SPATIO-TEMPORAL DATABASE SUPPORT FOR LONG-PERIOD SCIENTIFIC DATA

M Breunig[1*], AB Cremers[2], S Shumilov[2] and J Siebeck[2]

*[*1]Institute of Environmental Sciences, University of Vechta, P.O. Box 1553, 49364 Vechta, Germany*
Email: mbreunig@iuw.uni-vechta.de
[2]Institute of Computer Science III, University of Bonn, Roemerstr. 164, 53117 Bonn, Germany
Email: (abc,shumilov,siebeck)@cs.uni-bonn.de

ABSTRACT

The quickly increasing number of spatio-temporal applications in fields like environmental management, geology and mobile communication is a new challenge to the development of database management systems. After discussing the related work we present concepts and implementations for the modelling and persistent management of geological long-period scientific data. The spatio-temporal classes are based upon GeoToolKit, an object-oriented 3D/4D geodatabase kernel system. Range queries on spatio-temporal objects are described in detail. We demonstrate the application of the spatio-temporal model with the balanced restoration of structural Rhine Basin evolution, an ambitious geologicql application . Finally, we give an outlook on our future research in the context of spatio-temporal database services.

Keywords: Spatio-temporal DBMS, geo-database management systems, geo-information systems, geo-database kernel systems, GeoToolKit, geological applications, 3D/4D applications.

1 INTRODUCTION

Database management system support for scientific spatio-temporal applications includes the wide range from real time application support, i.e. database support for the management of quickly moving objects, to the support of long period processes. In real time navigation systems, for example, the time intervals of interest for moving cars are minutes or even seconds (Brinkhoff, 1999). That is why the solution of the database update problem is emerging in these applications (Brinkhoff & Weitkämper, 2001). For patient-based computer supported medical documentations, to take another example, the relevant time units between the snapshots of the database are days, weeks or even years. Finally, geo-scientific processes like the backward restoration of basins (Alms, Balovnev, Breunig, Cremers, Jentzsch, & Siehl, 1998; Thomsen & Siehl, 2002) include time intervals of several thousand or even million years between every snapshot of the database.

The requirements for the modelling of time are manifold in geo-scientific applications. They are covering time conceptions like the modelling of discrete time stamps or dynamic geo-processes. The fact that entities of the dynamic environment may continuously change both location and shape makes their modelling a complex task. The management of such time-dependent geometries imposes challenges on the database community (Sellis, 1999; Breunig, Türker, Böhlen, Dieker, Güting, Jensen, Relly, Rigaux, Schek, & Scholl, 2003). Indeed, database models must be capable of representing these entities adequately. At the same time, data analysis demands interactive query processing facilities ranging from selections for online animation with different levels of detail to complex set-based operations like spatio-temporal joins (Güting, Bohlen, Erwig, Jensen, Lorentzos, Schneider, & Vzirgiannis, 2000). On the way to working systems, one must carefully define and implement *object-based* operations, which are the pre-condition of spatio-temporal querying.

In this paper we restrict ourselves to the requirements and data types needed for long time period applications like the simulation of geological processes. The data management requirements of these

applications differ from real-time applications. The remainder of the paper is organised as follows. The next section reviews related work for spatio-temporal models for database systems. Then, the next section introduces our new model. On this basis, fundamental query facilities, the spatio-temporal range queries, are defined. Implementation strategies are also described. The partly implemented model is used for a geological application, described in the next section. Finally, we give conclusions and mention aspects for future work.

2 RELATED WORK

A consensus representational model for spatio-temporal data has not yet emerged, and the field is divided into different supported applications. A large amount of work is targeted at discrete changes to spatial data. Models that combine spatial and temporal models without adding operational support for, e. g., interpolation between object snapshots, are therefore not targeted at continuously changing spatial objects. For example, the work of (Bohlen, Jensen, & Skjellaug, 1998) falls into this category, as it extends SQL-92 by adding features from the temporal and spatial domain with the special focus on upward-compatibility between the new model and SQL-92. The Tripod project aims at extending the ODMG standard for object models with spatio-temporal capabilities (Griffiths, Fernandes, Paton, Mason, Huang, Worboys, Johnson, & Stell, 2001). The intended system is targeted at discrete changes, modelled as so-called *histories*. A model for discrete change along with operators has been developed by Worboys (1994). The well-known spatial representational model of simplicial complexes is extended by bitemporal elements, hence the model can capture both valid and transaction time. Worboys`s model is the basis of the work of Chen and Zaniolo (2000), who define a data model, query language and an internal representation along with algorithms, called SQL^{ST}. The authors intend to add support for continuously changing spatial objects (Chen & Zaniolo, 2000).

A further class of applications is the moving-objects-database that track the position of, e.g., vehicles or mobile phone users. Of primary interest are objects that (continuously) change their location, while other spatial properties remain unchanged, for example shape. But as changing the shape of objects over time is at the heart of the model presented below, we omit the description of work on moving-objects-databases (but see, for instance, Wolfson, 2002).

(Grumbach, Rigaux, & Segoufin, 1998) show how to utilise the so-called constraint database approach to represent and query spatio-temporal objects; however, the model is limited to discrete change of spatial data. Related to the approach based on constraints is the work of Yeh and de Cambray (1995) and Yeh and Feautrier (1998). The authors present a model for multidimensional data that is allowed to be a function of a variable. In particular, spatial data can thus be made a function of time. Internally, a spatio-temporal object is represented as a collection of convex objects that are the result of a sweeping operation: if the spatial values in the sequence have n dimensions, then the sweeping operator constructs an n+1-dimensional object by applying a so-called *behavioural function* for each time-step in the sequence. This results in a set of convex polyhedra. Non-convex objects are stored as operator-trees, similar to constructive solid geometry. The advantage of this representation is that the temporal versions of the operations intersection, union, or difference can be realised by their spatial versions, operating on the internal representation. However, other important operations like the Euclidean minimum distance remain a challenge. Furthermore, by representing spatio-temporal objects through n+1-dimensional polyhedra, movement is restricted, since polyhedra are defined to have non-curved (i.e. linear) bounding faces.

Chomicki and Revesz (1999) present a two-dimensional data model that is based on vertex movement (as is ours, see below), termed parametric 2-spaghetti data model. It generalises Worboys`s model (Worboys, 1994) by allowing vertex positions to be linear functions of time. A negative result is reported, which states that the intersection of two parametric 2-spaghetti relations cannot be always represented as a parametric 2-spaghetti relation (non-closure).

Based on the relational model, (Cai, Keshwani, & Revesz, 2000) present the so-called parametric rectangles spatio-temporal data model. In this model, one attribute of a relation can be taken from the domain of parametric rectangles. Such rectangles have as bounding faces polynomial functions in one variable. Then, for each t within the lifetime of a parametric rectangle, a rectangle r(t) is defined. The query language for this model is an extended version of relational algebra. The operators projection, intersection, union,

difference, and complement are defined for relations with the new attribute type. Further operators are defined. All operators are closed, i.e. the result can be represented (exact) within the model. Non-linear motion can be represented.

