Building Logical DVR Model of a Multidimensional Database

Paisarn Trakulsuk  
Assumption University  
Ramkhamhaeng 24, Huamark,  
Bangkok, Thailand 10240  
E-mail: paisarn.t@thaiairways.co.th

Vichit Avatchanakorn  
Assumption University  
Ramkhamhaeng 24, Huamark,  
Bangkok, Thailand 10240  
E-mail: vichit@au.ac.th

ABSTRACT

Warehouse Database is a multidimensional model generally implemented in relational databases that contains a data repository. This data repository integrates valuable information from multiple data sources. This paper presents a procedure to generate the Dimension, Variable, and Relative dimension (DVR) of multidimensional cubes used in the Multidimensional On-Line Analytical Processing (MOLAP) concept.

This proposed procedure uses the data mining approach to build an MDD database model. The procedure consists of classification, partitioning, and clustering modules. Classification modules verify an attribute of data entity as a candidate dimension or a variable of multidimensional cubes. Partitioning modules use candidate dimensions as bases for mapping with other attributes of the same data entity to form mapping data sets. This module uses the Internal Comparison Process (ICP) to manipulate data sets to validate the DVR model. Finally, the clustering module uses External Comparison Process (ECP) to manipulate all DVR models in the partitioning modules in order to group the related dimensions and associated data values.

1. Introduction

Database programs are one of the most essential aspects of computational sciences. It is applied in practically all fields, from commerce, education, the arts, to the sciences. The applications used to manipulate databases have been developed in various ways since 1946, when the first mainframe, the ENIAC, was created. One of the many developments in database is the Multidimensional Database (MDDB).

The MDDB supports a wide variety of analytical processes, all called OLAP (On-Line Analytical Processing). In short, OLAP is a set of tools for retrieving, organizing, and analyzing data. On the other hand, Data Warehouse is a general storage area of data from various sources. OLAP and Data Warehouse are complementary because the first presents an interface for better visualizing data stored in the data warehouse: Data Warehouse stores and manages data, while OLAP transforms the mass of data into useful and meaningful information. A review of related literature shows that current research is mainly focused on maintenance procedures, design of viewing procedures, and on the consistency of multiple views within data warehouse environments. There is a lack of material on how to create MDDB, as well as how to minimize errors in creating MDDBs.

In a recent study on maintenance procedures, Nam Huyn discussed ways to reduce maintenance costs by keeping views consistent without updating the warehouse yet preserving the views [1]. He introduces an algorithm to generate SQL queries resulting in answers that determine if a view can be maintained in a given situation and then generates an SQL update that maintains the view. In a study on view design, Yang, Karlapalem, and Li
present a framework that can be used to achieve a combination of good query performance and low view maintenance [4]. The study also introduces important issues on materialized view design in a distributed data warehouse environment. In a study on multiple-view consistency (MVC), Zhuge et al presents three layers of consistency for materialized views in a distributed environment [9]. This consistency develops simple and scalable algorithms for achieving MVC in a warehouse database.

The above examples show that the current emphasis is on maintenance of multidimensional views in warehouse databases. Although these are important factors in generating databases and in designing data models and data volumes, creating such databases should also be considered. This paper presents how to create multidimensional databases through automated process that identifies, corrects, or screens out errors in a large mass of data.

The errors referred to are of four types: 1) missing information, 2) incomplete information, 3) unrelated information, and 4) duplicated information. Missing information refers to data that is not present, such as an index key that requires data but contains no data. Incomplete information refers to data that is not complete, such as an index key that has only four out of five required data entries. Unrelated information refers to data that is not related to another; for instance, PRODUCTNAME is related to PRODUCTCODE. However, PRODUCTCODE is not related to SUPPLIERNAME. Duplicated information refers to data that is unnecessarily repeated in a database. For instance, two sets of data with exactly the same attributes. For instance, under the data set TYPE may be the attributes BLACK and WHITE. When the data set COLOR also uses the attributes BLACK and WHITE, this is an example of duplication. These types of data may cause an error in analysis, and therefore have to be identified, corrected, or screened out from the database.

How does this proposed system minimize the occurrence of errors in a warehouse database? The proposed program automatically generates or creates a DVR (Dimension, Variable, and Relative dimension) model that helps to minimize errors by 1) analyzing relationships between data sets, 2) identifying attributes in data sets for missing or unrelated data, 3) reducing unrelated or duplicated attributes into one.

At present, the process of transforming relationship patterns in a database from a bidimensional to a multidimensional pattern for multidimensional or DVR analysis is done via manual interface. The growing complexity of corporate databases and the need for faster analysis multidimensionally requires the process to be automated. What is needed today is a system that allows intelligent and meaningful analysis of many relationships between a great number of data sets. This requirement is met by the DVR model, which allows multidimensional relationships within the storage system. This paper presents a method of automating the process of generating DVR patterns of relationships in a database. This process yields faster analysis of more complex patterns as well as screens data that are not relevant to the analysis.

A DVR model consists of a finite set of grouping relationships. It is a combination of dimension attributes, variable attributes, and/or relative dimension attributes for decision-support requirements. The DVR model is essentially used to create a multidimensional cube that has the efficiency required for supporting OLAP.

