

A Probabilistic Multidimensional Data Model and Algebra for OLAP in Decision Support Systems

Bhaskara Reddy Moole
Wonder Technologies Corporation
13063 Marcey Creek Road, Herndon, VA 20171
bhaskarareddy@wonderotechnology.com

Although there are models defined for multidimensional data, they lack a comprehensive way to handle uncertain data. Uncertainty is pervasive over the real world and any model to represent real world data that ignores uncertainty imposes unacceptable limitations on the decision support systems in which it is used. No methods were proposed so far to incorporate uncertainty into multidimensional data models. We propose a probabilistic multidimensional data model and an associated algebra to handle uncertainty.

Keywords: OLAP, Data Warehouse, Multidimensional Data, Uncertainty, Probability, Probabilistic Data, Bayesian, Data Model, Decision Support

Introduction

The Data Warehouse and OLAP (On-Line Analytical Processing) are two emerging technologies, which enable the enterprises to handle extremely large amounts of data efficiently. Traditional RDBMS technology is unsuitable for this task because of the queries (aggregate / summary / grouping type) involved. The Data Warehouse is a term used to describe a very large repository of historical data. Kimball et al. [1] describe it as "The queryable source of data in the enterprise". The OLAP refers to the process of querying and presenting the data in the Data Warehouse. The concepts of "Data Warehouse" and "OLAP" are closely related in this sense. An excellent discussion of these concepts can be found in Kimball et al. [1] and Codd et al. [2]. These technologies are used in several industries such as retail sales, telecommunications, financial services, real estate, and in general business intelligence gathering. According to industry reports, the Datawarehousing and OLAP product sales reached \$9 billion in 1997. The market for these technologies is growing very fast. Current commercial products offering some of the features of Datawarehousing and OLAP include Red Brick from Red Brick Systems, EssBase from Hyperion, Express from Oracle, and IQ from Sybase.

The main criteria in selecting an OLAP product are described by Codd et al. [2]. They are:

1. Multidimensional Conceptual View

2. Transparency
3. Accessibility
4. Consistent Reporting Performance
5. Client-Server Architecture
6. Generic Dimensionality
7. Dynamic Sparse Matrix Handling
8. Multi-User Support
9. Unrestricted Cross-dimensional Operations
10. Intuitive Data Manipulation
11. Flexible Reporting
12. Unlimited Dimensions and Aggregation Levels

Some researchers enlisted selection criteria for the OLAP products, while others defined the functionalities required of these systems. A comprehensive collection of these features and functionalities is listed in Thomas and Datta [3]. They are reviewed below and a new set of functionalities to make existing OLAP systems more useful in decision support systems is added.

Functionality requirements of OLAP Systems

The multi-dimensional view of the data is an essential requirement of OLAP systems. Hence most of the functionality requirements assume the underlying construct is multi-dimensional. Most researchers and OLAP system vendors refer to this multi-dimensional construct as Data Cube. The functionality requirements are divided into four categories. First category is Data Cube Operations. In this category, slice, dice, drill-down, roll-up, and pivot operations are the most important. Slicing is the operation of selecting the dimensions used to view the cube. It is analogous to the selection operation in relational algebra. Dicing is the operation of selecting actual positions or values on a dimension. It is analogous to the projection operation in relational algebra. Roll-up is the operation of increasing the granularity along one or more dimensions. For example, an analyst with access to sales in a city may want to see the sales for the entire state to put the city results into correct perspective. He may want to see the sales at even higher level such as for the entire region. That is, roll-up operation performs analysis across a hierarchy of a dimension. Drill-down is the converse operation of decreasing the granularity. An analyst with access to region

sales may want to see selectively more detailed level data for a state and then for a city. It is traversing the dimensional hierarchy in decreasing level of granularity. Pivot refers to the aggregation of two or more dimensions to produce a new multidimensional view having an attribute for each grouping dimension and additional attributes for the aggregated measure. Second category of functionality requirements is Aggregation. In addition to standard SQL aggregate operators (e.g. MIN, MAX, SUM, AVG, COUNT), an OLAP system needs to support operators such as Ranking, Percentiles, Comparisons of aggregates, Attribute-based grouping, Trends, Time-dimension based aggregate comparisons and so on. Third category of functional requirements is Transformations. Force operator converts a dimension to a measure and Extract operator converts a measure into a dimension. Fourth category is related to handling of uncertain data. This category includes well-defined mechanism for representing, modifying, and transforming uncertain data consistently within the model as well as in associated operations. As we argue in later sections, without supporting uncertainty, the usefulness of OLAP systems will be limited to the point of being unacceptable for many real world business tasks.

