

The MADS Data Model

Concepts to understand the structure of your spatial and temporal data

Stefano Spaccapietra
Ecole Polytechnique Fédérale Lausanne (EPFL)
stefano.spaccapietra@epfl.ch

Christine Parent
Université de Lausanne HEC
christine.parent@unil.ch

Esteban Zimányi
Université Libre de Bruxelles
ezimanyi@ulb.ac.be

Successful information management relies on the ability to design accurate representations of the real world of interest, in spite of the diversity of perceptions from the applications sharing the same database. Current database management systems do not provide representation schemes that support adequate description of spatial and temporal features, and preserve the diversity of perceptions from the many users of a database while maintaining their consistency. This paper develops a data modelling paradigm where these facilities are fully supported within a user-oriented, conceptual modelling approach.

Keywords

Conceptual modeling, Database, Spatio-temporal data, Geographical Databases, GIS, Information Management.

Categories

H1, H2, H2.1.

1 - Introduction

The benefits of using a conceptual modeling approach to support database design are well known. Such an approach is commonly used for traditional databases, where the most used data models are the ER (entity-relationship) model [1] or UML (Unified Modeling Language) [2]. It is not so commonly used in geographical data management, where for a long time commercial systems have forced database designers to use logical data structures heavily influenced by internal implementation concerns.

A data model qualifies as conceptual if it enables a direct mapping between the perceived real world and its representation with the concepts of the model. In particular, a conceptual model does not have implementation-related concerns. While being conceptual, data models may have more or less expressive power, depending on the number of concepts and constructs they support. There is no obvious answer to the question of how much expressive power a data model should support. The problem lays in the fact that the complexity of reality is always beyond the expressive power of any data model. However, a very expressive model may be too complex and could eventually be rejected by users. Therefore, user comfort is a mandatory goal for conceptual data models. Since users must be able to participate in the database design effort they need to understand the modeling concepts and rules. Thus, an easy-to-understand visual support with clean, intuitive visual notations is the best means to facilitate discussion and assessment of a schema definition.

Simplicity and expressive power may be seen as contradictory requirements. The former looks for fewer constructs, the latter asks for more. Yet it is possible to combine both by following a golden principle: orthogonality. This principle says that the best way to handle multidimensional issues is to decompose them according to different facets, so that each facet can be handled independently from the other. Data modeling can be seen as a multidimensional issue, as the designer has to cope with different modeling dimensions: data structures, space, time, and representation, to name a few. The benefit of orthogonality is that understanding and solving one facet at a time is much simpler than solving all facets together.

The availability of an associated conceptual data manipulation language for querying and updating the data is a major plus. It allows users to perform all their database interactions using a single paradigm. Currently, we are far away from this ideal situation: Database users think in terms of ER or UML diagrams for schema definition and they have to speak relational SQL for data manipulation.

Finally, when addressing the requirements for data models for spatio-temporal databases, a few more requirements need to be considered. Regarding space modeling, it is essential for a conceptual data model to provide concepts for the description of both the discrete and the

continuous view of space, in a seamlessly integrated way. Both views are important for applications, which may use one or both simultaneously. Regarding time modeling, an additional requirement is the capability to address both transaction and valid time.

The data model described in this chapter has been designed to fulfill all of the above goals. The model is named MADS (for Modeling Application Data with Spatio-temporal features) A detailed presentation of the model can be found in [3. 4].

2. Thematic Data Structures

2.1 Object Types

Database objects or instances represent the real-world entities of interest to the applications. Object types group set of objects that, from the application point of view, may be considered as similar. An object type defines a set of properties of interest, i.e., attributes and methods. Fig. 1 illustrates an example of an object type. The population of an object type is defined as the set of instances of the type.

Attributes may be of two kinds. Those directly holding a value, e.g., number for AvalancheEvent, are called simple attributes. Those conveying a structuring concept, e.g., witnesses are called complex attributes. A complex attribute denotes a set of attributes that are either simple or complex. Every simple attribute is associated to a data type defining a value domain (i.e., the set of allowed values for the attribute) and the set of operations allowed on values in the domain.

Attribute cardinality denotes the number of values that an attribute may hold. Cardinality is defined by a couple numbers (min, max) defining the minimum and maximum number of values. Cardinalities define attributes as mandatory or optional depending on whether an object instance may be created without such attributes. Furthermore, cardinalities determine whether attributes are monovalued or multivalued, i.e., whether they hold at most one or multiple values. Multivalued attributes may be split into distinct categories, depending on the kind of collection they may hold, i.e., set, bag, or list.

To make sure that two different instances of an object type are distinguishable, even if they hold the same value, MADS assumes that each instance is identified by a system-defined attribute, called the object identifier, or in short oid. In addition, user-defined identifiers or keys express the fact that values of an attribute are unique within an object type. One or more keys can be defined for each object type. Fig. 1 shows the specification of the key, identified by the key icon, as consisting of the single attribute number.

