

# Modelling and Optimisation Issues for Multidimensional Databases<sup>1</sup>

Panos Vassiliadis

Spiros Skiadopoulos

National Technical University of Athens  
Department of Electrical and Computer Engineering  
Computer Science Division  
Knowledge and Database Systems Laboratory  
Zografou 15773, Athens, Greece  
{pvassil,spiros}@dbnet.ece.ntua.gr

**Abstract.** It is commonly agreed that multidimensional data cubes form the basic logical data model for OLAP applications. Still, there seems to be no agreement on a common model for cubes. In this paper we propose a logical model for cubes based on the key observation that a cube is not a self-existing entity, but rather a view over an underlying data set. We accompany our model with syntactic characterisations for the problem of cube usability. To this end, we have developed algorithms to check whether (a) the marginal conditions of two cubes are appropriate for a rewriting, in the presence of aggregation hierarchies and (b) an implication exists between two selection conditions that involve different levels of aggregation of the same dimension hierarchy. Finally, we present a rewriting algorithm for the cube usability problem.

## 1 Introduction

On-Line Analytical Processing (OLAP) is a trend in database technology based on the multidimensional view of data. Although multidimensional data cubes form the basic logical data model for OLAP applications, up to now, no common agreement has been obtained on the elements of a cube model. Several industrial standards already exist [13,14,15,16], yet, apart for the last one, none of them seems to propose a well-founded model for OLAP databases. In academia, several proposals on the modelling of cubes also exist [1,2,9,10,11,21]. Despite all these efforts, we feel that several key characteristics of a cube model have not been stressed, neither by the academia nor the industry (see [19] for a complete discussion). To this end, we present a *logical* model for cubes. This model extends the proposal of [21] in a more formal and systematic way. It deals with all the commonly encountered entities of a multidimensional model (dimension hierarchies, data cubes and cube operations) without being restricted from their physical implementation (e.g., ROLAP or MOLAP architectures). One of our

---

<sup>1</sup> This research is sponsored by the European Esprit Project "DWQ: Foundations of Data Warehouse Quality", No. 22469. We also wish to thank Prof. Timos Sellis and Dr. Dimitri Theodoratos for useful discussions on the topic.

key observations is that *a cube is not a self-existing entity, but rather a view (materialised or not) over an underlying data set*. This property allows us to develop complex operations, not dealt by other models so far (e.g., the drill-down operation and the change of aggregation function).

To our knowledge, existing OLAP tools behave in an “extensional” fashion. Cubes are treated simply as sets of tuples, ignoring the fact that they are produced as queries over an underlying detailed data set (e.g., the fact table of a data warehouse). Our framework, instead, suggests a different strategy: we keep the “history” of performed selections and thus, we are able to compute a new cube taking into account its “intentional” description. Therefore, we can define more complex operations (such as drill-down) and sequences of operations, which are not covered by other models. Our model is accompanied by an algebra powerful enough to capture the usual OLAP operations such as (a) *selection* over a cube, (b) *roll-up*, which means aggregation over a cube to coarser granularities of information and (c) *drill-down*, which involves de-aggregation of a specific cube and presentation of more detailed information.

The contribution of this paper lies not only in terms of expressiveness, but also we present results on optimisation issues for multidimensional databases. We investigate the *cube usability problem*, a variant of the relational view usability problem, for multidimensional cubes. We accompany our framework with optimisation techniques for the cube usability problem that enable the exploitation of existing cubes in order to compute new cubes. To handle the cube usability problem, we extend well-known techniques already found in the relational context on the containment of selection conditions [17]. We have observed that although quite a lot of work has been performed in the field of query containment and view usability in the context of relational databases [4,5,8], there exist no results to exploit the information about dimension hierarchies in the context of multidimensional databases. We present results on two major topics. First, we tackle the problem of containment of two selections, taking into account their marginal conditions in the presence of dimension hierarchies. Secondly, we come up with a set of axioms to characterise containment for expressions involving functionally dependent attributes. Although several results already exist to characterise query containment between expressions involving one domain [17], to our knowledge, no results exist for expressions involving different functionally dependent levels. For lack of space, all the proofs, as well as further explanations, are found in [20].

This paper is organised as follows. In Section 2 we present the logical cube model. Section 3 presents optimisation issues. Finally, in Section 4 we discuss our results and present future work.

## 2 Cubes for Multidimensional Databases

In this section we present the basic entities and operations of our model. Entities involve dimensions, data sets and cubes. Operations involve selections and change in the granularity of data. This model extends previous proposals of [2,10,21].

One of the main characteristics of OLAP applications is the *multidimensional view of data* in the perception of the user, which considers that information is stored in a *multidimensional array*, called *Cube* or *HyperCube*. Thus, a *Cube* is a group of data cells. Each cell is uniquely defined by the corresponding values of the dimensions of the cube. The contents of the cell are named *measures* and represent the measured values of the real world. Measures are functionally dependent, in the relational sense, on the dimensions of the cube.

A *dimension* is defined in [15] as “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”. Informally, a dimension models all the possible ways in which the user can group the detailed information stored in the multidimensional database with respect to a specific context. Each dimension has an associated *hierarchy of levels* of aggregated data i.e., it can be viewed from different levels of detail. Formally, a *dimension*  $D$  is a lattice  $(L, <): L = (L_1, \dots, L_n, ALL)$ . We require that the upper bound of the lattice is always the level  $ALL$ , so that we can group all the values of the dimension into the single value 'all'. The lower bound of the lattice is called the *detailed level* of the dimension. For instance, let us consider the dimension *Date* of Fig. 2. Levels of dimension *Date* are *Day*, *Week*, *Month*, *Year* and  $ALL$ . *Day* is the most detailed level. Level  $ALL$  is the most coarse level for all the dimensions. Aggregating to the level  $ALL$  of a dimension ignores the respective dimension in the grouping (i.e., practically groups the data with respect to all the other dimensions of the cube, except for this particular one).

The relationship between the values of the dimension levels is achieved through the use of the set of  $anc_{L_i}^{L_j}$  functions. A function  $anc_{L_i}^{L_j}$  assigns a value of the domain of  $L_2$  to a value of the domain of  $L_1$ . For instance  $anc_{Month}^{Year}(Feb-97) = 1997$ .

The major multidimensional operations are *selection* and *navigation*. Selection is used whereby a criterion is evaluated against the data or levels of a dimension in order to restrict the set of retrieved data. Navigation is a term used to describe the processes employed by users to explore a cube interactively by changing the granularity of the multidimensionally viewed data [15]. Possible navigation operations, which can be applied to a cube, are: (a) *Roll-up* which corresponds to the aggregation of data from a lower to a higher level of granularity within a dimension's hierarchy, (b) *Drill-Down* which is the inverse of roll-up and allows the de-aggregation of information moving from higher to lower levels of granularity and (c) *Slice* which corresponds to the grouping of data with respect to a subset of the dimensions of a cube. For instance, let us consider the dimension *Date*; aggregating from *Month* to *Year* is a roll-up operation and de-aggregating from *Month* to *Day* is a drill-down operation. In our model, the slice operation is modelled as a roll-up to level  $ALL$ .

