e \in S(\langle\text{ENTITY\_CLASS}\rangle),\ \forall ep \in e.\text{state\_periods},\ \forall k \in [ep.\text{begin}, ep.\text{end}],$
$\exists a \in e.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle\text{\_REL}.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle,\ \exists ap \in a.\text{state\_periods},$
$ap.\text{begin} \le k < ap.\text{end}$

If the granularities of the entity ($\text{temporal\_granularity}_e$) and the attribute ($\text{temporal\_granularity}_a$) are not the same, then the constraint 6.2.5 needs to be modified so that the granularity of the attribute is *cast* [14] to the granularity of the entity class.

$\forall e \in S(\langle\text{ENTITY\_CLASS}\rangle),\ \forall ep \in e.\text{state\_periods},\ \forall k \in [ep.\text{begin}, ep.\text{end}],$
$\exists a \in e.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle\text{\_REL}.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle,\ \exists ap \in a.\text{state\_periods},$
$cast(\text{temporal\_granularity}_a(ap.\text{begin}), \text{temporal\_granularity}_e)) \le k < cast(\text{temporal\_granularity}_a(ap.\text{end}), \text{temporal\_granularity}_e))$

6.2.1.2 Temporal Key Attribute

Identifiers (also called a *keys*) are one or more attributes used to identify members of an entity set. Annotating a key attribute renders it sequenced. In other words, a *temporal key* [75]—a sequenced constraint—is a uniqueness constraint *at any point in time*.

> **Constraint 6.2.6**: A temporal key attribute is defined as an attribute that is unique at any point in time. For example, if ⟨t-attrib⟩ is a key attribute, there will be an additional uniqueness constraint on this attribute. At each point of time within the temporal element, the number of entities with the same value of the key attribute is 1.
>
> $$\forall e_1, e_2 \in S(\langle\text{ENTITY\_CLASS}\rangle), \exists a_1 \in e_1.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle\_\text{REL}.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle,$$
> $$\exists a_2 \in e_2.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle\_\text{REL}.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{t-attrib}\rangle, \exists p_1 \in a_1.\text{state\_periods}, \exists p_2 \in a_2.\text{state\_periods},$$
> $$a_1.\langle\text{t-attrib}\rangle = a_2.\langle\text{t-attrib}\rangle \wedge \neg(\neg(p_1.\text{begin} < p_2.\text{begin}) \vee (p_1.\text{end} < p_2.\text{begin})) \Rightarrow e_1 = e_2$$

On the other hand, a *lifetime uniqueness constraint*—a type of non-sequenced constraint—implies that entities cannot have more one value during the entire lifespan of the entity. This constraint can be extended to specify a non-sequenced uniqueness constraint over a specified period of the lifespan of the entity. For example, "There will be no more than 350 leakance tests conducted during the first two years of the existence of a borehole site" is an example of non-sequenced uniqueness constraint over *a part of* the lifespan of the borehole site; such constraints may be specified in the data dictionary.

6.2.1.3 Temporal Composite Attribute

Composite attributes model facts in which users sometimes refer to the facts as a unit (or *composite*) and at other times refer specifically to the *components*. The composite and the component attributes can be temporal or non-temporal. For a composite attribute, where the composite is temporal, the domain values are a Cartesian product of $2^{dom(A1)}$, $2^{dom(A2)}$,…, $2^{dom(An)}$, where $A1$, $A2$,…, An are the components of the composite A: $2^{dom(A)} = 2^{dom(A1)} \times \ldots \times 2^{dom(An)}$.

	Non-temporal Composite Attribute	**Temporal Composite Attribute**
Non-temporal Component Attribute	Current composite with current components (*1*)	A single temporal element associated with all the components (*2*)
Temporal Component Attribute	Each component has an independent temporal element (*3*)	N/A (*4*)

Table 3: The semantics of a temporal/non-temporal composite/component attribute combinations

If the component attribute(s) are temporal, then each component has distinct temporal elements associated with them (cell *3* of Table 3). If the composite attribute is temporal, it implies that all the component attributes have the same temporal element associated with them (cell *2* of Table 3). As may be evident, it is not possible for both the component and the composite attributes to be temporal (cell *4* of Table 3), because it would imply that the application needs to capture both the history of a composite as a group and independently. If the user needs to capture history of the components separately, then modeling history associated with the composite would be extraneous. Similarly, if the user requirements call for capturing history of the composite (i.e., a temporal element associated with all the components) that for the components is not required. The cell *1* of Table 3 shows non-temporal composite attribute with non-temporal component attribute for a conventional conceptual model.

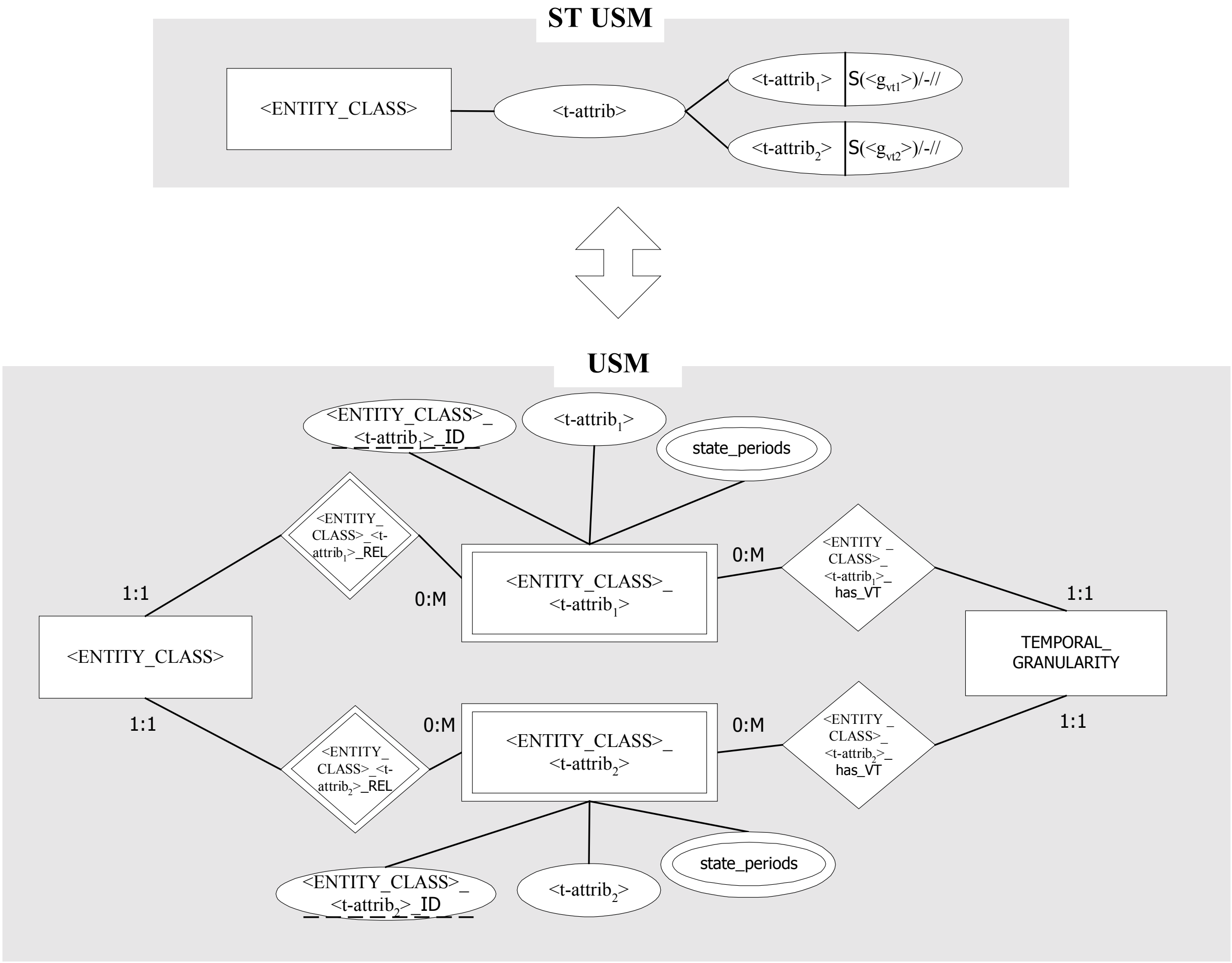

ST USM
<ENTITY_CLASS>
<t-attrib>
<t-attrib$_1$>
S(<g$_{vt1}$>)/-//
<t-attrib$_2$>
S(<g$_{vt2}$>)/-//
USM
<ENTITY_CLASS>_<t-attrib$_1$>_ID
<t-attrib$_1$>
state_periods
<ENTITY_CLASS>_<t-attrib$_1$>_REL
1:1
0:M
<ENTITY_CLASS>_<t-attrib$_1$>
0:M
<ENTITY _CLASS>_<t-attrib$_1$>_has_VT
1:1
TEMPORAL_GRANULARITY
<ENTITY_CLASS>
1:1
0:M
<ENTITY_CLASS>_<t-attrib$_2$>_REL
<ENTITY_CLASS>_<t-attrib$_2$>
0:M
<ENTITY _CLASS>_<t-attrib$_2$>_has_VT
1:1
<ENTITY_CLASS>_<t-attrib$_2$>_ID
<t-attrib$_2$>
state_periods

Figure 9: Temporal component attribute in ST USM and its semantics in USM

As shown in Figure 9, the semantics of a temporal composite attribute with temporal components is similar to a simple temporal attribute (Figure 8). As shown in Figure 10, a temporal composite attribute has a single temporal element associated with all the components of the attribute.

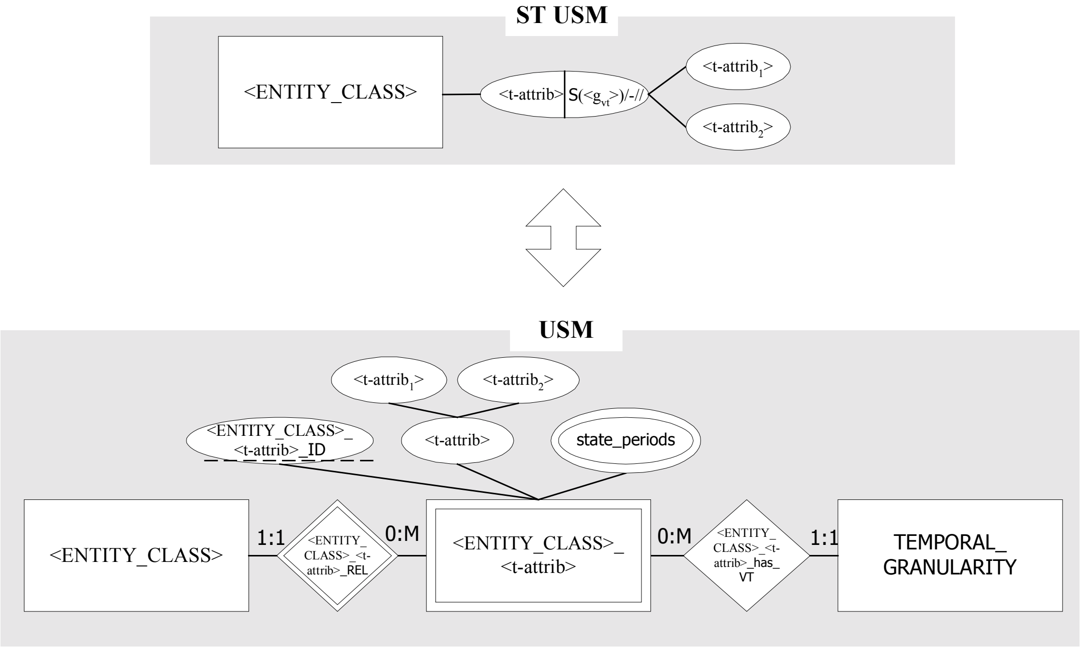


Figure 10: Temporal composite in ST USM and its semantics in USM

6.2.1.4 Bitemporal Attribute

The semantics of a bitemporal attribute (Figure 11) are similar to those of a bitemporal entity class (Figure 4); attribute_tstamp, a composite multi-valued attribute is composed of et_tstamp and tt_tstamp.

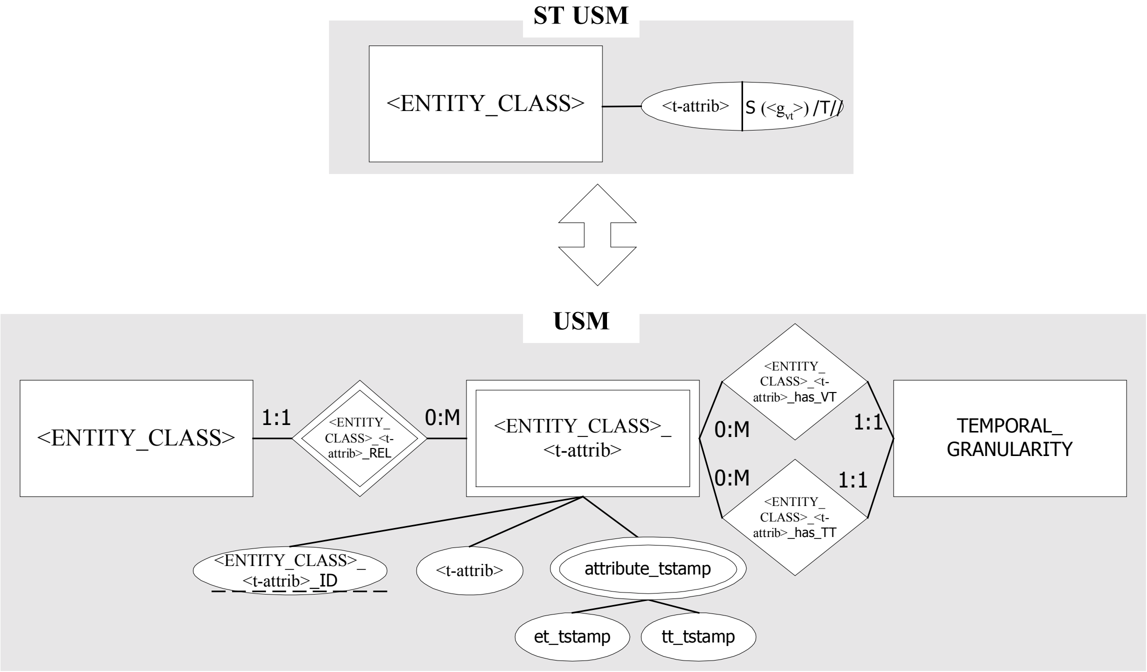


Figure 11: Bitemporal attribute in ST USM and its semantics in USM

There are two relationships between an ⟨ENTITY_CLASS⟩_⟨t-attrib⟩ and a TEMPORAL_GRANULARITY that define the association between an attribute *a*, and its corresponding valid time granularity (i.e., granularity_name = $\langle g_{vt} \rangle$) and transaction time granularity (i.e., granularity_name=g_{tt}). The definition-based constraints are similar to 6.1.1, 6.1.3–6.1.5 and bitemporal constraints related to a bitemporal attribute are similar to constraints 6.1.9–6.1.11.

6.2.2 Spatial Attribute

Spatial attributes represent properties that are geo-referenced with respect to the Earth and spatial annotations render the schema spatial.

As shown in Table 4, a non-spatial and spatial attribute can be associated with a non-spatial entity and spatial (or time-varying spatial) entity class.

	Non-Spatial Attribute	**Spatial Attribute**
Non-Spatial Entity Class	Conventional entity/attribute (*1*)	Space-varying attribute values (*2*)
Spatial/ Time-varying Spatial Entity Class	Value of the attribute applies to the entire geometry of the spatial entity (*3*)	Space-varying attribute values within the geometry of a spatial entity (*4*)

Table 4: The semantics of spatial/non-spatial attribute/entity combinations

A spatial attribute of a spatial entity class implies that the attribute has different values at different points within the geometry of the spatial entity (cell *4* of Table 4). For example, if spring_site (a point) were defined as a spatial attribute of spatial entity SPRING (a region), it would imply that multiple spring_site are spatially located within SPRING. A non-spatial attribute of a spatial entity implies that the value of an attribute applies to the entire geometry of the object (cell *3* of Table 4). For example status, a non-spatial attribute of BORE_HOLE_SITE, relates to an entire borehole site. A spatial attribute of a non-spatial entity implies a space-varying attribute (cell *2* of Table 4). The annotation syntax for a spatial attribute is the same as that for a spatial entity; thus, we do not describe the annotation syntax again.

We describe the semantics associated with a spatial attribute. A spatial attribute *A* of an entity class *E* with geometry is associated with an *attribute geometry* function $\psi_{E,A,xy}$. If both entity and attribute are spatial with the same granularity then some combinations of geometry are invalid: (i) if an entity is represented as a point, its attribute cannot be represented as a line/region; and (ii) if an entity is represented as a line, its attribute cannot be represented as a region. Based on the application requirements, a data analyst may annotate the attribute as spatial; e.g., in Figure 12 the attribute ⟨s-attrib⟩ is specified as spatial with an associated annotation phrase "//P(g_{xy})/P(g_{xy})/-".

Each entity *e* of ⟨ENTITY_CLASS⟩ optionally is related to ⟨ENTITY_CLASS⟩_⟨s-attrib⟩, a weak class corresponding to the spatial attribute of the entity. The semantics associated with a multi-valued and composite attribute are similar to those described in the previous section.

We now describe the constraints on a simple spatial attribute. The definition-based granularity constraints 6.1.15–6.1.17 will hold for HORIZONTAL_SPATIAL_GRANULARITY.

Constraint 6.2.7: cf. constraint 6.1.12

$\forall a \in S(\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{s-attrib}\rangle)$,

$a.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{s-attrib}\rangle\text{\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (granularity\_name)} = \langle g_{xy}\rangle$

Constraint 6.2.8: cf. constraint 6.1.13

$\forall a \in S(\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{s-attrib}\rangle), \exists g \in a.\text{geo}$,

$a.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{s-attrib}\rangle\text{\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (extent.xy\_minimum)} \leq$
$g.\text{xy\_point} \leq a.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{s-attrib}\rangle\text{\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (extent.xy\_maximum)}$

Constraint 6.2.9: cf. constraint 6.1.15

$\forall a \in S(\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{s-attrib}\rangle), \forall g \in a.\text{geo}$,

$a.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{s-attrib}\rangle\text{\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (extent.xy\_minimum)} \leq$
$g.\text{xy\_point} \leq a.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{s-attrib}\rangle\text{\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (extent.xy\_maximum)}$

Constraint 6.2.10: cf. constraint 6.2.5. If both the entity class and its attribute are spatial, the union of the geometry of an (spatial) attribute (*ag*) must be equal to the geometry of the associated entity (*eg*).

