

Conceptual Design of DVR from Graph Model

Paisarn Trakulsuk, Vichit Avatchanakorn

Graduate school of Computer Information Systems

Assumption University

Ramkhamhaeng 24, Huamark,

Bangkok, Thailand 10240

Tel: (662) 300-4543, 300-4553

Fax: (662) 713-1521

E-mail: paisarn.t@thaiairways.co.th - E-mail: vichit@au.ac.th

Abstract: The Multidimensional cube for OLAP (On-line Analytical Processing) is composed of dimensions, data value or Variable, and relative dimensions (DVR), the basic components of a Multidimensional Data Model (MDD). The model shows how to provide data support intelligence through multidimensional data analysis when there are huge amounts of online data. This paper presents a conceptual design of data analysis from a data warehouse into DVR. The design is a graph model of data catalogs that generate the DVR forms of a multidimensional cube. The graph structure's center node represents the fastest varying dimension, several branch nodes representing associate dimension attributes, and rectangles representing indexing of data values and measurements. These may also be represented on fact schemes that integrate relative information and external sources, as well as extends the algorithms to build DVR patterns in standard SQL form. This model shows how workload is reduced and expresses the input-output design phase of MDD.

Keywords: Data Modeling, Graph Model, Data Warehouse, OLAP.

1. Introduction

Data warehouses collect, integrate, and clean a wide variety of external data sources and base data into a single database. A data warehouse is used primarily for decision support system (DSS) and management information system (MIS). Thus, it may contain many data entities to speed up query processing for different resulting data arrays [2]. Metadata, an important aspect of a data warehouse, collects various definitions of data elements in warehouse databases. Metadata collates conceptual or logical data structures of both objects and entities, together with all the relationships of the data elements within the database [6]. In this paper, metadata will support a DVR model to generate its own multidimensional cube.

A DVR model is a pattern of configuring large masses of data into patterns meaningful for management analysis and decision-making. It is composed of dimension, data values or variables, and relative dimension. A *dimension* is a group of attributes that is chosen by management to arrange the data (How many suppliers of shrimp is needed during the high flying season compared to during the low flying season?). *Relative dimensions* are the attributes in each dimension (Who are the suppliers? What size, color, of shrimp is needed at the galley?) *Data values* or variables refer to units of quantity or value

measurements of the data. (How many kinds of dishes use shrimp? How many percent of passengers do not choose shrimp? How much less passengers fly during the off season?)

Previous research shows that a warehouse database leans on a multidimensional data model to generate data analysis. In this paper, the function is extended to initiating the data warehouse routine of transforming the data into a *DVR* model according to user specifications.

The accuracy of conceptual design is vital in building an information system so that it responds fully to user requirements. This paper presents a graph model that shows a routine for processing voluminous data from various sources into groups of related dimensions and associated data values. The focus of this paper is 1) to create a DVR of MDD to meet user requirements more accurately, 2) to design a routine that automatically creates a database format, 3) to speed up the MDD's input-output design phase, and 4) to minimize human error in the process. Using the graph theory this paper creates a DVR model based on relational format concepts of data processing.

Currently, the relational database is the standard operating format all over the world for high volume data processing. This format requires intensive man-hours to shape the mass of data into forms meaningful to management. At Thai Airways, an average of 2,016 man-hours per year is consumed for this routine, too slow to meet the demands of high-speed decision-making. This paper proposes MDD, an automated process of DVR creation that identifies, corrects, and screens out errors in a large mass of data. With MDD, the procedure takes only an hour, and never again. When factored against the number of man-hours spent in all companies all over the world, this represents incredible savings of time, effort, and money.

The graph model describes how data entities are collected from a database through three symbols: center nodes, branch nodes, and rectangle nodes. A primary attribute is presented in a center-node. Foreign*keys or character format attributes, and associated dimensions are presented in branch-nodes. Numeric format attributes are data values or variables presented in rectangle-node.

Curved lines show DVR relationships. Dashes show links between dimensions that support the OLAP drill-down concept. In complex data the lines overlap to rotate and compare information. The finished B-tree structure fact scheme is reorganized into DVRs of a multidimensional cube in terms of a standard SQL form supported by algorithms.

2. Background and Literature Review

Data warehousing has three phases that start from raw data and ends in knowledge extracted from the database. First, data is segmented according to some criteria. Second, data is cleaned, then transformed. Data is then stored in data warehouse with distinct structures.