We denote sets of tuples under a specific schema by the term *data set*. Moreover, we assume the existence of a *detailed data set*, i.e., a data set that is defined at the finest levels of granularity for all its dimensions. This detailed data set is the central source of data, which will populate any cubes produced during an OLAP session (e.g., a fact table in a data warehouse).

One of our key observations is that a *cube* is not a self-existing entity (as commonly encountered in the literature), but rather a view over an underlying detailed data set. As usual, a view (and thus a cube) can be either materialised or not. Therefore, a cube

can be seen either as a data set or simply a query. In our model, we retain this dual nature formally; a cube is not only a set of tuples, but also has a definition. This definition is a query that reduces the computation of the cube to a set of operations over the initial materialised detailed data set.

Formally, a *cube*  $c$  over the schema  $[L_1, \dots, L_n, M_1, \dots, M_m]$ , is an expression of the form:  $c = (DS^0, \varphi, [L_1, \dots, L_n, M_1, \dots, M_m], [agg_1(M_1^0), \dots, agg_m(M_m^0)])$ , where  $DS^0$  is a detailed data set over the schema  $S = [L_1^0, \dots, L_n^0, M_1^0, \dots, M_k^0]$ ,  $m \leq k$ ,  $\varphi$  is a detailed selection condition,  $M_1^0, \dots, M_m^0$  are detailed measures,  $M_1, \dots, M_m$  are aggregated measures,  $L_i^0$  and  $L_i$  are levels such that  $L_i^0 \prec L_i$ ,  $1 \leq i \leq n$  and  $agg_i$ ,  $1 \leq i \leq m$  are aggregated functions from the set  $\{sum, min, max, count\}$ .

Intuitively, to compute a cube, first we apply the selection condition to the detailed data set. Then, we replace the values of the levels for the tuples of the result, with their respective ancestor values at the levels of the schema of the cube and group them into a single value for each measure, through the application of the appropriate aggregate function. Note that a detailed data set can be trivially expressed as a cube, having a *true* selection condition and an arbitrary aggregation function. For instance, the cube of the detailed data set  $DS^0$  of Fig. 1 is expressed as:  $c^0 = (DS^0, true, [day, day, item, salesman, city, sales], sum(sales))$ .

This approach introduces a powerful expression mechanism, able to directly capture operations like drill-down and change of aggregate function and thus, aimed towards the modelling of sequences of operations, as normally encountered in OLAP systems. To our knowledge, no other model can capture these operations directly. The reduction of a cube's definition to a normalised form seems to be the only alternative that directly achieves this kind of functionality.

Formally, the model consists of the following elements:

- Each *dimension*  $D$  is a lattice  $(L, \prec)$  such that:  $L = (L_1, \dots, L_n, ALL)$  is a finite subset of *levels* and  $\prec$  is a partial order defined among the levels of  $L$ , such that  $L_1 \prec L_i \prec ALL$  for every  $1 \leq i \leq n$ .
- A family of functions  $anc_{L_i}^{L_j}$  satisfying the following conditions (extending [2]):
  1. For each pair of levels  $L_1$  and  $L_2$  such that  $L_1 \prec L_2$  the function  $anc_{L_1}^{L_2}$  maps each element of  $dom(L_1)$  to an element of  $dom(L_2)$ .
  2. Given levels  $L_1$ ,  $L_2$  and  $L_3$  such that  $L_1 \prec L_2 \prec L_3$ , the function  $anc_{L_1}^{L_3}$  equals to the composition  $anc_{L_1}^{L_2} \circ anc_{L_2}^{L_3}$ .
  3. For each pair of levels  $L_1$  and  $L_2$  such that  $L_1 \prec L_2$  the function  $anc_{L_1}^{L_2}$  is monotone, i.e.,  $\forall x, y \in dom(L_1), L_1 \prec L_2: x \prec y \Rightarrow anc_{L_1}^{L_2}(x) \leq anc_{L_1}^{L_2}(y)$ .
  4. For each pair of levels  $L_1$  and  $L_2$  the  $anc_{L_1}^{L_2}$  function determines a set of finite equivalence classes  $X_i$  such that:  $\forall x, y \in dom(L_1), L_1 \prec L_2: anc_{L_1}^{L_2}(x) = anc_{L_1}^{L_2}(y) \Rightarrow x, y$  belongs to the same  $X_i$ .
  5. The relationship  $desc_{L_1}^{L_2}$  is the inverse of the  $anc_{L_1}^{L_2}$  function -i.e.,  $desc_{L_1}^{L_2}(1) = \{x \in dom(L) : anc_{L_1}^{L_2}(x) = 1\}$ .
- Each *data set*  $DS$  over a schema  $S = [L_1, \dots, L_n, M_1, \dots, M_m]$  is a finite set of tuples over  $S$  such that the set  $[L_1, \dots, L_n]$  comprises a primary key (in the usual sense).

- Each *selection condition*  $\varphi$  is a formula in disjunctive normal form. An *atom* of a selection condition is `true`, `false` or an expression of the form  $x \theta y$ , where  $\theta$  is an operator from the set  $\{>, <, =, \geq, \leq, \neq\}$  and each of  $x$  and  $y$  can be one of the following: (a) a level  $L$ , (b) a value  $l$ , (c) an expression of the form  $\text{anc}_{L_1}^{L_2}(L_1)$  where  $L_1 < L_2$  and (d) an expression of the form  $\text{anc}_{L_1}^{L_2}(l)$  where  $L_1 < L_2$  and  $l \in \text{dom}(L_1)$ . The *detailed equivalent* of  $\varphi$ , denoted by  $\varphi^0$ , is a selection condition obtained through the following procedure: for each occurrence of a level name  $L$  in  $\varphi$ , we substitute it with the equivalent expression  $\text{anc}_{L^0}^{L^0}(L^0)$ , where  $L^0$  is the detailed level of the dimension to which  $L$  belongs. Note that the detailed equivalent of a selection condition is directly applicable to a detailed data set.
- Each *cube*  $c$  over the schema  $[L_1, \dots, L_n, M_1, \dots, M_m]$ , is an expression of the form:  $c = (DS^0, \varphi, [L_1, \dots, L_n, M_1, \dots, M_m], [\text{agg}_1(M_1^0), \dots, \text{agg}_m(M_m^0)])$ , where  $DS^0$  is a detailed data set over the schema  $S = [L_1^0, \dots, L_n^0, M_1^0, \dots, M_m^0]$ ,  $m \leq k$ ,  $\varphi$  is a detailed selection condition,  $M_1^0, \dots, M_m^0$  are detailed measures,  $M_1, \dots, M_m$  are aggregated measures,  $L_1^0$  and  $L_i$  are levels such that  $L_1^0 < L_i$ ,  $1 \leq i \leq n$  and  $\text{agg}_i$ ,  $1 \leq i \leq m$  are aggregated functions from the set  $\{\text{sum}, \text{min}, \text{max}, \text{count}\}$ . The expression characterising a cube has the following formal semantics:  

