MappingSets for Spatial Observation Data Warehouses

José R.R. Viqueira
COGRADE - CITIUS
Universidade de Santiago de Compostela, Spain
jrr.viqueira@usc.es

Sebastián Villarroya
COGRADE - CITIUS
Universidade de Santiago de Compostela, Spain
sebastian.villarroya@usc.es

David Martínez
COGRADE - CITIUS
Universidade de Santiago de Compostela, Spain
david.martinez.casas@usc.es

José A. Taboada
COGRADE - CITIUS
Universidade de Santiago de Compostela, Spain
joseangel.taboada@usc.es

ABSTRACT

The amount of time evolving spatial data that is currently being generated by automatic observation processes is huge. In general, observation data consists of both heterogeneous spatio-temporal data and relevant observation metadata. The former includes data of Spatial Entities (cities, roads, vehicles, etc.) and data of temporal evolution of both properties of Spatial Entities (population of a city, position of a vehicle, etc.) and properties of space (temperature, elevation, etc.). Real uniform integrated management of all these types of data is still not achieved by current models and systems. The present paper describes the design of a data modeling and management framework for observation data warehouses. A hybrid logical-functional data model based on the concept of MappingSet and relevant language enables the specification of spatio-temporal analytical processes. The framework in currently being implemented.

1. INTRODUCTION

According to [16], properties of entities (called Features of Interest - FOI) are either exact values assigned by some authority (names, prices, geometry of a municipality, etc.) or estimated by some observation process (height, classification, color, etc.). Observation processes may be classified in various different ways [15]. *Physical Processes* produce their data in some spatial context. They are usually hardware sensing devices that perform measurements either locally or remotely. Besides, they may be installed in either static or mobile platforms. *Non-Physical Processes* are computations that may be defined in some mathematical way. Any process may be either *Time-triggered* o *Event-triggered*. The former perform their results at some predefined time frequency. The latter are started by some external event at any moment in time.

Observation data has an inherent temporal nature. Besides, in many cases FOIs are also spatial. Therefore, systems devoted to observation data analysis should cope with spatial and spatio-temporal data analysis. In particular, they should support relevant functionality for the management of Spatial Entities and Spatial Coverages, and their evolution with respect to time [9, 20, 6]. *Spatial Entities* are entities of a given application domain that have geometric valued properties (rivers, municipalities, cities, etc.). *Spatial Coverages* are sets of functions with a common spatial domain that describe the continuous or discrete variation over space of some specific phenomenon (temperature, humidity, elevation above sea level, etc.).

The amount of data that is currently being obtained from automatic observation processes is huge and the estimated tendency is to have an exponential growth during the upcoming years. The analysis of all these data to support appropriate decision making is key challenge for future information systems. Many application domains exist that would benefit from innovative technologies in this area, including environmental observation and monitoring, natural disaster management, e-health, etc.

Based on the above, in the present paper a data modeling and management solution is proposed that enables spatio-temporal analysis in data warehouses of observation data. In particular, a proposed E-R extension enables the insertion of observation metadata in spatial models at a conceptual level. At a logical level, a new data model based on MappingSets enables the integrated management of any kind of spatial and temporal data. A MappingSet is a collection of Mappings, in the functional programming sense, defined on a common domain. Both Spatial Entities and Spatial Coverages and both Time-triggered and Event-triggered observation data are modeled uniformly with MappingSets.

The remainder of this paper is organized as follows. Section 2 describes other pieces of work related to the proposed solution. The MappingSet based spatio-temporal logical model is described in Section 3. The conceptual level E-R extension for observation data is described in Section 4, as it is also its translation to the MappingSet based logical model. Section 5 illustrates the spatio-temporal analysis capabilities of the model for the definition of Non-Physical spatio-temporal analytical processes. Finally, Section 6 concludes the paper and outlines lines of future work.

2. RELATED WORK

The OGC defines an abstract specification of a data model for Observations and Measurements [16] in a Geographic Information context. Various types of observations are supported, according to the data type of their values. Simple observations include: i) measurements that combine a value of a real type with a unit of measure, ii) categories whose results are items of enumerated types, iii) counts of integer types, iv) truth observations of boolean type, v) time observations and vi) geometric observations. Complex observations are record structures that combine various simple observation types. Metadata of each observation is also represented in the model. In particular, each observation references its observation Process, the observed Property and its related FOI, the time instant when the observation applies to the FOI observed property (phenomenon time) and the time instant when the Process obtained the result value (result time). Notice for example that if a sample of water is obtained from a river and next analyzed in a laboratory two different observation time instants are involved. Optionally, other metadata, parameters, data quality information and observation context may also be provided.

In [4] a conceptual model to represent observation data semantics is defined. Annotating the conventional data models of available heterogeneous datasets with observation and measurement conceptual constructs enables their integration at a semantic level. Integrated query of heterogeneous observation datasets becomes therefore possible after the annotation process. A similar approach is followed by the E-R extension proposed in the present paper.

Observation data has always a temporal nature. Besides, the spatial components of observation data and metadata is centric to many application domains, such as those related to environmental observation and monitoring. Spatial and temporal extensions of conceptual and logical data models have to be considered. Examples of spatio-temporal conceptual models are [18, 17]. Relational and object-relational spatio-temporal extensions are defined in the area of Spatial Databases [9, 20] to support spatial entity management. Field [6] and array algebras [3] are behind spatial coverage and array management systems [14, 5, 2]. Integrated management of spatial entities and coverages is also objective of some approaches [19, 12], that incorporate different structures for those data types. Integrated management of entities and coverages in a uniform manner is achieved by the MappingSet data model proposed in the present paper.