A first priority before handling and maintaining the process of multidimensional views is the creation of a DVR model of multidimensional cubes. The lack of a DVR model will limit analysis to two-dimensional associations. On the other hand, an incomplete DVR model will cause error contamination to occur in analytical procedures. Generating an MDDB at the initial stage is the best way to preclude these problems.
Operational Database

Cleansing and Transforming Process

Warehouse Database

Manual Processing of MDDB by analyst

Automatic DVR generation

MDDB

MDDB

MDDB

PARADIGM OF THE RESEARCH

Figure 1: Research Paradigm

Figure 1 above shows the paradigm of the research. In practice, a warehouse database is distilled from various operational databases by transforming or screening data at the data access layer using various tools or programs. Current data in warehouse database is usually stored in relational format. The data in this warehouse is voluminous because storage is at a low level of granularity. The system requires manual processing to analyze the database for errors. This paper proposes an automatic process of generating DVR, which summarizes data into MDDBs for quick access. A major advantage of this proposed system is the automation of the process of creating a DVR. A minor advantage of this proposed system is that it does away with manual processing for errors.

This paper develops three essential modules to generate a DVR model and discusses how to treat the data for error contamination. In the classification module, an attribute of data entity is verified as either a candidate dimension or a variable of multidimensional cubes. In the partitioning module, candidate dimensions are used as bases for mapping with other attributes of the same data entity to form mapping data sets. The data sets are then manipulated using the Internal Comparison Process (ICP) to validate the DVR model. Finally, all DVR models from the partitioning module are manipulated using the External Comparison Process (ECP) in order to group the related dimensions and the associated data values into a clustering module.

These three modules are applied together with data mining procedures in order to build an effective and logical DVR model of MDDB. In order to demonstrate the effectiveness of the proposed procedures, this paper presents tables of sales and products information as examples.

2. Generating Logical DVR Models

Figure 2 illustrates the step-by-step paradigm for generating a DVR model from a bidimensional form. The figure shows that Data Entities in the warehouse database are rearranged to form the DVR model representing dimension, variables as facts or measures, and relative dimension, followed by the three suggested modules.
1. **The Classification Module.** This module inspects if all data types are an indexed key, and verifies the candidate dimension of multidimensional cubes. On the other hand, if the attribute is verified as numeric, the data is classified as “variables”.

2. **The Partitioning Module.** In this module, the candidate dimensions from the classification module are used as bases for mapping with the other attributes of the same data entity to form mapping data sets. The data sets are manipulated using the Internal Comparison Process (ICP) in which each attribute in the data set is assigned as “actual dimension”, “variable” or “relative dimension.” The module produces a logical data model, the DVR model of an individual data entity.

3. **The Clustering Module.** The clustering module uses data sets of particular entities as bases for mapping with other data sets of various data entities. The mapped data sets are then processed using the External Comparison Process (ECP) in order to generate a new data set that shows the relationship between the base data and the rest of the data sets. The module generates a logical data model of multidimensional cubes via a union function of DVR models from various data entities.

The resulting DVR model will be analyzed for suitability to user needs and satisfaction in order to guide the consequent refining of the process. Inaccuracies in the original data as well as corrections in the procedure may be done manually at the two interface nodes (see Figure 2). Finally, the rebuilt logical DVR model should serve to guide the process of generating new physical DVR models.

3. **Classification Module**

At the initial stage, the classification module indicates the candidate dimensions or variables of any attribute in the data entity. The attribute is compared with the definite rule of the DVR model in order to establish the DVR indication. However, this module is still currently limited to classifying candidate dimension and variable only. For simplicity of notation, we denote the candidate dimension and candidate variable corresponding to an attribute \( x \) at data entity or data entity \( i \) in the paper as \( D_x^i \) and \( V_x^i \), respectively.

3.1 **Selection of Candidate Dimension**

Dimension is the logical grouping of attributes with a common atomic key relationship [10]. The grouping is subject-oriented: product, location and time. When an attribute in data entity is established for creating a DVR model, it’s data type will be classified as “candidate dimension” and recorded in the data definition. Any candidate dimension will be reclassified
as “actual dimension” in the partitioning module. The propositional function, a formal way for representing knowledge in terms of declarative sentences [13], specifies this concept thus: For any attribute \( x \) of data entity, if attribute \( x \) is an index key field or its data is either in character or in date format, attribute \( x \) then is defined as “candidate dimension”. Thus,

\[
(\forall x)(I_x \cup (\delta_{c,x} \cup \delta_{t,x})) \Rightarrow D'_i
\]

Where, \( I_x \) is the index key format, normally stored as key field, \( \delta_{c,x} \) is the type of data property which is equivalent to character format (denoted by “c”), or date/time format (denoted by “t”), and \( D'_i \) is the attribute \( x \) within dimension of data entity \( i \), when \( i > 0 \).

3.2 Selection of Candidate Variable

In the opposition, a variable is fact or a measure that is normally stored as a numeric field. This has been the focus of a decision-support investigation [10]. According to the remaining attributes after the candidate dimensions are indicated, the propositional function of defining candidate variable is this:

For any attribute \( x \), if its data type is in a numeric (binary, integer, or decimal) format, then define attribute \( x \) as “candidate variable,” thus:

\[
(\forall x)(\delta_{n,x}) \Rightarrow V'_i
\]

Where, \( \delta_{n,x} \) is data type equals to numeric format (denoted by “n”), and \( V'_i \) is attribute \( x \) within the variable of data entity \( i \), when \( i > 0 \).

The term relative dimension or associated data description of a dimension is simply a relationship with an ordinary attribute of a grouping relation. Initially, it is so complicated to be classified by the argument that the alternative is to convert it into “candidate dimension”. The last step of the partitioning module will repeatedly predicate the characteristic of actual dimension and relative dimension.