$$c = \{x \in \text{Tup}(L_1, \dots, L_n, M_1, \dots, M_m) \mid \exists y \in \varphi(DS^0), x[L_i] = \text{anc}_{L_1^0}^{L_i^0}(y[L_1^0]), 1 \leq i \leq n, \\ x[M_j] = \text{agg}_j(\{q \mid \exists z \in \varphi(DS^0), x[L_i] = \text{anc}_{L_1^0}^{L_i^0}(z[L_1^0]), 1 \leq i \leq n, q = z[M_j^0]\}), 1 \leq j \leq m\}.$$
- The *Cube Algebra* (CA) is composed of three operations (consider a cube  $c^a = (DS^0, \varphi^a, [L_1^a, \dots, L_n^a, M_1^a, \dots, M_m^a], [\text{agg}_1^a(M_1^0), \dots, \text{agg}_m^a(M_m^0)])$  over which the operations are applied):
  1. *Navigate*: Let  $S = [L_1, \dots, L_n, M_1, \dots, M_m]$  be a schema and  $\text{agg}_1, \dots, \text{agg}_m$  be aggregate functions. If  $L_1^a$  and  $L_i$  belong to the same dimension  $D_i$  and  $\text{agg}_i \in \{\text{sum}, \text{min}, \text{max}, \text{count}\}$  then, navigation is defined as follows:  

$$\text{nav}(c^a, S, \text{agg}_1, \dots, \text{agg}_m) = (DS^0, \varphi^a, S, [\text{agg}_1(M_1^0), \dots, \text{agg}_m(M_m^0)]).$$
  2. *Selection*: Let  $\varphi$  be a selection condition applicable to  $c^a$ . Then, we define the *selection* operation as:  

$$\sigma_\varphi(c^a) = (DS^0, \varphi^a \wedge \varphi^0, [L_1^a, \dots, L_n^a, M_1^a, \dots, M_m^a], [\text{agg}_1^a(M_1^0), \dots, \text{agg}_m^a(M_m^0)])$$
where  $\varphi^0$  is the detailed equivalent of the selection condition  $\varphi$ .
  3. *Split measure*: Let  $M$  be a measure of the schema of the cube  $c$ . Without loss of generality, let us assume that  $M$  is  $M_m$ . Then *split measure* is defined as follows:  

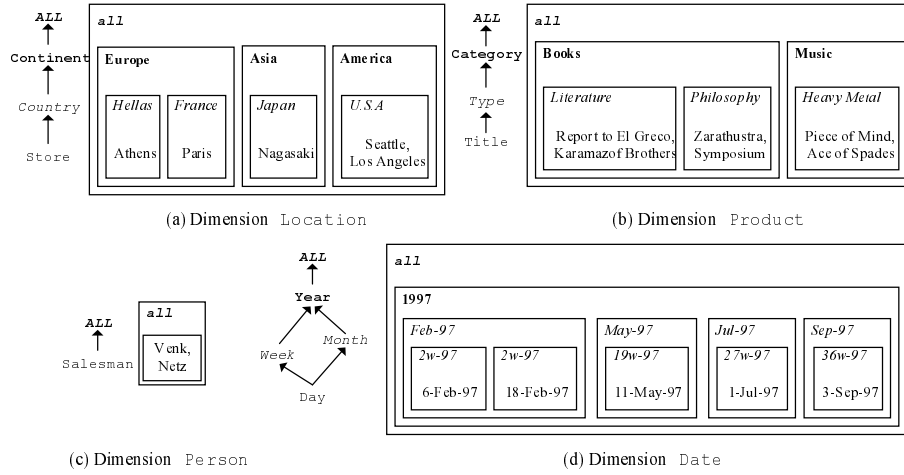
$$\pi_{M_m}(c^a) = (DS^0, \varphi^a, [L_1^a, \dots, L_n^a, M_1^a, \dots, M_{m-1}^a], [\text{agg}_1^a(M_1^0), \dots, \text{agg}_{m-1}^a(M_{m-1}^0)]).$$

Day	Title	Salesman	Store	Sales
6-Feb-97	Symposium	Netz	Paris	7
18-Feb-97	Karamazof brothers	Netz	Seattle	5
11-May-97	Ace of Spades	Netz	Los Angeles	20
3-Sep-97	Zarathustra	Netz	Nagasaki	50
3-Sep-97	Report to El Greco	Netz	Nagasaki	30
1-Jul-97	Ace of Spades	Venk	Athens	13
1-Jul-97	Piece of Mind	Venk	Athens	34

Fig. 1. Detailed Data Set  $DS^0$ .

**Theorem 1.** The Cube Algebra  $\mathcal{CA}$  is *sound* (i.e., the result of all the operations is always a cube) and *complete* (i.e., any valid cube can be computed as the combination of a finite set of  $\mathcal{CA}$  operations). ■

**Example 1.** To motivate the discussion we customise the example presented in [14] to an international publishing company with travelling salesmen selling books and CD's to stores all over the world. The database (Fig. 1) stores information about the sales of a title that a salesman achieved on a particular date and city. The dimensions of our example are Person, Location, Product and Date (Fig. 2). Measure Sales is functionally dependent on dimensions Date, Product, Person and Location. ■



**Fig. 2.** Dimensions

The organisation of information in different levels of aggregation (i.e., dimensions) is in hand because OLAP users are unlikely to directly ask questions about the detailed data that are stored in the database. Instead, they are more interested in aggregated information according to the categorisation groupings.

Following, we present three queries and the respective algebraic representation that could have been a typical sequence of operations during an OLAP session.

