ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
CONSIGLIO NAZIONALE DELLE RICERCHE

E. Pourabbas, M. Rafanelli, F.L. Ricci, F. Ferri

LINKING GEOGRAPHIC AND MULTIDIMESIONAL ENVIRONMENTS BY OLAP OPERATORS

R. 520 Dicembre 1999

Elaheh Pourabbas - Istituto di Analisi dei Sistemi ed Informatica del CNR, viale Manzoni 30 - 00185 Roma, Italy. Email : pourabbas@iasi.rm.cnr.it.

Maurizio Rafanelli - Istituto di Analisi dei Sistemi ed Informatica del CNR, viale Manzoni 30 - 00185 Roma, Italy. Email : rafanelli@iasi.rm.cnr.it.

Fabrizio L. Ricci - Istituto di Studi sulla Ricerca e la Documentazione Scientifica del CNR, via C. De Lollis 12 - 00185 Roma, Italy. Email : ricci@isrds.rm.cnr.it.

Fernando Ferri - Istituto di Studi sulla Ricerca e la Documentazione Scientifica del CNR, via C. De Lollis 12 - 00185 Roma, Italy. Email : ferri@isrds.rm.cnr.it.

ISSN: 1128-3378

2.

Abstract

Geographic Databases (GDB) facilitate the storage and manipulation of either geographic data and related attributes by graphic display functionalities and complex data structures. Sometimes queries expressed in GDB could refer to concepts which are abstraction of the reality and are represented through the dimensions in Multidimensional Databases (MDDB).

This paper presents a conceptual approach to allow the end user, working in GDB environment, to use data cube stored in MDDB. In this context, we need to extend the geographic data structure with some special "functional attributes". They support links between environments mentioned above through geographic dimensions always implicitly or explicitly present in MDDB. Therefore, the main objective of this paper is to propose a solution to answer queries involving data stored in both environments in a transparent way to the user. Then, a query language to support the integration of multidimensional operators with geographic operators is proposed. Finally, the main characteristics of the proposed approach are illustrated through some examples.

Keywords: Geographic and Multidimensional Databases, Full Contains and Full Containment Functions, Functional attributes, Query Language

1. Introduction

There are many advantages that the large community of users gain when connecting various data sources which can be resumed in the tremendous increase of the quantity of available data which allow enterprises to achieve higher competivity. The identification of unusual trends in particular applications can be discovered through the analysis of a large amount of data, suggesting opportunities for new business or for forecasting production needs. In this field, decision support systems that treat data in very large databases recently attracted research attention. These databases may represent business information (such as transaction data), medical information (such as patient treatments and outcomes), scientific data (such as large sets of experimental measurements), or spatial information (such as geographic data and the relative visualization such as maps).

Through graphic display functionalities and complex data structures, Geographic Databases (GDB) facilitates the storage and manipulation of either geographic data and related attributes or data which refer to the phenomena of interest.

The new challenge is to face the vast multiplication of data sources and quantity, and to link this data to adequate GDBs. A feature notably lacking in most GDBs is related to the capability for accessing and manipulating business data. Often business data is modelled as multidimensional data and stored in Multidimensional Databases (MDDB) in which analyses of transaction based business data (On-Line-Analytical-Processing, OLAP) [Sho97] is carried out. In this context, effective decision making involves integrating information from MDDB with GDB, all within a common framework of geographic data structure. The common key element for interfacing the two environments is the *geographic dimension*, which is always present implicitly or explicitly in MDDB. Since MDDB models have such constraints as "dimensions are linguistic categories corresponding to different ways of viewing the information", then each location dimension is a simple concept of hierarchy.

The paper's main aim is to treat the location dimension from a geographical point of view. This causes to view a simple geographic dimension of MDDB in a complex context of the associated concept in GDB. In this way, an instance (e.g.; Los Angels) of a geographic dimension (e.g.; Municipality) in MDDB, is treated as an instance of the geographic class *Municipality* with all its properties due to an approach that makes the kernel of this paper.

To motivate our proposal, let us consider the following example. A GDB user may like to ask to display the query result on a map by different combinations of geographic and non geographic data. Let the query be "Find the Italian regions adjacent to Tuscany, in which the number of cars sold in 1990, in the case of <Corolla>, was greater than 10,000". It needs not only to perform some OLAP operations along the "time" and "product" dimensions of the relative "Car_Sales" cube, but also to analyse the spatial relationships between regions. The latter refers to the performance of a well known geographic operator -adjacency- that can be performed only in GDB.

In this paper, we propose a model and a relative query language for both GDB and MDDB in order to support queries like the one mentioned above. The main contributions of this paper are the following:

Characterization We give some basic definitions for characterizing geographic data to support multidimensional data. We extend the notion of *Contains* relationship between geographic classes and objects to *Full-Contains* relationship. We examine the correspondence between this function and the analogue one in SDB in order to apply the OLAP and geographic operations. This allows to treat both environments indifferently.

Extension We present an approach that extends the geographic data structure through *functional attributes*. These attributes describe all the phenomena (e. g., "production", "sales",

etc.) represented by the SDB. We discuss some important aspects of the proposed approach when we deal with one, two or more geographic variables in multidimensional data. We study how a geographic class *deduces* the functional attributes from those of one that belongs to a higher level of the *ISA* hierarchy [SS77, KA90] with respect to the former. Then, we introduce the concept of *ISD* (IS Derived) relationship which represents how the functional attributes are resulted along the ISA hierarchy. It implies that a given cube in MDDB generates a set of functional attributes, each of them belongs to a given geographic class of ISA hierarchy, and it is generated from the above mentioned cube by OLAP operators.

Querying We can ask more specific "OLAP based" queries within GDB without incurring a significant complication.

The main contribution of the previous steps comes from the fact that the potentiality of the OLAP operators as well as all the results obtained by their application on the multidimensional data can be imported in the GDB. This fact permits to treat homogeneously, in the GDB, both geographic data and multidimensional data. We describe the proposed approach only at a conceptual level, releasing it from the relative logical and physical implementation. This permits the generality of the approach.

The paper is structured as followed: Section 2 refers to the most relevant notions of the multidimensional data needed to support the intended approach; Section 3 explains how to link the two above mentioned environments; Section 4 describes the extension of the geographic classes by the functional attributes and discusses two different cases when, in a data cube, one, two or more geographic variables are present; Section 5 shows the way to deduct the functional attribute; Section 6 illustrates a query example, and finally, Section 7 gives a brief summary and conclusion.

2. Background

In this section, we describe some basic notions of multidimensional data and geographic data needed for the description of the proposed approach. In the following we will briefly describe the basic notions on multidimensional and geographic data.

2.1. Multidimensional data

Multidimensional data refer either to statistical data [Mic91, Raf91], which mostly represent applications in the socio-economic area, or OLAP data [GBL96, GL97, Sho97], which emphasize business applications. In the OLAP area the conceptual representation of multidimensionality by *cube* was proposed (e.g., [AGS97]).

A cube is "a group of data cells arranged by the dimensions of the data" [Ola97]. A *dimension* is "a structural attribute of a cube, that is, a list of members, all of which are of a similar type in the user's perception of the data" [Ola97]. The set of cube dimensions represents the relative data multidimensionality.

Dimensions have often been associated with different hierarchically organised levels. These levels correspond to different granularities of viewing data. The name of each level is expressed by the corresponding *variable* name. In the following, we will use the terms level and variable indifferently.