Various different data management approaches are possible to deal with spatio-temporal observation data automatically generated by sensing devices. If we consider the data generated by each sensor as a virtual temporal relation, then the simplest approach is to consider *Materialized Views* of such virtual relations. Automatic maintenance of such views on the arrival of new data from sensors has to be solved by the system [8]. Automatically updating these views through Extraction Transformation and Load (ETL) processes on sensor data is the approach followed by the present framework.

A more sophisticated solution is to consider sensor data streams and to enable the continuous execution of queries on those input streams. Continuous query languages [1, 11] enable the definition of those continuous queries on both data streams and recorded relations. Operations to create relations from streams and streams from relations are at the core of those languages. A similar approach is followed by some languages specifically designed to access sensor networks [13, 7]. It is important to notice that spatial data, including spatial entities and spatial coverages and spatial analysis is not explicitly supported in these solutions.

3. SPATIO-TEMPORAL MAPPINGSET BASED DATA MANAGEMENT

This section introduces the MappingSet based data model that is the basis of the proposed framework. Temporal and spatial data types are first defined. Based on them Mappings and MappingSets are next formalized. Data management will be based on the intensional definition of MappingSets using both logical and functional paradigms.

Conventional data types include **Boolean**, **CString** (variable size character strings), **Int16**, **Int32**, **Int64** (integers), **Float32**, **Float64** (reals with floating point representation). Fixed point parametric type **Numeric(P,D)** consists of real numbers with a maximum of P digits, D of them are in the fractional part. In order to define temporal and spatial data types, 1D and 2D samplings are first formalized. Let R and I denote the set of real and integer numbers, respectively, then 1D and 2D samplings are defined as follows.

Definition 1. A **1D-sampling** S with resolution $r \in R$ and phase $p \in R$ is defined as the infinite subset of R
$\{x | x = i \cdot r + p, \forall i \in I\}$

Definition 2. Let vr_1, vr_2, vp_1 and vp_2 be four vectors in R^2 defined by respective directions D_1, D_2, D_1, $D_2 \in (-\pi, \pi]$ and respective magnitudes r_1, r_2, p_1 and p_2. A **2D-sampling** S with directions D_1, D_2, resolutions r_1, r_2 and phases $p_1 \in [-r_1/2, r_1/2]$, $p_2 \in [-r_2/2, r_2/2]$ is defined as the infinite subset of R^2

$$\begin{aligned}&\{(x, y) \in R_2 | \\ &\quad x = (i_1 r_1 + p_1)\cos(D_1) + (i_2 r_2 + p_2)\cos(D_2) \wedge \\ &\quad y = (i_1 r_1 + p_1)\sin(D_1) + (i_2 r_2 + p_2)\sin(D_2), \\ &\quad \forall i_1, i_2 \in I\}\end{aligned}$$

An element s of a 1D-sampling (2D-sampling) S is called a 1D-sample (2D-sample). Integer i, i_1, i_2 are called the *sampling coordinates* of s. $s(i)$, $s(i_1, i_2)$ denote respectively the 1D-sample and 2D-sample with sampling coordinates i and $(i_1,\ i_2)$. Figure 1 illustrates the above definitions with a geometrical representation.

Definition 3. **TimeInstant(D)** is defined as a finite subset of elements $s(i)$ of a 1D-sampling S with resolution 10^{-D} and phase 0 such that
$-2^{63} < i < 2^{63} + 1$
where each s(i) is interpreted as the time instant $1/1/1970 + s(i)$ seconds. Maximum allowed D is 6 (microsecond).

Definition 4. **TimeInstantSample(D, R)** is defined as a finite subset of elements $s(i)$ of a 1D-sampling S with resolution $R \cdot 10^{-D}$ and phase $(R \cdot 10^{-D})/2$ such that
$-2^{63} < i < 2^{63} + 1$
where each s(i) is interpreted as the time interval $[1/1/1970 + s(i)$ seconds, $1/1/1970 + (s(i) + R \cdot 10^{-D})$ seconds). Again, maximum allowed D is 6 (microsecond).

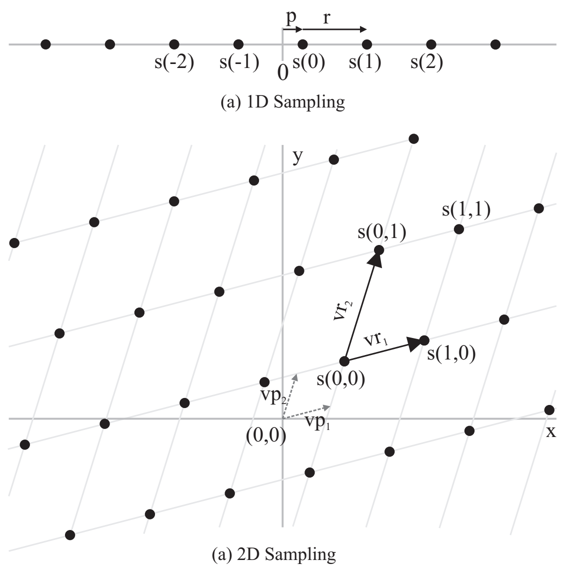

Figure 1: Illustration of 1D and 2D samplings.