As we mentioned earlier, the most important feature of OLAP systems is the "Multidimensional Conceptual View". It has been widely accepted that rendering the data using a multidimensional logical construct is the most important feature of Data Warehouses and OLAP systems. Several RDBMS vendors provided products based on this concept. The concept is generally known as "Data Cube". However, the RDBMS vendors extended their databases to include some of the prominent features without a common conceptual data model supporting them. This is a major drawback to the development of OLAP technology. Thomas and Datta have developed a conceptual model and algebra for On-Line Analytical Processing in Decision Support Systems [3]. Their model is a simple but generic model of data cube along with algebra to support OLAP operations on the data cube. Conceptually, this model can be considered as an extension to the relational model. The relational model is developed to handle tabular data structure while Thomas and Datta extended it to handle data cubes (multiple dimensions). However, their research completely ignores the issue of uncertainty.

The uncertainty is pervasive over the real world. A simple OLAP application for business competition analysis will need to handle uncertainty in the data. For example, if Koke wants to construct a data warehouse to store the estimated sales of competitor Bepsi's products over a period of time, the data will necessarily contain uncertainty. Similarly, weekly forecasts of sales normally contain uncertainty and when collected over a period of time will yield a data warehouse with uncertainty measure for each forecast. Such data warehouse can be used to analyze the accuracy of the models used to forecast and compare them with actual sales to tune the models to forecast better. These are two examples of uncertainty in Data Warehouses. There are

several sources of uncertainty. In empirical situations, uncertainty is a result of measurement errors and limits of measuring instruments. For example, weight of a product can be measured easily, but can never be certain about the actual weight because the scale could be malfunctioning. When measuring the distances between galaxies, limitation of instruments used will introduce uncertainty. The natural language also contains ambiguity and vagueness. For example, we use words such as tall and heavy without assigning any measures (vagueness). Similarly, we specify a fugitive criminal's height is between five and six feet (ambiguity). This uncertainty is at the cognitive level. The individuals and societies use uncertainty as a strategic tool. For example, a country with few nuclear weapons may want to project its arsenal much bigger, thus making information possessed by others very uncertain. The uncertainty is often introduced by not disclosing the information. Companies often do not disclose negative information to make the products more marketable. If an intelligence agency has to capture the information about terrorists from several informants with varying credibility, it needs to have a system, which can handle the uncertainty. Due to theories in Quantum Mechanics, it is now generally accepted that humans cannot find the exact position of a particle in motion, but can only predict its location in the a given region of space with some probability. Representing data about these particles requires the system to handle uncertainty. These are but, some simple examples of uncertainty-based information. A more comprehensive treatment of uncertainty-based information can be found in Klir and Wierman [4]. There are several frameworks to represent the uncertainty such as Fuzzy Set Theory, Evidence Theory, Possibility Theory, and Probability Theory. Among these, the Probability Theory is the most well understood and very thoroughly worked out theory. In this paper we will use the probability as the measurement of uncertainty. We take the view of the Bayesian school of probability. In this view, probability represents a person's belief strength in a given event or proposition (it is subjective). To support the objectivity requirement of scientific rationalism, such as in the case of probability derived from repetitive experiments, we require that the belief strength assigned to the event or proposition should be identical to the derived probability if it is available. We also note that Bayesian model requires this convergence of personal beliefs with probabilities derived through repetitive experiments, no matter what prior initial belief strengths are used.

So far, we discussed why we needed to handle uncertainty in OLAP systems. Now, we will discuss briefly how we can use the uncertainty represented in an OLAP system. As an example, take a multi-dimensional database having probability associated with sales of each product of competitor Bepsi, such as "In 1995, Bepsi sold 10 units of product P2 in Chicago at the price of \$100K with a probability 0.8" and "In 1995, Bepsi sold 10 units of product P2 at the price of \$110K in Chicago with a probability 0.1". These data elements represent uncertainty. These are in

contrast to statements like “In 1995, Bepsi sold 10 units of product P2 at the price of \$100K” which are not available. The database that can only handle certain data is useless for this application. If we can represent such uncertain data, we also have expanded the possibility for the querying this database with imprecise queries. As an example, consider a query “what are the most likely sales of Bepsi in Chicago in 1995” or “what is most likely average sales of P2”. Allowing such queries makes the system much more useful to the users in interpreting the answers. In addition, assume that the result can be returned with confidence levels assigned to them such as “Average sales of Bepsi is 95% certain to be between \$100K and \$108K”. This kind of formulation of result is possible by using decision-theoretic methods on uncertain data. Many researchers worked on different aspects of multi-dimensional data representation as well uncertainty handling in databases. However, there was no attempt to incorporate uncertainty into multidimensional models. The following section surveys most relevant research work reported in this area.

