World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:12, 2007

Conceptual Multidimensional Model

Manpreet Singh, Parvinder Singh, and Suman

Abstract**—**The data is available in abundance in any business organization. It includes the records for finance, maintenance, inventory, progress reports etc. As the time progresses, the data keep on accumulating and the challenge is to extract the information from this data bank. Knowledge discovery from these large and complex databases is the key problem of this era. Data mining and machine learning techniques are needed which can scale to the size of the problems and can be customized to the application of business. For the development of accurate and required information for particular problem, business analyst needs to develop multidimensional models which give the reliable information so that they can take right decision for particular problem. If the multidimensional model does not possess the advance features, the accuracy cannot be expected. The present work involves the development of a Multidimensional data model incorporating advance features. The criterion of computation is based on the data precision and to include slowly change time dimension. The final results are displayed in graphical form.

Keywords**—**Multidimensional, data precision.

I. Multidimensional Data Model

THE data involved in any business organization can be labeled as a as shown in Fig. 1. Any point inside the cube is at the intersection of the coordinates defined by the edges of the cube. For the business described above, the edges of the cube are Product, Market, and Time. Most people can quickly understand and imagine that the points inside the cube are where the measurements of the business for that combination of Product, Market, and Time are stored.

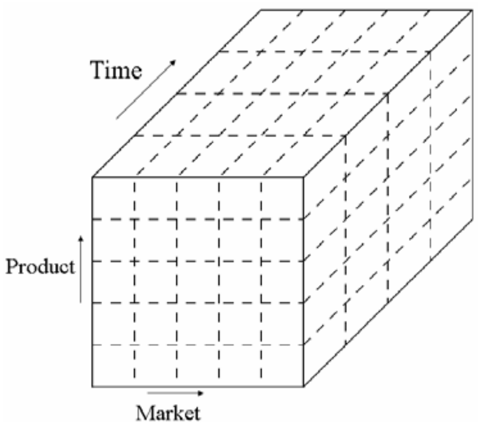


Fig. 1 Three dimensional model of a business

Manuscript received October 9, 2001.

Manpreet Singh is with the Department of CSE & IT, Guru Nanak Dev Engineering College, Ludhiana, India (e-mail: mpreet78@yahoo.com).

Parvinder Singh is with Department of CSE, Rayat and Bahara Institute of Engineering and Technology, Ropar, India (e-mail: author@lamar. colostate.edu).

Suman is with Department of Computer Engineering, Dronacharya College of Engineering, Gurgaon – 122001, India (e-mail: suman_aggroia@yahoo.com).

A data cube in data warehousing is not necessarily a three-dimensional (3-D) geometric structure, but is essentially *N* dimensional (N-D). The edges of the cube are called dimensions**,** which are the perspectives or entities with respect to which an organization wants to keep records. Each dimension may be associated with a dimension table**,** which describes the dimension.

For example, a dimension table for Product may contain such attributes as product Key, description, brand, category, etc., which can be specified by managers or data analysts. For those dimensions that are non-categorical, such as Time*,* the data warehouse system should be able to automatically generate the corresponding dimension table based on the data distribution. As a side note, the Time dimension is in fact of particular significance to decision support for trend analysis. Often it is desirable to have some built-in knowledge of calendars and other aspects of the time dimension.

In addition, a data cube in data warehousing is mainly constructed to measure the company's performance. Thus, a typical multidimensional data model is organized around a theme, which is represented by a fact table of some numerical measures — the objects of analysis. For example, a fact table may contain sales, budget, revenue, inventory, number of items sold etc. Each of the numerical measures depends on a set of dimensions, which provide the context for that measure. Therefore, the dimensions together are assumed to uniquely determine the measure, which is a value in the multidimensional space of dimensions.