Generally, the shift from a lower (more detailed) level to a higher (more aggregated) level is carried out by mapping. Mapping between two variables defines a *complete containment function* (denoted by the symbol $\rightarrow$) if this mapping is full, that is:

a) each variable instance of a lower level corresponds to only one variable instance of a higher level;

b) each variable instance of a higher level corresponds to at least one variable instance of a lower level.

A *measure* is a particular dimension of a cube [AGS97], which represents the extensional fashion of the phenomenon described by the cube, and which is, in general, a numeric value. Assigning a value to each dimension of a cube, the measure is obtained by *mapping* from this assignment. In this paper all the considered measures are summable, thus we respect one of the three necessary conditions for summarizability of Statistical Database [LS97].

If a dimension of a cube has associated levels organized hierarchically, then the above mentioned mapping is full for this dimension in this cube if and only if all the values of the measure exist (i.e., only if no cell contains the value "Not available"). The complete containment function respects the summarizability conditions (disjointness and completeness) of Statistical Databases described in [LS97, RS90]. As known in literature, a hierarchy is intentionally represented by a partial ordered set. Then, a *classification hierarchy* is any subset which defines a total order.

The operators used in this context have been already defined in [RR93] and are: Summarization, Restriction and Classification. These operators correspond to the following OLAP operators [CCS93]: Slice, Dice and Roll-up. In this paper we will use these last terms mentioned. For these operators we will refer to a function f that identifies the measure value from mapping of values assigned to each dimension of a given cube. The function f then denotes a cube.

Briefly, the *roll-up* operator decreases the detail of the measure, aggregating it along the dimension hierarchy. Let us consider a given hierarchy defined by the *complete containment function* $l_1 \rightarrow l_2$. For this operator, applied to the cube denoted by the function f, we use the symbolic representation: $roll-up_{l_1 \rightarrow l_2}^{sum(f)}(f)$. The *slice* operator omits one dimension of the cube. Let d_1, d_2, and d_3 be the dimensions of a given cube denoted by the function f. The slice on d_2, and d_3 represented through the symbol $slice_{d_2,d_3}^{sum(f)}(f)$ omits the dimension d_1. Note that when the dimension d_1 consists of a single value this operator is simplified as $slice_{d_2,d_3}(f)$. The *dice* operator restricts the dimension value domain of the cube to the defined values in the operator, in order to restrict the set of data retrieved. It is represented through $dice_e(f)$ where e is a built-in predicate.

2.2. Geographical data

The evolution of the geographic data model is characterised to represent information moving from the relational model towards the object oriented data model. An object-oriented model allows a more natural approach to define, represent, and manage object hierarchies (sub-classing), aggregation hierarchies, composition hierarchies, etc. [SV92].

Briefly, a *geographic class* is the set of geographic objects (instances) which have the same properties (*attributes*) and behaviour (*methods*). There is a geometric attribute (called *geom*); it can be one of the three different types of spatial object configurations, called *geo-features*, i.e., *geo-point, geo-polyline* and *geo-region* [FMR99]. A *Geo Entity* class is the superclass of all the geographic object classes. It is an *abstract class* and then it is not used to instance the objects but to create subclasses which inherit the attributes and the methods of the *Geo Entity* superclass. The classes are organized in a class-subclass hierarchy, called relation *ISA* using the concept of *single inheritance* of the paradigm. The classes and the instances are linked by the "*is-an-instance-of*" relationship.

The *methods* correspond to a set of operations which are based on the geometric, topological, and metric characteristics of the geographic objects. Moreover, in this paper we refer to three topological relationships which are *Geo-Union*, *Geo-Disjunction*, and *Geo-Touching* defined in [FMR99].

6.

There exist different types of relationships between geographic objects, such as the containment relation, which concept will be used in the following.

3. The link formulation between geographic and multidimensional data

Our approach defines a bijective mapping (through a function called γ) between geographic classes defined in a GDB and geographic variables defined in a MDDB. This mapping process implies that each geographic variable in MDDB corresponds to only one geographic class in the GDB, and vice versa. Therefore, each geographic class instance corresponds only to one instance of the geographic variable with the same name in MDDB, and vice versa. This assumption implies either that the geographic class/instance in GDB or the geographic variable/instance in MDDB refer to a unique geographical abstract concept. Note that given a generic instance of a geographic class, we deduce that by function γ the corresponding instance of geographic variable is then an instance of the geographic variable that is mapped to the above mentioned geographic class; and vice versa.

In the description below, we will focus only on objects with geo-region configuration. One of the well known spatial relationships between geo-regions is called *Is-in* . We will consider the dual of this relationship called *Contains*. The Contains relationship between a pair of geographic classes, named gc_1 and gc_2, indicates that the instances of the class gc_1 are included in the instances of gc_2. This implies that the geo-feature of gc_1 are internal to or on the boundary of the geo-feature of gc_2. For instance, at the intentional level REGION Contains MUNICIPALITY, and at the extensional level Tuscany *Contains* Florence, and Lazio *Contains* Roma.

A *Full-Contains* relationship between a pair of geographic classes gc_1 and gc_2 exists so that gc_2 Contains gc_1 if the union of the instances of the class gc_2 constitutes a complete closure of the union of the instances of the class gc_1; the instances of the class gc_1 are disjoined among them as well as the instances of gc_2 are disjoined among them.

We give the condition for the existence of the containment relationship *Full-Contains*.

Definition 3.1: Let gc_α and gc_β be two geographic classes. Between them the *Full-Contains* relationship exists (and we write gc_α *Full-Contains* gc_β) if:

- $\forall\, go_j$: go_j *is-an-instance-of* gc_β $\Rightarrow \exists\, !\ go_i$: ($go_i$ *is-an-instance-of* gc_α) and (go_i *Contains* go_j);
- $\forall\ go_i$: go_j *is-an-instance-of* gc_α $\Rightarrow \exists\ go_j$: ($go_j$ *is-an-instance-of* gc_β) and (go_i *Contains* go_j);
- $\forall\, go_{j1}$, go_{j2}: go_{j1} *is-an-instance-of* gc_β, go_{j2} is-*an-instance-of* gc_β, $go_{j1} \neq go_{j2}$, $\Rightarrow$ go_{j1} *Geo-Disjunction* go_{j2};
- $\forall\, go_i$: go_i *is-an-instance-of* gc_α $\Rightarrow \exists \{go_{j1}, \ldots, go_{jn}\}$: go_{j1} *is-an-instance-of* gc_β, ... , go_{jn} *is-an-instance-of* gc_β, go_j *Contains* go_{j1}, ... , go_j *Contains* go_{jn} : $go_i \equiv go_{j1}$ *Geo-Union* go_{j2} .... *Geo-Union* go_{jn}.

The *Full-Contains* relationship satisfies the conditions of the summarizability (disjointness and completeness) of Statistical Databases described in [LS97, RS90]. We note that the *complete containment function* and the *Full-Contains* relationship describe the same condition in two different ways: *Full-Contains* in a topological manner (for GDB environment), and the *Complete-Containment Function* in a linguistic categorical manner (for MDDB environment).

Property: The *Full-Contains* relationship is a partially ordered relation, i.e., it satisfies the following properties:

- *reflexivity:* Let gc be a geographic class, gc *Full-Contains* gc.
- *antisymmetry*: Let gc_α, gc_β be geographic classes where $gc_\alpha \neq gc_\beta$. If gc_α *Full-Contains* gc_β then gc_β *Full-Contains* gc_α is not satisfied.
- *transitivity:* Let $gc_\alpha, gc_\beta, gc_\delta$ be geographic classes where $gc_\alpha \neq gc_\beta \neq gc_\delta$. If gc_α *Full-Contains* gc_β and gc_β *Full-Contains* gc_δ then gc_δ *Full-Contains* gc_β.