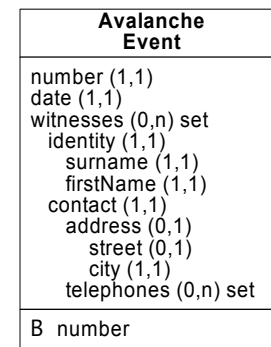


Fig. 1. A diagram of an object type

While attributes convey descriptive information, methods attached to object types, specify the operations that are specifically defined for the type. A method is specified by defining its signature (i.e., its name and the list of its input and output parameters), and its body (i.e., the actual code that will be executed when the method is activated).

2.2 Relationship Types

Objects are interrelated by relationships that provide paths to complementary information about the object. Relationships represent real-world links that are of interest to the application. As for object types, relationships that from the application perspective are considered as having the same characteristics are grouped in relationship types, where each relationship type conveys a link between two or more object types, as shown in Fig. 2. Associations in MADS are n-ary ($n \geq 2$). The set of instances of a relationship type constitutes its population.

MADS identifies two basic kinds of relationship types, association and multi-association, described next.

Associations are the most universally-known kind of relationship type. An association relationship type (referred to as an association type or as a relationship type) is a relationship type that links two or more object instances without imposing any specific semantics on the link. Fig. 2 shows a binary relationship type. The lines connecting the relationship to the linked object types materialize the roles within the relationship type. In a cyclic relationship type the same object type may be linked by two (or more) roles. For cyclic relationship types, roles linking the same object type must be equipped with a role name.

Cardinality constraints characterize each role of the relationship. For each role, its minimum and maximum cardinalities define the number of relationship instances that may link an instance of the object type linked by the role. For instance, cardinalities in Fig. 2 mean that an observer may have never observed an avalanche, and that an observer may have observed any number of avalanches, where these observations have an application-defined order. To make sure that these two (or more) instances of a relationship can be differentiated, MADS adopt a relationship identifier (shortly, rid). As object types, relationship types may be described by properties (attributes and methods).

In some situations the basic association relationship does not allow to accurately represent real-world links existing between objects. Fig. 3 shows two maps of the same area at different scales. Focusing on the area within the superimposed circles, we can see that the left-hand map shows five aligned buildings whereas at the same location the right-hand map shows only three buildings. Suppose now that the application stores the five buildings in the left-hand map as

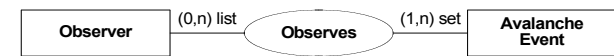


Fig. 2. A diagram showing a relationship type linking two object types

instances of the BuildingScale15'000 object type, and the three buildings in the right-hand map as instances of the BuildingScale25'000 object type. If the application requires to be able to correlate cartographic representations, at different scales, of the same real-world entities, this means creating a link between the 5 instances of BuildingScale15'000 on one side and the 3 instances of BuildingScale25'000 on the other side.

In order to be able to represent such situation MADS provides the multi-association construct. A multi-association relationship type is a relationship type that links, for each role, a non-empty set (or list) of instances of the linked object type. Consequently, each role in a multi-association relationship type bears two pairs of (minimum, maximum) cardinalities. A first pair is the conventional one that, as in association relationship types, defines for each object instance, how many relationship instances it can be linked to via the role. The second, additional, pair defines for each relationship instance, how many object instances it can link with this role. Its value for minimum is at least 1. Using a multi-association relationship type (visually represented by a double ellipsis), the above correspondence between cartographic buildings can be modeled as shown in Fig. 4.

Apart from cardinality constraints, multi-associations relationships share the same features as association relationships. They may be n-ary, cyclic, and bear properties. Their instances bear a unique and immutable rid.

2.3 Is-a Links

The is-a link, or the generalization/specialization relationship allows to relate a generic type (the supertype) and a specific one (the subtype) stating that they convey different representations of the same real-world objects. In the example of Fig. 5, the object type representing hazard zones is specialized in three subtypes for representing landslides, erosions, and avalanches zones.

Is-a links bear a population inclusion constraint enforcing that every object instantiated in the subtype is also instantiated in the supertype. The inclusion semantics naturally induces another well-known characteristic of is-a links: property inheritance, where all properties and links defined for the supertype also hold for the subtype. An immediate benefit of inheritance is type substitutability, i.e., enforcing the fact that wherever an instance of a type can be used in some data manipulation, an instance of any of its subtypes can be used instead.