Dimensions are hierarchical by nature. For example, dimension Time can be described by the attributes Year, Quarter, Month, and Day. Alternatively, the attributes of a dimension may be organized into a lattice, which indicates a partial order for the dimension. That is, the same Time dimension can have Year, Quarter, Month, Week, and Day instead. With this scheme, the Time dimension is no longer a hierarchy because some weeks in the year may belong to different months.

Therefore, if each dimension contains multiple levels of abstraction, the data can be viewed from different perspectives flexibly. A number of typical data cube operations:

A. Roll-up (increasing the level of abstraction)

B. Drill-down (decreasing the level of abstraction or increasing detail)

C. Slice and dice (selection and projection) and

D. Pivot (re-orienting the multidimensional view of data), exist to allow interactive querying and analysis of the data at hand. These operations are known as On-Line Analytic Processing (OLAP).

Decision makers would like to ask questions like "compute and rank the total sales by each country (or by each year)". They would also like to compare two numerical measures such as sales and budget aggregated by the same dimensions. Thus, another distinctive feature of the multidimensional data model is its stress on aggregation of numerical measures by one or more dimensions, which is one of the key operations mainly to speed up query processing time.

II. Features of Multidimensional Data Model

A. Explicit Hierarchies in Dimensions

The hierarchies in dimension should be captured explicitly by schema. This permits the user to drill-down and roll-up. In example, the hierarchies address<city<country should be captured.

B. Symmetric Treatment of Dimension and Measures

The data model should allow measures to be treated as dimensions and vice versa. For example the attribute age for product would be treated as a measure, to allow for computations such as average age, etc., but this should be able to define Age dimensions which allow to group the product into age groups.

C. Multiple Hierarchies in Each Dimension

A single dimension can have several paths for aggregating data. As an example, assume that there is time dimension on the date of purchase attribute. Days roll up to weeks and to months, but weeks do not roll up to year. To model this, multiple hierarchies in each dimension are needed.

D. Support for Aggregation Semantics

The data model should capture the aggregation semantics of the data, closely related to summarizability, and use this to provide a “safety net” that catches queries that might give results that have no meaning to the user. Aspect of this include built in support for avoiding double-counting of data and avoiding addition of non-additive data [7].

For example, when asking for the number of customer in different product groups, this should only count the same customer once per group, even though that customer may have several products in a group. The user should also be able to specify which aggregations are considered meaningful for the different kind of data available, and the model should provide a foundation for enforcing these specifications. As an illustration, it may not be meaningful to add inventory level together, but performing average calculation on them does make sense. In the field of statistical database, a closely related concept is summarizability which means that an aggregate result, e.g., total sales, can be computed by directly combining results from lower level aggregation, e.g., the sales for each store.

E. Non-Strict Hierarchies

The hierarchies in a dimension are always not strict, i.e. this can have many-to-many relationships between the different level in a dimension. In example, hierarchies are not strict. The data model should be able to handle these just as well as “ordinary” strict dimensions.

F. Non-Onto Hierarchies

Often, the hierarchies in a dimension are not balanced i.e. the path from the root to the leaves has varying length.

G. Non Covering Hierarchies

Another common feature of real world hierarchies is that links between two nodes in the hierarchies “skip” one or more levels. For example, the address “305 Rural Road” in the residence hierarchies is mapped directly to the country, bypassing the city level.

H. Many to many Relationships between Facts and Dimension

The relationships between facts and dimension are not always the classical many-to-one.

İ. Handling Change and Time

Although data change over time, it should be possible to perform meaningful analysis across time when data change. For example one product can be superseded by two new ones, but customers are still treated with old one. It should be possible to easily combine data across changes. The problem is referred to as handling slowly changing dimensions as part of this problem.

J. Handling Different Levels of Granularity

Fact data might be registered at different granularities. For example, the customer of a product may be registered differently by different experts. Some will use a very specific customer while others may use the less precise which cover several low level customer. It should still be possible to get correct analysis results when data is registered at different granularities .

