A Logical Approach to Multidimensional Databases*

Luca Cabibbo and Riccardo Torlone

Dipartimento di Informatica e Automazione, Università di Roma Tre
Via della Vasca Navale, 79 — I-00146 Roma, Italy
E-mail: {cabibbo,torlone}@dia.uniroma3.it

Abstract. In this paper we present $\mathcal{MD}$, a logical model for OLAP systems, and show how it can be used in the design of multidimensional databases. Unlike other models for multidimensional databases, $\mathcal{MD}$ is independent of any specific implementation (relational or proprietary multidimensional) and as such it provides a clear separation between practical and conceptual aspects. In this framework, we present a design methodology, to obtain an $\mathcal{MD}$ scheme from an operational database. We then show how an $\mathcal{MD}$ database can be implemented, describing translations into relational tables and into multidimensional arrays.

1 Introduction

An enterprise can achieve a great competitive advantage from the analysis of its historical data. For instance, the identification of unusual trends in sales can suggest opportunities for new business, whereas the analysis of past consumer demand can be useful for forecasting production needs. A *data warehouse* is an integrated collection of enterprise-wide data, oriented to decision making, that is built to support this activity [8, 9]. Actually, data analysis is not performed directly on the data warehouse, but rather on special data stores derived from it, often called *hypercubes* or *multidimensional* "fact" tables. These terms originate from the fact that the effectiveness of the analysis is related to the ability of describing and manipulating factual data according to different and often independent perspectives or "dimensions," and that this picture can be naturally represented by means of n-dimensional arrays (or cubes). As an example, in a commercial enterprise, single sales of items (the factual data) provide much more information to business analysts when organized according to dimensions like category of product, geographical location, and time. The collection of fact tables of interest for an enterprise forms a *multidimensional database*.

Traditional database systems are inadequate for multidimensional analysis since they are optimized for on-line transaction processing (OLTP), which corresponds to large numbers of concurrent transactions, often involving very few records. Conversely, multidimensional database systems should be designed for

* This work was partially supported by *CNR* and by *MURST*.

the so-called *on-line analytical processing* [5] (OLAP), which involves few complex queries over very large numbers of records. Current technology provides both OLAP data servers and client analysis tools. OLAP servers can be either relational systems (ROLAP) or proprietary multidimensional systems (MOLAP). A ROLAP system is an extended relational system that maps operations on multidimensional data to standard relational operations (SQL). A MOLAP system is instead a special server that directly represents and manipulates data in the form of multidimensional arrays. The clients offer querying and reporting tools, usually based on interactive graphical user interfaces, similar to spreadsheets.

In the various systems [11], multidimensional databases are modeled in a way that strictly depends on the corresponding implementation (relational or proprietary multidimensional). This has a number of negative consequences. First, it is difficult to define a design methodology that includes a general, conceptual step, independent of any specific system but suitable for each. Second, in specifying analytical queries, the analysts often need to take care of tedious details, referring to the "physical" organization of data, rather than just to the essential, "logical" aspects. Finally, the integration with database technology and the optimization strategies are often based on ad-hoc techniques, rather than any systematic approach. As others [1, 7], we believe that, similarly to what happens with relational databases, a better understanding of the main problems related to the management of multidimensional databases can be achieved only by providing a logical description of business data, independent of the way in which data is stored.

In this paper we study conceptual and practical issues related to the design of multidimensional databases. The framework for our investigation is $\mathcal{MD}$, a logical model for OLAP systems that extends an earlier proposal [3]. This model includes a number of concepts that generalize the notions of dimensional hierarchies, fact tables, and measures, commonly used in commercial systems. In $\mathcal{MD}$, dimensions are linguistic categories that describe different ways of looking at the information. Each dimension is organized into a hierarchy of levels, corresponding to different granularity of data. Within a dimension, levels are related through "roll-up" functions and can have descriptions associated with them. Factual data is represented by f-tables, the logical counterpart to multi-dimensional arrays, which are functions associating measures with symbolic coordinates.

In this context, we present a general design methodology, aimed at building an $\mathcal{MD}$ scheme starting from an operational database described by an Entity-Relationship scheme. It turns out that, once facts and dimensions have been identified, an $\mathcal{MD}$ database can be derived in a natural way. We then describe two practical implementations of $\mathcal{MD}$ databases: using relational tables in the form of a "star" scheme (as in ROLAP systems), and using multidimensional arrays (as in MOLAP systems). This confirms the generality of the approach.

The paper is organized as follows. In the rest of this section, we briefly compare our work with relevant literature. In Section 2 we present the $\mathcal{MD}$ model. Section 3 describes the design methodology referring to a practical example. The implementation of an $\mathcal{MD}$ database into both relational tables and multidimensional arrays is illustrated in Section 4. Finally, in Section 5, we draw some final conclusions and sketch further research issues.

Related work. The term OLAP has been recently introduced by Codd et al. [5] to characterize the category of analytical processing over large, historical databases (data warehouses) oriented to decision making. Further discussion on OLAP, multidimensional analysis, and data warehousing can be found in [4, 8, 9, 12]. Recently, Mendelzon has published a comprehensive on-line bibliography on this subject [10].