An approach based on abstract data types has been described by Güting et al. (2000). The authors define an abstract model for continuous change of spatial data. Starting with a set of basic data types (including spatial types), the type constructor *moving* is defined that, if applied to one of the basic data types, constructs a new type that represents a mapping from time into the domain of the argument to *moving*. The data types are complemented by operations on the such constructed spatio-temporal types. An interesting aspect is the *lifting* of operations. First steps towards the implementation of the model has been described by (Forlizzi, Güting, Nardelli, & Schneider, 2000). The authors propose a data model that enables a finite representation of the spatio-temporal data types. Central to the approach is a sliced representation of spatio-temporal objects. According to the sliced representation, a time-dependent object is given by a finite sequence of snapshots, while the object can be interpolated for instants in between. Movement and change of extended spatial objects is based on a restricted form of point-movement: between two slices the movement of those points, which are connected to form a segment, must be within a plane in (x,y,t)-space. It is described, how the specification of the data model can be mapped onto the secondary storage structures of the Secondo extensible DBMS (Dieker & Güting, 2000).

3 GEOMETRIC MODELLING OF 3D MOVING OBJECTS

The movement of a point *P* in 3D space can be modelled as a trajectory in 3D space. For a time-dependent non-point object *O(t)* with a three-dimensional extension, however, the change of its geometry, i.e. the change of its given point set (value set of x, y, z-coordinates) in time has to be additionally considered. The change of the geometry of a 3D object *O(t)* between two time steps t_i and t_{i+1} may have at least three reasons:

1. change of *O(t)´s* location between time t_i and time t_{i+1} by means of translation or rotation on O(t_i);
2. change of *O(t)´s* scale between time t_i and time t_{i+1} by means of zooming (change of the of the object´s "size") on O(t_i);
3. change of *O(t)´s* shape between time t_i and time t_{i+1} by means of individual mappings on the volume of O(t_i).

Obviously, each of these three possibilities can also occur in combination with each other. The "trajectory" of a 3D moving object *O(t)* is dependent on the specific change of the geometry of the moving object and can be defined as follows:

$$\text{traj}(O(t)) = f(\text{Geom}_{xyz}(t)) \quad (1)$$

where $Geom_{xyz}(t)$ is the changing geometry of the object in time composed by its point set. Thus $f(\text{Geom}_{xyz}(t))$ in Eq. 1 describes the change of the location, the scale or/and the shape of *O(t)*. Within this paper we assume that an object decomposed into some objects is treated as a set of new objects from this time on. This set of objects may be decomposed later and merged to a single object again.

4 SPATIO-TEMPORAL MODEL OF GEOTOOLKIT

In this section, a new model is presented as the basis for representing, storing, and querying spatio-temporal data. Originating in previous work within the GeoToolKit projects (Balovnev, Bode, Breunig, Cremers, Müller, Pogodaev, Shumilov, Siebeck, Siehl, & Thomsen, 2004), several aspects of the representational spatio-temporal model are not new. Instead, the new model consolidates main characteristics of the previous work, and one result of this section is the formalisation of the new model. Being omitted in this contribution, overviews of the previous projects and their representational model for spatio-temporal data can be found in (Alms et al., 1998; Breunig, 2001; Thomsen & Siehl, 2002). Furthermore, we argue that the prevalent representations of time in temporal database models do not comply with the requirements of the spatio-temporal (geometric) settings. The remainder of the section is organised as follows. First, it briefly introduces the GeoToolKit-model for spatial data, as the model is extended with time. Second, the section

describes an object-oriented class hierarchy that emerges from the spatial model and can be used in application schema modelling by the users of the system. Finally, the new representational model for spatio-temporal data is presented.

4.1 Spatial Model

GeoToolKit (Balovnev, Breunig, & Cremers, 1997; Balovnev, Breunig, Cremers, & Pant, 1998; Breunig, 2001), our prototype geo-database kernel system, offers 3D data types implemented by simplicial complexes for the management and manipulation of geometries in three-dimensional Euclidean space.

The simplicial complex model is among the topological models for spatial data (Egenhofer, Frank, & Jackson, 1990). It has been chosen for the GeoToolKit projects. The central concepts are primitive objects, called simplices, which are the building block for complex objects, called simplicial complexes. These complexes are sets of simplices conforming to a topological constraint: the intersection of two simplices must also be contained in the complex. This model proved adequate for applications of the GeoToolKit projects (Balovnev et al., 1998). While these concepts are dimension-independent, for the scope of this contribution the number of dimensions will be fixed to three spatial dimensions. The reason is the capability of integration of datasets ranging from different disciplines like geography, geology, soil science, or meteorology, the underlying space for the representational model is assumed to be three-dimensional.

Definition 4.1.1. (*d-simplex*) Let P denote a set of $d + 1$ affine independent points. The convex hull of P, conv(P), is called a *d-dimensional simplex* (*d*-simplex for short). The elements in P are called the vertices of a simplex s and are denoted by $s^{(i)}$, $0 \le i \le d$. The dimension of s is denoted by dim(s). The convex hull conv(P') of each subset $P' \subseteq P$ is called a *face* of s, denoted by $f_{\{i_0,\ldots,i_j\}}(s) = \text{conv}(\{ s^{(i_0)}, \ldots, s^{(i_j)} \})$ with $i_0, \ldots, i_j \in \{0, \ldots, d\}$ pairwise disjoint, $j \le d$. The set of points in $\mathbb{R}^3$ that form a simplex s, denoted by points(s), is given by

$$points(s) = \{ p \mid p = \sum_{i=0}^{d} \lambda_i s^{(i)}, \quad \lambda_i \ge 0, \sum_{i=0}^{d} \lambda_i = 1 \} \tag{2}$$

Simplices can be combined to form complex objects; however, they must then obey constraints to form proper simplicial complexes as expressed in the following definition.

Definition 4.1.2. (*simplicial complex*) A finite set C of simplices forms a *simplicial complex*, if (a) for each $s \in C$ the faces of s are members of C and (b) for each $s_1, s_2 \in C$ the following condition holds: $s_1 \cap s_2 = \emptyset$ or $s_1 \cap s_2$ is a face of both s_1 and s_2. The dimension of C is defined as $\max_{s \in C} \dim(s)$.

The following restriction of the class of simplicial complexes results in spatial objects that represent simple line, areal, and volume objects.

Definition 4.1.3. (*d-mesh*) Let $d > 0$. A d-dimensional simplicial complex C is called a d-mesh if (a) for every simplex $s \in C$ with $\dim(s) < d$ an $s' \in C$ exists such that s is a face of s', and (b) for every (d-1)-simplex s'' there are at most two simplices which contain s'' as faces.

A 1-mesh is also called a *polyline*, a 2-mesh is also called a *trianglular mesh*, and a 3-mesh is also called a *tetrahedral mesh*. Restriction (a) in definition 4.1.3 states that a polyline only contains segments, a triangular mesh only contains triangles, and a tetrahedral mesh only contains tetrahedra as simplices that are not faces of other simplices within the complex. Restriction (b) states a manifold-condition: a vertex in a polyline connects at most two segments, a segment in a triangular mesh connects at most two triangles, a triangle in a tetrahedral mesh connects at most two tetrahedra.

The core set of spatial classes of the GeoToolKit-system is oriented to these concepts. In particular, it includes classes for 0D-3D simplices (points, segments, triangles, and tetrahedra) and classes for 1D-3D simplicial meshes (polylines, triangular networks, and tetrahedral networks). Furthermore, class *Group* allows the aggregation of instances of spatial classes, forming objects that correspond to the (more general) simplicial complexes.

