Information Systems Vol. 16, No. 6, pp. 565–583, 1991
Printed in Great Britain. All rights reserved

0306-4379/91 $3.00 + 0.00
Copyright © 1991 Pergamon Press plc

ABSTRACT DATA TYPES FOR THE LOGICAL MODELING OF COMPLEX DATA

M. GARGANO,[1] E. NARDELLI[1] and M. TALAMO[2]

[1]Consiglio Nazionale delle Ricerche, Istituto di Analisi dei Sistemi ed Informatica, Viale Manzoni 30, 00185 Roma, Italy

[2]Università di Roma "La Sapienza", Dipartimento di Informatica e Sistemistica, Via Salaria 113, 00198 Roma, Italy

(Received 23 March 1990; in revised form 3 June 1991)

Abstract—In this paper we propose a logical data model for complex data. Our proposal extends the relational model by using abstract data types for domains specification and an extended relational algebra is also introduced. The introduction of the parameterized type Geometry(S), where S is a ground set of elements, allows the representation of complex aggregated data. As an example, we discuss how our model supports the definition of geographical DBMSs. Moreover, to show the generality of our approach, we sketch how the model can be used in the framework of statistical applications.

Key words: Logical model, Extended relational algebra, Abstract data types, Spatial Data, Geographical databases

1. INTRODUCTION

In these latter two decades, many application areas, for example office automation, interactive computer-aided design, geographic data processing, health data processing etc., characterized by more complex and varied data with respect to the traditional commercial and business applications, have been explored. This has stimulated the interest in employing the database technology to favor development and to increase the number of applications in such areas. It straightforwardly follows that such a technology, born and developed mainly to satisfy commercial and business data processing applications, has to be refined to adapt to the needs of the new application areas.

These new domains are generally dealing with the contemporary treatment of traditional alphanumeric data (i.e. numerals and strings) and spatial/geometric data. Yet, the peculiar requirements, whose analysis heavily determines the database design and the set of the functionalities available, are usually varying from one given application to another. This has promoted the development of special-purpose systems, i.e. environments furnishing a specialized technology for the efficient treatment of a well-defined and restricted set of applications.

The actual trends point towards the definition of systems for the integrated manipulation and representation of complex information. To pursue such an integration goal by exploiting the available technologies, a widely used approach rests on the coupling of traditional DBMSs and specialized systems that offer *ad hoc* technologies. The advantage of coupling different specialized technologies is that, with somewhat little effort, one can build a system managing traditional data by suitably setting up a communication channel between a DBMS (usually relational) and a specialized system for complex data. Furthermore, most of these systems guarantee on average good performances whenever they are designed for specific applications.

Geographical information systems are an important example of the attempt of extending database capabilities to deal with complex data by such integration policy and reflect, in paradigmatical way, the advantages and the drawbacks that this policy implies. Geographical information systems are concerned with the representation and manipulation of both alphanumeric and spatial/geometric data. In geographical applications the coupling policy relies both on the wide availability and reliability of the current DBMSs and on the powerful capabilities of specialized systems for the spatial/geometrical data processing. Nevertheless, available geographical information systems usually do not offer the advantages that database technology provides in traditional contexts. This happens mainly because integration issues rely directly on the physical implementation, without

reference to a well-defined logical architecture design, though much effort has been done in this regard [1–6].

In this paper we define a logical model that allows the modeling of complex information. We also show that by suitably instantiating some parameters of our model it supports the integrated modeling of spatial/geometric data in geographical databases and statistical applications.

The model rests on the relational model which we have suitably extended allowing abstract data types (ADTs) as domains. We have chosen the relational model to provide a formal and well-defined logical frame and we have extended it to provide a suitable representation and manipulation of complex information. Though the relational model defined by Codd [7] is a powerful tool for logical modeling of traditional applications thanks to its validated theoretical basis, it does not support an effective representation and an efficient manipulation of complex information [2, 4, 8–11].

The extension we propose rests on the definition of an ADT, namely Geometry(S), for dealing with data of a complex nature. It: (i) furnishes an expressive and powerful means for integrated modeling of complex data; (ii) provides primitives that directly model the kind of manipulation that the end-user performs on such data; and (iii) constitutes a basis for the formal and correct definition of algebraic high-level data manipulation and definition languages. As an example we show that Geometry(S) supports the representation and the manipulation of the spatial extension of spatial/geometric data. A system which implements the model and the primitives of the language is described in [12].

Our model provides advantages both to the designer and to the user. From the designer point of view, the model: (i) offers a formal frame for the correct specification of complex information at the logical level; and (ii) supports the separation of the abstraction levels in the database (physical independence of the data). From the user point of view, it guarantees an expressive and uniform view of the database content, while furnishing a means for a high-level problem oriented data manipulation language.

The paper can be logically divided into two parts. In the first one the logical data model and the extended relational algebra are introduced. In the second one, how our model applies to the case of geographical databases is illustrated. More specifically, in Section 2 the ADT Geometry(S), the extended relational model and the extended relational algebra are defined. In Section 3 the motivations pushing for the introduction of geographical DBMSs are discussed. In Section 4, how our model supports the definition of geographical databases is analyzed. In Section 5, query languages based on the manipulation primitives of the introduced model are presented. Finally, in Section 6 an example of the application of our model in the field of statistical databases is sketched.

2. THE EXTENDED RELATIONAL MODEL AND THE GEOMETRY ABSTRACT DATA TYPE

In this section we present and extensively discuss our logical data model, which extends the relational model and allows the use of ADTs to deal with complex data and their representation. This approach allows us to define attributes of this type and therefore to model complex aggregated data, on a well-founded theoretical basis. It also permits exploitation of the theoretical and practical knowledge acquired in the relational field by the database community. Moreover, the use of abstract data types leads to the definition of a richer and more powerful logical modeling tool.

ADTs in fact provide an effective technique for implementing and managing the separation between the physical structure of the objects of interest and their logical representation [12, 13]. ADTs have been considered extensively in semantic data modeling and are a widely used technique for database systems implementation [14].

An ADT describes a class of objects through their external properties instead of their physical representation. This allows the end-user to manipulate the objects at the abstraction level at which he looks at the reality, i.e. his conceptual view of the reality of interest. This makes it possible to define different physical representations according to the specific application needs. The introduction of ADTs also permits us to have different kinds of primitives according to the different requirements of the applications, thus contributing to support multiple views of the database.

Arguments also exist on the desirability to introduce user-defined types to tailor the database system to the needs of a specific user application [11a, 15, 16]. Non-traditional applications can be better handled if database systems with more flexible data typing are available. In this way some attributes can be of standard built-in types, while others can be non-standard types defined from the user for its specific modeling needs.

We think that the use of ADTs supports this user-oriented approach, but a crucial problem is maintaining the system performances at a reasonable level. A good solution that ensures both good flexibility and reasonable performances is to assure a minimal but powerful set of built-in types to have a flexible system without heavily affecting its performances. To be complete we also recall that the use of ADTs provides an excellent support during the phases of design and implementation of the relational database system [8, 17]. For example, a suitable use of ADTs permits to limit the access to a relation in predefined ways, guaranteeing a higher level of data security and data integrity [17].

Different techniques have been considered in the literature for the definition of ADTs [18, 19]. We consider here the algebraic approach to types, i.e. a type is a set of individual elements upon which some operations are defined. Since the set must be closed under these operations, the type is an algebraic structure. Thus, given an element 1 of type L the set of operations allowed on 1 is automatically and implicitly defined by L.

In the following sections we define our logical data model and then formally define the manipulation operators of the introduced algebra.

In the last section we introduce an algebraic structure, SHAPES, defined on a set of atomic elements and next we show how it corresponds to an ADT dealing with geometrical data, namely Geometry.

2.1. *The logical data model*

In this section our logical data model is presented. The model supports the representation of attributes whose domain is specified using ADTs.

