Spatio-Temporal Data Handling with Constraints

Stéphane Grumbach, Philippe Rigaux and Luc Segoufin

Abstract

Most spatial information systems are limited to a fixed dimension (generally 2) which is not extensible. On the other hand, the emerging paradigm of constraint databases allows the representation of data of arbitrary dimension, together with abstract query languages. The complexity of evaluating queries though might be costly if the dimension of the objects is really arbitrary.

In this paper, we present a data model, based on linear constraints, dedicated to the representation and manipulation of multidimensional data. In order to preserve a low complexity for query evaluation, we restrict the *orthographic dimension* of an object O, defined as the dimension of the components $O_1, ..., O_n$ such that $O = O_1 \times \cdots \times O_n$. This allows to process queries independently on each component, therefore achieving a satisfying trade-off between design simplicity, expressive power of the query language and efficiency of query evaluation.

We illustrate these concepts in the context of spatio-temporal databases where *space* and *time* are the natural components. This data model has been implemented in the DEDALE system and a spatio-temporal application, with orthographic dimension 2, is currently running, thus showing the practical relevance of the approach.

1 Introduction

With the increasing complexity of applications that store and manipulate multi-dimensional data, Database Management Systems (DBMS) are facing new and exciting challenges regarding their ability to efficiently represent and query such data, while preserving the declarative and user-friendly features of SQL-like query languages. It has been acknowledged for a long time that the relational model fails to handle multi-dimensional data with possibly infinite extensions and highly complex associated operations.

Spatial data constitute an example of multidimensional data to which a large effort has been devoted. During the past decade [18, 30] numerous research models and prototypes that favor ad hoc algebras and extensions of conventional database models have been developed. In most cases the models consist in extending relational DBMSs with abstract spatial data types (ADT) encapsulating geometric structures and operations (see for example [28, 29, 19]).

Unfortunately, due to the extensive range of applications which handle spatial data, there has not been so far a convergence towards a commonly accepted data model with well-defined data types and operations. More importantly, because of the importance of geographical data, this research is greatly influenced by the specific requirements of Geographical Information Systems (GIS) with respect to data representation and querying, and therefore limits itself to 2-dimensional data (mainly points, lines and polygons in the 2-dimensional space).

The recent field of constraint databases, initiated at the beginning of the decade [21], has lead to sound data models and query languages for multi-dimensional data [26, 17, 11, 12]. It allows to manipulate infinite relations of arbitrary dimension in a symbolic way with the formulae defining the relations. There have been many theoretical studies on constraint databases, and more recently prototypes have started to emerge. The most promising framework, that we adopt here, relies on linear constraints over a rational domain, which allow the representation of the standard geometric objects of computational geometry. Although the data complexity of querying such databases by standard means such as first-order queries has been shown to be tractable [21], it depends badly upon the dimension of the data [16].

In this paper, we present first a data model intended to represent and manipulate easily and efficiently sets of points in arbitrary dimension. The model is based on the constraint paradigm and is carefully tailored to meet the needs of practical applications. To master the complexity in the presence of high-dimensional data, we consider the *orthographic complexity* of the data, as introduced in [16].

Let us consider an example which illustrates this concept. Consider two 4-dimensional objects O and O' such that $O = O_1 \times O_2$, and $O' = O'_1 \times O'_2$, where the components O_1, O_2, O'_1, O'_2 are 2-dimensional objects. We will say that O and O' are 4-dimensional objects of orthographic dimension 2. As shown in [16], the complexity of performing queries over such objects is now reduced to a separated application of operations on the *components* of these objects. The intersection for instance satisfies $O \cap O' = (O_1 \cap O'_1) \times (O_2 \cap O'_2)$, so can be performed on 2-dimensional objects. This results in very powerful optimization of the query evaluation.

We consider here the practical application of this concept in the DEDALE system which is based on linear constraints [14]. The introduction of *components* in the data allows to control the dimension in which the manipulation is performed. This achieves a practical and satisfying trade-off between design simplicity, expressive power of the query language and efficiency of query evaluation.

We illustrate this data model with spatio-temporal applications manipulating time-evolving spatial objects: we introduce *space* and *time* as natural components of 3-d pointsets. This design results in an approximation of time evolution which is in the same spirit as the discrete geometric representation implied by linear constraints: a limited loss in accuracy is accepted in favor of computational efficiency. We show that spatio-temporal data can be manipulated at the cost of 2-dimensional data, and we obtain a data representation well supported by common graphical devices.

This work was motivated by practical spatio-temporal requirements from geographers of the French laboratory LAMA, Grenoble [4]. We demonstrate the effectiveness of our approach on a geographic application, both with respect to the query expression and to the query evaluation.

The final contribution of this work is the extension of the DEDALE prototype to handle the multi-dimensional data model. The above real-life spatio-temporal application runs with DEDALE. To the best of our knowledge, this is the first description of a practical spatio-temporal data system with sound formal foundations.

The remainder of the paper is organized as follows. Section 2 describes the data model and Section 3 defines the query language. Section 4 demonstrates the effectiveness of our approach by solving queries in a real spatio-temporal application context.

2 The Data Model

The data model relies on the constraint paradigm [21] whose basic idea is to finitely represent infinite collections of points in d-dimensional spaces. The finite representation uses *constraints* expressed in a first-order language interpreted in some arithmetical domain such as $\mathbb{Q}$ or $\mathbb{R}$. We consider here linear constraints in the first-order language $\mathcal{L} = \{\leqslant, +\} \cup \mathbb{Q}$ over the rational domain $\mathbb{Q}$ which offers a good trade-off between representational power and query complexity [12, 17], as well as equality and inequality constraints over some uninterpreted domain D.

Constraints over $\mathbb{Q}$ thus consist in equations and inequalities of the form:

$$\sum_{i=1}^{p} a_i\, x_i \Theta a_0$$

where Θ is a predicate among $=$ or $\leqslant$, the x_i's denote variables and the a_i's are integer constants. Note that rational constants can always be avoided in linear equations and inequalities. The multiplication symbol is used as an abbreviation, $a_i x_i$ stands for $x_i + \cdots + x_i$ (a_i times).

We next define the fundamental concept of linear constraint relation, the class of point sets which can be represented by a formula over $\mathcal{L}$.