4.2 Spatio-Temporal Model

This section introduces the representational model for spatio-temporal data. It extends the spatial model of simplicial complexes and combines the characteristics of the models developed in our previous projects. The main properties are: (1) it allows for both continuously and discontinuously moving and changing spatial objects; (2) the spatial domain is three dimensional; (3) it represents continuous change by *linear* vertex movement; (4) it separates vertices from mesh elements; (5) it allows for change in discretisation by time-stamping also on the simplex-level. The emerging concepts are temporal versions of the pure-spatial concepts: temporal simplices and temporal complexes, which each are interpreted as mappings from the temporal domain into the spatial domain. Furthermore, geometric constraints of the purely spatial model are injected into the spatio-temporal domain, for example, the constraint that at no instant of its lifetime a temporal triangle may degenerate to a line segment or point. Of course, a temporal simplex is allowed to degenerate at an instant outside of its lifetime, even at the boundary of one of its validity intervals if the interval is open on the respective side.

Time in the database context is well-defined; however, in the geometric setting of spatio-temporal databases, the concepts must be carefully adapted. In our model, time is interpreted as being isomorphic to the set of real numbers, which from now on we will use as the time domain. One element of this domain is called an *instant*. A temporal interval I is given by two instants t, t', where $t \leq t'$, permitting degenerate intervals [t, t]. Such intervals are needed, since the result of a range query and other operations can only be formed with such intervals available; for instance, the query "*When did temporal point (moving object) o intersect query plane p?*" can result in such a singleton. Furthermore, there is the special temporal interval *empty*. A temporal interval may be open on one or both sides. There are two special instants $-\infty$, ∞. The former is less than any other instant, the latter is bigger than any other instant. Therefore, the interval $[-\infty,\infty]$ exists and can be used, for instance, to denote the lifetime of pure-spatial objects. We denote the closure of a temporal interval I by $\bar{I}$ and the boundary of I by $\partial I = \{\inf I, \sup I\}$. A *temporal element* is defined as a set e of temporal intervals. If each pair of intervals in e does not overlap (, i.e., has no point in common), e is called minimal. The set of minimal temporal elements is denoted by $\mathbb{E}$. The elements in $\mathbb{E}$ are used to specify the lifetime of spatio-temporal objects.

As change is described through vertex movement, the first concept to be discussed is that of a temporal point. Then, temporal simplices and temporal meshes are introduced. A temporal point is a function from time into three-dimensional space:

$$v: \mathbb{R} \rightarrow \mathbb{R}^3 \tag{3}$$

The image of v is called *trajectory* of v, the curve in 3D

$$\text{traj}(v) = \{ v(t) \mid t \in \text{def}(v) \} \tag{4}$$

The set of such functions v is denoted by $\mathcal{V}$. One can break up function v into two components: (1) a position function that describes a curve in space (the trajectory); and (2) a velocity function that specifies the distance travelled on the trajectory, parameterised by time. However, for the model proposed here, a simpler scheme is adopted by limiting position functions to be piecewise linear and velocity functions to be piecewise constant. The implications are piecewise constant velocity and zero acceleration (except for the path vertices). More complex movement must be modelled by approximation. Furthermore, the functions v have elements from $\mathbb{E}$ as their domain. The former aspect of constant speed holds, precisely, for the time instants in the interior of the domain of a temporal point v. In the following, F^{lin} denotes the set of piecewise linear and continuous functions from time into $\mathbb{R}^3$. Hence, for two consecutive path vertices p_{t_1}, $p_{t_2} \in \mathbb{R}^3$ with time-values t_1 and t_2, respectively, the function evaluates for each instant t with $t_1 < t < t_2$ to

$$p_{t_1} + \frac{t - t_1}{t_2 - t_1}\left(p_{t_2} - p_{t_1}\right) \tag{5}$$

Definition 4.2.1. (*temporal point*) Let e denote a minimal temporal element. A *temporal point* v is a mapping

$$v: \{i \mid i \in e\} \rightarrow F^{\text{lin}} \tag{6}$$

that maps each interval in e to a piecewise linear, continuous function. The location of v at an instant $t \in i$, $i \in e$, is $v(t) := v(i)(t)$. The set of these mappings v is denoted by $\mathcal{V}^{\text{lin}}$.

To summarize, in our model a temporal point v consists of piecewise linear functions. Some support points mark changes in *direction*, some mark changes only in *velocity*, some mark both, and some mark discontinuities (see also Figure 2). No other restrictions, for example physical restrictions, apply to a temporal point except for those mentioned. Restrictions have to be imposed at the application level. We define the time-steps of a moving vertex v as the set of time values assigned to the support points of v.

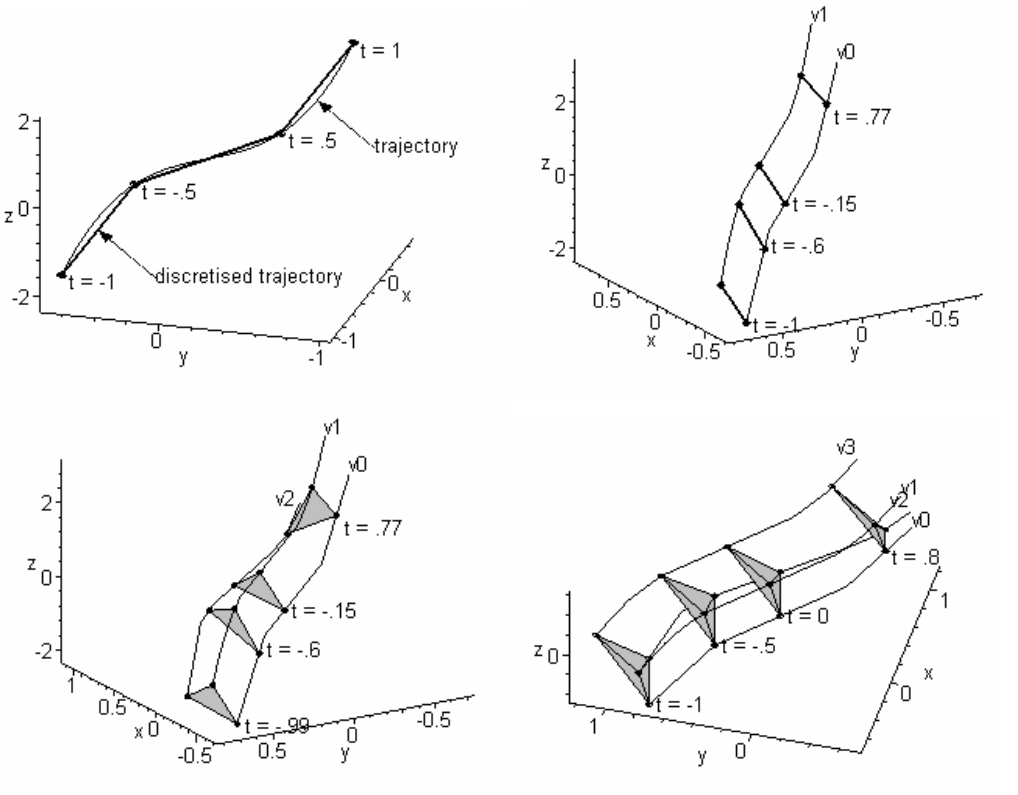


Figure 1. Temporal simplices. (a) A trajectory has been discretised linearly to form a temporal point; the support points are labeled by their time value. (b) Trajectory of two temporal points and some snapshots of the temporal segment defined by these points. (c) Similarly for a temporal triangle. (d) Similarly for a temporal tetrahedron.