Sometimes application requirements need to deviate from the standard inheritance rule where the definition of a property and its value are the same for the supertype and the subtype. As a first alternative refinement modifies the definition of the inherited property by making it more restrictive, but the value remains the same in both the supertype and the subtype. Another

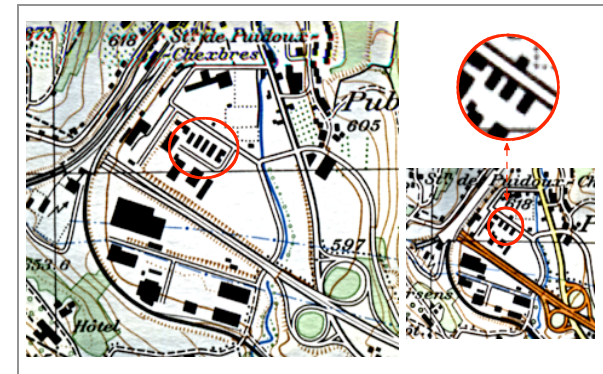


Fig. 3. An example of a data management situation that calls for using a multi-association relationship.



Fig. 4. An example of a multi-association relationship type.

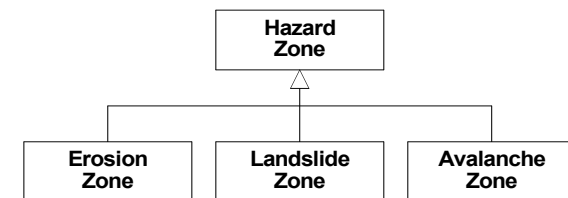


Fig. 5. Object types connected by is-a links.

alternative arises when applications require that instances in the subtype show a different value or behavior for an inherited property than the instances in the supertype. Redefinition aims at keeping substitutability of the property in the supertype with the corresponding property in the subtype for ensuring dynamic binding. Overloading relaxes this concern. For example, redefining in *AvalancheZone* the geometry attribute inherited from *HazardZone* allows keeping two different (e.g., at different resolution) but compatible specifications of the surface covered by the same zone. In overloading, the local definition and, in case of an attribute, the local value, simply replace the inherited ones, without any compatibility requirements.

MADS supports is-a links between relationship types, with the same characteristics (population inclusion semantics, property and role inheritance, possibility to add local properties). In Fig. 6 the designer wants to create a separate relationship type for observations of avalanches that are made by observers at the time of the avalanche (rather than after the avalanche). A sub-relationship type may have additional properties with respect to the super-relationship type. The diagram shows that both roles in *RealTimeObserves* have the same cardinalities as in *Observes* (plainly inherited roles are not redrawn in the sub-relationship type).

When allowing is-a links among relationship types, roles may be inherited with refined (i.e., more restrictive) cardinalities in the subtype. Further, as shown in Fig. 7, existing roles may be refined as associated to a subtype of the otherwise inherited object type. In this case the cardinalities of the role and the other role are inherited by *TrustedObserves* without modification. The design ensures that only observations made by accredited observers are registered as instances of *TrustedObserves*. Finally, new, local roles may be added with their respective object types (e.g., a binary relationship type may have as subtype a ternary relationship type).

2.4 Relationship Semantics

With this section we introduce a number of additions that can be used, in isolation or together with other additions, to give a specific semantics to relationships.

Aggregation relationships denote relationships that link an object seen as a whole with its component objects. As illustrated in Fig. 8, object types may be organized into aggregation hierarchies containing several levels. MADS associates specific terms, *isComponentOf* and *isComposedOf*, as semantic adornments to the roles to make explicit which object is the component and which object is the composed. Notice that aggregations are binary relationship types and they may be cyclic.

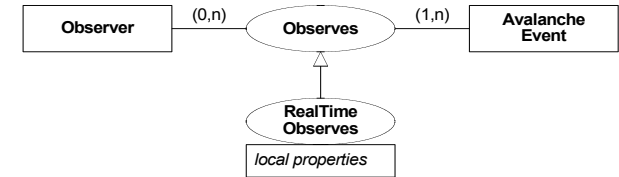


Fig. 6. A diagram showing an is-a link between two relationship types (with plain inheritance).

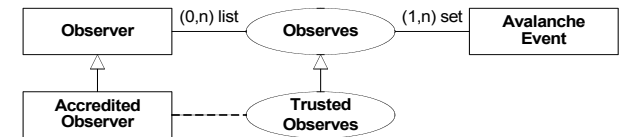


Fig. 7. A diagram showing a relationship subtype refining a role to link a subtype of the original object type.

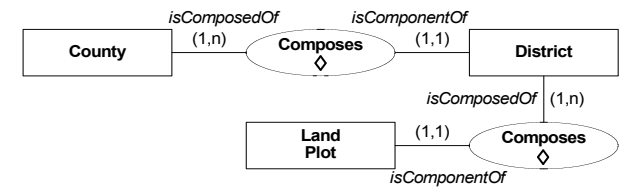


Fig. 8. An aggregation hierarchy in MADS, showing semantic adornment of roles. The rhomboid icon has been chosen to denote the aggregation semantics.