Related Work

In this section, we will provide an overview of the research performed in the areas of conceptual models for multidimensional databases and uncertainty in databases.

Several researchers and database vendors tried to address the Datawarehousing and OLAP problems. However, they all provided solutions to a specific set of problems. Most of the prior research concentrated on extending the Relational Database technology to handle the new functionality. Many did not include a comprehensive model for the multidimensional data representation and associated operations model. A comparison of these models is available in Vassiliadis and Sellis [5].

The earliest attempt at providing a comprehensive model was made by Li and Wang in their 1996 presentation at the Proc. Conf. Inform. Knowledge Management [6]. In 1997, Agrawal, Gupta, and Sarawagi have provided a model for multidimensional data [7]. In the same year, Gyssens and Lakshmanan provided another model for multidimensional data [8]. All these models placed several restrictions on dimensions, attributes, or the types of queries. Thomas and Datta [3] eliminated most of these restrictions and made their model more generic. Their model is comprehensive in the sense that it covers all of the characteristics identified by them for an OLAP system. In addition, they proved that the algebra for their model is closed and is more expressive than relational algebra.

Thomas and Datta defined a logical level construct “data cube” and algebra to operate on the data cube [3].

Definition 1 Cube: A *cube* is a generalized, abstract structure that serves as the foundation for the

multidimensional data cube model. A cube C is defined as a six-tuple $\langle C, A, f, d, O, L \rangle$ where:

- C is a set of m characteristics
- A is a set of t attributes
- f is a one-to-one mapping, $f: C \rightarrow 2^A$, which maps a set of attributes to each characteristic
- d is a boolean-valued function that partitions C into a set of dimensions D and a set of measures M
- O is a set of partial orders on the set C
- L is a set of cube cells, each having the structure $\langle \text{address}, \text{content} \rangle$, represented by $\langle L.AC, L.CC \rangle$

Predicate: A *predicate* P is a well-formed formula in first-order predicate logic. A predicate may be an *atomic predicate* or a *compound predicate*. An atomic predicate is a restriction on the domain of a single variable. A compound predicate is an expression of atomic predicates connected by logical operators \wedge (and), \vee (or), \neg (not), \rightarrow (implies), and \leftrightarrow (equivalent to).

l satisfies P : l , an instance of L , with the structure $\langle \text{address}, \text{content} \rangle$ satisfies predicate P if and only if:

Case 1: if an element of l is a dimension, then $l.AC$ satisfies P , otherwise $l.CC$ satisfies P , if P is atomic and the truth-value is TRUE.

Case 2: if P is a compound predicate, l satisfies P , when all the truth-values evaluated together with the connecting operators results in TRUE.

On this logical Cube structure, they proposed the following operators.

Restriction (Σ): The restriction operator restricts the values on one or more attributes based on specified conditions. This is similar to the selection operator in relational algebra.

Metric Projection (Π^M): The metric projection operator restricts the output of a cube to include only a subset of the original set of measures. This is similar to the projection operator in the relational algebra.

Rename (A): The rename operator is a set operator similar to the standard relational algebra rename operator.

Cubic Product (\otimes): The cubic product operator is a binary operator that can be used to relate any two cubes. This is similar to the Cartesian Product operator in the relational algebra. The Join operator is expressed in terms of Cubic Product.

Union (\cup): The union operator is a binary operator that finds the union of two cubes. This is similar to the union operator in the relational algebra.

Difference (\ominus): The difference operator is a binary operator that finds the difference of two cubes. This is similar to the difference operator in the relational algebra. The intersection operator is expressed in terms of difference.

Aggregation (Γ): The aggregation operator performs aggregation on one or more dimensional attributes. This operator allows repeated additions and multiplications in addition to the standard SQL aggregate functions such as MIN, MAX, SUM, AVG, and COUNT, and a new RANK function.

Force (Ψ): The force operator converts dimensions to measures.

Extract (Φ): The extract operator converts measures to dimensions.

This algebra is closed, meaning the results of operators are cubes and repeated application of the operators always results in cubes.

As we can see, this model has no means to represent the uncertainty. That is, either a cube cell has a value (known) or does not (unknown). As mentioned earlier, this puts a restriction on the decision support capabilities of the OLAP system and limits its usefulness. The relational model suffered from the same limitations. To alleviate these limitations in relational models, Dey and Sarkar [9] proposed an extension to the relational model and algebra. We discuss their "Probabilistic Relational Model and Algebra" briefly below.