Definition 1 Let $S \subseteq \mathbb{Q}^k$ be a k-ary relation. Relation S is a *linear constraint relation* if there exists a formula $\varphi(x_1, \ldots, x_k)$ in $\mathcal{L}$ with k distinct free variables $x_1, \ldots, x_k$ (called a *representation* of S) such that:

$$\mathbb{Q} \models \forall x_1 \cdots x_k (S(x_1, \ldots, x_k) \leftrightarrow \varphi(x_1, \ldots, x_k))$$

To say it in words: the set of linear constraint relations over $\mathbb{Q}^k$, denoted by *LCR*($\mathbb{Q}^k$), are the point sets which can be defined by a first-order formula in $\mathcal{L}$. The model supports a finite representation for infinite sets of points in d-dimensional space. For instance a convex polygon is simply represented as a conjunction of linear inequalities defining half-planes. It is worth noting that the dimension is not restricted. The trajectory of a moving object is a linear constraint relation in $LCR(\mathbb{Q}^3)$ which can be finitely represented as a formula over the variables x, y and t. Here is the finite representation of the trajectory in Fig. 1.

$$
\begin{array}{lr}
x = 10t \ \wedge \ y = 5t \ \wedge \ 0 \leqslant t \leqslant 1 & (a) \\
\vee & \\
y = 5t + x - 10 \ \wedge \ x = 10 \ \wedge \ 5 \leqslant y \leqslant 10 & (b) \\
\vee & \\
x = 3t + 4 \ \wedge \ 2y = 3t + 14 \ \wedge 2 \leqslant t \leqslant 4 & (c_1) \\
\vee & \\
x = 4t \ \wedge \ y = 2t + 5 \ \wedge \ 4 \leqslant t \leqslant 5 & (c_2)
\end{array}
$$

The above formula is in *disjunctive normal form* (DNF). Each conjunct represents a segment in the trajectory: for instance, the object moves from point P0 to point P1 during the time interval [0,1]. The speed for a given segment is a constant value which can be easily computed.

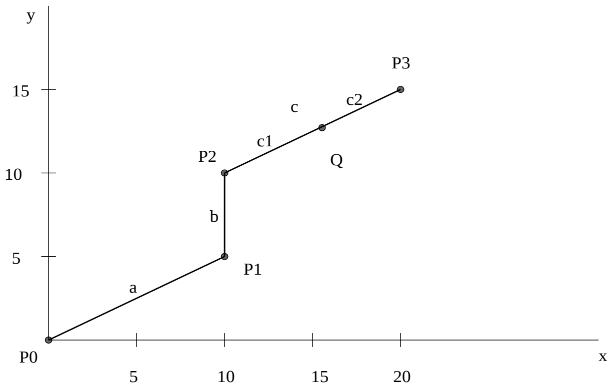

Figure 1: A trajectory

Following the now classical terminology, the formula representing the relations, when in disjunctive normal form, can be seen as a *symbolic relation*, composed of a finite set of *symbolic tuples* which are the conjuncts in the DNF representation. We often blur the distinction between the columns or attributes of a constraint relation, and the corresponding variable names.

We extend the previous classical concept of linear constraint relation and also consider relations that combine the uninterpreted domain D with the interpreted one $\mathbb{Q}$. The concept extends easily to such relations, with representation formulae in $\mathcal{L} \cup D$, with two sorts of constraints, (i) equality constraints over objects of D, and linear constraints over objects of $\mathbb{Q}$. We will denote by $LCR(D, \mathbb{Q}^2)$, the set of linear constraint relations over $D \times \mathbb{Q}^2$.

Sets of points can be manipulated via standard query languages that simulate, upon the constraints representation, the relational operations on infinite extensions [26, 17, 11]. The complexity of querying constraints depends exponentially on the number of variables involved (and thus on the dimension). This limits the practical usefulness of the model for high dimensions. We therefore introduce a restricted class of linear constraint relations with a simpler geometry which allows an upper bound on the complexity of queries.

We first recall the notion of *dependent* variables [5] with respect to a formula. Let $\varphi(x_1, \ldots, x_k)$ be a formula with k distinct free variables $x_1, \ldots, x_k$. Two variables which occur in the same linear constraint in $\varphi(x_1, \ldots, x_k)$ are said to be *dependent*. The dependency relation $\mathcal{R}$, which is the minimal equivalence relation containing the dependent variables, defines a partition $V/\mathcal{R}$ of the set of variables V, denoted the *orthographic partition* in the sequel. The *orthographic dimension* ℓ of a formula φ is the cardinality of the largest class in $V/\mathcal{R}$.

Definition 2 A linear constraint relation S is of *orthographic dimension* ℓ if there exists a representation of S with a formula φ of orthographic dimension ℓ.

Note that the orthographic dimension of a relation is not uniquely defined. We do not consider the intrinsic unique (e.g. minimal) orthographic dimension of relations, but merely, given ℓ, the relations that are of orthographic dimension at least ℓ.

To a relation, we can associate a partition of its attributes as follows.

Definition 3 A relation S admits an *orthographic decomposition* $\mathcal{P}$, if there exists a representation of S with a formula φ of orthographic partition $\mathcal{P}$. The subsets of the partition are called the *components* of the decomposition.

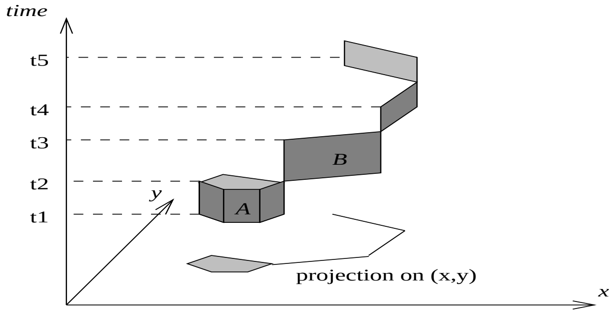

Figure 2: A spatio-temporal object with orthodim 2

Fig. 2 illustrates the trajectory of a moving object represented as a (spatio-temporal) linear constraint relation in $\mathbb{Q}^3$ of orthographic dimension 2. There are 4 tuples which associate space and time. Tuple A for instance has a geometry (a convex polygon in the plane) which gives the valid positions of the object during the interval $[t_1, t_2]$. It can be finitely represented as a conjunction of one constraint on t and 5 constraints on x and y (one for each of the 5 half-planes) with a orthographic partition $(\{t\}, \{x, y\})$. By adding disjunction, one can easily represent the evolution of the geometry (shape, position or both) with respect to time: tuple B is for instance a segment in the plane associated with the time interval $[t_2, t_3]$. Note that for each tuple T the following holds: $T \equiv \pi_{time}(T) \times \pi_{space}(T)$.