We introduce an example that makes use of the above mentioned concept.

Example 3.1: Let the geographic classes MUNICIPALITY, PROVINCE, and REGION in the GDB be subclasses of the geographic class Geo Entity defined by the following scheme:

CLASS = Geo Entity
attributes
name: string
geom: geo-region
surface: real

and between them exist the *ISA* relationships (see Figure 3.1). The *Full-Contains* relationships between the geographic classes are defined in Figure 3.1. ◊

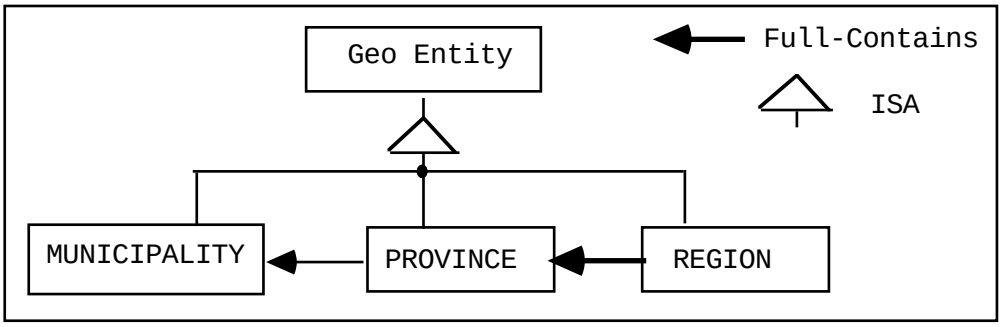

Figure 3.1. A scheme of GDB with Full-Contains relationships

The following theorems introduce mapping between the complete containment function defined on geographic variables in MDDB and the Full-Contains relationship defined on each geographic class in GDB. This mapping process allows us to extend all those properties in order to apply the OLAP and geographic operations when we move from one environment to another. In fact, they are introduced to explain how the lacking of information in one environment can be built up by the information of another environment. This occurs by the above mentioned bijective function γ.

Theorem 3.1: Let gc_α and gc_β be two geographic classes in a GDB, between which *Full-Contains* relationship (gc_α *Full-Contains* gc_β) exists. Let gv_α and gv_β be two geographic variables in a MDDB. If gc_α (gc_β) is mapped to gv_α (gv_β) by the function γ, then the *complete containment function* from gv_α to gv_β exists, and it is unique.

8.

Proof. Let go_α and go_β be, respectively, an instance of gc_α and gc_β, such that between them the *Full-Contain* relationship exists. Applying the bijective function γ to these two instances, i.e. $\gamma(go_\alpha) = giv_\alpha$, and $\gamma(go_\beta) = giv_\beta$, we obtain a generic pair $< giv_\alpha, giv_\beta >$. The relationship between each pair of such geographic variables instances obtained by the application of the function γ onto the instances of gc_α *Full-Contains* gc_β, is a total and surjective function, i.e. that has the same properties of the complete containment function.
We proof that the complete containment function is unique. We show the contrapositive.
Let the relationship $< giv_\alpha, giv_\beta >$ represents complete containment function. We assume there exists another complete containment function that maps another geographic variable instance giv'_α (instead of giv_α) to giv_β. It indicates that the geographic instance corresponding to giv'_α, and obtained by the function $\gamma^{-1}(giv'_\alpha) = go'_\alpha$ is mapped to $\gamma^{-1}(giv_\beta) = go_\beta$, between which Full-Contains relationship exist, and in the same time the same relationship exist between go_α and go_β. It means that to two different pairs $< giv_\alpha, giv_\beta >$, $< giv'_\alpha, giv_\beta >$ of two different complete containment functions correspond different instances $< go_\alpha, go_\beta >$, $< go'_\alpha, go_\beta >$ of the same Full-Contain relationship, that contradicts the Full-Contains definition. ❑

The theorems described below characterize the ones above and is shown through an example. It allows to render equivalent the partial ordered relations obtained by Full-Contains and full containment function. That is possible by creating geographic variables in MDDB due to the presence of the corresponding geographic class in GDB, and vice versa.

Example 3.2: Let us suppose that in MDDB two geographic variables Municipality and District are defined and between them exists a complete containment function. We suppose that in the GDB only the geographic class CITY is defined. The mapping between the geographic class MUNICIPALITY and the geographic variable City implies that for each instance of the variable City one instance of the geographic class MUNICIPALITY exists and vice versa. In GDB, the geographic class DISTRICT can be generated thanks to the DISTRICT *Full-Contains* MUNICIPALITY relationship.
By correspondence of Full-Contains to complete containment function, the "Geo-Union" of the geometric attributes of the MUNICIPALITY instances will generate the geometric attribute of a DISTRICT instance. ◊

Theorem 3.2: Let gc_α and gc_β be two geographic classes. Let the gc_α *Full-Contains* gc_β relationship exist. Let gv_β be a geographic variable. If gc_β is mapped to gv_β by function γ, then the same function allows that a geographic variable named gv_α be mapped to the geographic class gc_α, such that the *complete containment function* from gv_α to gv_β exists.

Proof. The proof of this theorem is derived from theorem 3.1., therefore its proof is straightforward. ❑

Example 3.2: We suppose that in GDB two geographic classes MUNICIPALITY and PROVINCE are defined and there exists between them a Full-Contains relationship. Supposing we have defined in the MDDB, only the geographic variable Municipality. The geographic variable Province can be generated thanks to the complete containment function from Municipality to Province. ◊

Given that link, we can introduce the following assumptions:

- The geographic dimension hierarchy of the MDDB is consistent to the same hierarchy present in the GDB.
- Depending on the above mentioned consistency, to each level and the relative domain of a geographic hierarchy defined in GDB corresponds an equivalent variable and relative domain in the analogue hierarchy defined in the MDDB.

4. Functional attributes

Suppose a user is working with an object-oriented GDB. If s/he desires to interface with MDDB, it is necessary to extend the data structure of GDB, by adding some particular attributes called functional attributes to the already existing ones of a geographic class. Then, with this assumption we classify all attributes that can be included in a geographic class as the following:

• identifier (name), for unique identification of a geographic class.
• geometric, for describing the spatial configuration of the class. It is important because the topological, and metric relationships are defined on this attribute.
• measurable, for defining the numeric measure associated to the geographic class, e. g., surface, length, width, depth, height, etc.
• descriptive, for defining particular information (such as, capital, flag, etc.) associated to the geographic class.
• functional, for describing all the phenomena represented by cubes in MDDB. Every attribute consists of a pair <cube name, {cube variable name}>. The former corresponds to the considered phenomenon (e.g., Car_Sales by model, month, municipality) and represents the cube measures. The latter is composed of all variable names except, the geographic one, because it is implicitly being the same class name (e.g. MUNICIPALITY). These variables are called *local variables*. Note that every functional attribute represents one and only one cube.

For simplifying the data processing, we assume that each cell of a cube contains only one value of a measure and that this value is not a missing or unavailable value.
Note that a functional attribute corresponds to one cube only. In the following, we will discuss how each cube corresponds to many functional attributes as to geographic variables present in the same cube.