Informally, Dey and Sarkar's Probabilistic Relational Model stamps every tuple in a relation with a probability. This probability stamp represents the joint distribution of all the attributes in the relation taken together (in this context, probability represents strength of our belief that a real world object exists). Formally, A *relation scheme* R is a set of *attribute names* $\{A_1, A_2, \dots, A_n\}$, one of which may be a probability stamp pS . A tuple x over R represents our belief about attributes in R of a real world object. If $pS \in R$, then a restriction $x(pS) > 0$ represents the probability of the object $x(R - \{pS\})$. Symbolically, $x(pS) = \Pr[R - \{pS\} = x(R - \{pS\})]$. If the $pS \notin R$, then relation scheme R is deterministic. In this case, every tuple on R is assigned a probability of 1, and is not explicitly written. i.e. the probabilistic relational model is reduced to the regular relational model. If any two tuples x and y on R have $x(R - \{pS\}) = y(R - \{pS\})$, then they are called value-equivalents (denoted by \cong). Value-equivalent tuples must be coalesced. That is they are turned into a single tuple z by combining their probability stamps using the coalescence operations. There are two coalescence operations: coalescence-PLUS and coalescence-MAX.

coalescence-PLUS (\oplus): It is used in the definition of the projection operation and is defined on two value-equivalent tuples x and y as:

$$z = x \oplus y \Leftrightarrow (x \cong y) \wedge (z \cong x) \wedge (z(pS) = \min\{1, x(pS) + y(pS)\})$$

Intuitively, when two value-equivalent tuples were combined using projection operation (i.e. we believe in both

of the tuples) their individual probability is summed together. If the result is greater than 1, then existence of the object is certain and is assigned a probability of 1.

coalescence-MAX (\odot): It is used in the definition of the union operation and is defined on two value-equivalent tuples x and y as:

$$z = x \odot y \Leftrightarrow (x \cong y) \wedge (z \cong x) \wedge (z(pS) = \max\{x(pS), y(pS)\})$$

Intuitively, when two value-equivalent tuples were combined using union operation (i.e. we believe in only one of the tuple, the one with higher strength of belief), maximum probability of these tuples is the probability for the result tuple.

Both these coalescence operators can be applied recursively on any number of value-equivalent tuples.

On this probabilistic relational model, Dey and Sarkar defined the following relational algebraic operations. These include all the regular relational algebra operations and a new operation called *conditionalization*.

Union (\cup): This union operation is identical to the relational union operation except that the value-equivalent tuples are coalesced using the coalescence-MAX operator.

Difference ($-$): The difference of two value-equivalent tuples ($x - y$) is obtained by subtracting the probability of y from the probability of x . If the result p is a positive number, then a value-equivalent tuple z (to tuples x and y) is made part of the result with the probability stamp p . If p is a negative number, no tuple is included in the result for these two tuples. If the tuples x and y are not value-equivalent, both are added to the result.

Projection (Π): The projection operation is identical to the relational projection operation except that the value-equivalent tuples are coalesced using the coalescence-MAX operator.

Selection (σ): The selection operation is identical to the relational selection operation. Applying it on a relation results in all the tuples that satisfy the selection predicate.

Natural Join (\bowtie): The natural join operation is identical to the relational natural join except that the probability of the result is obtained by multiplying the probabilities of joined tuples. By noting that the probability of each tuple is the joint probability for that set of attributes, we can see that the result set is union of both tuples and the result tuple's probability must be a joint probability of all the attributes together. We can also see that the resulting probability is meaningful only when all the attributes are mutually independent.

Rename (ρ): The rename operation is identical to the relational rename operation. If the probability stamp itself is renamed, it loses its meaning and becomes a user-defined

attribute and the result tuple will be assigned an implicit probability of 1.

Conditionalization (γ): The conditionalization operation is a new operation introduced in the probabilistic relational algebra for the Dey and Sarkar's model. It marginalizes the probability of a subset of the original set of attributes in the relation R by summing over all the probabilities for the tuples with this subset of attributes. Thus, the result still represents the conditional probability of the subset of attributes. This new probability is meaningful only when the result contains a key.

The other relational operations such as Intersection, Cartesian product, Theta-join, and Alpha-cut are defined in terms of the above operations. This algebra is closed and the model is in 1-NF.

In a later publication, Dey and Sarkar [10] reported a framework to modify the probabilities when the information is added, deleted, and updated. They used Bayesian Theory to revise the probabilities of the modified tuples.