$\forall e \in S(\langle\text{ENTITY\_CLASS}\rangle), \forall eg \in e.\text{geo}, \forall k \in eg.\text{xy\_point}$,

$\exists a \in e.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{s-attrib}\rangle\text{\_REL}.\langle\text{ENTITY\_CLASS}\rangle\_\langle\text{s-attrib}\rangle, \exists ag \in a.\text{geo}$,

$k \in ag.\text{xy\_point}$

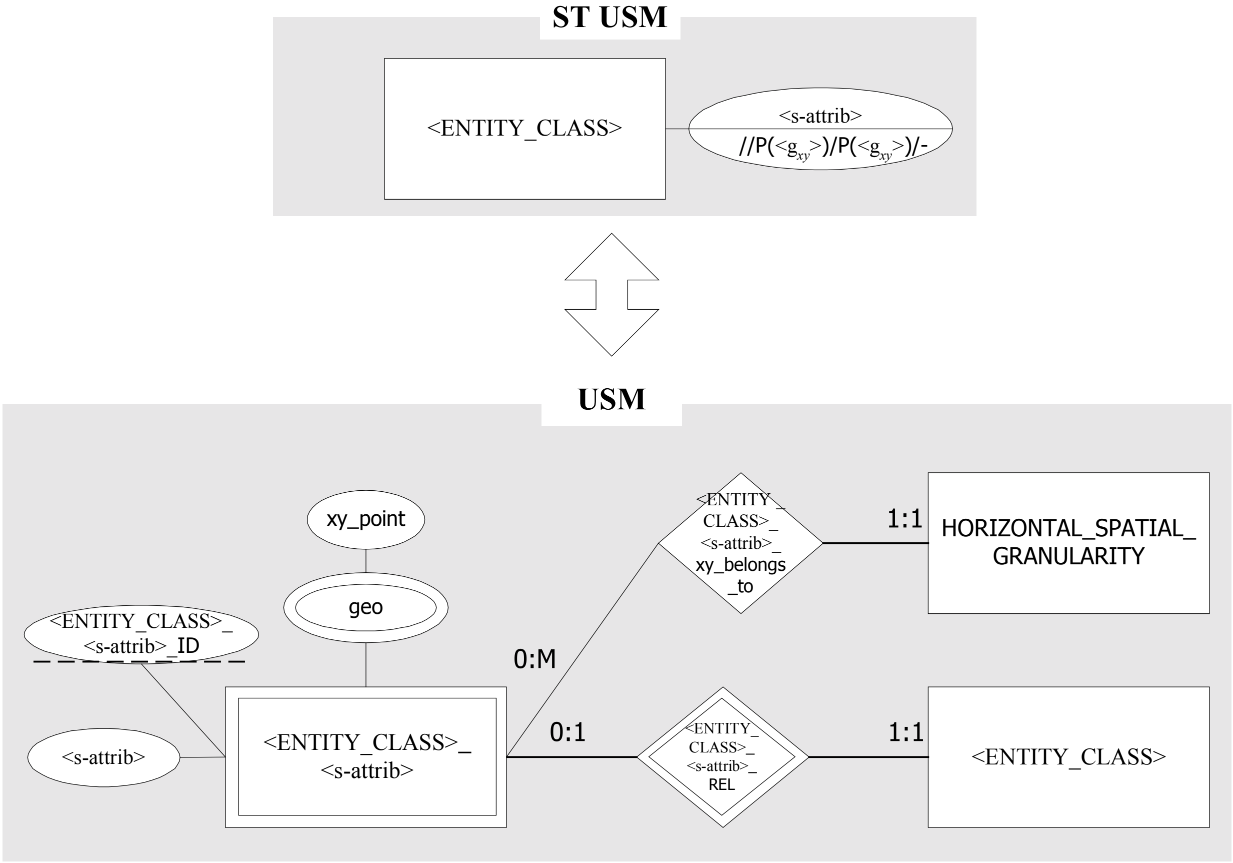

ST USM
<ENTITY_CLASS>
<s-attrib>
//P(<g_xy>)/P(<g_xy>)/-
USM
xy_point
geo
<ENTITY_CLASS>_<s-attrib>_ID
<s-attrib>
<ENTITY_CLASS>_<s-attrib>
<ENTITY_CLASS>_<s-attrib>_xy_belongs_to
0:M
1:1
HORIZONTAL_SPATIAL_GRANULARITY
0:1
<ENTITY_CLASS>_<s-attrib>_REL
1:1
<ENTITY_CLASS>

Figure 12: Spatial attribute in ST USM and its semantics in USM

6.2.3 Time-Varying Spatial Attribute

Like a time-varying spatial entity class, a time-varying spatial attribute is interpreted in two ways: (i) an attribute with associated geometry and valid time; and (ii) an attribute whose spatial geometry varies over time.

	Time-varying Spatial Attribute
Non-spatial, non-temporal Entity Class	Time-varying spatial property of currently valid entities (*1*)
Temporal Entity Class	Time-varying spatial property within the lifespan specified by entities (*2*)
Spatial Entity Class	Time-varying spatial property of currently valid entities within the geometry specified by spatial entity (*3*)
Time-varying Spatial Entity Class	Time-varying spatial property within the lifespan and geometry of a time-varying spatial entity (*4*)

Table 5: The semantics of time-varying spatial attribute/entity combinations

In the case (ii), a spatial attribute can change its shape and/or position over time. A time-varying spatial attribute (on *xy*-plane) with time-varying geometry has an associated $\psi_{E,A,xy}$ and $\varphi_{E,A,vt}$ that defines the position of the attribute in terms of spatial granularities ($SG_{E,A,xy}$) and temporal granularity ($TG_{E,A,vt}$).For time-varying spatial entities with a time-varying spatial attribute (cell *4* of Table 5), the spatial and temporal projection of a time-varying spatial attribute is equal to the spatial and temporal projection of a time-varying spatial entity. This is a direct implication of the semantics of an attribute in a conventional conceptual model where an attribute can have a value when the associated entity is defined.

Figure 13 shows a time-varying spatial attribute 〈st-attrib〉 in ST USM and the corresponding semantics in USM.

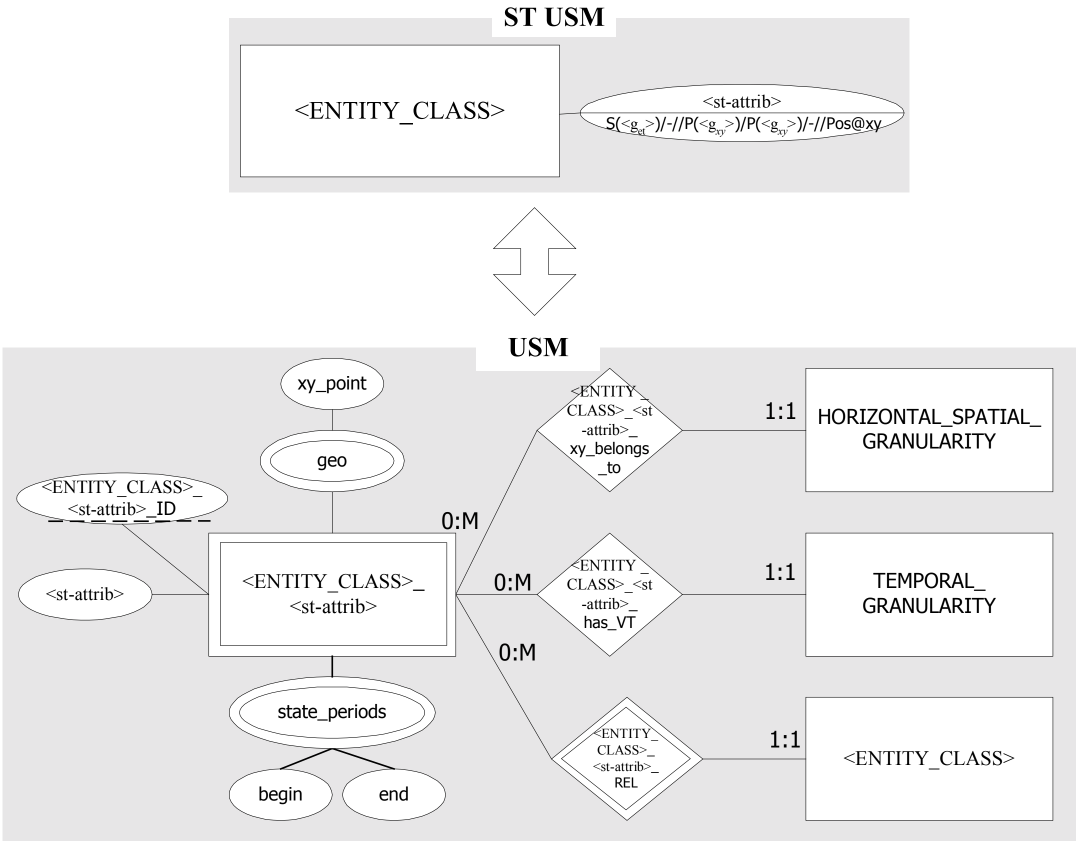

ST USM
<ENTITY_CLASS>
<st-attrib>
S(<g_et>)/-//P(<g_xy>)/P(<g_xy>)/-//Pos@xy
USM
xy_point
geo
<ENTITY_CLASS>_<st-attrib>_ID
<st-attrib>
<ENTITY_CLASS>_<st-attrib>
state_periods
begin
end
0:M
0:M
0:M
<ENTITY_CLASS>_<st-attrib>_xy_belongs_to
<ENTITY_CLASS>_<st-attrib>_has_VT
<ENTITY_CLASS>_<st-attrib>_REL
1:1
1:1
1:1
HORIZONTAL_SPATIAL_GRANULARITY
TEMPORAL_GRANULARITY
<ENTITY_CLASS>

Figure 13: Time-varying spatial attribute in ST USM and its semantics in USM

The constraints on a time-varying spatial attribute are similar to those on a time-varying spatial entity. Each point of time is associated with an attribute geometry.

6.3 Interaction Relationship

An interaction relationship relates members of an entity set with those of one or more entity sets.

6.3.1 Temporal Relationship

A temporal relationship implies that we want to keep track of the evolution of the interaction between temporal entities in a relationship. For example, if is_in were a temporal relationship between two temporal entity classes BORE_HOLE_SITE and PUMPLIFT, it would imply that an application might include queries like "In the last six months, what are the various pump lifts associated with borehole site 12345." A temporal/non-temporal relationship may be associated temporal/non-temporal entities.

As shown in Table 6, a temporal relationship and the participating entities include four possibilities.

	Non-temporal Relationship	**Temporal Relationship**
Non-temporal Entity Class	Currently valid relationship between currently valid entities (*1*)	N/A (*2*)
Temporal/ Time-varying Spatial Entity Class	Currently valid relationship among temporal (time-varying spatial) entities (*3*)	Temporal relationship among temporal (time-varying spatial) entities (*4*)

Table 6: The semantics of temporal/non-temporal relationship/entity class combinations

A temporal relationship can be defined only when all the participating entities are also temporal. Thus, a temporal relationship between non-temporal entities (cell *2* of Table 6) is not legal in ST USM, as that would imply existence of a relationship even when the associated entities do not exist (since the existence time of entities is unknown). This is again a direct implication of the semantics of a relationship in a conventional conceptual model where relationships can only be defined between entities that exist. If the participating entity classes are temporal but the relationship is not (cell *3* of Table 6) the entities participating in the relationship should be valid *now*. Formally, if $E_1, \ldots, E_n$ are temporal entity classes participating in a non-temporal relationship R, $\forall\ (e_1,\ldots, e_n) \in S(R)$, $\varphi_{E1,et}(e_1,UC) \wedge\ldots\wedge \varphi_{En,et}(e_n,UC)$.

Formally, a temporal relationship is associated with existence predicates such that $\forall\ (e_1,\ldots, e_n) \in S(R)$, $\forall\ i$, $\varphi_{R,et}(e_1,\ldots, e_n, i) \Rightarrow \varphi_{E1,et}(e_1, i) \wedge\ldots\wedge \varphi_{En,et}(e_n, i)$ that defines the lifespan of the relationship. Figure 14 illustrates the case where the participating entity classes, i.e., ⟨ENTITY_CLASS$_1$⟩ and ⟨ENTITY_CLASS$_2$⟩, are temporal and the relationship is also temporal. ⟨rel⟩_VT is the interaction class associated with ⟨rel⟩. ⟨role$_1$⟩ and ⟨role$_2$⟩ describe the role that an entity takes in a relationship and may not be explicitly specified in the schema. Usually the role name is specified when the same entity class participates with different roles in a relationship. Additionally, ⟨rel⟩_VT_ID is the key for the interaction class ⟨rel⟩_VT.

In this case, the temporal element of the relationship is constrained to be a subset of the temporal elements of the participating entities. Constraint 6.3.1 is a sequenced analog of the USM relationship where a relationship is defined between entities that exist.

Constraint 6.3.1: The temporal element of a temporal relationship is a subset of the intersection of the temporal elements of the participating entities.

$$\forall e_1 \in S(E_1), e_2 \in S(E_2), \ldots, e_n \in S(E_n), \forall(e_1, e_2, \ldots, e_n) \in \langle \text{rel} \rangle, \exists p_1 \in e_1.\text{state\_periods}, \ldots, \exists p_n \in e_1.\text{state\_periods},$$
$$\forall rp \in (e_1, e_2, \ldots, e_n).\text{state\_periods}, p_1.\text{begin} \leq \ldots \leq p_n.\text{begin} \leq rp.\text{begin} < rp.\text{end} \leq p_\text{n}.\text{end} \leq \ldots \leq p_1.\text{end}$$

Other constraints on a temporal relationship are similar to constraints 6.1.1–6.1.8. It is possible that the granularity of the relationship is not the same as the granularity of the participating entities; the *granularity lattice* [14] is used to convert the granularities.

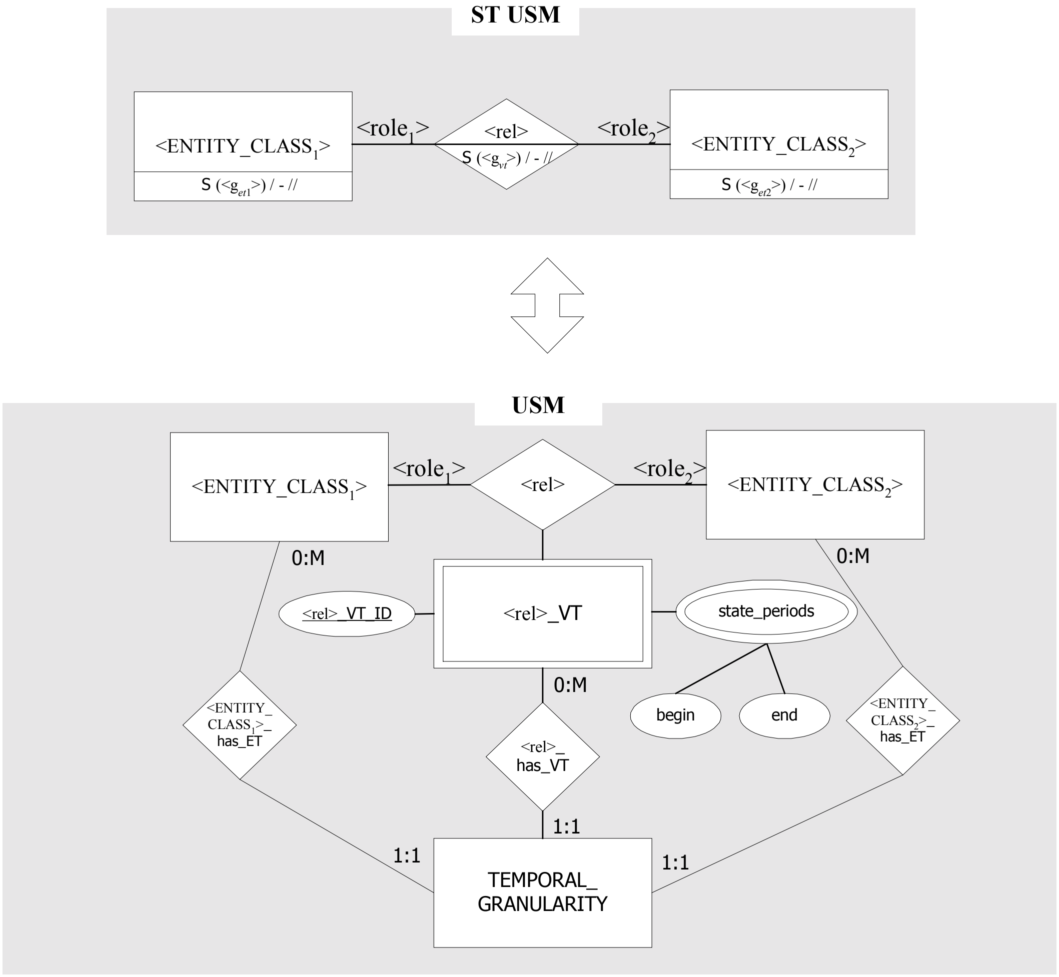


Figure 14: Temporal relationship in ST USM and its semantics in USM

The *cardinality constraint*—a *structural constraint*—specifies the number of relationship instances that an entity can participate in [16]. As described in Section 5, a cardinality constraint of 1:M associated with a pumplift in relationship is_in implies that a borehole site can have a minimum of 1 and a maximum of many (M) associated pumplifts. Temporal entities participating in a temporal relationship renders this constraint sequenced and is referred to as the *snapshot cardinality constraint* [80]. For example, in Figure 2 the relationship is_in for each BORE_HOLE_SITE can have a minimum of 0 and a maximum of many (M) associated PUMPLIFT; each PUMPLIFT is associated with exactly one BORE_HOLE_SITE (1:1) in the relationship is_in. When these entities are annotated as temporal (in Figure 25), it implies that, e.g., in the relationship is_in each BORE_HOLE_SITE can have a minimum of 0 and a maximum of many (M) associated PUMPLIFT at *each point in time* (represented by granularity index).

Figure 15 illustrates the case where both the participating ⟨ENTITY_CLASS⟩ and the relationship are bitemporal. ⟨rel⟩_VT_TT_ID is the key for the interaction class ⟨rel⟩_VT_TT.

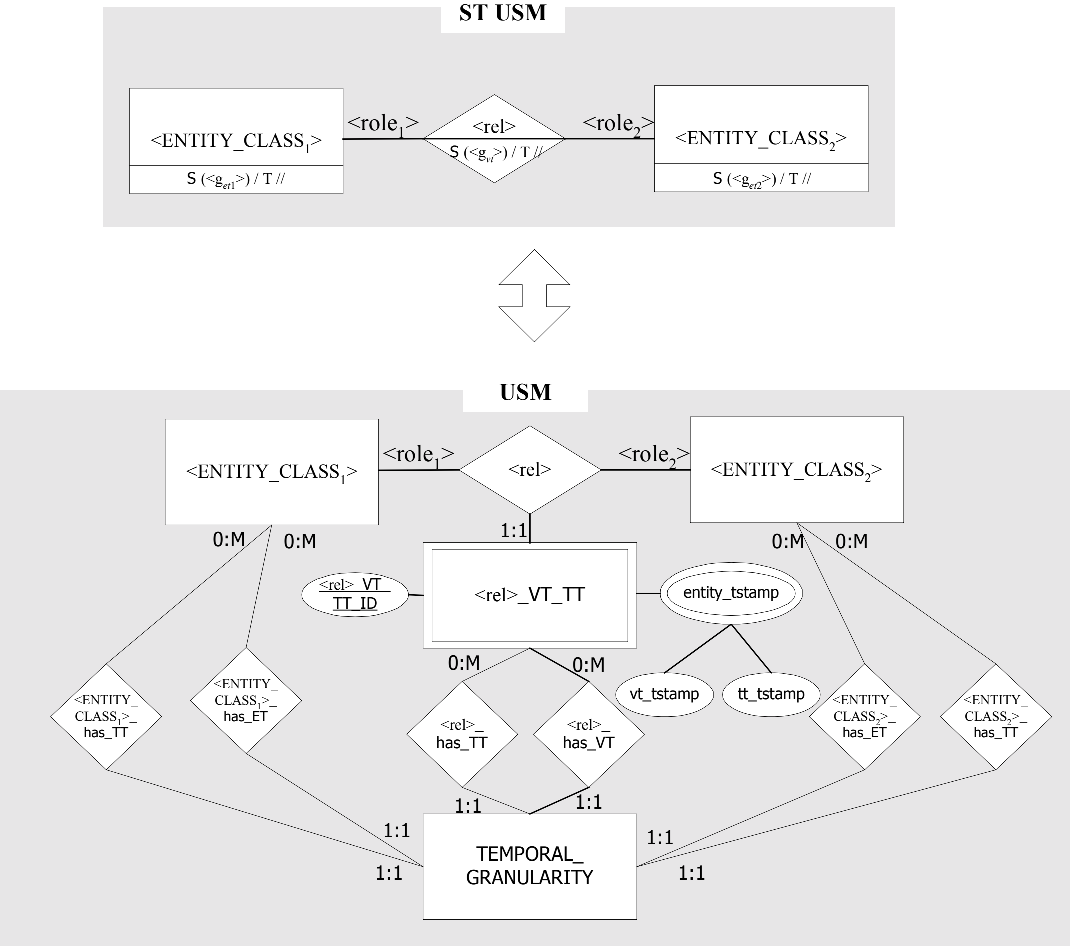


Figure 15: Bitemporal relationship in ST USM and its semantics in USM

Transaction time constraints on a temporal relationship between temporal entities are similar to constraints 6.1.9–6.1.11.

Additional temporal cardinality constraints could be depicted, though not graphically in the schema. For example, a non-sequenced cardinality constraint specified over the lifespan of the entities is an example of a non-sequenced constraint and is referred to as the *lifetime cardinality constraint* [80]. For example, a snapshot and lifetime cardinality constraint of 0:1 implies "at most one ever" and a snapshot constraint of 1:1 and a lifetime cardinality of 1:n implies "one at a time", where a cardinality of n represents a number greater than 1. As per the definition of these constraints [80]: ⟨snapshot maximum⟩ > 0; ⟨lifetime maximum⟩ > 0; ⟨snapshot minimum⟩ ≤ ⟨snapshot maximum⟩ ≤ ⟨lifetime maximum⟩; and ⟨snapshot minimum⟩ ≤ ⟨lifetime minimum⟩ ≤ ⟨lifetime maximum⟩.