When data is stored in relational forms at the data access layer, it is voluminous because of storage at the lowest level of granularity. It is necessary to produce summarized data in MDD in order to access the data volumes quickly. The two types of data summaries in MDD are *lightly summarized data*, which is distilled from low levels, and *highly-summarized data*, which is compact and easily accessible.

To rebuild the lowest detailed data into summarized data, data accuracy must be declared to prevent error contamination. There are several techniques to acquire and integrate information from heterogeneous sources and to query very large database efficiently. The E/R approach is presented as a graph model for data warehousing called Dimension Fact Model. This proposes a semi-automated routine based on pre-existing Entity/Relationship schemes [9]. The E/R model is used widespread as a conceptual formalism to provide standard documentation for relational information systems. Many efforts have been done to use the E/R scheme as input for designing non-relational database.

A decision tree (name, date) is a method of decision-making that is applicable when complex branching occurs in a structured decision process. A decision tree is particularly useful when it is essential to keep a string of decisions in a particular sequence. A decision tree allows description of conditions and actions on branch nodes. Thus, it was applied in this research to negotiate the complex routes of data warehouse structuring.

Most research about data warehousing focuses on control of multi-dimensional views within data warehouse environments such as maintenance techniques, view technique designs, and multiple views consistency. To reduce maintenance costs, Nam Huyn proposed a process of preserving the views and keeping them consistent without updating the warehouse [1]. The process introduces an algorithm to generate SQL queries, the answers of which determine if a view can be maintained in a given situation. The answers also generate SQL updates that maintain the view.

The process presents a framework of view design that achieves the best combination of good query performance and low view maintenance [3]. A framework is presented to highlight issues of materialized view design in a distributed data warehouse environment. For multiple views consistency (MVC), Yue Zhuge, et al, presents three layers of consistency for materialized views in a distributed environment [5]. This consistency develops simple and scalable algorithms for achieving MVC in a warehouse database.

These techniques emphasize the consistency and maintenance of multi-dimensional view in warehouse databases. These research papers imply the importance of maintenance and avoiding inconsistency. In this paper, the significance of generating physical multi-dimensional database is underscored as a significant factor in designing data models and data volumes. This paper suggests one way of building a DVR model using a B-tree diagram to produce a graphic structure. The focus of this research is not on handling and maintaining multi-dimensional view processes but on the creation of the databases' conceptual design.

Recent research in the area of conceptual design of data warehouse introduces the E/R (Entity/Relationship) diagrams conceptual model, on which the graph model of this research is based. Matteo Golfarelli, et al., used the E/R model to build a Dimensional Fact (DF) model. He also represents query patterns on a fact scheme made up of a set of markers placed on the dimension attributes [9]. The DF model was used as a basis to change the graph model into DVR.

Aric Shoshani introduces a statistical database (SDB) schema using a graph representation of a bi-dimensional table that represents attributes as nodes, and represents levels of hierarchical classification as branching lines [4]. This formed the basis of creating the symbolic representation of the multidimensional cube of this research.

3. Conceptual Design of DVR model

The DVR model was created in two steps. First, the structure of the relational format was changed into the graph model structure. Then, the data elements of the relational format were changed into dimensional fact elements of nodes and rectangles to represent various data dimensions. As a result, the bidimensional relational format became a multidimensional model, referred to in this paper as a Cube Model. The Graph Model is a two-dimensional representation of the multidimensional theory of the Cube Model.

The connecting links between the various dimensions and associated data values comprise the multidimensionality of the Cube Model. On paper, the Graph Model represents this multidimensionality with basic DVR components and relationships between the various groupings. A DVR form represents *dimension* as logical grouping of attributes with a common atomic key relationship. A *variable* represents facts or measures. A *relative dimension* represents datum associated with dimensional attributes. The relationships between the dimension attributes and their associated data values are called *grouping relation set*.

A data mass needs to pass major steps to become meaningful data: first is transformation to graph model, then automatic generation to DVR multidimensional format.

The processing of mass data from relational format to multidimensional format using automatic processing consists of 2 steps:

1. Generate automatically Graph model from relational format.
 - 1.1 Define the candidate dimensions and variables.
 - 1.2 Build the DVR in form of graph model.
 - 1.2.1 Identify the actual dimension, variable, and relative dimension.
 - 1.2.2 Define the main dimension and sub dimensions.
 - 1.2.3 Draw the graph model.
 - 1.3 Link the related graph models.
2. Create automatically MDD.