In the following sections, we will focus on the functional attributes. Depending on the geographic variable of a cube, we discuss some important properties and operations that can be meaningfully applied to such functional attributes. The properties described here are defined in the light of having to deal with the number of geographic variables in a data cube. For this reason, we found it necessary to discuss functional attributes structure and properties in the presence of one, two or more geographic variables.

Example 4.1: Figure 4.1 shows that every functional attribute in the geographic class MUNICIPALITY corresponds to a cube of MDDB in which the geographic variable has the class name (Municipality). Then, all and only all the cubes which have Municipality as a geographic dimension appear in the functional attributes of the corresponding geographic class MUNICIPALITY. ◊

4. 1. Case of one geographic variable

In the case of a cube with one geographic variable (see Figure 4.1), to every functional attribute of a given geographic class gc_α corresponds a cube of a MDDB in which the geographic variable gv_α has the same class name. Then, all and only all the cubes which have gv_α as a geographic dimension appear in the functional attributes of the corresponding geographic class gc_α.

10.

In the following functional attributes corresponding to cubes are presented.

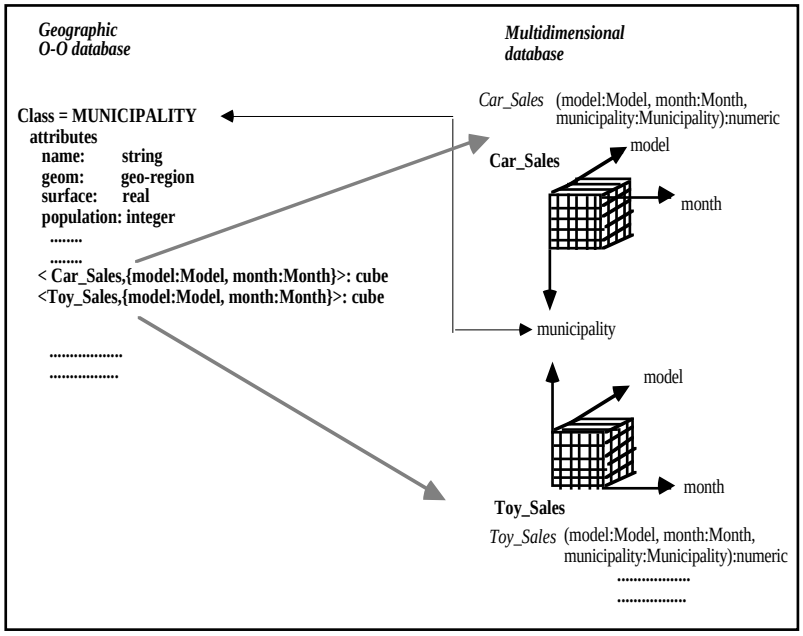

Figure 4.1. Mapping between a geographic class and a geographic variable

Definition. *4.1*: Let $Cube_i$ be defined on the variables $gv_\alpha, gv_\beta, \ldots, gv_n$, where gv_α is the only geographic variable. Let $giv_\alpha, giv_\beta, \ldots, giv_n$ be generic instance of variables $gv_\alpha, gv_\beta, \ldots, gv_n$. Let gc_α a geographic class and gv_α the corresponding geographic variable of a cube $Cube_i$. Let go_α be a generic instance of gc_α. Then, the functional attribute of the generic instance go_α of gc_α, corresponding to the above mentioned cube, is defined as the function:

$$slice_{giv_\beta,\ldots,giv_n}(dice_{giv_\alpha = go_\alpha}(Cube_i)) \qquad (4.1) \quad \Diamond$$

For a clear understanding of the above definition, we present an example that illustrates the mapping between the two environments in the case of one geographic variable.

Example 4.2: In Figure 4.2 the functional attributes of the instance (Roma) of the geographical class MUNICIPALITY refer to the slice of the cubes identified by the corresponding geographic variable instance (Roma) of the geographic variable Municipality.◊

4.2. Case of two or more geographic variables

In this paragraph, we will analyse how the presence of two or more geographic variables in a cube influences the organisation of the functional attributes of geographic classes and its dependence on these geographic dimensions.

Example 4.3: Let us consider a cube defined as the following scheme:

$$\#\,of\ patients\ \{Region\ of\ residence: Region,\ Region\ of\ hospitalization: Region,\ date: Month\}$$

As we can see, in the functional attribute of the geographic class REGION in GDB, it is not possible to choose which of the two geographic variables should be considered implicit. ◊

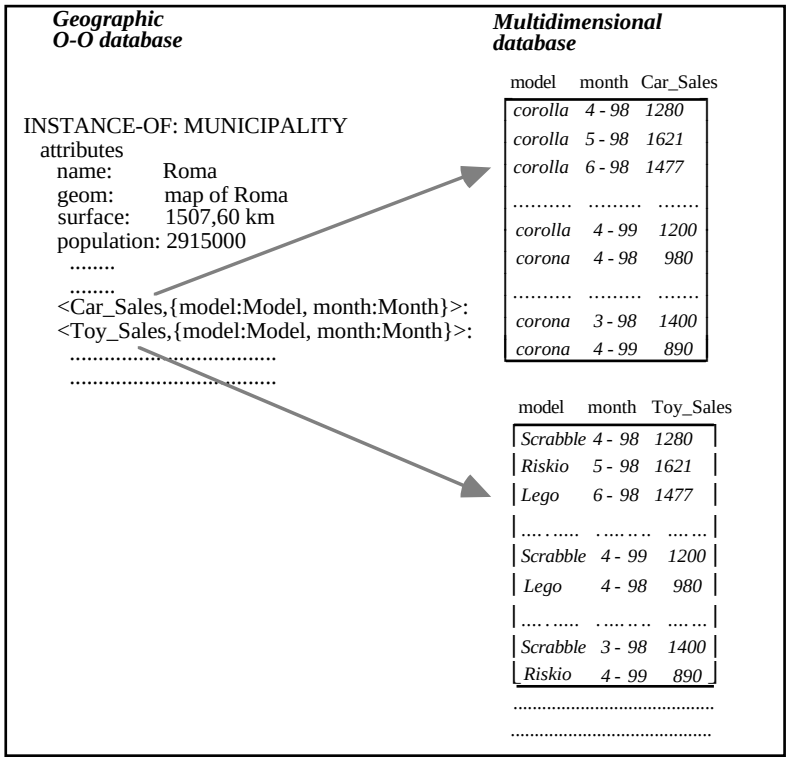

Figure 4.2. Instance of functional attributes "Car_Sales" and "Toy_Sales"

A solution to this problem can be conducted to specialise the geographic class depending on the number of the geographic variables of a cube. Each geographic variable will contribute to the generation of only one functional attribute. Note that, there will be generated as many superclasses as the level of these dimensions are in the geographic classification hierarchy. With this consideration, we will examine the following cases.
The following definitions consider the functional attributes corresponding to a cube with more geographic variables.

Case 1. There is no hierarchical relationship among the geographic variables of the cube.

Let us consider a cube with two geographic variables gv_α and gv_β. If gv_α and gv_β. are specializations of another variable gv_η, then this last one has to be considered to become a superclass of its own specializations. In each functional attribute of the specialised geographic classes, the corresponding cube is considered.

