# Efficient Multi-Dimensional Data Handling in Constraint Databases*

Stéphane Grumbach
IASI and INRIA

Philippe Rigaux
CNAM

Luc Segoufin
INRIA

**Résumé**

Un des principaux avantages des bases de données contraintes sur les systèmes de gestion de données spatiales traditionnels est la possibilité de manipuler uniformément des données de dimension arbitraire. Le coût de cette manipulation augmente cependant considérablement pour des dimensions élevées.

Nous introduisons dans cet article le concept de *dimension orthographique* d'un objet $O$ correspondant à la dimension maximale des objets composants $O_1, ..., O_n$ tels que $O = O_1 \times \cdots \times O_n$. Nous montrons que pour une classe de requêtes bien définie, l'évaluation peut s'effectuer indépendamment sur chaque composant. Ce résultat montre que l'on peut manipuler des données géométriques dans un espace de dimension arbitraire avec une complexité similaire à celle obtenue en dimension 2 ou 3.

Nous avons implanté cette technique dans le prototype DEDALE, pour une dimension orthographique fixée à 2. Nous montrons l'intérêt pratique de ce modèle sur une application spatio-temporelle, où les composants naturels sont *l'espace* et *le temps*.
**Mots-clés :** Bases de données contraintes, données spatio-temporelles et multi-dimensionnelles, complexité des langages de requêtes.

**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. The complexity of evaluating queries though might be costly if the dimension of the objects is really arbitrary.

In order to preserve a low complexity for query evaluation, we introduce the structural dimension of an object $O$, 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.

This data model has been implemented in the DEDALE system. We illustrate its practical interest in the context of spatio-temporal databases where *space* and *time* are the natural components.
**Keywords :** Constraints Databases, Spatio-temporal and multidimensional data, complexity issues in query languages.

*Work supported in part by the ESPRIT TMR project Chorochronos and in part by the french CNRS GDR CASSINI.

0. INRIA, Rocquencourt BP 105, F-78153 Le Chesnay, France — {Stephane.Grumbach,Luc.Segoufin}@inria.fr

0. Cedric/CNAM, 292 rue St Martin, F-75141 Paris Cedex 03, France — rigaux@cnam.fr

# 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 data model fail 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 [Güt94, SV97] 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 [RFS88, SV89, GS95]).

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 [KKR90], has lead to sound data models and query languages for multi-dimensional data [PVV94, GST94, KG94, GK97]. 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, rely on linear constraints over a rational domain, which allow the representation of the standard geometric objects of computational geometry. Although the complexity of querying such databases by standard means such as first-order queries has been shown to be tractable, it depends badly upon the dimension of the data.

To master the complexity, we suggest to consider the orthographic complexity of the data. 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 $O_1, O_2, O'_1, O'_2$ are 2-dimensional objects. We will say that $O$ and $O'$ are 4-dimensional objects of orthographic complexity 2. Now let us illustrate the problem of the complexity of performing operations on $O$ and $O'$. 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. The same holds for the projection for instance. Suppose we want to project out a dimension, it is sufficient to perform it on the constituent which contains that dimension. This results in very powerful optimization of the query evaluation.

The orthographic dimension of the data captures the maximal dimension in which the manipulation can be performed. For a class of queries which preserve the orthographic dimension, we develop an evaluation technique which permits an efficient evaluation. The model therefore achieves a satisfying trade-off between design simplicity, expressive power of the query language and efficiency of query evaluation.

We next consider the practical application of this concept in the DEDALE system which is based on linear constraints [GRS98b]. The introduction of *components* in the data model

allows to control the dimension in which the manipulation is performed.

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 lost 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 [CD98]. A 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 the query language. In Section 3, the orthographic dimension is introduced, and an efficient evaluation technique for a class of queries preserving the orthographic decomposition is proposed. Section 4 demonstrates the effectiveness of our approach by solving queries in a real spatio-temporal application context, and discusses its implementation in DEDALE.

# 2 The data model

In this section we describe the data model, which extends the work presented in [GRS98b].

## 2.1 Data representation

The data model relies on the constraint paradigm [KKR90] 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 [GST94, GK97], as well as equality and inequality constraints over some uninterpreted domain $D$.

Constraints over $\mathbb{Q}$ thus consists in linear equations and inequalities of the form: $\sum_{i=1}^{p} a_i\, x_i \Theta a_0$, where $\Theta$ 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.