In this paper, the warehouse database (denoted by DB_i) consists of *PRODUCT* and *SALESMAN* data entities. D denotes the candidate dimension, V denotes the candidate variable, and R denotes the relative dimension. Each attribute a_i of *PRODUCT* a data entity is defined to be a candidate dimension, written as $D_i^{PRODUCT}$. Similarly, each attribute a_i of *PRODUCT* data entity is defined to be candidate variable, written as $V_i^{PRODUCT}$, where i is the attribute of *PRODUCT* data entity. The information of *PRODUCT* and *SALESMAN* data entities is illustrated in Tables 1.1 to 1.3.

Table 1.1 - Data Entity of Product Information

1. Product	Char (8)	Primary key	Unique
2. Category	Char (2)	-	-
3. Size	Char (10)	-	-
4. Salesman Code	Char (5)	Foreign key	-
5. Manufacturer Code	Char (5)	Foreign key	-
6. Customer Code	Char (5)	Foreign key	-
7. Shipment Code	Char (2)	Foreign key	-
8. Order Date	DMY	-	-
9. Order Qty	Num (6, 0)	-	-
10. Price	Num (10, 2)	-	-

Table 1.2 - Data Entity of Salesman Information

1. Salesman Code	Char(5)	Primary Key	Unique
2. Area	Char(5)	-	-
3. Function	Char(8)	-	-
4. Commission Rate	Num(4,2)	-	-

Table 1.3 – Sample data elements of table 1.1

Product	Category	Size	Salesman	Manufacturer	Customer	Shipment	Order Date MM/YY	Order Qty	Price
A001	Audio	Medium	SE001	20M54	C3215	B1	10/1998	4	\$185.00
A002	Audio	Large	SE001	20M54	C1201	B2	11/1998	2	\$290.00
A002	Audio	Large	SE002	20M54	C3150	B2	11/1998	1	\$290.00
T029	TV	Inch29	NR014	21S01	C2145	D1	10/1998	1	\$900.00
T021	TV	Inch21	NR014	21S01	C2319	D1	11/1998	3	\$550.00
T021	TV	Inch21	SE001	21S01	C1201	B1	11/1998	4	\$550.00
T014	TV	Inch14	NR014	21S01	C3215	D1	12/1998	6	\$199.00
T014	TV	Inch14	SE002	21S01	C3150	B2	11/1998	5	\$199.00
T014	TV	Inch14	WS023	21S01	C1321	C1	10/1998	4	\$199.00
V034	VDO	Mono	WS021	21S01	C3124	C1	11/1998	1	\$210.00

3.1 Defining Candidate Dimension and Variable

3.1.1 Selection of Candidate Dimensions

In a great mass of data elements, one cannot identify whether an attribute is a dimension or a variable; the selection is automatic, not human. The system does the selection using what is called a propositional function. The result needs to be verified if it is really a dimension, so it is in the meantime called a *candidate dimension*. To verify, an algorithm is used to compare attribute by attribute, until the candidate dimension is verified as a real dimension. A real dimension is called *actual dimension*.

Each group of data elements includes so many dimensions that it is necessary to identify the actual dimensions, those that are relevant to the needs of the management. A *dimension* is the logical grouping of attributes with a common atomic key relationship [6]. This is a subject-oriented grouping that includes product, location and time.

How is a dimension identified exactly? First, when an attribute in a data entity is established as part of a *DVR* model, the attribute is classified as 'candidate dimension'. The propositional function is a formal way for representing knowledge in terms of declarative sentences [8]. The propositional function specifies that any attribute x of a data format is defined as a candidate dimension if the attribute is an index key field or its data is expressed either in character or date format, thus: $(\forall x)(I_x \cup (\delta_{cx} \cup \delta_{tx})) \Rightarrow D_x^i$ where, I_x is the index key format, normally stored as key field, δ_x is the type of data property which is equivalent to character format (denoted by "c"), or date/time format (denoted by "t"), and D_x^i is the attribute x within dimension of data entity i , when $i > 0$.

Second, any candidate dimension is reclassified as 'actual dimension' using a comparative algorithm as described in the next section (Section 3.2: *Building the B-tree Model*).

3.1.2 Selection of Candidate Variables

Each group of data includes so many variables, that it is necessary to identify the variables that are relevant to the needs of the management. A *variable* is a fact or measure that is generally stored as a numeric field [6]. According to the attributes remaining after the candidate dimensions are indicated, the propositional function of defining a candidate variable is that any attribute x is defined as a candidate variable if its data type is in a numeric (binary, integer, or decimal) format: $(\forall x)(\delta_{nx}) \Rightarrow V_x^i$ where, δ_n is data type in numeric format (denoted by " n "), and V_x^i is attribute x within the variable of data entity i , where $i > 0$.