Definition 5. **Date** is defined as a shorthand of TimeInstantSample(0, 86400).

Definition 6. **Point2D(P,D)** is defined as the finite subset of elements $s(i_1, i_2)$ of a 2D-sampling S with directions $D_1 = 0$ and $D_2 = \pi/2$, resolutions $R_1 = R_2 = 10^{-D}$ and phases $Ph_1 = Ph_2 = 0$ such that
$-10^P < i_1, i_2 < 10^P$

Definition 7. **Point2DSample(P,D,R)** is defined as the finite subset of elements $s(i_1, i_2)$ of a 2D-sampling S with implementation dependent directions D_1 and D_2, resolutions $r_1 = r_2 = K \cdot R \cdot 10^{-D}$ and phases $p_1 = p_2 = 0$ such that

1. $-10^P < i_1, i_2 < 10^P$
2. $K < max(\left|\cos(\frac{D_2-D_1}{2})\right|, \left|\sin(\frac{D_2-D_1}{2})\right|)$

TimeInstant and Point2D data types provide discrete representations for both time and space, where the user has control over the supported precision. Types TimeInstantSample and Point2DSample provide representations for temporal and spatial samplings at user defined resolution. It is noticed that each time instant is approximated by its closest lower TimeInstantSample, whereas each 2D point is approximated by its closest Point2DSample. It is out of the scope of this paper to demonstrate that K factor above ensures that any 2D point is approximated by a sample at a distance lower or equal to $R \cdot 10^{-D}$. Type castings are available for the above data types.

If T is either a numeric or temporal type, then data type **Interval(T)** is a new data type whose values are closed intervals over data type T. If t_1, t_2 are two elements of data type T, then $[t_1, t_2]$ is used to denote the relevant closed interval. Similarly, if S is spatial data type then the following geometric data types are also supported, based on the standard specification given by [10].

- **Geometry(S)**: Abstract type. Represents any vector geometry or set of geometries defined with elements of S.
- **LineString(S)**: Vector polylines defined by sequences of elements of S.
- **Polygon(S)**: Vector polygons, possibly with holes, whose borders are defined by sequences of elements of S.
- **GeometryCollection(S)**: Heterogeneous collections of Geometries.
- **MultiPoint(S)**: Homogeneous collection of elements of S.
- **MultiLineString(S)**: Homogeneous collection of elements of LineString(S).
- **MultiPolygon(S)**: Homogeneous collection of elements of Polygon(S).

Definition 8. If $ADT_1, ADT_2, \ldots ADT_n$ are not necessarily distinct data types, $A_1, A_2, \ldots, A_n$ are distinct names and RDT is a data type, then:

1. A **Mapping** with signature $M() : RDT$ is defined as a value of type RDT
2. A **Mapping** with signature
$M(A_1 : ADT_1, A_2 : ADT_2, \ldots, A_n : ADT_n) : RDT$
is defined as a partial function
$M : ADT_1 \times ADT_2 \times ADT_n \rightarrow RDT$

Operations are syntactic sugar for Mappings. Implicit castings between compatible data types are applied during Mapping invocations, enabling transparent transformation between temporal and spatial elements of different resolutions by applying constant interpolation. Various primitive mappings and operations are provided by the model. However, formalizing a complete set of them is out of the scope of the paper. Informal descriptions of required primitive mappings will be given throughout the paper.

A MappingSet is nothing but a set of Mappings that share a common domain defined as a n-ary relation over data types. Formalism is given below.

Definition 9. Let $C_1, C_2, \ldots, C_n$ be distinct names, ADT_1, ADT_2, ..., ADT_n be not necessarily distinct data types and RDT_1, RDT_2, ..., RDT_m be not necessarily distinct data types. Let also D be a n-ary relation with scheme $D(C_1 : ADT_1,\ C_2 : ADT_2,\ \ldots,\ C_n : ADT_n)$ defined as a finite subset of $ADT_1 \times ADT_2 \times \ldots \times ADT_n$. Then a **MappingSet** is defined in either of the three following forms:

1. A 1-tuple $MS = \langle D \rangle$.
2. A m-tuple $MS = \langle M_1, M_2, \ldots, M_m \rangle$, where each M_i is a Mapping with signature $M_i() : RDT_i$ defined as a value of RDT_i.
3. A (m+1)-tuple $MS = \langle D, M_1, M_2, \ldots, M_m \rangle$, where each M_i is a Mapping with signature $M_i(C_1 : ADT_1, C_2 : ADT_2, \ldots, C_n : ADT_n) : RDT_i$ defined as a partial function $M_i : ADT_1 \times ADT_2 \times ADT_n \rightarrow RDT_i$.

The evolution with respect to time of spatial entities and spatial coverages may be modeled with appropriate MappingSets that contain both Domain and Mappings. n-ary relationships are also modeled with MappingSets, usually without Mappings. MappingSets without Domain are also useful to record short collections of key-value pairs that are common in the specification of configuration settings.