Example 1: Consider the data entity \( SALE \) (shown in relational table) within a warehouse database.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Data Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCODE (Saleman code)</td>
<td>CHAR</td>
<td>5</td>
</tr>
<tr>
<td>SNAME (Saleman Name)</td>
<td>CHAR</td>
<td>30</td>
</tr>
<tr>
<td>ACODE (Area code)</td>
<td>CHAR</td>
<td>3</td>
</tr>
<tr>
<td>SALEVAL (Sale value)</td>
<td>NUM</td>
<td>10.2</td>
</tr>
<tr>
<td>COMM (Commission)</td>
<td>NUM</td>
<td>10.2</td>
</tr>
</tbody>
</table>

When the fundamental characteristics of the attributes \( SCODE, SNAME, ACODE \) are presented in the index-key field and/or character formats of data entity, these characteristics are defined as “candidate dimension”. Since the data types of attributes \( SALEVAL \) and \( COMM \) are in the numeric format, these characteristics are classified as “candidate variable”.

4. Partitioning Module

In the classification module, all attributes of data entities are categorized as “candidate dimension” and “candidate variable”. These attributes are relevant to the grouping of associated attributes in the partitioning module.

There are two main processes in the partitioning module: 1) the mapping process, and 2) the internal comparison process (ICP). The mapping process assigns the candidate
dimensions as base attributes and maps them with the other attributes of the same data entity to form mapping data sets. The mapping data sets are then manipulated with the ICP to achieve the actual dimension, variable, and relative dimension. The following paragraphs illustrate the entire process of mapping data sets.

Consider a warehouse database denoted by $DB_i$. Assume the data entities in $DB_i$ are $A, B, C$: $(DB_i = A, B, C)$; Suppose $A = \{a_1, a_2, a_3, a_4\}$, $B = \{a_1, a_5, a_6\}$ and $C = \{a_5, a_7\}$ where $a$ is an attribute. Assume $a_1$ and $a_2$ in data entity $A$ are candidate dimensions, while $a_3$ and $a_4$ are candidate variables. Attributes $a_1$ and $a_2$ are mapped together with other attributes in data entity $A$. Let $\overline{A}$ be the mapped data sets of $A$.

This concept is presented thus:

$$\overline{A} = \{(a_1 \circ a_2),(a_1 \circ a_3),(a_1 \circ a_4),(a_2 \circ a_3),(a_2 \circ a_4)\}$$

To prove the mapping of data set $\overline{A}$, base attributes and consequence attributes are substituted into a mapping table as shown below.

<table>
<thead>
<tr>
<th>Base</th>
<th>Attributes of $A$</th>
<th>Mapping Data Set</th>
<th>Base</th>
<th>Attributes of $A$</th>
<th>Mapping Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>-</td>
<td>$a_1 \circ a_2$</td>
<td>$a_4$</td>
<td>-</td>
<td>$a_4 \circ a_3$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>-</td>
<td>$a_2 \circ a_3$</td>
<td>$a_5$</td>
<td>-</td>
<td>$a_5 \circ a_4$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>-</td>
<td>$a_3 \circ a_4$</td>
<td>$a_6$</td>
<td>-</td>
<td>$a_6 \circ a_3$</td>
</tr>
<tr>
<td>$a_4$</td>
<td>-</td>
<td>$a_4 \circ a_3$</td>
<td>$a_3$</td>
<td>-</td>
<td>$a_3 \circ a_4$</td>
</tr>
</tbody>
</table>

Table 1: Mapping table of partitioning module

The mapping data sets are then manipulated using ICP in which each data set is validated to form the DVR model. ICP consists of the following processes:

**PROCESS 1**: According to the commutativity of classical set operations, when the data sets are symmetric and represent similar attributes, a data set is invited to form a DVR model.

**PROCESS 2**: The base attribute of a data set must become a candidate dimension, otherwise the data set is disregarded.

**PROCESS 3**: After completing the execution of process 1, if the attributes in the data set are candidate dimensions, check each of the characters by using the SQL process to count the value of each attribute and compare them. If the values of two attributes are similar, then go to process 4, otherwise, split each attribute into independent dimensions.

**PROCESS 4**: Since the data set of process 3 may result in two attributes, find which one is the actual dimension and which one is the relative dimension. Look at the data set to find which one is present in the index key and classify that as “actual dimension”. Classify the other one as “relative dimension”. Replace the join function symbol ($\circ$) with the related function symbol ($\Rightarrow$).

**PROCESS 5**: The use of process 4 will result in a data set that presents the actual and relative dimensions. Other data sets with the same attributes as the relative dimensions will be deleted.