The term *relative dimension* refers to the description of the associated data of a dimension. It defines the relationships of an ordinary attribute to a grouping relation. The first step of defining the relative dimension is to convert it into a candidate dimension. The next step is to repeatedly predicate the characteristic of actual dimension and relative dimension. (See Section 3.2)

Example 1: Define candidate dimension and variable of data entity PRODUCT.

Table 1.1 shows that the attribute PRODUCT is the primary key field of the data entity. This is thus assigned as 'candidate dimension.' Other attributes (SALESMAN, MANUFACTURER, CUSTOMER, and SHIPMENT) present the foreign key fields of the data entity. These attributes are thus assigned as candidate dimensions corresponding to the PRODUCT attribute. The attributes of CATEGORY, SIZE, and ORDER DATE are assigned as candidate dimension because they are in character or date formats. The attributes of ORDER QUANTITY and PRICE are classified as candidate variables because they are in numeric format. The result is:

$$D_{product}^{PRODUCT}, D_{salesman}^{PRODUCT}, D_{manufacturer}^{PRODUCT}, D_{customer}^{PRODUCT}, D_{shipment}^{PRODUCT}, \\ D_{category}^{PRODUCT}, D_{size}^{PRODUCT}, V_{orderquantity}^{PRODUCT}, V_{price}^{PRODUCT}$$

3.2 Building the B-tree of DVR model

The previous chapter discusses the automated process of identifying candidate dimensions and variables within a mass of high-volume data. This chapter presents the process of changing candidate dimensions and variables into actual dimensions and variables.

The *DVR* is a finite set of grouping relations, a combination of the attributes of dimension, variable, and/or relative dimension. Grouping relation generally contains dimension as base attributes carrying useful information. The variables attribute and relative dimension are related to any dimension attribute called 'associated attributes' and are derived from the base attributes. The candidate dimension, which is the primary index of a data entity, is assigned as 'main dimension,' the starting point of the Graph Model.

The starting point of a tree structure is the main dimension, which is represented by a major circle. From this, other candidate dimension attributes (sub-dimensions) follow in the form of smaller circles. For instance, elements of Table 1.1 are shown in Figure 1 with lines that delineate the relationships between the main dimension PRODUCTS and the sub-dimension attributes. There are no relationships between the sub-attributes.

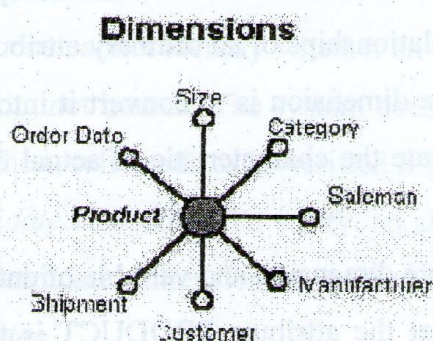


Figure 1: Table 1.1 Elements as Graph Model Form

This example shows the advantages of using multidimensional over bidimensional data analysis. The column form of Table 1.1 and the Graph Model on Figure 1 contain the same data. However, the multidimensional graph model of Figure 1 shows the many relationships between elements, which are not clear on the bidimensional Table 1.1.

The greater number of relationships between data attributes yields a more complete, more consistent body of information for decision-making. With the multidimensional model of data presented in the XYZ form of Graph Theory, management is able to integrate multiple data for more effective decision-making. As complex organizations grow, more information is processed in more and more complex interrelationships that grow exponentially. Existing data processing routines are becoming inadequate increasingly growing complexity.

3.2.1 Identifying actual dimension, variable, and relative dimension

After attributes of data entities have been automatically categorized as 'candidate dimension' and 'candidate variable' in Section 3.1.2, they must be compared together within the same data. This repeated predication is necessary entity in order to verify whether the candidate dimensions and variables are actual dimensions, variables, and relative dimensions.

The comparative process follows:

1. Choose one data entity from the warehouse database.
2. From the data entity, choose one candidate dimension as primary index key.
3. Use this primary index key to start a data map. Map the primary index key with the other attributes in the data entity.
4. After mapping, use a comparative algorithm to screen and remove all incomplete, missing, unrelated, and duplicate data.
5. Program the algorithm to identify each attribute in each data set as either 'actual dimension,' 'variable,' or 'relative dimension.' (Or use commercial software such as Personal Express Tool™, Metacube™, etc.)
6. Execute the program.