**Query 1.** Find the maximum sales by month, category of item, salesman and country.

$$c^1 = \text{nav}(\text{DS}^0, [\text{Month}, \text{Category}, \text{Salesman}, \text{Country}, \text{Max\_val}], \text{max}(\text{sales})) = (\text{DS}^0, \text{true}, [\text{Month}, \text{Category}, \text{Salesman}, \text{Country}, \text{Max\_val}], \text{max}(\text{sales}))$$

**Query 2.** Find the maximum sales outside the American continent by month, category of item, salesman and country.

$$c^2 = \sigma_{\text{anc}_{\text{continent}}(\text{country}) \neq \text{'America'}}(c^1) = (\text{DS}^0, \text{anc}_{\text{city}}^{\text{continent}}(\text{City}) \neq \text{'America'}, [\text{Month}, \text{Category}, \text{Salesman}, \text{Country}, \text{Max\_val}], \text{max}(\text{sales})).$$

**Query 3.** Find the summary of sales outside the continent of America by month, type of title and country of store.

$$c^3 = \text{nav}(c^2, [\text{Month}, \text{Type}, \text{All}, \text{Country}, \text{Sum\_val}], \text{sum}(\text{Sales})) = (\text{DS}^0, \text{anc}_{\text{city}}^{\text{continent}}(\text{City}) \neq \text{'America'}, [\text{Month}, \text{Type}, \text{All}, \text{Country}, \text{Sum\_val}], \text{sum}(\text{sales})).$$

During this particular OLAP session the user has performed (a) a roll-up from the detailed data set, (b) a selection and (c) a slicing (of dimension `Person`), a drill down (from `Category` to `Type` level) and a change in the aggregation function (from `max` to `sum`).

In the first operation, one can notice that the semantics of the navigation operation allow us to use an arbitrary name (e.g., `Max_val`) for the measure that computes the maximum value per group of aggregation. In the second operation, notice that the expression  $\text{anc}_{\text{country}}^{\text{continent}}(\text{Country})$  which is directly applicable to the schema (and data) of the cube  $c^1$  is transformed to its equivalent  $\text{anc}_{\text{city}}^{\text{continent}}(\text{City})$ , that directly applies to the detailed data set  $DS^0$ , through the use of the definition of the detailed selection condition.

The presented model stresses the fact that *a cube we can treated both as a query and as a set of tuples*. We believe that this aspect of OLAP was neglected in the previous approaches. In this example, the contribution of treating cubes as views over the detailed data set is eminent. Actually, the fact that we have retained the history of selections permits us to be able to drill-down and change the aggregation function. Otherwise, to perform the drill-down operations we should employ a join operation of  $c^2$  with  $DS^0$ . The same also holds for the change in the aggregation function. Using the history of selections we can (a) avoid to perform a costly join operation and (b) possibly further optimise the execution of the operation through the use of already computed cubes. The second possibility will be investigated in Section 3.

As we have already stressed, this is a *logical* model for cubes. We do not advocate that the *physical* computation of the results of an operation should actually be computed all the way back from the detailed data set. Actually, although drill-down and change of aggregation function can be performed directly, only through the use of the semantics of our model, can the selection and roll-up operations be performed over the original cube, without referring to the detailed data set. In the case of selection, it suffices to simply pass all the tuples of the cube from the filter of the applied selection condition. In the case of roll-up to coarser levels of granularity, it also suffices to group the tuples of the cube and apply the appropriate aggregate function. These simple optimisation strategies are generalised in Section 3 with a more powerful approach, capable of detecting whether any cube can be computed from the data of another cube, simply by comparing their definitions.

### 3 The Cube Usability Problem

**Problem description.** There are several cases where there is the need to decide whether a view can be recomputed from another view. In the case of OLAP, the problem can be stated as follows: the OLAP user selects some data and performs an operation over them. The result of the new query can be computed, of course, from the detailed data. Nevertheless, it is possible that previously computed and cached results, or existing materialised views, could also allow the computation of the requested information. As a general statement, we could say that the problem lies in whether the

computation of a new cube can be performed from an intermediate level of aggregation, than from the detailed data set.

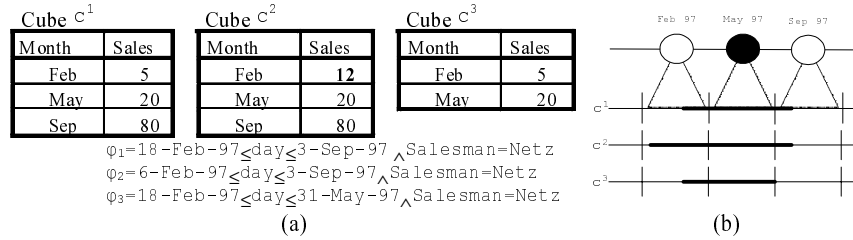
Formally, let  $DS^0$  be a detailed data set. Let also  $c^{old}$  and  $c^{new}$  be two cubes defined over  $DS^0$ . By definition, cubes  $c^{old}$  and  $c^{new}$  can be calculated from  $DS^0$ . The *cube usability problem* lies on determining whether the tuples of  $c^{old}$  can be used to compute cube  $c^{new}$ . It is clear that the cube usability problem is a variant of the *view subsumption* problem, already investigated in the field of relational databases [18].

**Shortcomings of current approaches.** Too much effort has been spent, in the past, to tackle the problem of view subsumption and query rewriting in the presence of views [4,5,8,22]. Nevertheless, the previous approaches are relational-oriented and lack to deal with specific characteristics of the multidimensional modelling. We will use two examples to demonstrate these shortcomings.

**Example 2.** Intuitively, someone would expect, that in order to solve the cube usability problem, the new cube  $c^{new}$  should:

1. be defined over the same dimensions with  $c^{old}$  and at a higher or equal level;
2. be defined over the same measure of  $DS^0$ . Moreover, the aggregation functions  $agg^{new}$  and  $agg^{old}$  should be the same;
3. have a more restrictive selection condition than  $c^{old}$ , i.e.,  $\phi^{new}$  is contained in  $\phi^{old}$  in the usual relational sense.

Checking conditions 1 and 2 is an easy task. To perform the comparison of Condition 3, we need to transform the selection conditions of the two cubes in order to treat them as conjunctive queries [17]. One could argue that existing relational techniques are adequate to handle this problem. Unfortunately, as we will show, there are cases where those techniques are not sufficient.