The Domains and Mappings of a MappingSet may be defined either extensionally or intensionally. If a extensional definition of the Domain is given, then both extensional and intensional definitions of Mappings are allowed. On the other hand, an intensional definition of the Domain may only be accompanied by intensional definitions of Mappings. Generally, an extensional definition is a sequence of all the elements of Domain and Mappings in some specific order. Both row-wise and column-wise orderings may be used. It is even possible to combine row and column-wise orders for different components and Mappings. If the data type of a Domain component is of some integer or sampling data type, then its extensional definition might be given in the form of a collection of sequence definitions. In general, a sequence definition has an start element, a size and a step. For example, for an integer data type, a sequence starting at 5, with size 4 and step 2 describes the following list $< 5, 7, 9, 11 >$. For a TimeInstantSample data types, a sequence starting at "$2013 - 05 - 0215 : 00 : 45.06$", with size 2 and step 30.42 describes the following sequence of type TimeInstant(2, 3042) ¡"$2013 - 05 - 0215 : 00 : 20.22$", "$2013-05-0215 : 00 : 50.64$"¿. [1]. For Point2DSample data types, starting element is fo type Point2D and step has to be given by two pairs (direction, resolution).

Spatio-temporal analysis is enabled through the intensional definition of Mappings and MappingSets. Mappings may be intensionally defined with functional, conditional and aggregate expressions.

Functional expression. A Mapping M with signature M(D): DT may be defined by a expression of the form

M(D) := e

where e is a functional expression of data type DT that may include variables referencing components of D, mappings, operations, constants and castings.

Conditional Expression. A Mapping M with signature M(D): DT may be defined by a expression of the form

$$\begin{array}{llll} \text{M(D)} := & \text{CASE} & b_1 & \text{THEN} \quad e_1 \\ & \text{CASE} & b_2 & \text{THEN} \quad e_2 \\ & \ldots & & \\ & \text{CASE} & b_n & \text{THEN} \quad e_n \\ & [\text{OTHERWISE } e_{n+1}] & & \end{array}$$

where each b_i is a *functional expression* that yields a value of Boolean type and each e_i is a *functional expression* that yields a value of type DT. The semantics are the obvious ones.

Aggregate Expression. A Mapping M with signature M(D): DT may be defined by a expression of the form

M(D):= agge
OVER {P}

where P is a domain relational calculus predicate and agge is an functional expression where variables bounded to MappingSet domains in P must be used as arguments of aggregate functions. Various aggregate functions are provided by the system including both statistical and rank functions. MappingSet domains may also be intensionally defined.

[1]Notice that the start instant of the sequence is automatically adapted to match the underlying time representation for type TimeInstant(2, 3042)

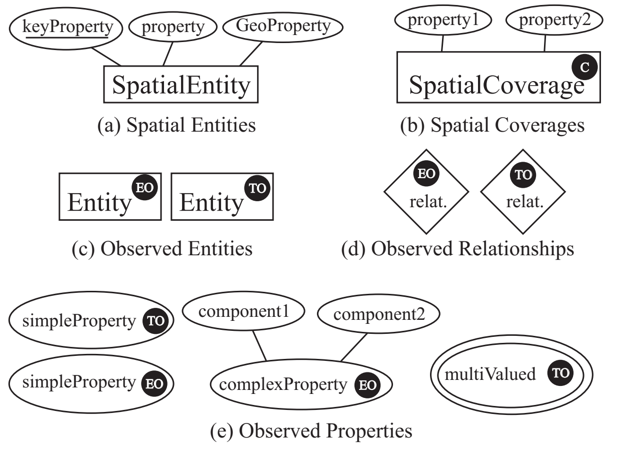

Figure 2: E-R Diagram Notation for Spatial and Observation Data.

Intensional Domain. Let e be a functional expression that yields a value s of either Interval(T) or Geometry(S) data type, whose base type T, S is either some integer type or some sample type. Then, SAMPLING(e) yields all the elements of type T or S contained in s. Based on this, the domain D of a MappingSet M may be defined by an expression of the form

$\{(e_1, e_2, \ldots, e_n)|P\}$

where P is a domain relational calculus predicate and each e_i is either a functional expression or an expression of the form $SAMPLING(e)$, where e is also a functional expression. Expressions e and e_i may include variable names bounded to MappingSet domain components in P. Given that nested structures are not allowed in the model, if an expression $SAMPLING(e)$ is used then the result relation has to be unnested.

4. MODELING OBSERVATION DATA WAREHOUSES

The data model described in this section captures observation data semantics and integrates them with spatial entities and coverages. An E-R extension is proposed in Subsection 4.1 to model observation metadata. The translation of such a conceptual model to the MappingSet based logical model is explained in Subsection 4.2.

4.1 Conceptual Data Model

Contrary to conventional metadata that is recorded at the level of entity and property types, some observation metadata has to be recorded at the level of entity and property instances, i.e., combined with the data itself. This is the case for example of observation time instants and observation processes.