In the following section, we will present our probabilistic multidimensional data model and algebra to operate on the model. In subsequent sections, properties of this model are briefly discussed and areas for further research are identified. Last section contains conclusions.

Probabilistic Multidimensional Data Model and Algebra

Our Probabilistic Multidimensional Data Model is defined similar to the Thomas and Datta's multidimensional data model and Dey and Sarkar's probabilistic relational model. Our model addresses the shortcomings in the OLAP model of Thomas and Datta by incorporating probability into the model. The Dey and Sarkar's probabilistic model provides us a guideline in incorporating this. In this section, we follow the conventions used in Thomas and Datta, and Dey and Sarkar very closely.

Informally, in our model, we stamp each cell in the cube with a probability measure as shown in Figure 1. This probability stamp pS represents strength of our belief that there exists a real world object with given cell values. This is in contrast to the Thomas and Datta's model where a cell with a set of values represents existence of a real world object with certainty. Since we are using probability as the measure of the strength of our belief, its domain is $(0,1]$. When pS is 0, we are certain the real world object does not exist and when it is 1, we are certain it exists. When it is 0, we do not represent the cell values for that object. When it is 1, we will not write pS explicitly in the cube cell. In this deterministic case, our model reduces to the Thomas and Datta's model. We will use the Sales cube shown in Figure 1 as a running example throughout this section.

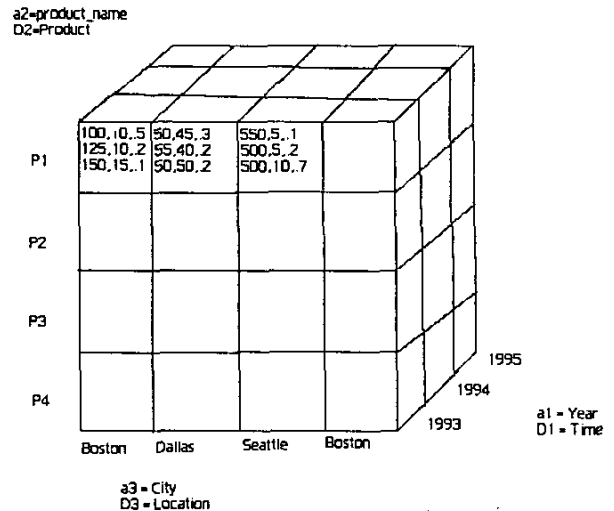


Figure 1. Cube Example with Notation

Definition 1 Cube: A cube is a logical structure comprising of a six-tuple $\langle C, A, f, d, O, L \rangle$ where:

- C is a set of m characteristics $\{c_1, c_2, \dots, c_m\}$ where each c_i is a characteristic having domain $(\text{dom } C)$, one of which may be BELIEF. If BELIEF is not a characteristic, then the cube is deterministic.
- A is a set of t attributes $\{a_1, a_2, \dots, a_m\}$ where each a_i is an attribute name having domain $\text{dom } A$, one of which may be a probability stamp pS . $\text{dom}(pS)$ is $(0,1]$. We assume that there exists an arbitrary total order on A , \leq_A . Thus, the attributes in A (and any subset of A) can be listed according to \leq_A . Moreover we say that each $a_i \in A$ is *recognizable* to the cube C .
- f is a one-to-one mapping, $f: C \rightarrow 2^A$, which maps a set of attributes to each characteristic. The mapping is such that
 - $\forall i, j, i \neq j, f(c_i) \cap f(c_j) = \emptyset$ i.e. pairwise disjoint attribute sets
 - $\forall x, x \in A, \exists c, c \in C, x \in f(c)$ i.e. all attributes are mapped
 - $f(\text{BELIEF}) \rightarrow \{pS\}$, iff BELIEF $\in C$ i.e. BELIEF is always mapped to pS

Hence, f partitions the set of attributes among the characteristics. $f(c)$ is referred to as the *schema* of c .

- d is a boolean-valued function that partitions C into a set of dimensions D and a set of measures M . Thus, $C = D \cup M$ where $D \cap M = \emptyset$. The function d is defined as:

$$\forall x \in C, d(x) = \begin{cases} 1 & \text{if } x \in D, \\ 0 & \text{otherwise.} \end{cases}$$

- O is a set of partial orders such that each $o_i \in O$ is a partial order defined on $f(c_i)$ and $|O| = |C|$. In other words, the schema for each characteristic c_i , has a partial order o_i associated with it.

