Multi-Scale Partitions: Application to Spatial and Statistical Databases*

Philippe Rigaux[1,2] and Michel Scholl[1,2]

[1] Cedric/CNAM, 292 rue St Martin, F-75141 Paris Cedex 03, France
[2] INRIA, BP 105, F-78153 Le Chesnay Cedex, France.
{rigaux,scholl}@asimov.cnam.fr

Abstract. We study the impact of scale on data representation from both the modelling and querying points of view. While our starting point was geographical applications, statistical databases also address this problem of data representation at various levels of abstraction. From these requirements, we propose a model which allows: (i) database querying without exact knowledge of the data abstraction level, (ii) the computation of multiple representations of data, one per abstraction level, and (iii) its application to the computation of statistical summaries. The model has been partially implemented with the DBMS O_2 by means of tree-structured domains: we give some examples which illustrate the above features.

1 Introduction

Multi-scale representation of data has raised attention for a long time in Geographic Information Systems (GIS), because of its major importance for cartographic representation [SM89, BM91, LR94, MLW95]. However, the *cartographic generalisation* process intended to convert spatial data from one scale-dependent representation to another may also be considered from an *abstract* point of view [BW88] in relation with non-cartographic requirements such as those of spatial analysis.

Abstract generalization is closely linked to *multiple representation* [BD89, Gup89, Jon91, vO91] within Database Management Systems (DBMS): object representation changes according to the level of abstraction at which data is represented (see [RS94] for a general overview of the multi-scale issues within GIS databases). In this paper we propose a model for multi-scale representation. We show that this model not only applies to spatial databases but also to statistical databases [Sho82, Gho86, RS90]. We restrict our attention to a simple but very common situation where data is defined as an *hierarchy of partitions* of a single space. The model we define allows to represent and query databases containing such hierarchies.

* Work partially supported by the French *CNRS GDR Cassini*.

We discuss different ways of implementing such hierarchies which use hierarchical atomic values domains. We show a few examples of queries using O_2SQL [BCD89], the query language of the object-oriented DBMS (ODBMS) O_2 [BDK91], and give some guidelines for query optimization in the presence of such hierarchies. In particular we show that by rewriting some queries one might avoid the expensive use of spatial indices, using instead some "logical" indexing mechanism as provided by the hierarchy of partitions.

Although motivated by applications where there is a strict decomposition of the space, this work is useful for a variety of situations beyond GIS and statistical data analysis. In particular hierarchical domains can solve problems related to cooperative query answering [Mot86, Mot88, CCL91]. We give an example where the user can query a database with incomplete knowledge of the domain of the attributes present in the query: the user does not know at which abstraction level data is represented.

Section 2 introduces the problem through examples. A model for multi-scale representation is proposed in Section 3. Section 4 shows how the model can be applied to solve the examples of Section 2 and simple cooperative query answering situations. In Section 5, an implementation with O_2SQL is illustrated.

2 Examples of Multi-Scale Databases

Spatial generalization: Figure 1.A shows three simple objects (land parcels). At a lower scale (Figure 1.B) we have a different view of land use: we can only tell apart cereals from flowers. As a consequence, space partitioning based on land use classification results in three or two objects depending on the scale[3].

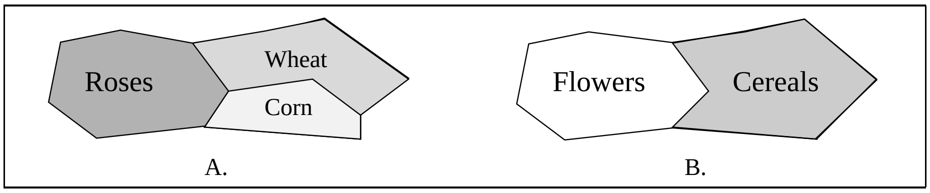

Fig. 1. : Impact of scale on spatial data

Statistical summaries: Table 1 displays a table of wine sales: for each wine (column) the sale in a given state (row) is given. Note that wines as well as states[4] are organized as *classification hierarchies* [RS90] with respective roots Bordeaux's wines and France. France is divided into administrative regions further split into states. Similarily, Bordeaux's wines are split into several production regions further split into single wines.

[3] The word *scale* will often be used in place of *level of abstraction* in the sequel.
[4] Such attributes are often denoted *category attributes* in the literature.