An extension of the E-R model is next proposed to incorporate spatial and observation data semantics in conceptual models. *Spatial Entity types* are represented in diagrams as conventional entities (see Figure 2(a)). Spatial Coverage Types are represented as entities tagged with the symbol

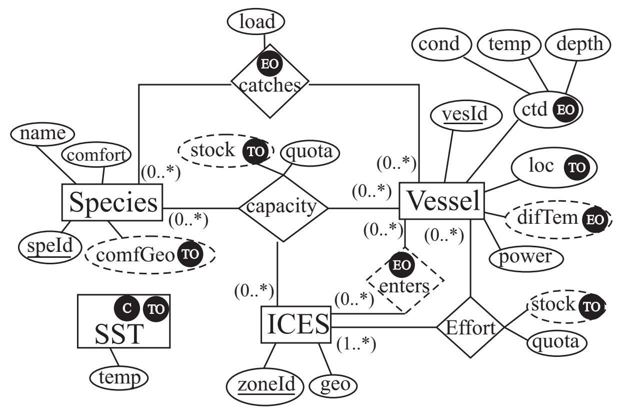

Figure 3: E-R Diagram of a Running Application Example.

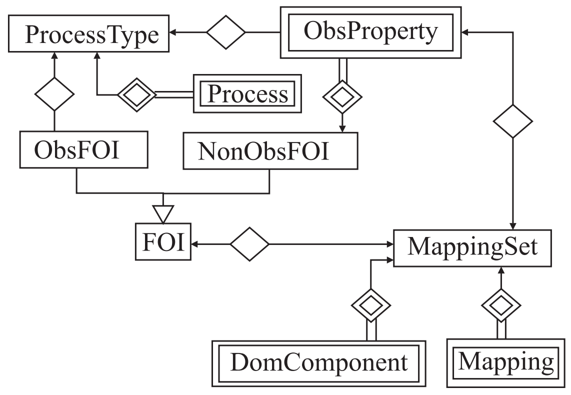

Figure 4: E-R Diagram of the Frameworks Catalog.

C (see Figure 2(b)). Entity Types, either spatial or not, and Coverages whose whole data is obtained through an observation Process are tagged with either symbol TO if it is a *Time-triggered Process* or symbol EO if it is an *Event-triggered Process* (see Figure 2(c)). Relationships resulting from observation processes are tagged in the same way (see Figure 2(d)). Finally, properties of either Entity or Coverage types that are obtained through observation processes are also tagged with the same TO and EO symbols, as it is shown in Figure 2(e) for simple, complex and multivalued properties.

To illustrate the use of the above notation the E-R diagram of a reduced running application example is given in Figure 3. Spatial Entity Type *ICES* records fishing zones defined by the International Council for the Exploration of the Sea (ICES). Spatial Coverage *SST* records Sea Surface Temperature at each location of the sea, daily produced by the Moderate Resolution Imaging Spectroradiometer (MODIS) sensor installed in the Terra and Aqua NASA satellites. Entity type *Vessel* records data of fishing vessels, including an identifier (*vesId*) and its engine *power*. Vessels incorporate CTD sensors that enable obtaining triples of water conductivity, water temperature and depth. Every time a ctd observation is performed a *Non-Physical Process* is executed that computes the difference with the value given by MODIS and provides it as a derived property *difTemp*. Vessels also incorporate GPS sensors from which locations are obtained every 30 seconds. Entity type *Species* records data of fishing species, including an identifier *specId*, species *name* and an interval of temperature values where the fish feels comfortable (property *comfort*). The derived property *comfGeo* records the geometry of the area of the sea where comfortable temperatures for the fish are located. This property is obtained by a *Non-Physical Process* from the SST data. Property *load* of relationship *catches* records the values measured by the vessel bascule for each species. The authorized fishing capacity of a vessel is given by two parameters. The Fishing Effort gives a measure of the number of days weighted by the vessel engine power that the vessel may stay in each zone. Relationship *Effort* records both the initial *quota* and the available one (property *stock*). Available Fishing Effort *stock* is obtained by a *Non-Physical Procress* using the quota and the vessels GPS information. The Fishing Capacity gives the kilograms of each species that the vessel may get from each zone. Again both *quota* is recorded and *stock* is computed by a *Non-Physical Process*.

The translation of the above model to the MappingSet logical model of the framework is explained in the following subsection.

4.2 MappingSet Based Logical Model

To support the implementation of the conceptual model of the previous section, observation metadata has to be added to the frameworks catalog. Thus, the catalog contains metadata of the defined Mappings and MappingSets and metadata related to the various observation processes, including observation properties and features of interest. The E-R diagram of such catalog structures is given in Figure 4.

Entity types *MappingSet*, *DomComponent* and *Mapping* record general metadata of the MappingSets. Entity type *FOI* records metadata of Features of Interest, and it references the MappingSet that records its data. FOIs that are fully generated by observation processes are registered in *ObsFOI*. The remainder FOIs, i.e., those that combine observed with non observed properties are represented by entity type *NonObsFOI*. Each observed property of such a FOI is represented by a weak entity of type *ObsProperty*, which references the MappingSet that records its data. Finally, *ProcessType* records metadata of the various types of observation processes registered in the framework. Metadata of each specific instance of each process type is recorded in weak entity type *Process*. Notice the difference between the process type "Vessel Bascule" that obtains values of *load* property of relationships *catches* and the specific bascule installed in each vessel that must be referenced from each observation.