Let U be a given set of names and let T be a finite set of type symbols $T = \{T_1, \ldots, T_m\}$. The elements of U are called "attributes". A function $type: U \rightarrow T$ is defined that associates to each attribute $A \in U$ a unique symbol $t = type(A) \in T$. To any attribute $A \in U$ a finite domain, called the domain of A, of values of type $t = type(A) \in T$ is associated and it is denoted by $dom(A)$. If the elements of $dom(A)$ are sets then A is called a *set-valued* attribute. Two attributes A_i, A_j such that $dom(A_i) = dom(A_j)$ are called *type consistent* if $type\ (A_i) = type(A_j)$.

We shall assume that for each attribute A_i, $type(A_i)$ and hence $dom(A_i)$ have been formally specified by means of an ADT definition. This implies that $T_i = type(A_i)$ denotes the ADT whose domain is $dom(A_i)$ on which manipulations have to be carried on, only using the operation defined in the ADT specification.

A relation schema $R_n = (A_1, \ldots, A_n)$ (where n is called the *degree* of the relation schema R_n) is a subset of U. We briefly write R_n to denote a relation schema R_n with attributes $A_1, \ldots, A_n$, or simply R when no ambiguity is possible. An instance of a relation schema R_n is a set $\{v_1, \ldots, v_h\}$ where each v_i, called "tuple", belongs to $\{dom(A_1) \times \cdots \times dom(A_n)\}$. An instance of R_n is denoted by $R_n(I)$.

Let X be an attribute. We denote with f_x an associative and commutative binary function that takes as source data two values of type $type(X)$ and returns a unique computed value of type $type(X)$. Thanks to the associative property that allows the iterative application of the function it is possible to extend it to take as source data a subset of values of type $type(X)$ and to return a unique computed value of $type(X)$. Therefore we write $f_x\{A, B, C, \ldots\}$ instead of $((A f_x B) f_x C) \ldots$, and we call it a *fusion function* on $(dom(X))$.

Notice that the domain $dom(X)$ of a fusion function is individualized through the ADT $T_i = type(X)$ associated to X. The ADT T_i in fact uniquely identifies the domain of elements $dom(X)$ and the set of the operations allowed on such a set. Consequently, f_x is a typed function, i.e. it takes a subset of elements of a given type T_i and returns a single element of the same type. Fusion functions are employed when computations on sets of attributes values are needed to answer extended relational queries (see [7] of the following section and Section 5).

2.2. The extended relational algebra

In this section the set $\mathbb{E}$ of the expressions of the extended relational algebra is formally defined and the syntax and the semantics of each element of $\mathbb{E}$ are given.† Borrowing the approach and the notation from [20], let us consider a database schema $D = (R_1, \ldots, R_M)$, where R_i is a relation schema:

(1) **Literals.** For any $c \in \cup_{A \in U} dom(A)$, $\{c\} \in \mathbb{E}$, it has degree 1 and $\{c\}(I) = \{c\}$.
(2) **Relations.** For each R_j in D, $R_j \in \mathbb{E}$ and $R_j(I) = r_j$.
(3) **Projection.** Let $a \in \mathbb{E}$, and let X be a subset of attributes of a. Then $a[X] \in \mathbb{E}$, and its degree is equal to card(X), where card(X) is the function that returns the number of elements in X, and $a[X](I) = \{t[X] \mid t \in a(I)\}$. We assume that card($\emptyset$) = 0 and $a[\emptyset](I) = \emptyset$ denotes the empty relation.
(4) **Cross Product.** Let $a_n, b_m \in \mathbb{E}$. Then $(a_n \times b_m) \in \mathbb{E}$ its degree is $n + m$ and $(a_n \times b_m)(I) = t_1 \circ t_2 \mid t_1 \in a_n(I) \wedge t_2 \in b_m(I)$, where $\circ$ denotes concatenation.
(5) **Restriction.** Let $a \in \mathbb{E}$ and let X and Y be two type consistent attributes of a. If Θ is the set of binary relation operators common to both type(X) and type(Y) then $a[X\theta Y] \in \mathbb{E}$, where $\theta \in \Theta$ and $a[X\theta Y](I) = \{t \mid t \in a(I) \wedge t[X]\theta t[Y]\}$.
(6) **Union, difference.** Let $a_n, b_n \in \mathbb{E}$, and such that their corresponding attributes are type consistent. Then $a_n \cup b_n$ and $a_n - b_n$ belong to $\mathbb{E}$ and both have degree n. They are defined as follows:

$$(a_n \cup b_n)(I) = \{t \mid a_n(I) \vee t \in b_n(I)\},$$
$$(a_n - b_n)(I) = \{t \mid t \in a_n(I) \wedge t \notin b_n(I)\}.$$

(7) **G-Compose.** Let $R_n \in \mathbb{E}$, let X and $Y = \{Y_1, \ldots, Y_k\}$ be two non-intersecting subsets of attributes of R_n and let $F_y = (f_{y1}, f_{y2}, \ldots, f_{yk})$, $k \leqslant n$, be a collection of k fusion functions defined respectively on $dom(Y_1)$, $dom(Y_2), \ldots, dom(Y_k)$. Then $G\text{-Compose}_X(F_y; Y)(R_n) \in \mathbb{E}$ and it has degree equal to card(X) + card(Y). Semantically, for each tuple $p \in R_n(I)$, let v_p be the tuple defined as follows‡:

$$v_p[X] = p[X],$$
$$v_p[Y_i] = f_{yi}\{t[Y_i] \mid t \in R_n(I) \wedge t[X] = p[X]\}, \quad i = 1, \ldots, k.$$

Then§

$$G\text{-Compose}_X(F_y; Y)(R_n)(I) = \{y_p \mid p \in R_n(I)\}.$$

(8) **G-Decompose.** Let $R_n \in \mathbb{E}$, X and Y be two subsets of attributes of R_n, $X \cap Y = \emptyset$, and card(Y) = 1. Then $G\text{-Decompose}_X(Y)(R_n) \in \mathbb{E}$ and it has degree card(X) + card(Y). Semantically, for each tuple $p \in R_n(I)$, let t be a tuple variable and let U_p be the set defined as follows:

$$U_p = \begin{cases} \{t \mid (\exists y) y \in p[Y] \wedge t[Y] = \{y\} \wedge t[X] = p[X]\}, & \text{if } Y \text{ is a set-valued attribute,} \\ \{p\} & \text{otherwise.} \end{cases}$$

Then¶

$$G\text{-Decompose}_X(Y)(R_n)(I) = \cup_{p \in R_n(I)} U_p.$$

As a syntactical choice, if any of the X or Y attributes coincides with the empty collection it will be omitted in the notation of G-Compose and G-Decompose operators.

---

†For examples regarding operators here introduced see Sections 5 and 6, illustrating the application of the extended relational algebra in the case of a geographical survey database and statistical manipulations.
‡We shall assume that $p[\emptyset]$ is defined as the identity element with respect to concatenation between tuples.
§The effect of $G\text{-Compose}_\emptyset(F_y; Y)$ is that all tuples of R_n (projected on Y) are "fused" in a single one whose Y-value is generated by the application of the fusion function. The effect of the $G\text{-Compose}_X(F_y; Y)$, when $k = 1$, is analogous of the operator "Aggregate Formation" of [20].
¶The effect of $G\text{-Decompose}_\emptyset(Y)$ is that all tuples of R_n (projected on Y) are "decomposed".

2.3. *The SHAPES algebra and the abstract data type "Geometry(S)"*

Given a not empty finite set S, we denote by H_S the set $H_S = P(P(S))$ where $P(S)$ is the powerset of S (hence H_S is the set of all sets of sets of elements of S). The elements of S are called *atoms*.

Definition 1†

Given a not empty finite set S, we call SHAPES on S the 6-tuple $\Psi_S = (H_S, \cup, \cap, \cap^*, geo, compl)$, where $\cap^*$, *geo* and *compl* are three operators defined on H_S, for all $A, B \in H_S$, as follows:

$$A \cap^* B = \{c \in P(S): (\exists a)(\exists b)\, a \in A \wedge b \in B \wedge c = a \cap b\},$$

$$geo(A) = \{X\}, \quad \text{where } X = \{x \in S: a \in A \wedge x \in a\},$$

$$compl(A, B) = \{Y\}, \quad \text{where } Y = \{y \in S: a \in A \cup B \wedge y \notin a\}$$

and $\cup$ and $\cap$ respectively denote the union and the intersection operations in the way they are used in set theory, i.e.:

$$A \cup B = \{c \in H_S: c \in A \vee c \in B\},$$

$$A \cap B = \{c \in H_S: c \in A \wedge c \in B\}.$$

Proposition 1

Ψ_S is an algebraic structure (briefly an algebra).