Wine sales			Bordeaux			
			Haut Médoc		St Emilion	
			St Julien	Pauillac	Lussac	Sables
France	Ile de France	Paris	1 500	870	550	700
		Yvelines	2 150	760	300	660
	Languedoc Roussillon	Gard	1 900	800	500	460
		Herault	1 100	670	330	230

Table 1: wine sales/states

Basically from Table 1, one would like to compute Table 2. In this table, sales are computed per administrative region and per production region. As a matter of fact, such a table is called a *summary* [Mal88, CMM88, dd94] of Table 1. Note that another *summary* of Table 2 would give the sale of Bordeaux per administrative region.

Wine sales		Bordeaux	
		Haut Médoc	St Emilion
France	Ile de France	5 280	2 210
	Languedoc Roussillon	5 140	1 750

Table 2: wine sales/region

Clearly from Tables 1 and 2, each *summary* is built on *tree-domain valued* category attributes represented in Figure 2 (A and B). More specifically, each *summary* includes for each category attribute only values from a given *abstract level* of the tree. $\{St\ Julien, Pauillac, Lussac, Sables\}$ is an abstract level. $\{Ht\ Medoc, St\ Emilion\}$, is another abstract level (Figure 2.A).

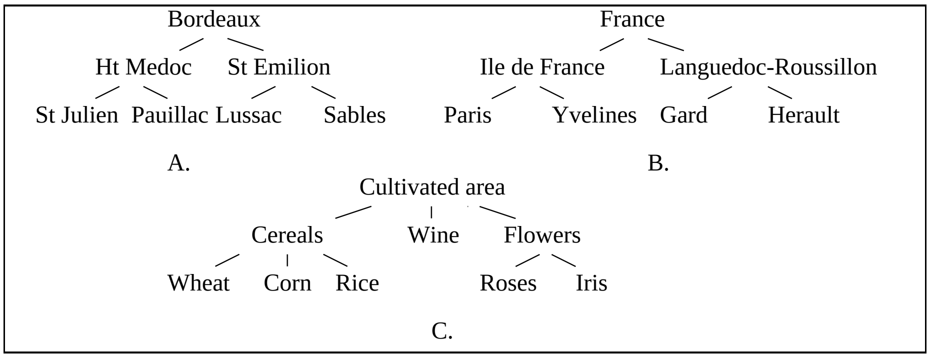

Fig. 2. : Tree-domain values

Land use (example 1) is also structured as a tree given in Figure 2.C: land use in Figure 1.A is taken among the leaves of the hierarchy in Figure 2.C, and simplified in a more abstract level including $\{Flowers, Cereals\}$ in Figure 1.B.

A common feature to the above spatial and statistical multi-scale examples is the *partition* of objects according to some *descriptive* attribute (land use, sale state, wine). In both cases, a *simplification* of these attributes values relative to a change of scale implies an *aggregation* of some other attribute (in our examples: *geometric union* or *integer addition*).

We end this section with examples of queries on multi-scale databases. Consider the relation schema Parcel (land-owner, countyId, Luse, Geo): France Counties are split into smaller areas called parcels according to land use. The land owner of the parcel is recorded as well as the county to which it belongs and the geometry of the parcel. We assume that there is no other parcel in the same county with same land owner and same land use (a parcel might include several geometric zones). Here are two simple queries on such a relation.

1. Display the map of land use.

2. Classify counties in more abstract areas with schema (Luse, Geo) where the land use is chosen among $\{Wine, Cereals, Flowers, ...\}$ and the countyId and land-owner attributes have been projected out.

To answer query 1, one has to project first on (Luse, Geo) then nest [AB86] on Luse and finally compute for each tuple the parcel geometric union of the geometries (deleting inter parcel borders). Similarly, to answer query 2, one has first to project out attributes land-owner and countyId, then to replace attribute Luse by its ancestor in the hierarchy belonging to a more abstract level of the tree (e.g. Corn is replaced by Cereals), and then nest and perform the (geometric) union.

We started with a database including the single relation Parcel. One might as well have started with a set of administrative partitions of the French territory (e. g. one abstraction level for regions and one for counties) and a representation for the land use hierarchy. A typical query on such a database would be to obtain for example a (thematic) map of the Aquitaine counties where only the wine production areas (St Emilion, Pauillac, etc...) are represented. Such a query requires some overlay between the map of counties and the map of land use.

The following section presents a data model for such multiple scale databases. *Tree-domains*, *abstract levels*, *objects partitions* and *summaries* are more formally defined as well as a few operations for allowing queries such as those above.