**PROCESS 6**: When the consequent attributes are indicated as candidate variables, the mapping data set will be superseded and the join function ($\circ$) will be replaced with the associated function ($\Rightarrow$).
PROCESS 7: To prevent duplication of data sets, whenever the homogeneous base attributes indicate a dimension and consequent attributes are indicated as variables in any data set, only one completed data set is retained. The rest are deleted. ■

PROCESS 8: Attributes represented as dimensions will be replaced by \( D_x \), where \( x \) is the dimension number. The sign \( V_x \) will replace the attributes that are represented as variable, where \( x \) is the variable number. Finally, the sign \( R_x \) will replace the attributes that are represented as relative dimension, where \( x \) is the relative dimension number. ■

The following example illustrates the ICP steps of building a logical DVR model:

**Example 2:** Assume data entity 1 of warehouse database is SALE data entity that consists of attribute SCODE (Saleman Code), SNAME (Saleman Name), ACODE (Area Code), SALEVAL (Sale Value), and COMM (Commission Value), as presented in the following table:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Data Type</th>
<th>Length</th>
<th>DVR Forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. SCODE (Index Key)</td>
<td>Char</td>
<td>5</td>
<td>D</td>
</tr>
<tr>
<td>2. SNAME</td>
<td>Char</td>
<td>30</td>
<td>D</td>
</tr>
<tr>
<td>3. ACODE (Foreign Key)</td>
<td>Char</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>4. SALEVAL</td>
<td>Num</td>
<td>10.2</td>
<td>V</td>
</tr>
<tr>
<td>5. COMM</td>
<td>Num</td>
<td>10.2</td>
<td>V</td>
</tr>
</tbody>
</table>

First, let attribute SCODE be \( a_1 \), attribute SNAME be \( a_2 \), attribute ACODE be \( a_3 \), attribute SALEVAL be \( a_4 \), and attribute COMM be \( a_5 \). Each candidate dimension (eg. \( a_1 \), \( a_2 \), \( a_3 \)) is used as base attributes and performs mapping with other attributes of SALE data entity.

The result of mapping data sets are:

\[ A = A_1 \cup A_2 \cup A_3 \]

First, let the sample data elements are:

<table>
<thead>
<tr>
<th>Rec.</th>
<th>SCODE</th>
<th>SNAME</th>
<th>ACODE</th>
<th>SALEVAL</th>
<th>COMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10200</td>
<td>John Smith</td>
<td>EAST</td>
<td>$200,000.00</td>
<td>$1,000.00</td>
</tr>
<tr>
<td>2</td>
<td>10200</td>
<td>John Smith</td>
<td>WEST</td>
<td>$150,000.00</td>
<td>$750.00</td>
</tr>
<tr>
<td>3</td>
<td>10200</td>
<td>John Smith</td>
<td>NORTH</td>
<td>$750,000.00</td>
<td>$3,750.00</td>
</tr>
<tr>
<td>4</td>
<td>10210</td>
<td>Caroline Jone</td>
<td>SOUTH</td>
<td>$59,000.00</td>
<td>$295.00</td>
</tr>
<tr>
<td>5</td>
<td>10210</td>
<td>Caroline Jone</td>
<td>WEST</td>
<td>$850,000.00</td>
<td>$3,250.00</td>
</tr>
<tr>
<td>6</td>
<td>10240</td>
<td>Tom Hogan</td>
<td>WEST</td>
<td>$410,000.00</td>
<td>$2,050.00</td>
</tr>
</tbody>
</table>

The result of \( A \) is prepared for access in ICP in order to validate the form of the DVR model. Considering process 1, when there is a duplicated data set, only one will be selected, the rest are deleted. When data sets (\( a_1 \approx a_2 \)) and (\( a_2 \approx a_3 \)) are compared and found similar, only one data set is selected, the rest are deleted. Following that process, the results are thus:

\[ A = \{ (a_1, A_1), (a_1, A_2), (a_1, A_3), \ldots, (a_1, A_n), \ldots \} \]
The data sets of $\overline{A}$ are performed to access data as in process 3. Consider the example of data set $(a_1 \Rightarrow a_2)$, both verified as candidate dimensions in the classification module. The comparison between attributes $a_1$ and $a_2$ in data set $(a_1 \Rightarrow a_2)$ are performed by using this query process:

```sql
SELECT count(*) FROM SALE
GROUP BY SCODE
ORDER BY SCODE
```

The value of $\Gamma_{\text{count}}^\delta(D\text{IST}(a_1)) = 3$ and $\Gamma_{\text{count}}^\delta(D\text{IST}(a_2)) = 3$, shows that the attributes $a_1$ and $a_2$ are related. Therefore, this data set would access the next algorithm. If the solution of attributes $a_1$ and $a_3$ of data set $(a_1 \Rightarrow a_3)$ is not similar, the process will automatically represent them as dimensions: attribute $a_1$ as $D_1(D_1, \text{SCODE})$ and attribute $a_3$ as $D_2(D_2, \text{ACODE})$.

Referring to the access of data set $(a_1 \Rightarrow a_2)$, when the data type of attribute $a_1$ is an index-key field, the process will automatically represent attribute $a_1$ as a dimension ($D_1, \text{SCODE}$) and attribute $a_2$ as a relative dimension ($R_1, \text{SNAME}$). The data set will be filled in for new data set $(a_1 \Leftrightarrow a_2)$. Referring to process 2, when attribute $a_2$ is represented as relation, then the mapping of data sets in which attribute $a_2$ is used as base will be cancelled.