Proof—Ψ_S is clearly closed under the operations $\cup$, $\cap$, $\cap^*$, *compl* and *geo*.

An element A of Ψ_S is called a Shape. If A contains exactly one element (i.e. a single set of atoms) it is called simple. Otherwise, each of its elements (i.e. each one of the set of atoms it is made up by) is called A-component. Informally speaking, when the $\cap^*$ operator is applied to two Shapes A and B it returns a Shape C whose components are the result of the intersection of every A-component with every B-component. When the *geo* operator is applied to a Shape A containing more than one component it returns a simple Shape whose single component is the union of all the A-components. If *geo* is applied to a simple Shape A it returns Shape A itself. Figures 1a–f show in an informal way the operators of the algebra Ψ_S introduced in this section. The shaded part of the drawings represents the result of the indicated operation.

The introduction of an algebra corresponds to the definition of an ADT whose syntactical part is concerned with the signature of the algebra (operational syntax) and whose semantical part with its interpretation (operational semantics) [19].

To define the ADT Geometry(S), which to be concise, we borrow the notation from [18] using the technique based on the declaration of pre-condition and post-condition on the operations of the ADT. Some brief comments appear enclosed in square brackets.

NAME
[name of the introduced ADT]

Geometry(S: set)

SETS
[sets used for the formal definition of the introduced ADT]

S	set of atoms
G	set of elements of type Geometry(S)
Bool	set of Boolean values consisting of *true* and *false*

†The symbol $\cap$ is used both on the right and left side of the formulas to denote the usual operator of intersection. Though on the left side the elements are belonging to $P(P(S))$ and on the right one to $P(S)$ we hope the difference is clear from the context.

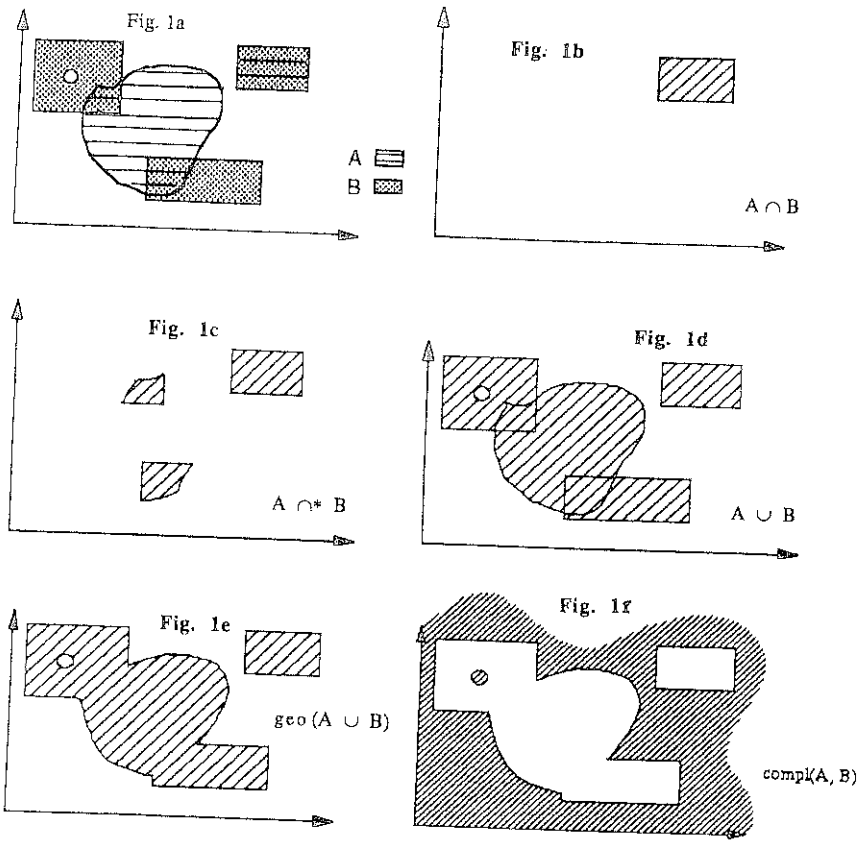

Fig. 1

SYNTAX

[declaration of the set of the operations available for the introduced ADT]

createdatum$G$ $\rightarrow G$

[creates a datum of type Geometry]

empty $G \rightarrow$ Bool

[given a datum of type Geometry says if it is empty or not]

$\cup$ $G \times G \rightarrow G$

$\cap$ $G \times G \rightarrow G$

$\cap^*$ $G \times G \rightarrow G$

Compl $G \times G \rightarrow G$

geo $G \rightarrow G$

SEMANTICS

[Range $a, b, \ldots = K$ means that the variables $a, b, \ldots$ denote elements of the set K]

[since all the preconditions are true (i.e. no condition is required to be able to apply any operation) they are omitted]

range $A, B, C = G$,
range $a, b, b', c, c' = P(S)$,
range $el = S$,
range $x =$ Bool,

post-createdatum$G(\ ; A) ::= a \in A \Leftrightarrow a = \emptyset \equiv \{\ \}$,
post-empty$(A; x) ::= (A = \emptyset \wedge x = true) \vee (\neg (A = \emptyset) \wedge x = false)$,
post-$\cup$ $(A, B; C) ::= c \in C \Leftrightarrow c \in A \vee c \in B$,
post-$\cap$ $(A, B; C) ::= c \in C \Leftrightarrow c \in A \wedge c \in B$,
post-$\cap^*$ $(A, B; C) ::= c \in C \Leftrightarrow (\exists a)(\exists b)\ a \in A \wedge b \in B \wedge c = (a \cap b)$,

$$post\text{-}compl(A, B; C) ::= c \in C \Leftrightarrow el \in c \wedge (a \in A \cup B \Rightarrow el \notin a) \wedge (c, c' \in C \Rightarrow c = c'),$$

$$post\text{-}geo(A; B)\dagger ::= (a \in A, el \in a \Rightarrow (\exists b)\, b \in B \wedge el \in b) \wedge (b \in B, el \in b \Rightarrow (\exists a)\, a \in A \wedge el \in a) \wedge (b, b' \in B \Rightarrow b = b').$$

The ADT Geometry can be built on any ground set of atomic elements since the nature of the atoms does not affect its definition. In other words, ADT Geometry is parametric with respect to the set of the atoms. Furthermore, since ADTs are definable in an incremental way, it is also possible to start from a generic set of elements on which an algebra is defined (i.e. in S some operations are defined and S is closed under these operations) and to build on top of this latter algebra the Geometry algebraic structure. To be concise, in the following we will denote Geometry(S) by G_S.

3. RATIONALE FOR GEOGRAPHICAL DBMSs

Geographical databases deal with a huge quantity of both descriptive and, usually bidimensional, spatial/geometric data. A descriptive data refers to conventional information about a given geographical entity (e.g. the flow of a river, the density of population of a given region, the average of some scalar quantities). The spatial/geometric data generally refers to any geometrical modification of the spatial extension of a given geographical entity (e.g. the shape and location of a geographical region or a river).

The rationale for the definition of geographical database management systems is that of obtaining a treatment of geographical data with the advantages of the well-developed technology of traditional DBMSs.