3 A Model for Multiple Scale Partitions

3.1 Preliminaries

Let E be a subset of the plane: $E \subset R^2$ and $P \in 2^E$ be a partition of E, i.e. $\bigcup_P = E$ and $\forall p, p' \in P$, $p \cap p' = \emptyset$. Let $\wp(E)$ be the set of partitions of E and $\preceq$

be the following partial order on $\wp(E)$: $P \preceq P'$ iff $\forall p \in P, p' \in P', p \cap p' \in \{p', \emptyset\}$. Let $U = \{A_1, ..., A_n\}$ be the universe of *descriptive* attributes defined on domains $D(A_1), ..., D(A_n)$ and let O be any relation with schema $\{A_i^1, A_i^2, ..., A_i^q, Geo\}$, with $q \in [1, n]$ attributes in U and one geometric attribute Geo with domain $D(Geo) = 2^E$. O is called *geographic* relation.

Definition 1: A *cover* on E is a geographic relation O such that:

1. the relational projection $\pi_{Geo}(O)$ is a partition of E: $\pi_{Geo}(O) \in \wp(E)$
2. $A_1, ..., A_n \rightarrow Geo$ and $Geo \rightarrow A_1, ... A_n$[5].

An *elementary* cover is a cover with a single attribute $A \in U$. Let $\Omega(E)$ be the set of covers on E (for short Ω).

3.2 Geometric Operations on Ω

Let *intersect* be the intersection predicate $Geo \times Geo \rightarrow Bool$ and *intersection* $Geo \times Geo \rightarrow Geo$ be the set intersection on the plane. The *overlay* or *geometric join*[6] between two geographic relations O, O' denoted $O \bowtie_{Geo} O'$ is defined as:

$$O \bowtie_{Geo} O' = \{[a_1, ...a_m, a'_1, ...a'_p, g] | a_1, ...a_m, geo \in O, a'_1, ...a'_p, geo' \in O', geo \\ intersect\ geo', g = intersection\ (geo, geo')\}$$

Let O be a cover with schema $S' = \{A'_1, ..., A'_m, Geo\}$ and $S = \{A_1, ..., A_q\}$ be any subset of the descriptive attributes of S'. The *geometric projection* of O on S denoted $\prod_{S,Geo}(O)$ or $\prod_S(O)$ when there is no ambiguity, is defined as:

$$\prod_S(O) = apply_{\sum_{Geo}}(nest_S(\pi_{S,Geo}(O)))$$

where $apply_f(O)$ applies f to some attribute of each tuple of O, $nest_S$ is the N1NF grouping operation [BRS82, AB86, SS86] on S and $\sum_{Geo}$: $\{Geo\} \rightarrow Geo$ is the (geometric) sum agregate function where $+$ is (overloaded with) the set union over E[7].

Theorem 1: Ω is closed under geometric projection and geometric join.

Proof (sketch): one easily shows [Rig95] that $\pi_{Geo}(\prod_S(O))$ is a partition of E and that there is a bijection between $O' = \prod_S(O)$ and $\pi_{Geo}(O')$. The same argument holds for geometric join.

Corollary: $\pi_{Geo} : \Omega \rightarrow \wp(E)$ defines an homomorphism. $\bigcap$ is homomorphic to geometric join where $\bigcap$ is the intersection operation in $\wp(E)$ defined as follows: let $P1$ and $P2$ be two partitions; then $P1 \bigcap P2 = \{p_1\ intersection\ p2 | p_1 \in$

[5] where $\rightarrow$ denotes functional dependency.
[6] Also called in the literature *Spatial Join*.
[7] $nest_S(\pi_{S,Geo}(O))$ has for a schema $\{A_1, ..., A_q, B\}$ where B is of type $set(Geo)$. $\sum_{Geo}$ is applied to the B-value of each tuple of $nest_S(\pi_{S,Geo}(O))$.

$P1, p_2 \in P2, p_1$ *intersect* $p_2\}$. We have: $\pi_{Geo}(O \bowtie_{Geo} O') = \pi_{Geo}(O) \bigcap \pi_{Geo}(O')$, and $\pi_{Geo}(\prod_{S,Geo}(O)) = h(\pi_{Geo}(O))$ where h is homomorphic to geometric projection.

Let $\leq$ be the partial order in Ω induced by $\preceq$: $O \leq O'$ iff $\pi_{Geo}(O) \preceq \pi_{Geo}(O')$. The following properties hold from the above:

Property 1: $\prod_S(O) \leq O$, $O \leq O \bowtie_{Geo} O'$, $O' \leq O \bowtie_{Geo} O'$.

Property 2: $\prod_{A_1,\cdots,A_q,Geo}(O) = \prod_{A_1,Geo}(O) \bowtie_{Geo} \cdots \bowtie_{Geo} \prod_{A_q,Geo}(O)$.