The trajectory of figure 1 can also be represented by a relation in $\mathbb{Q}^3$ of orthographic dimension 2. The main difference with the previous representation of the same trajectory is that the speed for a given segment can no longer be recovered.

$$\begin{array}{lll}
x - 2y = 0 \ \wedge \ 0 \leqslant x \leqslant 10 \ \wedge & 0 \leqslant t \leqslant 1 & \text{(A)} \\
\qquad\qquad\qquad\qquad \vee & & \\
x = 10 \ \wedge \ 5 \leqslant y \leqslant 10 \ \wedge & 1 \leqslant t \leqslant 2 & \text{(B)} \\
\qquad\qquad\qquad\qquad \vee & & \\
x - 2y + 10 = 0 \ \wedge \ 10 \leqslant x \leqslant 20 \ \wedge & 2 \leqslant t \leqslant 5 & \text{(C)}
\end{array}$$

The limitation of the orthographic dimension entails limitations on the representation power of the relations. For instance we do no longer know the speed of the object during the time interval [t1,t2], nor can we know the exact location at a given time. We could use unrestricted constraints over x, y and t to introduce stronger bounds between space and time, at the price of costlier query processing. We will show in Section 4 that this representation remains sufficiently general to model a large number of cases dealing with moving objects, or objects whose shape is changing with time.

In the following, we denote by *components* those members of nested attributes which are represented by isolated constraints. The *arity* of a component is the number of variables in its constraints. *Space* is a component of arity 2, *time* a component of arity 1.

We do not consider a unique (e.g. thiner) decomposition associated to a relation, but conversely, given a fixed decomposition, the relations that admit this decomposition.

Definition 4 The class of linear constraint relations which can be defined over the orthographic partition $((x_{D_1}, ..., x_{D_{k_0}}), (x_{q_{1,1}}, ..., x_{q_{1,k_1}}), ..., (x_{q_{i,1}}, ..., x_{q_{i,k_i}}))$ (where the variables x_D are ranging over D while the other x_i are ranging over $\mathbb{Q}$) is denoted by $LCR(D^{k_0}, \mathbb{Q}^{k_1}, \ldots, \mathbb{Q}^{k_i})$.

Within each tuple in the representation of a relation in $LCR(D^{k_0}, \mathbb{Q}^{k_1}, \ldots, \mathbb{Q}^{k_i})$, a constraint involves either the variables x_{D_i} or the variables $x_{q_{l,1}}, ..., x_{q_{l,k_l}}$ for some l.

The data model integrates the linear constraint relations into non-first normal form relations, to deal more easily with the (possibly infinite) geometric extension of objects. For simplicity the nesting is limited to one level, which suffices for spatial data. In addition, we allow to construct sets of points in the uninterpreted domain as well as in the rational domain, in order in particular to represent non-geometric time-evolving attribute.

We next introduce the data types. We assume the existence of two atomic types $\mathcal{Q}$ and U respectively the rational and uninterpreted type with domains $\mathbb{Q}$ and D.

Definition 5 *The complex types are defined as follows:*

i. $\{[A_1 : U, \ldots, A_{k_0} : U, A_{l_1} : \mathcal{Q}^{k_1}, \ldots, A_{l_i} : \mathcal{Q}^{k_i}]\}$ *(where the A_j are attribute names), denotes a set type with domain $LCR(D^{k_0}, \mathbb{Q}^{k_1}, \ldots, \mathbb{Q}^{k_i})$.*

ii. If $T_1, ..., T_n$ are atomic or set types, and $A_1, ..., A_n$ are attribute names, $[A_1 : T_1, ..., A_n : T_n]$ is a tuple type *with domain : $dom([A_1 : T_1, ..., A_n : T_n]) = \{[A_1 : a_1, ..., A_n : a_n] | a_i \in dom(T_i)\}$.*

iii. If T is a tuple type, $\{T\}$ is a relation type *with domain : $dom(\{T\}) = \wp_f(dom(T))$, where $\wp_f(S)$ denotes the set of finite subsets of S.*

A *relation schema* is a relation type. A *database schema* is a finite collection of relation types. An *instance* of a relation schema is defined as usual.

Here is an example of a relation schema for moving objects with time-varying activities which will be used in the application presented in Section 4. The atomic type *string* is used as an uninterpreted type.

$$\begin{array}{rll} PEOPLE = & \{[& \text{name} : string, \\ & & \text{category} : string, \\ & & \text{activity} : \{[\text{name} : string, \text{time} : \mathcal{Q}]\}, \\ & & \text{trajectory} : \{[\text{space} : \mathcal{Q}^2, \text{time} : \mathcal{Q}]\} \\ &]\} & \end{array}$$

In the sequel of the paper the word *relation* is used for an object of relation type, and we distinguish *relations* from *sets* of set types. The notion of orthographic dimension extends naturally to complex relations. A partition of the variables is defined according to all the formulae involved in the complex relation.

Table 1 shows an instance of the relation $PEOPLE$: *name* and *category* are represented as classical atomic values, while the nested attributes *Activity* and *Trajectory* are relations in respectively $LCR(D, \mathbb{Q})$ and $LCR(\mathbb{Q}^2, \mathbb{Q})$ which are represented as FO formulas. For example, in *trajectory*, a constraint c bounds either the first two coordinates (interpreted as x and y) or the last one (interpreted as *time*). The trajectory can be seen as a sequence of time intervals associated with the point set where the object can be found during this interval (see Fig. 2).

Name	Category	Activity	Trajectory
John	Tourist	(*name*="*Sleep*" $\wedge\ 2 < t < 10$) $\vee$ (*name*="*Eat*" $\wedge\ 11 < t < 12$) $\vee$ (*name*="*Ski*" $\wedge\ 11 < t < 15$) ...	$(x = 12 \wedge y = 13 \wedge 1 < t < 11)$ $\vee$ $(3x - 2y \leqslant 49 \wedge\ 23x + 2y \geqslant 134$ $\wedge\ 11 < t < 16)$ $\vee$ ...

Table 1: An object with nested attributes representing point sets in a multidimensional space

The data model leaves several possibilities to represent the same data. For instance we could as well have represented the activities of people by a second relation with type $\{name : string, activity : string, time; \{\mathcal{Q}\}\}$. In that case, time is factorized and *activity* is stored as a traditional value in a column of a relation while in the previous definition, it was represented with constraints in a nested attribute. We consider the proposed schema to be more natural. In addition it greatly simplifies the expression of queries since it eliminates the need for specifying tedious joins: see Section 4.