**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:

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

We denote by $LCR(\mathbb{Q}^k)$ the set of linear constraint relations over $\mathbb{Q}^k$. Following the now classical terminology, the formula representing a relation, when in disjunctive normal form

(DNF), can be seen as a *generalized relation*, composed of a finite set of *generalized tuples* which are the conjuncts in the DNF. 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 $D \times \mathbb{Q}^2$.

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 inequations defining half-planes (note that the linear data model implies an approximation of the actual geometric representation of most spatial objects).

Sets of points can be manipulated via standard query languages that simulate, upon the constraints representation, the relational operations on infinite extensions [PVV94, GST94, KG94].

The complexity of querying constraint databases is strongly related to constraint solving complexity. As we will see in more details, this complexity exponentially depends on the number of variables involved. 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 [CGK96] 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* $\ell$ of a formula $\varphi$ is the cardinality of the larger class in $V/\mathcal{R}$.

**Definition 2** A linear constraint relation $S$ is of *orthographic dimension* $\ell$ if there exists a representation of $S$ with a formula $\varphi$ of orthographic dimension $\ell$.

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 $\ell$, the relations that are of orthographic dimension at least $\ell$.

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

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

**Example 1** Figure 1 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)

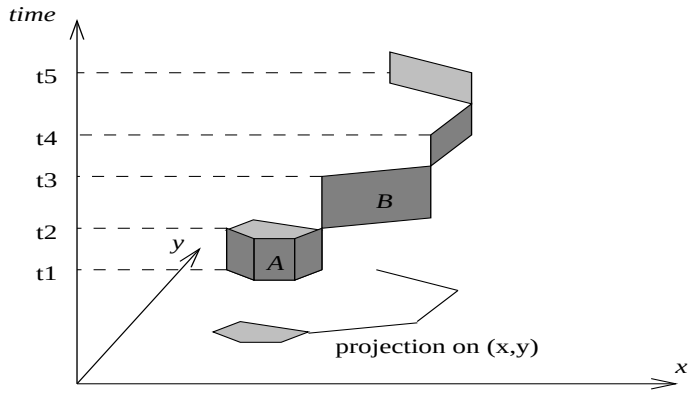


Figure 1: A spatio-temporal object

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: $T \equiv \pi_{space}(T) \times \pi_{time}(T)$.

We do not consider a unique (e.g. thiner) decomposition associated to a relation, but conversely, given a fix 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 variable $x_D$ 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. The necessity of nested relations for spatial data has already been advocated in [BBC97, GRS98b]. 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 attributes.

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:*

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

2. *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 : $dom([A_1 : T_1, ..., A_n : T_n]) = \{[A_1 : a_1, ..., A_n : a_n] | a_i \in dom(T_i)\}$.*

3. *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 $S$.*

| Name | Category | Activity | Trajectory |
|---|---|---|---|
| John | Tourist | $(name="Sleep" \wedge 2 < t < 10)$ | $(x = 1230 \wedge y = 134 \wedge 1 < t < 11)$ |
| | | $\vee\ (name="Eat" \wedge 11 < t < 12)$ | $\vee (3x - 2y \leqslant 49 \wedge 23x + 2y \geqslant 134 \ldots$ |
| | | $\vee\ (name="Ski" \wedge 11 < t < 15)$ | $\wedge 11 < t < 16)$ |
| | | $\ldots$ | $\ldots$ |

Figure 2: An object with nested attributes representing pointsets in a multidimensional space

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.

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

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 decomposition extends naturally to complex relations. A partition of the variables is defined according to all the formulae involved in the complex relation.

**Example 2** Figure 2 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 pointset where the object can be found during this interval (see Fig. 1).

The limitation of the orthographic dimension entails limitations on the representation power of the relations. In spatio-temporal applications for instance, the trajectory of a moving object has a discrete representation in the database: its position cannot be represented as a function of time. This is analogous to the approximation of geometric shapes with linear constraints. We show however 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.

## 2.2 Query language

We now consider a query language to manipulate the complex relations introduced above. We recall the DEDALE algebra defined in [GRS98b]. 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*, $\sigma_F$, 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*, $\pi$, 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 generalized tuples defining them[1].

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

2. $\pi_{\overline{x}}\, R_1 = \{\pi_{\overline{x}}\, t \mid t \in e_1\}$,
   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 [Sch86] 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 $\mathbb{Q}$.