Two remarks are noteworthy:

1. The above model is based on geographic objects, i.e. on the partition of a geometric space. We could have chosen any other set to be partitioned as well. As an example, instead of E, we might have chosen a set of people.

2. For the sake of generality we shall call such a set (e.g. geometry or people) a *population*. Then assume a set of populations $E_1, \cdots, E_p$, and let $\wp(E_i), i \in [1,p]$ denote the set of partitions on E_i. Now let G_i be an attribute with domain $\wp(E_i), i \in [1,p]$. A *generalized* cover is a geographic relation O with schema $\{A_1, \cdots, A_q, G_1, \cdots, G_p\}$ such that $\forall i \in [1,p]$, $\pi_{G_i}(O)$ is a partition in $\wp(E_i)$ and there is a bijection between O and $\pi_{G_i}(O)$. Because of space limitations, we refer to [Rig95] for the definition of the operations of (population) join and projection in that case[8]. As an example, we might consider for O a set of counties with descriptive attributes (the county name and the state name) and with population attributes (the set of people living in this county[9]) and the geometry of the county. A state's population and geometry is obtained by *geometric projection* on the state name, the people and the geometry. We might as well have only kept for each state either the set of people or the geometry (projecting out the other population attribute). In the sequel, we shall assume for the sake of simplicity there is a single "population" attribute Geo defined on E.

3.3 Tree Structured Domains and Statistical Summaries

We assume that a descriptive attribute A takes its values on a finite Poset $(D(A), \trianglelefteq)$. Graph G_A associated with poset $(D(A), \trianglelefteq)$ is defined as: elements of $D(A)$ are nodes of G_A and there is an edge from a to b iff $a \trianglelefteq b$ and there is no c such that $a \triangleleft c \triangleleft b$. We restrict G_A to be a tree denoted T_A. A *cut* in T_A is defined as:

1. Let R be the root of T_A. Then $\{R\}$ is a cut;

[8] The (population) projection is slightly more intricate than above: it involves the N1NF union [AB86].

[9] We assume that a single person cannot be reported in several counties.

2. $sons(R)$ is a cut, where $sons(a)$ is the set of sons of node a in the tree;

3. let C be a cut and $v \in C$ such that $sons(v) \neq \emptyset$. Then $C' = (C - v) \cup sons\ (v)$ is a cut.

The following holds: (a) $\forall c, c' \in C$, c and c' are not comparable, and (b) $\forall c \in T_A$ but $\notin C$, there exists $c' \in C$ such that either $c \lhd c'$ or $c' \lhd c$. We define the following partial order on cuts of a given tree. $C \ll C'$ iff $\forall y \in C', \exists x \in C$ such that $x \unlhd y$. C is said to be more *abstract* than C'[10]. Figure 3 displays the set of cuts for the land-use tree domain of Figure 2.C, ordered by $\ll$. The most abstract cut is the root. $leaves(T_A)$, the set of leaves of T_A, is the least abstract cut (see Figure 3). A cut in the tree partitions the elements of $D(A)$ into two subsets $T_{A,<C} = \{e | e \in D(A), \exists c \in C, e \lhd c\}$ the set of elements "above" the cut and $T_{A,\geq C}$ the remaining elements. Let $v \in T_{A,\geq C}$. Then $Ancestor_C(v) \stackrel{\text{def}}{=} v' \in C | v' \unlhd v$. Let $v \in T_{A,<C}$, then $Descendants_C(v) \stackrel{\text{def}}{=} \{v' | v' \in C, v \lhd v'\}$.