Data is now ready for analysis for relationships among the attributes of one data entity. The Graph Model now incorporates the element of relationship between the primary and other attributes: the small circles are white to indicate sub-dimensions while black circles indicate relative dimensions (See Section 5.2).

Comparative Algorithm:

At each data mapping:

- Initially, set VL_1 , VL_2 to empty.
- Count data values of first attribute (a_1) of data mapping into VL_1 .
- Do while data mapping is not empty
 - If second attribute (a_2) of data mapping is candidate dimension.
 - Performed SQL process of second attribute
 - Count data values into VL_2
 - If VL_1 equal to VL_2
 - Set a_2 to be relative dimension.
 - Else ---- Set a_2 to be actual dimension.
 - Else -- Performed SQL process of second attribute
 - If VL_1 equal to VL_2
 - Set a_2 to be relative dimension.
 - Else --- Set a_2 to be actual variable
- VL_2 is set to empty.

End Do

End Algorithm

Example 2: Identifying actual dimension, variable, and relative dimension of *PRODUCT* data entity

1. Extract the file labeled *PRODUCT* data entity from the warehouse database.
2. From the file's many attributes, select $D_{product}$ as the primary index key.
3. Compare $D_{product}$ with the other attributes.

4. Data mapping is accomplished as: $\{D_{product} \cup D_{salesman}\}$, $\{D_{product} \cup D_{manufacturer}\}$, $\{D_{product} \cup D_{customer}\}$, $\{D_{product} \cup D_{shipment}\}$, $\{D_{product} \cup D_{category}\}$, $\{D_{product} \cup D_{size}\}$, $\{D_{product} \cup V_{orderquantity}\}$, $\{D_{product} \cup V_{price}\}$
5. Execute comparative algorithm.

The program executes a routine that counts the item number of $D_{product}$ attribute by aggregate function (e.g. *sum*, *avg*, *min*, *count*) into variable VL_1 . The routine is executed until the data mapping set is empty. At the first data mapping set, the comparative algorithm represents the difference between $D_{product}$ and $D_{salesman}$ as candidate dimension. The $D_{salesman}$ attribute is counted by aggregate function of the SQL process and recorded into variable VL_2 .

6. Identify actual and relative elements.

The program executes a routine that compares the variables VL_1 and VL_2 . If the results show absolute dissimilarity, then the $D_{salesman}$ attribute is assigned as actual dimension. If the results show similarity, then, the attribute is assigned as relative dimension.

The program executes the routine to process the candidate dimensions of MANUFACTURER, CUSTOMER, and SHIPMENT with PRODUCT attribute and assigned as actual dimensions. If the Step 6 routine shows a similarity of relationships, the candidate dimensions are labeled as relative dimensions. If dissimilar, they are labeled as actual dimension. When candidate variables of PRICE are similar to PRODUCT, PRICE is then assigned as a relative dimension. The candidate variable of ORDER QUANTITY is automatically assigned as actual variable.

3.2.2 Defining Main and Sub-dimensions

Figure 1 shows the relationship between main and sub dimensions, both indicated as actual dimensions. An actual dimension, a primary index key, is assigned as main dimension. The other actual dimensions are assigned as sub-dimensions.

For instance, a $D_{product}$ attribute is a primary index key of the data entity labeled as PRODUCT. The process assigns PRODUCT attribute as main dimension or finest aggregation of the tree structure, represented by a large circle. On the other hand, SALESMAN, MANUFACTURER, CUSTOMER, SHIPMENT, and ORDER DATE attributes are assigned as sub-dimensions represented by tiny circles that are attached to the major circle. At this step, the routine may be programmed as follows:

```

DEFINE product DIMENSION TEXT WIDTH 8
DEFINE salesman DIMENSION TEXT WIDTH 5
DEFINE manufacturer DIMENSION TEXT WIDTH 5
DEFINE customer DIMENSION TEXT WIDTH 5
DEFINE shipment DIMENSION TEXT WIDTH 2
DEFINE order_date DIMENSION TEXT WIDTH 10

```

This is an example of creating dimension attributes in multidimensional database via Personal Express Tools™. Small, black circles connected solely to the major circle represent relative dimension attributes: SIZE, PRICE, and CATEGORY attributes are linked directly to the primary key, PRODUCT. The program may be compiled as follows:

```

DEFINE size VARIABLE TEXT <product>
DEFINE price VARIABLE DECIMAL <product>
DEFINE category VARIABLE TEXT <product>