**Fig. 3.** Cube usability problems with marginal conditions.

Let us consider the detailed data set  $DS^0$  of Fig. 1. Let  $c^i$ ,  $1 \leq i \leq 3$  be cubes defined as  $c^i = [DS^0, \phi_i, [Month, ALL, ALL, ALL, ALL, ALL, Sales], \text{sum}(\text{sales})]$ . Fig. 3a presents the Month level, the Sales measure and the selection conditions for each of the cubes. The problem is whether a new cube  $c^3$  can be computed using the tuples of one of the existing cubes  $c^1$  and  $c^2$ . Since Conditions 1, 2 and 3 hold, one could argue that this is feasible. Yet, as we can see in Fig. 3a, only  $c^1$  can be used to compute  $c^3$ . The intuitive explanation of the problem is depicted in Fig. 3b. There are three horizontal axes defined at the day level, each for one of the cubes  $c^1$ ,  $c^2$  and  $c^3$ . Each bold line denotes the set of days participating in the computation of the respective cube. All cubes are defined at the month level; consequently, we partition the three axes with respect to the function  $\text{anc}_{\text{day}}^{\text{month}}$ . As we can see, we have three



partitions: Feb' 97, May' 97 and Sep' 97. Cube  $c^3$  can be computed from  $c^1$  because for all the partitions of  $c^3$  (i.e., Feb' 97, May' 97), cubes  $c^1$  and  $c^3$  cover exactly the same days. This does not hold for  $c^1$  and  $c^2$ . ■

**Example 3.** Suppose the case, where a cube  $c^1$  has a selection condition  $\varphi_1 = \text{arr.year} < \text{dep.year}$  (where `arr` denotes dimension arrival date and `dep` denotes the dimension departure date). Suppose also that a cube  $c^2$  is defined at the month level and has a selection condition  $\varphi_2 = \text{arr.month} < \text{dep.month}$ . We can see that cube  $c^1$  can be computed from  $c^2$ . This means that if  $c^2$  is materialised we can use its tuples to compute  $c^1$ . We are able to perform this kind of reasoning because we take advantage of the relationship between months and years, expressed through the dimension hierarchies, and the family of `anc` functions. To our knowledge, there is no effort in the view subsumption literature that uses this kind of knowledge. ■

**Contribution.** In this section, we will show that the cube usability problem is reduced to simple tests and operations. Different tests apply for different classes of queries. We explore the selection conditions of two categories: (a) selection conditions with atoms involving values (i.e., of the form  $L\theta 1, L\theta \text{anc}_{L_1}^{L_2}(1)$ , etc.) and (b) selection conditions with atoms involving only levels (i.e., of the form  $L_1\theta L_2, L\theta \text{anc}_{L_1}^{L_2}(L_1)$ , etc.). We will examine the optimisation issues for the former in Section 3.1 and for the latter in Section 3.2. Finally, Section 3.3 presents a theorem with sufficient criteria and the corresponding rewriting algorithm for both cases of the cube usability problem under consideration.

In the rest of the paper, for reasons of simplicity, we will deal with cubes having only one measure. All our results can be easily extended to cubes having an arbitrary number of measures [5]. Let  $c^{\text{new}} = (DS^0, \varphi^{\text{new}}, [L^{\text{new}}, M^{\text{new}}], \text{agg}^{\text{new}}(M))$  be the new cube and  $c^{\text{old}} = (DS^0, \varphi^{\text{old}}, [L^{\text{old}}, M^{\text{old}}], \text{agg}^{\text{old}}(M))$  be the candidate cube, where  $L^{\text{new}}$  and  $L^{\text{old}}$  are sets of levels coming from dimension sets  $D^{\text{new}}$  and  $D^{\text{old}}$  respectively,  $M^{\text{new}}$  and  $M^{\text{old}}$  are measures, and finally,  $\text{agg}^{\text{new}}$  and  $\text{agg}^{\text{old}}$  are aggregate functions.

### 3.1 Equivalent Transformations for Atoms Involving Values

Suppose two levels  $L^{\text{old}}$  and  $L^{\text{new}}$ , such that  $L^{\text{old}} < L^{\text{new}}$ . Function  $\text{anc}_{L^{\text{old}}}^{L^{\text{new}}}$  defines a partition over the values of  $L^{\text{old}}$  with respect to the values of  $L^{\text{new}}$  (e.g., the partition of `year` to `month`). Suppose now, two atoms  $a_1$  and  $a_2$  over  $L^{\text{old}}$ , as in the case of Fig. 3. To perform an aggregation to  $L^{\text{new}}$ , the two atoms must hold the same ranges of values for each and every partition that  $L^{\text{new}}$  defines over  $L^{\text{old}}$ . Generalising this observation, in the case where two selection conditions involve a larger conjunction of atoms, we must:

1. transform the selection conditions to concrete ranges for each dimension;
2. reduce the atoms to the same level, using appropriate transformations (so that they can be compared);
3. check whether the broader selection condition is defined identically for the marginal constraints of the other selection condition.

The following auxiliary definition introduces the notion of *dimension interval*, which is a concrete range over the domain of a certain dimension level.

**Definition 1:** A *dimension interval* (DI) is one of the following (a) `true`, (b) `false` and (c) an expression of the form  $l_1 \leq L \leq l_2$ , where  $L$  is a variable ranging over the level of a dimension and  $l_1$  and  $l_2$  are values. ■

Atom	Dimension Interval
<code>true</code>	<code>True</code>
<code>false</code>	<code>False</code>
$\text{anc}_L^{L'}(L) = l$	$\min(\text{desc}_L^{L'}(l)) \leq L \leq \max(\text{desc}_L^{L'}(l))$
$\text{anc}_L^{L'}(L) < l$	$-\infty < L \leq \max(\text{desc}_L^{L'}(\text{prev}(l)))$
$l < \text{anc}_L^{L'}(L)$	$\min(\text{desc}_L^{L'}(\text{next}(l))) \leq L < +\infty$
$\text{anc}_L^{L'}(L) \leq l$	$-\infty < L \leq \max(\text{desc}_L^{L'}(l))$
$l \leq \text{anc}_L^{L'}(L)$	$\min(\text{desc}_L^{L'}(l)) \leq L < +\infty$

**Fig. 4.** Transformation from atoms to dimension intervals

Fig. 4 shows how single atoms can be transformed to DI's. Values  $-\infty$  and  $+\infty$  have the obvious semantics. Moreover, functions `prev` and `next` result in the previous and the following value of  $l$  in the domain of  $L$  respectively.

**Algorithm** `Check_Atoms_Usability`.

**Input:** Two conjunctions of atoms **a** and **b** involving only values, and a set of levels  $L'$ .

**Output:** `true` if  $\mathbf{a} \subseteq_{L'} \mathbf{b}$ , `false` otherwise.

1. Write all atoms of **a** and **b** as DI's using the transformations of Fig. 4 (where the level  $L$  is the detailed level of each dimension).
2. Group all DI's of **a** and **b** by dimension level and produce for every set a single DI' having the most restrictive boundaries. Let **a'** and **b'** be the result, respectively.
3. If there exists a DI `false` in **a'** Then Return `true`.
4. If there exists a DI `false` in **b'** Then Return `false`.
5. For every DI  $a$  of **a'**
  6. If  $a$  is defined over dimension level  $D_i$ ,  $L^0$  that does not exist in any DI of **b'** Then
  7. Introduce DI  $-\infty \leq D_i, L^0 \leq \infty$  to **b'**.
  8. EndFor
9. If **b'** has more DI's than **a'** Then Return `false`.
10. For every DI  $a = (A_s, A_e)$  of **a'**
  11. Let the DI  $b = (B_s, B_e)$  of **b'** involving the same dimension with  $a$ . Let also  $L'$  be the respective level in  $L'$ .
  12. Case  $A_s < B_s$  or  $B_e < A_e$  or  $b = \text{false}$
  13. Return `false`
  14. Case  $A_s \neq \min(\text{desc}_{L'}^{L'}(\text{anc}_{L'}^{L'}(A_s)))$  and  $A_s \neq B_s$
  15. Return `false`
  16. Case  $A_e \neq \max(\text{desc}_{L'}^{L'}(\text{anc}_{L'}^{L'}(A_e)))$  and  $A_e \neq B_e$
  17. Return `false`
  18. EndFor
  19. Return `true`