Additionally, the user may also explicitly specify other non-sequenced temporal constraints between temporal entities participating in a temporal relationship. These constraints, based on Allen's predicates, are constraints on the temporal projections of entities in a binary temporal relationship. Temporal constraints between temporal entities in a relationship may be overlaps, finished, during, starts and equals; disjoint is obviously not possible. A non-sequenced constraint needs to be consistent with the sequenced constraints; so meets is not possible either.

Constraint 6.3.2: Temporal constraint overlaps.
$\forall (e_1, e_2) \in S(\langle \text{rel} \rangle), e_1 \in S(\langle \text{ENTITY\_CLASS}_1 \rangle), e_2 \in S(\langle \text{ENTITY\_CLASS}_2 \rangle),$
$\forall p_1 \in e_1.\text{state\_periods}, \exists p_2 \in e_2.\text{state\_periods}, (p_1.\text{end} > p_2.\text{begin}) \wedge (p_1.\text{begin} < p_2.\text{begin}) \wedge (p_1.\text{end} < p_2.\text{end})$

Constraint 6.3.3: Temporal constraint finished.
$\forall (e_1, e_2) \in S(\langle \text{rel} \rangle), e_1 \in S(\langle \text{ENTITY\_CLASS}_1 \rangle), e_2 \in S(\langle \text{ENTITY\_CLASS}_2 \rangle),$
$\forall p_1 \in e_1.\text{state\_periods}, \exists p_2 \in e_2.\text{state\_periods}, (p_1.\text{begin} > p_2.\text{begin}) \wedge (p_1.\text{end} = p_2.\text{end})$

Constraint 6.3.4: Temporal constraint during.
$\forall (e_1, e_2) \in S(\langle \text{rel} \rangle), e_1 \in S(\langle \text{ENTITY\_CLASS}_1 \rangle), e_2 \in S(\langle \text{ENTITY\_CLASS}_2 \rangle),$
$\forall p_1 \in e_1.\text{state\_periods}, \exists p_2 \in e_2.\text{state\_periods}, (p_1.\text{begin} < p_2.\text{begin} \leq p_2.\text{end} < p_1.\text{end})$

Constraint 6.3.5: Temporal constraint starts.
$\forall (e_1, e_2) \in S(\langle \text{rel} \rangle), e_1 \in S(\langle \text{ENTITY\_CLASS}_1 \rangle), e_2 \in S(\langle \text{ENTITY\_CLASS}_2 \rangle),$
$\forall p_1 \in e_1.\text{state\_periods}, \exists p_2 \in e_2.\text{state\_periods}, (p_1.\text{begin} = p_2.\text{begin}) \wedge (p_1.\text{end} < p_1.\text{end})$

Constraint 6.3.6: Temporal constraint equals.
$\forall (e_1, e_2) \in S(\langle \text{rel} \rangle), e_1 \in S(\langle \text{ENTITY\_CLASS}_1 \rangle), e_2 \in S(\langle \text{ENTITY\_CLASS}_2 \rangle),$
$\forall p_1 \in e_1.\text{state\_periods}, \exists p_2 \in e_2.\text{state\_periods}, (p_1.\text{begin} = p_2.\text{begin}) \wedge (p_1.\text{end} = p_2.\text{end})$

The temporal constraints meets^{-1}, overlaps^{-1}, finished^{-1}, during^{-1} and starts^{-1} can be similarly defined.

6.3.2 Spatial Relationship

A spatial relationship refers to relationships with the geometries of the participating entities.

As shown in Table 7, a spatial relationship can be associated with a spatial (or time-varying spatial) and non-spatial entity class.

	Non-spatial Relationship	**Spatial Relationship**
Non-spatial Entity Class	Traditional relationship semantics (*1*)	N/A (*2*)
At least one Spatial/ Time-varying Spatial Entity Class	Association among entities; not related to geometries (*3*)	Association with at least one geometry of the participating entities (*4*)

Table 7: The semantics of spatial/non-spatial relationship/class combinations

A spatial relationship between one or more spatial (or time-varying spatial) entities (cell *4* of Table 7) specifies an explicit relationship between the geometries of the participating entities; in other words, this implies an association with at least one spatial projection among the participating entities. For example, occurs_in ("//-/-/L(ft)") is a spatial relationship between BORE_HOLE and LITHOLOGY. While LITHOLOGY is a non-spatial entity class, BORE_HOLE is a spatial entity class. A spatial relationship means that different parts of a borehole (along the *z*-dimension) are associated with different lithologies. Capturing this relationship would enable answering queries like "What is the lithology at a depth of 50 feet of a specified borehole?" A non-spatial relationship between spatial entities (cell *3* of Table 7) implies a relationship among entities that is unrelated to its geometry. A spatial relationship between non-spatial entities (cell *2* of Table 7) is illegitimate as it contradicts the definition of a spatial relationship, i.e., it is a relationship between the spatial projection of spatial entities.

A spatial relationship has constraint on its existence predicate: $\forall\ (e_1,\ldots, e_n) \in S(R), \forall\ i, \psi_{E1,et}(e_1,\ldots, e_n, i) \Rightarrow \psi_{E1,et}(e_1, i) \wedge\ldots\wedge \psi_{En,et}(e_n, i)$.

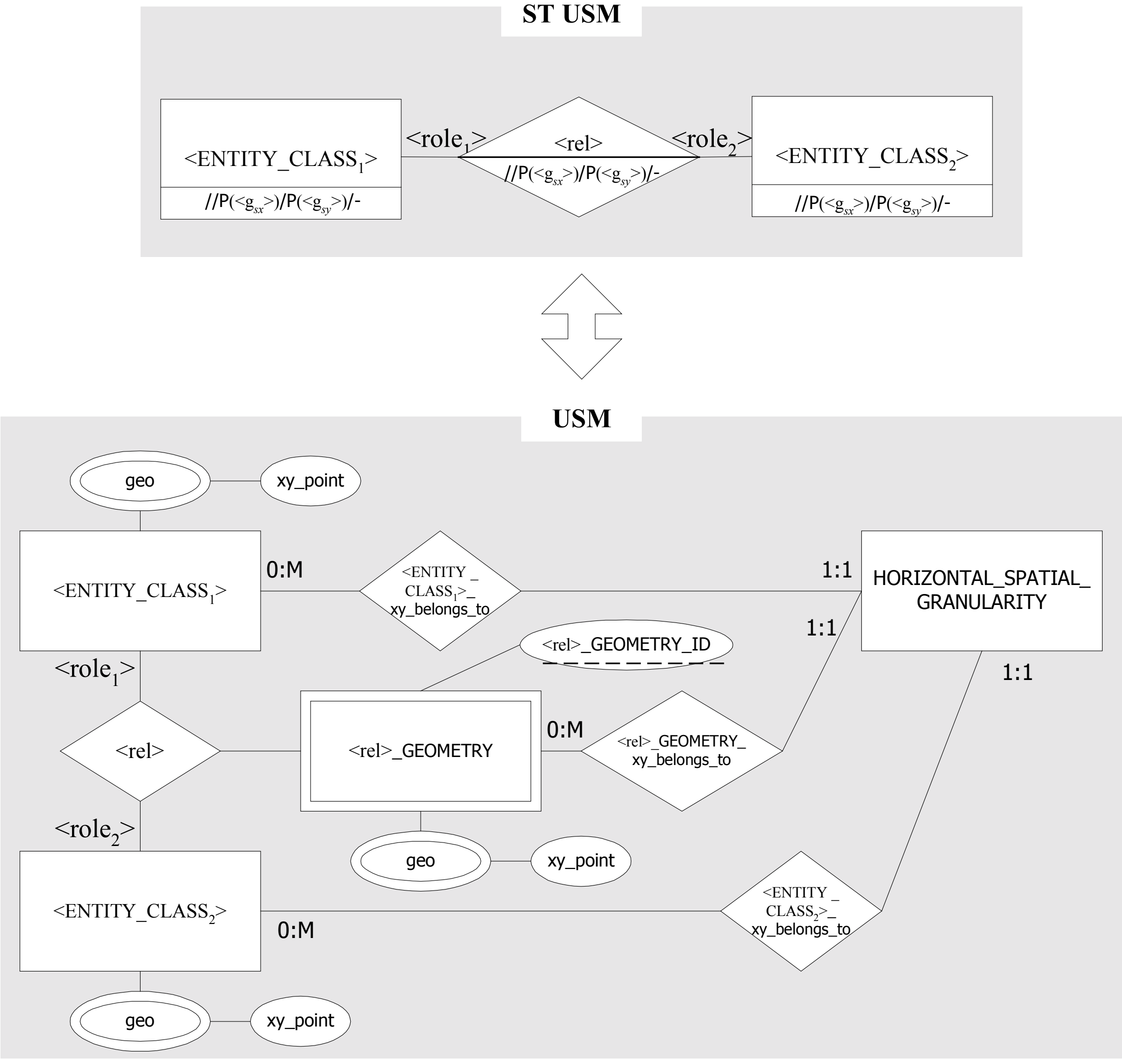

ST USM
<ENTITY_CLASS$_1$>
//P(<g$_{sx}$>)/P(<g$_{sy}$>)/-
<role$_1$>
<rel>
//P(<g$_{sx}$>)/P(<g$_{sy}$>)/-
<role$_2$>
<ENTITY_CLASS$_2$>
//P(<g$_{sx}$>)/P(<g$_{sy}$>)/-
USM
geo
xy_point
<ENTITY_CLASS$_1$>
0:M
<ENTITY_CLASS$_1$>_xy_belongs_to
1:1
HORIZONTAL_SPATIAL_GRANULARITY
1:1
<rel>_GEOMETRY_ID
1:1
<role$_1$>
<rel>
<rel>_GEOMETRY
0:M
<rel>_GEOMETRY_xy_belongs_to
<role$_2$>
geo
xy_point
<ENTITY_CLASS$_2$>
0:M
<ENTITY_CLASS$_2$>_xy_belongs_to
geo
xy_point

Figure 16: Spatial interaction relationship in ST USM and its semantics in USM

This is a spatial sequenced analog of the USM where a relationship can be defined only between (existing) entities. For a spatial relationship between spatial entities, the spatial projection on a relationship instance is a subset of the spatial projection on the corresponding participating spatial entities.

As shown in Figure 16, a spatial relationship is a relationship between the geometry of the participating entities. ⟨rel⟩_GEOMETRY represents an interaction class, and $\langle \text{role}_1 \rangle$ and $\langle \text{role}_2 \rangle$ in the schema exemplify that roles are not altered when an ST-USM schema is mapped to a translated USM schema. Note that typically the role names are not always explicitly shown on the schema; roles name are included when the role of an entity in a relationship is ambiguous (e.g., in a recursive relationship). Additionally, it is assumed that the spatial granularity of the participating entities and the relationship are the same.

Constraints on spatial relationships are similar to constraints 6.1.12–6.1.18 described before.

Constraint 6.3.8: The spatial projection of the relationship is a subset of the spatial projection of the participating spatial entities.

$\forall\, e_1 \in \mathcal{S}(\langle \text{ENTITY\_CLASS}_1 \rangle),\, e_2 \in \mathcal{S}(\langle \text{ENTITY\_CLASS}_2 \rangle),\, \forall\, (e_1, e_2) \in \mathcal{S}(\langle \text{rel} \rangle),$

$\forall ge_1 \in e_1.\text{geo},\, \forall ge_2 \in e_2.\text{geo},\, \forall gr \in (e_1, e_2).\langle \text{rel} \rangle\text{\_GEOMETRY.geo},$

$\forall k \in gr.\text{xy\_point} \Rightarrow \exists i \in ge_1.\text{xy\_point} \wedge \exists j \in ge_2.\text{xy\_point}$

Topological constraints like meets, equals, inside and covers—a type of spatial non-sequenced constraints—among spatial regions could be added, outside the of the graphical schema. The semantics of topological constraints is specified in Table 8, where $A°$ denotes the interior of A (where A is a spatial region composed of non-empty set of points) and δA denotes the boundary of A [62]. For example A disjoint B implies that intersection of the boundaries and interior is empty (∅).

	$\delta A \cap \delta B$	$A° \cap B°$	$\delta A \cap B°$	$A° \cap \delta B$
A* disjoint *B	∅	∅	∅	∅
A* meets *B	¬∅	∅	∅	∅
A* equals *B	¬∅	¬∅	∅	∅
A* inside *B	∅	¬∅	¬∅	∅
A* covered_by *B	¬∅	¬∅	¬∅	∅
A* contains *B	∅	¬∅	∅	¬∅
A* covers *B	¬∅	¬∅	∅	¬∅
A* overlaps *B	¬∅	¬∅	¬∅	¬∅

Table 8: Eight possible topological relationships between two spatial regions [62]

All of these constraints (except disjoint) are possible between any two spatial entities represented by a region. This is so because the non-sequenced spatial constraints must be consistent with the sequenced spatial constraints.

6.3.3 Time-Varying Spatial Relationship

A time-varying spatial relationship implies the need to capture the temporal changes in a spatial relationship.

As shown in Table 9, a time-varying spatial relationship can only be specified between time-varying spatial entities. A time-varying spatial relationship implies the need to store time-varying geometries, which in turn means that the participating entities must be time-varying spatial also. An inherent constraint is that the time related to time-varying spatial relationship should be a subset of the intersection of the time related to the interacting time-varying spatial entities.

	Time-varying Spatial Relationship
Non-spatial, non-temporal Entity Class	N/A
Temporal Entity Class	N/A
Spatial Entity Class	N/A
Time-varying Spatial Entity Class/ At least one time-varying spatial and all temporal entity class	Time-varying spatial association between time-varying spatial entities

Table 9: The semantics of time-varying spatial relationship/entity combinations

The semantics of a time-varying spatial relationship is similar to that of a spatial relationship. The interaction class ⟨rel⟩_GEOMETRY in Figure 16 is also associated with a multi-valued composite attribute state_periods that represents the temporal element associated with ⟨rel⟩_GEOMETRY. Additionally, ⟨rel⟩_GEOMETRY is also related to TEMPORAL_GRANULARITY via a relationship ⟨rel⟩_GEOMETRY_has_ET.

6.4 Weak Entity Class

Members of a weak entity class depend on members of another class—sometimes referred to as an *identifying entity class* [16]—for their existence. In other words, an entity class that does not have a key attribute of its own may be referred to as a weak entity class.

6.4.1 Temporal Weak Entity Class

The semantics of a weak entity class are similar to those of a (strong) entity class described in the previous section. We describe the additional constraints associated with a weak entity class.

As shown in Table 10, a temporal/non-temporal weak entity class can be associated with a temporal/non-temporal identifying class.

	Non-temporal Weak Entity Class	**Temporal Weak Entity Class**
Non-temporal Identifying Entity Class	Currently valid weak entity associated with currently valid identifying entities (*1*)	N/A (*2*)
Temporal Identifying Entity Class	Currently valid weak entity associated with temporal identifying owner (*3*)	Temporal weak entity associated with a temporal identifying owner (*4*)

Table 10: The semantics of temporal/non-temporal weak entity class/identifying entity class combinations

If both the identifying owner and the weak entity class are temporal (cell *4* of Table 10), then the temporal element of the weak entity must be a subset of the temporal entity of its corresponding identifying entity. If the identifying entity class is temporal and the weak entity class is not (cell *3* of Table 10), it implies that the identifying entities with corresponding weak entities should be currently valid. A non-temporal identifying owner of a temporal weak entity (cell *2* of Table 10) is not legitimate since the temporal element of the identifying entity is unknown and the constraint between the temporal element of a weak entity and its identifying entity cannot be defined. As a result, the weak entity might exist for time granules when the strong entity does not exist. If the weak entity is allowed to exist for time periods that the identifying entity does not exist, then it invalidates the very definition of the weak entity class. Cell *1* of Table 10 represents the conventional semantics of currently valid weak entity associated with currently valid identifying entity.

Constraint 6.4.1: For cell *3* in Table 10 above, the entity *e* of the identifying entity class *E* must exist *now*. In this case, the identifying relationship is defined between a non-temporal weak entity and the temporal identifying entity as of *now*; in other words, $\varphi_{E,et}$ (*e*, *UC*).

$$\forall\, w \in \langle\text{WEAK\_ENTITY\_CLASS}\rangle,\ \exists\, e \in \langle\text{ENTITY\_CLASS}\rangle,\ (e.\langle\text{WEAK\_RELATIONSHIP}\rangle.\langle\text{WEAK\_ENTITY\_CLASS}\rangle{=}w) \wedge (\exists p \in e.\text{state\_periods},\ p.\text{end} = UC)$$

Constraint 6.4.2: For cell *4* of Table 10, the temporal element of the identifying entity *e* must always be a subset of the temporal element of the weak entity *w*.

$$\forall\, w \in \langle\text{WEAK\_ENTITY\_CLASS}\rangle,\ \forall pw \in w.\text{state\_periods},\ \exists\, e \in \langle\text{ENTITY\_CLASS}\rangle,\ \exists\, pe \in e.\text{state\_periods},\ e.\langle\text{WEAK\_RELATIONSHIP}\rangle.\langle\text{WEAK\_ENTITY\_CLASS}\rangle = w,\ pe.\text{begin} \le pw.\text{begin} < pw.\text{end} \le pe.\text{end}$$

The weak entity and its identifying owner are not constrained to have the same granularity. There are three possibilities—the granularity of weak entity $\langle g_{etw} \rangle$ (i.e., $\text{granularity\_name}_w$) is finer, coarser or equal to that of the identifying owner $\langle g_{ete} \rangle$ (i.e., $\text{granularity\_name}_e$). The constraint 6.4.2 described above is for special case when the granularities of the weak entity and its identifying owner are equal. If the granularity of the weak entity ($\text{granularity\_name}_w$) is finer-than/coarser-than that of its identifying owner ($\text{granularity\_name}_e$), the constraint 6.4.2 may be expressed such that the granularity of the weak entity is *cast* [14] to the granularity of the identifying owner.

$$\forall\, w \in \langle\text{WEAK\_ENTITY\_CLASS}\rangle,\ \forall pw \in w.\text{state\_periods},\ \exists\, e \in \langle\text{ENTITY\_CLASS}\rangle,\ \exists\, pe \in e.\text{state\_periods},\ e.\langle\text{WEAK\_RELATIONSHIP}\rangle.\langle\text{WEAK\_ENTITY\_CLASS}\rangle = w,$$
$$pe.\text{begin} \le cast(\text{granularity\_name}_w(pw.\text{begin}),\ \text{granularity\_name}_e) < cast(\text{granularity\_name}_w(pw.\text{end}),\ \text{granularity\_name}_e) \le pe.\text{end}$$

6.4.2 Spatial Weak Entity Class

As shown in Table 10, a spatial/non-spatial weak entity class can be associated with a spatial/non-spatial identifying owner.

	Non-spatial Weak Entity Class	**Spatial Weak Entity Class**
Non-spatial Identifying Entity Class	Traditional semantics of conventional weak entity class/ identifying owner (*1*)	Geo-referenced weak entity (*2*)
Spatial Identifying Entity Class	The geometry of weak entity is assumed to be the same as that of its identifying owner (*3*)	Spatial weak entity associated with a spatial identifying owner (*4*)

Table 11: The semantics of spatial/non-spatial weak entity class/identifying entity class combinations

If both the identifying owner and the weak entity are spatial (cell *4* of Table 10), then the geometry of a weak entity is constrained to be a subset of the geometry of its identifying owner. A non-spatial identifying class associated with a weak spatial entity (cell *2* of Table 10) implies a geo-referenced weak entity having an associated geometry.

The additional constraint on a spatial weak entity is similar to that on a temporal weak entity class, constraint 6.4.2.

6.4.3 Time-Varying Spatial Weak Entity Class

A time-varying spatial weak entity class implies that the identifying owner is temporal or time-varying spatial. The constraints on a time-varying spatial weak entity class are similar to those on a weak spatial and temporal entity class.

6.5 Subclass

A subclass is an abstraction in which similar objects are related to a higher-level generic object called a superclass. In this section, we describe the semantics of a temporal, spatial and time-varying spatial subclass.