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

4. $R_1 - R_2 = \{t_1 \wedge t_2 \mid t_1 \in e_1, t_2 \in (e_2)^c\}$,
   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

[1] A more complete description of the algebra is to be found in [GRS98b].

| Name | Trajectory |
|---|---|
| John | $(x = 1230 \wedge y = 134 \wedge 1 < t < 11 \wedge$ $x_{min} \leqslant x \leqslant x_{max} \wedge y_{min} \leqslant y \leqslant y_{max})$ $\vee (3x - 2y \leqslant 49 \wedge 23x + 2y \geqslant 134 \ldots$ $x_{min} \leqslant x \leqslant x_{max} \wedge y_{min} \leqslant y \leqslant y_{max})$ $\wedge 11 < t < 16$ $\ldots$ |

Figure 3: The result of $MAP_{\lambda X[X.name,(Rect \times X.trajectory)]}(People)$

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).

**Example 3** 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 Figure 2, this yields the relation of Figure 3.

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: 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 [GRSS97].

In summary, any relational algebraic expression can be applied through the $MAP$ operator to pointsets: 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.

The complexity of the algebraic operators is studied in [GRS98a]. The complexity of normalization and simplification (and therefore satisfaction) of $n$ linear constraints in dimension $d$ can be found in [Sch86, GO97]. It is essentially exponential in the dimension. Using the Fourier-Motzkin algorithm, one can eliminate one variable of a convex set of $n$ facets in dimension $d$ in time complexity of $n^2$. If $d$ variables needs to be eliminated, the time complexity is then $n^{2^d}$. A more subtle algorithm would be to *simplify* after eliminating each variable, then to reduce the output to something linear in $n$. The overall complexity to eliminate $d$ variables is then : $O((n^2 + 2^{2^d}n) + ... + (n^2 + 2^{2^d}n))$ which is $O(d2^{2^d}n^2)$.

Figure 4 summarizes the costs of the algebraic operators in dimension 1, 2, 3 and $d$ with respect to the following parameters: $n$ is the total number of constraints in the relations, $t$ is the number of tuples, and $k$ the number of variables projected out. All complexities are given modulo a coefficient factor. The blow up in complexity comes from projection, set difference, and simplification.

| Dimension $\rightarrow$ Operator $\downarrow$ | 1 | 2 | 3 | d |
|---|---|---|---|---|
| $\times$ | $t^2$ | $t^2$ | $t^2$ | $t^2$ |
| $\cup$ | 1 | 1 | 1 | 1 |
| $\pi_x$ | $n$ | $n \log n$ | $n^2$ | $k2^{2^k} n^2$ |
| $\sigma_F$ | $t$ | $t$ | $t$ | $t$ |
| $-$ | $n \log n$ | $n^2$ | $n^4$ | $n^{d+1}$ |
| $simplify$ | $n$ | $n$ | $n$ | $(2^{2^d})n$ |

Figure 4: Complexity of the operators in the dimension

The previous results suggest that it is highly desirable to limit the query evaluation process to physical operations that manipulate only pointsets whose dimension is low (less than 3). This is investigated in the next section.

# 3 Query evaluation

In order to allow an efficient evaluation, a first possible restriction is to consider $d$-dimensional databases which consist only of relations with a bounded orthographic dimension $\ell \leqslant d$ (typically 2 or 3). The first natural question which then arises is whether we can define a class of queries that can be computed in closed form over such databases, that is queries preserving the orthographic dimension. Furthermore, if such a class exists, one must show that the evaluation of queries belonging to this class takes advantage of the orthographic dimension of the input.

In this section we first introduce the natural concept of queries preserving the *orthographic dimension*. Intuitively, this property states that a query preserves the initial dependencies between the attributes in the input and the global orthographic dimension. We then give a syntactic restriction which respects this property, and show that these queries can be efficiently processed by using only operations over low-dimensional pointsets. The reader is referred to [GRS98a] for a formal presentation.

**Definition 6** Let $s$ be a database schema, and $\mathcal{P}$ be an orthographic decomposition of the relations in $s$. A query $Q$ over $s$ *preserves the orthographic decomposition* $\mathcal{P}$ if for each instance $I$ of orthographic decomposition $\mathcal{P}$, $q(I)$ admits a orthographic decomposition which refines $\mathcal{P}$[2].

Note that the definition does not restrict the constraints in the query $q$: therefore intermediate results which are not of orthographic dimension $\ell$ are allowed. This gives more

[2]A partition $P_1$ is a refinement of $P_2$ if each element of $P_1$ is fully contained in an element of $P_2$.

flexibility to the query language, yet preserving nice properties for query evaluation, as shown in the rest of this section. The following examples illustrate this concept.