K. Handling Imprecision

Finally, it is very common to be able to capture directly the imprecision in the data and allow queries to take into account. For example the customer has varying precision and it is important that this is captured and communicated to the user.

III. Handling Imprecision

Alternative queries may be used when the data is not precise enough to answer queries precisely, i.e., when the data used to group on is registered at granularities coarser than the “grouping” categories.

A. Overview of Approach

Along with the model definition, here presented how the case study would be handled in the model. This also showed how imprecision could be handled, namely by mapping facts to dimension values of coarser granularities when the information was imprecise. An illustration of the approach, showing how the possible spectrum of imprecision in the data is captured using categories in a dimension, is seen in Fig. 2.

The approach has a nice property, provided directly by the dimensional "imprecision" hierarchy described above. When the data is precise enough to answer a query, the answer is obtained straight away, even though the underlying facts may have varying granularities. The general approach to handling a query starts by testing if the data is precise enough to answer the query, in which case the query can be answered directly. Otherwise, an alternative query is suggested. In the alternative query, the categories used for grouping are coarsened exactly so much that the data is precise enough to answer the (alternative) query. Thus, the alternative query will give the most detailed precise answer possible, considering the imprecision in the data. Examining our algebra, we see that imprecision in the data will only affect the result of two operators, namely selection and aggregate formation (the join operator tests only for equality on fact identities, which are not subject to imprecision). Thus, this need only handle imprecision directly for these two operators; the other operators will just "pass on" the results containing imprecision untouched. However, if this can handle imprecision in the grouping of facts, ordinary OLAP style "slicing/dicing" selection is also handled straightforwardly, as slicing/dicing is just selection of data for one of a set of groups.

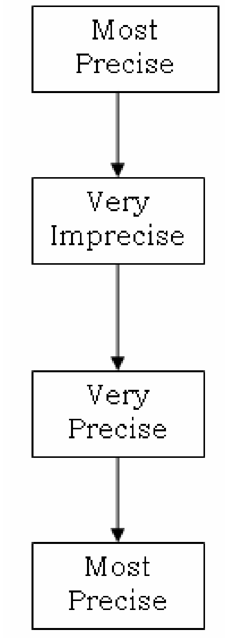


Fig. 2 The Spectrum of Imprecision

Following this reasoning, the general query that must be considered is:

α [C_1 ………..Cn, D_{n+1}, g](M), where M is an n-dimensional MO, C_1 ………Cn are the "grouping" categories, D_{n+1} is the result dimension, and g is the aggregation function. The evaluation of the query proceeds (logically) as follows. First, facts are grouped according to the dimension values in the categories C1….Cn that characterizes them. Second, the aggregate function g is applied to the facts in each group, yielding an "aggregate result" dimension value in the result dimension for each group. The evaluation approach is given by the pseudo-code below. The text after the "//" sign are comments.

```
Procedure EvalImprecise (Q, M)              //Q is a query is an MO.
If PreciseEnough (Q, M) then Eval (Q, M) //if data is precise enough, use normal evaluation
Else
O'= Alternate (Q, M)                        //Suggest alternate query
If Q' is accepted then Eval (Q', M)         //use normal evaluation
for alternate query
else
Handle Imprecision in Grouping for Q
Handle Imprecision in Aggregate Computation for Q
Return imprecision in Aggregate Computation for Q
Return Imprecise Result of Q
  end if
  end if
```

Here overall approach to handling the imprecision in all phases will be to use the granularity of the data, or measures thereof, to represent the imprecision in the data. This allows for simple and efficient handling of imprecision [8].

A. Alternative Queries

The first step in the evaluation of a query is to test whether the underlying data is precise enough to answer the query. This means that all facts in the MO must be linked to categories that are less-than-or-equal" to the "grouping" categories in the query.