- L is a set of cube cells. A cube cell is represented as an (address, content) pair.
 - The address in this pair is an n -tuple, $\langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle$, where n is the number of dimensional attributes in the cube, i.e. $n = |A_d|$, where A_d represents set of all dimensional attributes; i.e., $A_d = \cup_{i \in D} f(d_i)$. Each address component, α_i , represents a position along the "axis" of a dimensional attribute in A based on \leq_A (e.g., the third component of the address, α_3 , corresponds to the third dimensional attribute in A in \leq_A -order). One of these components may be pS.
 - The content of a cube cell is a k -tuple, $\langle \chi_1, \chi_2, \dots, \chi_k \rangle$, where k is the number of metric attributes in the cube, i.e., $k = |A_m|$, where A_m represents the set of all metric attributes; i.e. $A_m = \cup_{i \in M} f(m_i)$. Each content component, χ_i , represents the element of the content that corresponds to a particular metric attribute. χ_i corresponds to the i th metric attribute in A in \leq_A -order. One of these components may be pS.
 - The total probability of all the cells having the same address component must be no more than one. i.e.

$$\forall l \in L, \sum_{\substack{y \in L \\ l.AC = y.AC}} y.CC(pS) \leq 1.$$

- Two cells i and j are said to be *value-equivalent* iff the address component of i is identical to the address component of j and the content component of i without pS is identical to the content component of j without pS, when $d(\text{BELIEF})=0$. Value-equivalence is denoted by \cong . That is (see below for the notation),

$$i \cong j \Leftrightarrow \begin{cases} i \in L, \\ j \in L, \\ \text{if } \text{BELIEF} \in C \text{ then } d(\text{BELIEF}) = 0, \\ i.AC = j.AC \wedge i.CC - \{pS\} = j.CC - \{pS\} \end{cases}$$

Value-equivalent cells are not allowed and must be coalesced using the coalescence operations defined in the next section.

The following notations are used:

- $g: A \rightarrow C$, such that $g(a) = c$ iff $a \in f(c)$
- structural address component of L is denoted as L.AC
- structural address component of cell l is denoted as $l.AC$
- i th address component of cell l is denoted as $l.AC[i]$

- address component of a cell corresponding to an attribute name $aname$ is denoted as $l.AC(aname)$
- structural content component of L is denoted as L.CC
- structural content component of cell l is denoted as $l.CC$
- i th content value component of cell l is denoted as $l.CC[i]$
- content component of a cell corresponding to an attribute name $aname$ is denoted as $l.CC(aname)$

Example for the probabilistic multidimensional data model

To clarify the above definition of the probabilistic multidimensional data model, we will use the data cube example shown in figure 1. Note that multiple content components are written into a single box for convenience and separating them each into their individual cells does not affect as long as their address components are properly represented. This Sales cube represents the data collected by Koke company for the sales of competitor Bepsi's products. Since it is not possible to get the exact sales information, Koke's agents are allowed to report a guess based on empty cans being recycled and attach a probability measure to each report.

The sales cube has:

- The characteristic set $C = \{\text{TIME}, \text{PRODUCT}, \text{LOCATION}, \text{SALES}, \text{BELIEF}\}$, ($m = 5$)
- The attribute set $A = \{\text{day}, \text{week}, \text{month}, \text{year}, \text{product_name}, \text{size}, \text{store_name}, \text{city}, \text{state}, \text{region}, \text{amount}, \text{quantity}, \text{pS}\}$, ($t = 14$)
- schema of C:
 - $f(\text{TIME}) = \{\text{day}, \text{week}, \text{month}, \text{year}\}$
 - $f(\text{PRODUCT}) = \{\text{product_name}, \text{size}\}$
 - $f(\text{LOCATION}) = \{\text{store_name}, \text{city}, \text{state}, \text{region}\}$
 - $f(\text{SALES}) = \{\text{amount}, \text{quantity}\}$
 - $f(\text{BELIEF}) = \{\text{pS}\}$
- dimension function d:
 - $d(\text{TIME}) = 1$ i.e. TIME is a dimension
 - $d(\text{PRODUCT}) = 1$ i.e. PRODUCT is a dimension
 - $d(\text{LOCATION}) = 1$ i.e. LOCATION is a dimension
 - $d(\text{SALES}) = 0$ i.e. SALES is a measure
 - $d(\text{BELIEF}) = 0$ i.e. BELIEF is a measure
- A sample partial order on the Sales cube is as follows:
 - $O_{\text{TIME}} = \{\langle \text{day}, \text{week} \rangle, \langle \text{day}, \text{month} \rangle, \langle \text{day}, \text{year} \rangle, \langle \text{week}, \text{month} \rangle, \langle \text{month}, \text{year} \rangle\}$
 - $O_{\text{PRODUCT}} = \{\langle \text{product_name}, \text{size} \rangle\}$