```

3.2.3 Drawing the Graph Model

DVR models are drawn as graphs using symbols such circle nodes, boxes that are connected with lines, dashes, and arrows. The main dimension is shown as a center-node in the Graph Model by a large circle. Sub-dimensions are shown as branch nodes, represented by small circles of two types: Relative dimensions are solid black and connected solely to the center node, actual dimensions are white and may be connected to either the center node or to branch-nodes. Actual variables are boxes labeled with attribute names in numeric format and linked solely with the center node. Measurements are in boxes linked to the actual variables solely. They are separated from the actual variable box because they are only formulas while actual variable boxes store physical data values. An example is in the following Graph Model 2:

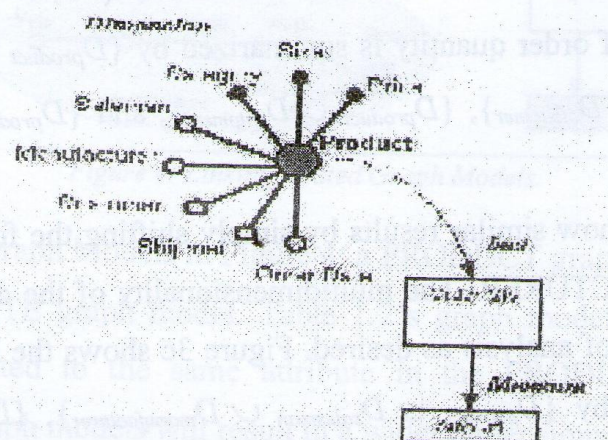


Figure 2: Graph model of PRODUCT data entity

Assume the measurement of the attribute AMOUNT is included in this graph model, as a result of the multiplication of the attributes ORDER QUANTITY and PRICE. The model shows a box with the attribute name linked by a related actual variable. This is an example of a graph model utilizing only bidimensional (2D) relationships. For example, the

SALESMAN branch node and the PRODUCT center node represents only a row-and-column relationship associated with a tabulated form of quantity and price value.

However, a Graph Model can be fully utilized to conduct a multidimensional view simply by connecting lines into the desired nodes. The model below illustrates the adaptivity of a 2D graph model into 3D graph model.

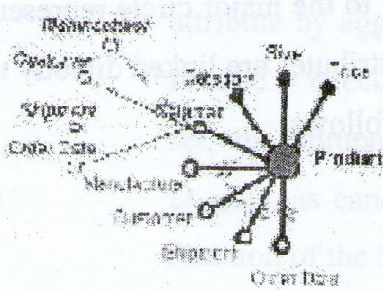


Fig. 3a: Shifting connecting lines to analyze data in relation to SALESMAN dimension.

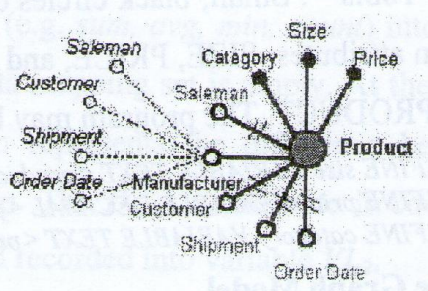


Fig. 3b: Shifting connecting lines to analyze data in relation to MANUFACTURER dimension

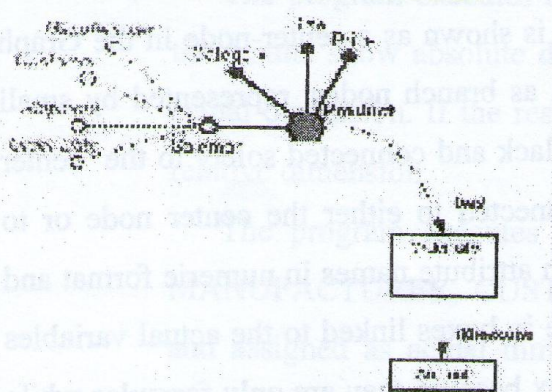


Fig.3c : 3-D graph model

Figure 3: Adapting a 2D Analysis into a 3D dimension.