Transition relationships [5] allow to keep track of the dynamic evolution of objects in the database, i.e., how they appear and disappear within the populations of object types. They are binary association relationship types whose roles are given specific semantic adornments, *isSource* and *isTarget*, to specify the direction of the evolution. They may bear properties that describe information of interest about the evolution process. Fig. 9 shows a transition relationship type allowing to record information on acquisition of buildings by public organizations, e.g., who has taken the decision to use this public building as a crisis building and when the decision was taken. Transition relationships are constrained by the fact that they only relate instances with the same identity. The transition may leave the source instance in existence or require that it ceases existence when migrating to the target type.

Generation relationships [6] record information about creation of entities of a given type from other entities of the same or different type. An example is provided by land restructuring processes, where a set of land plots is reconfigured to form a different set of new land plots (cf. Fig. 10). The generation semantics induces semantic adornment of roles: *isSource* (designating the type of the generator instances) and *isTarget* (designating the type of the generated instances). In this particular example the generation semantics is associated with a cyclic multi-association relationship type. Generation relationships may bear properties. As several entities of possibly different types may concur in the process of creating one or several new entities of possibly different types, generation relationships are n-ary. The instances they link have different oids as they represent different real-world entities.

3 Spatio-Temporal Data Structures

3.1 Locating Objects in Space and Time

The intrinsic characteristic of spatio-temporal database systems is that they allow locating objects in space and in time. Locating an object means specifying the place the object occupies with respect to a given framework that holds all the possible places for an object. In GIS terminology, the place is termed the object extent, defined as the set of points that the object occupies in space (spatial extent) and in time (temporal extent). In order to efficiently define manipulate and query spatial and temporal extents, specific data types must to be provided. Tables 1 and 2 list the spatial and temporal data types supported by MADS, along with their associated icons. The spatial and temporal data types come with associated operations and predicates that users may use when writing their queries.

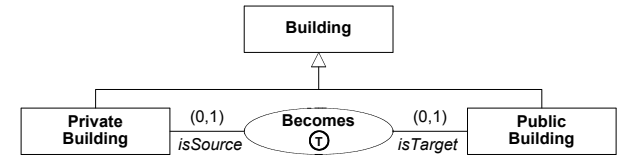


Fig. 9. A transition relationship type used to keep information about the transition from private to public buildings. The encircled T icon denotes transition semantics.

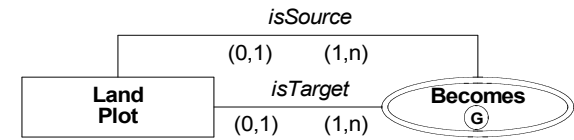


Fig. 10. Modeling land restructuring with a cyclic multi-association relationship type with generation semantics, denoted by the encircled G icon.

Spatial data types	Icon	Spatial data types	Icon
Geo		SimpleGeo	
Point		Line	
OrientedLine		Surface	
SimpleSurface		ComplexGeo	
PointBag		LineBag	
OrientedLineBag		SurfaceBag	
SimpleSurfaceBag			

Tab. 1. Spatial data types in MADS, and associated icons.

3.2 Describing Space and Time Using the Discrete View

The approach to define the spatial and temporal extents of the phenomena of interest using specific data types is referred to as the discrete view (or object view) of space and time. In the discrete view, space regions and time segments are either occupied by some autonomous entities, or are empty. In MADS space and time description is orthogonal to data structure description, which means that a phenomenon may be enhanced by spatial and temporal features whatever data structure (i.e., object, relationship, attribute) has been chosen to represent it.

A spatial (temporal) object type is an object type that holds spatial (temporal) information pertaining to the object itself as a whole. For instance, in Fig. 11 a) both object types are spatial while only LandPlot is temporal. Following common practice, we call spatio-temporal an object type that either has both a spatial and a temporal extent, separately, or has a time-varying spatial extent (i.e., its spatial extent changes over time and the history of extent values is recorded). Spatial and temporal information at the object or relationship type level is kept in dedicated system-defined attributes: geometry for space and lifecycle for time.

A spatial (temporal) attribute is a simple attribute whose value domain belongs to one of the known spatial (temporal) data types. A spatio-temporal attribute is a time-varying spatial attribute. Each object and relationship type, whether spatio-temporal or not, can have zero or several spatial, temporal, and spatio-temporal attributes. For instance, in Fig. 11 b) the LandPlot object type includes, in addition to its spatial extent, a spatial attribute buildings holding the spatial extent of all its constructed areas.

Spatial, temporal, and spatio-temporal relationship types hold spatial and/or temporal information pertaining to the relationship as a whole, exactly as for an object type. For instance, a Crosses spatial relationship type may be defined to hold the spatial extent covered by the intersection of two roads.