The rules that enable the transformation of the conceptual model of the previous section to MappingSets are now given next. Each Entity Type, either Spatial or not, generates a relevant MappingSet, whose domain is defined by key properties and whose Mappings are defined by the remainder properties. See for example Entity Types Vessel, Species and ICES in Figure 3 and relevant MappingSets in Figure 5. Each Spatial Coverage generates a MappingSet, whose domain has just one component of some Point2DSample type and whose Mappings are generated from coverage properties. Each Relationship Type with cardinalities various to

```
MAPPINGSET Vessel
DOMAIN
 vesId: CString
MAPPINGS
 power(vesId:CString):Numeric(6,2)

MAPPINGSET Vessel_loc
DOMAIN
  obsTime: TimeIntantSample(0, 30),
  vesId: CString
MAPPINGS
  loc(phenTime: TimeIntantSample(0, 30),
      vesId:CString):Point2D
  process(obsTime: TimeIntantSample(0, 30),
          vesId:CString):CString

MAPPINGSET Vessel_ctd
DOMAIN
  obsTime: TimeIntant(0),
  vesId: CString
MAPPINGS
  cond(obsTime: TimeIntantSample(0, 30),
       vesId:CString):Numeric(4,1)
  condUOM(obsTime: TimeIntantSample(0, 30),
          vesId:CString):CString
  temp(obsTime: TimeIntantSample(0, 30),
       vesId:CString):Numeric(5,2)
  tempUOM(obsTime: TimeIntantSample(0, 30),
          vesId:CString):CString
  depth(obsTime: TimeIntantSample(0, 30),
        vesId:CString):Numeric(5,2)
  depthUOM(obsTime: TimeIntantSample(0, 30),
           vesId:CString):CString
  process(obsTime: TimeIntantSample(0, 30),
           vesId:CString):CString

MAPPINGSET Species
DOMAIN
 speId: CString
MAPPINGS
 name(speId:CString):CString
 comfort(speId:CString):Interval(Numeric(5,2))

MAPPINGSET ICES
DOMAIN
  zoneId: CString
MAPPINGS
  geo(zoneId:CString):Polygon(Point2D(9,2))
```

```
MAPPINGSET Capacity
DOMAIN
  vessel: CString
  ices: CString
  species: CString
MAPPINGS
  quota(vessel:CString, ices:CString
        species:CString):Numeric(7,3)
  quotaUOM(vessel:CString, ices:CString
           species:CString):CString

MAPPINGSET Effort
DOMAIN
 vessel: CString
 ices: CString
MAPPINGS
 quota(vessel:CString,
       ices:CString):Numeric(7,3)
 quotaUOM(vessel:CString,
          ices:CString):CString

MAPPINGSET Catches
DOMAIN
  species: CString
  vessel: CString
  obsTime: TimeInstant(0)
MAPPINGS
  load(species:CString, vessel:CString,
       obsTime:TimeInstant(0)): Numeric(7,3)
  loadUOM(species:CString, vessel:CString,
          obsTime:TimeInstant(0)): CString
  process(species:CString, vessel:CString,
          obsTime:TimeInstant(0)): CString

MAPPINGSET SST
DOMAIN
  loc:Point2DSample(9,2,100000)
  obsTime:Date
MAPPINGS
  temp(loc:Point2DSample(9,2,100000),
       obsTime:Date):Numeric(5,2)
  tempUOM(loc:Point2DSample(9,2,100000),
          obsTime:Date):CString
  process(loc:Point2DSample(9,2,100000),
          obsTime:Date):CString
```

Figure 5: MappingSets for a Running Application Example.

various generates a MappingSet whose domain is defined from the key properties of the participating Entity Types. Properties of those Relationship Types generate Mappings in such a MappingSet. See for an example Relationship Types capacity and effort in Figure 3 and MappingSets Capacity and Effort in Figure 5. If an Entity, Coverage or Relationship Type is tagged with the symbol TO, then a component named *obsTime* of some TimeInstantSample(D,R) data type is added to the MappingSet Domain to enable the recording of observation time.[2] Besides, a Mapping named *process* is also added to obtain the id of the process used to produce the observation. See for example Spatial Coverage Type SST in Figure 3 and relevant MappingSet SST in Figure 5. If symbol EO is used instead, then the data type of component obsTime is some TimeInstant(D). See for example Relationship Type catches in Figure 3 and relevant Catches MappingSet in Figure 5. In any of the above cases, an entity of type ObsFOI has to be added to the catalog with relevant relationships to its process type and MappingSet.

If a simple or complex property is tagged with symbol TO then such property is not added as a Mapping to the relevant MappingSet. Instead, a separate MappingSet is created for the property whose domain has components to reference the key of its Entity Type (FOI of the relevant observation) and has a component named *obsTime* of some TimeInstantSample(D,R) type to record observation time. The property itself is added as a Mapping to the MappingSet and an additional Mapping named *process* is added to record the id of the process that generates the observation. An example is loc property of Entity Type Vessel in Figure 3 and relevant Vessel_loc MappingSet in Figure 5. If symbol EO is used instead then the transformation is exactly the same except for the fact that Domain component *obsTime* is of some TimeInstant(D) type. For an example see ctd property of Vessel Entity Type in Figure 3 and relevant Vessel_ctd MappingSet in Figure 5. In any of the above cases an entity of type ObsProperty is added to the catalog, with appropriate references to its MappingSet, ProcessType and NonObsFOI.

Once the MappingSets are created and the required metadata are added to the catalog, the insertion of observation data may be started. ETL tasks are continuously executed to maintain the data warehouse updated with latest observation data, using extensional MappingSet definitions. Each observation is appended to the appropriate MappingSet with its observation time and reference to its process and FOI.