The $\mathcal{MD}$ model illustrated in this paper extends the multidimensional model proposed in [3]. While the previous paper is mainly oriented to the introduction of a declarative query language and the investigation of its expressiveness, the present paper is focused on the design of multidimensional databases.

The traditional model used in the context of OLAP systems is based on the notion of star scheme or variants thereof (snowflake, star constellation, and so on) [8, 9]. A star scheme consists of a number of relational tables: (1) the fact tables, each of which contains a composed key together with one or more measures being tracked, and (2) the dimension tables, each of which contains a single key, corresponding to a component of the key in a fact table, and data describing a dimension at different levels of granularity. Our model is at an higher level of abstraction than this representation, since in $\mathcal{MD}$ facts and dimensions are abstract entities, described by mathematical functions. It follows that, in querying an $\mathcal{MD}$ database, there is no need to specify complex joins between fact and dimension tables, as it happens in a star scheme.

To our knowledge, the work by Golfarelli et al. [6] is the only paper that investigates the issue of the conceptual design of multidimensional databases. They propose a methodology that has some similarities with ours, even though it covers only conceptual aspects (no implementation issue is considered). Conversely, our approach relies on a formal logical model that provides a solid basis for the study of both conceptual and practical issues.

Other models for multidimensional databases have been proposed (as illustrated next) but mainly with the goal of studying OLAP query languages. A common characteristic of these models is that they are generally oriented towards a specific implementation, and so less suitable to multidimensional design than ours.

Agrawal et al. [1] have proposed a framework for studying multidimensional databases, consisting of a data model based on the notion of multidimensional cube, and an algebraic query language. This framework shares a number of characteristics and goals with ours. However, $\mathcal{MD}$ is richer than the model they propose, as it has been defined mainly for the development a general design methodology. For instance, dimensional hierarchies are part of the $\mathcal{MD}$ model, whereas, in Agrawal's approach, they are implemented using a special query language operator. Moreover, their work is mainly oriented to an SQL implementation into a relational database. Conversely, we do not make any assumption on the practical realization of the model.

Gyssens and Lakshmanan [7] have proposed a logical model for multidimensional databases, called MDD, in which the contents are clearly separated from structural aspects. This model has some characteristic in common with the star scheme even though it does not necessarily rely on a relational implementation. Differently from our approach, there are some multidimensional features that are not explicitly represented in the MDD model, like the notion of aggregation levels in a dimension. Moreover, the focus of their paper is still on the development of querying and restructuring languages rather than data modeling.

2 Modeling Multidimensional Databases

The MultiDimensional data model ($\mathcal{MD}$ for short) is based on two main constructs: dimension and f-table. *Dimensions* are syntactical categories that allow us to specify multiple "ways" to look at the information, according to natural business perspectives under which its analysis can be performed. Each dimension is organized in a hierarchy of *levels*, corresponding to data domains at different granularity. A level can have *descriptions* associated with it. Within a dimension, values of different levels are related through a family of *roll-up functions*. *F-tables* are functions from *symbolic coordinates* (defined with respect to particular combinations of levels) to *measures*: they are used to represent factual data.

Formally, we fix two disjoint countable sets of *names* and *values*, and denote by $\mathcal{L}$ a set of names called *levels*. Each level $l \in \mathcal{L}$ is associated with a countable set of values, called the *domain of* l and denoted by $\textsc{dom}(l)$. The various domains are pairwise disjoint.

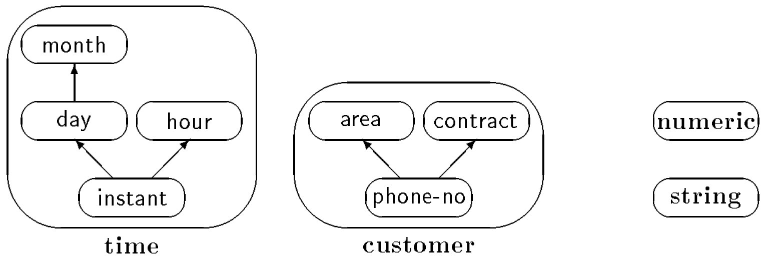
Definition 1 (Dimension). *An* $\mathcal{MD}$ dimension *consists of:*

- *a finite set of* levels $L \subseteq \mathcal{L}$*;*
- *a partial order* $\preceq$ *on the levels in* L *— whenever* $l_1 \preceq l_2$ *we say that* l_1 rolls up to l_2*;*
- *a family of* roll-up functions*, including a function* $\textsc{r-up}_{l_1}^{l_2}$ *from* $\textsc{dom}(l_1)$ *to* $\textsc{dom}(l_2)$ *for each pair of levels* $l_1 \preceq l_2$ *— whenever* $\textsc{r-up}_{l_1}^{l_2}(o_1) = o_2$ *we say that* o_1 rolls up to o_2*.*

A dimension with just one level is called *atomic*. For the sake of simplicity, we will not make any distinction between an atomic dimension and its unique level.