3 Query Language

We now consider a query language to manipulate the complex relations introduced above. We briefly recall the DEDALE algebra defined in [14]. The basic operations are introduced below.

- *set operations*: *union*, $\cup$, *intersection*, $\cap$, and *set difference*, $-$, apply to pairs of inputs of the same *set* or *relation* type.
- *selection*, σ_F, applies to inputs of *relation* or *set* type. F is an atomic constraint over variables corresponding to the attributes given by name or position. This constraint is either of linear form (e.g. $4X + 3Y = 2$), or a set membership constraint (e.g. $X \in S$).
- *projection*, π, applies to inputs of *set* or *relation* type.

  For objects t of tuple type, $t.i$ denotes the i'th attribute when relevant.
- *Cartesian product*, $\times$, applies to pairs of inputs of either both *set* types or both *relation* types.
- *Restructuring*, MAP applies to inputs of *relation* type. If $E(X)$ is an algebraic expression of tuple type T', with a tuple variable $X : T$, and R is a relation of type $\{T\}$, then $MAP_{\lambda X.E(X)}(R)$ defines the relation of type $\{T'\}$, in which each tuple t of R, has been replaced by $E(t)$.

The interpretation on inputs of *relation* type corresponds to the semantics of classical relational algebra over finite relations, while the interpretation of the operations on inputs of *set* type corresponds to the semantics of the operations on linear constraint relations as presented below (note that the two semantics coincide at the abstract level of potentially infinite relations). Let R_1 and R_2 be two relations, and respectively e_1 and e_2 be sets of symbolic tuples defining them:

i. $\sigma_F(R_1) = \{t_1 \wedge F \mid t_1 \in e_1\}$.

ii. $R_1 \times R_2 = \{t_1 \wedge t_2 \mid t_1 \in e_1, t_2 \in e_2\}$.

iii. $\pi_{\overline{x}}\, R_1 = \{\pi_{\overline{x}}\, t \mid t \in e_1\}$,
where

$$\pi_{\overline{x}}\, t = \bigwedge_{1 \leqslant k \leqslant K, 1 \leqslant \ell \leqslant L} b^k\overline{x} - b_0^k \leqslant a_0^\ell - a^\ell\overline{x} \;\wedge\; \bigwedge_{1 \leqslant i \leqslant I} c^i\overline{x} \leqslant c_0^i.$$

is given by the Fourier-Motzkin Elimination method [32] from a tuple t defining a polyhedron $P(\overline{x}, y) \subseteq \mathbb{Q}^{n+1}$ described by the inequalities (once the coefficients of y have been normalized):

$$\begin{cases} a^\ell\overline{x} + y \leqslant a_0^\ell & \text{for } \ell = 1, \ldots, L \\ b^k\overline{x} - y \leqslant b_0^k & \text{for } k = 1, \ldots, K \\ c^i\overline{x} \leqslant c_0^i & \text{for } i = 1, \ldots, I \end{cases}$$

where $\overline{x}$ ranges over $\mathbb{Q}^n$, and y over $\mathbb{Q}$.

iv. $R_1 \cup R_2 = e_1 \cup e_2$.

v. $R_1 - R_2 = \{t_1 \wedge t_2 \mid t_1 \in e_1, t_2 \in (e_2)^c\}$,
where e^c is the set of tuples or disjuncts of a DNF formula corresponding to $\neg e$.

As can be seen from the above definitions, the semantics of operators applied to sets of points is particular: the purpose is to simulate relational operators applied to infinite relations, and therefore to deliver a correct mathematical representation of the result that complies with the constraint representation. For selection and cross product, this is done in a somehow lazy way, by just concatenating the input(s). The result might be inconsistent or redundant: a semantic evaluation, denoted *simplification*, must be carried out at some step of the query execution process in order to eliminate redundancies and to detect inconsistencies (if a formula is unsatisfiable, the corresponding nested set is empty and the tuple must be removed).

As a simple example, consider the following *clipping query* over the relation *People*: "*give those people who crossed the rectangle Rect and the associated trajectory*". Assuming that *Rect* is represented by the constraints $\{x_{min} \leqslant x \leqslant x_{max}, y_{min} \leqslant y \leqslant y_{max}\}$, the logical expression is

$$MAP_{\lambda X.[X.name, (Rect \times X.trajectory)]}(People)$$

Since *Rect* and *People* are defined partly on the same variables, the expression $Rect \times X.trajectory$ is a (natural) join on x and y. As in the relational case, its semantics is the triplets (x, y, t) in the trajectory of *People* such that (x, y) can be also found in *Rect*. Since both extensions are infinite, this result is represented by linear constraints. Upon the instance of Table 1, this yields the relation of Table 2.

The result features a new formula for *trajectory* which represents exactly those points whose coordinates satisfy both the formula that represented the initial trajectory and the formula representing the rectangle: in other words the intersection on x and y. Note that this computation is purely symbolic:

Name	Trajectory
John	$(x = 12 \wedge y = 13 \wedge 1 < t < 11$ $\wedge\ x_{min} \leqslant x \leqslant x_{max} \wedge y_{min} \leqslant y \leqslant y_{max})$ $\vee$ $(3x - 2y \leqslant 49 \wedge 23x + 2y \geqslant 134$ $\wedge\ x_{min} \leqslant x \leqslant x_{max} \wedge y_{min} \leqslant y \leqslant y_{max})$ $\wedge\ 11 < t < 16$ ...

Table 2: The result of the *clip* query

it remains to test for emptiness (the formula might be unsatisfiable) and to extract a convenient representation depending on the required format of the output. This can be done using constraint solving operations described in [13].

In summary, any relational algebraic expression can be applied through the MAP operator to point sets: this allows to express easily intersections (spatial joins), (geometric) projections, differences, complex selections, etc. The language is abstract and general: it is uniform for alphanumerical and spatial data, and does not limit the dimension or the geometric type of objects.

4 Application to Spatio-Temporal Data

In this section, we illustrate the use of the multidimensional data model proposed in Section 2 in a real life application [4], where the spatio-temporal behavior of several types of actors in a ski resort is considered. The resort is known as a set of buildings and areas with well-defined utilities, e.g. hotels, night clubs, various kinds of stores and of course the skiing area. The typical behavior of human actors (tourists for instance) is statistically modeled with respect to their socio-economical category (age, income, nationality, ...) and represented as a sequence of activities and positions during a typical vacation day.