- $O_{LOCATION} = \{(store_name, city), \langle city, state \rangle, \langle state, region \rangle\}$
 - $O_{SALES} = \{\}$
 - $O_{BELIEF} = \{\}$
 - An example of L is as follows:
 - Let us assume the following domains for the attributes
 - $A = \{year, product_name, city, amount, quantity, pS\}$
 - $dom\ year = \{1993, 1994, 1995\}$
 - $dom\ product_name = \{P1, P2, P3\}$
 - $dom\ city = \{Boston, Dallas, Seattle, Chicago\}$
 - $dom\ amount = \{0, 1, 2, \dots\}$
 - $dom\ quantity = \{0, 1, 2, \dots\}$
 - $dom\ pS = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$
 - With the above assumptions, an example set of cells is shown below.
 - $l = \langle l.AC, l.CC \rangle$
 - $l.AC = \langle 1993, P1, Boston \rangle$ corresponding to $\langle year, product_name, city \rangle$
 - $l.CC = \langle 100, 10, 0.5 \rangle$ corresponding to $\langle amount, quantity, pS \rangle$
- Therefore,
 $l = \langle \langle 1993, P1, Boston \rangle, \langle 100, 10, 0.5 \rangle \rangle$
 This cell represents the Probability of “Sales of P1 by Bepsi in Boston for 1993 are 10 shipments with a revenue of 100k” is 0.5.

Similarly, the values shown in box one of the figure 1 can be written as:

$$\{ \langle \langle 1993, P1, Boston \rangle, \langle 100, 10, 0.5 \rangle \rangle, \langle \langle 1993, P1, Boston \rangle, \langle 125, 10, 0.2 \rangle \rangle, \langle \langle 1993, P1, Boston \rangle, \langle 150, 15, 0.1 \rangle \rangle \}$$

The above definition for the multidimensional data model is very similar to the Thomas and Datta’s model. In the next section, we will provide an operational model for this data model.

Algebra for Probabilistic Multidimensional Data Model

Our algebra is extended from the algebra provided for data cubes by Thomas and Datta. Below, we use conventions and definitions from Dey and Sarkar’s Probabilistic Relational Model as well. We start with some basic definitions that are used throughout the rest of this section.

As we mentioned in the previous section, value-equivalent cells must be coalesced. There are two types of coalescence operations defined on the value-equivalent cube cells. Both these coalescence operators can be applied recursively on any number of value-equivalent cells.

coalescence-PLUS (\oplus): This operator is used in the definition of the projection operation and is defined on two value-equivalent cells x and y as:

$$z = x \oplus y \Leftrightarrow (x \cong y) \wedge (z \cong x) \wedge (z.CC(pS) = \min\{1, x.CC(pS) + y.CC(pS)\})$$

Intuitively, when two value-equivalent cells are combined using projection operation (i.e. we believe in both of the cells) their individual probability is summed together. If the result is greater than 1, then existence of the object is certain and is assigned a probability of 1. Recursive application is denoted as:

$$\bigoplus_{i=1}^m x_i = (\dots((x_1 \oplus x_2) \oplus x_3) \oplus \dots \oplus x_{m-1}) \oplus x_m$$

coalescence-MAX (\odot): This operator is used in the definition of the union operation and is defined on two value-equivalent cells x and y as:

$$z = x \odot y \Leftrightarrow (x \cong y) \wedge (z \cong x) \wedge (z.CC(pS) = \max\{x.CC(pS), y.CC(pS)\})$$

Intuitively, when two value-equivalent cells are combined using union operation (i.e. we believe in only one of the cells, the one with higher strength of belief), maximum probability of these cells is the probability for the result cell. Recursive application is denoted as:

$$\bigodot_{i=1}^m x_i = (((\dots(x_1 \odot x_2) \odot x_3) \dots \odot x_{m-1}) \odot x_m)$$

To denote coalescence performed on cells with value-equivalence defined over an attribute subset S , we write “ \oplus over S ” or “ \odot over S ”.

We will proceed to define our algebra operators after providing two more definitions.

Predicate P: A predicate is a well-formed formula in first-order predicate logic.