Considering process 5 and process 6, the existing data sets that are composed of relative dimensions (attribute $a_2$) will be erased. An example is data sets $(a_1 \Rightarrow a_3), (a_1 \Rightarrow a_4), (a_1 \Rightarrow a_5), (a_1 \Rightarrow a_6), (a_1 \Rightarrow a_7), (a_1 \Rightarrow a_8), (a_1 \Rightarrow a_9)$, $(a_2 \Rightarrow a_7), (a_2 \Rightarrow a_8), (a_2 \Rightarrow a_9)$. When the remaining data sets that are composed of dimensions and variables are replaced by the symbol ($\Leftrightarrow$), the results are:

$\overline{A} = \{(a_1 \Leftrightarrow a_7, a_3, a_5, a_6, a_8), (a_1 \Rightarrow a_7, a_3, a_5, a_6, a_8), (a_1 \Rightarrow a_7, a_3, a_5, a_6, a_8)\}$

The different data values of any dimension in mapping data sets will be integrated into unique data sets (referring to process 7). For example, data sets $(a_1 \Rightarrow a_4)$ and $(a_1 \Rightarrow a_5)$ are integrated as $(a_1 \Rightarrow a_4, a_5)$. This will result thus:

$\overline{A} = \{(a_1 \Leftrightarrow a_4, a_5), (a_1 \Rightarrow a_4, a_5), (a_1 \Rightarrow a_4, a_5)\}$

As attributes $a_1$ and $a_3$ are used as dimensions in any steps of ICP, they are represented as actual dimensions. The correspondence of the SQL statement of the data set $(a_1 \Rightarrow a_4, a_5)$ is presented as:

```sql
SELECT distinct SCODE, sum(SALEVAL), sum(COMM)
FROM SALE
GROUP BY SCODE
```

Where attribute $a_1$ will be replaced with $\text{SCODE}$, attribute $a_4$ will be replaced with $\text{SALEVAL}$, and attribute $a_5$ will be replaced with $\text{COMM}$, the elements of dimension $D_1, \text{SCODE}$ and associated data values $V_1, \text{SALEVAL}, V_2, \text{COMM}$ are presented as:

<table>
<thead>
<tr>
<th>Rec.</th>
<th>SCODE</th>
<th>SALEVAL</th>
<th>COMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10200</td>
<td>$1,100,000.00</td>
<td>$5,500.00</td>
</tr>
<tr>
<td>2</td>
<td>10210</td>
<td>$709,000.00</td>
<td>$3,545.00</td>
</tr>
<tr>
<td>3</td>
<td>10240</td>
<td>$410,000.00</td>
<td>$2,050.00</td>
</tr>
</tbody>
</table>

The results of the data sets will be replaced with the sign of a DVR model from process 8 that is represented as: $D_1, \text{SCODE}, R_1, \text{SNAME}, D_2, \text{ACODE}, V_1, \text{COMM}$. The following are DVR models of this example:
The above samples are data values actually generated from DVR models. However, data analysts may reform or rebuild logical DVR models manually at any of the two interface nodes (see Figure 2).

5. Clustering Module

While the partitioning module provides a powerful platform to form logical DVR models of data entity, a warehouse database however, contains various beneficial data entities and data-sharing capabilities from external sources. Thus, the relationships among relative dimensions of numerous entities are required to link with associated attributes.

As in partitioning modules, there are two particular steps in the clustering module process: First, the arrangement of grouping relation set, a finite set of dimension attributes and associated variable attributes, is performed by mapping process. Second, the use of the external comparison process (ECP) aggregates the grouping relation sets of any entities together. The grouping relation set is presented in the form of a data set.

The data sets are rearranged in series to verify the relationships in the mapping table that is composed of base and related data sets. One of them is assigned as base data set and is joined to related data sets to discover relationships. Whenever the comparison process indicates the existence of a relationship between base and related data sets, a new data set will be generated (The process will be mentioned later). If the comparison process of a particular base data set is accomplished, that particular data set will be removed from the series in order to stop repetitive comparison. Another data set will be sequentially assigned as the next base data set and the process will resume until there is no base data set left or until the related data set is empty.

For example, let a warehouse database be denoted as DB, with three data entities: A, B, and C, assigned into DVR models in a partitioning module. Data entity A contains the attribute $\{a_1, a_2, a_3, a_4\}$ and the data sets are $(a_1 \Rightarrow a_3, a_4)$. Data Entity B contains the attribute $\{a_1, a_5, a_6\}$ and the data sets are $(a_1 \Rightarrow a_6), (a_5 \Rightarrow a_6)$. Data entity C contains the attribute $\{a_5, a_7\}$ and data set is $(a_5 \Rightarrow a_7)$. Consequently, the data sets of DB, may be assigned as: $(a_1 \Rightarrow a_3, a_4), (a_2 \Rightarrow a_3, a_4), (a_1 \Rightarrow a_6), (a_5 \Rightarrow a_6), (a_5 \Rightarrow a_7)$. Let data set $(a_1 \Rightarrow a_3, a_4)$ be firstly assigned as base data set and the others as related data sets. The execution of ECP from $(a_1 \Rightarrow a_3, a_4)$ as base data set yields:

$(a_1 \Rightarrow a_3) \cup (a_2 \Rightarrow a_3, a_4)$
$(a_1 \Rightarrow a_3, a_4) \cup (a_1 \Rightarrow a_6)$
$(a_1 \Rightarrow a_3, a_4) \cup (a_5 \Rightarrow a_6)$
$(a_1 \Rightarrow a_3, a_4) \cup (a_5 \Rightarrow a_7)$
Similarly, the utilization of ECP for \((a_2 \Rightarrow a_5, a_4), (a_1 \Rightarrow a_6), (a_5 \Rightarrow a_7)\) as base data sets results in following table:

<table>
<thead>
<tr>
<th>Base Data Set</th>
<th>Related Data Set</th>
<th>Mapping Data Set</th>
<th>ECP Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>((a_1 \Rightarrow a_5, a_4))</td>
<td>((a_1 \Rightarrow a_5, a_4))</td>
<td>((a_1 \Rightarrow a_5, a_4)) (\cup) ((a_2 \Rightarrow a_5, a_4))</td>
<td>-</td>
</tr>
<tr>
<td>((a_1 \Rightarrow a_6))</td>
<td>((a_1 \Rightarrow a_6))</td>
<td>((a_1 \Rightarrow a_6)) (\cup) ((a_2 \Rightarrow a_6))</td>
<td>-</td>
</tr>
<tr>
<td>((a_5 \Rightarrow a_7))</td>
<td>((a_5 \Rightarrow a_7))</td>
<td>((a_5 \Rightarrow a_7)) (\cup) ((a_2 \Rightarrow a_7))</td>
<td>-</td>
</tr>
<tr>
<td>((a_5 \Rightarrow a_6))</td>
<td>((a_5 \Rightarrow a_6))</td>
<td>((a_5 \Rightarrow a_6)) (\cup) ((a_2 \Rightarrow a_6))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_7))</td>
<td>((a_2 \Rightarrow a_7))</td>
<td>((a_2 \Rightarrow a_7)) (\cup) ((a_1 \Rightarrow a_7))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_6))</td>
<td>((a_2 \Rightarrow a_6))</td>
<td>((a_2 \Rightarrow a_6)) (\cup) ((a_1 \Rightarrow a_6))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_5))</td>
<td>((a_2 \Rightarrow a_5))</td>
<td>((a_2 \Rightarrow a_5)) (\cup) ((a_1 \Rightarrow a_5))</td>
<td>-</td>
</tr>
<tr>
<td>((a_1 \Rightarrow a_7))</td>
<td>((a_1 \Rightarrow a_7))</td>
<td>((a_1 \Rightarrow a_7)) (\cup) ((a_2 \Rightarrow a_7))</td>
<td>-</td>
</tr>
<tr>
<td>((a_1 \Rightarrow a_6))</td>
<td>((a_1 \Rightarrow a_6))</td>
<td>((a_1 \Rightarrow a_6)) (\cup) ((a_2 \Rightarrow a_6))</td>
<td>-</td>
</tr>
<tr>
<td>((a_1 \Rightarrow a_5))</td>
<td>((a_1 \Rightarrow a_5))</td>
<td>((a_1 \Rightarrow a_5)) (\cup) ((a_2 \Rightarrow a_5))</td>
<td>-</td>
</tr>
<tr>
<td>((a_1 \Rightarrow a_4))</td>
<td>((a_1 \Rightarrow a_4))</td>
<td>((a_1 \Rightarrow a_4)) (\cup) ((a_2 \Rightarrow a_4))</td>
<td>-</td>
</tr>
<tr>
<td>((a_1 \Rightarrow a_3))</td>
<td>((a_1 \Rightarrow a_3))</td>
<td>((a_1 \Rightarrow a_3)) (\cup) ((a_2 \Rightarrow a_3))</td>
<td>-</td>
</tr>
<tr>
<td>((a_1 \Rightarrow a_2))</td>
<td>((a_1 \Rightarrow a_2))</td>
<td>((a_1 \Rightarrow a_2)) (\cup) ((a_2 \Rightarrow a_2))</td>
<td>-</td>
</tr>
<tr>
<td>((a_1 \Rightarrow a_1))</td>
<td>((a_1 \Rightarrow a_1))</td>
<td>((a_1 \Rightarrow a_1)) (\cup) ((a_2 \Rightarrow a_1))</td>
<td>-</td>
</tr>
<tr>
<td>((a_1 \Rightarrow a_0))</td>
<td>((a_1 \Rightarrow a_0))</td>
<td>((a_1 \Rightarrow a_0)) (\cup) ((a_2 \Rightarrow a_0))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_0))</td>
<td>((a_2 \Rightarrow a_0))</td>
<td>((a_2 \Rightarrow a_0)) (\cup) ((a_1 \Rightarrow a_0))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_9))</td>
<td>((a_2 \Rightarrow a_9))</td>
<td>((a_2 \Rightarrow a_9)) (\cup) ((a_1 \Rightarrow a_9))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_8))</td>
<td>((a_2 \Rightarrow a_8))</td>
<td>((a_2 \Rightarrow a_8)) (\cup) ((a_1 \Rightarrow a_8))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_7))</td>
<td>((a_2 \Rightarrow a_7))</td>
<td>((a_2 \Rightarrow a_7)) (\cup) ((a_1 \Rightarrow a_7))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_6))</td>
<td>((a_2 \Rightarrow a_6))</td>
<td>((a_2 \Rightarrow a_6)) (\cup) ((a_1 \Rightarrow a_6))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_5))</td>
<td>((a_2 \Rightarrow a_5))</td>
<td>((a_2 \Rightarrow a_5)) (\cup) ((a_1 \Rightarrow a_5))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_4))</td>
<td>((a_2 \Rightarrow a_4))</td>
<td>((a_2 \Rightarrow a_4)) (\cup) ((a_1 \Rightarrow a_4))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_3))</td>
<td>((a_2 \Rightarrow a_3))</td>
<td>((a_2 \Rightarrow a_3)) (\cup) ((a_1 \Rightarrow a_3))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_2))</td>
<td>((a_2 \Rightarrow a_2))</td>
<td>((a_2 \Rightarrow a_2)) (\cup) ((a_1 \Rightarrow a_2))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_1))</td>
<td>((a_2 \Rightarrow a_1))</td>
<td>((a_2 \Rightarrow a_1)) (\cup) ((a_1 \Rightarrow a_1))</td>
<td>-</td>
</tr>
<tr>
<td>((a_2 \Rightarrow a_0))</td>
<td>((a_2 \Rightarrow a_0))</td>
<td>((a_2 \Rightarrow a_0)) (\cup) ((a_1 \Rightarrow a_0))</td>
<td>-</td>
</tr>
</tbody>
</table>

Example 3: Referring to example 2, when the data sets are already assigned as DVR models in the partitioning module, the actual dimensions of the SALEMAN code and AREA code are examined to discover associations between them.