From the user community point of view it is widely acknowledged [2, 4, 5, 36] that an integrated geographical database must satisfy user requirements of:

- —supporting non-atomic domains for the representation of spatial data, so to consider that the user looks at the spatial extension of a geographical entity as an atomic concept. This allows manipulation at the logical level, the spatial/geometric extension of geographical objects according to this conceptual view;
- —symmetrical management descriptive and spatial/geometric characteristics of geographical entities so as to make it possible to identify an entity specifying any of the two aspects;
- —supporting direct spatial selection, i.e. searching for geographical entities on the basis of their spatial/geometrical properties. This allows location of the set of any spatial data contained in a given area. For example we can ask for the "the set of all lakes and rivers contained in region A" where region A is given making reference only to its spatial/geometrical extension and not to its internal encoding as it is instead done in [21];
- —supporting direct spatial computation that allows answering of queries like "Return the area of region A" or "Return the length of the river I am pointing at";
- —supporting indirect spatial selection, i.e. searching for geographical entities on the basis of their descriptive characteristics and processing their spatial/geometric aspects. This permits to answer queries like "Display a map of Italy with all cities where the average annual income of citizens is greater than 10,000 dollars" or "Show shape and location of all lakes with an area greater than 100 square km";
- —supporting multiple logical representation of the data. Different applications can have different logical views of the same kind of data. For example a given application can see a geographical region represented by the subset of the points of the plane occupied by the region itself and, by another application, as the minimum convex polygon covering the region, and so on;
- —supporting the usual traditional operations on conventional data.

To satisfy such requirements, efficiency is a very crucial problem in geographical databases because the representation of a simple piece of geographical information may need a huge quantity of

†The right side of this condition has not been written in the most synthetic way, but in a form which allows to better evidence properties of the *geo* operation.

storage memory. Many research efforts have been done to try to solve this problem by investigating new physical data structures and algorithms for the efficient representation, manipulation and querying of geometrical data and many successes have been obtained [4, 11, 17, 22–27, 34, 35].

Though putting the foundations for the specification of geographical DBMSs is an objective not yet reached, there is a general agreement, founded on the above-mentioned requirements, in stating a reasonable set of characteristics that a geographical DBMS should own. Among these, some directly derive from those of the traditional DBMSs (e.g. the design of a friendly user-interface, physical independence, . . .) and they assume a particular relevance in geographical databases. Some others (e.g. multiple physical representation) are peculiar to geographical applications and are generally not needed in traditional applications.

The most relevant characteristics that a geographical DBMS should own are briefly discussed in the following:

—**Friendly user-interface.** This allows guarantee of a high-level interaction between the user community and the database hiding low-level representation details. This is a desirable characteristic in geographical applications more than in the traditional ones where the graphical support needed is of little significance and the design of sophisticated interfaces is less important. Appropriate pointing devices to formulate query and update data by directly pointing to images on the screen greatly support a friendly interaction between the end-user and the geographical database.

—**Multiple views.** This allows different views of the data contained in the database to be made available and users to to support information sharing among different applications.

—**Physical independence.** It guarantees the independence of the application from the physical representation of the data. Database technology mainly rests on the separation of the abstraction levels in the database [28]. A consequence of physical independence is that it allows support of the optimization of the physical level without affecting the user. Thus the separation among the logical and physical representation of the data is a desirable characteristic above all in geographical DBMSs where efficiency issues must be carefully considered.

—**Multiple physical representation.** This guarantees the possibility of representing, at the internal level, the objects of interest by using different representation schema, according to the needs of the applications. For example, geographical applications generally require both continuous and discrete representation of objects to support complex geometrical algorithms in an efficient way [2, 23].

—**Data definition language.** In geographical applications, like in other non-traditional application fields, it should be desirable to have at disposal a data definition language that permits the definition of typed domains. This allows automatic association to the data of interest the set of the operations that we would like to perform on such data. In this way the database capability to support the modular management of very specific applications is also improved. It is obvious that the tradeoff between the expressiveness of the language and the overall efficiency of the system should be balanced.

—**Data manipulation language.** As it also follows from the previous point, the data manipulation language should furnish a high-level query language and a problem-oriented manipulation language able to support the use of suitable hardware devices like mice and graphic tablets.

It is therefore desirable that a logical model:

—guarantees to the user a uniform and expressive view of geographical entities, still allowing him to refer to them through both descriptive and/or geometrical/spatial characteristics;
—supports the definition of high-level data manipulation languages and the design of physical structures for the efficient answering mixed queries;
—provides to the designer a formal methodology for the correct specification of the geographical information at the logical level.

Our proposal is based on the definition of a suitable logical data model that supports the representation, the manipulation and querying of geographical information. This model, which

integrates descriptive and spatial/geometric data, defines a reference schema for the integration among different subsystems while providing the user with a high-level data manipulation language [29].

The model guarantees to the user a uniform view of geographical entities, while allowing him to refer to them through both descriptive and/or geometrical/spatial characteristics. Since the query language, by means of the logical model, does not rest on the physical level, the independence between integrated physical systems and query language can be guaranteed and a more flexible overall system results.

The introduction in our model, first sketched in an architectural framework in [29], of the ADT Geometry(S) supports: (i) the definition of high-level manipulation and definition primitives; and (ii) the separation of the levels of abstraction and, as a consequence, the model satisfies the requirements expressed by the points considered above.

4. HOW THE MODEL SUPPORTS THE REPRESENTATION OF GEOGRAPHICAL OBJECTS

A geographical entity is an element of the reality of interest that can be seen under any one of its two aspects: the descriptive one and the spatial/geometric one [6, 30]. The former provides descriptive characteristics and properties of the object, while the latter furnishes its shape and its spatial location. For example, the descriptive nature of the entity "Mississippi river" can be expressed by a collection of values as its name, length, flow etc. while the geometric one can be its shape, its location in some reference system, etc. From the point of view of the logical data modeling, the choice of a relational model leads to represent the logical structure of data in tables. In such a context, the descriptive nature of a geographical entity (e.g. the flow of a river) can straightforwardly correspond to a single-valued attribute whose underlying domain is traditional (e.g. alphanumeric). On the other side, attributes defined on traditional atomic domains cannot adequately represent the geometrical nature of geographical objects. In fact, using traditional domains naturally leads to represent an object distributing its spatial/geometrical characteristics on several relations as in [21, 31, 32]. In this way one obtains a fragmented logical data representation, far from the way the user looks at the reality of interest.

In order that the logical structure of data reflects the way the user looks at data themselves, the logical model must therefore support non-atomic domains, able to represent in a complete way the spatial/geometrical component of an object. Our model allows the integrated representation of geographical entities thanks to the introduction of the ADT Geometry(S) that supports SHAPES-valued attributes. Since a single element of SHAPES, that is an instance of ADT Geometry(S), uniquely defines the shape and location of an object, the spatial/geometrical nature of a geographical entity can be fully represented at the logical level by a single-valued attribute. The logical view of the database thus can straightforwardly support the user view of the geographical reality of interest, because the logical model explicitly takes into account that the user considers the spatial/geometrical nature of a geographical entity as an atomic concept.

To be concrete we must specify what is the set of atomic elements S we use as a basis of our ADT Geometry(S). In this section we focus on the case in which the spatial/geometrical nature of a geographical entity is described by sets of points belonging to a raster decomposition of the plane. Thus, S is a finite subset of the points of the plane that have integer coordinates (*raster plane*). Consequently, our model allows representation of geometrical extensions of geographical entities even when such extensions correspond to non-connected sets. To be concise in the following we call A_i a *descriptive attribute* if $type(A_i) = T_i$ and we call A_i a *geometrical attribute* if $type(A_i) = G_{S_i}$, where $S_i = dom(A_i)$.

Let us now formally define what we mean by raster plane. Let $\mathbb{I}$ denote the set of integers.

Definition 2

Given two closed intervals I_1, I_2 of $\mathbb{I}$, we call a raster plane, denoted by $\mathscr{R}$, the Cartesian product of I_1 and I_2, i.e. $\mathscr{R} = I_1 \times I_2$.

RIVERS			
Name	Crossed-region	Length	River-shape
river_1	A	2	{ (riv_11) }
river_1	B	4	{ (riv_12) }
river_2	B	2.5	{ (riv_2) }
river_3	A	4.7	{ (riv_31) }
river_3	H	5	{ (riv_32) }
river_3	B	6	{ (riv_33) }
river_4	C	1	{ (riv_4) }
river_5	B	10	{ (riv_5) }

Fig. 2. Relation RIVERS

Following the notation of Section 2.1, $H_{\mathscr{R}}$ will denote the set $P(P(\mathscr{R}))$. Given a raster plane, we can define the SHAPES algebra $\Psi_{\mathscr{R}} = (H_{\mathscr{R}}, \cup, \cap, \cap^*, compl, geo)$ in the same way as the previous paragraph. We can now use the algebra $\Psi_{\mathscr{R}}$ to model the spatial/geometrical nature of a geographical reality of interest. The way geographical information is thus modeled and can be manipulated will be introduced in the next paragraph where we shall define primitives for querying and updating.