In order to perform the test for data precision, there need to know the granularities of the data in the different dimensions. For this, for each MO, M, this maintains a separate precision *MO*, Mp. The precision *MO* has the same number of dimensions as the original MO. For each dimension in the original MO, the precision MO has a corresponding "granularity" dimension. The i'th granularity dimension has only two categories, Granularity_i and $\top_{pi}$. There is one *value* in a "Granularity" category for each category in the corresponding dimension in M. The set of facts F is the same as in M, and the fact-dimension relations for Mp map a fact f to the dimension value corresponding to the category that f was mapped to in M. The determination of whether a given query can be answered precisely is dependent on the actual data in the MO, and can change when the data in the MO is changed. Thus, this need to update the precision MO along with the original MO when data changes.

Formally, given an MO, M= (S, F, D, R), where:

$$S = (F, D),\ D = \{Ti, i = 1 \ldots n\},$$

Ti = (Ci, $\leq Ti), Ci = \{Cij\}, D = \{Di, i = 1......n\}$. and Rp = (Rpi,i= 1……n) we define the

precision MO,Mp, as:

$$Mp = (S_p, F_p, D_p, R_p),$$
where

$S_p = (F_p, Dp)$, $Fp = F$, $D_p = \{T_{pi}, I = 1\ldots n\}$,

$T_{pi} = \{Granularity_i, T_{pi}\}$,

$F_p = F$, $D_p = \{D_{pi}, 1=1\ldots n\}$,
$D_{pi} = (C_{pi}, \leq_{pi})$, $C_{pi} = \{Granularity_i, Tpi\}$,
$Granularity_i = \{GD_i (e) | e \in D_i\}$, $T_{pi} = \{T_i\}$, and
$e_1 \leq_{pi} e_2 \Leftrightarrow (e_1 = e_2) \vee (e_1 \in Granularityi \wedge e_2 = Ti$, and
$R_{pi} = \{(f, G_{Di}(e)) | (f, e) \in R_i\}$

The test to see if the data is precise enough to answer a query ü can be performed by rewriting Q = α [C1……Cn, Dn+1,g](M) to a "testing" query Qp = α [G1……..Gn,Gn+1 Set Count] where Gi' is the corresponding "granularity" component in Dpi if Ci $\neq \top_i$ Otherwise, Gi = $\top_i$. Thus, we group only on the granularity components corresponding to the components that the expert has chosen to group on.

IV. Handling Time/Slowly Changing Dimensions

Dimensions that change over time are called Slowly Changing Dimensions. For instance, a product price changes over time; People change their names for some reason; Country and State names may change over time. These are a few examples of Slowly Changing Dimensions since some changes are happening to them over a period of time.

Slowly Changing Dimensions are often categorized into three types namely Type1, Type2 and Type3 as explained in figure:

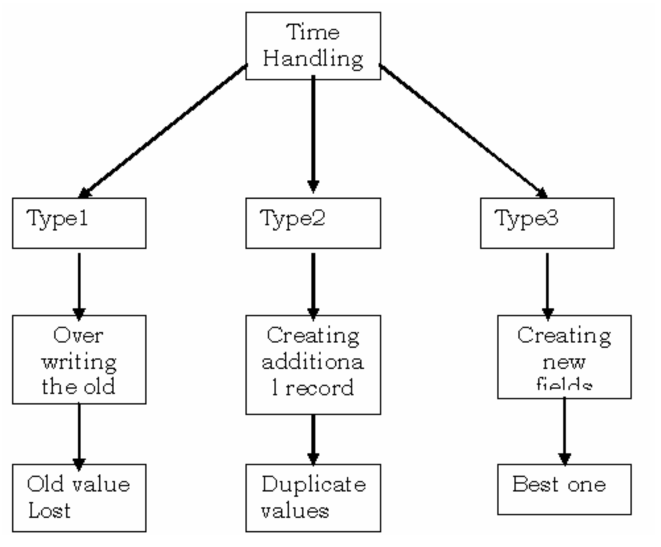


Fig. 3 Time Handling

The following section deals with how to capture and handling these changes over time.