The actual variables of COMMISSION value and SALE value are investigated if they can be integrated when actual dimensions are related. The attributes of data entity A are placed in the form of a DVR model, thus:

```
-- D_1, SCODE => V_1, SALEVAL, V_2, COMM
-- D_1, SCODE => R, SNAME
```
Additional data sets and their sample data elements from data entities \( B, C \) are assumed to completely form the DVR models, thus:

\[
\begin{align*}
\text{-- } & D_1^2.ACODE \Rightarrow V_1^2.SALEVAL, V_1^2.COMM \\
\text{-- } & D_1^2.PCODE \Rightarrow V_1^2.SALEVAL, V_1^2.COSTVAL \\
\end{align*}
\]

\[
\begin{array}{|c|c|c|c|}
\hline
\text{C CODE} & \text{SALEVAL} & \text{PCODE} & \text{COSTVAL} \\
\hline
410100 & \$1,150,000 & A2100 & \$760,000 \\
410200 & \$359,000 & A2200 & \$809,000 \\
410300 & \$710,000 & A2500 & \$650,000 \\
\hline
\end{array}
\]

\[
\begin{align*}
\text{-- } & D_1^2.PCODE \Rightarrow V_1^2.SALEVAL \\
\end{align*}
\]

\[
\begin{array}{|c|c|c|}
\hline
\text{S CODE} & \text{PCODE} & \text{SALEVAL} \\
\hline
10200 & A2100 & \$359,000 \\
10200 & A2200 & \$750,000 \\
10210 & A2200 & \$359,000 \\
10210 & A2500 & \$650,000 \\
10240 & A2100 & \$410,000 \\
\hline
\end{array}
\]

The following table presents the arrangement of DVR models in sequence, which utilized ECP to generate new data sets, thus:

\[
\begin{array}{|c|c|c|}
\hline
\text{Base Data Set} & \text{Related Data Set} & \text{(ECP) Execution Result} \\
\hline
D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_1^1.COMM & D_1^1.ACODE \Rightarrow V_1^1.SALEVAL, V_1^2.COMM & \rightarrow D_1^2.SCODE = D_1^2.ACODE \Rightarrow V_1^1.SALEVAL, V_1^2.COMM \rightarrow \rightarrow \rightarrow \rightarrow \\
\hline
D_1^1.ACODE \Rightarrow V_1^1.SALEVAL, V_1^1.COMM & D_1^1.ACODE \Rightarrow V_1^1.SALEVAL, V_1^2.COMM & \\
\hline
D_1^2.SCODE \Rightarrow D_1^2.PCODE \Rightarrow V_1^2.SALEVAL, V_1^2.COSTVAL & D_1^2.SCODE = D_1^2.PCODE \Rightarrow V_1^2.SALEVAL, V_1^2.COMM & \\
\hline
\end{array}
\]

The data set \( (D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_1^1.COMM) \) is firstly set up to be a base data set, and then compared with the existing grouping relation sets, thus:

\[
\begin{align*}
(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_1^1.COMM) & \cup (D_1^1.ACODE \Rightarrow V_1^1.SALEVAL, V_1^2.COMM) \\
(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_1^1.COMM) & \cup (D_1^1.ACODE \Rightarrow V_1^1.SALEVAL, V_1^2.COMM) \\
(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_1^1.COMM) & \cup (D_1^1.ACODE \Rightarrow V_1^1.SALEVAL, V_1^2.COMM) \\
(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_1^1.COMM) & \cup (D_1^2.PCODE \Rightarrow V_1^2.SALEVAL, V_1^2.COSTVAL) \\
(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_1^1.COMM) & \cup (D_1^2.PCODE \Rightarrow V_1^2.SALEVAL, V_1^2.COMM) \\
\end{align*}
\]

The data set \( (D_1^1.SCODE \leftrightarrow R_1.SNAME) \) is not examined for relationships because the relative attribute depends on one dimension attribute. Consider the data set \( (D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \). When the dimension attributes \( D_1^1.SCODE \) and \( D_2^1.ACODE \) come from data entity \( A \) and are separated as independent dimensions by using the ICP of the partitioning module, these are disregarded.

Consider the ECP of data set \( (D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_1^1.COMM) \cup (D_2^2.CCODE \Rightarrow V_1^2.SALEVAL) \). For process 3, when dimensional attributes \( D_1^1.SCODE \) and \( D_2^1.CCODE \) do not reveal a relationship, the process skips these data sets.
Finally, when the ECP of data set \((D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM)\) and 
\((D_2^2.SCODE \bowtie D_2^2.PCODE \Rightarrow V_2^1.SALEVAL)\) are compared, referring to process 1 and process 4, the intimate of dimension attribute SCODE defined in data dictionary is similar, the new data set \((D_1^1.SCODE \bowtie D_2^2.PCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM)\) will be additionally invented.