Extending purely spatial simplices and simplicial complexes, d+1 temporal points in $\mathcal{V}^{\text{lin}}$ can be combined to form a d-simplex at every instant in their common lifetime. The following definition makes this precise (see also Figure 1):

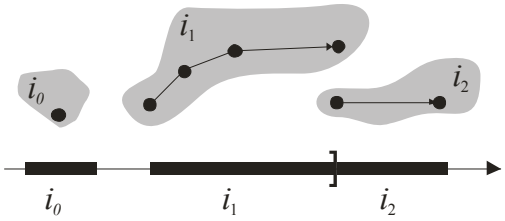


Figure 2. Trajectory of a temporal point and its temporal element of validity $e=\{i_0, i_1, i_2\}$ (time-axis indicated below). During i_0, the temporal point does not move (permitted by the above definitions). Between i_0 and i_1 it is undefined. The transition from i_1 to i_2 is a discontinuity. The bracket on the time-axis indicates that the point location at $i_1^+ = i_2^-$ is determined by the function for i_1.

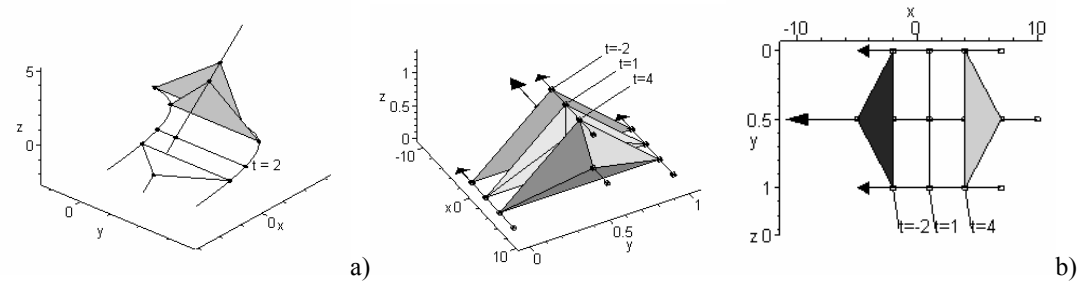
Definition 4.2.2. (*temporal simplex*) Let V denote a set of d+1 temporal points in $\mathcal{V}^{\text{lin}}$. Let $e \in \mathbb{E}$ be a temporal element. A temporal d-simplex s[V, e] is the function that maps each instant in e to the convex hull of V(t):

$$s[V, e]: e \rightarrow P(\mathbb{R}^3) \qquad (7)$$

$$t \mapsto \text{conv}(V(t))$$

For reasons of integrity, the following conditions on s[V, e] must hold: (1) the temporal points v_i in V must be valid during e, hence $e \subseteq \text{domain}(v_i)$; (2) for all $t \in e$ the points in V(t) must be affine independent. Similar to the pure-spatial case, the dimension d=|V|-1 of s[V, e] is denoted by dim s, each proper subset V' of V forms a (temporal) face s[V', e] of s[V, e].

The second condition in definition **4.2.2** states that at every instant in its lifetime, a temporal segment may not degenerate to a point, a temporal triangle may not be without area, and a temporal tetrahedron may not be flat (Figure 3).



a) b)

Figure 3. Two examples for temporal simplices violating the affine-independence condition of definition 4.2.2. (a) A temporal triangle invalid at t=2. (b) A temporal tetrahedron invalid at t=1; left: side view, right: top view.

Definition 4.2.3. (*temporal simplicial complex*) Let S denote a finite set of temporal simplices and $e \in \mathbb{E}$ a temporal element. A temporal simplicial complexes C[S, e] is the function that maps each instant $t \in e$ to the (pure-spatial) simplicial complex S(t). For reasons of integrity, the following conditions on C[S, e] must hold: (1) the temporal simplices s_i in S must be valid during e, hence $e \subseteq \text{domain}(s_i)$; (2) for each instant t in e the set S(t) must meet the properties of a simplicial complex (see definition 4.1.2).

Parallel to the pure-spatial case, the set of temporal simplicial complexes is restricted to from temporal d-meshes that means temporal polylines, temporal triangular meshes, and temporal tetrahedral meshes.

Definition 4.2.4. (*temporal d-mesh*) Let C[S, e] denote a temporal simplicial complex. C[S, e] is called a *temporal d-mesh* if (1) for each t in e the set S(t) is a valid d-mesh and (2) for each t, t' in e the condition dim(S(t))=dim(S(t')) holds.

4.3 Discussion

The presented representational model for spatio-temporal objects allows continuous change over time, induced by the linear functions of temporal points. Nevertheless, discontinuous changes to the geometry are not ruled out, for two reasons. First, a temporal *d*-mesh is permitted to contain simplices that change the mesh`s geometry in a discontinuous way at an instant. Second, as the trajectory of a temporal point is permitted to contain discontinuities, this is propagated to the level of temporal simplices and temporal meshes.

The model is representational only. Neither does it contain physical constraints, nor are there any restrictions for the transition from one timestep to the next that go beyond the injected spatial restrictions. As a consequence, it is left unspecified how to construct a spatio-temporal object, e.g., out of different (purely spatial) snapshots, or how to refine approximation. Application-dependent as such construction operations are, it is arguable whether they should be tightly integrated into a database model, and the assumption for this work is that they are to be maintained on an application's level.

Furthermore, it is easy to verify that the proposed model also conforms to the taxonomy of basic spatio-temporal processes according to (Claramunt, Parent, & Theriault, 1997). Every basic spatio-temporal process can be represented: (1) *stability*, since the trajectory of a temporal vertex can degenerate to a point yielding no movement at all; (2) *deformation*, since the trajectories of temporal vertices can be speci_ed independently; (3) *expansion/contraction*, since the trajectories of temporal vertices can be the result of a scaling function; (4) *rotation* for a rotation function, linearly approximated; (5) *translation* for a translation function applied to every temporal vertex in a temporal mesh. Additionally, also change in topology can be represented, although more demanding on the application`s level that must formulate a mathematical model to compute such changes.

Regarding the representable spatio-temporal objects as time-dependent point sets, it is worth examining the closure under set-based operations union, intersection, and difference. Conceptually, the definition of these operations is extended straightforwardly for the spatio-temporal setting. Given two spatio-temporal objects o1 and o2, the result of such an operation is a function of time, the domain of which is the intersection of the two input domains of o_1 and o_2. For example, the intersection operation can be defined as follows:

$$o_1 \cap_T o_2 : t \mapsto o_1(t) \cap o_2(t) \text{, where domain}(o_1 \cup_T o_2) = \text{domain}(o_1) \cap \text{domain}(o_2) \quad (8)$$

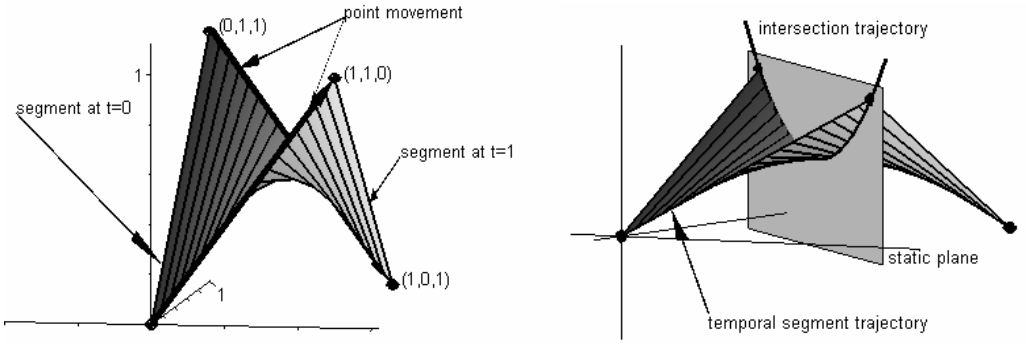


Figure 4. To the proof of proposition 3.4.1. (left) Trajectory of a temporal segment. (right) Intersection with a (non-temporal) plane resulting in a temporal point with non-linear trajectory.