**Example 4** Let $R : \{[name : string, nested : \{A : \mathcal{Q}^2, B : \mathcal{Q}^2\}]\}$ be a relation. Consider the following queries:

1. $q \equiv MAP_{\lambda X.[X.name, \sigma_{A.1<B.1}(X.nested)]}(R)$
2. $q' \equiv MAP_{\lambda X.[X.name, \pi_{A.1,B.1}(\sigma_{A.1<B.1}(X.nested))]}(R)$
3. $q'' \equiv MAP_{\lambda X.[X.name, \pi_A(\sigma_{A.1<B.1}(X.nested)), \pi_B(\sigma_{A.1<B.1}(X.nested))]}(R)$

Query $q$ does not preserve the orthographic decomposition since one can easily find an instance $I$ of $R$ of orthographic dimension 2 such that $q(I)$ is of orthographic dimension 4 (with the variables defining $A$ and $B$ no longer independent).

For query $q'$, observe that the result is of orthographic dimension 2, thanks to the projection $\pi_{A.1,B.1}$. But $q'$ does not preserve the orthographic decomposition since the constraint $A.1 < B.1$ which occurs in $q'(I)$ breaks the independence between $A$ and $B$.

Finally, $q''$ preserves the orthographic decomposition, although it contains intermediate results which are of orthographic dimension 4.

As other preservation properties, the preservation of the orthographic decomposition is not decidable. We therefore introduce a class of queries which captures only a subset of orthographic decomposition preserving queries but which enjoys the following properties: (i) it is syntactic, (ii) included in the orthographic decomposition preserving queries, and (iii) can be evaluated efficiently. Basically, it consists in checking that when a selection introduces a binding between independent variables, this binding is further eliminated through a projection.

Let $E$ be an algebraic expression. We recursively define the collection of bad bindings, as the set $BB(E)$ of pairs of components which are linked at each step of the computation process.

1. If $E$ is a relation name or a variable, $BB(E) = \emptyset$.
2. If $E = [E_1, ..., E_n]$, $BB(E) = BB(E_1) \cup \cdots \cup BB(E_n)$.
3. If $E = \sigma_F(E_1)$, then $BB(E)$ is the union of $BB(E_1)$ with the collection of pairs $< C_i, C_j >$ such that $x$ is a variable of $C_i$, $y$ is a variable of $C_j$ and $x$ and $y$ are bound in $F$.
4. If $E = \pi_{x_{i_1},...,x_{i_n}}(E_1)$, then $BB(E)$ is obtained by removing from $BB(E_1)$ all pairs $p$ such that one of the components in $p$ is only defined with variables that don't appear in $\{x_{i_1}, \ldots, x_{i_n}\}$.
5. If $E = E_1 \Theta E_2$ with $\Theta$ among $\{\cup, \cap, -, \times\}$, then $BB(E) = BB(E_1) \cup BB(E_2)$.
6. If $E = MAP_{\lambda X.E_1(X)}$, $BB(E) = \emptyset$.

Now if $E$ is an algebraic expression such that $BB(E) = \emptyset$, $E$ is said to be bad-binding free. For instance, for the expression $E = \sigma_{A.1<B.1}(X.nested)$ in query $q$ above, $BB(E) = \{< A, B >\}$. For $E = \pi_A(\sigma_{A.1<B.1}(X.nested))$ in query $q''$, $BB(E) = \emptyset$.

We can now introduce the syntactic restriction on queries.

**Definition 7** *$\mathcal{P}$-Safe* queries are defined with all the operators of the language with the restriction that whenever a $MAP_{\lambda X.E(X)}$ is used, $E(X)$ should be bad-binding free.

Query $q''$ in Example 4 is *$\mathcal{P}$-Safe* while $q$ and $q'$ are not. It is easy to show that every *$\mathcal{P}$-safe* query preserves the orthographic decomposition. The natural question is whether this restriction captures all the orthographic decomposition preserving queries. It is shown in [GRS98a] that every Boolean Combination of Conjunctive Queries which preserves the orthographic decomposition $\mathcal{P}$ is equivalent to a *$\mathcal{P}$-safe* query. We denote by SBCCQ this class of queries in the sequel.

We investigate now the evaluation of *$\mathcal{P}$-safe* queries over $d$-dimensional instances of orthographic dimension $\ell$. Since we allow intermediate results of higher orthographic dimension, one might expect that this evaluation may involve operators on $d$-dimensional data. Fortunately, we exhibit in the following a strategy which guarantees that each operator actually applies to pointsets whose dimension is bounded by $\ell$.