Consider the execution of ECP from \((D_1^1.SCODE \bowtie D_2^2.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM)\) as base data set yields:

\[
(D_1^1.SCODE \bowtie D_2^2.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.CCODE \Rightarrow V_1^2.SALEVAL)
\]

\[
(D_1^1.SCODE \bowtie D_2^2.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \Rightarrow V_2^1.SALEVAL, V_2^1.COMM)
\]

\[
(D_1^1.SCODE \bowtie D_2^2.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.SCODE \bowtie D_2^2.PCODE \Rightarrow V_1^2.SALEVAL)
\]

In the data comparison of \((D_1^1.SCODE \bowtie D_2^2.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.SCODE \bowtie D_2^2.PCODE \Rightarrow V_2^1.SALEVAL)\), the new grouping relation set will be \((D_1^1.SCODE \bowtie D_2^2.ACODE \bowtie D_2^2.PCODE \Rightarrow V_1^1.SALEVAL, V_2^2.COMM)\) when dimension attributes \(D_1^1.SCODE = D_2^2.SCODE\) and variable attributes \(V_1^1.SALEVAL = V_2^2.SALEVAL\). The corresponding SQL statement is like this:

\[
\begin{align*}
\text{SELECT} & \ A.SCODE, A.ACODE, B.PCODE, \text{sum}(A.SALEVAL), \text{sum}(A.COMM) \\
\text{BY} & \ SALE A, PRODUCT B \\
\text{WHERE} & \ A.SCODE = B.SCODE \\
\text{GROUP BY} & \ A.SCODE, A.ACODE, B.PCODE
\end{align*}
\]

The outcome of new grouping relation sets is like this:

\[
\begin{align*}
- & \ D_1^1.SCODE \bowtie D_2^2.PCODE \Rightarrow V_1^1.SALEVAL, V_2^2.COMM \\
- & \ D_1^1.SCODE \bowtie D_2^2.ACODE \bowtie D_2^2.PCODE \Rightarrow V_1^1.SALEVAL, V_2^2.COMM \\
- & \ D_1^1.SCODE \bowtie D_2^2.PCODE \Rightarrow V_1^1.SALEVAL, V_2^2.COSTVAL
\end{align*}
\]

The grouping relation sets of partitioning modules and combined grouping relation sets of clustering module are represented in the form of logical DVR models. However, analysts may reform or rebuild logical DVR models at the designated interface nodes (see Figure 2). The resulting DVR model forms will automatically generate an SQL statement in the programming process in order to import data values. This is discussed in the following paragraphs.

6. Building of a Physical DVR Model

Three modules (classification, partitioning, and clustering) generate DVR models which are repeatedly reviewed for error contamination by analysts at the designated interface nodes. In addition to archiving data volumes, analysts may approve missing dimensions or change the relation set of dimensions at any of the designated interface nodes.

In the next step, grouping relation sets will rearrange the order of dimensions to the satisfaction of the analysts. In order to crosscheck for data errors, sample data volumes from the relational database will verify the achieved forms of each DVR model.

6.1 Data Model Relocation Process

The order of dimensions will affect the variable’s layout in a table or report. The first dimension listed in the variable is the fastest access data, the next dimension is the next fastest, and the last dimension is the slowest access.

Assume the combined grouping relation set is \(D_1^1.SCODE \bowtie D_1^2.ACODE \bowtie D_1^2.PCODE \Rightarrow V_1^1.SALEVAL, V_2^2.COMM\). The dimension attribute SCODE appears on the first order following ACODE and PCODE. Thus, dimension SCODE is the fastest access data.
that appears in the columns. The dimension ACODE is the next fastest that appears in rows, and dimension PCODE is the last to appear in the page, as shown in Figure 3a below.

Assuming that the dimension attribute PCODE is desired to be in the column-level as indicated in Figure 3b below, then the grouping relation set is modified as $D_1^P \cdot PCODE \approx D_1^A \cdot ACODE \Rightarrow V_1^R \cdot SALEVAL, V_2^C \cdot COMM$.

Although the series of dimensions in a grouping relation set are not significant in conceptual view, it is easily to rotate the multidimensional cube. Processing MDDB will be easily accessed through the efficient use of CPU and data storage.

6.2 Data Loading and Building Physical Data Model

The sample data from the warehouse database may be transferred into DVR model forms using the import command of OLAP tools. These tools organize one portion of output data in order to test data consistency and accuracy. To build the physical DVR model, SQL statement will be generated automatically and then automatically executed to create data value, and automatically downloaded into the DVR models (refer to the use of SQL statements in Example 3).

7. Conclusion

Existing literature shows a serious gap in the field of database management, particularly in minimizing human errors in the process of database maintenance. The errors come from the current process of transforming relationship patterns in a database from a bidimensional to a multidimensional pattern for multidimensional or DVR analysis, which is done via manual interface.

This paper fills the gap by proposing an automated process that identifies, corrects, summarizes, and screens out errors in a large mass of data, using a method that allows multidimensional relationship via a program that generates DVR patterns of complex relationships in a database as well as screens irrelevant data. This paper explains the basic
process of creating DVR models that are validated by propositional functions. This paper also illustrates how to relocate data models and efficiently load data in MDDB.

The procedure uses data mining principles to design DVR models in MOLAP via three modules: classification, partitioning, and clustering. These modules work together to generate the multidimensional patterns of relationships using logical formulas that classify, map, and group variables in the data sets. The procedure is applicable to warehouse databases to automatically create DVR models that make MDDB work.

8. References