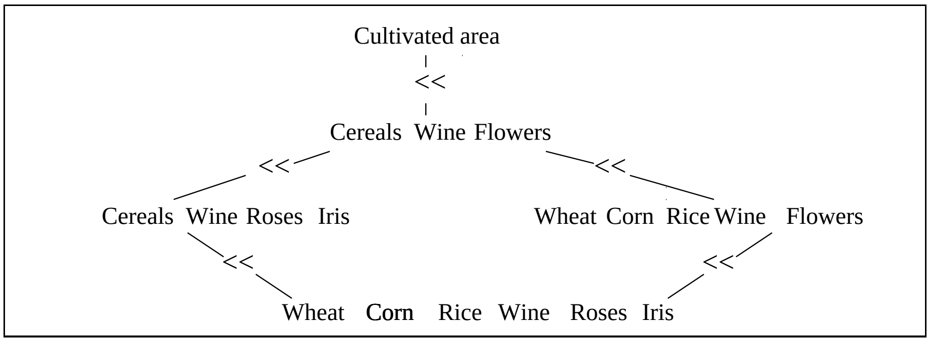

Fig. 3. : The cuts of the land-use tree domain of Figure 2.C

Interpretation of a set of cuts (on E): the interpretation of a set $\mathcal{C}$ of cuts on T_A is the mapping $I : \mathcal{C} \rightarrow 2^{\Omega_A}$ from $\mathcal{C}$ to the powerset of elementary covers defined on A (O has for a schema $[A, Geo]$) such that: $I(R)$ is the singleton set of covers $\{o\}$, where o is the relation with single tuple $[A = R, Geo = E]$ and for each cut $C \in \mathcal{C}$, $I(C) = O \in \Omega_A$ such that $\pi_A(O) = C$.

(Statistical) summaries: let O be a cover on E and A defined on an hierarchical domain $D(A)$ with tree T_A. We restrict Ω to the set of objects O such that $\pi_{A,Geo}(O) \in I(\mathcal{C})$. We call such objects *summaries*[11]. Let O be defined on cut C' and C be a more abstract cut than C': $C \ll C'$. Then we call *generalization of O up to C* denoted $gen_{A:C}(O)$ or $gen_C(O)$ when there is no ambiguity,

[10] A cut is a slice in a tree. It implements a *level of abstraction* as introduced in Section 2. Note that the values of a cut might not be at the same depth in T_A.

[11] Summaries may of course have other descriptive attributes: they are not necessarily elementary covers. These other attributes might be regular attributes or attributes also defined on a tree domain.

the expression: $gen_C(O) = apply_{A:Ancestor_C}(O)$, where in each tuple attribute A value a is replaced by its ancestor belonging to cut C. $gen_C(O)$ is no longer a cover in the general case, since there are several tuples with the same value for A and therefore $A_1, \cdots, A_q$ is not anymore a key for $gen_C(O)$. But the following expression:

$$apply_{\sum_{Geo}}(nest_{A_1 \cdots A \cdots A_q}(gen_{A:C}(O)))$$

denoted $summary_{A:C}(O)$ is a summary called *summary* of O on C.

Theorem 2: Ω (restricted to cuts) is closed under operation *summary*.

Proof: similar to that of Theorem 1 (see [Rig95]).

As an example look at the tree associated with the administrative splitting of the USA into states, counties, etc. Then a counties view of USA is a summary r with two attributes, ADM (whose value is a county name) and Geo (which describes the geometry of the county). ADM is defined on the "County" cut of the USA. Then let C be a more abstract cut on states. $summary_{ADM:C}(r)$ is a relation in which each tuple represents a state and has two attributes: the state name (ADM) and the state geometry (Geo). I induces an order on $I(\mathcal{C})$: $r1 \leq r2$ iff $C_1 \ll C_2$ where $r1$ ($r2$) is a summary of O on C_1 (C_2). We have also: $r1 \leq r2$ iff $\pi_{Geo}(r1) \preceq \pi_{Geo}(r2)$. Here are some properties on operations *gen* and *summary*:

Property 3: $gen_{A:C}(O \bowtie_{Geo} O') = gen_{A:C}(O) \bowtie_{Geo} O'$ where we assume that A is an attribute of cover O.

Property 4: $summary_{A:C}(O) = summary_{A:C}(\prod_{A,Geo}(O)) \bowtie_{Geo} \prod_{S-A}(O)$, where O has for schema S.

Property 5:
$summary_{A_1:C_1}(summary_{A_2:C_2}(O)) = summary_{A_2:C_2}(summary_{A_1:C_1}(O)) \stackrel{\text{def}}{=} summary_{A_1:C_1,A_2:C_2}(O) \stackrel{\text{def}}{=} apply_{\sum_{Geo}}(nest_{A_1A_2}(gen_{A_1:C_1,A_2:C_2}(O)))$ where $gen_{A_1:C_1,A_2:C_2}(O) \stackrel{\text{def}}{=} gen_{A_1:C_1}(gen_{A_2:C_2}(O)) = gen_{A_2:C_2}(gen_{A_1:C_1}(O))$

This property states that (i) generalization on separate attributes is commutative, so is summary, and (ii) that one can save one (expensive) *nest* operation and one *apply* operation by doing at once all generalizations.