For instance the tourist category A will be described as the following succession of activities: {*Sleeping, Walking, Skiing, Eating, Reading, Watching Movie, etc.*}, each activity being associated with a time interval. The trajectory is described by partitioning the day in time slices and associating with each slice the position(s) where a representative of the category is likely to be found during this time slice.

The ultimate goal of the study is to improve the organization of the ski resort by detecting places where no one ever goes, equipments which are under-utilized, categories which share the same behavior, spatial distribution of tourists in the resort with respect to time and so on. The data modeling of this application is quite simple. It consists of two collections of objects:

i. A classical map (2-dimensional spatial objects) that describes the buildings and areas of interest in the ski resort.

ii. A set of moving objects, each of which represents some typical socio-economical behavior. For simplicity, we will consider only two such objects named John and Monica.

Note that moving objects contain two time-varying attributes: their activity (pure relational information) and their geometry (shape and position). Observe also that while the shape here is irrelevant since we represent each object by a single point, nothing in the model prevents us from describing complex shaped moving objects. We give below the DEDALE schema for this application:

Relation Resort
(name : **string**,
geom: (space: **float(2)**)
)

Relation People
(name : **string**
activity: (alpha: **string**,
time : **float(1)**)
traj : (space: **float(2)**,
time : **float(1)**)
)

Although extremely simple, this schema allows for a wide range of queries illustrating various combinations of spatial, temporal and relational criterias. We give a sample list of these queries in the sequel, along with some comments on either query design or its underlying evaluation in DEDALE. The same application gives rise to complex temporal relationships and queries which have been considered in the context of the TEMPOS project: see [7, 31].

The query language follows the SQL syntax to construct algebraic expressions on nested attributes in the **select** clause as well as boolean expressions on nested attributes in the **where** clause. If p_1 and p_2 denote two point set attributes (which can be either nested attributes or the results of algebraic expressions), then:

i. p_1 **join** p_2 denotes a *natural* join on the components common to p_1 and p_2.

ii. p_1 **union** p_2 denotes the union of p_1 and p_2.

iii. p_1 **except** p_2 denotes the difference of p_1 and p_2.

iv. $p_1.c[.i]$ projects p_1 on the i^{th} coordinate of component c. When there is only one component, its name can be omitted.

v. $p_1.c.i$ **as** j renames the i^{th} coordinate of c as j.

join is the most general operation: depending on the existence of common components in p_1 and p_2, one obtains a cross-product (no common components), a join (some common components), or an intersection (same components). In the latter case the equivalent **inter** keyword can be used for the sake of clarity.

The renaming of components or variables is obtained with **as** and allows to control the semantics of the **join**. Observe that the (spatial) selection is obtained as (p_1 **join** *cst*) where *cst* is any formula that represents a point set. For clarity, we sometimes use the equivalent syntax **restrict** p_1 **with** *cst*.

We can use the boolean counterpart of an operator **op**, denoted **op?**, in the **where** clause. It returns **true** if the resulting set is non empty and **false** otherwise. Therefore the general form of a query is

select $att_1, \ldots, att_n,\ E_1, \ldots E_m$
from $R_1[, ...]$
where $B_1, \ldots B_l$

where att_i denotes atomic attributes, E_j algebraic expressions over nested attributes, and B_k boolean algebraic expressions.

All queries below run in the current implementation of DEDALE.

i. *Where is John between 12 and 14?*

   select (**restrict** traj **with** '12 < time < 14').space
   from People
   where name = 'John'

   A temporal query with spatial output: the constraint '12 < time < 14' is added to the trajectory of John, and tuples in the resulting *traj* relation which are still satisfiable (if any) are projected on the *space* component.

ii. *When does Monica stay at the bar's terrace?*

   select (p.traj **join** r.geom).time
   from Resort r, People p
   where p.name = 'Monica'
   and r.name = 'Terrace'

   A spatial query with temporal output. The spatial join on the *space* component common to *t.traj* and *s.geom* yields the part of Monica's trajectory inside the terrace's geometry: its projection on *time* is the result.

iii. *Where is John while Monica is at the bar's terrace?*

   select (p2.traj **join** (p1.traj **join** r.geom).time).space
   from Resort r, People p1, People p2
   where p1.name = 'Monica'
   and p2.name = 'John'
   and r.name = 'Terrace'

This query is a composition of a spatial join and a temporal join. The internal join retrieves time t during which Monica is at the terrace; t is the input argument to the temporal join with John's trajectory.

iv. *Show the places where Monica sleeps*

select ((**restrict** activity **with** "alpha = 'Sleep' ") **join** traj).space
from People
where name = 'Monica'

Here, a pure temporal query based on alphanumerical criteria ("retrieve the periods that correspond to a given activity") is composed with a temporal join. Note that the temporal join is "internal" to a single object since it involves the two nested relations (*activity* and *traj*) of Monica.

v. *Where did Monica and John meet?*

select (p1.traj **inter** p2.traj).space
from People p1, People p2
where p1.name = 'Monica'
and p2.name = 'John'

The join involves simultaneously time and space.

vi. *When were Monica and John at the same place?*

select (p1.traj **join** r.geom).time **inter** (p2.traj **join** r.geom).time
from Resort r, People p1, People p2
where p1.name = 'Monica'
and p2.name = 'John'

Each object in the ski resort map is intersected with the trajectories of John and Monica. This yields the places where both of them spent some time during a typical day. A further join on *time* restricts the result to the places where they both were at the same instant.

vii. *Who ate in the skiing area, and when?*

select ((**restrict** p.activity **with** "alpha = 'Eat' " **join** p.traj)
join r.geom).time, p.name
from Resort r, People p
where r.name = 'Skiing Area'

A first internal temporal join between the two time-varying attributes of a moving actor gives places where this actor eats. The following spatial join checks whether these places intersect the skiing area.

viii. *What did John between the ski and his dinner?*

select name,
(((activity **join** "alpha = 'Ski' ").[time **as** t1]
join activity
join (activity **join** "alpha = 'Eat' ").[time **as** t2]
) **join** "t1.1 $\leqslant$ time.1 $\leqslant$ t2.1").alpha
from People
where name = 'John'

The comparison between three time components entails both a blow-up in dimension due to the cross-products and an unrestricted selection that links variables coming from different components. This yields an intermediate result whose global dimension is 9. Fortunately, the evaluation techniques over dimension-restricted queries allow to apply only operations on time intervals. In that case, the evaluation is as follows: $t1.1$ should be less than $Max(time.1)$ and $t2.1$ should be greater than $Min(time.1)$. The result is correct, as long as a projection of some component (in that case *alpha*) is made as a last operation, as required for dimension-restricted queries.