To clarify better how our model supports the integrated representation of geographical information let us consider an example of a geographical survey database. The database has the following relation schema, where each underlined attribute is a candidate key for the relation it belongs to (therefore each geometric attribute can be, or can belong to, a candidate key):

RIVERS(Name, Crossed-region, Length, *River-shape*)
LAKES(*Name*, Crossed-region, Area, *Lake-shape*)
REGIONS(*Name*, Area, Density, *Region-shape*)

In this example we have:

U = {Name, Crossed-region, Length, River-shape, Lake-shape, Region-shape, Area, Density},
T = {string, char, float, $G_{\mathscr{R}}$} where $\mathscr{R}$ is the raster plane defined above.

We focus on the relation RIVERS shown in Fig. 2. Analogous considerations hold for the relation schema LAKES and REGIONS. We shall denote with $\mathbb{R}$ the domain of real numbers. We have:

type(Name) = string	*dom*(Name) = {Strings}
type(Crossed-region) = char	*dom*(Crossed-region) = {Char}
type(Length) = float	*dom*(Length) = $\mathbb{R}$
type(River-shape) = $G_{\mathscr{R}}$	*dom*(River-shape) = $H_{\mathscr{R}}$

A tuple of relation RIVERS represents information about a single segment of a river. In particular, for each river, the name of the crossed region, the part of its spatial extension, called segment, individualized by its intersection with the crossed region and the length of such a segment are given. Clearly the intersection may result in more than one segment, in which case more than one tuple is generated. Since River-shape is of type $G_{\mathscr{R}}$ each value of this attribute is a list of couples like $\{\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}\}$ because the geometrical/spatial nature of rivers is modeled by a set of pairs of integers (set of points of the raster plane). For the sake of shortness, we have denoted such values by "*riv_ij*" denoting by this expression the list of points that compose the jth segment of the river "river_i". The crossed region is represented and accessed by its name. If we have interest in it, we can choose to represent the crossed region by its shape as the relation RIVERS-2 shows in Fig. 3 (where obviously *type*(Crossed-region) = $G_{\mathscr{R}}$ and one element of Crossed-region has been denoted by "reg_X" analogously to "*riv_ij*").

This example also shows that in the model it is possible to represent maps by relations in which every tuple corresponds to a region defined by a set of descriptive attributes and by one single geometric attribute as in [3, 6]. Since in our model there are no restrictions on the number of geometric attributes in a relation schema, a relation with two (or more) geometric attributes does

RIVERS-2			
Name	Crossed-region	Length	Segment
river_1	{ {reg_A} }	2	{ {riv_11} }
river_1	{ {reg_B} }	4	{ {riv_12} }
river_2	{ {reg_B} }	2.5	{ {riv_2} }
river_3	{ {reg_A} }	4.7	{ {riv_31} }
river_3	{ {reg_H} }	5	{ {riv_32} }
river_3	{ {reg_B} }	6	{ {riv_33} }
river_4	{ {reg_C} }	1	{ {riv_4} }
river_5	{ {reg_B} }	10	{ {riv_5} }

Fig. 3. Relation RIVERS-2.

not require to be interpreted as a relation with only a single geometric attribute as in [3, 6] but can be interpreted only on the basis of the constraints expressed by the chosen database design criteria.

5. QUERY AND MANIPULATION PRIMITIVES

In this section we show that the algebra we have introduced through our model constitutes the basis for a high-level query language, which can be directly employed by the user to manipulate geographical information. We first explain in detail how the algebra manipulates relations defined in our model. Successively we present the query language.

5.1. *Fundamental primitives*

We extended the relational algebra essentially by defining two new algebraic operators that are capable of directly manipulating SHAPES-valued attributes. Due to the underlying formal frame introduced by the ADT Geometry($\mathscr{R}$), the standard relational operators can deal with SHAPES-valued attributes, as it was shown in Section 2. In this section we clarify, through examples based on the geographical survey database presented before, the use of the operators G-Compose and G-Decompose, formally defined in Section 2.

5.1.1. $G\text{-}Compose_X(F_y; Y)(R_n)$. Referring to relation RIVERS (Fig. 2) let us suppose that we want to group the river segments of those rivers that cross the same region. This example illustrates the use of the G-Compose operator when $X \equiv$ Crossed-region, $Y \equiv$ River-shape and $F_y \equiv (f_{\text{River-shape}}) \equiv \cup$. In this case the operator groups the values of the attribute River-shape applying to the grouped values the fusion function $\cup$ and discards the remaining attributes. The grouping is done on the basis of the equality of the values of the attribute Crossed-region. The following expression can be issued:

$$G\text{-Compose}_{\text{Crossed-region}}(\cup; \text{River-shape})(\text{RIVERS}).$$

What is obtained can be represented in a relation RIVERS-BY-REGION, presented in Fig. 4.

The value of the geometric attribute River-shape in each tuple of the relation RIVERS-BY-REGION is thus the set of the river segments belonging to the same region.

Let us now illustrate an example of the G-Compose operator, applied to the relation RIVERS, where $X \equiv$ Name, $Y \equiv$ River-shape and $F_y \equiv (f_{\text{River-shape}}) \equiv geo$. In this case the operator groups the values of the attribute River-shape applying to the grouped values the fusion function *geo* and discards the remaining attributes. The grouping is done on the basis of the equality of the values of the attribute Name. When the fusion function *geo* is used in this way, the G-Compose operator

RIVERS-BY-REGION	
Crossed-region	River-shape
A	{ {riv_11}, {riv_31} }
B	{ {riv_12}, {riv_2}, {riv_5} , {riv_33}}
C	{ {riv_4} }
H	{ {riv_32} }

Fig. 4. An example of a G-Compose operator.

RIVERS-REGION	
Crossed-region	River-shape
A	{ {riv_11} }
A	{ {riv_31} }
B	{ {riv_12} }
B	{ {riv_2} }
B	{ {riv_5} }
B	{ {riv_33} }
C	{ {riv_4} }
H	{ {riv_3} }

Fig. 5. G-Decompose operator.

RIVERS-3	
Name	River-shape
river_1	{ {riv_11, riv_12} }
river_2	{ {riv_2} }
river_3	{ {riv_31, riv_32, riv_33} }
river_4	{ {riv_4} }
river_5	{ {riv_5} }

Fig. 6. An example of a G-Compose operation.

simply "merges" the values of a specified (set of) attribute(s) (e.g. $Y \equiv$ River-shape in relation RIVERS) on the basis of the equality of the values of another (set of) attribute(s) (e.g. $X \equiv$ Name in relation RIVERS). Relation RIVERS-3 in Fig. 6 is obtained by applying the following expression:

$$G\text{-Compose}_{\text{Name}}(geo;\ \text{River-shape})(\text{RIVERS}),$$

to relation RIVERS of Fig. 2.

By $\{\{\text{riv\_11}, \text{riv\_12}\}\}$ we indicate the simple shape containing the union between the set of the points denoting the spatial extension of the river segment riv_11 and that denoting the spatial extension of the river segment riv_12. The same notation is adopted for all tuples in the above relation and in the following ones.

Since the use of the fusion function *geo* in the G-Compose operator also allows to obtain a new spatial/geometric entity by merging entities values stored in the database, it is also reasonable to calculate some new values related to the new entity obtained from the values associated to the merge entities. Let us illustrate this possibility by an example.

Let us assume that, for the attribute Length of relation RIVERS of Fig. 2, *type*(Length) denotes the ADT "Integer" whose formal specification allows the use of the operation "+" in the sense it is used for performing addition among integer numbers. Using this operation as a fusion function while applying the G-Compose operator permits us to compute the total length of the rivers in the relation RIVERS. In this relation only the length of the segments of each river is represented. We can merge the values of the River-shape attribute on the basis of the values equality of the Name attribute. For each river, i.e. for each value of the Name attribute, we can also sum the length of its segments by applying the fusion function "+". In such a way the total length and the corresponding shape of each river is obtained. This information was not explicitly represented in the original relation RIVERS, but it is now derived by aggregating information (length values) from that relation.