4 Applications of the Model

We shall first solve the queries of Section 2 by using the operations defined in Section 3. Consider first tables 1 and 2. Table 1 can be implemented by a relation with schema: WINESALE1(wine,area,amount), where wine is defined on the cut containing the leaves of the tree domain for wines, area is a state (belonging to the leaves cut of the administrative areas tree domain) and amount is the sale amount. Let CW be the more abstract cut {Haut Medoc, St Emilion} and CR be the cut of administrative regions. Then the *summary* in Table 2 is a relation WINESALE2 with same schema as WINESALE1 and equal to:

$$WINESALE2 = summary_{Wine:CW,Area:CR}(WINESALE1)$$

The second example we considered in Section 2 was that of Parcels (land-owner, countyId, Luse, Geo), where for each tuple land use is taken in the leaves cut of the hierarchy in Figure 2.C. The land use cover LUSE (Luse, Geo) is obtained by (geometric) projection $\Pi_{Luse,Geo}(Parcels)$.

$$summary_{Luse:CL}(\pi_{countyId,Luse,Geo}(Parcels))$$

computes a more abstract representation of Parcels on the T_{Luse} cut CL: {Cereals, Wine, Flowers,...}(see Figure 2.C).

Let ADMIN(admId, Geo) be a cover recording France (administrative) regions. The following expression computes the cover representing wine production areas in the Aquitaine region:

$$\sigma_{admId='Aquitaine'}(ADMIN) \bowtie_{Geo} \sigma_{Luse='Wine'}(LUSE).$$

We assume now that ADMIN is not anymore the administrative region cover but the counties cover (adminId is defined on the counties cut CC), and that LUSE is a cover defined on the cut CW {Ht Medoc, St Emilion}. Assume that we look for the administrative region where "St Emilion" is produced.

$$\sigma_{Luse='StEmilion'}(gen_{admId:CR}(ADMIN \bowtie_{Geo} LUSE))$$

computes such an answer. Indeed, we first overlay the covers LUSE and ADMIN, we then generalize the result to administrative regions (admId will take its values on the cut of regions) and finally make a selection on Luse. This expression is of course equivalent to: $gen_{admId:CR}(ADMIN) \bowtie_{Geo} \sigma_{Luse='StEmilion'}(LUSE)$, by pushing selection inside and using property 3.

The following expressions give the counties where 'St Emilion' is produced. The first expression assumes that ADMIN is the counties cover (defined on cut CC). The second one assumes ADMIN is the (administrative) regions cover and $Desc_{C'}(R)$ is defined as $\{c \in C' | gen_{A:C}(c)\ in\ R\}$ where attribute A of cover R is defined on cut C (more abstract than cut C').

$$ADMIN \bowtie_{Geo} \sigma_{Luse='StEmilion'}(LUSE)$$
$$DESC_{CC}(ADMIN \bowtie_{Geo} \sigma_{Luse='StEmilion'}(LUSE)) \bowtie_{Geo}$$
$$\sigma_{Luse='StEmilion'}(LUSE)$$

The second expression introduces a redundancy. It suggests to first select the regions which intersect the areas where St Emilion is produced (geometric join), then take all counties inside such regions ($DESC_{CC}$), and finally only for those counties perform the (geometric) join. The latter expression might provide a better performance depending on the implementation. Indeed instead of using for filtering a (physical) spatial index on counties expensive in terms of design and maintenance, it is suggested that the query optimizer chooses faster access paths provided by the semantics of the hierarchies application. Finally assume again ADMIN(admId, geo) is defined on the cut CC of counties and that in LUSE, Luse is defined on a cut {Pauillac, St Julien,...}. Then the query "Counties where 'Ht Medoc' is produced" is answered by the following expression:

$$ADMIN \bowtie_{Geo} \sigma_{Luse='HtMedoc'}(gen_{Luse:CW}(LUSE))$$

Such a query would provide a null answer with regular database query languages which require a complete knowledge of the attribute domain. In a companion paper [RS94] we suggested the use of attributes defined on tree structured domains to answer such queries, but without requiring a semantics as strong as that of Section 3 (without requiring the relations to be covers). Several authors have addressed this issue in a more general context known as "Cooperative Query Answering" which deals with syntactically correct queries which provide a NULL answer. For example, the *Type Abstraction Hierarchy* of [CCL91] combines an hierarchy similar to ours with object-oriented class hierarchies. [Mot86, Mot88] also allow some query generalization.