Definition 2 (Scheme). *An* $\mathcal{MD}$ scheme *consists of:*

- *a finite set* D *of* dimensions*;*
- *a finite set* F *of* f-table schemes *of the form* $f[A_1 : l_1\langle d_1\rangle, \ldots, A_n : l_n\langle d_n\rangle] : l_0\langle d_0\rangle$*, where* f *is a name, each* A_i $1(\leq i \leq n)$ *is a distinct name called* attribute of f*, and each* l_i $(0 \leq i \leq n)$ *is a level of the dimension* d_i*;*
- *a finite set* Δ *of* level descriptions *of the form* $\delta(l) : l'$*, where* l *and* l' *are levels and* δ *is a name called* description of l*.*



Rate [*hour* : hour, *contract* : contract, *calling-area* : area, *called-area* : area] : numeric
Duration [*calling* : phone-no, *called* : phone-no, *start* : instant] : numeric
Monthly-Bill [*customer* : phone-no, *period* : month] : numeric

Owner (phone-no) : string

Fig. 1. The sample `TelCo` scheme

Note that in an f-table we annotate the dimension corresponding to each level: this is because a level may belong to different dimensions. However, we will omit the dimensions in an f-table scheme when they are clear from the context.

Consider for instance a telecommunication company interested in the analysis of its operational information. Data about phone calls can be organized along dimensions **time** and **customer**. The corresponding hierarchies are depicted on top of Figure 1. Two further atomic dimensions are used to represent **numeric** values and **strings**. Level phone-no (telephone numbers) rolls up to both area (the geographical area in which the telephone is located, identified by an area code) and contract (characterized by rates at different hours). The domain associated with the level instant contains timestamps like *Jan 5, 97, 10AM:45:21*. This value rolls up to *10AM* in the level hour and to *Jan 5, 97* in the level day. Several f-tables can be defined in this framework, as described in the same figure. Rate represents the cost for a minute of conversation between a customer in a *calling-area* (having a contract of type *contract*) and a customer in a *called-area*, and starting at a specific *hour*. The second f-table associates with each call (issued by a *calling* to a *called* party at some time) the Duration in seconds. Monthly-Bill is a derived f-table that aggregates the revenues by phone number and month. Finally, *Owner* is a level description associating the name of a customer with a phone number.

Instances can be defined over f-tables as follows.

Definition 3 (Coordinate and Instance). *Let $\mathcal{S} = (D, F, \Delta)$ be an $\mathcal{MD}$ scheme. A* (symbolic) coordinate *over an f-table scheme $f[A_1 : l_1\langle d_1\rangle, \ldots, A_n : l_n\langle d_n\rangle] : l_0\langle d_0\rangle$ in F is a function mapping each attribute name A_i $(1 \leq i \leq n)$ to an element in $\textsc{dom}(l_i)$. An* instance *over f is a partial function that maps coordinates over f to elements of $\textsc{dom}(l_0)$. An* instance *over a level description $\delta(l) : l'$ in Δ is a partial function from $\textsc{dom}(l)$ to $\textsc{dom}(l')$.*

An *entry* of an f-table instance f is a coordinate over which f is defined. The actual value that f associates with an entry is called a *measure*.

hour	*contract*	*calling-area*	*called-area*	RATE
6AM	*Family*	*06*	*02*	0.44
7AM	*Family*	*06*	*02*	0.72
8AM	*Family*	*06*	*02*	1.12
		...		...
6AM	*Pro*	*06*	*055*	0.80
7AM	*Pro*	*06*	*055*	0.80
8AM	*Pro*	*06*	*055*	1.35
		...		...

MONTHLY-BILL	*Jan-97*	*Feb-97*	*Mar-97*
06-555-123	129	231	187
06-555-456	429	711	664
02-555-765	280	365	328

phone-no	*Owner*
06-555-123	*John*
06-555-456	*Ann*
02-555-765	*Mary*

Fig. 2. A sample instance over the `TelCo` scheme

A possible instance over the `TelCo` scheme is shown in Figure 2. A symbolic coordinate over the f-table RATE is [hour : *7AM*, contract : *Family*, calling-area : *06*, called-area : *02*]. The actual instance associates the measure 0.72 with this entry. The description *Owner* associates the string *John* with the value *06-555-123* of level phone-no. Note that two different graphical representations for f-tables are used in Figure 2: a table for RATE and an array MONTHLY-BILL. This suggests that several implementations of a same f-table are possible.

It is apparent that our notion of "symbolic coordinate" is related with that of "tuple" in the relational model. It can also be noted that the notation we use for symbolic coordinates resembles subscripting into a multi-dimensional array (although in a non-positional way). There is however an important difference between f-tables and multi-dimensional arrays. Specifically, in arrays, "physical" coordinates vary over intervals within linearly-ordered domains, whereas we do not pose any restrictive hypothesis on the domains over which coordinates range. In this sense, our notion of coordinate is "symbolic."

Roll-up functions are a distinctive feature of our model: they describe *intensionally* how values of different levels are related. Such a description is indeed independent of any effective implementation, which can be based on stored relations, built-in functions, or external procedures. Moreover, roll-up functions provide a powerful tool for querying multidimensional data, since they allow us to specify how data must be aggregated, and how f-tables involving data at different levels of granularity can be joined [3].