Relation RIVERS-4 in Fig. 7 is obtained by applying the following expression:

$$G\text{-Compose}_{\text{Name}}(+, geo;\ \text{Length},\ \text{River-shape})(\text{RIVERS}),$$

to the relation RIVERS of Fig. 2, where $(+, geo;$ Length, River-shape) denotes that the fusion function "+" has been applied to the attribute Length and the fusion function *geo* has been applied to the attribute River-shape.

RIVERS-4		
Name	Total-length	River-shape
river_1	6	{ {riv_11, riv_12} }
river_2	2.5	{ {riv_2} }
river_3	15.7	{ {riv_31, riv_32, riv_33} }
river_4	1	{ {riv_4} }
river_5	10	{ {riv_5} }

Fig. 7. G-Compose operation with two fusion functions.

The value of the geometrical attribute in RIVERS-4 is a simple Shape since it results from the application of the fusion function *geo*. If follows that the resulting relation cannot be disaggregated in the original components.

Notice that a safe use of the fusion functions in the G-Compose operator is assured because, due to our ADT approach, the manipulation functions allowed on a set of values of an attribute A_i are the only operation defined in the ADT specification of $type(A_i)$.

5.1.2. G-Decompose. Referring to the relation RIVERS-BY-REGION of Fig. 4, let us suppose that we want to derive the segments of the represented rivers, with respect to the region they cross. The following expression can be issued:

$$G\text{-Decompose}_{\text{Crossed-region}}(\text{River-shape})(\text{RIVERS-BY-REGION}).$$

The result can be represented in a relation RIVERS-REGION shown in Fig. 5. Notice that the resulting relation is the projection of the relation RIVERS on attributes Crossed-region and River-shape (recall that the application of the G-Compose operator has discarded the remaining attributes). In general, this operator disaggregates the values of a specified attribute (e.g. River-shape) with respect to a specified (set of) attribute(s) (e.g. Crossed-region).

The operator has decomposed each Shape into its constituent components, and therefore now each value of the geometric attribute River-shapes is the simple Shape (i.e. a Shape having only one component) of a river segment that crosses a given region. Notice that it can be said that the relation RIVERS-BY-REGION has been normalized by the application of the G-Decompose operator. At the same time, the values of any geometric attribute is always a single element of SHAPES though it may generally be not a simple shape. That differentiates our approach from the others, which are based on a hierarchical nested representation of the data [9–11].

5.2. A high-level query language

In this section we show that the introduced extended relational algebra constitutes the basis of a high-level geographical database query and manipulation language. In other words, our approach allows us to define a query language for manipulating both geometrical and descriptive attributes in a homogeneous way. At the same time, due to the use of ADTs, we are able to efficiently implement its primitives using data structures suited to the characteristics of the data. The description of a system implementing the language sketched in this section can be found in [12].

The examples of this section have the purpose of explaining the semantics of the operators we have introduced and giving an idea of the applicability of our algebra as a data manipulation and querying language. We give some examples to show how general geographical queries (i.e. queries involving predicates on both geometric and descriptive properties of geographical data) can be formulated by suitable expressions of the algebra. We still use the geographical survey database presented before, with some added relations. Examples which are a straightforward extension of the classical relational approach have been omitted.

- **Geometrical selection.** Having at disposal the relation CULTIVATIONS (Fig. 8) representing cultivations and chemical treatment of pieces of lands, the user can ask the database to "return the names of the rivers whose river segments cross some tobacco cultivated lands".

CULTIVATIONS

Cultivation	Chemical	Cult-quantity	Chem-quantity	Shape
corn	A	20000	13	{ [A1] }
potatoes	B	10050	17	{ [A2] }
tobacco	R	14500	23	{ [A3] }
corn	C	37850	30	{ [A4] }
potatoes	B	15070	20	{ [A5] }

Fig. 8. Relation CULTIVATIONS.

In this query the information can be directly extracted from the database issuing the expression (the arguments of the operator have been put on different lines to increase readability):

$$\text{Project}_{\text{Name}}(\text{Select}_{\text{cultivation} = \text{'tobacco'}} (\text{Join}_{\text{River-shape} \cap^* \text{Shape}}(\text{RIVERS, CULTIVATIONS})))$$

where $\cap^*$ has been used as join† operator.

A possible formulation in an SQL-like language is:

SELECT Name
FROM
  SELECT Name, Cultivation
  FROM RIVERS, CULTIVATIONS
  WHERE RIVERS.River-shape INTERSECTS CULTIVATIONS.Shape
  WHERE Cultivation = "tobacco".

Note that the use of an operator able to manipulate SHAPES-valued attributes as the join operator has the effect of performing what can be called a "spatial/geometric join" on the geographical entities involved. In fact, its effects are similar, from a logical point of view, to those of the "classical" join when applied to descriptive attributes.

- **Checking spatial differences.** The user can ask the database to calculate the difference in spatial–geometrical terms of the specified geographical entities (e.g. for eliminating slivers following a map overlay) by issuing a query like "return the parts of tobacco cultivated lands, which are not in region $R1$".

We can consider the $\cap^*$ operator as a fusion function on the domain SHAPES, since $\cap^*$ is an operator defined in the formal specification of ADT Geometry($\mathscr{R}$). If we assume to have extracted from relation REGIONS of previous figure the relation:

$$\text{REGION-1} = \text{Select}_{\text{Name} = \text{'}R1\text{'}}(\text{REGIONS}),$$

and from relation CULTIVATIONS of Fig. 8:‡

$$\text{CULT-1} = G\text{-Compose}(\cup; \text{Shape})\ (\text{Select}_{\text{cultivation} = \text{'tobacco'}}(\text{CULTIVATIONS})),$$

then the query can be answered by issuing the following expression (the arguments of the operator have been put on different lines to increase readability):

$$G\text{-Compose}(\cap^*; \text{Shape})\ ([G\text{-Compose}(\mathit{Compl}; \text{Shape})\ (\text{REGION-1})] \cup \text{CULT-1})$$

In the outer G-Compose operation we have $X \equiv \emptyset$, $Y \equiv$ Shape and $F_y \equiv (f_{\text{Shape}}) \equiv \cap^*$. In the inner G-Compose $X \equiv \emptyset$, $Y \equiv$ Shape and $F_y \equiv (f_{\text{Shape}}) \equiv \mathit{Compl}$. The result of the query is a relation constituted by one (geometrical) attribute whose unique value represents the requested difference. A possible expression in a SQL-like language is:§

SELECT Shape FROM
  MERGE APPLYING $\cap^*$ TO Shape
  FROM
    ((MERGE APPLYING *Compl* TO Shape
    FROM REGION-1)
    UNION CULT-1)

---

†**θ-Join.** Let $a_n, b_m \in \mathrm{E}$, and let X and Y be two type-consistent attributes belonging to a_n and b_m, respectively. A θ-join between a_n and b_m on attributes X and Y is defined as follows (see also point 5 in Section 2.2):

$$a_n[X\theta Y] = (a_n \times b_m)[X\theta Y] = \{t_1 \circ t_2 \mid t_1 \in a_n \in t_2 \in b_m \wedge a_n[X]\theta b_m[Y]\}.$$

‡Recall (see the end of Section 2.2) that the set X of attributes with respect to which it is legal to make the composition can be empty: this permits to apply the fusion function to all tuples in a relation. For example this option is used in the inner G-Compose to merge all shapes in a single one.

§We suppose to have an SQL-like operator "MERGE [G] [WRT X] [APPLYING f_y to Y]FROM R" corresponding to our G-Compose$_X(F_y; Y)(R)$, where square brackets represent optionality.

REGIONS		
Name	Area	Shape
R1	200	{ {reg_R1} }
R2	200	{ {reg_R2} }
R3	250	{ {reg_R3} }
R4	120	{ {reg_R4} }

Fig. 9. Relation REGIONS.