Figure 2 uses the Graph Model on the bidimensional (2D) level, similar to tabular representation. The result of order quantity is summarized by $\{D_{product} \cup D_{salesman}\}$, $\{D_{product} \cup D_{manufacturer}\}$, $\{D_{product} \cup D_{customer}\}$, $\{D_{product} \cup D_{shipment}\}$, and $\{D_{product} \cup D_{orderdate}\}$. (See Fig. 3a and 3b.)

Figures 3a and 3b show similar results by simply shifting the focus of analysis from SALESMAN to MANUFACTURER. The multidimensionality of the data analysis is in the flexibility of shifting focus of analysis as desired. Figure 3c shows the summary of ORDER QUANTITY as processed by $\{D_{product} \cup D_{salesman} \cup D_{manufacturer}\}$, $\{D_{product} \cup D_{salesman} \cup D_{customer}\}$, $\{D_{product} \cup D_{salesman} \cup D_{shipment}\}$, and $\{D_{product} \cup D_{salesman} \cup D_{orderdate}\}$. The results of processing AMOUNT and the ORDER QUANTITY are the same.

To transform a graph model into a DVR using Personal Express Tool™ the routine is:

* Variable Process:

```
DEFINE orderqty_01 VARIABLE <product salesman manufacturer>
LD Orderqty of product-salesman-manufacturer dimension
```

```

DEFINE orderqty_02 VARIABLE <product salesman customer>
LD Orderqty of product-salesman-customer dimension
DEFINE orderqty_03 VARIABLE <product salesman shipment>
LD Orderqty of product-salesman-shipment dimension
DEFINE orderqty_04 VARIABLE <product salesman orderdate>
LD Orderqty of product-salesman-orderdate dimension

```

* Measurement Process:

```

DEFINE amount_01 FORMULA orderqty_01*Price
LD amount of product-salesman-manufacturer dimension
DEFINE amount_02 FORMULA orderqty_02*Price
LD amount of product-salesman-customer dimension
DEFINE amount_03 FORMULA orderqty_03*Price
LD amount of product-salesman-shipment dimension
DEFINE amount_04 FORMULA orderqty_04*Price
LD amount of product-salesman-orderdate dimension

```

The creation of a DVR based on the Graph Model as shown in Figure 3 shows that a three-dimensional graph model is insufficient to analyze the problem in all its aspects. In such instances, the model is easily reorganized into as many dimensions as desired.

3.3 Linking Related Graph Models Together

When dimension attributes of various data entities are related, Graph Models may be linked using a dotted line to connect the related dimension attributes. An example follows.

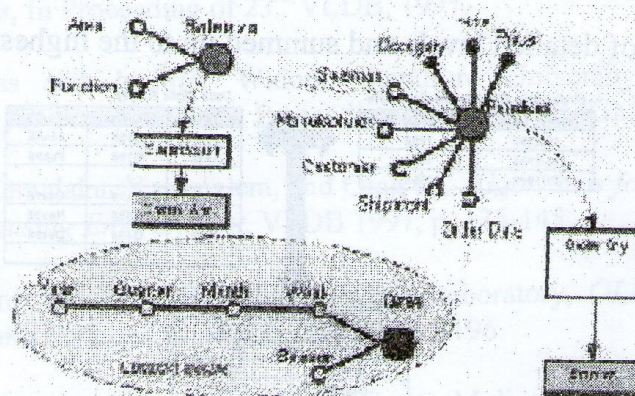


Figure 4: Linking related Graph Models.

There are three Graph Models in Figure 4: a PRODUCT graph model, a SALESMAN graph model, and a DATE graph model. Table 1.2's graph model with the base attribute SALESMAN is associated to the same attribute in the PRODUCT graph model. The connection of the two graph models will result in a wider dimensional analysis.

For example, management may want to know how much commission came out of the purchases of customer X. The analysis is completed when two graph models are combined

as: $\{D_{product} \cup D_{salesman} \cup D_{customer} \cup D_{orderdate} \Rightarrow V_{commission}\}$.

3.3.1 Additional Aids

Illustrating multidimensional views in other ways, (e.g. Roll-Ups, Drill-Downs) can be useful techniques to present levels of hierarchical data. These techniques are helpful because they create attributes for each level on the hierarchy. For instance, an ORDER DATE attribute may be defined into levels of WEEK, MONTH, QUARTER, and YEAR. The attribute AREA may be divided into geographical levels such as: CONTINENT, COUNTRY, REGION, CITY, and so on. Users can roll-up and drill down through multiple dimensions concurrently.