6.5.1 Temporal Subclass

A subclass and a superclass is a type of abstraction that allows modeling the same real-world object at different levels of abstraction. An application may require capturing the lifespan of the specific abstraction (i.e., the subclass) and not that of the generic abstraction (i.e., the superclass). For example, if GROUND_WATER_STATION is modeled as a superclass and SPRING is its temporal subclass, it implies that the application needs to store the lifespan of SPRING and only currently valid GROUND_WATER_STATION is required for the application.

As shown in Table 12, a temporal/non-temporal superclass can be associated with a temporal/non-temporal subclass.

	Non-temporal Subclass	**Temporal Subclass**
Non-temporal Superclass	Currently valid superclass and subclass (*1*)	Currently valid temporal subclass associated with a superclass (*2*)
Temporal Superclass	N/A (*3*)	Temporal subclass associated with a temporal superclass (*4*)

Table 12: The semantics of temporal/non-temporal subclass/superclass combinations

Cell *1* of Table 12 represents the existing semantics of conventional superclass/subclass. If both a subclass and its superclass are specified as temporal (cell *4* of Table 12), then each subclass entity can have a temporal element independent of the corresponding superclass entity. However, the temporal element of the subclass is constrained to be a subset of the temporal element of the superclass. A temporal superclass by its definition implies that the subclass is also temporal; that is why cell *3* of Table 12 is not legitimate. This is a direct implication of the semantics of subclass/superclass in a conventional conceptual model where the subclass inherits all the properties of the superclass. For example, if GROUND_WATER_STATION—a superclass—is modeled as temporal, then SPRING inherits all the properties and relationships of GROUND_WATER_STATION including its lifespan. As exemplified by cell *2* of Table 12, a non-temporal superclass may have a temporal subclass. However, the subclass entities must be currently valid.

Formally, a subclass entity *C* with existence time is associated with an existence predicate $\varphi_{C,et}$ that is similar to that of an entity existence predicate $\varphi_{E,et}$ described in an earlier section. However, there is one additional constraint: if both the subclass and its superclass are temporal (e.g., both GROUND_WATER_STATION and SPRING are modeled as temporal), the temporal element associated with an entity *e*/*C* of the subclass (i.e., *C*) must be a subset of the temporal element of the corresponding entity *e* of the superclass (i.e., *E*). Formally, $\forall\ e \in S(E), \forall\ i, (\exists\ e/C \in S(C) \wedge \varphi_{E,et}(e, i)) \Rightarrow \varphi_{C,et}(e/C, i)$. If only the superclass is annotated as temporal (Figure 17) then entities of the subclass inherit the temporal elements of the entities of the superclass. In this case, both the entity *e* in the superclass (i.e., *E*) and a corresponding entity *e*/*C* in the subclass (i.e., *C*) have the same temporal element associated with them; i.e., the subclass inherits the temporal element of the superclass. Formally, $\forall\ e \in S(E), \forall\ i, \exists\ e/C \in S(C), \varphi_{E,et}(e, i) \Rightarrow \varphi_{C,et}(e,i)$.

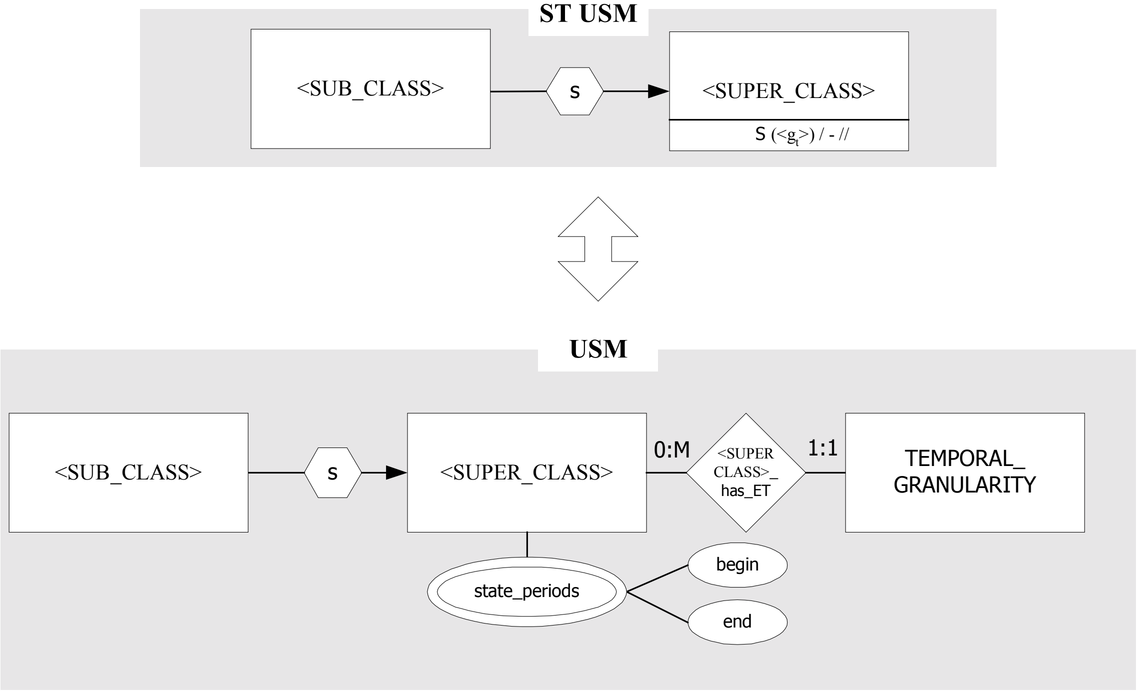


Figure 17: Subclass with temporal superclass in ST USM and its semantics in USM

Besides the constraints associated with a temporal entity class described in a previous section, we describe below the constraints specific to a temporal subclass.

Constraint 6.5.1: If only the superclass is modeled as temporal then the temporal element associated with an entity of the subclass is equal to the temporal element of the corresponding entity of the superclass.

$\forall e/C \in \langle \text{SUB\_CLASS} \rangle, \forall cp \in e/C.\text{state\_periods}, \exists e \in \langle \text{SUPER\_CLASS} \rangle, \exists ce \in e.\text{state\_periods},$
$ce.\text{begin} = cp.\text{begin} \wedge ce.\text{end} = cp.\text{end}$

If a superclass is non-temporal and the subclass is annotated as temporal (Figure 18), then the user is interested in the evolution of the entities of the subclass and just needs to store the currently valid entities of the superclass. A constraint associated with a temporal subclass is that the entity belonging to the temporal subclass must be valid *now*.

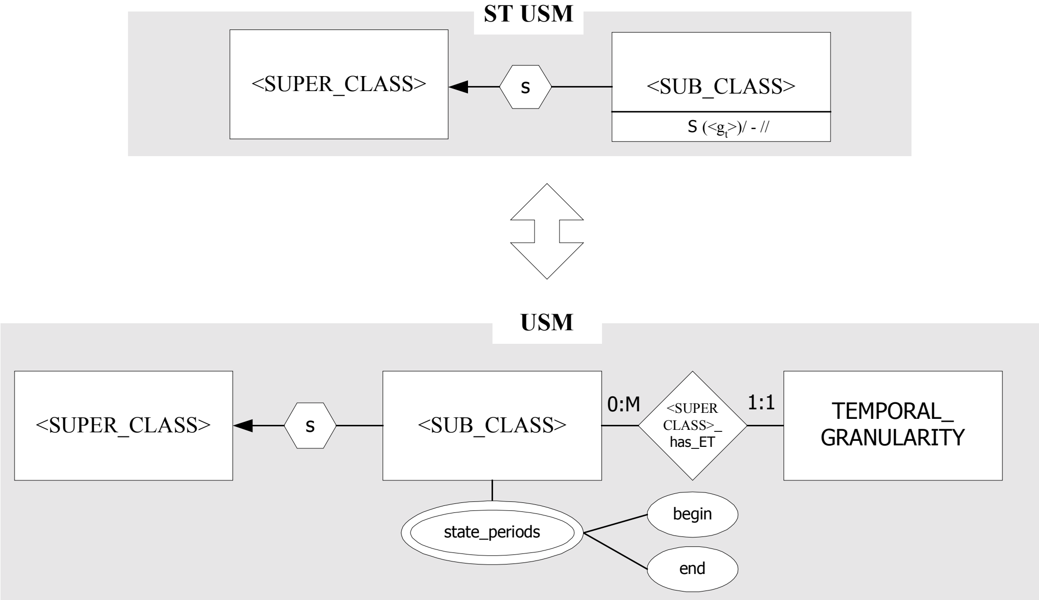


Figure 18: Temporal subclass in ST USM and its semantics in USM

Constraint 6.5.2 is similar to the constraint 6.4.1 described before.

Constraint 6.5.2: The temporal entity e/C of the subclass C must exist *now*. In other words, $\varphi_{C,et}$ (e/C, UC).

$\forall\ e/C \in \langle\text{SUB\_CLASS}\rangle, \exists\ p \in e/C.\text{state\_periods}, (p.\text{end} = UC \vee p.\text{begin} = UC)$

If both the superclass and the subclass are specified as temporal then an entity in the ⟨SUPER_CLASS⟩ and the corresponding entity in the ⟨SUB_CLASS⟩ have separate temporal elements associated with them (Figure 19). However, there is a constraint between the temporal elements of an entity e/C in the ⟨SUB_CLASS⟩ and the corresponding entity e in the ⟨SUPER_CLASS⟩. The temporal element of e/C must be a subset of the temporal element of e.

Constraint 6.5.3: If a superclass and a subclass are temporal then the temporal element associated with the entity of a subclass must be a subset of the temporal element of the corresponding entity of a superclass.

$\forall\ e/C \in \langle\text{SUB\_CLASS}\rangle, \forall cp \in\ e/C.\text{state\_periods}, \exists\ e \in \langle\text{SUPER\_CLASS}\rangle, \exists\ ep \in\ e.\text{state\_periods},$
$ep.\text{begin} \le cp.\text{begin} \le cp.\text{end} \le ep.\text{end}$

If the granularity of the subclass entity ($\text{temporal\_granularity}_{\text{sub}}$) and the corresponding entity in the superclass ($\text{temporal\_granularity}_{\text{sup}}$) is not the same, then the constraint 6.5.3 needs to be modified to the following.

$\forall\ e/C \in \langle\text{SUB\_CLASS}\rangle, \forall cp \in\ e/C.\text{state\_periods}, \exists\ e \in \langle\text{SUPER\_CLASS}\rangle, \exists\ ep \in\ e.\text{state\_periods},$
$ep.\text{begin} \le cast(\text{temporal\_granularity}_{\text{sub}}(cp.\text{begin}), \text{temporal\_granularity}_{\text{sup}}) \le$
$cast(\text{temporal\_granularity}_{\text{sub}}(cp.\text{end}), \text{temporal\_granularity}_{\text{sup}}) \le ep.\text{end}$

In a conventional conceptual model, the relationship among subclasses is modeled by the *disjointness/ overlapping* [16] constraint. This constraint on a temporal subclass induces a temporally sequenced constraint: overlapping or disjoint at each point in time.

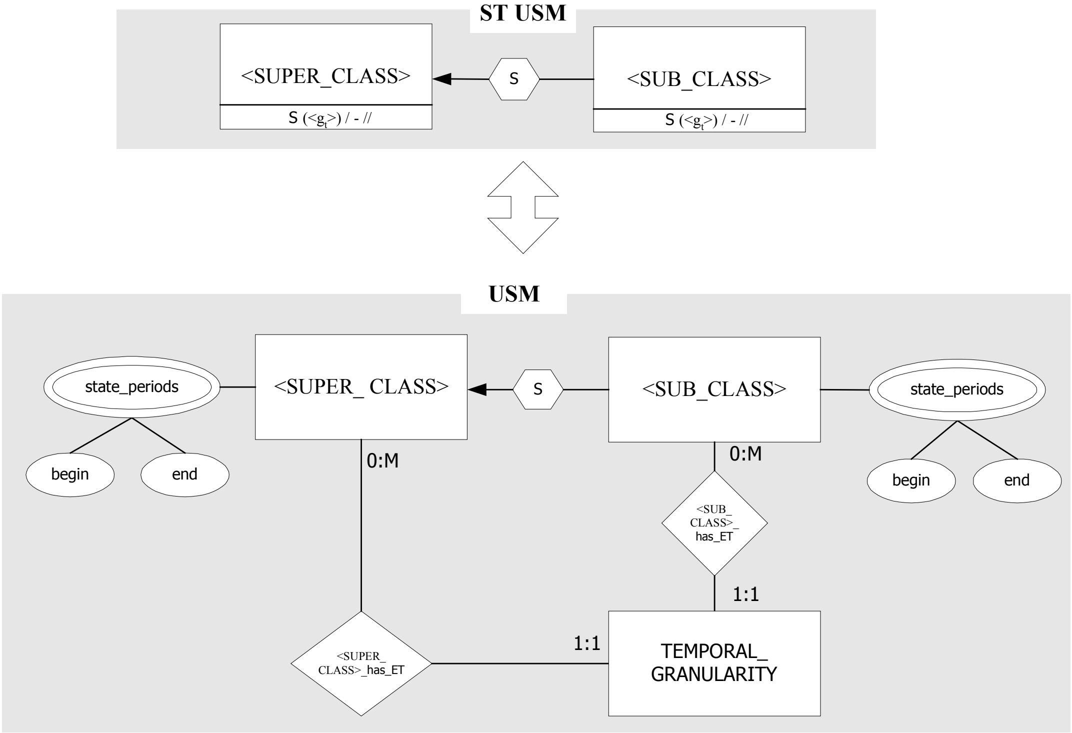


Figure 19: Temporal superclass and subclass in the ST USM and its semantics in USM

Non-sequenced [75] temporal relationships among temporal subclasses can be defined outside the graphical schema using Allen's predicate [3]. The *temporal disjointness* refers to an entity that belongs to multiple subclasses at different time granules and refers to before. If an entity is allowed to exist in multiple subclasses at the same time, then it may be referred to as *temporal overlapping* and refers to overlaps, finished, during, starts or equals.

The temporal subclass can be: (i) *temporal attribute-defined*, where the *discriminator* attribute itself may be temporal and an entity may belong to multiple subclasses (of same superclass) at different times; and (ii) *temporal roster-defined*, where the user may model an entity in different subclasses (of same superclass) at different times. A discriminator is an attribute of the ⟨SUPER_CLASS⟩ whose values determine the target ⟨SUB_CLASS⟩; the valid values of the discriminator are NULL (i.e., an entity in ⟨SUPER_CLASS⟩ does not belong to any ⟨SUB_CLASS⟩) and ⟨SUB_CLASS⟩. An entity may belong to a subclass for some period of time and it may make a transition to another subclass. There are two possibilities: (a) transitioning entity ceases to be an entity in source subclass and (b) transitioning object is a member of source and destination subclasses [89]. A temporal class relationship with multi-valued temporal discriminator indicates that the transitioning object can be a member of multiple ⟨SUBCLASS⟩ at any point in time, i.e., belongs to the case (b). A single-valued temporal discriminator would indicate the case (a).

6.5.2 Spatial Subclass

An application may need to capture geometry related to the specific type (i.e. subclass) and not the generic type. For example, GROUND_WATER_STATION may not be modeled as spatial while its subclasses BORE_HOLE and SPRING are both modeled as spatial.

As shown in Table 13, a spatial/non-spatial superclass can be associated with spatial/non-spatial subclass.

	Non-spatial Subclass	**Spatial Subclass**
Non-spatial Superclass	Traditional semantics of superclass/ subclass (*1*)	Only the subclass has an associated geometry (*2*)
Spatial Superclass	N/A (*3*)	Geometry of the subclass is constrained to be a subset of the geometry of the superclass (*4*)

Table 13: The semantics of spatial/non-spatial subclass/superclass combinations

If the superclass is specified as spatial, the entities of the subclass are implicitly spatial; that is why cell *3* of Table 13 is not legitimate.

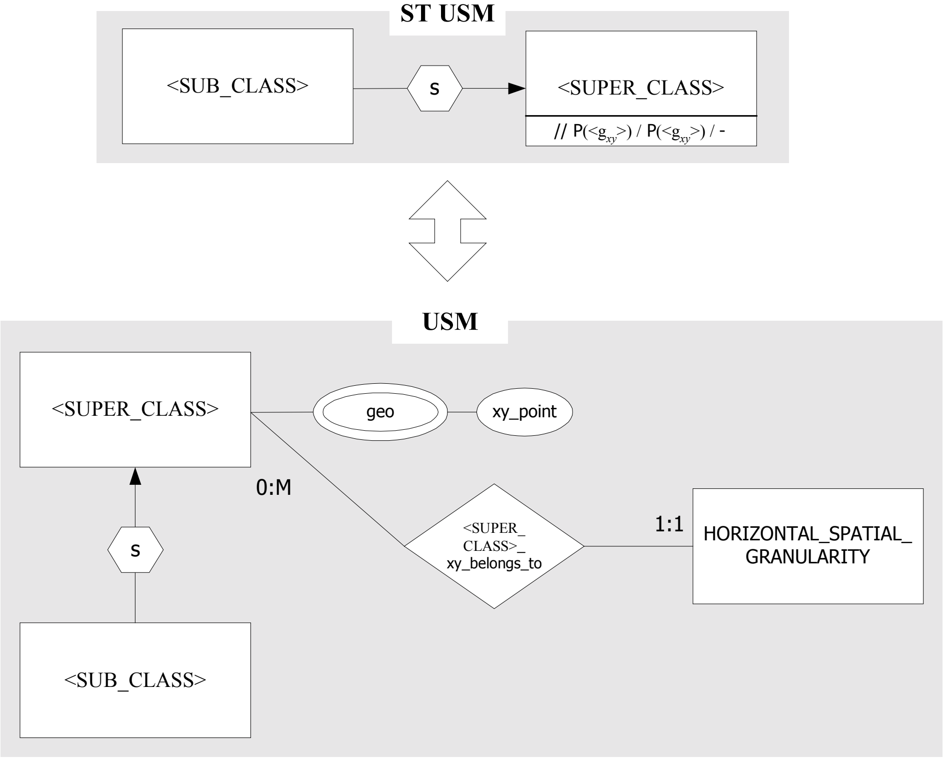


Figure 20: Spatial superclass in ST USM and its semantics in USM

If only the ⟨SUPER_CLASS⟩ is specified as spatial (Figure 20), then the ⟨SUB_CLASS⟩ inherits the geometry of the ⟨SUPER_CLASS⟩. Both the entity *e* of the ⟨SUPER_CLASS⟩ and the corresponding entity *e*/*C* of the ⟨SUB_CLASS⟩ have the same geometry. That is, $\forall i, \forall e \in S(E), (\psi_{E,xy}(e, i) \wedge \exists\, e/C \in S(C)) \Rightarrow \psi_{C,xy}(e/C, i)$; in other words, the spatial projection of a superclass entity and the corresponding entity in the subclass are the same. If the ⟨SUPER_CLASS⟩ is non-spatial and the ⟨SUB_CLASS⟩ is annotated as spatial (cell *2* of Table 13), it implies that the user needs to model the specific (i.e., subclass) abstraction as spatial and the spatiality of the generic abstraction (i.e., superclass) is not important; this is especially useful to model spatial entities with different types of associated geometries that have some common (non-spatial) properties. For example, GROUND_WATER_STATION is modeled as a non-spatial superclass with SPRING and BORE_HOLE as its spatial subclasses; here, SPRING is represented as a region while BORE_HOLE is represented as a point in the horizontal plane and line in the vertical plane. The semantics related to this case are similar to the semantics of a spatial entity class (e.g., Figure 5).