The "Product" table mentioned below contains a product named, Product1 with Product ID being the primary key. In the year 2004, the price of Product1 was $150 and over the time, Product1's price changes from $150 to $350. With this information, let us explain the three types of Slowly Changing Dimensions.

TABLE I
Product Price in 2004

Product ID(PK)	Year	Product name	Product price
1	2004	Product1	$150

A. Type 1(Overwriting the Old Values)

In the year 2005, if the price of the product changes to $250, then the old values of the columns "Year" and "Product Price" have to be updated and replaced with the new values. In this Type 1, there is no way to find out the old value of the product "Product1" in year 2004 since the table now contains only the new price and year information.

TABLE II
Product Price in 2005

Product ID(PK)	Year	Product Name	Product price
1	2005	Product1	$250

B. Type 2 (Creating another Additional Record)

In this case, the old values will not be replaced but a new row containing the new values will be added to the product table. So at any point of time, the difference between the old values and new values can be retrieved and easily be compared. This would be very useful for reporting purposes.

TABLE III
Product Price for 2004 and 2005

Product ID(PK)	Year	Product name	Product price
1	2004	Product1	$150
1	2005	Product1	$250

The problem with the above mentioned data structure is "Product ID" cannot store duplicate values of "Product1" since "Product ID" is the primary key. Also, the current data structure doesn't clearly specify the effective date and expiry date of Product1 like when the change to its price happened. So, it would be better to change the current data structure to overcome the above primary key violation.

TABLE IV
PRODUCT TABLE WITH DATE AND TIME

Product ID(PK)	Effective Date Time(PK)	Year	Product name	Product Price	Expiry Date Time
1	01-01-200412.00AM	2004	Product1	$150	12-31-2004 11.59PM
1	01-01-2005 12.00AM	2005	Product1	$250	

In the changed Product table's Data structure, "Product ID" and "Effective Date Time" are composite primary keys. So there would be no violation of primary key constraint. Addition of new columns, "Effective DateTime" and "Expiry DateTime" provides the information about the product's effective date and expiry date which adds more clarity and enhances the scope of this table. Type2 approach may need additional space in the data base, since for every changed record, an additional row has to be stored. Since dimensions are not that big in the real world, additional space is negligible.

C. *Type 3 (Creating New Fields)*

In this case, the latest update to the changed values can be seen. Example mentioned below illustrates how to add new columns and keep track of the changes. From that, Here able to see the current price and the previous price of the product: Product1.

TABLE V
PRODUCT HISTORY FOR 2005

Product ID (pk)	Current year	Product name	Current product price	Old product price	Old year
1	2005	Product1	$250	$150	2004

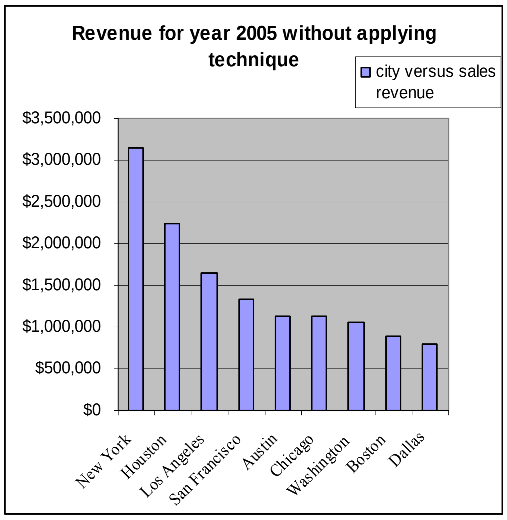
The problem with the Type 3 approach is over years, if the product price continuously changes, then the complete history may not be stored, only the latest change will be stored. For example, in year 2006, if the product 1's price changes to $350, then we would not be able to see the complete history of 2004 prices, since the old values would have been updated with 2005 product information.