The description of the evaluation techniques for queries preserving the orthographic dimension is beyond the scope of this paper: the interested reader is referred to [16].

The multidimensional data model introduced in Section 2 has been implemented in the DEDALE prototype initially designed for spatial data. This work has been described in [13] where a detailed description for the various algorithms involved in constraint manipulation can be found. Fig. 3 illustrates the architecture of the prototype.

DEDALE is implemented on top of the O_2 Database Management System [1]. It relies as much as possible on O_2 existing modules, and in particular standard OQL evaluation is used for the part of queries on relational-like attributes. However, for the query evaluation mechanisms related to constraint data, DEDALE provides its own set of primitives, algorithms and access paths.

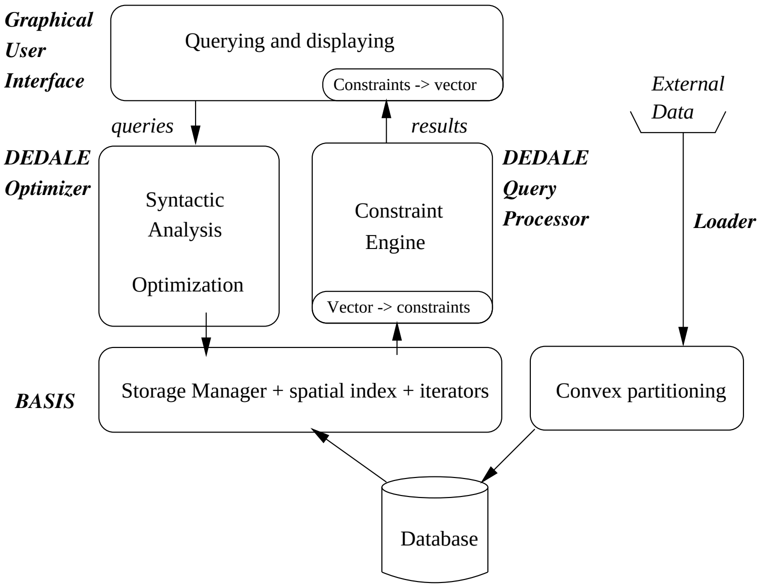

Figure 3: Architecture of DEDALE

In the current setting, alphanumerical, 1 and 2-dimensional constraints can be represented and queried. Thus unrestricted d-dimensional nested attributes can be manipulated, providing that the components are of orthographic dimension 2: relations of orthographic dimension 2 enjoy a low polynomial complexity and dimension 2 is well suited to common graphical devices. Algebraic operators are applied separately on components, with the simple following heuristics: alphanumerical and components of dimension 1 are processed first, then simplified in order to test for satisfiability. The costly operations on components of dimension 2 are then carried out on satisfiable tuples. More advanced evaluation rules are under investigation.

DEDALE has a Graphical User Interface (GUI) which allows to query the database with parameters such as points or rectangles, and to display query results including maps and other thematic data. Fig. 4 shows a screen copy of DEDALE while working on the application given by the French laboratory LAMA, Grenoble [4].

5 Discussion and Related work

The problem of spatio-temporal database management is becoming a topic of growing interest in the database community [10, 20, 34]. The applications involving moving objects, such as vehicle tracking, raise severe problems to DBMSs. In particular the representation of time, which is an evolving attribute, the combination of temporal with spatial constraints, the indexing, the uncertainty of the position, are central to the management of moving objects in databases.

The MOST (Moving Objects Spatio-Temporal) model is presented in [33]. It relies on the concept of *dynamic attributes*. Intuitively, the value of such an attribute changes over time according to a given function with no need for an explicit update. The motion information (speed + direction) of a mobile object is represented by dynamic attributes, one for each coordinate of the space embedding the trajectory of this object. Dynamic attributes need to be updated only when the motion information changes (for

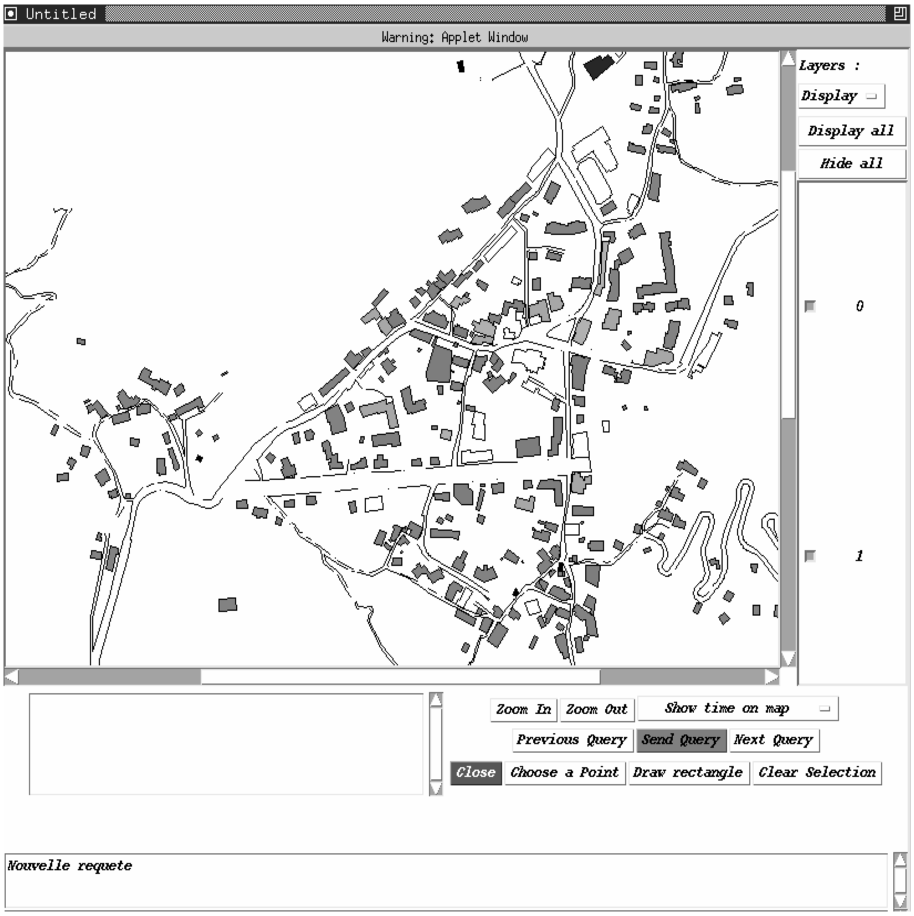

Figure 4: The application running on DEDALE

instance when a taxi takes a turn). The FTL (Future Temporal Logic) is proposed to extend existing query languages such as SQL or OQL with temporal operators. The language features also some spatial predicates which permit to express queries related to the position of moving objects.