If both the ⟨SUPER_CLASS⟩ and the ⟨SUB_CLASS⟩ are modeled as spatial, then the spatial properties of both the superclass and subclass are important for the user. In this case, the geometry of the ⟨SUB_CLASS⟩ entity must be a subset of the geometry of the corresponding entity in the ⟨SUPER_CLASS⟩. Formally,

$$\forall i, \forall e/C \in S(C), \psi_{C,xy}(e/C, i) \Rightarrow \psi_{E,xy}(e, i)$$

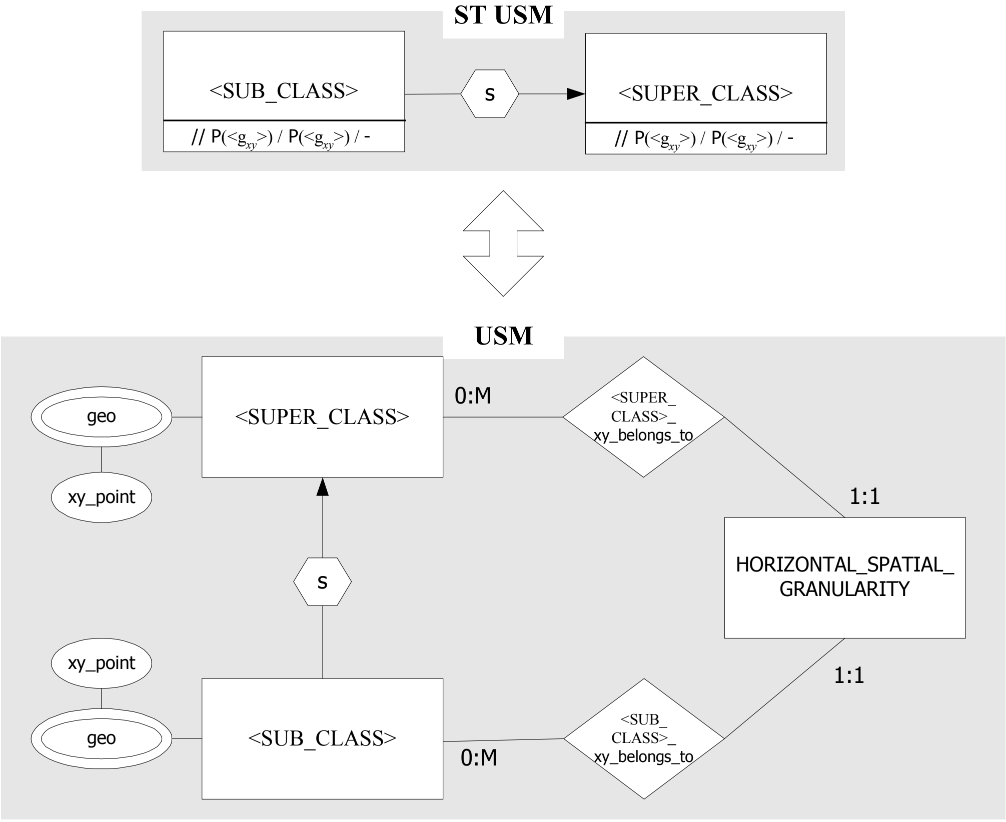

ST USM
<SUB_CLASS>
// P(<gxy>) / P(<gxy>) / -
S
<SUPER_CLASS>
// P(<gxy>) / P(<gxy>) / -
USM
geo
xy_point
<SUPER_CLASS>
0:M
<SUPER_CLASS>_xy_belongs_to
1:1
HORIZONTAL_SPATIAL_GRANULARITY
S
1:1
xy_point
geo
<SUB_CLASS>
0:M
<SUB_CLASS>_xy_belongs_to

Figure 21: Spatial subclass with a spatial superclass in ST USM and its semantics in USM

As shown in Figure 21, the spatial ⟨SUB_CLASS⟩ and spatial ⟨SUPER_CLASS⟩ are associated with a corresponding geometry, possibly of different type. If the subclass and superclass have the same spatial granularity (e.g., degree) and the spatial projection of the superclass is a point (or a line), the subclass can have geometry of point (or point and line only) only.

Similar to a temporal subclass, a spatial subclass can be: (i) *spatial attribute-defined*, the discriminator attribute may be spatial and an entity in a superclass may belong to multiple spatial subclasses at different points in space; (ii) *spatial roster-defined*, the user may ascribe the same entity to different subclasses at different points in space (represented by granularity index).

The relationships between subclasses modeled by a disjointness/overlapping constraint can be extended to include spatiality. The *spatial disjointness constraint* can be specified by disjoint while meets, overlaps, covers, covered_by, inside, contains and equals shows different levels of *spatial overlapping constraints*.

6.5.3 Time-Varying Spatial Subclass

A time-varying spatial subclass may have: (i) spatial superclass, (ii) temporal superclass, (iii) non-spatial and non-temporal superclass, or (iv) time-varying spatial superclass. An entity in a time-varying spatial subclass, with spatial superclass, must have an associated geometry that is a subset of the geometry of the corresponding entity in the superclass. Similarly, an entity in a time-varying spatial subclass must have a temporal element that is a subset of the temporal element of the corresponding entity in the superclass. A time-varying spatial subclass with non-spatial and non-temporal superclass imposes no constraints on its members.

An inherent constraint associated with a time-varying spatial subclass and a time-varying spatial superclass is that the temporal element and geometry of the subclass must be a subset of the temporal element and geometry of the superclass. There are two sub-cases here: (i) changing shape or position of the subclass; and (ii) changing shape or position of both the superclass and the subclass. For sub-case (i), various shapes or positions of the subclass must be within the geometry specified for the superclass. For sub-case (ii), each position or shape of an entity in the subclass must be a subset of the corresponding shape or position of entity in the superclass.

6.6 Composite Class

A composite relationship defines a new class called a composite class that has other class (or subsets of a class) as its members. Each member of a composite class is referred to as a *composite* and members of a composite are derived from the base class. All composite classes are strong, i.e., composites do not depend on entities of any other class for their existence.

6.6.1 Temporal Composite Class

A composite entity class may have existence time and transaction time associated with it, which is indicated by annotation phrase S($\langle g_t \rangle$) (or E($\langle g_t \rangle$)) and T, respectively. A temporal composite class represents the evolution of a composite over time where the composite may be homogeneous or heterogeneous. If the classes that are members of the composite are subclasses of a common class, the composite is referred to as *homogeneous*, else *heterogeneous*. For example, DISCHARGE is the base class of IO_DISCHARGE, which is represented as a composite class. A set of DISCHARGE enumerated (Enum) by the user constitutes IO_DISCHARGE for, say, project 1525 (where project(IO_DISCHARGE) = project 1525). In this example, the composite class is heterogeneous.

As shown in Table 14, a non-temporal/temporal composite can be associated with a temporal/non-temporal component. The composite is a higher-level abstraction whose existence is independent of the existence of its components. If both the composite and the component are specified as temporal (cell *4* of Table 14), it implies that the user wants to capture the lifespan of the composite and the time periods within its lifespan when it is composed of its components. In this case, the temporal elements of components are constrained to be a subset of the temporal elements of the corresponding composite. A temporal composite associated with a non-temporal component (cell *2* of Table 14) implies that the user is interested in the lifespan of the higher-level abstraction only. Suppose PUMPLIFT_TYPE is temporal composite and PUMPLIFT is its non-temporal component.

	Non-temporal Composite Class	**Temporal Composite Class**
Non-temporal Component Class	Currently valid composite and component (*1*)	Temporal composite associated with currently valid component (*2*)
Temporal Component	Temporal component associated with currently valid composite (*3*)	Temporal composite associated with temporal component (*4*)

Table 14: The semantics of temporal/non-temporal composite/component combinations

The lifespan of, e.g., low power, medium power and high power (i.e., entities of PUMPLIFT_TYPE) pump lift types is required for the application and there may be different sets of pump lifts in the specified category at different time granularity index of a PUMPLIFT_TYPE. A temporal component associated with a currently valid composite (cell *3* of Table 14) implies that the application will not need to answer questions like "What were the low power pump lifts that have been operational since June 28, 2001." The cell *4* of Table 14 represents the conventional semantics of a composite/component.

Formally, a composite entity class with existence time is associated with an existence predicate $\varphi_{E^*,et}$ that is similar to the entity existence predicate $\varphi_{E,et}$ described in an earlier section.

$$\varphi_{E^*,et} : \mathbf{P}(S(E)) \times \mathbf{Z} \rightarrow \mathbf{B}$$

The constraints specified for temporal composite existence predicate are similar to those on a temporal entity existence predicate defined in an earlier section. Additionally, if both the composite class and its the component class ($e \in S(E)$) are temporal, then $\forall e_1,\ldots,e_n/E^* \in S(E^*)$, $\varphi_{E^*,et}(e_1,\ldots,e_n/E^*, i) \Rightarrow \varphi_{E,et}(e_1, i) \wedge \ldots \wedge \varphi_{E,et}(e_n, i)$. Intuitively a composite can exist at those time granularity indexes when its components exist. If the composite is temporal and its component is not, then this constraint is not required. If the component is temporal while its composite is not, each temporal component is required to be legitimate *now*. In other words, $\forall e_1,\ldots,e_n/E^*$, $\varphi_{E,et}(e_1,UC) \wedge \varphi_{E,et}(e_2,UC) \wedge\ldots\wedge \varphi_{E,et}(e_n,UC)$.

Figure 22 illustrates the case where a temporal ⟨COMPOSITE_CLASS⟩ has a ⟨BASE_CLASS⟩ that is also temporal. Besides the generic constraints associated with a temporal entity class, there is one additional constraint on a temporal composite with a temporal component.

Constraint 6.6.1: The temporal element of a composite is subsumed by the temporal elements of the participating components. In other words, the begin index of the composite is greater than or equal to the largest begin index of the participating base members and the end index of the composite is less than or equal to the smallest end index of the participating base members.

$\forall\ e_1, e_2,\ldots, e_n/E^* \in \langle\text{COMPOSITE\_CLASS}\rangle$, $\forall\ cp \in (e_1, e_2,\ldots, e_n/E^*)$.state_periods, $\exists\ e_1, e_2,\ldots, e_n \in \langle\text{BASE\_CLASS}\rangle$,
$\exists\ cb_1 \in e_1$.state_periods,…, $\exists\ cb_n \in e_n$.state_periods,
cb_1.begin ≤ … ≤ cb_n.begin ≤ cp.begin ≤ cp.end ≤ cb_1.end ≤ … ≤ cb_n.end

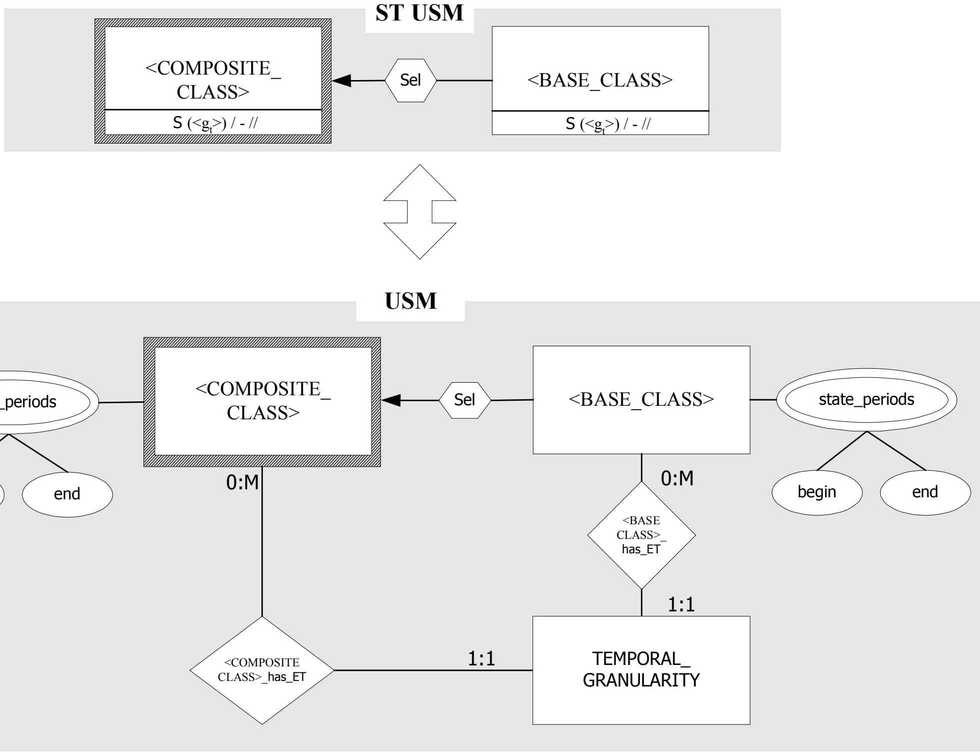


Figure 22: Temporal composite class in ST USM and its semantics in USM

A temporal composite class with a temporal component class may be: (i) *temporal-attribute selected*, where the composite is formed on the basis of different values of a temporal attribute over time; (ii) *temporal enumerated*, the composite has different time-varying components enumerated by the user. Suppose PUMPLIFT_TYPE is temporal composite class and PUMPLIFT is its temporal component class. Let us assume that PUMPLIFT_TYPE is based on average power consumed, e.g., low efficiency, medium efficiency and high efficiency. In this example, average power consumed is a temporal attribute whose value determines the composite. Every month, based on the average consumption of power, a pumplift is categorized into appropriate PUMPLIFT_TYPE. If PUMPLIFT_TYPE is determined by the user, say every month, then temporal composite is referred to as temporal enumerated.

6.6.2 Spatial Composite Class

The semantics of a spatial composite class are similar to that of a temporal composite class.

As shown in Table 15, a spatial/non-spatial composite can be associated with a spatial/non-spatial component.

	Non-spatial Composite	Spatial Composite
Non-spatial Component	Traditional composite/ component semantics (*1*)	Spatial composite associated with non-spatial component (*2*)
Spatial Component	Spatial component associated with composite (*3*)	Spatial composite associated with spatial component (*4*)

Table 15: The semantics of spatial/non-spatial composite/component combinations

If both the component and the composite are specified as spatial (cell *4* of Table 15), then the spatial projection of the components is constrained to be a subset of the spatial projection of the composite. Here, an inherent constraint is that the geometry of a spatial composite class includes the geometry of its base class members.

$\forall e_1, e_2,\ldots,e_n/E^*, \psi_{E^*,xy}(e_1, e_2,\ldots,e_n/E^*, i) \Rightarrow \psi_{E,xy}(e_1,i) \wedge \psi_{E,xy}(e_2,i) \wedge\ldots\wedge \psi_{E,xy}(e_n,i)$

6.6.3 Time-Varying Spatial Composite Class

A time-varying spatial composite class may have (i) a temporal base class, (ii) a spatial base class, (iii) non-spatial and non-temporal base class, and (iv) a time-varying spatial base class. A time-varying spatial composite with temporal base class is such that the temporal element of the base class members is equal to temporal element of the corresponding composite. Similarly constraint holds for case (ii). For case (iv), four possibilities exist, (a) shape or position of both composite and the base class members changes, (b) shape of a base class and position of the composite class changes and vice-versa. In both the cases, the geometry of the composite must include the geometry of the base class members at all points of time. Formally, $\forall\ e_1, e_2,\ldots,e_n/E^*, \forall j, \xi_{E^*,xy,et}(e_1, e_2,\ldots,e_n/E^*, i, j) \Rightarrow \xi_{E,xy,et}(e_1,i, j) \wedge \xi_{E,xy,et}(e_2,i, j) \wedge\ldots\wedge \xi_{E,xy,et}(e_n,i, j)$.

6.7 Grouping Class

Members of a grouping class are collections of members of some other class. A grouping is similar to a *part-of* relationship [8]. If a grouping entity class consists of entities from the same class, it is referred to as *homogeneous*; else *heterogeneous*. For example, a grouping class BORE_HOLE is homogeneous with respect to HOLE_INTERVAL.

6.7.1 Temporal Grouping Class

A temporal grouping class implies the lifespan of the group is pertinent for an application.

As shown in Table 16, temporal/non-temporal grouping can be associated with temporal/non-temporal base class.

	Non-temporal Grouping Class	**Temporal Grouping Class**
Non-temporal Base Class	Currently valid grouping class and base class (*1*)	N/A (*2*)
Temporal Base Class	Currently valid temporal base class with temporal grouping class (*3*)	Temporal grouping and base class (*4*)

Table 16: The semantics of temporal/non-temporal grouping/base combinations

A temporal grouping class with a temporal base class (cell *4* of Table 16) implies that the application will include queries like "What were the hole intervals of the borehole 1524 on June 28, 2001" and "For borehole 1524, what are the hole intervals it has had at different points of time." In this case, the temporal element of the composite is a subset of the intersection of the temporal elements of the components. Thus, borehole 1524 can exist when each of its hole interval exists. A temporal grouping of a non-temporal base class (cell *2* of Table 16) entities is not legitimate, as a grouping is always construed as a weak entity and cannot exist when its base entities do not exist. A non-temporal grouping of a temporal base class (cell *3* of Table 16) means applications will only include current queries on the grouping, e.g., "What are the hole intervals of the borehole 1524 now." In this case, the base class entities that participate in a grouping must be currently legitimate because a non-temporal grouping is assumed to be legitimate *now*. In other words, $\forall e_1,\ldots,e_n/E^G$, $\varphi_{E1,et}(e_1, UC) \wedge \varphi_{E2,et}(e_2, UC) \wedge\ldots\wedge \varphi_{En,et}(e_n, UC)$.

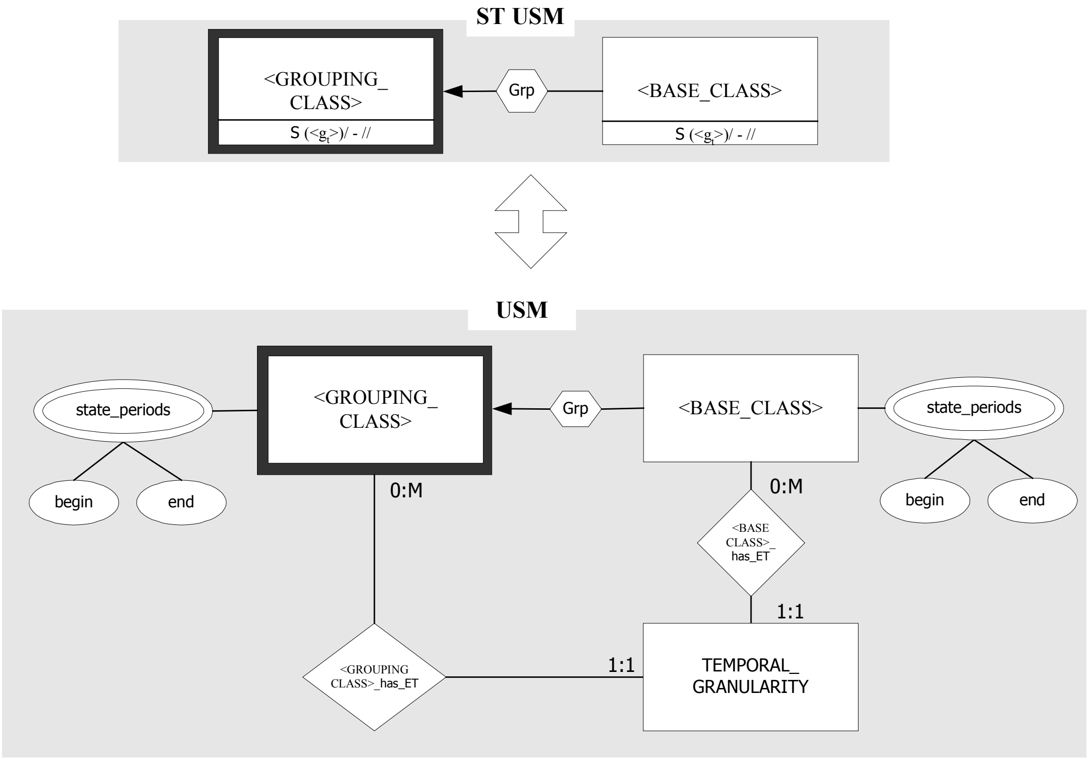


Figure 23: Temporal grouping class in ST USM and its semantics in USM

Figure 23 shows a temporal grouping class with a temporal base class. The temporal element of the ⟨GROUPING_CLASS⟩ is a subset of the temporal element of the ⟨BASE_CLASS⟩ members.

$$\forall\, e_1,\ldots,e_n/E^{G},\ \varphi_{E^{G},et}(e_1,\ldots,\ e_n/E^{G},\ i) \Rightarrow \varphi_{E1,et}(e_1,i) \wedge\ldots\wedge \varphi_{En,et}(e_n,i)$$

Constraint 6.7.1: The temporal element of a grouping is a subset of the temporal element of the intersection of the temporal elements of the participating base members. In other words, the begin index of a grouping is greater than or equal to the largest begin index of the participating base members and the end index of the grouping is less than or equal to the smallest end index of the participating base members.

$$\forall\, e_1,\ldots,\ e_n/E^{G} \in \langle\text{GROUPING\_CLASS}\rangle,\ \forall\, cp \in (e_1,\ldots,\ e_n/E^{G}).\text{state\_periods},\ \exists\, e_1,\ e_2,\ldots,\ e_n \in \langle\text{BASE\_CLASS}\rangle,$$
$$\exists\, cb_1 \in e_1.\text{state\_periods},\ldots,\ \exists\, cb_n \in e_n.\text{state\_periods},$$
$$cb_1.\text{begin} \le \ldots \le cb_n.\text{begin} \le cp.\text{begin} \le cp.\text{end} \le cb_1.\text{end} \le \ldots \le cb_n.\text{end}$$

$$\forall\, e_1,\ldots,\ e_n/E^{G} \in \langle\text{GROUPING\_CLASS}\rangle,\ \forall\, j,\ \exists\, e_1,\ldots,\ e_n \in \langle\text{BASE\_CLASS}\rangle,\ \exists\, i_1,\ldots,\ i_m$$
$$e_1(\text{state\_periods}_{i1}.\text{begin}) \le \ldots \le e_n(\text{state\_periods}_{im}.\text{begin}) \le e_1,\ \ldots,\ e_n/E^{G}(\text{state\_periods}_j.\text{begin}) <$$
$$e_1,\ldots,\ e_n/E^{G}(\text{state\_periods}_j.\text{end}) \le e_1(\text{state\_periods}_{i1}.\text{end}) \le \ldots \le e_n(\text{state\_periods}_{im}.\text{end})$$

6.7.2 Spatial Grouping Class

A spatial grouping implies that the geometry needs to be associated with the group.