3 Design of MultiDimensional Databases

In this section we show how $\mathcal{MD}$ schemes can be obtained from conceptual schemes. We assume to have an E-R scheme [2] at our disposal describing an (integrated) view of operational databases. We assume that this scheme describes a "primitive" data warehouse containing all the operational information that can support our business processing, but not yet tailored to this activity. The construction of this scheme can require a number of foregoing activities, including the reverse-engineering of several data sources and their integration into a global conceptual scheme; we will not discuss these activities here, since they are beyond the scope of the paper. We make however a number of assumptions on the initial E-R scheme. First, we assume that the scheme does not contain generalization hierarchies and that all its attributes are simple (no multivalued or composed attributes). Then, we assume that the scheme is complete, in the sense that it contains all the information that can be extracted from our operational databases and that can be used in the analytical processing. Finally, we assume that the scheme is fully normalized and minimal, that is, all the concepts appear only once (no derived concepts).

The methodology we propose for building an $\mathcal{MD}$ database starting from a pre-existing E-R scheme consists of four steps.

1. Identification of facts and dimensions.
2. Restructuring of the E-R scheme.
3. Derivation of a dimensional graph.
4. Translation into the $\mathcal{MD}$ model.

Actually, the first two steps are not strictly sequential, but in many cases proceed in parallel: during the restructuring of the E-R scheme, selected facts and dimensions can be refined and modified. Then, the process proceeds sequentially, since each phase requires the completion of the previous one.

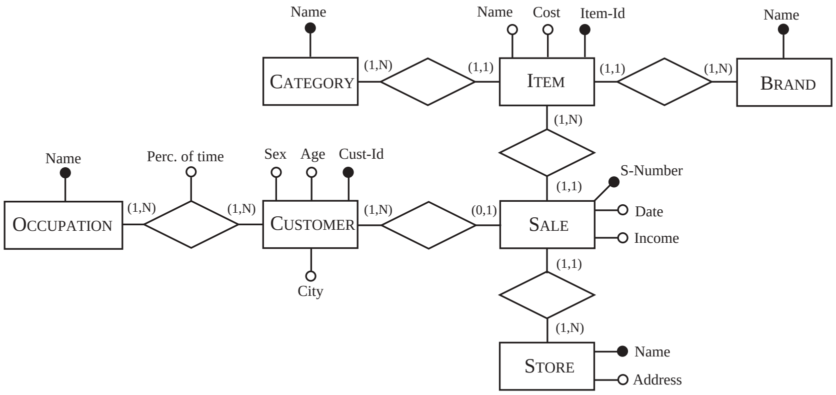


Fig. 3. An E-R scheme describing the `Retail` operational database

The methodology will be illustrated in the remaining of the section referring to the `Retail` database, whose E-R scheme is reported in Figure 3 (we adopt the notation used in the book [2]). This scheme describes a number of information about a company holding a chain of stores in which several products are sold. Some information about frequent customers of the company are available.

3.1 Identification of Facts and Dimensions

The first activity consists in a careful analysis of the given E-R scheme whose aim is the selection of the facts, the measures, and the dimensions of interest for our business processing. We call *facts* the concepts in the E-R scheme (entities, relationship, or attributes) on which the decision-making process is focused. A *measure* is instead an atomic property of a fact that we intend to analyze (generally a numeric attribute of a fact or a count of its instances). Finally, a *dimension* is a subscheme of the given E-R scheme that describes a perspective under which the analysis of a fact can be performed.

Let us consider the `Retail` database. We could be interested, on one hand, in the identification of trends in the volume of sales and in the corresponding incomes and, on the other hand, in the analysis of the variation of production costs of the items on sale. Thus, in this case the facts are the entity SALE and the attribute *Cost* of the entity ITEM. The measures for the former fact are the number of sales (count of instances of the entity) and the incomes of the sales (attribute *Income*). The only measure for the latter is the value of the cost itself. Note that, in some cases, a fact has several aspects that need to be evaluated, in others the measure of a fact coincides with the fact itself.

Along a dimension, the analysis of a fact is performed by consolidating (i.e., aggregating) data [5]. Therefore, we can identify a dimension by navigating the scheme, starting from each fact and including concepts that suggest a way to group data (for example, entities related by one-to-many relationships, or categorical attributes like age or sex). Let us consider for instance the fact entity SALE. We can see that each sale is related to the corresponding item sold and each item is related to the corresponding category and brand. It follows that sales can be examined according to the types of product sold at different levels of aggregation (single item, category of items, brand). Thus, a possible dimension for the analysis of the sales is the typology of the product sold. This dimension includes the entities ITEM, BRAND, and CATEGORY. We can also observe that, for some sale, we have information about the related customer; customers can be grouped by age, sex, city of residence (according to the corresponding categorical attributes), and occupation. Hence, a further dimension for the analysis of the sales is the typology of the customer. This dimension includes the entities CUSTOMER and OCCUPATION (and the corresponding attributes). Following similar considerations, we can conclude that the location of the sales is another possible dimension for their analysis: this dimension includes at this time just the entity STORE. Finally, we can identify a temporal dimension for the analysis of the sales (attribute *Date* of the entity SALE): this is generally a fundamental dimension in multidimensional processing.