**Fig. 5.** Algorithm `Check_Atoms_Usability`

In general, to determine whether a cube  $c^{\text{old}}$  can be used for the computation of  $c^{\text{new}}$ , we need to partition the detailed level of each dimension according to the

respective level of  $c^{new}$ . If for each partition of  $c^{new}$ , there exists an identical partition of  $c^{old}$ , then  $c^{old}$  can be used to compute  $c^{new}$ . We formalise this relationship between two cubes, through Definition 2.

**Definition 2. L-containment:** Let  $\mathbf{D}$  be a set of dimensions and  $\varphi^{old}, \varphi^{new}$  be two selection conditions involving levels only from  $\mathbf{D}$ . Let  $\mathbf{L}$  be a set of levels, each belonging to a different dimension of  $\mathbf{D}$ . Let also the two cubes  $c^{new} = (DS^0, \varphi^{new}, [\mathbf{L}, \mathbf{M}], \text{agg}(\mathbf{M}))$  and  $c^{old} = (DS^0, \varphi^{old}, [\mathbf{L}, \mathbf{M}], \text{agg}(\mathbf{M}))$ , defined over an arbitrary detailed data set  $DS^0$ . Selection condition  $\varphi^{new}$  is **L-contained** in  $\varphi^{old}$  (denoted by  $\varphi^{new} \subseteq_L \varphi^{old}$ ) if  $c^{new} \subseteq c^{old}$  for any data set  $DS^0$ . ■

To tackle the problem of cube usability between cubes of different aggregation granularities, we introduce Algorithm `Check_Atoms_Usability` that is checking the containment of conjunctions of atoms that involve values. Notice that our analysis does not include  $\neq$ . This case will be handled in Section 3.3.

For Example 1, Algorithm `Check_Atoms_Usability` deduces that  $\varphi_1$  L-contains  $\varphi_3$  (with respect to level `Month`), while  $\varphi_2$  does not. Moreover, it is interesting to see that if one considers the `year` level, neither  $\varphi_1$  nor  $\varphi_2$  L-contains  $\varphi_3$ .

### 3.2 Equivalent Transformations for Atoms Involving only Levels

Following [17], we assume the existence of two infinite, totally ordered domains,  $\mathbf{L}$  and  $\mathbf{L}'$  isomorphic to the integers. Let also  $f$  be a total, monotone function over  $\mathbf{L}$ , mapping the values of domain  $\mathbf{L}$  to the values of domain  $\mathbf{L}'$ . The family of `anc` functions fulfils these requirements.

We assume that we are given a collection of inequalities of the form  $X < Y$ ,  $X \leq Y$ ,  $X \neq Y$ ,  $f(X) < f(Y)$ ,  $f(X) \leq f(Y)$ ,  $f(X) \neq f(Y)$  and equalities of the form  $f(X) = f(Y)$ . We do not allow equalities of the form  $X = Y$ . If such a subgoal is found in a query, we substitute every occurrence of  $X$  with  $Y$ . We also eliminate any pair of inequalities  $f(X) \leq f(Y)$  and  $f(Y) \leq f(X)$ , where  $X, Y$  are distinct variables, with  $f(X) = f(Y)$ .

We will use the following set of axioms for these inequalities:

A1	$X \leq X$	A8	$X \leq Z, Z \leq Y, X \leq W, W \leq Y$ and $W \neq Z$ imply $X \neq Y$
A2	$X < Y$ implies $X \leq Y$	A9	$X \leq Y$ implies $f(X) \leq f(Y)$
A3	$X < Y$ implies $X \neq Y$	A10	$f(X) < f(Y)$ implies $X < Y$
A4	$X \leq Y$ and $X \neq Y$ imply $X < Y$	A11	$f(X) \neq f(Y)$ implies $X \neq Y$
A5	$X \neq Y$ implies $Y \neq X$	A12	$f(X) \leq f(Y)$ and $f(Y) \leq f(X)$ implies $f(X) = f(Y)$
A6	$X < Y$ and $Y < Z$ imply $X < Z$	A13	$f(X) = f(Y)$ and $f(Y) \leq f(Z)$ implies $f(X) \leq f(Z)$
A7	$X \leq Y$ and $Y \leq Z$ imply $X \leq Z$	A14	$f(X) = f(Y)$ and $f(Y) \neq f(Z)$ implies $f(X) \neq f(Z)$
		A15	$f(X) = f(Y)$ implies $f(X) \leq f(Y)$

**Fig. 6.** Axioms for L-containment checking.

We assume that our models are assignments of integers to variables. Expressions of the form  $f(X)$  are also treated as variables. For variables of the form  $X$  we apply axioms A1 to A9 and for variables of the form  $f(X)$  we apply axioms A1 to A15.

**Theorem 2.** The axioms A1–A15 are sound and complete. ■

In order to check whether one set of inequalities  $T$  follows from another set of inequalities  $S$  we compute the closure  $S^+$  by applying the axioms A1-A15 until they no longer generate any new inequalities. Then, we check whether  $T$  is a subset of  $S^+$ .

### 3.3 Testing Cube Usability

In this section, we combine the results of Sections 3.1 and 3.2 to provide a test for several cases of cube usability. One can transform any kind of formula using logical transformations [6] to an equivalent formula consisting of disjunctions of conjunctions which do not involve  $\neq$  and  $\neg$ . Theorem 3 provides sufficient criteria for a cube  $c^{old}$  to be used for the computation of another cube  $c^{new}$ . Algorithm `Cube_Usability` describes the specific steps to be followed for this computation.