Intersection and difference can be defined in the same manner. While any spatio-temporal object, as defined in the previous section, is a time-dependent point set, not every point set corresponds to a spatio-temporal object. The notion of a time-dependent point set is more general. Hence, the set of time-dependent point sets is partitioned into two classes, where the first class contains those point sets that can be represented by a

spatio-temporal object and the second class contains those point sets that cannot be represented by a spatio-temporal object as defined in the previous section. For this reason, it is interesting to see if operations on spatio-temporal objects can "lead out" of the first class, that is, if operations are closed under certain operations. More concrete, since these point sets have the representation given above, closure means here that the result of a set-based operation on any two given point sets *under the given representation* can always be brought into this representation. If there are cases where the result of such an operation cannot be represented within the model, then the model is not closed under this particular operation.

Proposition 4.3.1. The set of temporal simplicial complexes with the $\mathcal{V}^{\text{lin}}$ model of temporal points is not closed under intersection.

Proof: It must be shown that there are temporal simplices the intersection of which cannot be expressed in terms of the model. To this end, it suffices to construct an example based on a temporal segment. The temporal vertices v_1; v_2 of this segment are defined as follows:

$$\begin{aligned} v_1\colon [0,1] &\to \mathbb{R}^3 \\ 0 &\mapsto (0;0;0) \\ 1 &\mapsto (1,1,0) \\ v_2\colon [0,1] &\to \mathbb{R}^3 \\ 0 &\mapsto (0;1;1) \\ 1 &\mapsto (1,0,1) \end{aligned} \tag{9}$$

Hence, the first vertex performs a linear move from the origin to (1; 1; 0), the second vertex performs a linear move from (0; 1; 1) to (1; 0; 1) (see Figure 4). The trajectory of the so defined temporal segment is a bilinear interpolation of the aforementioned points: a curved surface. Therefore, intersecting the temporal segment with a static triangle can result in moving vertex trajectory that is non-linear (Figure 4). Indeed, only if in this case the triangle is placed such that it lies within a plane parallel to either xy- or yz-plane, a linear vertex movement results; however, all other placements would result in a non-linear movement.

Hence, to create a temporal complex by intersection, the resulting vertex trajectories must be approximated piecewise linear. A similar result for the two-dimensional space has been reported by Chomicki and Revesz (1999). However, the result presented here is stronger in the following sense. It shows that for three dimensions the intersection operation is not closed, even for the case of intersecting a spatio-temporal object with a pure-spatial object. An implication of the proof above is that other operations of interest are also affected, for example the trajectory-operator. Given a spatio-temporal object o, this operation results in a pure-spatial object corresponding to o`s trajectory. While in two dimensions, the model would support this operation, the situation is different in three dimensions. The trajectory of the temporal segment in the proof above forms a non-linear surface that cannot be represented within the model.

5 SPATIO-TEMPORAL DATABASE

5.1 Range queries for spatio-temporal objects

The conceptual model presented in the previous section will prove useful only if it is accompanied by retrieval facilities. In the following we describe intersection-operations for range queries. This allows the user to retrieve parts of a spatio-temporal object that intersect a query object. Thereby, the query object, which can be a plane, a half-space, a bounding box, or a spatio-temporal object, enables the user to cut out a part of a spatio-temporal object that is intersects the query object during a given temporal interval. From a user's point of view, range queries aim at restricting spatio-temporal objects to the "interesting" sub-regions of $\mathbb{R}^3$ and the temporal domain. They support therefore a closer object-inspection and, in particular, object-animation.

Technically, several advantages arise. Integrating these operations into the database kernel prevents client applications from loading parts of objects not contained in the current region of interest (both spatial and temporal). Furthermore, by an orthogonal design the results of operations can themselves be used as input to further querying. The reason for this is that the result is itself a spatio-temporal object. In the remainder of this section we define and demonstrate the processing of these operations.

We focus on the following set of operations that can be applied on any spatio-temporal object:

range_query: $Q \rightarrow stObject$ (10)

Here, Q denotes a query object that can be a plane, a half-space, a bounding box, or a spatio-temporal object. The meaning of the query performed on a spatio-temporal object o and a query object q is defined as follows:

o.range_query(q) := { (s, e) | $s \in o$, $e = s$.when-intersects(q), $e \neq \emptyset$ } (11)

Put differently, the operation extracts those sub-parts of o that intersect the query object q, limited to the temporal element at which the intersection happens.

The range queries as defined above do not generate new point data; instead, the returned simplices only reference existing temporal points. Alternatively, range queries can be defined such that simplices not completely contained in the query object are to be re-triangulated. Put differently, the intersection of a spatio-temporal object and the query object is computed (in the sense of Eq. (8)). During this process, new temporal points are generated, which stem from the intersection of the boundary or interior with the query object. However, proposition **4.3.1** holds and the computation cannot be exact. Here, we focus on the first alternative of range queries without re-triangulation.

To perform a range query, the problem is broken up into smaller units, each of which performs an elementary operation (see Figure 5). Linear moves of a temporal point occur during two consecutive time-steps on its time-line: with linear point movement, only two time-steps are needed to compute the position of the vertex in between. Then, consecutive time-steps on the merged time-line mark time intervals during which both temporal points perform linear moves (see big arrow in Figure 5). Hence, on this interval temporal simplices can be represented in $O(1)$ space. In general, elementary spatio-temporal $O(1)$-operations apply to temporal simplices, limited to consecutive time-steps on their merged time-lines. Such a limited temporal simplex is called spatio-temporal *primitive* object.

Definition 5.1.1. (*spatio-temporal primitive object*) Let o denote a spatio-temporal object. The object o is called a spatio-temporal primitive object, if o is a plane, half-space, bounding box, or temporal simplex s[V; e], and for the latter case: $e = \{(0; 1)\}$, and for each $v \in V : v_{|e}$ moves on a straight line.

The rationale behind the definition is as follows. Given two spatio-temporal primitive objects s_1[V_1; e_1] and s_2[V_2; e_2], a primitive operation operates upon these primitives, limited to the interval $e_0 = e_1 \setminus e_2$. A different perspective on primitive operations is through the merged time-line. Given two (complete) temporal simplices s; s', one can merge the time-lines of both objects and choose the intervals for which both objects are defined (see Figure 5). For each such interval, two spatio-temporal primitive objects can be obtained by transforming the interval to (0; 1) (through translation and scaling). Importantly, both objects are continuous on (0; 1) such that methods from calculus can be applied to perform basic computations, like the minimum Euclidean distance between the spatio-temporal primitive objects or testing for intersections.

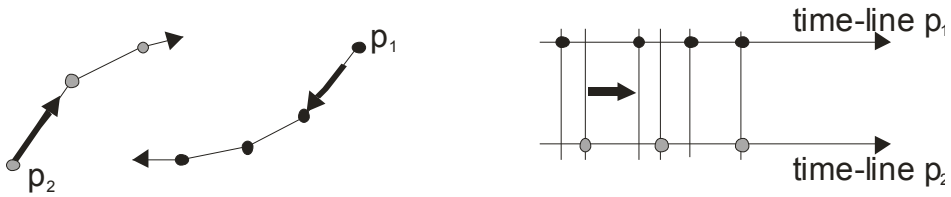


Figure 5. The merged time-line (right) of two temporal points (left) induces a sequence of spatio-temporal primitive objects. The bold arrow indicates a time interval for a spatio-temporal primitive object.