3.2 Restructuring of the E-R Scheme

This activity consists in a reorganization of the original E-R scheme in order to describe facts and dimensions in a better, more explicit way. The goal of this step is the production of a new E-R scheme that can be directly mapped to the $\mathcal{MD}$ model. We believe that it is useful to perform this activity within the E-R model since, in this way, the mapping between the operational and the multidimensional database can be easily derived.

The restructuring can be divided into a number of activities as described in the following paragraphs.

Representing facts as entities. Generally, facts correspond to entities of the initial E-R scheme, but they can also be described by attributes or relationships. In these cases, they need to be translated into entities (according to the usual information-preserving transformations [2]) since facts become of central interest in the analytical processing. Also, this transformation simplifies the steps that follows the restructuring phase.

For instance, in our example, the production cost of the items is represented by means of an attribute. This attribute can be easily transformed into an entity Cost of Item by adding a one-to-one relationship between the new entity and the entity Item, as shown in Figure 4. Each instance of this new entity is identified (externally) by the corresponding item.

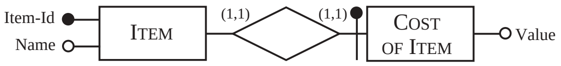


Fig. 4. A restructuring of the entity Item in the E-R scheme in Figure 3

Adding dimensions. It may happen that, for some fact in the E-R model, there are dimensions of interest for its analysis that are missing in the scheme (and therefore, according to our hypotheses, are not represented extensionally in the operational databases) but can be derived either from external databases or from meta-information associated with our data sources. For instance, we could be interested in the temporal validity of a fact or in the geographical origin of certain information. Such dimensions need to be represented explicitly in the E-R scheme.

Let us consider the cost of the items in our `Retail` database. It is reasonable that in the ordinary transaction processing we are only interested in the current cost of an item and therefore no historical information is available about it. Assume however that we know the exact time of the update operations, and that the costs change once for month on the average. Since an effective analysis of costs can be performed only if we compare them in different periods of time, we need to add temporal information about costs. According to the meta-information available, this can be done by restructuring the entity Cost of Item as described in Figure 5. From a practical point of view, this historical data can be obtained from the operational database by means of incremental updates that add, each month, a new instance to the entity Cost of Item, according to the current value of the attribute *Cost* in the original database.

Refining the levels of each dimension. Within each dimension, we need to select and represent in an explicit way the various levels of aggregation that

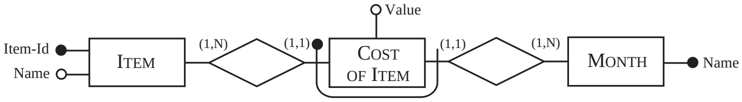


Fig. 5. A restructuring of the entity COST OF ITEM in Figure 4

are of interest in the analysis of facts (e.g., category and brand of an item) and distinguish them from the concepts that are only descriptive but cannot be used in the analysis since do not allow to perform aggregations (e.g., address and telephone number of a store). In practice, this step requires to perform one of the following transformations: replacing many-to-many relationships, adding new concepts (entities or attributes) to represent new levels of interest, selecting a simple identifier for each level entity, and removing irrelevant concepts.

Let us consider the dimension customer in our example. Within this dimension we can aggregate customers according to their age, sex, and city of residence (through the corresponding attributes of the entity CUSTOMER). If we need to aggregate customers also with respect to their occupation, we cannot use directly the corresponding entity since, according to the many-to-many relationship between CUSTOMER and OCCUPATION, each customer has in general several occupations. However, we can replace this entity by a new entity MAIN OCCUPATION describing the occupation of a customer in most of the time, so that the relationship is transformed from many-to-many into one-to-many (see Figure 6). Let us now turn our attention to the dimension location that contains just the entity STORE. We could be interested in aggregating the stores according to the city and to the geographical area (note that this information can be derived from the attribute *Address* and from "built-in" knowledge). This can be made explicit by adding new entities CITY and AREA as shown in Figure 6. For the new entities, it is important to choose a simple identifier (possibly natural if one exists). Finally, let us consider the dimension time, assuming that the dimension product does not require restructuring. We would like to aggregate sales according, for instance, to days, months, special periods (e.g., Easter, school opening, Christmas), quarters, and years. This can be done, again according to built-in knowledge, by adding new entities and one-to-many relationships as shown in Figure 6. When all the dimensions have been examined in this way, the final step consists in removing all the concepts contained in the scheme (entities, attributes, and relationships) that are useless in the analysis processing (among them, uninteresting levels of aggregation).

The E-R scheme we obtain in our example after the restructuring phase is reported in Figure 6. Note that the scheme has been annotated with facts and dimensions. Note also that a dimension does not include descriptive attributes (e.g., attribute *Address* of entity STORE).