**Theorem 3.** Suppose a detailed data set  $DS^0 = [L_1^0, \dots, L_n^0, M^0]$  and two cubes  $c^{old} = (DS^0, \varphi_{old}, [L_1^{old}, \dots, L_n^{old}, M_{old}], agg_{old}(M^0))$  and  $c^{new} = (DS^0, \varphi_{new}, [L_1^{new}, \dots, L_n^{new}, M_{new}], agg_{new}(M^0))$ . If

1.  $agg_{old} = agg_{new}$ ,
2.  $L_i^{old} \prec L_i^{new}$ ,  $1 \leq i \leq n$ , and
3. one of the following two cases holds for  $\varphi_{old}$  and  $\varphi_{new}$ :
  - $\varphi_{old}$  and  $\varphi_{new}$  involve conjunctions of atoms only of the form  $L_i \theta L_j$ , all the levels  $L_i, L_j$  are higher from the respective levels of the schema of  $c^{old}$  (i.e.  $L_{i,j}^{old} \prec L_{i,j}$ ) and  $\varphi_{old}$  belongs to the closure of  $\varphi_{new}$ , or,
  - $\varphi_{old}$  and  $\varphi_{new}$  involve conjunctions of atoms of the form  $L \theta 1$  and  $\varphi_{new} \subseteq [L_1^{new}, \dots, L_n^{new}] \varphi_{old}$ ,

then Algorithm `Cube_Usability` correctly computes  $c^{new}$  from the tuples of  $c^{old}$ .  $\square$

Theorem 3 tests for usability, pairs of cubes involving conjunctive selection conditions which do not involve  $\neq$  and  $\neg$ . Cubes involving disjunctive selection conditions can be treated in the usual way [17].

**Example 4.** Let  $c^{new}$  and  $c^{old}$  be the cubes over  $DS^0$  of Fig. 1 defined as follows.

$$c^{old} = (DS^0, \varphi_{old}, [Month, Country, Type, Salesman, Sum\_old], sum(Sales))$$

$$\text{and}$$

$$c^{new} = (DS^0, \varphi_{new}, [Month, Country, Category, Salesman, Sum\_new], sum(Sales))$$

where  $\varphi_{old} = 18\text{-Feb-97} \leq \text{day} \wedge \text{day} \leq 3\text{-Sep-97} \wedge \text{anc}_{\text{Item}}^{\text{Category}}(\text{Item}) = \text{"Books"}$

and  $\varphi_{new} = 1\text{-Mar-97} \leq \text{day} \wedge \text{day} \leq 3\text{-Sep-97} \wedge \text{"Literature"} \leq \text{anc}_{\text{Item}}^{\text{Type}}(\text{Item}) \wedge \text{anc}_{\text{Item}}^{\text{Type}}(\text{Item}) \leq \text{"Philosophy"}$ .

To check whether  $c^{new}$  can be computed from  $c^{old}$  we apply Theorem 3. The schemata and aggregation functions of the two cubes are compatible (conditions (a), (b) of Theorem 3). Moreover,  $\varphi_{new}$  is  $L$ -contained from  $\varphi_{old}$  with respect to the levels of  $c^{new}$ . Following, Lines 2-10 of Algorithm `Cube_Usability`, we transform  $\varphi_{new}$  so that it can be applied to the schema of cube  $c^{old}$ . The transformations of Lines 3-8 result in

$$\varphi_{new \circ} = \text{Mar-97} \leq \text{Month} \wedge \text{Month} \leq \text{Sep-97} \wedge \text{"Literature"} \leq \text{Type} \wedge \text{Type} \leq \text{"Philosophy"}.$$

**Algorithm** Cube\_Usability.

**Input:** A detailed data set  $DS^0 = [L_1^0, \dots, L_n^0, M^0]$  and two cubes  $c^{old} = (DS^0, \varphi_{old}, [L_1^{old}, \dots, L_n^{old}, M_{old}], agg_{old}(M^0))$  and  $c^{new} = (DS^0, \varphi_{new}, [L_1^{new}, \dots, L_n^{new}, M_{new}], agg_{new}(M^0))$  such that  $\varphi_{old}$  and  $\varphi_{new}$  involve either (a) conjunctions of atoms of the form  $L\theta l$  or (b) conjunctions of atoms of the form  $L\theta L'$  where  $L$  and  $L'$  are levels and  $l$  is a value.

**Output:** A rewriting that calculates cube  $c^{new}$  from the tuples of  $c^{old}$ .

1. If all atoms of  $\varphi_{old}$  and  $\varphi_{new}$  involve conjunctions of atoms of the form  $L\theta l$  Then
2.     For every atom  $a = anc_{L^0}^{L^0}(l) \theta l$  in  $\varphi_{new}$  (or equivalent to this form)
3.         If  $L^{old}$  is the respective level in the schema of  $c^{old}$  and  $L^{old} < L$  Then
4.             Transform  $a$  to  $anc_{L^{old}}^{L^0}(l) \theta l$
5.         EndIf
6.         ElseIf  $L^{old}$  is the respective level in the schema of  $c^{old}$  and  $L < L^{old}$  Then
7.             Transform  $a$  to  $L^{old} \theta' anc_{L^0}^{L^{old}}(l)$  where  $\theta' = \theta$  except for two cases:
  - (a)  $a = anc_{L^0}^{L^0}(l) < l$  and  $l \neq \min(desc_{L^0}^{L^{old}}(anc_{L^0}^{L^{old}}(l)))$  where  $\theta' = \leq$ ,
  - (b)  $a = anc_{L^0}^{L^0}(l) > l$  and  $l \neq \max(desc_{L^0}^{L^{old}}(anc_{L^0}^{L^{old}}(l)))$  where  $\theta' = \geq$
8.         EndIf
9.     EndFor
10. EndIf
11. If all atoms of  $\varphi_{old}$  and  $\varphi_{new}$  involve conjunctions of atoms of the form  $a = anc_{L^0}^{L^0}(L^0) \theta anc_{L^{0'}}^{L^{0'}}(L^{0'})$  (or equivalent to this form), where both  $L$  and  $L'$  are higher than the respective levels of  $c^{old}$  Then
12.     For every atom  $a = anc_{L^0}^{L^0}(L^0) \theta anc_{L^{0'}}^{L^{0'}}(L^{0'})$  in  $\varphi_{new}$
15.         Transform  $a$  to  $anc_{L^{old}}^{L^0}(L^{old}) \theta anc_{L^{old'}}^{L^{old'}}(L^{old'})$
16.     EndFor
17. EndIf
18. Apply the transformed selection condition to  $c^{old}$  and derive a new data set  $DS^1$ .
19. Replace all the values of  $DS^1$  with their ancestor values at the levels of  $c^{new}$ , resulting in a new data set  $DS^2$ .
20. Aggregate ("group by" in the relational semantics) on the tuples of  $DS^2$ , so that we produce  $c^{new}$ .