As shown in Table 17, a spatial/non-spatial grouping can be associated with spatial/non-spatial components.

	Non-spatial Grouping	**Spatial Grouping**
Non-spatial Base Class	Traditional semantics (*1*)	N/A (*2*)
Spatial Base Class	Non-spatial grouping with spatial component (*3*)	Spatial grouping and component (*4*)

Table 17: The semantics of spatial/non-spatial grouping/base class combinations

Similar to a temporal grouping class a spatial grouping class with a non-spatial base class is not legitimate because a grouping class is inherently a weak class, i.e., the existence of a group depends on the existence of its components. If the components do not have an associated geometry, the group cannot have it too (cell *2* of Table 17). If the both grouping and the base class are specified as spatial, the spatial projection of grouping is a subset of the spatial project of its components. For example, a county may be perceived to be a grouping of river segments, dams and cities.

$$\forall e_1,\ e_2,\ldots,e_n/E^{G},\ \psi_{E^{G},xy}(e_1,\ e_2,\ldots,e_n/E^{*},\ i) \Rightarrow \psi_{E,xy}(e_1,i) \vee \psi_{E,xy}(e_2,i) \vee\ldots\vee \psi_{E,xy}(e_n,i)$$

6.7.3 Time-Varying Spatial Grouping Class

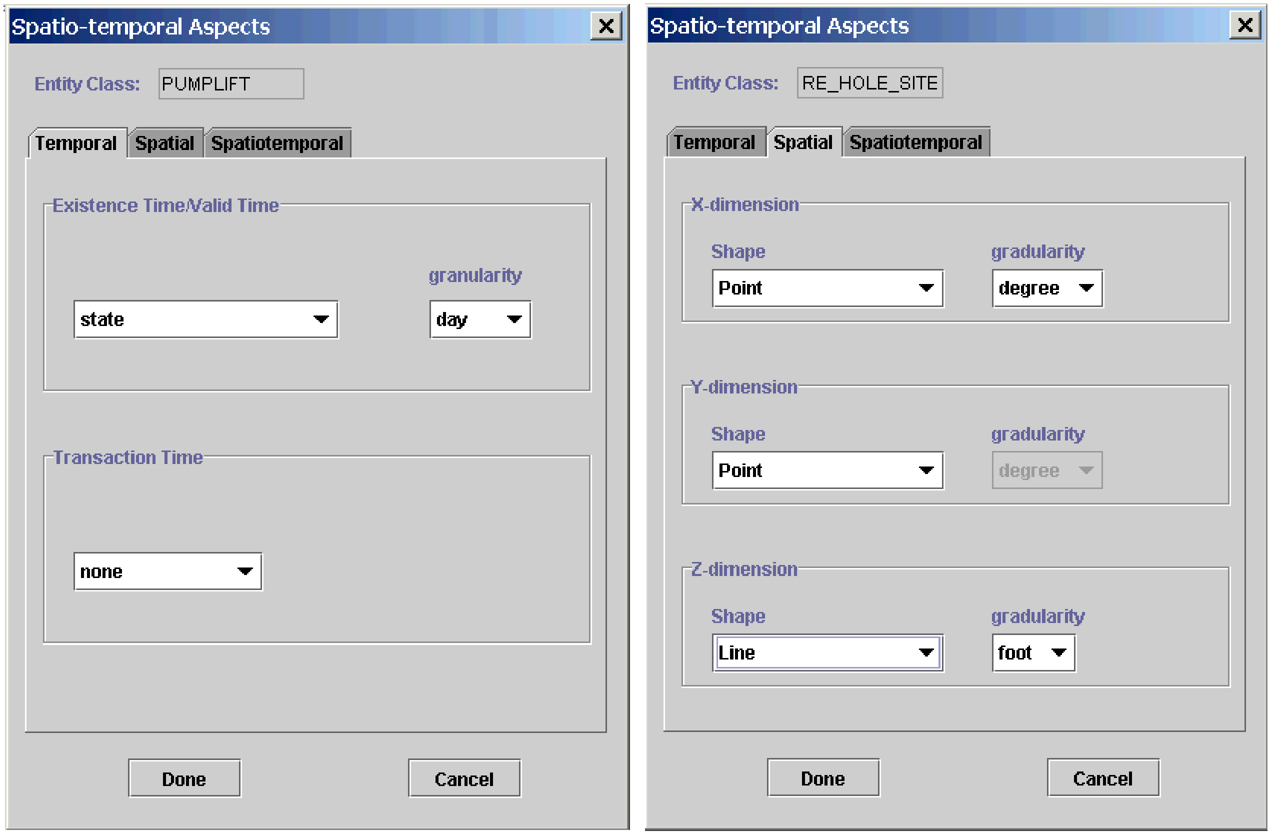
The semantics of a time-varying spatial grouping class are similar to those of a time-varying spatial composite class.

7 A Spatio-temporal Application: Reprise

In this section, we annotate the schema for the hydrogeologic application described in Section 2. We demonstrate how our annotation-based approach is practical and straightforward to implement. We briefly describe how we are using annotations to develop a prototype spatio-temporal design environment [61].

Based on the temporal and spatial requirements described in Section 2, the data analyst captures the spatio-temporal requirements of the user using annotations. At this time the data analyst asks the application users questions like: Do you want to store the history or only the current value of this fact? Do you want to capture the history of facts (valid time) or sequence of updates (transaction time), or both? What is the associated temporal granularity? Does the fact need to be modeled as an event or a state? Is it important to store the geographical reference for the objects? What is the shape of the objects? What is the associated spatial granularity? Can the spatial shape/position for these objects change over time? Accordingly, the data analyst annotates the schema shown in Figure 2 resulting in the annotated schema (or ST USM schema) shown in Figure 25.

We are developing a spatio-temporal design environment where we have implemented annotations via pop-up boxes. Once the data analyst along with the user has developed "what" is important for the database application, the analyst enters the "when" and "where" requirements using the pop-up box (Figure 24). Based on the information entered in the pop-up box the schema is automatically annotated (like Figure 25). For example, the pop-up boxes in Figure 24 will result in an annotation phrases of "S(day)/-//" and "//P(deg)/P(deg)/L(ft)".



Entity Class: PUMPLIFT

Existence Time/Valid Time: state; granularity: day

Transaction Time: none

Entity Class: RE_HOLE_SITE

X-dimension: Shape: Point; gradularity: degree

Y-dimension: Shape: Point; gradularity: degree

Z-dimension: Shape: Line; gradularity: foot

Figure 24: Pop-up boxes for annotating the schema

We continue with the schema shown in Figure 2 where "what" is important for the database application was specified. Figure 25 captures the spatio-temporal requirements using annotations. For example, SPRING needs to be represented as a region with horizontal spatial granularity of degree; accordingly, the annotation phrase for entity class

SPRING is "//R(deg)/R(deg)/-". The attribute test (of entity class BORE_HOLE_SITE) is a temporal attribute represented as an event with temporal granularity of day. A borehole may have different lithology at different depths. LITHOLOGY is a non-spatial entity while BORE_HOLE is a spatial entity. The occurs_in relationship is spatial, associating LITHOLOGY to different depths of BORE_HOLE; that is why the annotation for occurs_in is "//-/-/L(ft)".

Notice that all the cardinality constraints in Figure 25 are sequenced. We had mentioned spatiality "creeping in" the "what" schema shown in Figure 2.

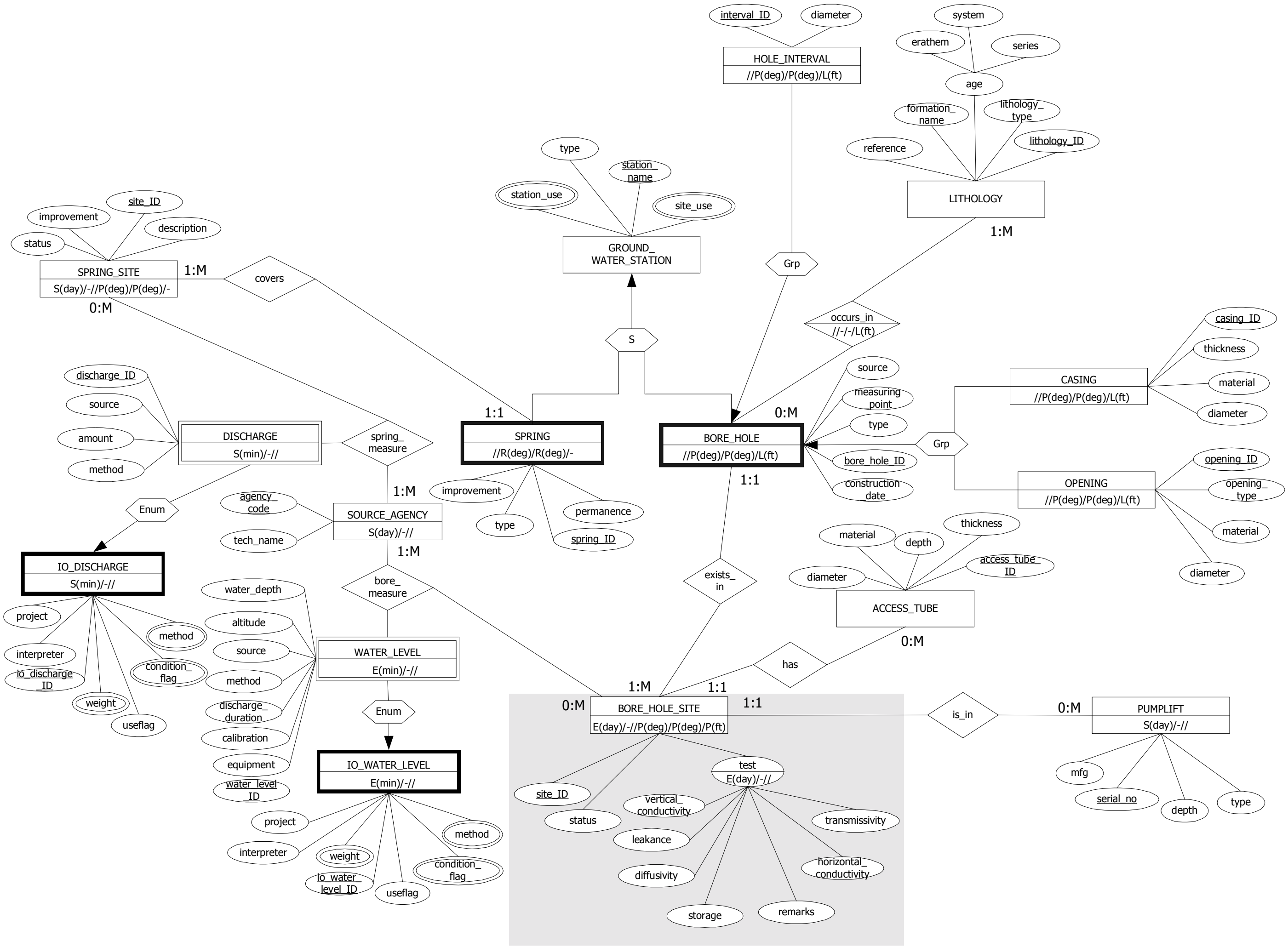


Figure 25: An annotated schema (ST USM) for the ground-water flow model

For example in Figure 2, a borehole is associated with at least one and a maximum of many (0:M) lithology (over its depth), implies a non-sequenced constraint. Also notice the same cardinality constraint in Figure 25 has been changed to 1:1 implying that each point of space along the depth of a borehole is associated with exactly one lithology. The non-sequenced constraint (0:M) is not shown in the schema. Thus, while annotating the schema, we recommend that all the constraints be carefully revisited to ensure that spatiality or temporality had not "crept in" the schema while specifying "what". If it has "crept in", the schema is simplified while it is annotated. While we show only the sequenced constraints explicitly in the schema, we have provided semantics related to non-sequenced constraints; these constraints are not shown only because it would make the schema over-crowded and difficult to understand.

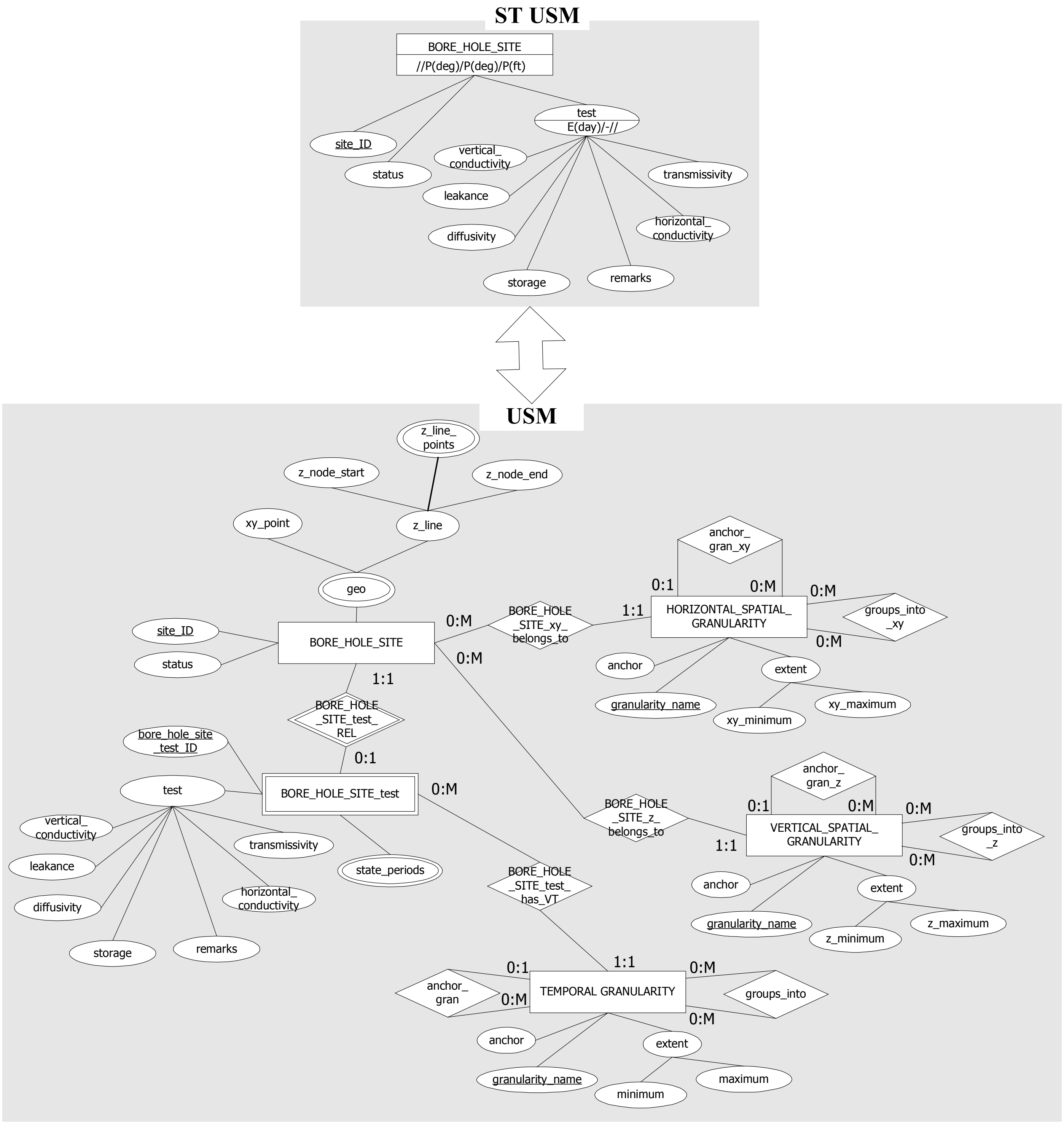


Figure 26: ST USM schema and its semantics using translated USM schema

Since a translated schema for the complete ST USM schema in Figure 25 would be very involved, we take a small portion of the ST USM schema (grayed part in Figure 25) and show the spatio-temporal semantics using a translated USM schema in Figure 26. As may be evident, the semantics of the annotated schema are quite involved. In our spatio-temporal conceptual design methodology, the annotated schemas are envisioned to be employed for capturing spatio-temporal requirements of the users, validating their requirements; the annotated schema is then used as a communication vehicle while capturing the users' requirements. On the other hand, the translated USM schema is to be used for the subsequent development of a logical schema; it explicitly shows the spatio-temporal semantics in terms of the abstractions of a conventional conceptual model. Below we list the relevant constraints associated with the translated USM schema (in Figure 26); these constraints are implied by the ST USM schema in Figure 25.