5 Implementation with O_2SQL

This model has already been partially implemented with the DBMS O_2 [Rig95, PR95]. We illustrate such an implementation by means of a few examples from Section 4. There are several ways of implementing a set of ordered covers. One simple way is to explicitly represent the usual covers by O_2 roots of persistence [BDK91], each object of a cover being explicitly linked to its "sons" in a less abstract cover and to its "father" in a more abstract cover. This is the case for the representation of administrative partitions of a country. Basically there would be the following roots of persistance:

Regions: set (Region), Counties: set(County)

defined on the classes Region and County with structure:

```
Class Region tuple (regionId:String,
                    counties:set(County),
                    geo : Geo)
```

```
Class County tuple(countyId: String,
                   region:Region,
                   geo:Geo)
```

It is not clear whether the representation of several land use covers with explicit links would be useful or realistic since the choice of a cut in the land use tree domain is highly dynamic and dependent on the application or the query. Instead it is more realistic to represent a useful cover for land use (e.g. that including values such that "St Emilion", "Haut Medoc", etc.), and to implement the tree structured domain of landuse as follows[12]:

```
Class Landuse tuple (luse: String,          Class Luse tuple (luse: String,
                     geo : Geo)                               sons: set(Luse),
                                                              father: Luse)
name Luses: set (Landuse)                   name Treeluse: Luse
```

In the above, Luses is the database entry for the cover of land use, while TreeLuse is the root of the land use tree. Because of space limitations we shall only present the implementation of three among the queries of Section 4.

query 1: administrative region where "St Emilion" is produced. There are two ways of answering this query (one associated with each equivalent expression, see Section 4).

```
(i) select r
    from   r in Regions, c in r.counties, l in Luses
    where  c.geo intersect l.geo
    and    l. luse = 'St Emilion'
```

We keep the counties whose geometry intersects St Emilion's area geometry and we follow an explicit path in the composition hierarchy. Obviously one will prefer the following expression. But the above expression would be the only way to answer the query if the geometry were only stored at the county level (and not at the region level).

```
(ii) select r
     from   r in Regions, l in Luses
     where  r.geo intersect l.geo
     and    l. luse = 'St Emilion'
```

Checking only whether a region intersects the St Emilion's area is of course much faster than checking all counties since the number of regions is by several orders of magnitude smaller than the number of counties and since a single intersection test costs the same price whether it is a region or a county (in both cases the geometry is represented with a polygon of same complexity: the number of points of the polygon is of the same order). The optimizer should take into account property 3 (see Section 3.3) to rewrite query (i) into its equivalent form (ii).

query 2: Counties producing 'St Emilion'. For the same reasons the following redundant query takes advantage of the semantics of partitions to accelerate access to the relevant counties. It saves the use of a spatial index on counties.

[12] in [RS94] it is suggested to extend the O_2 data model by a poset constructor Poset($\preceq$, τ) which allows to represent a finite set of elements of type τ partially ordered by $\preceq$. This constructor is supposed to be equipped with several functions. This extension would allow a performant implementation of functions on posets.

```
select  c
from    r in Regions, c in r.counties, l in Luses
where   r.geo intersect  l.geo
and     c.geo intersect  l.geo
and     l.luse = 'St Emilion'
```

query 3: Counties where 'Ht Medoc' is produced. This query is similar to the previous one except for the last clause that selects all wines whose ancestor is 'Ht Medoc' in the tree with (named) root TreeLuse:

```
select  c
from    r in Regions, c in r.counties, l in Luses
where   r.geo intersect  l.geo
and     c.geo intersect  l.geo
and     TreeLuse→IsAncestor(l.luse,'Ht Medoc')
```

Assuming as in [RS94] an extension of O_2 with a Poset constructor, a root of persistance TreeLuse: Poset (Luse) for implementing the landuse hierarchy and a binary predicate $\preceq$ testing whether a node is an ancestor of another one in a given poset, the above query would be rewritten as:

```
select  c
from    r in Regions, c in r.counties, l in Luses, l' in Treeluse
where   r.geo intersect l.geo
and     c.geo intersect l.geo
and     l'.luse='Ht Medoc'
and     l'.luse ⪯ l.luse
```

Acknowlegments. We thank Gérard d'Aubigny for helpful discussions within the multi-scale representation group of the GDR Cassini. We are also indebted to Claudia Medeiros, Jean-Marc Saglio and the anonymous referees for their careful readings which significantly improved the quality of the paper.

References

[AB86] S. Abiteboul and N. Bidoit. Non First Normal Form Relations: an Algebra Allowing Data Restructuring. *Journal of Computer and System Sciences*, 1986.

[BCD89] F. Bancilhon, S. Cluet, and C. Delobel. A Query Language for an Object-Oriented Database System. In *2nd Int. Worshop on Database Programming Languages (DBPL)*, pages 301–322, 1989.