3.3 Derivation of a Dimensional Graph

Starting from the restructured E-R scheme, we can now derive a special graph that we call *dimensional*. A dimensional graph represents, in a succinct way,

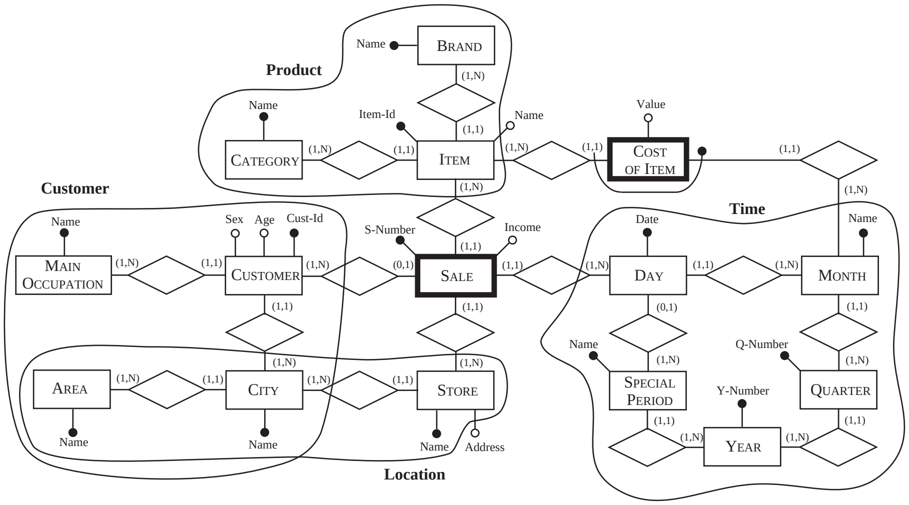


Fig. 6. A restructuring of the E-R scheme reported in Figure 3

facts and dimensions of the restructured E-R scheme. In particular, each node of the graph corresponds to a specific concept (entity or attribute) and represents a domain as follows: if the node corresponds to an entity, it represents the domain of the key of the entity; if the node corresponds to an attribute, it represents the domain of the attribute. The arc between two nodes represents a function between the corresponding domains (the arc is dashed if the function is partial). Figure 7 reports the dimensional graph obtained from the E-R scheme in Figure 6. In this graph, the node Item represents the domain of the attribute *Item-Id*; similarly, the node Month represents the domain of the attribute *Name* of the corresponding entity; instead, the node Income represents the domain of the attribute *Income* of the entity Sale. It is easy to see that this graph can be derived automatically and has the same information content as the original scheme. Note also that the dimensions become sub-graphs of the dimensional graph. In the dimensional graph we can distinguish four kinds of nodes: *fact nodes* are denoted by bold margins (they originate from fact entities); *level nodes* are those occurring in a dimension; *descriptive nodes* are the nodes outside the dimensions that have an incoming arc outgoing from a level node (they originate from descriptive attributes); and *measure nodes* are the nodes outside the dimensions that have an incoming arc outgoing from a fact node (they originate from measures).

3.4 Translation into the $\mathcal{MD}$ Model

The $\mathcal{MD}$ dimensions can be directly derived from the dimensional graph. Specifically, we have an $\mathcal{MD}$ dimension for each dimension of the dimensional graph and, for each dimension, we have an $\mathcal{MD}$ level for each node and a roll-up function for each arc of the corresponding sub-graph. The sub-graphs of the di-

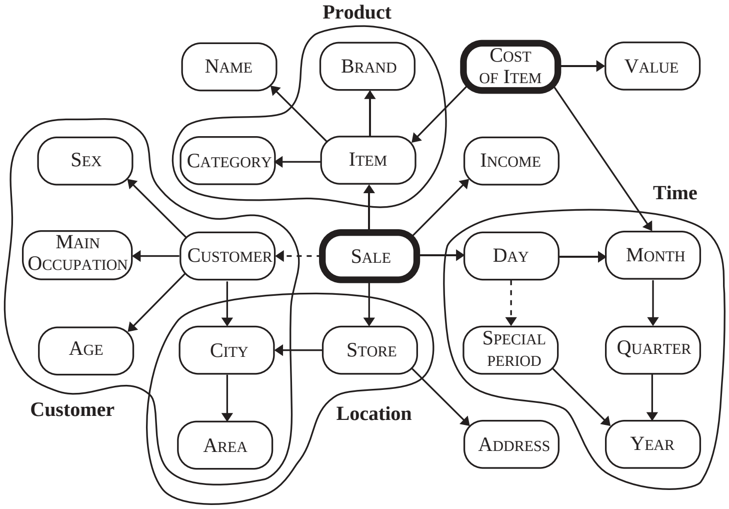


Fig. 7. The dimensional graph obtained from the scheme in Figure 6

mensional graph associated with the various dimensions denote the partial order on the $\mathcal{MD}$ levels. We need also to define a number of atomic dimensions to represent measure nodes and descriptive nodes. In our example, we can define a **numeric** dimension for sale incomes and item costs, and a dimension **string** for item names and store addresses. We can then define a $\mathcal{MD}$ level description for each descriptive node. In our example, we have a description *Name* of the level item and a description *Address* of the level store.