Temporality associated to object/relationship types or to attributes can be given two different interpretations. Transaction time keeps information on when some data element was stored in and deleted from the database. On the other hand, valid time conveys information on when a given fact, stored in the database, is considered valid from the application point of view. MADS supports valid time, which is the most usual requirement for the applications we have dealt with.

Temporal object or relationship types (cf. Fig. 12) allow users to keep the lifecycle of their instances. It is important for many applications to allow the membership of an instance in an object or relationship type to be suspended and reactivated. This allows describing situations like a professor temporarily leaving for a sabbatical. Consequently, in MADS a lifecycle is described by a peculiar time-varying attribute associating to each instant of the time domain one of the four possible status values: scheduled, active, suspended, and disabled.

Temporal data types	Icon	Temporal data types	Icon
Time	🕒	SimpleTime	🕒 _s
Instant	🕒	Interval	🕒
ComplexTime	🕒 _c	InstantBag	🕒
IntervalBag	🕒	TimeSpan	🕒

Tab. 2. Temporal data types in MADS, and associated icons.

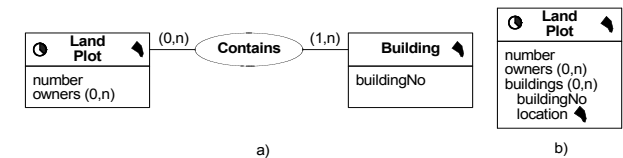


Fig. 11. Alternative schemas for land plots and buildings.



Fig. 12. Temporal object and relationship types.

As a consequence of capturing the lifecycle for instances of temporal types, the database may keep instances that are valid in the past, presently, and in the future. On the contrary, usual non-temporal types keep only currently-valid instances. For example, the object types as well as the relationship type in Fig. 12 are temporal. The temporal icons specify the temporal data type associated to the active state. Thus, in the example both observers and their assignments to avalanche monitored zones may be suspended and reactivated later on, while avalanche monitored zones have an active state which is continuous.

3.3 Space, Time, and Is-a Links

Generalization links may relate spatial and non-spatial object types. Any subtype of a spatial object type inherits the geometry of its supertype. For example, in Fig. 13 a), private buildings bear the same geometry they have when seen as generic buildings. Conversely, a non-spatial object type can have a spatial subtype: e.g., only cars equipped with a GPS system have a spatial extent as in Fig. 13 b).

The refinement, redefinition, and overloading mechanisms seen in Sect. 2.5 allow adjusting inheritance of spatial and temporal features to application's needs. Overloading can be used to model different spatial representations of the same real-world entity at different scales, as in Fig. 13 c). Objects of the subtype Road1:20'000 have two geometries: the one inherited from the supertype and the locally redefined one. Refinement allows attaching a more specific spatial or temporal data type to the subtype. For example, in Fig. 13 d) the generic concept hazard zone is used to describe both landslide zones and avalanche zones, but for operational reasons landslide zones are to be represented as simple surfaces while avalanche zones are to be represented as a set of oriented lines.

Similar considerations apply to temporal specifications. Subtypes of a temporal supertype are automatically temporal too (lifecycle is inherited), and non-temporal supertypes may have temporal subtypes. Further, the lifecycle may be overloaded or redefined in the subtype.

3.4 Constraining Relationships with Space and Time Predicates

The links among spatial and temporal object types play an important role in the description of spatio-temporal phenomena. For instance, in Fig. 14 the observations of avalanches by an observer are made only while the observer is on duty. We call constraining relationship type this kind of relationship type, bearing a specific spatial or temporal predicate on the geometries or lifecycles of the linked objects. Like any relationship type, they support attributes and methods, allowing designers to describe properties of spatial and temporal links.

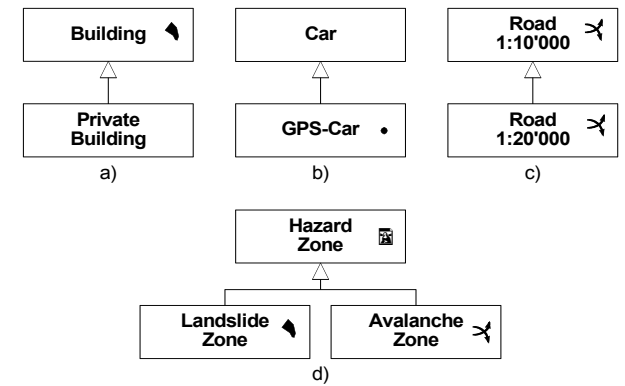


Fig. 13. Inheritance, refinement, redefinition, and overloading of spatiality: a) normal inheritance, b) adding local geometry, c) geometry redefinition or overloading, and d) geometry refinement.