First observe that for relations of orthographic dimension $\ell$, it is possible to apply each operator except selection independently on each component. For instance, the intersection of two objects composed of components can be processed by composing, *at the tuple's level*, the intersection of the corresponding components. The same holds for union and set difference. Cartesian product does not affect the components. Projection applies to the appropriate components.

In the case of selection on the other hand, the result might not be of orthographic dimension $\ell$ any more (see Example 4). However, decomposition-preserving queries can be transformed so as to involve only operations in dimension $\ell$: The peculiar form of these queries which guarantees that a selection $s$ over separated components is followed by a projection implies that an *approximate evaluation* of $s$ is sufficient. We can then replace such selections by simpler expressions which respect the orthographic decomposition.

We thus introduce a new selection operator which preserves the orthographic decomposition, and can be used in place of the classical selection. The new symbolic selection, denoted by $\tilde{\sigma}_F$, computes, for each generalized tuple $t$, the projection $\pi_{C_i}(F \wedge t)$ for each component $C_i$ and returns $\tilde{\sigma}_F(t) = \pi_{C_1}(F \wedge t) \times \ldots \times \pi_{C_n}(F \wedge t)$. Note that $\tilde{\sigma}_F(t)$ is not equal to $\sigma_F(t)$. It defines an approximation of $\sigma_F(t)$, such that the two coincide on each component. We show how this new selection can be used in place of the traditional one.

$R$

$$
\begin{array}{|c|}\hline
x_1 + x_2 \leqslant 0 \\
x_1 - 2x_2 + 5 \geqslant 0 \\
\vdots \\
\wedge \\
x_3 + x_4 - 6 \leqslant 0 \\
x_3 - 4x_4 + 18 \leqslant 0 \\
\vdots \\ \hline
\end{array}
$$

$\sigma_{x_4 \leqslant x_1 \leqslant x_3}(R)$

$$
\begin{array}{|c|}\hline
x_1 + x_2 \leqslant 0 \\
x_1 - 2x_2 + 5 \geqslant 0 \\
\vdots \\
\wedge \\
x_3 + x_4 - 6 \leqslant 0 \\
x_3 - 4x_4 + 18 \leqslant 0 \\
\vdots \\
\wedge \\
x_4 \leqslant x_1 \leqslant x_3 \\ \hline
\end{array}
$$

$\tilde{\sigma}_{x_4 \leqslant x_1 \leqslant x_3}(R)$

$$
\begin{array}{|c|}\hline
x_1 + x_2 \leqslant 0 \\
x_1 - 2x_2 + 5 \geqslant 0 \\
\vdots \\
x_1 \leqslant \alpha \\
\alpha' \leqslant x_1 \\
\wedge \\
x_3 + x_4 - 6 \leqslant 0 \\
x_3 - 4x_4 + 18 \leqslant 0 \\
\vdots \\
\beta \leqslant x_3 \\
x_4 \leqslant \beta' \\
x_4 \leqslant x_3 \\ \hline
\end{array}
$$

where

$$\alpha = Sup_{x_3}\{x_3 + x_4 - 6 \leqslant 0, x_3 - 4x_4 + 18 \leqslant 0, ...\}$$

$$\alpha' = Inf_{x_4}\{x_3 + x_4 - 6 \leqslant 0, x_3 - 4x_4 + 18 \leqslant 0, ...\}$$

$$\beta = Inf_{x_1}\{x_1 + x_2 \leqslant 0, x_1 - 2x_2 + 5 \geqslant 0, ...\}$$

$$\beta' = Sup_{x_1}\{x_1 + x_2 \leqslant 0, x_1 - 2x_2 + 5 \geqslant 0, ...\}$$

It is easy to see that $\sigma_{x_4 \leqslant x_1 \leqslant x_3}(R)$ and $\tilde{\sigma}_{x_4 \leqslant x_1 \leqslant x_3}(R)$ will yield the same result after projecting on the components $A = \{x_1, x_2\}$ and $B = \{x_3, x_4\}$. Intuitively, a query $\sigma_{x \leqslant y}$ with $x$ and $y$ belonging respectively to $A$ and $B$ can be transformed by adding to each tuple $t$ the constraints $A.x \leqslant y_{max}$ and $B.y \geqslant x_{min}$: The rewriting process takes advantage of the bounding boxes of each component which can either be computed on the fly, or stored in the database. Note that this strategy is only valid for $\mathcal{P}$-*safe* queries since its correctness depends heavily on the final projections on $A$ and $B$.