F-tables can be defined as follows. For each fact node in the dimensional graph, we first select a combination of levels from the "associated" dimensions, that is, the dimensions for which there is an arc from the fact node to them. More than one level can be selected for each dimension and not all the dimensions associated with a fact node must be chosen. Then, we need to define a mapping θ, possibly involving aggregations, describing the result of the f-table. This mapping can be: (1) a count of a collection of facts, or (2) an expression over a measure. The f-table instance can be built as follows: for each possible tuple t of values over the chosen levels, we have a collection Φ_t of instances of the fact (for instance, in our example, given a specific item and a day, we have a set of sales associated with them). Then, the tuple t becomes an entry of the f-table, and the measure associated with this entry is obtained by applying the mapping θ to Φ_t.

In the `Retail` database, we have already identified three measures: (1) the number of items sold, (2) the revenues, and (3) the cost of items. The first two measures are described daily for each item and store, whereas the third is given on a monthly basis. These measures can be represented by the following f-tables.

1. Sale[*period* : day, *product* : item, *location* : store] : numeric, defined over the fact Sale by the mapping **count**(Sale);
2. Revenue[*period* : day, *product* : item, *location* : store] : numeric, defined over the fact Sale by the mapping **sum**(Income(Sale));

3. CostOfItem[*period* : month, *product* : item] : numeric, defined over the fact Cost of Item by the mapping Value(Cost of Item).

We can also be interested in some partially aggregated data. For instance, the analysis of the customers' purchases, by age, category of items, and year, can be performed with the following f-table:

PurchaseByAge[*age* : age, *products* : category, *period* : year] : numeric,

which is defined over the fact Sale by the mapping **sum**(Income(Sale)).

4 Implementation of MultiDimensional Databases

In this section we show how an $\mathcal{MD}$ database can be practically implemented, using a relational database (as in ROLAP systems) or a set of multidimensional arrays (as in MOLAP systems).

4.1 Relational Databases

The natural representation of a multidimensional database in the relational model consists of a collection of "fact" and "dimension" tables. The former are normalized, whereas the latter can be denormalized. Since there exist several different definitions of star schemes, we refer in the following to a basic formulation [8, 9]. We develop the mapping for a star scheme; however, the approach can be easily adapted to variants of this model (e.g., the snowflake scheme).

A *star scheme* representing an $\mathcal{MD}$ database can be built as follows. We have: (1) a relation scheme R_d for each non-atomic dimension d, and (2) a relation scheme R_f for each f-table f. The atomic dimensions do not need to be represented since they generally correspond to basic domains.

- R_d contains an attribute A_l for each level l occurring in d, an attribute A_δ for each description δ of a level in d, and an attribute A_d denoting a (generated) key for R_d.
- R_f contains an attribute A_f for the measure of f and, for each attribute A_i of f over a level $l\langle d\rangle$, an attribute A_i whose domain coincides with the domain of the key A_d of the relation R_d.

The corresponding instances are defined as follows.

- The relation R_d contains a tuple t_v for each value v of each level l occurring in d. The tuple t_v is defined as follows: $t_v.A_d$ is a unique identifier k_v for the value v; $t_v.A_l = v$; for each description δ of l, $t_v.A_\delta = \delta(v)$; for each level l' to which l rolls up, $t_v.A_{l'} = \text{R-UP}_l^{l'}(v)$ and, for each description δ' of l', $t_v.A_{\delta'} = \delta'(\text{R-UP}_l^{l'}(v))$. The other attributes carry nulls.
- The relation R_f contains a tuple t_e for each entry e of f. If e equals $[A_1 : v_1, \ldots, A_n : v_n]$ and v_0 is the corresponding measure, then $t_e.A_f = v_0$ and, for each attribute A_i, $t_e.A_i = k_{v_i}$ $(1 \leq i \leq n)$. Note that a value v in the entry is represented by k_v (which is a key for the dimension relation identifying v), rather than by v itself.

Dimension relations:
Customer(*c-id*, *customer*, *age*, *sex*, *occup*, *city*, *area*)
Location(*l-id*, *store*, *address*, *city*, *area*)
Product(*p-id*, *item*, *category*, *brand*)
Time(*t-id*, *date*, *month*, *quarter*, *period*, *year*)

Fact relations:
Revenue(*p-id*, *t-id*, *l-id*, *revenue*)
Sale(*p-id*, *t-id*, *l-id*, *sale*)
CostOfItem(*p-id*, *t-id*, *value*)
PurchaseByAge(*c-id*, *p-id*, *t-id*, *income*)

Location

l-id	*area*	*city*	*store*	*address*
l_1	North	Venice	La Gondola	Rialto
l_2	North	Milan	Boys 'R Us	P. Cordusio
l_3	Center	Rome	Sun City	P. Navona
...	...	...	...	...
l_{100}	North	Venice	*null*	*null*
l_{101}	North	Milan	*null*	*null*
...	...	...	...	...
l_{1000}	North	*null*	*null*	*null*
...	...	...	...	...

Product

p-id	*item*	*category*	*brand*
...	...	...	...
p_{43}	Trivia	Toy	Micro
...	...	...	...