Hence, to perform a range query on a spatio-temporal object o and a query object Q, the solution space consists of all pairs of primitive objects:

$$S = \{ (p, p') \mid p, p' \text{ primitive objects}, p \text{ in } o, p' \text{ in } Q \} \quad (12)$$

This space must be searched for intersecting pairs.

The concept of a *gstepIterator* is used to iterate over the spatio-temporal primitive objects of a spatio-temporal object. The implementation of *gstepIterators* for temporal meshes is outside the scope of this contribution; however, a *gstepIterator* can be bound to a spatio-temporal object, optionally parameterised by an axis-aligned bounding box and a temporal interval. The parameters are used to limit the iteration to those primitive objects that intersect the bounding box (spatially) and the interval (temporally).

The following procedure implements a range query for a spatio-temporal object o and a query object Q. Recall that Q can be a plane, halfspace, axis-aligned bounding box, or a spatio-temporal object. In the latter case, Q can have an index for its constituent parts. Such an index is utilised in line 4 by bounding the iterator i2 with the minimum bounding box around the trajectory of p1 and the validity interval of p1 (hence, the name of the procedure is justified.) With the index, the candidate set of relevant primitive objects in Q—those that intersect p1—can be found fast. The description of our indexing technique for spatio-temporal objects is outside the scope of this contribution. Importantly, in the case that both o and Q are supported by indexes, a different implementation is possible. Put briefly, the implementation traverses both index trees in a synchronised, depth-first way to find the pairs of relevant primitive objects. However, the details of this computation are omitted. If Q is not indexed, i2 iterates over all primitive objects contained in Q (nested-loop-fashion). The case of o being indexed can be handled accordingly.

Range_query_index_nested_loop(*stObject o*, *Q*)

```
 1: stObject result
 2: gstepIterator i1(o)
 3: while (p1 = i1.next()) do
 4:   gstepIterator i2(Q, p1.mbb, p1.interval)
 5:   while (p2 = i2.next()) do
 6:     e = p1.when_intersects(p2)
 7:     if (e ≠ ∅) then
 8:       result.insert(p1, e)
 9:     end if
10:   end while
11: end while
12: return result
```

6 APPLICATION OF THE CONCEPTS

We have tested the temporal extension of GeoToolKit within a concrete geological scenario. One of the objectives was to examine the balanced restoration of structural Rhine Basin evolution (Thomsen & Siehl, 2002). The geological model of this application consists of stratigraphic and fault surfaces extended by a time attribute. The animated surfaces of that model were visualized in VRML (Shumilov, Thomsen, Cremers, & Koos, 2002) and sent to GeoToolKit for spatio-temporal database querying.

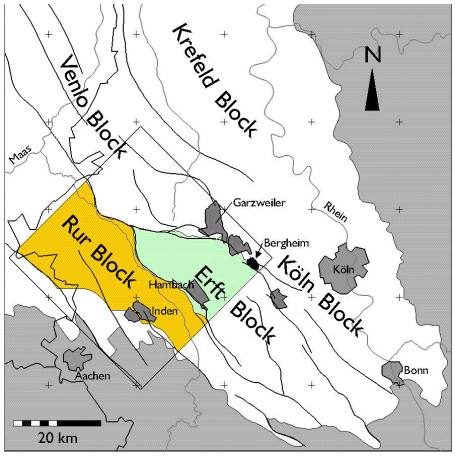


Figure 6. Position of the Bergheim mine (arrow) in the Lower Rhine Embayment.

One of the objectives of the geological model was to examine the balanced restoration of structural Rhine Basin evolution. This geological process started in tertiary period, i.e. about 65 million years ago. The exact geological movements, however, are unknown. Therefore different possible variants for the time sequences have to be examined by the geologists.

The region under consideration is the area of an open pit lignite mine "Bergheim" which has a diameter of about 4km and a relative depth of about 500 m (see Figure 6). The Bergheim mine has been in operation by Rheinbraun AG, Köln, since 1984 with an annual output of 15.7 Mill. tons (1996/97) of lignite.

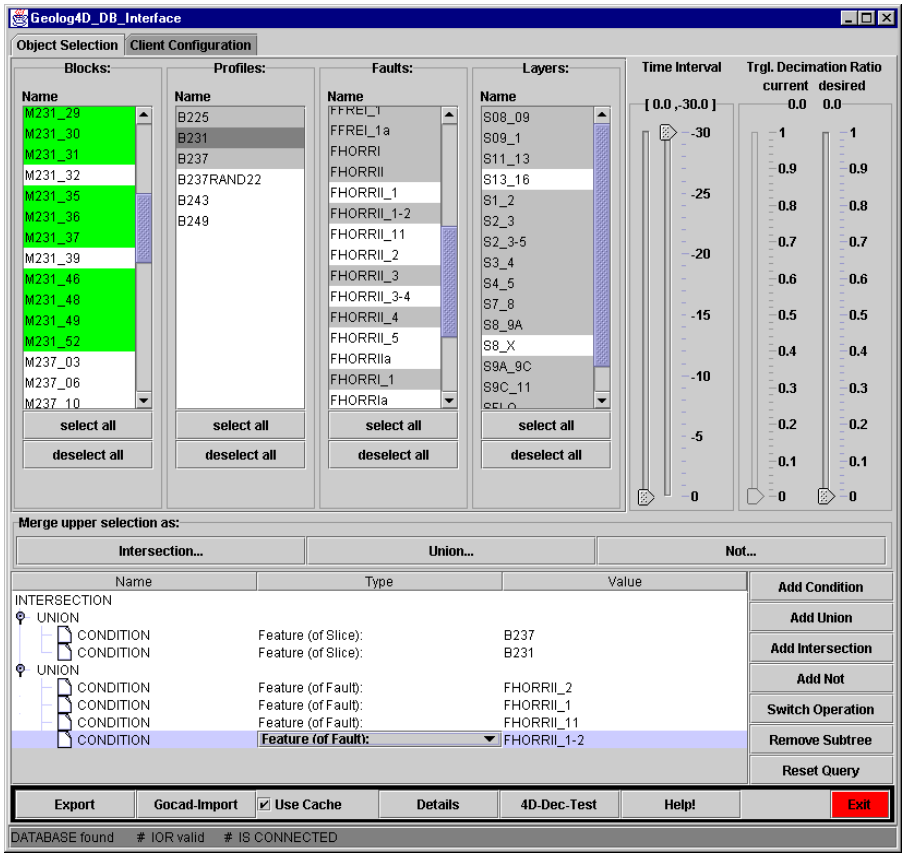


Figure 7. A screenshot of the client's graphical user interface.

The originally static 3D-model on the basis of approx. 20 digitized profile sections of Rheinbraun AG was prepared for the movements. Rouby's method of map plane restoration (Polthier & Rumpf, 1994) and other custom programs were used for the calculation of the movement. The interactive design of the geometry was performed with GOCAD[tm] (TSurf, 2002) 3D modeling program, while the VRML was used for visualization and animation (Shumilov et al., 2002).