This approach is strongly oriented towards traffic monitoring applications. An interesting characteristic is that queries about the (near) future can be formulated, such as "give the position of all the objects at time "*now + 4*".

In order to compare with our data model, it is interesting to give the constraint representation of a dynamic attribute:

$$x = \alpha_x t + \beta_x \ \wedge \ y = \alpha_y t + \beta_y \ \wedge t_{update} \leqslant t$$

This illustrates the differences between the two approaches. First a dynamic attribute consists of only *one* symbolic tuple, while our data model handles a disjunction (a set) of symbolic tuples. In other words, dynamic attributes do not store the history of updates, and queries about the past are not possible, while the constraint data model permits to represent the whole trajectory.

A second important difference is that MOST relies on a strong integration of the *space* and *time* components since the x and y variables are described with linear functions of t. This gives a more precise description of the trajectory. As a matter of fact this representation is easily captured by the constraint data model, but the properties of the orthographic dimension are lost, which means that the query evaluation process might involve 3D geometric algorithms.

An alternative approach to modeling moving objects consists to design the appropriate data types and operations [9, 8]. This extends the traditional approach of adding new types to the relational data model to handle geo-referenced objects [19]. The modeling viewpoint is similar to ours, since spatio-temporal data types are designed to represent the history of moving objects, but the approach is nevertheless quite different. Instead of considering an abstract and general query language such as first-order queries, the authors advocate the specification of a set of operations applied to instances of abstract data types which can be, for instance, *moving regions* or *moving points*. The specification mostly consists of a precise description of the signature and semantics of each operation.

An interesting discussion on the various classes of spatio-temporal applications which are of interest to the modeling of moving objects can be found in [8]. A taxonomy is presented which distinguishes applications that rely on a *stepwise constant geometry* from applications which need a more complete integration of space and time, like for instance a continuous description of an object's motion.

Clearly the modeling of moving objects proposed in the present paper captures only the first category. This can be considered as a limitation. Since a trajectory is a set of line segments, each of which is valid during a certain time interval, the exact position of the moving object during this interval is unknown. As explained above, the motivation for this approximation is clear: we can guarantee in that case that the geometric complexity depends only on the orthographic dimension, fixed to 2. This illustrates the fact that a crucial property of query languages for multidimensional data should be the independence of the complexity of query evaluation from the dimension of the embedding space. This is achieved with the concept of orthographic dimension

Nevertheless some applications may require a tighter integration of the *space* and *time* components. An interesting issue is thus to investigate whether we can relax the assumption of orthogonality. In a recent paper [15] we study the class of symbolic relations where some variables are defined as linear equations of others (such as x and y functions of t in the case of moving objects), and show that the same property of low query evaluation complexity can be obtained for a subclass of first-order queries.

Another relevant topic concerns the management of uncertainty in databases. Indeed the representation of a trajectory might be considered as loose approximation of the position of the object, while giving the position as a function of time is probably too strict since it assumes a constant speed. We would like in fact to associate some uncertainty to the location of each moving object. The importance of this issue has been acknowledged by several authors [23, 37, 27]. [37] adapts the FTL language to process "may" and "must" queries, and discusses the update policy of dynamic attributes with respect to the uncertainty. The authors of [27] consider the uncertainty of the sample positions provided by the Global Positioning System, as well as indexing and query processing issues. [23] study uncertainty in a constraints database context.

In the area of constraint databases [21], some works address the modeling of spatio-temporal data [6]. The nesting of symbolic relations in "flat" tuples has been proposed, independently from the DEDALE data model, in [2]. A practical query language based of nested constraint relations is described in [24]. Among other constraint-based prototype, it is worth mentioning C3 [3] which does not specifically address spatio-temporal applications.

As far as we know, there is no implementation of the above proposals. The most important prototype we are aware of is DOMINO [38], implemented to test the capabilities of the MOST model. Let us mention finally that the indexing of mobile objects generates a growing interest in the recent database literature [35, 22, 36, 25].

6 Conclusion

In this paper we described a data model dedicated to the representation and manipulation of multidimensional data. The main characteristics of the model are its design simplicity, its sound foundations which results in a general and non-specialized query language and some carefully chosen restrictions on the representation of data, mainly motivated by the necessity of preserving a low complexity for query evaluation. In our data model, d-dimensional objects are stored as constraints on d variables, with an upper bound on the number of variables that can occur in a single constraint. This leads to the intuitive concept of *component*, well illustrated by *time* and *space* in spatio-temporal applications. Despite the approximation which is made on the trajectory of moving objects, we show that these design choices are relevant for the classes of spatio-temporal applications that deal with stepwise constant geometries. The result is a powerful tool that allows to query a spatio-temporal database in a purely declarative way, without requiring from the end-user any geometric skill or the manipulation of a new type system. In addition, its implementation in DEDALE validates the model and proves its practical interest.

Future work will address query evaluation techniques in the presence of multi-dimensional data with an arbitrary number of components. We feel that the strong structure implied by our restricted constraints offers great potential for clever optimization of complex queries.

Acknowledgments

The authors wish to warmly thank Michel Scholl for his careful reading and valuable comments that greatly helped to improve earlier drafts of this paper. We are also indebted to the MUST group of the GDR Cassini for in-depth discussions on spatio-temporal applications.

References

[1] F. Bancilhon, C. Delobel, and P. Kanellakis, editors. *Building an Object-Oriented Database System: The Story of O_2*. Morgan Kaufmann, San Mateo, California, 1992.

[2] A. Belussi, E. Bertino, and B. Catania. An Extended Algebra for Constraint Databases. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):686–705, 1998.

[3] A. Brodsky and V.E. Segal. The c^3 constraint object-oriented database system: an overview. In *Constraint Databases and Applications, Proc. second international workshop on Constraint Databases Systems (CDB97)*, Lecture Notes in Computer Science, pages 134–159, 1997.

[4] S. Chardonnel and P. Dumolard. Personal communication.

[5] J. Chomicki, D.Q. Goldin, and G. Kuper. Variable Independence and Aggregation Closure. In *Proc. ACM Symp. on Principles of Database Systems*, pages 40–48, 1996.

[6] J. Chomicki and P. Revesz. A Geometric Framework for Specifying Spatio-temporal Objects. In *Proc. Intl. Workshop on Time Representation and Reasoning*, 1999.