Revenue

p-id	*t-id*	*l-id*	*revenue*
...	...	...	...
p_{43}	t_{99}	l_2	15.95
...	...	...	...

CostOfItem

p-id	*t-id*	*value*
...	...	...
p_{43}	t_{504}	6.50
...	...	...

Time

t-id	*date*	*month*	*quarter*	*period*	*year*
...	...	...	...	...	...
t_{99}	Apr 1, 97	Apr97	2Q97	Easter97	1997
...	...	...	...	...	...
t_{504}	*null*	Apr97	2Q97	*null*	1997
...	...	...	...	...	...

Customer

c-id	*customer*	*age*	*sex*	*occup*	*city*	*area*
...	...	...	...	...	...	...
c_{79}	Joe	31	M	teacher	Rome	Center
...	...	...	...	...	...	...

Fig. 8. Star scheme of the `Retail` database

As an example, the star scheme representation of the the `Retail` database is shown in Figure 8. Note that the instance is just outlined. The relation Location is more detailed to show the structure of a dimension table.

The scheme so obtained can be optimized in several ways. For instance, relation schemes corresponding to different f-tables over the same levels can be merged, suitable indexes can be defined, and some views involving aggregation can be materialized [4]. The issue of optimization is however beyond the scope of this paper.

4.2 Multidimensional Arrays

We now briefly outline how an $\mathcal{MD}$ database can be represented by means of multidimensional arrays. Since there is no agreed model for MOLAP systems, we assume that factual data are represented by means of matrices whose indexes range over contiguous, initial segments of the natural numbers.

First of all, for each dimension d of our $\mathcal{MD}$ scheme we define a bijection β_d assigning a unique integer to each value of each level in d. More specifically, if m is the number of those values, β_d associates with each of them an integer varying from 1 to m (and vice versa). In this way, we obtain a one-to-one correspondence between symbolic and numeric coordinates. Then, an f-table $f[A_1 : l_1\langle d_1\rangle, \ldots, A_n : l_n\langle d_n\rangle] : l_0\langle d_0\rangle$ is represented by a n-dimensional matrix, storing each measure v_0, corresponding to the symbolic entry $[A_1 : v_1, \ldots, A_n : v_n]$, in the cell having physical coordinate $[\beta_{d_1}(v_1), \ldots, \beta_{d_n}(v_n)]$.

An $\mathcal{MD}$ dimension can be represented by means of a special data structure, with a hierarchical organization according to the partial order between the levels. This data structure is used to store both the roll-up functions and the assignment between values of levels and integers. We can then use this structure as an index to access the multidimensional arrays. The resulting scheme can be tuned by using the tools provided by the specific storage system chosen.

5 Conclusions

In this paper we have proposed a framework, based on a logical data model, for the design of multidimensional databases. Another fundamental aspect of OLAP systems is querying, a task that can be pursued using different paradigms. On one hand, the final user should be enabled to perform point-and-click operations by means of graphical metaphors. On the other hand, the sophisticated user that needs to express more complex queries should be allowed to use a declarative, high-level language. Finally, query optimization can be effectively performed by using a procedural, algebraic language, possibly referring to the underlying representation of data. We have started the study of declarative query languages for OLAP systems in [3]. We are currently investigating the other paradigms, arguing that $\mathcal{MD}$ is well-suited for the manipulation of multidimensional databases, since it allows the user to disregard implementation aspects.

References

1. R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. In *13th Int. Conf. on Data Engineering*, pages 232–243, 1997.
2. C. Batini, S. Ceri, and S. Navathe. *Conceptual Database Design.* Benjamin/Cummings, 1992.
3. L. Cabibbo and R. Torlone. Querying multidimensional databases. In *6th Int. Workshop on Database Programming Languages (DBPL'97)*, 1997.
4. S. Chaudhuri and U. Dayal. An overview of Data Warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1):65–74, March 1997.
5. E.F. Codd, S.B. Codd, and C.T. Salley. Providing OLAP (On Line Analytical Processing) to user-analysts: an IT mandate. Arbor Software White Paper, *http://www.arborsoft.com.*
6. M. Golfarelli, D. Maio, and S. Rizzi. Conceptual design of data warehouses from E/R schemes. In *31st Hawaii Intl. Conf. on System Sciences*, 1998.
7. M. Gyssens and L.V.S. Lakshmanan. A foundation for multi-dimensional databases. In *33rd Int. Conf. on Very Large Data Bases*, pages 106–115, 1997.
8. W.H. Inmon. *Building the Data Warehouse.* John Wiley & Sons, 2nd ed., 1996.
9. R. Kimball. *The Data Warehouse Toolkit.* John Wiley & Sons, 1996.
10. A. O. Mendelzon. Data warehousing and OLAP: a research-oriented bibliography. *http://www.cs.toronto.edu/∼mendel/dwbib.html.*
11. N. Pendse and R. Creeth. The OLAP Report. *http://www.olapreport.com.*
12. A. Shoshani. OLAP and statistical databases: 6imilarities and differences. In *16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems*, pages 185–196, 1997.