A typical session with the client interface might look as follows: During work with a 3D/4D modelling application, the user wishes to retrieve geometry objects from a remote database for insertion into his local model. After starting the client interface and establishing the connection with the database, the GUI window with several sections appears (Figure 7). Its listbox windows display possible attribute values, (Figure 7, sections to be read from right to left) which can be used for object selection. Objects are classified according to four basic *geological types*: "Layers" (stratigraphic surfaces), "Faults" (tectonic discontinuities), and "Profiles" (geological sections). The list "Blocks" displays the volume blocks of the model. These *geological types* are typical for many geological applications, and do not limit the usability of the interface to the single Bergheim model.

The user navigates through the database, marking attribute values in the listboxes, which are implicitly combined by a boolean "AND" operation. In Figure 3 attribute "Profile" with value *B249* was selected, i.e. all objects that contain a "profile" boundary surface *B243*. On Figure 7, the selected profile value is marked in dark gray. In addition, we observe that "Fault" *FA12*, *FA42*, ..., and "Layer" *S 11_13*, *S13_16*, *S1_2*, *S2-3*, etc. are marked in light grey, signalling other attribute values taken by the selected objects. The user then may pick or unpick additional objects, or modify the selection condition by choosing other attributes.

6.1 Complex queries

To define more complex queries, the user may use a query constructor and manipulate the query tree displayed in the lower part of the GUI (Figure 7). The query is composed of simple boolean operations (OR, AND, NOT) combining simple type-value pair conditions. Figure 7 displays a query consisting of a union of two union operations: the first operation selects blocks with names *M231_02*, *M231_10*, and the second selects objects with the attribute "profile" *B249*.

The user can add new operations to the query by activating buttons "Add Union", "Add Intersection", "Add Not". Button "Switch Operation" permits to change the type of an existing operation. Activating button "Add Condition", the user can define a new condition and combine it into the selected boolean operation. Attributes selected in one of the listboxes can also be inserted in the query using one of the buttons "Intersection", "Union" or "Not". Two more buttons permit to remove either subtrees or to reset the whole query. Besides defining custom query expressions, the tree structure is also useful for checking the graphical selection described above.

Whereas the interface originally was used for the Bergheim kinematic model with complex volume objects, it can be also be used for geological applications handling only surfaces. Activating button "Details" one can select partial objects for retrieval. We did not implement purely spatial queries provided by GeoTooKit in this particular prototype, as in the present application, attribute-based queries provide sufficient problem-specific criteria and navigation facilities. Purely spatial queries e.g. bounding box queries are supported by the database server and may be implemented in a future version of the client interface.

A preview operation will be often used in practical work before loading the selected data for further processing in specialized modelling programs like GOCAD. Therefore, when a graphical representation is required, the retrieved objects are converted to VRML and visualized using a common VRML-capable WWW Browser (Figure 8).

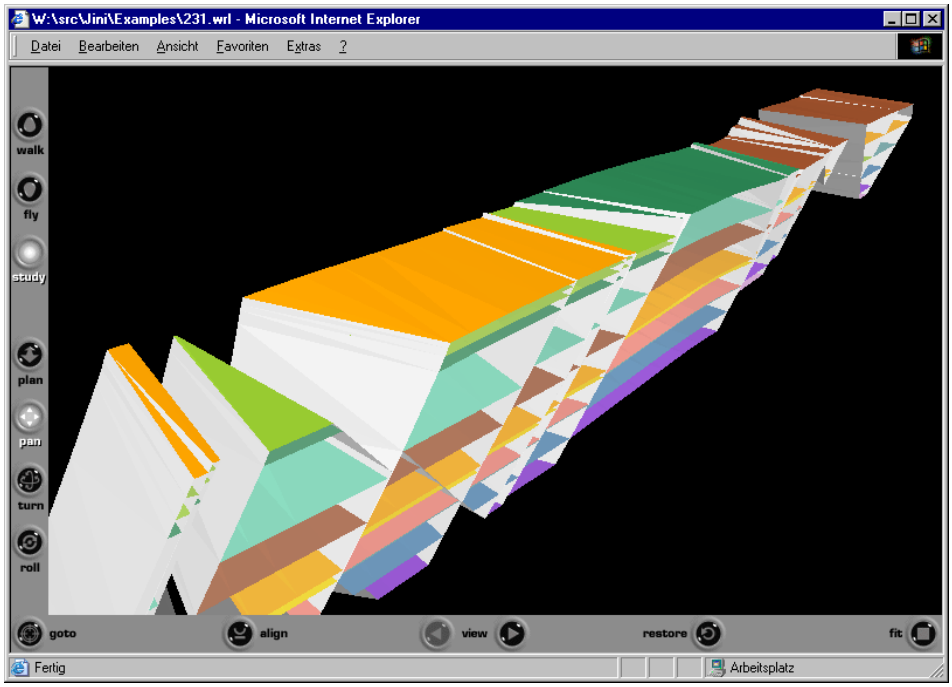


Figure 8. A result of a query visualized in a VRML browser (in our prototype Cortona™).

6.2 Time-dependent queries

When retrieving time-dependent geometry objects, the user may specify a time instance **t** or a time interval $[t_1,t_2]$ by means of two slide rulers at the left side of the query window (Figure 7). If a single time instance **t** is specified, a 3D "snapshot" of the object at time **t** will be returned. If instead a time interval $[t_1,t_2]$ is specified, a new 4D object resulting from truncation of the original 4D object is returned with $[t_1,t_2]$ as the new interval of validity. If necessary, the truncation is preceded by interpolation of the object's states at the boundaries of the new interval. Obviously **t** and t_1,t_2 must be inside of the original interval of validity.

The objects or object parts retrieved from the database are kept in the client cache for further use. By means of a commercial VRML browser (in our prototype Cortona™) Figure 8, the user can preview the selected objects before storing them, or forwarding to a geoscientific application, e.g. GOCAD for further processing.

6.3 Queries at multiple resolutions

The volume block objects of Bergheim model consist of a limited number of surfaces with few triangles. This model cannot be considered as a typical geoscientific application, where several dozens of large triangulated surfaces with up to some 100000 triangles, must be managed. This would imply a considerable amount of time to be spent on data transmission and visualization. In this case, first it is reasonable to retrieve queried objects for a small scale overview of a large area at reduced resolution, and later add detail only to those objects that are finally requested. Using a progressive transmission technique (Shumilov et al., 2002), we avoid redundancy during later retrievals of selected objects at improved level of detail. The third slide ruler (Figure 7) allows the user to specify a reduction factor ranging from 0.0 (no reduction) to 1.0 (maximal reduction) - a factor however that may not be attainable due to constraints imposed on the decimation.

7 CONCLUSION

In this paper we have introduced an approach to model 4D persistent objects for long-period geological applications. The spatio-temporal model has been embedded into GeoToolKit, an object-oriented 3D/4D database kernel system. We have demonstrated the definition of geometric range queries for spatio-temporal objects. The concepts have been evaluated within spatio-temporal database queries of a concrete geological application dealing with the balanced restoration of the Lower Rhine Basin, Germany.

The presented database operations have to be further evaluated in benchmarks with very large data sets. In our future work we intend to provide the database operations within the open GeoToolKit system architecture as part of a database service for spatio-temporal applications.

8 ACKNOWLEDGEMENTS

We thank the group of Agemar Siehl (Geological Institute) of Bonn University for many discussions on geological problems and concepts. Without their suggestions our conceptual and implementation work would have been definitively less successful. The financial support of the German Research Foundation (DFG) within the Collaborative Research Centre 350 and the Graduate Research Centre "Landform" is greatly acknowledged.