[BD89] B. P. Buttenfield and J. Delotto. Multiple Representation: Report on the Specialist Meeting. Technical Report 89-3, National Center for Geographic Information and Analysis, Santa Barbara, 1989.

[BDK91] F. Bancilhon, C. Delobel, and P. Kannelakis, editors. *The O_2 Book*. Morgan Kaufmann, 1991.

[BM91] B.P. Buttenfield and R. P. McMaster. *Map Generalization: Making Rules for Knowledge Representation*. Longman Scientific&Technical, New-York, 1991.

[BRS82] F. Bancilhon, P. Richard, and M. Scholl. On Line Processing of Compacted Relations. In *8th Int. Conference on Very Large Data Bases (VLDB)*, Mexico, 1982.

[BW88] K. E. Brassel and R. Weibel. A Review and Conceptual Framework of Automated Map Generalization. *Int. Journal of GIS*, 2(3):229–244, 1988.

[CCL91] W. W. Chu, Q. Chen, and R. Lee. Cooperative Query Answering via Type Abstraction Hierarchy. In S. M. Deen, editor, *Cooperating Knowledge Based Systems*, pages 271–292. Elsevier Science Publishing Co. Inc, 1991.

[CMM88] M. C. Chen, L. McNamee, and M. Melkanoff. A Model of Summary Data and its Application in Statistical Databases. In *Int. Conference on Statistical and Scientific Database Management (SSDBM)*, pages 356–387, Rome (Italy), 1988. LNCS No. 339, Springer-Verlag.

[dd94] C. d'Aubigny and G. d'Aubigny. Agrégation Spatiale et Résumés Statistiques. *Revue de géomatique*, 4(3-4):307–336, 1994.

[Gho86] S. Ghosh. Statistical Relational Tables for Statistical Database Management. *IEEE Transactions on Software Engineering*, 12(12):1106–1116, 1986.

[Gup89] S. G. Guptill. Speculations on Seamless, Scaleless Cartographic Data Bases. In *Auto-Carto 9*, pages 436–443, 1989.

[Jon91] C. B. Jones. Database Architecture for Multi-scale GIS. In *ASPRS/ACSM*, pages 1–14, Baltimore, 1991.

[LR94] J. P. Lagrange and A. Ruas. Data & Knowledge Modelling for Generalization. In *Int. Conference on Spatial Data Handling (SDH)*, pages 1099–1117, Edinburgh, 1994.

[Mal88] F. M. Malvestuto. The Derivation Problem for Summary Data. In *Int. Conference on Management of Data (ACM-SIGMOD)*, pages 82–89, Chicago, 1988.

[MLW95] J. C. Muller, J. P. Lagrange, and R. Weibel, editors. *GISs and Generalization*. Taylor & Francis, London, 1995.

[Mot86] A. Motro. Query Generalization: A Method for Interpreting Null Answers. In *Int. Workshop on Expert Database Systems*, pages 597–616, 1986.

[Mot88] A. Motro. VAGUE: a User Interface to Relational Databases that Permits Vague Queries. *ACM Transactions on Office Information Systems*, 6(3):187–214, 1988.

[PR95] J. P. Peloux and P. Rigaux. A Loosely Coupled Interface to an Object-Oriented Geographic Database. In *Spatial Information Theory (COSIT)*, 1995. To appear.

[Rig95] P. Rigaux. Interfaces Visuelles et Représentation Multiple dans les Bases de Données Spatiales. Thèse de doctorat CNAM, 1995.

[RS90] M. Rafanelli and A. Shoshani. STROM : a Statistical Object Representation Model. In *Int. Conference on Statistical and Scientific Database Management (SSDBM)*, pages 14–29, Charlotte (USA), 1990. LNCS No. 420, Springer-Verlag.

[RS94] P. Rigaux and M. Scholl. Multiple Representation Modelling and Querying. In *Int. Workshop on GIS (IGIS'94)*, pages 59–69, Ascona, Switzerland, 1994. LNCS No. 884, Springer-Verlag.

[Sho82] A. Shoshani. Statistical Databases: Characteristics, Problems, and Some Solutions. In *Int. Conference on Very Large Databases (VLDB)*, Mexico, 1982.

[SM89] K. S. Shea and R. Macmaster. Cartographic Generalization: When and How to Generalize. In *Auto-Carto 9*, pages 56–67, 1989.

[SS86] H. J. Schek and M. H. Scholl. The Relational Model with Relation-Valued Attributes. *Information Systems*, 1986.

[vO91] P. van Oosterom. The Reactive Tree: A Storage Structure for a Seamless, Scaleless geographic Database. In *Auto Carto10*, pages 393–407, 1991.