5. DEFINITION OF SPATIO-TEMPORAL ANALYTICAL PROCESSES

The capabilities provided by the framework for the intensional definition of MappingSets enable the specification of spatio-temporal analytical processes. These capabilities are now illustrated with some examples.

Example 1. Define a *Non-Physical Process* that obtains a derived observed property that computes the difference between the temperature measured by the CTD and the sea surface temperature produced for the same location by MODIS (see *difTemp* derived property of Vessel in Figure 3).

```
MAPPINGSET Vessel_difTem
DOMAIN
  {(obsTime, vesId) | Vessel_ctd(obsTime, vesId)}
MAPPINGS
  difTem(obsTime, vesId):=
    SST.temp(Vessel_loc.loc(obsTime, vesId), obsTime) −
    Vessel_ctd.temp(obsTime, vesId)
  difTemUOM(obsTime, vesID):=
    Vessel_ctd.tempUOM(obsTime, vesId)
  process(obsTime, vesID):= “difTemProcess”
```

In the expression above it is noticed that automatic castings of spatial and temporal types are performed during the evaluations of Mappings Vessel_loc.loc and SST.temp.

Example 2. Define a *Non-Physical Process* that detects when a vessel leaves an ICES zone to enter a new one (see *enters* derived relationship in Figure 3).

```
ICESFromLoc(loc):=
  singleton(zone)
  OVER {ICES(zone) ∧ within(loc, ICES.geo(zone))}

MAPPINGSET enters
DOMAIN
  {(vesId, ICESFromLoc(Vessel_loc.loc(obsTime, vesId)),
    obsTime) |
    Vessel_loc(obsTime, vesId) ∧
    ICESFromLoc(Vessel_loc.loc(obsTime, vesId)) <>
    ICESFromLoc(Vessel_loc.loc(predecessor(obsTime), vesId))}
MAPPINGS
  process(vesId, zoneId, obsTime):= “entersProcess”
```

In the above expression, Mapping *within*(g_1, g_2) yields true if geometry g_1 is within geometry g_2. Aggregate function *singleton*(S) yields the element contained in the unitary set S. Finally, Mapping *predecessor*(ts) yields the time sample that precedes time sample ts in its data type.

Example 3. Define a *Non-Physical Process* that produces a measure of the remaining fishing effort for each vessel and ICES zone for each of the preceding 60 days. Consumed fishing effort is obtained from the temporal evolution of vessel location data and ICES zone geometries (see derived property stock of relationship type Effort in Figure 3).

```
ICESFromLoc(loc):=
  singleton(zone)
  OVER {ICES(zone) ∧ within(loc, ICES.geo(zone))}

consumed_effort(vesId, zoneId, obsTime) :=
  ((count(obsTime2)*30)/86400)*Vessel.power(vesId)
  OVER {Vessel_loc(obsTime2, vesId2) ∧
    obsTime2 < obsTime ∧ vesId2 = vesId ∧
    ICESFromLoc(Vessel_loc.loc(obsTime2, vesId2)) = zoneId
    }

MAPPINGSET Effort_stock
DOMAIN
  {(vesId, zoneId,
    SAMPLING([cast(difTime(now(), 60 Days) AS Date),
      cast(now() AS Date)])) | Effort(vesId, zoneId)}
MAPPINGS
  stock(vesId, zoneId, obsTime):=
    Effort.quota(vesId, zoneId) −
    consumed_effort(vesId, zoneId, obsTime)
  stockUOM (vesId, zoneId, obsTime):=
    Effort.quotaUOM(vesId, zoneId)
  process(vesId, zoneId, obsTime):= “EffortStockProcess”
```

In the above expression, Mapping *now*() yields the current system time instant. Mapping *difTime*(t, i) subtracts time interval i from time instant t.

[2]Currently we restrict to phenomenon time semantics, however, it can be extended with result time and other required metadata.

Example 4. Define a *Non-Physical Process* that obtains the evolution with respect to to time during the last 7 days of the geometry of the comfort zone for each species. Comfort zone is obtained from the temperature interval defined for each species and the sea surface temperature generated by MODIS (see derived property comfGeo of entity type species in Figure 3).

```
MAPPINGSET Species_comfGeo
DOMAIN
  {(speId,
    SAMPLING([cast(difTime(now(), 7 Days) AS Date),
      cast(now() AS Date)])) | Species(speId)}
MAPPINGS
  comfGeo(speId, obsTime):=
    vectorize(loc)
    OVER { SST(loc, obsTime2) ∧ obsTime = obsTime2 ∧
      within(SST.temp(loc, obsTime2), Species.comfort(speId))}
  process(vesId, zoneId, obsTime):= “ComfortZoneProcess”
```

In the above expression, aggregate function *vectorize*(loc) obtains the vector geometry that surrounds the set of sample locations loc. Mapping within(e, i) yields true if element e is within interval i.