Consider now the more general example of a simple conjunctive $\mathcal{P}$-*safe* query $q(R) = \pi_z(\sigma_F(R))$, in the context of a relation of dimension 3 and orthographic dimension 2, with $\mathcal{P} = (\{x, y\}, \{z\})$. We focus on the evaluation of $q$ on a single tuple, $O$ (see Figure 5.a).

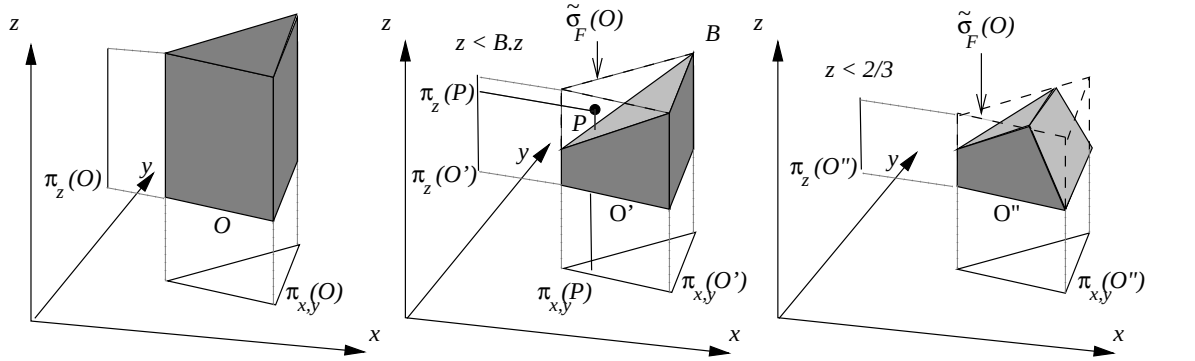


Figure 5: Selection over a 3-d object of orthographic dimension 2

In Figure 5.b: the selection $\sigma_F(O)$, with $F = x > z$, yields a polytope $O'$. The exact value of $O'$ is not relevant for the final output of the query which is $\pi_z(O')$: $O'$ can be approximated by $\tilde{\sigma}_F(O)$ (in dotted lines on Figure 5.b). As previously, the following comments hold. First, the result features a new constraint $z < \alpha$ where $\alpha$ is the highest coordinate of $O'$, $B.z$: importantly, the point $B$ can be simply computed using $F$ and the bounding-box of $O$. Second, a point in $\tilde{\sigma}_F(O)$, $P$ for instance, might not belong to $O'$, but its projections on $\{x, y\}$ and $\{z\}$ belong to $\pi_{x,y}(O')$ and $\pi_z(O')$ respectively. In other words, we have:

$$\pi_z(\sigma_F(O)) = \pi_z(\tilde{\sigma}_F(O)) = \pi_z(\pi_{x,y}(F \wedge O) \times \pi_z(F \wedge O))$$

In Figure 5.c, $F = x > z \wedge 2x + z < 2$ consists of two constraints. The same technique applies: we approximate the result of $\sigma_F(O)$ by the cross-product of its components on $\{x, y\}$ and $\{z\}$. The new constraint in the final result is here $z < 2/3$, a constant value which cannot be obtained from the bounding-box, but results from the intersection of the half-planes $x > z$

and $2x + z < 2$. Note that it depends on the query *and not on the database* and can thus be computed only once.

In summary, (1) an approximate evaluation of $\sigma_F$, $\tilde{\sigma}_F$, which keeps the exact information regarding each component, and **not** the temporary link between two components. is sufficient in the case of $\mathcal{P}$-$safe$ queries, and (2) $\tilde{\sigma}_F$ can be efficiently computed by using the bounding box of each tuple.

The full strategy, generalized to any dimension, for evaluating $\tilde{\sigma}_F$ over an instance $I$ with components $\{C_1, C_2, \ldots, C_n\}$, is as follows:

1. **Step 1** First compute $S^1$, the constraints which depends only on the query: one projects $F$, using the Fourier-Motzkin algorithm, on each component. Although this requires an operations over non-restricted data, this operation is done on the query and not on the database. For instance, in the example of Figure 5.c, $z < 2/3$ is obtained by applying Fourier-Motzkin on $F = x > z \wedge 2x + z < 2$.