The MDD model can easily incorporate such additional procedures. Existing databases can easily be processed using these add-ons. The MDD model can be reprogrammed to automatically process these new dimensions. This not only widens the range of analysis, it also allows local programmers to utilize routines developed elsewhere to enrich their processing functions.

For example, the year 1998 of ORDER DATE attribute is connected to year 1998 in DATE data entity. The QUARTER sublevel of year 1998 is divided into Q198, Q298, Q398, and Q498. The MONTH sublevel of Q198 is divided into JAN98, FEB98, and MAR98, all embedded as aggregation of detailed levels and summed up to the highest level as follows:

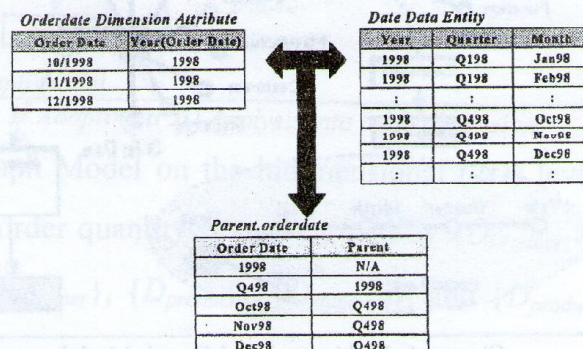


Figure 5: Joining of Order Date Dimension with Date Data Entity.

The routine may be programmed thus:

```
DEFINE parent.orderdate RELATION orderdate <orderdate>
CALL execution_1
ROLLUP orderqty_04 OVER orderdate USING parent.orderdate
ROLLUP amount_04 OVER orderdate USING parent.orderdate
```

The system will automatically aggregate values of actual data to the higher level, as follows:

Shipment	1998	Q498	Oct98	Nov98	Dec98
B1	\$2940.00	\$2940.00	\$740.00	\$2200.00	0.00
B2	\$1865.00	\$1865.00	0.00	\$1865.00	0.00
C1	\$1006.00	\$1006.00	\$796.00	\$210.00	0.00
D1	\$3744.00	\$3744.00	\$900.00	\$1650.00	\$1194.00

4. Conclusion

This paper introduces the DVR model, a more efficient alternative to the relational database, the standard operating format all over the world for high volume data processing. The model was created with this four-point focus: 1) to create a DVR that meets user requirements more accurately, 2) to design a routine that automatically creates an MDD database format, 3) to speed up the input-output design phase, and 4) to minimize human error.

The DVR structure identifies data attributes as candidate dimension or/and candidate variables using the propositional function. The attributes are then compared by comparative algorithm for relationship between attributes. The outcome is automatically generated as MDD by linking similar dimension attributes to build the multidimensional views. Data loading is expressed as query patterns. Validation and rebuilding is by user interaction. Roll-up and drill down features of *OLAP* can be incorporated as self-activating routines to increase efficiency.

5. References

- [1] Nam Huyn, Stanford University, *Multiple-View Self-Maintenance in Data Warehousing Environments*, In Proceeding of 23rd VLDB, 1997.
- [2] Dallen Quass and Jennifer Widom, Standford University, *On-Line Warehouse View Maintenance for Batch Updates*, Proceeding of the ACM SIGMOD Conference, 1997.
- [3] Jian Yang, Kamalakara Karlapalem, and Qing Li, *Algorithms for Materialized View Design in Data Warehousing Environment*, VLDB 1997, p. 136-145.
- [4] Aric Shoshani, Lawrence Berkeley National Laboratory, *OLAP and Statistical Databases: Similarities and Differences*, PODS 1997, p.185-196.
- [5] Yue Zhuge, Janet L. Wiener and Hector Garcia-Molina, Standford University, *Multiple View Consistency for Data Warehousing*, In Proceeding of the International Conference on Data Engineering, 1997.
- [6] Harry S. Singh, *Data Warehousing Concepts, Technologies, Implementations, and Management*, Prentice Hall PTR, 1998.
- [7] E. F. Codd and S.B. Codd, *OLAP with TM/1* E. F. Codd & Associates, *On-Line Analytical Processing (OLAP) White Paper*, 1994.
- [8] George J. Klir, Ute H. St. Clair and Bo Yuan, *Fuzzy Set Theory: Foundations and Applications*, Prentice-Hall International, Inc., 1997
- [9] Matteo Golfarelli, Dario Maio, Stefano Rizzi, *Conceptual Design of Data Warehouses from E/R Scheme*, Published in the Proceeding of the Hawaii International Conference On System Sciences, January 6-9, 1998.