6. CONCLUSIONS AND FURTHER WORK

A data model and data management framework has been proposed spatio-temporal analysis of data in data warehouses of spatial observation data. The approach consists of an E-R extension for observation data to be used at a conceptual level and a new logical level model that combines logical and functional paradigms. The advantages of the approach can be summarized as follows:

- General purpose observation data and metadata coexist with application specific Spatial Entities and Coverages, enabling efficient analysis over the whole set.
- Few primitive Mappings combined with general purpose logical and functional expressions enable the integrated management of any kind of spatial and spatio-temporal data. Besides, both data and analytical processing is unified under the well known mathematical concept of function.
- Parametric temporal and spatial types enable the user to have control over the precision and resolution of underlying time and space representations.
- Specific constructs for the specification of sampled and non-sampled domain components together with the absence of nested structures simplifies efficient implementation.

Further work is mainly related to efficient implementation structures and algorithms and the extension of the framework to deal with continuous queries on sensor data.

7. ACKNOWLEDGMENTS

This work has been partially supported by the Spanish Ministry of Science and Innovation (TIN2010-21246-C02-02).

8. REFERENCES

[1] A. Arasu, S. Babu, and J. Widom. The cql continuous query language: semantic foundations and query execution. *The VLDB Journal*, 15(2):121–142, June 2006.

[2] P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, and N. Widmann. The multidimensional database system rasdaman. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98, pages 575–577, New York, NY, USA, 1998. ACM.

[3] P. Baumann and S. Holsten. A comparative analysis of array models for databases. In T.-h. Kim, H. Adeli, A. Cuzzocrea, T. Arslan, Y. Zhang, J. Ma, K.-i. Chung, S. Mariyam, and X. Song, editors, *Database Theory and Application, Bio-Science and Bio-Technology*, volume 258 of *Communications in Computer and Information Science*, pages 80–89. Springer Berlin Heidelberg, 2011.

[4] S. Bowers, J. Madin, and M. Schildhauer. A conceptual modeling framework for expressing observational data semantics. In Q. Li, S. Spaccapietra, E. Yu, and A. Oliv, editors, *Conceptual Modeling - ER 2008*, volume 5231 of *Lecture Notes in Computer Science*, pages 41–54. Springer Berlin Heidelberg, 2008.

[5] P. G. Brown. Overview of scidb: large scale array storage, processing and analysis. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pages 963–968, New York, NY, USA, 2010. ACM.

[6] J. a. P. Cerveira Cordeiro, G. Câmara, U. Moura De Freitas, and F. Almeida. Yet another map algebra. *Geoinformatica*, 13(2):183–202, June 2009.

[7] I. Galpin, C. Brenninkmeijer, A. Gray, F. Jabeen, A. Fernandes, and N. Paton. Snee: a query processor for wireless sensor networks. *Distributed and Parallel Databases*, 29(1-2):31–85, 2011.

[8] A. Gupta and I. S. Mumick. Materialized views. chapter Maintenance of materialized views: problems, techniques, and applications, pages 145–157. MIT Press, Cambridge, MA, USA, 1999.

[9] R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM Trans. Database Syst.*, 25(1):1–42, Mar. 2000.

[10] International Organization for Standardization (ISO). *Information technology – Database languages – SQL multimedia and application packages – Part 3: Spatial. ISO/IEC 13249-3:2011*, 2011.

[11] N. Jain, S. Mishra, A. Srinivasan, J. Gehrke, J. Widom, H. Balakrishnan, U. Çetintemel, M. Cherniack, R. Tibbetts, and S. Zdonik. Towards a streaming sql standard. *Proc. VLDB Endow.*, 1(2):1379–1390, Aug. 2008.

[12] M. Kersten, Y. Zhang, M. Ivanova, and N. Nes. Sciql, a query language for science applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, AD '11, pages 1–12, New York, NY, USA, 2011. ACM.

[13] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, Mar. 2005.

[14] M. Neteler and H. Mitasova. *Open Source GIS: A GRASS GIS Approach. Third edition.* Springer, New York, USA, 2008.

[15] Open Geospatial Consortium (OGC). *OpenGIS Sensor Model Language (SensorML) Implementation Specification*, 2007. http://www.opengeospatial.org/standards/sensorml.

[16] Open Geospatial Consortium (OGC). *Geographic Information: Observations and Measurements. OGC Abstract Specification Topic 20*, 2010. http://www.opengeospatial.org/standards/om.

[17] C. Parent, S. Spaccapietra, and E. Zimányi. Spatio-temporal conceptual models: data structures + space + time. In *Proceedings of the 7th ACM international symposium on Advances in geographic information systems*, GIS '99, pages 26–33, New York, NY, USA, 1999. ACM.

[18] N. Tryfona, R. Price, and C. Jensen. Chapter 3: Conceptual models for spatio-temporal applications. In T. Sellis, M. Koubarakis, A. Frank, S. Grumbach, R. Güting, C. Jensen, N. Lorentzos, Y. Manolopoulos, E. Nardelli, B. Pernici, B. Theodoulidis, N. Tryfona, H.-J. Schek, and M. Scholl, editors, *Spatio-Temporal Databases*, volume 2520 of *Lecture Notes in Computer Science*, pages 79–116. Springer Berlin Heidelberg, 2003.

[19] A. Vaisman and E. Zimányi. A multidimensional model representing continuous fields in spatial data warehouses. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 168–177, New York, NY, USA, 2009. ACM.

[20] J. Viqueira and N. Lorentzos. Sql extension for spatio-temporal data. *The VLDB Journal*, 16(2):179–200, 2007.