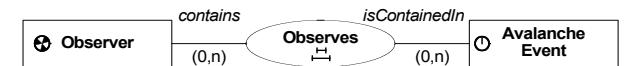


Fig. 14. A synchronization relationship type of kind SyncWithin.

Topological kind	Icon	Topological kind	Icon
TopoDisjoint		TopoTouch	
TopoOverlap		TopoCross	
TopoWithin		TopoEqual	
TopoGeneric			

Tab. 3. MADS topological relationship types with specific icons.

MADS supports topological relationships for the spatial dimension and synchronization relationships for the temporal dimension. For the most frequently-used constraining relationships MADS define specific icons, shown in Tables 3 and 4.

MADS allows associating multiple semantics to a relationship type. For example, a relationship type between spatial objects that has aggregation semantics often also conveys a topological (TopoWithin) semantics. Topology and synchronization semantics may also characterize the same relationship type, thus enforcing both a spatial and a temporal constraint. An example is shown in Fig. 15.

3.5 Describing Space and Time Using the Continuous View

Beyond the discrete view, there is a need to support another perception of space and time: The continuous view (or field view). This view focuses on the description of phenomena that are observable over a given space/time extent and are not related to any specific object within the extent.

An attribute is said to be space-varying (resp. time-varying) if its value changes according to the point in space (resp. instant in time) into consideration. A typical example of space-varying attribute is land elevation. Attributes may be both space-varying and time-varying, called also spatio-temporal attributes. For example, the value of land occupation depends on both the point in space and the instant in time that are being considered. Fig. 16 shows examples of varying attributes and their associated notation in MADS. Varying attributes are defined by a function whose domain in spatial and/or temporal extent. The range of the function can be any set of values (e.g., Real for temperature, Integer for rainfall). The range may be monovalued or multivalued; it may be simple or complex.

A topological relationship may link moving or deforming objects, i.e., spatial objects whose geometries are time varying. Fig. 17 shows such an example. In this case two possible interpretations can be given to this predicate, depending on whether it must be satisfied either in at least one instant or in every instant belonging to both time extents of the varying geometries [7]. Applied to the example of Fig. 17, the two interpretations result in keeping in the relationship either the land plots intersecting a risk zone for at least some time, or the land plots that always intersects a risk zone.

Synchronization kind	Icon	Synchronization kind	Icon
SyncPrecede	←	SyncMeet	⊥
SyncWithin	⊥	SyncOverlap	⊥
SyncStart	⊥	SyncFinish	⊥
SyncEqual	⊥	SyncDisjoint	⊥
SyncGeneric	⊥*		

Tab. 4. MADS synchronization relationship types with specific icons.

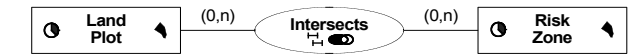


Fig. 15. A topological (TopoOverlap) and synchronization (SyncOverlap) relationship type.

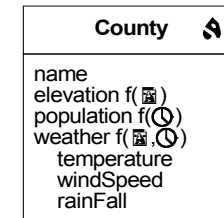


Fig. 16. A County object type with a space-varying attribute (elevation), a time-varying attribute (population), and a space- and time-varying attribute (weather).



Fig. 17. An example of topological relationship that links spatial object types with deforming geometries.

4 Supporting Multiple Perceptions and Multiple Representations

4.1 Rationale for Multiple Representations

Databases store representations of identifiable real-world phenomena that are of interest to a given set of applications. However, while the real world is supposed to be unique, its representation depends on the intended purpose. A known difficulty in database design is to reconcile divergent requirements of the applications sharing the same database. These requirements vary both in terms of what information is to be kept and in terms of how the information is to be represented. To support such heterogeneity of perceptions, the database has to be able to hold multiple representations of the same real-world phenomenon (cf. Fig. 18).

Geographical applications show additional requirements in terms of multiple representations.

One of them relates to spatial resolution, defined as the smallest granule of space that can be individually denoted. Choosing an appropriate resolution level is essential in map production. The ideal setting would be to keep the geometry information at the most precise resolution, and automatically compute geometries at less precise resolutions from those at more precise ones, a process called cartographic generalization [8]. Unfortunately, there is no complete set of algorithms that automatically derives a map at some resolution from a map at a more precise resolution. Thus, map production systems need to keep the geometries of objects at different resolutions.

Geographical databases are also subject to classical semantic resolution differences, where an application may see a road as a single object, while another one may see it as a sequence of road sections, each one represented as an object. As another example, geographical databases need to support hierarchical value domains for attributes, where values are chosen depending on level of detail. Fig. 19 shows a typical example.