MY-REGION
Region-shape
{ {reg_A} }

Fig. 10. An example of a constant relation.

- **Windowing.** The user can ask the database the specified geographical entities that fall in a given area by issuing a query like "return the names and the shapes of the river segments that cross the region A, defined on the screen".

The region user depicts on the screen can be represented by a constant relation since this is a literal of the algebra expressions [see point (1) of Section 2.2]. If MY-REGION is the relation shown in Fig. 10, where $\{\{\text{reg\_}A\}\}$ denotes the spatial extension of the region A, the query can be expressed by the following expression of the algebra:

$$\text{Project}_{\text{Name,River-shape}}(\text{Join}_{\text{River-shape} \cap^* \text{Region-shape}}(\text{RIVERS,MY-REGION}))$$

where again $\cap^*$ has been used as join operator.

A possible expression in an SQL-like language is:

SELECT Name, River-shape
FROM RIVERS
WHERE RIVERS.River-shape INTERSECTS MY-REGION.Region-shape

- **Clipping.** The user can ask to "clip to the window region the rivers intersecting the window defined on the screen".

The query of the previous example extracts the tuples of relation RIVERS satisfying the given property of intersection. The query of this example is asking for the intersection between the window the user has defined on the screen and the rivers represented in the database. We are therefore calculating something new by using both the information contained in the database and that furnished from the user. Note that clipping also allows us to answer queries as "Return all the cities within a range of 5 miles from Rome" issuing a constant relation where the value of its (geometrical) attribute is the circle having center in Rome and radius 10 miles.

The clipping window can be represented as in the previous example. To obtain the piece of a river that falls into the window we need to select the points common to both the river shape and the window shape.

The following expression can be written (the arguments of the operator have been put on different lines to increase readability):

$$G\text{-Compose}(\cap^*; \text{River-shape})$$
$$([G\text{-Compose}(geo; \text{River-shape})(\text{RIVERS})] \cup \text{MY-REGION})$$

In the outer G-Compose operation we have $X \equiv \emptyset$, $Y \equiv$ River-shape and $F_j \equiv (f_{\text{River-shape}}) \equiv \cap^*$. In the inner G-Compose $X \equiv \emptyset$, $Y \equiv$ River-shape and $F_j \equiv (f_{\text{River-shape}}) \equiv geo$. The result of the query is a relation constituted by one attribute whose unique value is the requested intersection represented by a simple shape.

Note that a syntactically cleaner expression for this expression would have required in the inner G-Compose a renaming of the geometrical attribute. For the sake of simplicity we have not introduced renaming in our model, even if it can be added without changing the expressions of the extended algebra, but with a few syntactical modifications.

If it is required to maintain the distinction among rivers after clipping then the query has to start from relation RIVERS-3 (Fig. 6) where each tuple corresponds to a river. Issuing an expression

based on the use of the G-Compose operator with fusion function $F_y \equiv (f_{\text{River-shape}}) \equiv \cup$ we can preserve the identity of the rivers:

G-Compose ($\cap^*$; River-shape)
([G-Compose ($\cup$; River-shape) (RIVERS-3)]$\cup$MY-REGION)

In this case we in fact obtain a relation with one geometrical attribute whose value is not a simple shape, but a shape where each component represents a single segment of river intersected with the given window. Then the obtained shape can therefore be disaggregated in the original components. The choice between the two formulations depends on the needs of the user.

An SQL-like expression for the former formulation of the query can be the following:

```
SELECT River-shape
FROM
  MERGE APPLYING ∩* TO River-shape
  FROM
    ((MERGE River-shape
    FROM RIVERS)
    UNION MY-REGION)
```

and for the latter formulation the following:

```
SELECT River-shape
FROM
  MERGE APPLYING ∩* TO River-shape
  FROM
    (MERGE APPLYING∪TO River-shape
    FROM RIVERS-3)
UNION MY-REGION
```

To increase the expressiveness of the language we can also suppose to have an SQL-like operator "COMPOSE Y [WRT X] FROM R" corresponding to our G-Compose$_X(\cup; Y)(R)$. Thus, an analogous expression can be used for the latter formulation of the query using the "COMPOSE" operator instead of "MERGE":

```
SELECT River-shape
FROM
  MERGE APPLYING ∩* TO River-shape
  FROM
    (COMPOSE River-shape
    FROM RIVERS-3)
    UNION MY-REGION
```

6. A STATISTICAL APPLICATION

Let us consider the example of the survey database of Figs 11 and 12 and suppose that we want to perform statistical manipulations on the represented data. We show that by suitably changing the ground set S of the ADT Geometry(S), our model provides at least the same facilities of representation and primitives of manipulation of other well-known approaches in the field of

WEEKEND-TV			
TV-Program	TV-Station	Viewers	Sponsors
{{RED}}	1	8	{{1},{2}}
{{BLACK}}	2	14	{{4}}
{{BLUE}}	2	10	{{3}}
{{YELLOW}}	3	6	{{1},{2}}
{{PINK}}	3	5	{{4}}

Fig. 11. A relation from a television survey database.

WEEKDAY-TV			
TV-Program	TV-Station	Viewers	Sponsors
{{RED}}	1	8	{{1},{3}}
{{RED}}	2	12	{{1},{2}}
{{BLUE}}	3	20	{{1},{3}}

Fig. 12. A relation from a television survey database.

statistical database query languages. To be concrete we will refer to the running example in [20], rewriting it here with the slight modifications needed to manipulate the represented data in our approach.

Let Σ be a suitable finite set of strings of characters and let H_Σ denote the set $P(P(\Sigma))$. Given H_Σ, we can define the SHAPES algebra $\Psi_\Sigma = (H_\Sigma, \cup, \cap, \cap^*, compl, geo)$ and $Geometry(\Sigma)$ in the same way as Section 2.3. In an analogous way, given a finite subset I of the integer numbers, we can define the SHAPE algebra on the set $H_I = P(P(I))$, denoted by Ψ_I. We have:

$$\text{WEEKEND-TV}(\underline{\text{TV-Program}}, \underline{\text{TV-Station}}, \text{Viewers}, \text{Sponsors})$$

$$\text{WEEKDAY-TV}(\underline{\text{TV Program}}, \underline{\text{TV-Station}}, \text{Viewers}, \text{Sponsors})$$

$$type(\text{TV-Program}) = G_\Sigma \quad dom(\text{TV-Program}) = H_\Sigma$$

$$type(\text{TV-Station}) = \text{Integer} \quad dom(\text{TV-Station}) = \mathbb{I}$$

$$type(\text{Viewers}) = \text{Integer} \quad dom(\text{Viewers}) = \mathbb{I}$$

$$type(\text{Sponsors}) = G_I \quad dom(\text{Sponsors}) = H_I$$

We now give some examples of statistical manipulations of this database using our extended relational algebra:

1. "Give the total number of weekday viewers for each TV program" can be answered by issuing the expression:

$$G\text{-Compose}_{\text{TV-Program}}\ (+; \text{Viewers})\ (\text{WEEKDAY-TV})$$

that returns the relation of Fig. 13:

2. "Give the TV-Programs of weekend-day for each TV-Station calculating the total viewers for each station" can be answered by issuing the expression (Fig. 14):

$$G\text{-Compose}_{\text{TV-Station}}(+, \cup; \text{Viewers}, \text{TV-Program})\ (\text{WEEKEND-TV})$$

SUM-VIEWERS	
TV-Program	Viewers
{{RED}}	20
{{BLUE}}	20

Fig. 13. Total number of weekday viewers for each program.

STATION-PROGRAMS-VIEWERS		
TV-Station	TV-Programs	Viewers
1	{{RED}}	8
2	{{BLACK},{BLUE}}	24
3	{{YELLOW},{PINK}}	11

Fig. 14. The programs and total number of weekend viewers for each station.

JOINT-SPONSORSHIP	
TV-Station	Sponsors
1	{{1,2}}
2	{{3,4}}
3	{{1,2,4}}

Fig. 15. The joint sponsors.