Constraint 1: cf. 6.1.3
$\forall\, e \in S(\text{TEMPORAL\_GRANULARITY}),\ e(\text{extent.minimum}) < e(\text{extent.maximum})$

Constraint 2: cf. 6.1.4
$\forall\, e \in S(\text{TEMPORAL\_GRANULARITY}),\ \neg\ \text{has}(e.\text{anchor\_gran}) \Rightarrow\ \neg\ (\exists\, e_2 \in \text{TEMPORAL\_GRANULARITY} \wedge e \neq e_2 \wedge \neg\ \text{has}(e_2.\text{anchor\_gran}))$

Constraint 3: cf. 6.1.5
$\forall\, e \in S(\text{TEMPORAL\_GRANULARITY}),\ \neg\ \text{has}(e.\text{anchor\_gran}) \Rightarrow \neg\ \text{coarser-than}(e.\text{groups\_into})$

Constraint 4: cf. 6.2.1
$\forall\, a \in S(\text{BORE\_HOLE\_SITE\_test}),\ a.\text{BORE\_HOLE\_SITE\_test\_has\_VT.TEMPORAL\_GRANULARITY}(\text{granularity\_name}) = \langle g_{vt}\rangle$

Constraint 5: cf. 6.2.4
$\forall\, a \in S(\text{BORE\_HOLE\_SITE\_test}),\ \forall\, i,$
$a.\text{BORE\_HOLE\_SITE\_test\_has\_VT.TEMPORAL\_GRANULARITY (extent.minimum)} \leq\ a(\text{state\_periods}_i\text{.begin}) <$
$a(\text{state\_periods}_i\text{.end}) \leq\ a.\text{BORE\_HOLE\_SITE\_test\_has\_VT.TEMPORAL\_GRANULARITY (extent.maximum)}$

Constraint 6: cf. 6.2.5
$\forall\, j,\ \forall\, e \in\ S(\text{BORE\_HOLE\_SITE}),\ \exists\, i,$
$e(\text{state\_periods}_i\text{.begin}) \leq\ e.\text{BORE\_HOLE\_SITE\_test\_REL.BORE\_HOLE\_SITE\_test (state\_periods}_j\text{.begin}) <$
$e.\text{BORE\_HOLE\_SITE\_test\_REL.BORE\_HOLE\_SITE\_test (state\_periods}_j\text{.end}) \leq e(\text{state\_periods}_i\text{.end})$

Constraint 7: cf. 6.1.12
$\forall\, e \in\ S(\text{BORE\_HOLE\_SITE}),\ \ e.\text{BORE\_HOLE\_SITE\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (granularity\_name)} = \text{deg}$

Constraint 8: cf. 6.1.13
$\forall\, e \in\ S(\text{BORE\_HOLE\_SITE}),\ \exists i,\ e.\text{BORE\_HOLE\_SITE\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (extent.xy\_minimum)} \leq$
$e(\text{geo}_i\text{.xy\_point}) \leq e.\text{BORE\_HOLE\_SITE\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (extent.xy\_maximum)}$

Constraint 10: cf. 6.1.14
$\forall\, e \in\ S(\text{BORE\_HOLE\_SITE}),\ \forall i,\ e.\text{BORE\_HOLE\_SITE\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (extent.xy\_minimum)}$
$\leq e(\text{geo}_i\text{.xy\_point}) \leq e.\text{BORE\_HOLE\_SITE\_xy\_belongs\_to.HORIZONTAL\_SPATIAL\_GRANULARITY (extent.xy\_maximum)}$

Constraint 9: cf. 6.1.15
$\forall\, e \in\ S(\text{HORIZONTAL\_SPATIAL\_GRANULARITY}),\ \ e(\text{extent.xy\_minimum}) < e(\text{extent.xy\_maximum})$

Constraint 11: cf. 6.1.16
$\forall\, e \in\ S(\text{SPATIAL\_GRANULARITY}),\ \neg\ \text{has}(e.\text{anchor\_gran\_xy}) \Rightarrow \neg\ (\exists\, e_2 \in\ \text{SPATIAL\_GRANULARITY} \wedge e_2 \neq e_2\ \wedge \neg\text{has}(e_2.\text{anchor\_gran\_xy})$

Constraint 12: cf. 6.1.17
$\forall\, e \in\ S(\text{SPATIAL\_GRANULARITY}),\ \neg\ \text{has}(e.\text{anchor\_gran\_xy}) \Rightarrow \neg\ \text{coarser-than}(e.\text{groups\_into\_xy})$

Constraint 13: cf. 6.1.18
$\forall\, e \in\ S(\text{BORE\_HOLE\_SITE}),\ \forall\, i,\ e(\text{geo}_i\text{.xy\_point}) \in\ \mathit{point}$

Constraint 14: cf. 6.1.19
$\forall\, e \in\ S(\text{BORE\_HOLE\_SITE}),\ \forall\, i,\ e(\text{geo}_i\text{.xy\_point}) \in\ \mathit{point}\ \ \wedge e(\text{geo}_i\text{.z\_line}) \in\ \mathit{line}$

Constraint 15: cf. 6.1.20

$$\forall\, e \in S(\text{BORE\_HOLE\_SITE}),\ \forall i,\ (\exists j,\ e(\text{geo}_i.\text{z\_line.z\_node\_start}) = e(\text{geo}_j.\text{xy\_point}) \wedge e(\text{geo}_i.\text{z\_line.z\_node\_end}) = e(\text{geo}_j.\text{xy\_point})) \wedge$$
$$(\forall\, k,\ e(\text{geo}_i.\text{z\_line.z\_node\_points}_k) = e(\text{geo}_j.\text{xy\_point}))$$

As shown by this example, a few straightforward annotations capture involved underlying spatio-temporal data semantics of the application.

8 Mapping Rules

Mapping rules provide correspondences between conceptual and logical model constructs, and are applied in logical design. The mappings to a logical schema depend on the type of logical model. Standard SQL (Structured Query Language), the basis for a relational schema, does not include time support except for user-defined time. As a result, over 50 temporal query languages have been proposed [32]; most are a result of extending SQL for the temporal domain. Change proposals to SQL3 [77, 78] include temporal semantics. On the other hand, Open GIS Consortium has proposed extensions to SQL that supports simple geospatial collections, referred to as SQL/OGIS [48]. For each of these languages, mapping rules may be developed to convert an ST USM schema to a logical schema in that language. Although details related to mapping rules are outside the scope of this paper, we provide an overview and give an example of these rules to show how the captured semantics can be embedded into the database.

As described earlier in this paper, our spatio-temporal conceptual design methodology first focuses on the non-spatial and non-temporal aspects of the application; the resulting schema is referred to as the *Core USM Schema*. Next the Core USM schema is augmented via annotations with spatio-temporal aspects resulting in the *ST-USM Schema*. There are three kinds of mapping rules; these depend on the logical model used and the path chosen to reach the logical schema from the conceptual schema. We term these mapping approaches as *conceptual translation*, *logical translation* and *direct ST translation* and differentiate them in Figure 27. The conceptual translation and the logical translation assume a logical model that supports some spatial semantics and no temporal semantics (e.g., Oracle Spatial [50, 51] and Informix Datablades [67]); the direct ST translation assumes a spatio-temporal logical model.

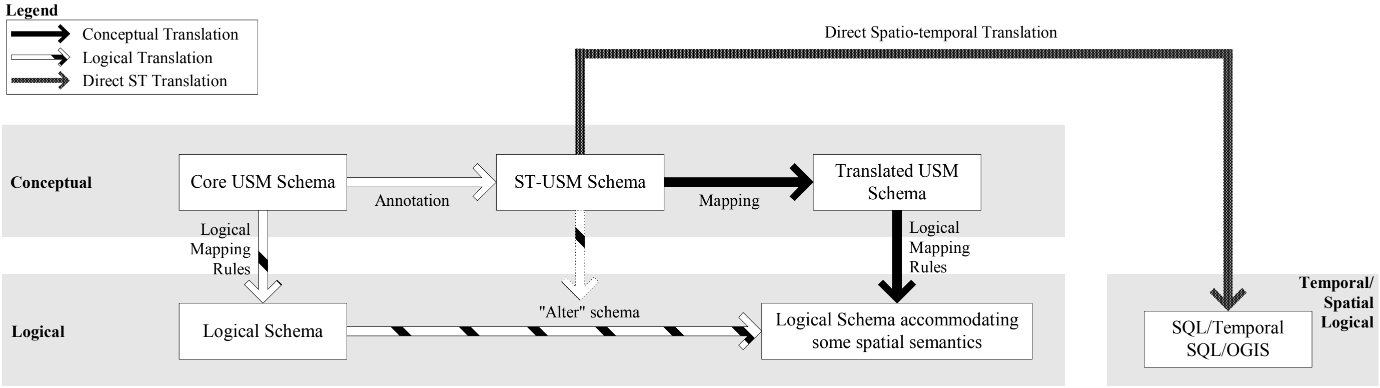


Figure 27: Mappings from Conceptual to Logical Schema

In the conceptual translation, the spatio-temporal conceptual schema (i.e., ST-USM schema) is mapped to a *Translated USM Schema* with explicit representation of the temporal and spatial semantics. Section 6 described the temporal and spatial semantics via these translation rules. This translated USM schema is next mapped to the logical schema that accommodates some spatial semantics. This mapping is an extension of the standard USM to a relational mapping.

In the logical translation, a core conceptual schema is mapped to a conventional logical schema. Next, the conventional logical schema is "alter"ed based on the annotations in a (supposed) ST-USM schema. While both (i) and (ii) require the ST USM schema, in (i) the translation (i.e., explicating the spatial and temporal semantics) is at the conceptual level, while in (ii) this translation is at the logical level.

In the direct ST translation, a spatio-temporal conceptual schema is mapped to a spatio-temporal logical schema. None of the existing DBMS products yet support a spatio-temporal logical model. So, we do not discuss this case any further, other than to note that this mapping is an extension of a mapping previously described from ER to SQL/Temporal [75].

Using the conceptual translation approach, we continue with the example shown in Figure 26. We describe the DDL (Data Definition Language) statements for creating a few of the corresponding tables using Oracle 8i.

DDL 1: The BORE_HOLE_SITE table corresponds to the entity class of the same name. In this table, geometry (geo) is defined using a pre-defined object type, SDO_GEOMETRY. SDO_GEOMETRY object is composed of: (i) SDO_GTYPE (e.g., point, line string, polygon); (ii) SDO_SRID that is used to store the spatial projection; (iii) SDO_POINT_TYPE that optimizes space for storing points; (iv) SDO_ELEM_INFO_ARRAY is a triplet that define the first element in SDO_ORDINATE, element type and its interpretation (e.g., straight line and circular arc); (v) SDO_ORDINATE, an array, which contains ordinates that make up geometry. For a multi-line string in 3D (i.e., borehole), SDO_GTYPE (defines the type of geometry stored in the object geo) is specified by a code of 3006.

```
CREATE TABLE BORE_HOLE_SITE (
site_ID VARCHAR2(10) PRIMARY KEY,
status VARCHAR2(100),
geo MDSYS.SDO_GEOMETRY);
```

An example tuple in this table would be:

```
INSERT INTO BORE_HOLE_SITE VALUES ('1000', 'The site was flowing recently',
          MDSYS.SDO_GEOMETRY (3006, NULL, NULL, MDSYS.SDO_ELEM_INFO_ARRAY (1,2,1),
          MDSYS.SDO_ORDINATE_ARRAY (20,30,40)));
```

DDL 2: This table includes a surrogate key (test_ID) in the DDL statement out of pragmatism.

```
CREATE TABLE BORE_HOLE_SITE_test (
test_ID number(10) PRIMARY KEY,
site_ID CONSTRAINT bhs_test_fk1 REFERENCES BORE_HOLE_SITE ON DELETE CASCADE,
vertical_conductivity NUMBER(8,4),
leakance NUMBER(8,4),
diffusivity NUMBER(8,4),
storage NUMBER(8,4),
horizontal_conductivity NUMBER(8,4),
transmissivity NUMBER(8,4),
remarks VARCHAR2(1000));
```

The constraint “ON DELETE CASCADE” implies that if a tuple is deleted from the referenced table (e.g., BORE_HOLE_SITE), all the corresponding tuples in the referencing table are deleted (e.g., BORE_HOLE_SITE_test); this constraint is added because BORE_HOLE_SITE_test is a weak entity CLASS.

DDL 3: The multi-valued attribute state_periods of BORE_HOLE_SITE_test is rendered as a separate table. This is the usual translation rule for conventional conceptual diagrams. In this table “ON DELETE CASCADE” constraint is chained to BORE_HOLE_SITE; in other words, if a tuple is deleted from BORE_HOLE_SITE, all the corresponding tuples in BORE_HOLE_SITE_test and BHS_state_periods will be deleted.

```
CREATE TABLE BHS_state_periods (
test_ID NUMBER(10) CONSTRAINT bhs_sp REFERENCES BORE_HOLE_SITE_test ON DELETE CASCADE,
begin DATE,
end DATE,
CONSTRAINT bhs_test_sp2 PRIMARY KEY (test_ID, begin, end));
```

Assertion: In the schema, we have a sequenced constraint for BORE_HOLE_SITE_test (the test partial key): there can only be one value of, e.g., leakance, at any point in time. This can be enforced with the following assertion [75].

```
CREATE ASSERTION bhs_test_assertion
CHECK (NOT EXISTS (SELECT *
                FROM BORE_HOLE_SITE_test AS I1
                WHERE 1 < (SELECT COUNT(leakance)
                FROM BORE_HOLE_SITE_test AS I2
                WHERE I1.leakance = I2.leakance
                        AND I1.BEGIN < I2.END
                        AND I2.BEGIN < I1.END))
        AND NOT EXISTS (SELECT *
                FROM BORE_HOLE_SITE_test AS I
                WHERE I.SSN IS NULL)
);
```

This assertion can be implemented using a trigger in Oracle [75].

```
CREATE OR REPLACE TRIGGER bhs_test_TRIGGER
AFTER INSERT OR UPDATE ON BORE_HOLE_SITE_test
DECLARE
        valid INTEGER;
BEGIN
        SELECT 1
        INTO valid
        FROM DUAL
        WHERE NOT EXISTS (SELECT *
                FROM BORE_HOLE_SITE_test  I1, BORE_HOLE_SITE_test I2
                WHERE I1.leakance = I2.leakance
                        AND I1.BEGIN < I2.END
                        AND I2.BEGIN < I1.END
                        AND I1.rowed <> I2.rowed )
                AND NOT EXISTS (SELECT *
                        FROM BORE_HOLE_SITE_test AS I
                        WHERE I.leakance IS NULL);
EXCEPTION
        WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR (-20001, ‘leakance is a sequenced key’);
END;
```

In summary, for the fragment shown in Figure 26, a total of nine tables (BORE_HOLE_SITE, BORE_HOLE_SITE_test, groups_into_xy, groups_into_z, BORE_HOLE_SITE_state_periods, TEMPORAL_GRANULARITY, groups_into, VERTICAL_SPATIAL_GRANULARITY, HORIZONTAL_SPATIAL_GRANULARITY) and nineteen assertions are needed. This emphasizes the clarity that the annotations in Figure 25 provide.

9 Related Work

Over the last two decades, many temporal conceptual models have been proposed, e.g., TERM (Temporal Entity Relationship Model) [38, 39], RAKE (Relationships, Attributes, Keys and Entities) [21], TEER (Temporal EER) [17, 18], TERC+ [89] and TimeER Model [26]. The reader is referred to an excellent survey of extant temporal conceptual models by Gregersen and Jensen [26]. In the last couple of years, two major spatio-temporal conceptual models have been proposed: MADS (Modeling of Application Data with Spatiotemporal) [52-54] and GeoER [29], which was extended to STER (Spatiotemporal Entity Relation) [82]. The other spatial models, e.g., GEO Object Oriented Analysis (GEOOOA) [40] and Geographic Data Management with Object-Oriented Techniques (GODOT) [27, 28], focus on the semantics of geometry associated with objects and are really not conceptual models.

STER [82] applies spatio-temporal concepts to the modeling constructs of the ER model [11]. Spatial properties of objects are captured by *fields*, a function from geometric figure to a domain of descriptive attribute values. STER provides formalism to model existence time and transaction time associated with an entity type, and valid time and transaction time related to an attribute and a relationship. In MADS [52-54], spatiality can be associated with object types, attributes, relationships and aggregation.

Although the existing body of literature presents some insight into spatio-temporal conceptual modeling, there are many unresolved issues. The key issue in defining a formalism at conceptual level is that it that it should be straightforward so that data analysts and users can understand it, and be formal so that it can subsequently be translated into an implementable logical schema. As it evident, these two requirements of simplicity and comprehensiveness are conflicting. Both the prior proposals focus on the first aspect; in other words, the extant models informally focus on syntax than on the associated semantics. If the semantics are not clearly defined, the querying capabilities of the given schema are ambiguous. Thus conceptual design does not achieve its objective, i.e., capture the requirements of the users.

We instead define an annotation-based approach in which annotations capture the spatio-temporal semantics in a straightforward manner. We provide the spatio-temporal semantics formally using the translated USM schema and constraints specified in first order logic. For example, using the constraints we specify what exactly is a temporal element (constraints 6.1.6 and 6.1.7). These constraints can be translated to assertions on the database that can help prevent incorrect data entry. Thus, with our annotation-based approach, we have achieved both simplicity and comprehensiveness in spatio-temporal conceptual modeling.

STER briefly mentions spatial granularities: "positions may be viewed at different granularities." Similarly, in MADS, *dbspace* and *dbtime* defines the area and time frame in the database. *Spacezones* and *timezones* define the spatial and temporal elements in the dbspace and dbtime, respectively. Dbspace and dbtime is similar to our spatial and temporal granularities. However, neither of these models describes any mechanism to capture the associated semantics of temporal and spatial granularity. Our spatio-temporal model explicitly includes spatial and temporal granularities.

STER [82] defines spatial attributes as "derived properties" from space. Thus, spatial attributes are "properties of the embedding space and indirectly become properties of spatial objects via their position in space." The authors justify this representation with an example of a "soil type" which exists whether or not the entity "landparcel" exists. However, this representation contradicts the definition of an attribute—attributes are properties *of* an object. An alternative, which will be expressed in ST USM, is to model "soil type" as a non-spatial entity class having a spatial relationship with a spatial entity class "landparcel." Then "soil type" is interpreted to exist on its own irrespective of the "landparcel"; spatial relationship between spatial entities implies that there can be different regions within a landparcel with different soil types. This is consistent with the users' requirements, which would include queries about soil type *of* a landparcel. As exemplified by this example, our proposed model does not change the semantics of any conventional model construct.

10 Conclusion

Thrift [86] posits time and space as resources. Guptill [36] argues that the model of space and time needs to be reconciled with the conceptual models developed in the database community. Lee and Isdale [54] argue that there is a need for a special purpose conceptual model that is suitable for GIS applications. In this paper, we propose a spatio-temporal conceptual design approach that is backward compatible with a non-spatio-temporal conceptual model and that takes into account how humans store and use spatial data. Our proposed approach does not change the syntax of the conventional conceptual model, and is straightforward to use from the perspective of users, data analysts and CASE tool vendors.

We are developing a spatio-temporal conceptual design environment, DISTIL [61], which aids in capturing the users' spatiotemporal requirements resulting in a conceptual schema (ST USM); DISTIL will also translate the conceptual schema to a logical schema. DISTIL will automate consistency checking during conceptual design. For example, when the data analyst tries specifying a relationship as temporal when the associated entity classes are non-temporal, DISTIL will identify this inconsistency. We are also exploring how ST USM can be used as a canonical model for information integration of distributed spatio-temporal data. In the future, we plan to extend annotations to incorporate schema versioning [63, 64]. We also plan to provide a mechanism for modeling spatio-temporal constraints in a conceptual schema, such as the lifetime constraints discussed in section 6.2.1.2 and the topological constraints discussed in section 6.3.2. Finally, we also plan to incorporate the telic/atelic distinction that has been shown to be useful in logical temporal models [81], extending this distinction to the spatial dimension.

Acknowledgements

This work was supported in part by NASA grant 314401, NSF grants EIA-0080123 and IIS-0100436, and by a grant from the Boeing Corporation.

References

[1] R. Agarwal, P. De and A. P. Sinha, “Comprehending Object and Process Models: An Empirical Study,” *IEEE Transactions on Software Engineering*, Vol. 25, No. 4, pp. 541-556, 1999.

[2] J. F. Allen, “Maintaining knowledge about temporal intervals,” *Communications of the ACM*, Vol. 26, No. 11, pp. 832-843, 1983.

[3] J. R. Anderson and G. H. Bower, *Human Associative Memory*. Washington, D.C.: V. H. Winston & Sons, 1973.

[4] C. Batini, S. Ceri and S. B. Navathe, *Conceptual Database Design: An Entity-Relationship Approach*: Benjamin/Cummings Publishing Company, 1992.

[5] C. Bettini, C. E. Dyreson, W. S. Evans, R. T. Snodgrass and X. S. Wang, “A Glossary of Time Granularity Concepts,” in *Temporal Databases: Research and Practice*, O. Etzion, S. Jajodia, and S. Sripada, Eds.: Springer-Verlag, 1998, pp. 406-413.

[6] C. Bettini, S. Jajodia and S. X. Wang, *Time Granularities in Databases, Data Mining and Temporal Reasoning*. Berlin: Springer-Verlag, 2000.

[7] M. H. Böhlen, C. S. Jensen and R. T. Snodgrass, “Temporal Statement Modifiers,” *ACM Transactions on Database Systems*, Vol. 25, No. 4, pp. to appear, 2000.

[8] M. L. Brodie, “On the Development of Data Models,” in *On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases and Programming Languages*, M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, Eds.: Springer-Verlag, 1984, pp. 19-47.

[9] M. Brosey and B. Shneiderman, “Two Experimental Comparisons of Relational and Hierarchical Database Models,” *International Journal of Man-Machine Studies*, Vol. 10, No. 6, pp. 625-637, 1978.

[10] P. A. Burrough, “Are GIS Data Structures Too Simple Minded?,” *Computers and Geosciences*, Vol. 18, No. 4, pp. 395-400, 1992.

[11] P. P. Chen, “The Entity-Relationship Model - Toward a Unified View of Data,” *ACM Transactions of Database Systems*, Vol. 1, No. 1, pp. 9-36, 1976.

[12] F. A. D'Agnese, C. C. Faunt, A. K. Turner and M. C. Hill, “Hydrogeologic evaluation and numerical simulation of the Death Valley Regional ground-water flow system, Nevada and California,” U.S. Geological Survey Water Resources 96-4300, 1997.

[13] B. David, M. V. D. Herrewegen and F. Salge, “Conceptual Models for Geometry and Quality of Geographic Information,” in *Geographic Objects With Indeterminate Boundaries*, P. A. Burrough and A. Frank, Eds.: Taylor & Francis, 1996, pp. 352.