9 REFERENCES

Alms, R., Balovnev, O., Breunig, M., Cremers, A.B., Jentzsch, T. & Siehl A. (1998) Space-Time Modelling of the Lower Rhine Basin supported by an Object-Oriented Database. *Physics and Chemistry of the Earth, 23*(3), 251-260. Elsevier Science.

Balovnev, O., Breunig, M. & Cremers A.B. (1997) From GeoStore to GeoToolKit: the second step. *Proceedings 5th Intern. Symposium on Spatial Databases.* Lecture *Notes in Computer Science No.1262.* (pp. 223 – 237). Berlin, Germany: Springer.

Balovnev, O., Breunig, M., Cremers, A.B. & Pant M. (1998) Building Geo-Scientific Applications on Top of GeoToolKit: a Case Study of Data Integration. *Proceedings of the 10th Intern. Conference on Scientific and Statistical Database Management.* (pp. 260-268). Los Alamitos, CA, USA: IEEE Computer Society.

Balovnev, O., Bode, T., Breunig, M., Cremers, A.B., Müller, W., Pogodaev, G., Shumilov, S., Siebeck, J., Siehl, A., & Thomsen, A. (2004) The Story of the GeoToolKit – an Object-Oriented Geodatabase Kernel System. *GeoInformatica* 8(1)(accepted for publication).

Bohlen, M. H., Jensen, C.S., & Skjellaug, B. (1998) Spatio-Temporal Database Support for Legacy Applications. *Proceedings of the ACM Symposium on Applied Computing* (pp. 226-234). Atlanta, GA, USA.

Breunig, M. (2001) On the Way to Component-Based 3D/4D Geoinformation Systems. *Lecture Notes in Earth Sciences, No. 94.* , 199p., Heidelberg: Springer.

Breunig, M., Türker, C., Böhlen, H., Dieker, S., Güting, R.H., Jensen, C.S., Relly, L., Rigaux, P., Schek H.J. & Scholl M. (2003) Architecture and Implementation of Spatio-Temporal DBMS. In Koubarakis, M. & Sellis, T. (Eds) *Spatio-Temporal Databases – The CHOROCHRONOS Approach*. Lecture Notes in Computer Science Vol. 2520, pp. 219-264, Heidelberg: Springer.

Brinkhoff, T. (1999) Requirements of Traffic Telematics to Spatial Databases. *Proceedings of the 6th Intern. Symposium on Large Spatial Databases. Lecture Notes in Computer Science. Vol. 1651* (pp. 365-369). Hong Kong, China.

Brinkhoff, T. & Weitkämper, J. (2001) Continuous Update Queries within an Architecture for Querying XML-Represented Moving Objects. *Proceedings 7th International Symposium on Spatial and Temporal Databases (SSTD). Lecture Notes in Computer Science.* Berlin, Germany: Springer.

Cai, M., Keshwani, D. & Revesz, P.Z. (2000) Parametric Rectangles: A Model for Querying and Animation of Spatiotemporal Databases. *Proceedings of the Conference on Extending Database Technology*, *Lecture Notes in Computer Science 1777* (pp. 430-444). Berlin, Germany: Springer.

Chen, C.X & Zaniolo, C. (2000) SQL ST: A Spatio-Temporal Data Model and Query Language. *Proc. of the International Conference on Conceptual Modeling/ the Entity Relationship Approach* (pp. 96-111). Lisbon, Portugal.

Chomicki, J. & Revesz, P. Z. (1999) Constraint-based interoperability of spatiotemporal databases. *GeoInformatica*, *3*(3), 211-243.

Claramunt, C., Parent, C. & Theriault, M. (1997) Design patterns for spatio-temporal processes. In Spaccapietra, S. & Maryanski, F. (Eds.) *Searching for Semantics: Data Mining, Reverse Engineering* (pp. 415-428). New York: Chapman & Hall.

Dieker, S. & Güting, R.H. (2000) Plug and Play with Query Algebras: SECONDO—A Generic DBMS Development Environment. *Proc. of the International Database Engineering and Application Symposium* (pp. 380-392). Yokohama, Japan.

Egenhofer, M. J., Frank, A. U., & Jackson, J. P. (1990). A topological data model for spatial databases. In Buchmann, A., Günther, O., Smith, T. R. & Wang, Y.-F. (Eds.) *Proc. of the 1st Symposium SSD on Design and Implementation of Large Spatial Databases*. *Lecture Notes in Computer Science 409*. Berlin, Germany: Springer.

Forlizzi, L., Güting, R.H., Nardelli, E. & Schneider, M. (2000) A data model and data structures for moving objects databases. *Proc. of the ACM SIGMOD International Conference on Management of Data* (pp. 319-330). Dallas, Texas, USA.

Griffiths, T., Fernandes, A.A.A., Paton, N. W., Mason, K., Huang, B., Worboys, M., Johnson, C. & Stell, J. (2001) Tripod: A Comprehensive System for the Management of Spatial and Aspatial Historical Objects. *Proc. 9th ACM Int. Symposium on Advances in Geographic Information Systems.* (pp. 118-123). Atlanta, GA, USA.

Grumbach, S., Rigaux, P. & Segoufin, L. (1998) Spatio-Temporal Data Handling with Constraints. *Proc. of the ACM International Workshop on Advances in Geographic Information Systems* (pp. 106-111). Washington, DC, USA.

Güting, R. H., Bohlen, M. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M. & Vazirgiannis, M. (2000) A foundation for representing and querying moving objects. *ACM Transactions on Database Systems 25*(1), 1-42.

Polthier, K. & Rumpf, M. (1994) A concept for Time-Dependent Processes. Goebel et al., (Eds), *Visualization in Scientific Computing* (pp. 137-153). Vienna: Springer.

Sellis, T. (1999) Research Issues in Spatio-temporal Database Systems. In Güting, R.H., Papadias, D., Lochovsky, F. (Eds.). *Advances in Spatial Databases*. *Lecture Notes in Computer Science 1651*. Heidelberg: Springer.

Shumilov, S., Thomsen, A., Cremers, A.B. & Koos, B. (2002) Management and visualisation of large, complex and time-dependent 3D objects in distributed GIS. *Proc. of the 10th ACM International Symposium on Advances in Geographic Information Systems*. McLean, USA (in press).

Thomsen, A. & Siehl, A. (2002) Towards a balanced 3D kinematic model of a faulted domain - the Bergheim open pit mine, Lower Rhine Basin. *Geologie en Mijnbouw*. (pp. 251-256). Ede, the Netherlands: Veenman Drukkers.

TSurf Inc. (2002) *GOCAD 3D modeling system*. Retrieved June 7, 2003 from the TSurf Inc website: http://www.t-surf.com.

Wolfson, O. (2002) Moving Objects Information Management: The Database Challenge. *Proc. of the 5th International Workshop on Next Generation Information Technologies and Systems*. Lecture Notes in Computer Science 2382. (pp. 75-89). Heidelberg: Springer.

Worboys, M. (1994) A Unified Model for Spatial and Temporal Information. *The Computer Journal*, *37*(1), 26-34.

Yeh, T.S. & de Cambray, B. (1995) Modeling Highly Variable Spatio-Temporal Data. *Proc. of the 6th AustraliAsian Database Conference* (pp. 221-230). Glenelg/Adelaida, South Australia.

Yeh, T.S. & Feautrier, P. (1998) A unified spatial and spatiotemporal model using polyhedra. [Technical Report]. PRiSM - Laboratoire de recherche en informatique. CNRS FRE-2510. Versailles: Versailles Cedex University.