2. **Step 2.a** For each tuple $t$ in $I$, compute the constraints $S^2$, in time $O(|F|)$ for each component $C_i$ as follows:

   (a) For each constraint $cst$ in $F$, put $cst$ in the form $f(x_1^i, \ldots x_l^i)\ \Theta\ \Phi((x_1^j, \ldots x_m^j)$ where $f$ and $\Phi$ are linear functions, the $x^i$ are the variables of $C_i$ and $\Theta \in \{\leqslant, \geqslant\}$.

   (b) If $\Theta$ is $\geqslant$ (resp. $\leqslant$), compute the local minimum (resp. maximum) $L$ of $\Phi$ using the values of $mbb(t)$,

   (c) Output the constraint $f(x_1^i, \ldots x_l^i)\Theta L$.

3. **Step 2.b** Finally, construct $\tilde{\sigma}_F(t)$ as $t \wedge S^1 \wedge S^2$.

It was shown in [GRS98a] that this algorithm yields the expected approximation, $\tilde{\sigma}_F$. We now define $ALG^\ell$ as those queries which can be expressed by dimension-preserving operators on $\ell$-dimensional poinsets.

**Definition 8** $ALG^\ell$ is the class of queries which can be expressed by the following operations:

- $\cup$, $\times$
- $\pi^\ell$, $-^\ell$, and $simplify^\ell$ which are the usual projection, simplification and negation but restricted to inputs of dimension $\ell$.
- $\tilde{\sigma}_F$.

Every query in $ALG^\ell$ over an instance of orthographic dimension $\ell$ can be evaluated in parallel on each components of the orthographic decomposition $\mathcal{P}$ since each operator in $ALG^\ell$ preserves $\mathcal{P}$. Further more, the complexity of evaluating such queries depends only on $\ell$. It remains to characterize the class of queries over databases with od $\ell$ which is captured by $ALG^\ell$. We can state the following fundamental result.

**Theorem 1** Let $s$ be a database schema of orthographic decomposition $\mathcal{P}$. Let $q$ be a $\mathcal{P}$-safe BCCQ over $s$. Then $q$ can be rewritten in an equivalent query in $ALG^\ell$.

This shows that SBCCQ queries can be evaluated with operators whose complexity is expressed with respect to $\ell$, the orthographic dimension, independently from the global dimension. Therefore, providing that the orthographic dimension, fixed in the schema, is low, poinsets in arbitrary dimension can be manipulated efficiently. In the case of spatio-temporal data for instance, this allows to manipulate 3-dimensional pointsets at the cost of dimension 2, as shown in the next section.

# 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 proposed by the LAMA laboratory of french geographers in Grenoble, France. This application considers the spatio-temporal behavior of several types of actors in a ski resort. 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 statisticaly 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: {*Sleep, Walk, Ski, Eat, Read, 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:

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

2. A set of moving objects, each of which representing 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**,
**nested** geom: (space: **rational(2)**)

**Relation** Mobile
(name : **string**
**nested** activity: (alpha: **string**,
time : **rational(1)**)
**nested** traj : (space: **rational(2)**,
time : **rational(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 query language, which is equivalent to the algebra of Section 2, relies on an SQL-like syntax, with the added possibility to introduce algebraic expressions on nested attributes in the **select** clause. These expressions are constructed using the operators **join**, **union**, **minus**, the projection '.' and the renaming of variables using the keyword **alias**. **join** is the most general operation: $p_1$ **join** $p_2$ denotes a natural join over the pointsets $p_1$ and $p_2$ which takes into account the names of the components: 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.

**alias** allows to rename components or variables and thus 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 pointset. For clarity, we sometimes use the equivalent syntax **restrict** $p_1$ **with** *cst*.

In order to respect the decomposition restriction, any algebraic expression, say $E$, is associated with a final projection on one or several of its components $C_i$, using the syntax $(E).(C_1, \ldots, C_n)$. Variables in nested attributes $A$ can be explicitely addressed by a (limited) path expression of the form $A.C.i$ where $i \leqslant k$ is the $k^{th}$ coordinate of the component $C$.

All queries below but the last one, which relies on the algorithm of Section 3, run in the current implementation of DEDALE.

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

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

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

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

   **select** (m.traj **join** r.geom).time
   **from** Resort r, Mobile m
   **where** m.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*) gives the part of Monica's trajectory inside the terrace's geometry: its projection on *time* is the result.

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

```
select   (m2.traj join (m1.traj join r.geom).time).space
from     Resort r, Mobile m1, Mobile m2
where    m1.name = 'Monica'
and      m2.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.

4. *Show the places where Monica sleeps*

```
select   ((restrict activity with "alpha = 'Sleep' ").time join traj).space
from     Mobile
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 and a spatial join. Note that the temporal join is "internal" to a single object since it involves the two nested relations (*activity* and *traj*) of Monica.