4.2 Multiple Representation Modeling

Perception is guided by a specific interest in data management and determines a corresponding representation fitting that interest. In MADS the set of identifiable perceptions supported by a database are referred to as perception stamps, or just stamps, and are denoted as s_1, s_2, \dots, s_n . From data definitions (metadata) to data values, anything in a database relates to one or several

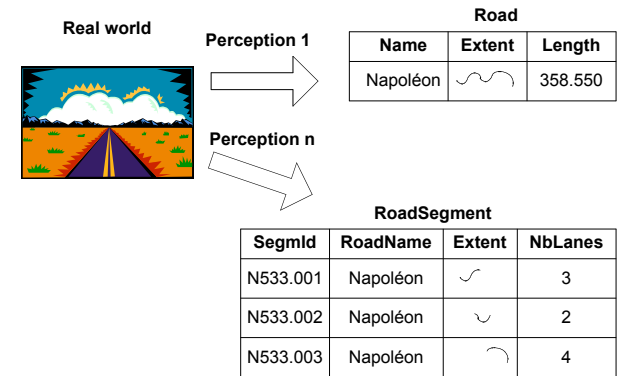


Fig. 18. Different perceptions of the same reality, in this case leading to different representations.

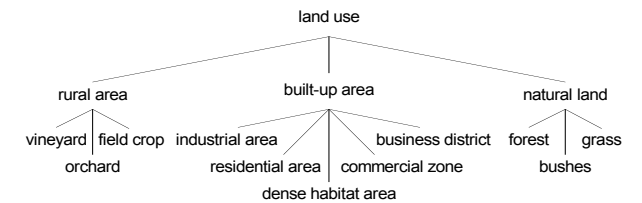


Fig. 19. An example of a hierarchical domain.

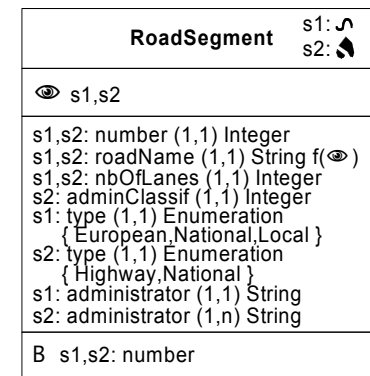


Fig. 20. An illustration of a bi-representation type, bearing stamps s_1 and s_2 .

perceptions. Stamping an element of a schema defines for which perceptions the element is relevant.

There are two complementary techniques to organize multiple representations. One solution is to build a single object type that contains several representations, the knowledge of “which is which” being provided by the stamps of the properties of the type. Following this approach, in Fig. 20 the designer defines a single object type, RoadSegment, and associates to it the stamps identifying the two perceptions, say s1 and s2, identified using the \square icon. We say RoadSegment is a perception-varying object type, as the actual representation of road segments changes from one perception to another.

Another solution to organize alternative representations is to define two separate object types, each one bearing the corresponding stamp (cf. Fig. 21). The knowledge that the two representations describe the same entities is then conveyed by linking the object types with a relationship type that holds a specific inter-representation semantics (indicated by the \circ icon). In this example the same real-world road segment is materialized in the database as two object instances. Instances of the relationship type Corresponds tell which object instances represent the same road segment.

The actual representation of instances of perception-varying object types changes from one perception to another. Consider the multi-representation object type RoadSegment (cf. Fig. 20) with stamps s1 and s2. The spatial extent of the type is represented either as a surface (more precise description, stamp s2) or as a line (less precise description, stamp s1) depending on resolution. Further, representation s1 needs attributes road segment number, road name, number of lanes, type, and administrator (denoting the maintenance firm in charge). Representation s2 needs attributes road segment number, road name, number of lanes, administrative classification, type, and administrator. While the road segment number and the number of lanes are the same for s1 and s2, the name of the road is different, although a string in both cases. For instance, the same road may have name “RN85” in representation s1 and name “Route Napoléon” in s2. We call this a perception-varying attribute, identified as such by the $f(\square)$ notation. The road segment type takes its values from predefined lists, the lists being different for s1 and s2. Finally, s2 may record several administrators for a road segment, while s1 records only one. Moreover perceptions s1 and s2 share a common key, the attribute number, i.e., no two instances, belonging to the same perception or different perceptions, may have the same number value.

Stamps may be also associated at the instance level. This allows defining different subsets of instances that are visible for different stamps among those supported at the type level. Thus, multi-perception types have a system attribute called perceptions allowing to keep track of the

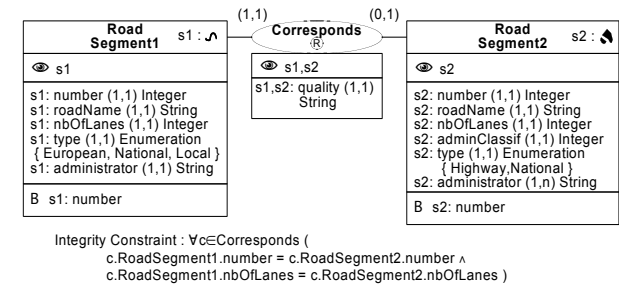


Fig. 21. The RoadSegment type (from Fig. 20) split into two mono-representation object types and an inter-representation relationship type.