Therefore, the problem of defining functional attribute can be solved by the following steps:
a) rendering the geographic class gc_η a superclass;
b) definition of a specialisation of the superclass gc_η named gc_α with the functional attribute defined as below:

```
CLASS = gc_α
attributes
  attr_1: Kind
```

12.

```
   ..............
   <cube-name_k, {gc_β : gc_η, ….}>: cube
```

c) definition of a specialisation of the superclass gc_η named gc_β with the following functional attribute:

```
CLASS = gc_β
attributes
 attr_1: Kind
 ..............
 <cube-name_k,  {gc_α : gc_η, …}>: cube
```

With this approach the organisation of the functional attributes of the geographic class becomes dependent on the phenomena described by the cube and will assume different semantics. Respectively in steps b and c, the measures described by the cube represented in the functional attribute are depending respectively from gc_β and gc_α.

E*xample 4.4*: Let us consider a cube as defined in Example 4.3 with two geographic variables. It is possible to see that Region of residence and Region of hospitalization dimensions are a specialisation of the "region concept", then this last one is considered to become a superclass of its own specializations. In each functional attribute of the specialised geographic classes, i.e., REGION OF RESIDENCE and REGION OF HOSPITALIZATION, the corresponding cube is considered. Then,

```
CLASS = REGION OF RESIDENCE
  attributes
  name: string
    ..............
 <#of patients,{Region of
    hospitalization:Region, date:Month}>: cube

CLASS = REGION OF HOSPITALIZATION
 attributes
  name: string
   ..............
  <# of patients, {Region of
     residence:Region, date:Month}>: cube                    ◊
```

Case 2. There is a hierarchical relationship between the Geographic variables of the cube.

Let us consider a cube with two geographic variables gv_α and gv_β. If gv_α is a specialisation of another variable gv_η and gv_β is a specialisation of another variable gv_μ, then gv_η and gv_μ can be considered to become superclasses of their specializations. In each functional attribute of the specialised geographic classes, the corresponding cube is considered.
Therefore, the problem of the functional attribute definition can be solved by the following steps:

a) rendering the geographic classes gc_η and gc_μ; the superclasses;
b) definition of a specialisation of the superclass gc_η named gc_α with the functional attribute defined as below:

```
CLASS = gc_α
```

attributes
 $attr_1$: Kind
  ....
 <cube-name$_k$, {gc_β : gc_μ , ….}>:cube

c) definition of a specialisation of the superclass gc_μ named gc_β with the following functional attribute:

CLASS = gc_β
attributes
  $attr_1$: Kind
  ....
 <cube-name$_k$, {gc_α : gc_η, …}>:cube

The following example illustrates the mapping between the two environments in the case of two geographic variables among which there is no hierarchical relationship.

Example 4.5: Let us consider the following cube:

# of patients {*Region of residence : Region, Municipality of hospitalization : Municipality, date : Month*}

It is possible to see that Region of residence is a specialisation of the "region concept" and Municipality of hospitalization is a specialisation of the "municipality concept", then these can be considered to become the superclasses of its specializations. In each functional attribute of the specialised geographic classes, i.e., REGION OF RESIDENCE and MUNICIPALITY OF HOSPITALIZATION, the corresponding cube is considered. Then,

```
CLASS = REGION OF RESIDENCE
attributes
  name: string
  ..............
 <# of patients, {Municipality of hospitalization: Municipality,
   date: Month}>: cube

CLASS = MUNICIPALITY OF HOSPITALIZATION
  attributes
   name: string
   ..............
  <# of patients, {Region of
    residence: Region, date: Month}>: cube
```

The formula (4.1) can be extended even in the case of two variables.

5. The deduction of functional attributes

Each geographic variable of a cube belongs to a level of the geographic hierarchy. Along this hierarchy, we can obtain a new cube by aggregating all values along the geographic classification hierarchy by a roll-up operator. To this so called aggregate geographic variable, corresponds a geographic class with the same name. The functional attribute of this class is defined by the name of such a new cube and local variables, naturally making implicit only the geographic variable.

Note that once the functional attributes of a geographic class have been defined, we have to analyze some important issues related to the deduction of functional attributes from other

geographic classes considering that between them there existed previously a *Full-Contains* relationship.

Since, a *Full-Contains* relationship concerns the geometric property of the geographic classes and objects, for describing the deduction of functional attributes, we introduce the *is-derived* (for brevity it is denoted by ISD and it is represented graphically by a grey arrow) relationship between two classes or objects.

For describing this relationship, let gc_1 and gc_2 be two geographic classes. Let gc_2 *Full-Contains* gc_1 be satisfied and the functional attributes of class gc_1 be known. We define the concept of ISD relationship in the following definition.

Definition 5.1: An ISD relationship between two geographic classes is a triple $gc_2\ ISD\ gc_1$ where ISD indicates that the functional attributes of gc_1 are also the functional attributes of gc_2.

The functional attributes of class gc_2 are calculated from the corresponding functional attributes of gc_1 by means of roll-up, dice, and slice operators applying on the same cube from which the functional attribute of gc_1 are generated. Obviously, the functional attributes of a geographic class are inherited from its superclasses along the *ISA* hierarchy. In fact, if a functional attribute of a class gc_2 "is derived" from class gc_1 and class gc_3 is a subclass of gc_2 (see Figure 5.1), then in the functional attributes of gc_3 are included also the functional attributes of gc_1.

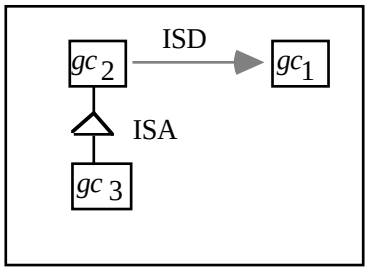

Figure 5.1. The inheritance of functional attribute

Property: The ISD is a partially ordered relation, i.e., it satisfies the following properties.

- *reflexivity:* Let gc be a geographic class, $gc\ ISD\ gc$.
- *antisymmetry*: Let gc_i, gc_j be geographic classes where $i \neq j$. If $gc_i\ ISD\ gc_j$ then $gc_j\ ISD\ gc_i$ is not satisfied.
- *transitivity:* Let gc_i, gc_j, gc_k be geographic classes where $i \neq j$ and $j \neq k$. If $gc_i\ ISD\ gc_j$ and $gc_j\ ISD\ gc_k$ then $gc_i\ ISD\ gc_k$.

This property is predictable because the ISD relationship corresponds to the *Full-Contains* relationship.

The transitive property of this relationship, indicates that the functional attributes of class gc_k will be included in the set of functional attributes of the class gc_i. They are calculated by a roll-up operator composed of a concatenation of complete containment functions along the geographic dimension.

Example 5.1: The class PROVINCE will also admit the functional attribute of the class MUNICIPALITY as it is graphically represented in Figure 5.2. This figure describes PROVINCE *ISD* MUNICIPALITY.

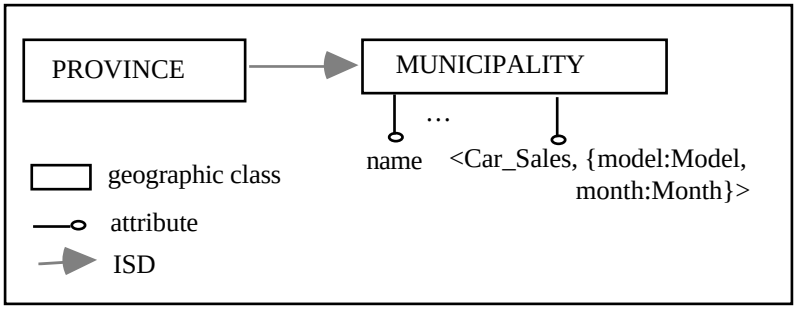

Figure 5.2. Graphical representation of a part of the extended GDB scheme

For each instance of PROVINCE, for example "Milan", this attribute is calculated by the formula (5. 1).

$$slice_{model,month}(dice_{province=Milan}\ (roll\text{-}up^{sum(Car\_Sales)}_{Municipality\rightarrow Province}(Car\_Sales))) \qquad (5.1) \qquad \Diamond$$

In a similar way as a geographic variable, we will discuss the *ISD* relationship between geographic classes. In this case, we have the generation of more functional attributes corresponding to different specialised geographic classes that are interrelated by the *ISD* relationship.