[14] C. E. Dyreson, W. S. Evans, H. Lin and R. T. Snodgrass, “Efficiently Supporting Temporal Granularities,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 4, pp. 568-587, 2000.

[15] C. E. Dyreson and R. T. Snodgrass, “Supporting Valid-Time Indeterminacy,” *ACM Transactions on Database Systems*, Vol. 23, No. 1, pp. 1-57, 1998.

[16] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, Second ed. Redwood City, CA: Benjamin/ Cummings Publishing Co., 1994.

[17] R. Elmasri, G. Wuu and V. Kouramajian, “A Temporal Model and Query Language for EER Databases,” in *Temporal Databases: Theory, Design and Implementation*, A. Tansel, Ed. Benjamin/ Cummings,, 1993, pp. 212-229.

[18] R. Elmasri and G. T. J. Wuu, “The Temporal Model and Query Language for ER databases,” in Proceeding of the 6th International Conference on Data Engineering, pp. 76-83, 1990.

[19] M. Erwig, R. H. Güting, M. Schneider and M. Vazirgiannis, “Abstract and Discrete Modeling of Spatio-temporal Data Types,” in Proceedings of the 6th International Symposium on Advances in Geographic Information Systems, Washington, United States, pp. 131-136, 1998.

[20] B. Falcidieno, C. Pienovi and M. Spagnuolo, “Descriptive Modeling and Prescriptive Modeling in Spatial Data Handling,” in Proceedings of the International Conference on GIS - From Space to Territory: Theories and Methods of Spatiotemporal Reasoning in Geographic Space, Pisa, Italy, pp. 122-135, 1992.

[21] S. Ferg, “Modeling the Time Dimension in an Entity-Relationship Diagram,” in 4th International Conference on the Entity-Relationship Approach, pp. 280-286, 1985.

[22] A. U. Frank, “Spatial Concepts, Geometric Data Models, and Geometric Data Structures,” *Computers and Geosciences*, Vol. 18, No. 4, pp. 409-417, 1992.

[23] A. U. Frank and D. M. Mark, *Language Issues in GIS*, vol. 2: Zurich, Longman Scientific Technical, 1991.

[24] M. F. Goodchild, “Geographic Information Systems,” in *Ten Geographic Ideas that Changed the World*, S. Hanson, Ed. New Brunswick, New Jersey: Rutgers University Press, 1997.

[25] H. Gregersen and C. Jensen, “Conceptual Modeling of Time-Varying Information,” TIMECENTER Technical Report TR-35, September 10 1998.

[26] H. Gregersen and C. S. Jensen, “Temporal Entity-Relationship Models-A Survey,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 3, pp. 464-497, 1999.

[27] O. Gunther and J. Lamberts, “Object-Oriented Techniques for the Management of Geographic and Environmental Data,” *The Computer Journal*, Vol. 37, No. 1, pp. 16-25, 1994.

[28] O. Gunther and W. F. Riekert, “The Design of GODOT: An Object-Oriented Geographical Information System,” *IEEE Data Engineering Bulletin*, Vol. 16, No. 3, pp. 1993.

[29] T. Hadzilacos and N. Tryfona, “An Extended Entity-Relationship Model for Geographic Applications,” *SIGMOD Record*, Vol. 26, No. 3, pp. 24-29, 1997.

[30] B. C. Hardgrave and P. D. Nikunj, “Comparing Object-Oriented and Extended-Entity-Relationship Data Models. Summer 1995, pp. 15-21,” *Journal of Database Management*, Vol. 5, No. 2, pp. 15-21, 1995.

[31] R. Hull and R. King, “Semantic Database Modeling: Survey, Applications, and Research Issues,” *ACM Computing Surveys*, 210-260, 1987.

[32] C. S. Jensen, C. E. Dyreson, M. Bohlen, J. Clifford, R. Elmasri, S. K. Gadia, F. Grandi, P. Hayes, S. Jajodia, W. Kafer, N. Kline, N. Lorentzos, Y. Mitsopoulos, A. Montanari, D. Nonen, E. Peresi, B. Pernici, J. F. Roddick, N. L. Sarda, M. R. Scalas, A. Segev, R. T. Snodgrass, M. D. Soo, A. Tansel, R. Tiberio and G. Wiederhold, “A Consensus Glossary of Temporal Database Concepts-February 1998 Version,” in *Temporal Databases: Research and Practice*, O. Etzion, S. Jajodia, and S. Sripada, Eds.: Springer-Verlag, 1998.

[33] C. S. Jensen and R. T. Snodgrass, “Semantics of Time-Varying Information,” *Information Systems*, Vol. 21, No. 4, pp. 311-352, 1996.

[34] C. S. Jensen and R. T. Snodgrass, “Temporal Data Management,” *Transactions of Knowledge and Data Engineering*, Vol. 11, No. 1, pp. 36-44, 1999.

[35] V. Khatri, S. Ram and R. T. Snodgrass, “Supporting User-defined Granularities and Indeterminacy in a Spatiotemporal Conceptual Model,” *Annals of Mathematics and Artificial Intelligence*, 36, forthcoming.

[36] Y.-G. Kim and S. T. March, “Comparing Data Modeling Formalisms,” *Communications of the ACM*, Vol. 38, No. 6, pp. 103-115, 1995.

[37] L. Kleinrock, *Queueing SystemsTheory*. New York: John Wiley & Sons, 1975.

[38] M. R. Klopprogge, “TERM: An Approach to include the Time Dimension in the Entity Relationship Model,” in Proceedings Second International Conference Entity Relationship Approach, pp. 477-512, 1981.

[39] M. R. Klopprogge and P. C. Lockemann, “Modelling Information Preserving Databases: Consequences of the Concept of Time,” in 9th International Conference on Very Large Data Bases, Florence, Italy, pp. 399-416, 1983.

[40] G. Kosters, B.-U. Pagel and H.-W. Six, “Object Oriented Requirements Engineering for GIS-Applications,” in Proceedings of the International Conference on ACM Geographical Information System, pp. 1995.

[41] R. Laurini and D. Thompson, *Fundamentals of Spatial Information Systems*. London: Academic Press, 1992.

[42] Y. C. Lee and M. Isdale, “The Need for a Spatial Data Model,” in Proceedings of Canadian Conference in GIS-91, pp. 1991.

[43] J. Lipski, “On Semantic Issues Connected with Incomplete Information Databases,” *ACM Transactions of Database Systems*, Vol. 4, No. 3, pp. 262-296, 1979.

[44] D. M. Mark and A. U. Frank, “Experiential and Formal Models of Geographic Space,” *Environment and Planning*, Vol. 23, No. 1, pp. 3-24, 1996.

[45] J. L. Mennis, D. J. Peuquet and L. Qian, “A Conceptual Framework for Incorporating Cognitive Principles into Geographical Database Representation,” *International Journal of Geographic Information Science*, Vol. 14, No. 6, pp. 501-520, 2000.

[46] S. Morehouse, “The Role of Semantics in Geographic Data Modeling,” in Proceedings of the 4th International Symposium in Spatial Data Handling, Zurich, Switzerland, pp. 689-698, 1990.

[47] C. Murray, “Oracle Spatial User's Guide and Reference,” , vol. 2001 Oracle, http://technet.oracle.com/docs/products/spatial/content.html, 2000.

[48] OGIS, “OpenGIS Simple Feature Specification for SQL,” Open GIS Consortium, Inc. 99-049, May 5 1999.

[49] Oracle, “Oracle8^{TM} Time Series Cartridge,” Oracle Corporation, Redwood City, CA Release 8.0.4, A57501-01, 1997.

[50] Oracle, “Coordinate Systems User's Guide,” Oracle Inc., Release 8.1.6 April 28 2000.

[51] Oracle, “Oracle8i Documentation, Release 8.1.6,” Oracle, http://technet.oracle.com/docs/products/oracle8i/doc_index.htm, December 1999.

[52] C. Parent, S. Spaccapietra and E. Zimanyi, “Spatio-temporal conceptual models: Data structures + space + time,” in Proceedings of the 7th ACM Symposium on Advances in Geographic Information Systems, Kansas City, USA, pp. 1999.

[53] C. Parent, S. Spaccapietra and E. Zimányi, “Conceptual modeling for federated geographical information systems over the Web,” in Proceedings of International Symposium on Information Systems and Technology for Network Society, Fukuoka, Japan, pp. 173-182, 1997.

[54] C. Parent, S. Spaccapietra, E. Zimányi, P. Donini, C. Plazanet and C. Vangenot, “Modeling spatial data in the MADS conceptual model,” in Proceedings of the 8th International Symposium on Spatial Data Handling, SDH'98, Vancouver, Canada, pp. 138-150, 1998.

[55] D. Parkes and N. Thrift, *Time, spaces and places*. New York: John Wiley & Sons, 1980.

[56] J. Peckham and F. Maryanski, “Semantic Data Models,” *ACM Computing Surveys*, Vol. 20, No. 3, pp. 153-189, 1988.

[57] D. J. Peuquet, “It's about Time: A Conceptual Framework for the Representation of Temporal Dynamics in Geographic Information Systems,” *Annals of the Association of American Geographers*, Vol. 83, No. 3, pp. 441-461, 1994.

[58] D. Pfoser and N. Tryfona, “Requirements, Definitions, and Notations for Spatiotemporal Application Environments,” in Proceedings of the 6th International Symposium on Advances in Geographic Information Systems, Washington, United States, pp. 124-130, 1998.

[59] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, Fourth ed. Singapore: Mc-Graw Hill, 1997.

[60] S. Ram, “Intelligent Database Design using the Unifying Semantic Model,” *Information and Management*, Vol. 29, No. 4, pp. 191-206, 1995.

[61] S. Ram, R. T. Snodgrass, V. Khatri and Y. Hwang, “DISTIL: A Design Support Environment for Conceptual Modeling of Spatio-Temporal Requirements,” in Proceedings of the 20th International Conference on Conceptual Modeling (ER2001), Yokohama, Japan, pp. 2001.

[62] P. Rigaux, M. O. Scholl and A. Voisard, *Spatial Databases: With Application to GIS*: Morgan Kaufmann Publishers, 2001.

[63] J. F. Roddick, “A Survey of Schema Versioning Issues for Database Systems,” *Information and Software Technology*, Vol. 37, No. 7, pp. 383-393, 1995.

[64] J. F. Roddick and R. T. Snodgrass, “Schema Versioning,” in *The TSQL2 Temporal Query Language*, R. T. Snodgrass, Ed. Boston: Kluwer Academic Publishers, 1995, pp. 427-449.

[65] D. Rumelhart, P. Lindsay and D. Norman, “A process model for long-term memory,” in *Organization of Memory*, E. Turving and W. Donaldson, Eds. New York: Academic Press, 1972.

[66] E. A. Rundensteiner, A. Bic, J. P. Gilbert, and M. Yin, “Set restrictions on semantic groupings,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, No. 2, pp. 193-204, 1994.

[67] C. Scannell, D. Ashkenas, T. Day, R. Gutman, A. Guttman, J. Klebanoff, M. McDevitt, G. Morgan, C. Renck, R. Uleman, M. V. Lunteren and J. Zhang, “Informix Geodetic Datablade Module: User's Guide,” http://www.informix.com/answers/english/docs/datablade/5544.pdf, 2001.

[68] P. Shoval and M. Even-Chaime, “Data Base Schema Design: An Experimental Comparison between Normalization and Information Analysis,” *Data Base*, Vol. 18, No. 3, pp. 30-39, 1987.

[69] P. Shoval and I. Frumermann, “OO and EER Conceptual Schemas: A Comparison of User Comprehension,” *Journal of Database Management*, Vol. 5, No. 4, pp. 28-38, 1994.

[70] A. Silbershatz, H. Korth and S. Sudarshan, *Database System Concepts*, Third Edition ed: WCB/ McGraw Hill, 1997.

[71] J. Simmonds, "Informix TimeSeries Datablade Module: User's Guide," http://www.informix.com/answers/english/docs/datablade/6577.pdf, 2000.

[72] D. F. Sinton, "The Inherent Structure of Information as a Constraint to Analysis: Mapped Thematic Data as a Case Study," *Harvard Papers on GIS*, Vol. 7, 1-17, 1978.

[73] R. T. Snodgrass, "The Temporal Query Language TQuel," *ACM Transactions of Database Systems*, Vol. 12, No. 2, pp. 247-298, 1987.

[74] R. T. Snodgrass, "Temporal Object-Oriented Databases: A Critical Comparison," in *Modern Database Systems: The Object Model, Interoperability and Beyond*, W. Kim, Ed. Reading, Massachusetts: Addison-Wesley Publishing Company, 1995, pp. 703.

[75] R. T. Snodgrass, *Developing Time-Oriented Database Applications in SQL*. San Francisco: Morgan Kaufmann Series in Data Management Systems, 1999.

[76] R. T. Snodgrass and I. Ahn, "Temporal Databases," *IEEE Computer*, Vol. 19, No. 9, pp. 35-42, 1986.

[77] R. T. Snodgrass, M. H. Böhlen, C. S. Jensen and A. Steiner, "Adding Transaction Time to SQL/Temporal," ISO-ANSI SQL/Temporal Change Proposal, ANSI X3H2-96-152r ISO/IEC JTC1/SC21/WG3 DBL MCI-143, 1996.

[78] R. T. Snodgrass, M. H. Böhlen, C. S. Jensen and A. Steiner, "Adding Valid Time to SQL/Temporal," ISO-ANSI SQL/Temporal Change Proposal, ANSI X3H2-96-151r ISO/IEC JTC1/SC21/WG3 DBL MCI-142, 1996.

[79] J. Star and J. Estes, *Geographic Information Systems: An Introduction*: Englewood Cliffs, Prentice Hall, 1990.

[80] B. Tauzovich, "Toward Temporal Extensions to the Entity-Relationship Model," in Proceedings of 10th International Conference Entity Relationship Approach, pp. 163-179, 1991.

[81] P. Terenziani and R. T. Snodgrass, "Reconciling Point-based and Interval-based Semantics in Temporal Relational Databases: A Proper Treatment of the Telic/Atelic Distinction," TimeCenter Technical Report TR-60 2001.

[82] N. Tryfona and C. S. Jensen, "Conceptual Data Modeling for Spatiotemporal Applications," *Geoinformatica*, Vol. 3, No. 3, pp. 245-268, 1999.

[83] V. J. Tsotras, C. S. Jensen and R. T. Snodgrass, "An Extensible Notation for Spatiotemporal Index Queries," *SIGMOD Record*, Vol. 27, No. 1, pp. 47-53, 1998.

[84] J. W. van Roessel, "Design of a Spatial Data Structure using the Relational Normal Form," *International Journal of Geographical Information Systems*, Vol. 1, No. 1, pp. 33-50, 1987.

[85] M. Wachowicz, *Object-Oriented Design for Temporal GIS*: Taylor and Francis, 1999.

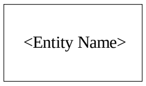
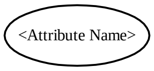
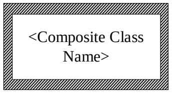
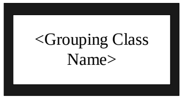
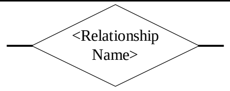
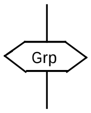
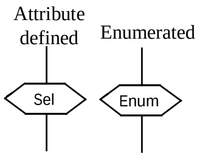
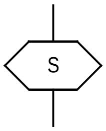
[86] M. F. Worboys, "Computation with Imprecise Geospatial Data," *Computer, Environment and Urban Systems*, Vol. 22, No. 2, pp. 85-106, 1998.

[87] M. F. Worboys, "Imprecision in Finite Resolution Spatial Data," *GeoInformatica*, Vol. 2, No. 3, pp. 257-279, 1998.

[88] E. N. Yourdon, *Modern Structured Analysis*: Prentice Hall, 1990.

[89] E. Zimanyi, C. Parent, S. Spaccapietra and A. Pirotte, "TERC+: A Temporal Conceptual Model," in Proceeding International Symposium Digital Media Information Base, pp. 1997.

Appendix A: Summary of the USM Notations

USM Constructs	Description	USM Graphical Notation
Entity Class	A collection of entities for which common characteristics are to be modeled	
Attribute	Properties of members of an entity class	
Composite Class	A set of members of some other class taken as a whole is called a composite class	
Grouping Class	A class of entities whose members are physically or logically composed of members or sets of members from some other entity classes are called grouping class	
Interaction Relationship	Associations between members of one entity class and one or more other entity classes	
Grouping Relationship	Grouping relationship results in the definition of a *group*	
Composite Relationship	Two types of composite relationships: *attribute-defined* and *enumerated*	
Generalization/ Specialization	Generalization has similar objects abstracted to form a higher order objects. Specialization is the inverse of generalization	

Appendix B: Annotation Syntax in BNF

```
⟨annotation⟩                            ::=   ϵ | ⟨temporal annotation⟩ // ⟨spatial annotation⟩
                                              | ⟨temporal annotation⟩ // ⟨spatial annotation⟩ // ⟨time-varying spatial annotation⟩

⟨temporal annotation⟩                   ::=   ϵ | ⟨valid time⟩ / ⟨transaction time⟩
⟨valid time⟩                            ::=   ⟨state⟩ (⟨g_t⟩) | ⟨indeterminate state⟩ (⟨g_t⟩) | ⟨event⟩ (⟨g_t⟩)
                                              | ⟨indeterminate event⟩(⟨g_t⟩) | -
⟨transaction time⟩                      ::=   T | -
⟨state⟩                                 ::=   S | State
⟨indeterminate state⟩                   ::=   ⟨state⟩~ | ⟨state⟩+-
⟨event⟩                                 ::=   E | Event
⟨indeterminate event⟩                   ::=   ⟨event⟩~ | ⟨event⟩+-

⟨spatial annotation⟩                    ::=   ϵ | ⟨horizontal geometry⟩ / ⟨vertical geometry⟩
⟨horizontal geometry⟩                   ::=   ⟨geometry⟩ (⟨g_xy⟩) / ⟨geometry⟩ (⟨g_xy⟩)
⟨vertical geometry⟩                     ::=   ⟨geometry⟩ (⟨g_z⟩) | -
⟨geometry⟩                              ::=   ⟨point⟩ | ⟨indeterminate point⟩ | ⟨line⟩ | ⟨indeterminate line⟩ | ⟨region⟩
                                              | ⟨indeterminate region⟩ | ⟨user defined⟩ | -
⟨point⟩                                 ::=   P | Point
⟨indeterminate point⟩                   ::=   ⟨point⟩~ | ⟨point⟩+-
⟨line⟩                                  ::=   L | Line
⟨indeterminate line⟩                    ::=   ⟨line⟩~ | ⟨line⟩+-
⟨region⟩                                ::=   R | Region
⟨indeterminate region⟩                  ::=   ⟨region⟩~ | ⟨region⟩+-

⟨time-varying spatial annotation⟩       ::=   ϵ | ⟨position varying⟩ | ⟨shape varying⟩ | ⟨position varying⟩ / ⟨shape varying⟩
⟨position varying⟩                      ::=   ⟨position⟩@⟨varying in dimension⟩
⟨shape varying⟩                         ::=   ⟨shape⟩@⟨varying in dimension⟩
⟨position⟩                              ::=   Pos | Position
⟨shape⟩                                 ::=   Sh | Shape
⟨varying in dimension⟩                  ::=   x | y | z | xy | yz | xz | xyz

⟨g_t⟩                                   ::=   ⟨day⟩ | ⟨hour⟩ | ⟨minute⟩ | ⟨second⟩ | ⟨user defined⟩
⟨day⟩                                   ::=   day
⟨hour⟩                                  ::=   hr | hour
⟨minute⟩                                ::=   min | minute
⟨second⟩                                ::=   sec | second
⟨g_sxy⟩                                 ::=   ⟨mile⟩ | ⟨dms-degree⟩ | ⟨dms-minute⟩ | ⟨foot⟩ | ⟨user defined⟩
⟨g_sz⟩                                  ::=   ⟨mile⟩ | ⟨foot⟩ | ⟨user defined⟩
⟨mile⟩                                  ::=   mile
⟨dms-degree⟩                            ::=   dms-deg | dms-degree
⟨dms-minute⟩                            ::=   dms-min | dms-minute
⟨foot⟩                                  ::=   ft | foot
```