**Fig. 7.** Algorithm Cube\_Usability

Month	Type	Salesman	Country	Sum_old
Feb-97	Literature	Netz	USA	5
Sep-97	Philosophy	Netz	Japan	50
Sep-97	Literature	Netz	Japan	30

(a)

Month	Type	Salesman	Country	Sum_1
Sep-97	Philosophy	Netz	Japan	50
Sep-97	Literature	Netz	Japan	30

(b)

Month	Category	Salesman	Country	Sum_2
Sep-97	Book	Netz	Japan	50
Sep-97	Book	Netz	Japan	30

(c)

Month	Category	Salesman	Country	Sum_new
Sep-97	Book	Netz	Japan	80

(d)

**Fig. 8.** Calculating  $c^{new}$  from  $c^{old}$

We apply the transformed selection condition to  $c^{old}$  (depicted in Fig. 8a) and derive a new data set  $DS^1$  (depicted in Fig. 8b). Then, we replace all the values of  $DS^1$  with their ancestor values at the levels of  $c^{new}$  (Line 19), resulting in a new data set  $DS^2$

(depicted in Fig. 8c). Finally, we aggregate the tuples of  $DS^2$  and we produce  $c^{new}$  (depicted in Fig. 8d). ■

## 4 Discussion and Future Work

We have presented a *logical* model for cubes based on the key observation that a cube is not a self-existing entity, but rather a view over an underlying data set. The proposed model is powerful enough to capture all the commonly encountered OLAP operations such as selection, roll-up and drill-down, through a sound and complete algebra. We have showed how this model can be used as the basis for processing cube operations and have provided syntactic characterisations for the problems of cube usability. Theorem 3, which provides these syntactic characterisations, is very important for the usual operations of the model. Two of the most eminent cases are: (a) navigation from a certain cube  $c$  to a cube having all its levels higher (or equal) than the respective levels of  $c$  and (b) selection over a certain cube  $c$  where all the levels acting as variables are higher (or equal) than the levels of  $c$ .

Of course, the applicability of Theorem 3 is not restricted in these two simple cases. Normally, an OLAP screen contains more than one cubes [14]. Thus, an interactive OLAP session produces many cubes which possibly overlap. Computing a new set of cubes can possibly be achieved by using already computed and cached cubes (provided that they fulfil the criteria of Theorem 3). Consequently, the results on the problem of cube usability can be used both for the query optimisation and the caching processes. The cube usability results can also be applied in the problem of data warehouse design, where the optimal set of views (with respect to query and maintenance cost) has to be derived. Testing for cube usability can avoid redundancy in the final data warehouse schema and improve the run-time of the design algorithm [12].

As future work, we plan to incorporate our results in a system under construction in NTUA. The modelling parts could be extended to take into account aspects of the hierarchy structure (partial ancestor functions, hierarchies that are not well captured as lattices, etc.). The theoretical results over query processing can be extended to handle optimisation issues for a broader set of selection conditions, partial rewritings and optimisation of the physical execution for cube operations. Finally, a challenging issue is how to devise smarter algorithms for the cube usability problems.

## References

1. R. Agrawal, A. Gupta, and S. Sarawagi. Modelling multidimensional databases. Technical report, IBM Almaden Research Center, San Jose, California, 1995.
2. L. Cabbibo and R. Torlone. Querying Multidimensional Databases. *6th International Workshop on Database Programming Languages (DBPL6)*, 1997.
3. S. Chaudhuri, K. Shim. Optimizing Queries with Aggregate Views. In *Proceedings of the 5th International Conference on Extending Database Technology (EDBT-96)*, Avignon, France, March 25-29, 1996.

4. S. Chaudhuri, S. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *Proceedings of the 11th International Conference on Data Engineering (ICDE)*, IEEE Computer Society, pp. 190-200, Taipei, March 1995.
5. S. Cohen, W. Nutt, A. Serebrenik. Rewriting Aggregate Queries Using Views. *Proceedings of the 18<sup>th</sup> ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Philadelphia, Pennsylvania. ACM Press, 1999.
6. H.B. Enderton, A Mathematical Introduction to Logic. Academic Press, 1972.
7. M. Gebhardt, M. Jarke and S. Jacobs. A toolkit for negotiation support on multi-dimensional data. In *Proceedings of ACM SIGMOD International Conference on Management of Data*. Tucson, Arizona, 1997.
8. A. Gupta, V. Harinarayan, and D. Quass. Aggregate query processing in data warehouses. In *Proceedings of the 21<sup>st</sup> International Conference on Very Large Data Bases (VLDB)*, Zurich, Switzerland, Morgan Kaufmann Publishers, August 1995.
9. M. Gyssens, L.V.S. Lakshmanan. A Foundation for Multi-Dimensional Databases. In *Proceedings of the 23<sup>rd</sup> International Conference on Very Large Databases (VLDB)*, Athens, August 1997.
10. W. Lehner. Modeling Large Scale OLAP Scenarios. In *Proceedings of the 6<sup>th</sup> International Conference of Extending Database Technology (EDBT-98)*, 1998.
11. C. Li, X. Sean Wang. A Data Model for Supporting On-Line Analytical Processing. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, 1996.
12. S. Ligoudistianos, T. Sellis, D. Theodoratos, and Y. Vassiliou. Heuristic Algorithms for Designing the Data Warehouse with SPJ Views. In *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery, (DaWaK)*, Lecture Notes in Computer Science, Vol. 1676, Springer, 1999.
13. Metadata Coalition. Metadata Interchange Specification (MDIS v. 1.1). <http://www.metadata.org/standards/toc.html> 1997.
14. Microsoft Corp. OLEDB for OLAP February 1998. Available at <http://www.microsoft.com/data/oledb/olap/>
15. OLAP Council. The APB-1 Benchmark. 1997. Available at <http://www.olapcouncil.org/research/bmarkly.htm>
16. TPC. TPC Benchmark H and TPC Benchmark R. Transaction Processing Council. June 1999. Available at <http://www.tpc.org/>
17. J. Ullman. Principles of Database and Knowledge-Base Systems. Volume II: The New Technologies. Computer Science Press. 1989.
18. J.D. Ullman. Information integration using logical views. *Proceedings of the 6th International Conference on Database Theory (ICDT-97)*, Lecture Notes in Computer Science, pp. 19-40. Springer-Verlag, 1997.
19. P. Vassiliadis, T. Sellis. A Survey on Logical Models for OLAP Databases. *SIGMOD Record*, vol. 28, no. 4, December 1999.
20. P. Vassiliadis, S. Skiadopoulos. (Not) Yet Another Model for Multidimensional Databases (Extended version). Technical Report, KDBSL 1999. Available at <http://www.dblab.ece.ntua.gr/~pvassil/publications/cube99.ps.gz>
21. P. Vassiliadis. Modeling Multidimensional Databases, Cubes and Cube Operations. In *Proceedings of 10<sup>th</sup> International Conference on Scientific and Statistical Database Management (SSDBM)*, Capri, Italy, July 1998.
22. P. Larson, H. Z. Yang. Computing Queries from Derived Relations. In *Proceedings of the 11<sup>th</sup> International Conference on Very Large Data Bases (VLDB)*, Stockholm, Sweden, Morgan Kaufmann Publishers, August 1985.