Suppose now that all the sponsors of weekend programs are joined for a given period during which they cannot separately finance TV-stations. By applying a G-Compose operator with fusion function *geo* we can represent these new entities. For example issuing the expression:

$$G\text{-Compose}_{\text{TV-Program}}(geo;\ \text{Sponsors})(\text{WEEKEND-TV})$$

we obtain the relation shown in Fig. 15.

Notice that in this relation the sponsors cannot be separated but by updating the database. Thus using the fusion functions *geo* or $\cup$ it is possible to manage in different ways the concepts of a binded group and of simple collection of elements.

7. CONCLUSIONS

We have presented a logical data model, based on an extension of the relational model, which provides the capability of defining and manipulating complex entities by introducing ADTs as domains.

Mainly we have shown that by suitably choosing the ground set of the atoms S, the introduced ADT Geometry(S) allows the representation and manipulation of complex data at a very high level. In fact, it introduces a formal frame that allows the representation of complex aggregated data at the logical level. Also, the extended relational algebra constitutes a high-level procedural language that makes it possible to the user to deal with complex entities as if they were atomic elements.

As an example, we have shown that a suitable choice of the ground set S supports the representation and the manipulation of spatial–geometrical information. As far as geographical databases are concerned, our model seems applicable to any physical representation schema based on a discrete decomposition of the plane, thus allowing the modeling of complex spatial entities whose representation is based on such decompositions. This will allow our model to support multiple physical representations as requested in most of the application domains dealing with complex information. The degree of generality of plane decompositions that can be supported by the model is an issue we are investigating.

As a further example, to show the generality of our approach, we have also sketched how, choosing a different ground set S, it is possible to represent and manipulate information in statistical applications.

A theme worthy of investigation is a fully-parametric definition of the ADT Geometry(S). Even if the specification of the ADT Geometry(S) allows the parametric use of the ground domain without affecting the model we have defined, nevertheless we need to understand the implications of allowing a complex structure to the elements of the ground set S to model arbitrary complex entities.

Moreover, we are working on the theoretical properties of our model: a first result about completeness and soundness properties can be found in [33].

Acknowledgments—We thank Giorgio Gambosi for useful discussions. Insightful comments from one of the referees greatly helped in improving quality of presentation. This work has been partially supported by "Multidata" project within the Italian National Research Council's "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo" research programme and, "Esprit BRA Working Group No. 3191 "Basic Goods".

REFERENCES

[1] I. Bracken and C. Webster. Towards a typology of geographical information systems. *Int. J. Geograph. Inform. Syst.* 3 (1989).

[2] J. A. Orestein and F. A. Manola. Probe spatial data modeling and query processing in an image database application. *IEEE Trans. Software Engng* **14**, 611–629 (1988).
[3] R. H. Guting. Gral: an estensible relational database system for geometric application. *Proc. 15th VLDB*, Amsterdam (1989).
[4] N. Roussopoulos, C. Faloutsos and T. G. Sellis. An efficient pictorial database system for PSQL. *IEEE Trans. Software Engng* **14**, 639–651 (1988).
[5] S. Abiteboul, G. Shool, G. Gardarin and E. Simon. Towards DBMSs for supporting new applications. *Proc. VLDB*, Kyoto (1986).
[6] M. Scholl and A. Voisard. Modeling of thematic maps: an application to geographic databases. *Int. Symp. on the Design and Implementation of Large Spatial Databases*, Santa Barbara (1989).
[7] E. F. Codd. A relational model for large shared data banks. *Commun. ACM* **13**, 377–387 (1970).
[8] J. Schmidt. Type concepts for database defintion. *Proc. Int. Conf. on Data Bases*, Haifa, Israel (1979).
[9] F. Bancilhon and S. Khoshafian. A calculus for complex objects. *J. Comput. Syst. Sci.* **38** (1989).
[10] S. Abiteboul and N. Bidoit. Non first normal form relations: an algebra allowing data restructuring. *J. Comput. Syst. Sci.* **33** (1986).
[11] R. Hull and C. K. Yap. The format model: a theory of database organization. *J. Assoc. Comput.* **31** (1984).
[11a] G. Gardarin, J. P. Cheiney, G. Kirnan, D. Pastre and H. Stora. Managing complex objects in an extensible relational DBMS. *Proc. 15th VLDB*, Amsterdam (1989).
[12] F. Arcieri, L. Barella, P. Dell'Olmo, M. Gargano and E. Nardelli. CARTECH: a SQL-based geographical information system. *Proc. Convention Unix*. I2U, Milan (1991)
[13] B. Meyer. Reusability: the case for object-oriented design. *IEEE Software* **Mar** (1987).
[14] M. Stonebraker, B. Rubenstein and A. Guttman. Application of abstract data types and abstract indices to CAD bases. *IEEE Trans. Software Engng* (1983).
[15] S. L. Osborn and T. H. Heaven. The design of a relational database system with abstract data types for domains. *ACM Trans. Database Syst.* (1996).
[16] *Proc. Int. Conf. on Extending Data Base Technology*, Venice (1988).
[17] L. Rowe and K. Schoens. Data abstraction, views and updates in Rigel. *Proc. ACM-SIGMOD* (1979).
[18] P. Thomas, H. Robinson and J. Emmas, Abstract data types, their specification, representation and use. *Oxford Applied Mathematics and Computing Science Series*. Clarendon Press, Oxford (1988).
[19] S. Kamin. Final data types and their specification. *ACM Trans. Program. Lang. Syst.* **5** (1983).
[20] G. Ozsoyoglu, M. Ozsoyoglu and V. Matos. Extending relational algebra and relational calculus with set-valued attributes and aggregate functions. *ACM Trans. Database Syst.* **12**, 566–592 (1987).
[21] W. I. Grosky. Toward a data model for integrated pictorial databases. *Comput. Vis. Graph. Image Process* **25**, 371–382 (1984).
[22] F. Arcieri and P. Dell'Olmo. A data structure for the efficient treatment of topological join. *Fourth Int. Symp on Computer and Information Sciences*, Turkey (1989).
[23] O. Gunther. Efficient structures for geometric data management. *Lecture Notes in Computer Science*, Vol. 337. Springer-Verlag, New York (1988).
[24] P. Valduriez, S. Khoshafian and G. Copeland. Implementation techniques of complex objects, *Proc. 12th VLDB*, Kyoto (1986).
[25] Y. Chen, S. S. Iyengar and R. L. Kashyap. A new method of image compression using irreducible covers of maximal rectangles. *IEEE Trans. Software Engng* **14**, (1988).
[26] A. Henrich, H. W. Six and P. Widmayer. The LSD tree: spatial access to multidimensional point and non point objects. *Proc. 15th VLDB*, Amsterdam (1989).
[27] W. G. Aref and H. Samet. Efficient processing of window queries in the pyramid data structure. *Proc. PODS* (1990).
[28] J. Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, Palo Alto, CA (1988).
[29] M. Gargano and E. Nardelli. A logical data model for integrated geographical databases. *Proc. ACM–IEEE Int. Conf. on Systems Integration* (1990).
[30] G. Gambosi, E. Nardelli and M. Talamo. A conceptual model for the representation of statistical data in geographic information systems. *Proc. IV Int. Working Conf. on Statistical and Scientific Database Management*, Rome (1988).
[31] N. S. Chang and K. S. Fu, A relational database system for images. *Pictoral Information Systems* (S. K. Chang and K. S. Fu, Eds), pp. 288–321. Springer, New York (1980).
[32] N. S. Chang and K. S. Fu. Picture query languages for pictorial data-base systems. *Computer* **11**, 23–33 (1981).
[33] M. Gargano, E. Nardelli and M. Talamo. A model for complex data: completeness and soundness properties. *Proc. Int. Workshop on DBMSs for Geographical Applications*, Capri (1991).
[34] N. Roussopoulos and D. Leifker. Direct spatial search on pictorial databases using packed R-trees. *Proc. ACM SIGMOD* (1985).
[35] H. Samet. Hierarchical representation of collections of small rectangles. *ACM Comput. Surv.* **20** (1988).
[36] T. R. Smith, S. Menon, J. L. Star and J. E. Estes. Requirements and principles for the implementation and construction of large-scale geographical information systems. *Int. J. Geograph. Inform. Syst.* **1**, 13–31 (1987).