The definition process of a functional attribute into a specialised geographic class, starting from the geographic dimension of a cube isomorphic to the metaclass of the specialised geographic class, consists of the following steps:

1. Creation of a specialised geographic class for each specialised geographic variable of the cube. These specialised geographic variables are the levels of a hierarchy that is called *multiplicity* of the starting hierarchy. This *multiplicity of hierarchy* [PR99, PR00] has the names of the levels which include the specialization of the hierarchy, while the instances of the domains are exactly the same of the starting hierarchy.
2. Creation of the specialised functional attribute into each of the above mentioned specialised geographic classes;
3. Creation of an *ISD* relationship between each of these subclasses created in the step 1 and the geographic metaclass parent of the geographic specialized geographic class.

Example 5.2: Let us consider Figure 3.1. We extend this GDB with a cube defined by the following scheme:

# of patients {*Province of residence : Province*, *Municipality of hospitalization : Municipality*, *date : Month*}

Note that in the initial phase of correspondence between MDDB and GDB, the classes MUNICIPALITY, PROVINCE, and REGION become superclasses.
Then, depending on the proposed approach we have:
1. the automatic generation of *ISD* relationship between each pair of superclasses which are already related with the Full-Contains relationship;

2. the generation of the specialised geographic class PROVINCE OF RESIDENCE with the functional attribute

<#of patients {Municipality of hospitalization: Municipality, date: Month}>;

3. the generation of the specialised geographic class REGION OF RESIDENCE and the *ISD* relationship between this class and this defined in STEP 2;

4. the generation of the geographic class MUNICIPALITY OF HOSPITALIZATION with the functional attribute

<#of patients {Province of residence: Province, date: Month}>;

5. the generation of the geographic classes PROVINCE OF HOSPITALIZATION and REGION OF HOSPITALIZATION and *ISD* relationship between them. ◊

Analogously in the above discussion, the derived functional attributes were not created into the geographic classes which are at a higher more aggregate level in the hierarchy.
In Figure 5.3 the result of this process is shown. Each previous labelled step is represented with the same label in Figure 5.3.

6. Number of the Functional Attributes

Previously we defined two different types of functional attributes, that can be obtained from a "cube base". They are:

a) *derived* functional attributes, obtained along geographic class hierarchy by the *ISD* relationship;

b) *deduced* functional attributes, obtained from local variables of the functional attribute related to a given cube base, by the OLAP operator application (i.e., roll-up and slice).

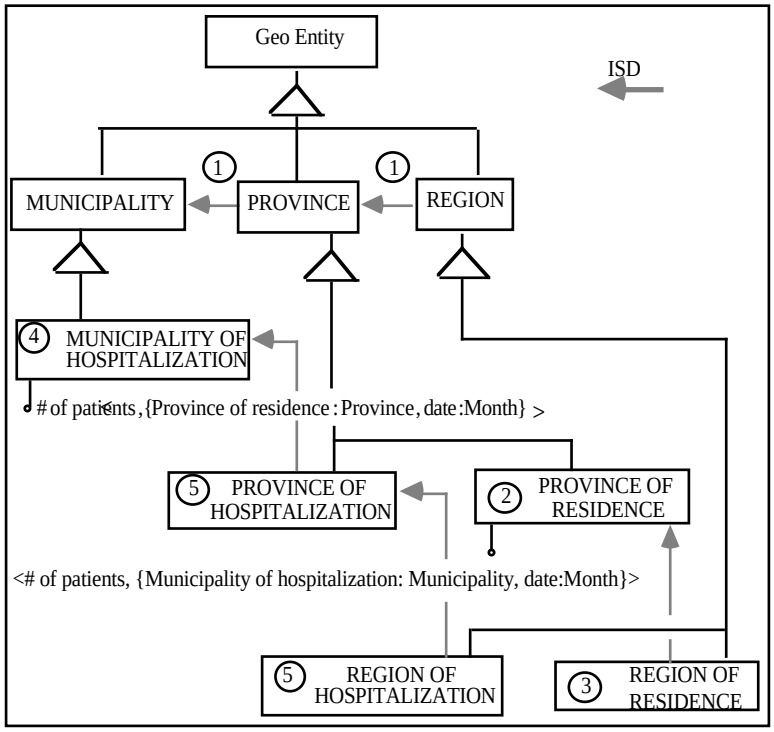

Figure 5. 3. Example of different steps for the generation of specialized geographic classes.

In order to calculate the total number of the functional attributes (base, deducted, and derived), we consider the following symbols:

- M is the number of different cubes in a multidimensional database;
- N_i represents the total number of variables (geographic and non-geographic) of the *i*-th cube;
- V_i represents the number of geographic variables of the *i*-th cube;
- $l_{i,h}$ represents the number of base and derived functional attributes, generated by a geographic variable ($1 \le h \le L_{V_i}$) of the cube.
- $f_{i,k}$ is the number of the functional attributes generated by any variable $k \ge 1$ of the *i*-th base cube in a class by the operators roll-up and slice.

Any geographic variable of each independent cube generates a base functional attribute: in a generic cube exist V_i geographic variables. Therefore, we have the following definition.

***Definition** 6.1*: Given a generic cube. The number of base functional attributes ($A_{i_{(B)}}$) of this cube is $A_{i_{(B)}} = V_i$.

As above-mentioned, moving along the classification hierarchy of the geographic dimension, we obtain a functional attribute derived by the ISD relationship.

Definition 6.2: The number of base and derived functional attributes ($A_{i_{(B,D)}}$) of a cube is:

$$A_{i_{(B,D)}} = \sum_{h=1}^{V_i} l_{i,h} .$$

We can calculate the total number of the attributes generated by a cube in a geographic class by the next theorem.

Theorem 6.1: The total number of base, derived, and deducted functional attributes of a cube is given as follows:

$$A_{i(Tot)} = \prod_{k=1}^{N_i} (f_{i,k} + 1) \sum_{j=1}^{V_i} \frac{l_{i,h}}{l_{i,h} + 1}$$

Proof: For each base and derived functional attribute we can generate the deduced functional attributes by the operators roll-up and slice applied on the local variables. Let $P_{i,h}$ be the number of local variables of a generic functional attribute and let $S_{i,h,q}$, $q \ge 1$ be the number of deduced functional attributes generated by the application of the operators roll-up and slice on a generic local variable. Then, the number of deduced functional attributes obtained by all the local variables is:

18.

$$\prod_{q=1}^{P_{i,h}}(S_{i,h,q}+1)$$

From a generic geographic variable h we obtain $l_{i,h}$ base and derived functional attributes; then, the number of derived functional attributes generated by a geographic variable of a cube is:

$$l_{i,h}\prod_{q=1}^{P_{i,h}}(S_{i,h,q}+1)$$

Hence, the number of geographic variables is V_i , then the total number of base, derived, and deducted functional attributes generated by a cube is:

$$\sum_{h=1}^{V_i} l_{i,h}\prod_{q=1}^{P_{i,h}}(S_{i,h,q}+1)$$

As we can observe, the geographic variables of the cube appear twice as $l_{i,h}$ and $S_{i,h,q}$.