5. *Where did Monica and John meet?*

```
select   (m1.traj inter m2.traj).space
from     Mobile m1, Mobile m2
where    m1.name = 'Monica'
and      m2.name = 'John'
```

The join involves simultaneously time and space.

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

```
select   (m1.traj join r.geom).time inter (m2.traj join r.geom).time
from     Resort r, Mobile m1, Mobile m2
where    m1.name = 'Monica'
and      m2.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.

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

```
select   m.name, (((restrict activity with "alpha = 'Eat' ").time join traj)
                      join r.geom).time
from     Resort r, Mobile m
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.

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

```
select   ((activity join "alpha = 'Ski' ").[time alias time1]
           join activity
           join (activity join "alpha = 'Eat' ").[time alias time2]
         ) join "time1.1 ⩽ time.1 ⩽ time2.1").alpha
from     Mobile
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. The evaluation of this query involves only operations on time intervals: $time1.1$ should be less than $Max(time.1)$ and $time2.1$ 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.

The multidimensional data model introduced in Section 2 has been implemented in the DEDALE prototype initially designed for spatial data. DEDALE has been extended to handle *components* as introduced in the present paper. 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 : as shown in Section 3, 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.

# 5 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 result 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. These restrictions can be seen a an extension of the classical trade-off between efficiency of computation and accuracy of representation which were already encountered for geometric data in the vector and raster representation modes and the linear constraint data model. 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 on a real-life application that these design

choices are relevant and result in a powerful tool that provides novel query functionalities. 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 optimisation in complex queries. In addition, we currently investigate the introduction in DEDALE of temporal index structures [ST94] to complete the already existing set of spatial indices.

## 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

[BBC97] A. Belussi, E. Bertino, and B. Catania. Manipulating spatial data in constraint databases. In *Intl. Conf. on Advances in Spatial Databases (SSD'97)*, pages 115–141. Springer Verlag, LNCS 1262, 1997.

[CD98] S. Chardonnel and P. Dumolard. Personal communication, 1998.

[CGK96] 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.

[GK97] S. Grumbach and G. Kuper. Tractable recursion over geometric data. In *International Conference on Constraint Programming*, 1997.

[GO97] Jacod E. Goodman and Joseph O'Rourke. *Handbook of Discrete and Computational Geometry.* CRC Press, 1997.

[GRS98a] S. Grumbach, P. Rigaux, and L. Segoufin. On the Orthographic Dimension of Constraint Databases. Technical report, INRIA-VERSO, 1998. Submitted for publication. Available at `ftp.inria.fr/INRIA/Projects/verso/VersoReports-141.ps.gz`.

[GRS98b] S. Grumbach, P. Rigaux, and L. Segoufin. The DEDALE System for Complex Spatial Queries. In *Proc. ACM SIGMOD Symp. on the Management of Data*, 1998.

[GRSS97] S. Grumbach, P. Rigaux, M. Scholl, and L. Segoufin. DEDALE: A spatial constraint database. In *Intl. Workshop on Database Programming Languages (DBPL'97)*, 1997.

[GS95] R.H. Güting and M. Schneider. Realm-Based Spatial Data Types: The ROSE Algebra. *The VLDB Journal*, 4(3):243–286, 1995.

[GST94] 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.

[Güt94] R.H. Güting. An Introduction to Spatial Database Systems. *The VLDB Journal*, 3(4), 1994.

[KG94] P. Kanellakis and D. Goldin. Constraint programming and database query languages. In *Manuscript*, 1994.

[KKR90] P. Kanellakis, G Kuper, and P. Revesz. Constraint query languages. In *Proc. 9th ACM Symp. on Principles of Database Systems*, pages 299–313, Nashville, 1990.

[PVV94] 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.

[RFS88] N. Roussopoulos, C. Faloutsos, and T. Sellis. An Efficient Pictorial Database System for PSQL. *IEEE Transactions on Software Engineering*, 14(5):639–650, 1988.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.

[ST94] B. Salzberg and V.J. Tsotras. A Comparison of Access methods for Time Evolving Data. Technical Report NU-CCS-94-21 and CATT-TR-94-81, Northeastern . and Polytechnic U., 1994.

[SV89] 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.

[SV97] M. Scholl and A. Voisard, editors. *Proc. Intl. Symp. on Large Spatial Databases (SSD)*. LNCS No. 1262. Springer-Verlag, 1997.