TABLE VI
PRODUCT HISTORY FOR 2006

Product ID(PK)	Year	Product Name	Product Price	Old Product Price	Old Year
1	2006	Product1	$350	$250	2005

V. RESULTS AND DISCUSSION

Motivated by the popularity of On-Line Analytical Processing (OLAP) systems for analyzing business data, multidimensional data models have become a major database research area. However, current models do not handle well the complex data found in some real-world systems. This present a real-world case study from the retail business, where this track customer product their names, social security numbers, dates of birth, ages, and places of residence. The average is chosen to suit the purpose of giving a rough “measure of inclusion.” justifies requirements that a multidimensional data model must satisfy in order to support the complex data found in real-world applications. Requirements not handled by current models include data precision and slowly changing time dimension, twelve previously proposed data models are evaluated according to the requirements, and it is shown that none of them satisfies more than four requirements fully or partially. A new, extended multidimensional data model is purposed, which addresses two major requirements. The data model improves over previously proposed models by supporting precision and slowly changing time dimension. Especially, time is handled by adding valid time and transaction time to the basic model. Here propose algebra on the multidimensional objects from the model, and this show that it is closed and at least as strong as relational algebra with aggregation functions. The results are realized by the chart in Fig. 4 which tells the difference in existing model and proposed model.



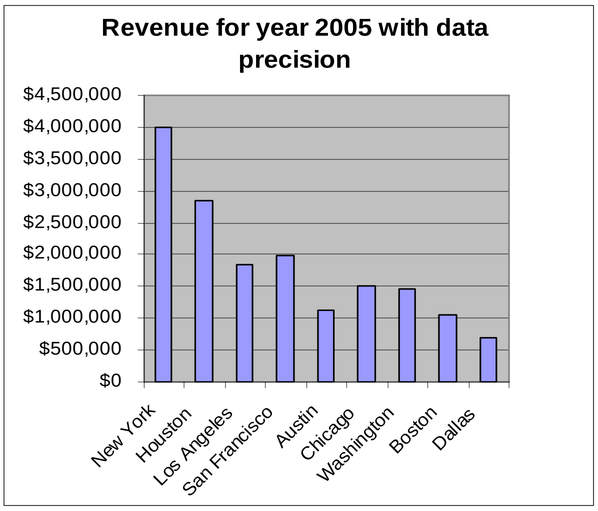


Fig. 4 Revenue as shown by the System

REFERENCES

[1] P. Adriaans and D. Zantinge *"Data Mining"*, Pearson Education, USA, 2002, pp. 1-200.

[2] Alexandros Karakasidis "*ETL queues for active data warehousing" Sixth* International Conference on Extending Database Technology, USA, 2005, pp. 153–165.

[3] A. L. P. Chen, J-S. Chiu and F. S. C. Tseng, *"Evaluating Aggregate Operations over imprecise Data"*, IEEE Transactions on Knowledge and Data Engineering, Vol8, 1996, pp.273–284.

[4] C. Bettini, C. E. Dyreson, W. S. Evans, R. T. Snodgrass, X. S. Wang, "*A Glossary of Time granularity Concepts",* In Temporal Databases: Research and Practice, 1998, pp. 406–413.

[5] C. Li and X. S. Wang, "*A Data Model for Supporting On-Line Analytical Processing"* Fifth International Conference on Information and Knowledge Management, 1996, pp. 81–88.

[6] Chang-Sub Park, young Ho Kim, Yoon-Joon Lee, "Rewriting OLAP Queries using Materialized Views and Dimension Hierarchies in Data warehouses" IEEE, 2001, pp. 515-523.

[7] Daniel A. Keim, Hans-Peter Kriegel, "Visualization technique for Mining Large databases: A Comparison", IEEE Transaction on Knowledge and Data Engineering,Vol 8., 1996, pp.923-938.