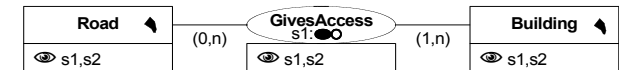


Fig. 22. A relationship type with perception-varying semantics.

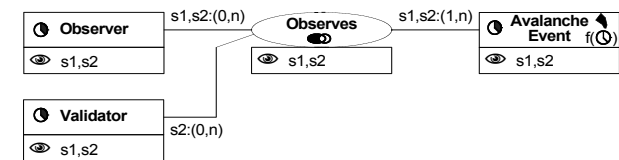


Fig. 23. A relationship type with an additional role specific to perception s2.

visibility of its instances. For example, as the object type RoadSegment in Fig. 20 has two stamps, it is possible to define instances that are only visible to s1, instances that are only visible to s2, and instances that are visible to both s1 and s2.

Relationship types are as dependent on perception as object types are. Therefore they may have different representations, and representations may be stamped with the same semantics and according to the same rules as for object types. Moreover, the structure (roles and association/multi-association kind) and the semantics (e.g., aggregation, topology, synchronization) may also have different definitions depending on the perception.

Fig. 22 shows an example of different semantics, where the designer defined the relationship GivesAccess as 1) a topological adjacent relationship type for perception s1, and 2) a plain relationship without any peculiar semantics or constraint for perception s2.

The kind association/multi-association of a relationship type may also depend on the perception. Similarly, the roles of a relationship type may also be perception dependent. Further, the same relationship type may be a binary relationship type in one perception and a ternary relationship type in another perception. For example, Fig. 23 shows that an observation is perceived in s1 as a relationship between an observer and an avalanche event, while perception s2 sees the same observation as also involving a validator having validated the observation.

5 Conclusion

This chapter presented the modeling concepts underlying the MADS conceptual data model. The objective that influenced most the design of MADS is that of orthogonality between the four dimensions: structural, spatial, temporal, and multi-representation. This allows building a model that is both simple (since these concepts are independent) and powerful (since these concepts may be freely combined). Orthogonality also allows answering to another fundamental requirement, the possibility to mix in one application classical and/or spatio-temporal and/or multi-represented data. To support these four modeling dimensions a set of data types has been defined. Another fundamental advantage of MADS is that it comes with an associated data manipulation language, which is also at the conceptual level. Thus, users can define and manipulate (i.e., query and update) the database using the same conceptual formalism. Such facilities are usually not supported by current conceptual approaches.

MADS has been tested in several applications. As an example, [4] presents the Risks application used as test case in the MurMur project and develops its complete conceptual schema. The advantages of MADS in terms of conciseness and readability appeared evidently. Designers of the application schema and corresponding queries also appreciated the comfort of not having to take into consideration the technical particularities of existing GISs or DBMSs. The project

developed a set of prototype tools supporting the MADS model, for schema definition as well as for query definition and visualization, thus making the approach operational in enterprise.

References

- [1.] P.P. Chen, "*The Entity-Relationship Model: Towards a Unified View of Data*", *ACM Transactions On Database Systems*, 1 (1) 1976: 9–36.
- [2.] J. Rumbaugh, I. Jacobson, and G. Booch, "*The Unified Modeling Language, Reference Manual*", 2nd edition. Addison Wesley, 2005
- [3.] C. Parent, S. Spaccapietra, and E. Zimányi, "*The MurMur Project: Modeling and Querying Multi-Represented Spatio-Temporal Databases*", *Information Systems*, available online April 2005.
- [4.] C. Parent, S. Spaccapietra, and E. Zimányi, "*Conceptual Modeling for Traditional and Spatio-Temporal Applications : The MADS Approach*", Springer, 2006.
- [5.] G. Hall and R. Gupta, "*Modeling transition*", [in] Proceedings of the 7th International Conference on Data Engineering, ICDE'91. IEEE Computer Society Press, 1991, pp. 540–549
- [6.] R. Gupta and G. Hall, "*An abstraction mechanism for modeling generation*". [in] Proceedings of the 8th International Conference on Data Engineering ICDE'92, 1992, pp. 650–658.
- [7.] M. Erwig and M. Schneider, "*Spatio-Temporal Predicates*", *IEEE Transactions on Knowledge and Data Engineering*, 14(4)2002: 881–901.
- [8.] R. Weibel and G. Dutton, "*Generalizing spatial data and dealing with multiple representations*", [in] P.A. Longley, M.F. Goodchild, D.J. Maguire, D.W. Rhind, eds., *Geographical information systems: Principles, Techniques, Management and Applications*, vol. 1, pp. 125–155. 2nd edition. John Wiley, 1999.