$$\ldots + l_{i,h}(l_{i,h'}+1)\prod_{q=1}^{P_{i,h}-1}(S_{i,h,q}+1) + l_{i,h'}(l_{i,h}+1)\prod_{q=1}^{P_{i,h}-1}(S_{i,h,q}+1) + \ldots =$$

$$= \ldots + \frac{l_{i,h}}{l_{i,h}+1}(l_{i,h}+1)(l_{i,h'}+1)\prod_{q=1}^{P_{i,h}-1}(S_{i,h,q}+1) + \frac{l_{i,h'}}{l_{i,h'}+1}(l_{i,h}+1)(l_{i,h'}+1)\prod_{q=1}^{P_{i,h}-1}(S_{i,h,q}+1) + \ldots$$

where the term $\prod_{q=1}^{P_{i,h}-1}(S_{i,h,q}+1)$ refers only to non geographic variables.

Therefore, N_i is the number of geographic and non geographic variables of a cube, we have:

$$A_{i_{Tot}} = \sum_{h=1}^{V_i} l_{i,h}\prod_{q=1}^{P_{i,h}}(S_{i,h,q}+1) = \prod_{k=1}^{N_i}(f_{i,k}+1)\sum_{h=1}^{V_i}\frac{l_{i,h}}{l_{i,h}+1}$$ ❑

Theorem 6.2: The number of base functional attributes generated by M independent cubes is:

$$A_B = \sum_{m=1}^{M} N_m.$$

The number of base, and derived functional attributes generated by M independent cubes is:

$$A_{B,D_{(Tot)}} = \sum_{i=1}^{M} \sum_{h=1}^{V_i} l_{i,h}.$$

Then, the total number of the base, derived, and deduced functional attributes generated by M cubes is:

$$A_{Tot} = \sum_{i=1}^{M} \left(\prod_{k=1}^{N_i} (f_{i,k} + 1) \sum_{h=1}^{V_i} \frac{l_{i,h}}{l_{i,h}+1} \right).$$

Proof: The proof is trivial because the M cubes are independent among them, then all the results of the Theorem 6.1 are satisfied for all cubes. ❑

Example 6.1: We consider a multidimensional database which consists of the following cubes:

- Car_Sales(day: Day, city: City, model: Type): numeric;
- # of patients(region of hospitalization: Region, region of residence: Region, month: Month}: numeric;
- Transportation_matrix(city of input: City, province of output: Province, day: Day):numeric.

Moreover, the following hierarchies are also defined:

- Day → Month → Year;
- City → Province → Region;
- Type → Vehicle.

Therefore, the numeric values are:

- $M = 3$,
- $V_{Car\_Sales} = 1$,
- $V_{\#\ of\ patients} = 2$,
- $V_{Transportation\_matrix} = 2$,
- $N_{Car\_Sales} = 3$,
- $N_{\#\ of\ patients} = 3$,
- $N_{Transportation\_matrix} = 3$,
- $f_{City} = l_{City} = 3$,
- $f_{Province} = l_{Province} = 2$,
- $f_{Region} = l_{Region} = 1$,
- $f_{Day} = 3$,
- $f_{Month} = 2$,
- $f_{Type} = 2$.

The number of the base functional attributes in the multidimensional database is $A_B = 5$. The number of the base, and deduced functional attributes generated by the cubes are respectively:

- $Car\_Sales \rightarrow A_{Car\_Sales_{(B,D)}}=3;$
- $\#\ of\ patients \rightarrow A_{\#of\ patients_{(B,D)}} = 2,$
- $Transportation\_matrix \rightarrow A_{Transportation\_matrix_{(B,D)}} = 5,$

and generated by all the cubes is $A_{B,D_{(Tot)}} = 10$.
Then, the number of the base, deduced and derived functional attributes generated by the cube is respectively:

- $Car\_Sales \rightarrow A_{Car\_Sales_{(Tot)}} = 36,$
- $\#\ of\ patients \rightarrow A_{\#\ of\ patients_{(Tot)}} = 12,$
- $Transportation\_matrix \rightarrow A_{Transportation\_matrix_{(Tot)}} = 68,$

Finally, the total number of the base, deduced, and derived functional attributes generated by all the cubes is $A_{Tot}=116$.

From the previous example we see that with a little number of basic cubes we obtain a large number of functional attributes (basic, deduced, and derived). In order to reduce this large number to only functional attributes corresponding to the basic cubes, we introduced the *ISD* relationship, which models all the derived functional attributes , as well as a set of methods, defined in the root class Geo Entity, which perform the above mentioned OLAP operators to model the deduced functional attributes. Note that these operations will be carried out in the multidimensional environment.

7. A query example

In this section, we give a query example based on the extended GDB cube represented in Figure 4.1 that concerns the proposed approach.

Example 7.1: "Find all the regions adjacent to the Tuscany region, in which the number of cars sold in 1990, for the <Corolla> product, was greater than 10,000"
The procedure for solving the query is the following:

1) individualize the geographic class and geographic object/s involved by the geographic operator in the query;
In the case of the above mentioned query they are regions touching Tuscany;
2) solving this part of the query;
In the above query the result of the geographic operator *Geo-Touching* gives these instances: "Liguria, Emilia-Romagna, Umbria, Lazio"
3) individualize the functional attribute in the geographic class which is the "target" of the query;
In our case, "Car_Sales" in "regions"
4) select the identified cube from the functional attribute in MDDB;
In our example, select the cube "Car_Sales"
5) satisfy the constraints on the non geographic dimension of the cube;
In our case, apply the *dice* operator to the specified values for the Time and Model dimensions
6) satisfy the constraints on the measure of the cube;

In this query, all the instances of the measure
7) select through the *dice* operator all slices of the cube obtained by the target of the query corresponding to the values of the measure that satisfy the previous constraints;
In our query, they are the regions with the number of cars sold in 1990, for the <Corolla> product, was greater than 10,000.
8) Pass to the GDB the list of the geographic variable instances that satisfy the query. This means to individualize the corresponding geographic objects.

At conceptual level, in the extended geographic environment this query may be expressed by the following SQL-like expression:

```
SELECT    R1.name
FROM      REGION R1, R2
WHERE
      R1.Car_Sales(model=Corolla,
      year=1990)>10,000
 AND
      R1.geom Geo-Touching R2.geom
 AND
      R2.name=Tuscany
```

Let us recall that the functional attribute

$$< Car\_Sales, \{model:Model, month:Month\} >$$

is obtained from the cube generated by performing the formula:

$$roll-up^{sum(Car\_Sales)}_{Municipality \rightarrow Region}(Car\_Sales) \qquad (7.1)$$

where the containment function Municipality → Region is the concatenation of the two containment functions Province → Region and Municipality → Province.
The formula (7.1) is equivalent to the formula:

$$roll-up^{sum(Car\_Sales)}_{Province \rightarrow Region}(roll-up^{sum(Car\_Sales)}_{Municipality \rightarrow Province}(Car\_Sales)) \qquad (7.\ 2)$$

This query can be solved in pictorial fashion [FMR99] as shown in Figure 7.1. It is solved through the following steps:

- in the geographical database:
  Region *Geo-Touching* Region=Tuscany
- and in the multidimensional database:

$$dice_{Car\_Sales>10,000}(dice_{model=Corolla}(dice_{year=1990}(roll-up^{sum(Car\_Sales)}_{Month \rightarrow Year}(Car\_Sales)))) \qquad (7.3) \quad \Diamond$$

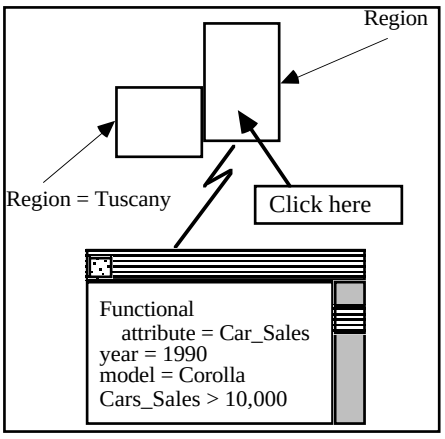

Figure 7.1. **A pictorial query formulation**

Note that all the operations which refer to the geographical information are carried out only in the geographical database. The results of these operations are automatically inherited in the multidimensional database, where the dice and roll-up operators are performed.

8. Discussions and Conclusions

This paper motivates and describes an extension of a geographic database with Multidimensional data. The extension is based on the use of the mapping between the same geographic hierarchy presented in GDB and MDDB environments. This extension is obtained by introducing the new attributes called functional attributes in the geographic classes. These functional attributes refer to cubes of a MDDB. This simple linking mechanism from geographic to interrelated cube data permits to have access and use data stored in the MDDB.

The application of any of the above mentioned OLAP operators to any other non-geographic dimensions of the examined cube creates a new cube which, in its turn, produces a new functional attribute of the geographic class corresponding to the geographic variable in the GDB. Therefore: a) if the operation is *slice*, the dimension to which slice is applied disappears both in the new cube and the new functional attribute of the same geographic class to which it refers; b) if the operation is *roll-up*, the variable corresponding to the higher level of the relative dimension appears in the new cube, and in the new functional attribute of the same geographic class to which it refers there will be a similar name substitution; c) finally, if the operation is dice, the solution is the same as b.

Every time a functional attribute is created in a geographic class, there is an automatic possibility to access all functional attributes deriving from the OLAP operators, such as roll-up and slice. For example, in the geographic class MUNICIPALITY (see Example 4.1) the presence of the functional attribute <Car_Sales, {model: Model, month:Month}> implies that the following functional attributes are derived by slice operator: <Car_Sales, {model: Model}>, <Car_Sales, {month:Month}>, <Car_Sales, {}>.

A manipulation of cubes in a MDDB that generates derived cubes, does not automatically create the relative links to the corresponding functional attributes derived in the corresponding GDB geographic class. Thus, we keep just one conceptual link between the two environments expressed through a geographic variable/class.

Therefore, if we have a cube in a MDDB which geographic variable is linked to a class in GDB and if, consequently, in such a class we have a functional attribute that refers to the previous cube, then the isomorphism created between the two entities confirms that to each derived cube corresponds a derived functional attribute and vice versa, without memorizing

the relative link. To create such a link would mean to physically integrate DBs, while we want to keep them independent except for the geographic link (which must be consistent).

If the manipulation of a cube refers to the geographic variable, for example, reclassifying province to region, analogously it is not necessary to create a link between the derived cube and the new geographic class to which it refers (in this case, region) since there is a relation *ISD* between the two geographic classes (province and region).

At present, the implementation of a prototype based on the proposed approach is in progress. It uses a MDDB based on the Relational Model and an Object-Oriented GDB. Future research will be devoted to the problem of missing data in MDDB in the context of our proposal.

References

[AGS97] Agrawal R., Gupta A., Sarawagi S. "Modelling Multidimensional Databases" *13th International Conference on Data Engineering - ICDE''97*, Birmingham, U. K., April 7-11, 1997, pp. 232-243.

[CCS93] Codd E.F., Codd S.B., Salley C.T. "Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate".*http://www.cs.toronto.edu/~mendel/dwbib.html*.. Technical rep-ort 1993.

[FMR99] Ferri F., Massari F., Rafanelli M. "PQL: A Pictorial Query Language for Geographic Features in an Object-Oriented Environment" *Journal of Visual Languages and Computing*, Vol.10,pp.641-671, 1999.

[GBL96] Gray J., Bosworth A., Layman A., Pirahesh H. "Data cube: a relational aggregation operator generalizing group-by, cross-tabs and subtotals" Proceed. *12th IEEE International Conference on Data Engineering - ICDE'96*, New Orleans, Louisiana, Feb.26-Mar.1, 1996, pp. 152-159.

[GL97] Gyssens M., Lakshmanan L.V.S. "A foundation for multidimensional databases" Proc. of the *23th Intern. Conference on Very Large Data Bases - VLDB'97*, Athens, Greece, 26-29 August, 1997, pp. 106-115.

[KA90] Khoshafian S., Abnous R. *Object-Orientation - Concepts, Languages, Databases, User Interfaces*; Wiley, New York, 1990.

[LS97] Lenz H.J., Shoshani A. "Summarizzability in OLAP and statistical data bases" Proceed. of the *9th International Conference on Scientific and Statistical Data Management - SSDBM'97*, 1997, pp. 132-143.

[Mic91] Michalewicz Z. "Statistical and Scientific Databases" Michalewicz Z. Editor, Ellis Horwood Publ., 1991.

[Ola97] OLAP Council, "The OLAP glossary". The OLAP Council 1997.*http://www.olapcouncil.org.*

[PR99] Pourabbas E., Rafanelli M. "Characterization of Hierarchies and Some Operators in OLAP Environment" ACM Second Int. *Workshop on Data Warehousing and OLAP - DOLAP '99*, Kansas City, USA, Nov. 6, 1999, pp.54-59.

[PR00] Pourabbas E., Rafanelli M. "Hierarchies and Relative Operators in the OLAP environment" *Journal of ACM Sigmod Record*, Vol. 29, N. 1, March 2000.

[Raf91] Rafanelli M. "Statistical and scientific database management systems" in *Encyclopedia of Computer Science and Technology*, Kent & Williams Ed.s, M.Dekker Publ., Vol. 23, Suppl. 8, 1991, pp.369-409

[RR93] Rafanelli M., Ricci F.L. "Mefisto: a functional model for statistical entities" *Journal of IEEE Transactions on Knowledge and Data Engineering*, August 1993, Vol.5, No.4, pp. 670-681.

[RS90] Rafanelli M., Shoshani A. "STORM: A Statistical Object Representation Model" Proceed. of the *5th Intern. Confer. on Statistical and Scientific Database Management - SSDBM'90*, Lecture Notes in Computer Science, Springer Verlag Pub., Vol. 420, 1990, pp.14-29.

[Sho97] Shoshani A. "OLAP and Statistical Databases: Similarities and Differences" in Proc. of the *16th ACM Symposium on Principles of Database Systems - PODS '97*, pp. 185-196.

[SS77] Smith J.M., Smith D.C.P. "Database abstractions: Aggregation and generalization"; *ACM Transactions on Database Systems* 2, 2, pp.105-133, 1977.

[SV92] Scholl M., Voisard A. "Object-Oriented Database Systems for Geographic Applications: An Experiment with O2", *The O2 Book*, F. Bancilhon, C. Delobel and P. Kanellakis (Eds.), Morgan Kaufmann, San Mateo, California, 1992.