[7] M. Dumas, M.-C. Fauvet, and P.-C. Scholl. Handling Temporal Grouping and Pattern-Matching Queries in a Temporal Object Model. In *Proc. Intl. Conf. on Information and Knowledge Management*, pages 424–431, 1998.

[8] M. Erwig, Güting, M. Schneider, and M. Vazirgiannis. Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. *GeoInformatica*, 3(3):269–296, 1999.

[9] M. Erwig, R.H. Güting, M. Schneider, and M. Vazirgiannis. Abstract and Discrete Modeling of Spatio-Temporal Data Types. In *Proc. Intl. Symp. on Geographic Information Systems*, pages 131–136, 1998.

[10] A.U. Frank, S. Grumbach, R. H. Güting, C.S. Jensen, M. Koubarakis, N.A. Lorentzos, Y. Manolopoulos, E. Nardelli, B. Pernici, H.-J. Schek, M. Scholl, T.K. Sellis, B. Theodoulidis, and P. Widmayer. Chorochronos: A Research Network for Spatiotemporal Database Systems. *SIGMOD Record*, 28(3):12–21, 1999.

[11] D. Goldin and P. Kanellakis. Constraint Query Algebras. *Constraints*, 1(1/2):45–83, 1996.

[12] S. Grumbach and G. Kuper. Tractable Recursion over Geometric Data. In *International Conference on Constraint Programming*, pages 450–462, 1997.

[13] S. Grumbach, P. Rigaux, M. Scholl, and L. Segoufin. DEDALE: A Spatial Constraint Database. In *Proc. Intl. Workshop on Database Programming Languages*, pages 38–59, 1997.

[14] S. Grumbach, P. Rigaux, and L. Segoufin. The DEDALE System for Complex Spatial Queries. In *Proc. ACM SIGMOD Symp. on the Management of Data*, pages 213–224, 1998.

[15] S. Grumbach, P. Rigaux, and L. Segoufin. Manipulating Interpolated Data is Easier than you Thought. In *submitted*, 1999.

[16] S. Grumbach, P. Rigaux, and L. Segoufin. On the Orthographic Dimension of Constraint Databases. In *Proc. Intl. Conf. on Database Theory*, pages 199–216, 1999.

[17] S. Grumbach, J. Su, and C. Tollu. Linear Constraint Query Languages: Expressive Power and Complexity. In D. Leivant, editor, *Logic and Computational Complexity*, Indianapolis, 1994. Springer Verlag. LNCS 960.

[18] R.H. Güting. An Introduction to Spatial Database Systems. *The VLDB Journal*, 3(4):357–399, 1994.

[19] R.H. Güting and M. Schneider. Realm-Based Spatial Data Types: The ROSE Algebra. *The VLDB Journal*, 4(3):243–286, 1995.

[20] C. Jensen and M. Scholl, editors. *Proc. of the VLDB workshop on Spatio-Temporal Database Management*, Edinburgh (Scotland), September 1999.

[21] P. Kanellakis, G Kuper, and P. Revesz. Constraint Query Languages. *Journal of Computer and System Sciences*, 51(1):26–52, 1995. A shorter version appeared in PODS'90.

[22] G. Kollios, D. Gunopolos, and V.J. Tsotras. On Indexing Mobile Objects. In *Proc. ACM Symp. on Principles of Database Systems*, pages 261–272, 1999.

[23] M. Koubarakis. The Complexity of Query Evaluation in Indefinite Temporal Constraint Databases. *Theoretical Computer Science*, 171(1/2), 1997.

[24] G. Kuper, S. Ramaswamy, K. Shim, and J. Su. A Constraint-Based Spatial Extension to SQL. In *Proc. Intl. Symp. on Geographic Information Systems*, 1998.

[25] M.A. Nascimento, J.R.O. Silva, and Y. Theodoridis. Evaluation of Access Structures for Discretely Moving Points. In *Intl. Workshop on Spatio-Temporal Database Management (STDBM'99)*, LNCS 1678, 1999.

[26] J. Paredaens, J. Van den Bussche, and D. Van Gucht. Towards a Theory of Spatial Database Queries. In *Proc. 13th ACM Symp. on Principles of Database Systems*, pages 279–288, 1994.

[27] D. Pfoser and C.S. Jensen. Capturing the Uncertainty of Moving-Object Representations. In *Proc. Intl. Conf. on Large Spatial Databases (SSD)*, pages 111–132, 1999.

[28] N. Roussopoulos, C. Faloutsos, and T. Sellis. An Efficient Pictorial Database System for PSQL. *IEEE Transactions on Software Engineering*, 14(5):639–650, 1988.

[29] M. Scholl and A. Voisard. Thematic Map Modeling. In *Proc. Intl. Symp. on Large Spatial Databases (SSD)*, LNCS No. 409, pages 167–192. Springer-Verlag, 1989.

[30] M. Scholl and A. Voisard, editors. *Proc. Intl. Symp. on Large Spatial Databases (SSD).* LNCS No. 1262. Springer-Verlag, 1997.

[31] P.-C. Scholl, M.-C. Fauvet, and J.-F. Canavaggio. Un Modèle d'Historique pour un SGBD Temporel. *TSI*, 17(3), mars 1998.

[32] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1986.

[33] A. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and Querying Moving Objects. In *Proc. IEEE Intl. Conf. on Data Engineering (ICDE)*, pages 422–433, 1997.

[34] S. Spacapietra, editor. *Proc. of the DEXA Workshop on spatio-temporal data models and languages*, Firenze (Italy), September 1999. IEEE Computer Society.

[35] Y. Theodoridis, T. K. Sellis, A. Papadopoulos, and Y. Manolopoulos. Specifications for Efficient Indexing in Spatiotemporal Databases. In *Intl. Conf. on Scientific and Statistical Database Management*, 1998.

[36] Y. Theodoridis, J. R. O. Silva, and M. A. Nascimento. On the Generation of Spatiotemporal Datasets. In *Intl. Conf. on Large Spatial Databases (SSD'99)*, 1999.

[37] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez. Cost and Imprecision in Modeling the Position of Moving Objects. In *Proc. IEEE Intl. Conf. on Data Engineering (ICDE)*, pages 588–596, 1998.

[38] O. Wolfson, A. P. Sistla, B. Xu, J. Zhou, and S. Chamberlain. DOMINO: Databases fOr MovINg Objects tracking. In *Proc. ACM SIGMOD Symp. on the Management of Data*, pages 547–549, 1999.