- An *atomic predicate* is a restriction on the domain of a single attribute or characteristic. e.g. ($year = 1994$)
- A *compound predicate* is a logical expression of atomic predicates. The logical operators are \wedge (and), \vee (or), \neg (not), \rightarrow (implies), and \leftrightarrow (equivalent to). It is of the form: $P = p_1 \langle op \rangle p_2 \langle op \rangle \dots \langle op \rangle p_n$. e.g. ($year = 1994$) \wedge ($(quantity < 15) \vee (amount > 100)$)

l satisfies P: l , an instance of L , with the structure $\langle address, content \rangle$ satisfies predicate P if and only if:

Case 1: if an element of l is a dimension, then $l.AC$ satisfies P , otherwise $l.CC$ satisfies P , if P is atomic and the truth-value is TRUE.

$$P(l) = TRUE \begin{cases} a \text{ in } P, a \in f(d_i), d_i \in D, P(l.AC[a]) = TRUE \\ OR \\ a \text{ in } P, a \in f(m_i), m_i \in M, P(l.CC[a]) = TRUE \end{cases}$$

e.g. Upper left most corner cell in the cube of figure 1 satisfies $P=(\text{year}=1993)$

Case 2: if P is a compound predicate, l satisfies P, when all the truth-values evaluated together with the connecting operators results in TRUE.

$$\forall p_i \in P, a \text{ in } p_i, p_i(l) = Q_i \quad P(l) = Q_1 \langle op \rangle Q_2 \langle op \rangle \dots \langle op \rangle Q_n$$

e.g. Upper left most corner cell in the cube of figure 1 satisfies

$$P=(\text{year}=1993) \wedge (\text{city}=\text{"Boston"}) \wedge (\text{product_name}=\text{"P1"})$$

We can also define a fuzzy membership function for BELIEF characteristic, which maps between natural language sentences and probability. For example, concepts like probably, likely, most likely, certainly, etc. can be mapped to the probability numbers between 0 and 1 using the fuzzy membership functions. These functions can also be used to map probability to all natural language words describing the uncertainty. The process of fuzzification and defuzzification is performed to convert human terminology of uncertainty into a crisp probability measure and vice-versa. Such a function is strictly a helper function and is not part of the model or algebra, as our model for uncertainty is based on probabilities. This function is generally applied for restricting the cells with a desired strength of belief to appear in the output. As an example, let us assume that the query is "Select most likely maximum sales from Sales cube". Let us also assume a fuzzy membership function defined for this cube maps fuzzy sets "certain", "most likely", "very likely", "likely", "unlikely", and "very unlikely" to crisp sets 1.00, 0.99-0.70, 0.75-0.55, 0.60-0.40, 0.45-0.25, and 0.30-0.00. Of course, these are graded memberships, so a formal definition of these fuzzy sets will elaborate the membership gradation very clearly using alpha-cuts, height, plimth and other properties for fuzzy sets. Using this mapping we determine that "most likely" is described with a strength of belief between 0.99 - 0.75. Therefore, our selection predicate can be formed to include "pS >= 0.75 and pS < 1.00". This selects cells with probability greater than 0.75. Among the resultant cells, we can select the cell with maximum quantity.

Similarly, we can combine the probability distributions for each object with probability distributions of other objects when calculating aggregate values and assign a probability distribution to the result. By doing this, instead of answering a query like "what is the mean sales for Chicago?" with a pointed answer like "25", we can answer using a confidence-interval statement like "The mean sales for Chicago is 95% certain to be between 23 and 27". A comprehensive treatment of the probabilistic data can be found in [11, 12, 13, 14].

These are some simple examples to demonstrate the power of probabilistic multi-dimensional data.

Now, we define algebra operations for the probabilistic data cube. Each operator is presented in the format used by Thomas and Datta, as follows: the operator name, symbol, a textual description, input, output, mathematical notation,

and a simple example of the operator. All the examples use the Sales cube shown in figure 1.

Restriction (Σ): The *restriction* operator restricts the values on one or more attributes based on specified conditions, where a given condition is in the form of a predicate. This is similar to the selection operator in relational algebra. Only cells that satisfy the predicate are retrieved into the result cube. If there are no cells that satisfy P, the result is an empty cube. Note that pS may also be restricted in the predicate, thus selecting cells representing only real world objects that have satisfied the belief constraints. This operator can be applied multiple times. The order of application is not significant.

The algebra of restriction operator is defined as follows:

Input: A cube $C_1 = \langle C, A, f, d, O, L \rangle$ and a predicate P

Output: A cube $C_0 = \langle C, A, f, d, O, L_0 \rangle$ where $L_0 \subseteq L$ and $L_0 = \{l \mid (l \in L) \wedge (l \text{ satisfies } P)\}$.

Mathematical Notation: $\Sigma_P(C_1) = C_0$

A Simple Example: If we want to know the sales for P1 in Boston during the year 1993, then we use $\Sigma_{(\text{year}=1993 \wedge \text{product_name}=\text{"P1"} \wedge \text{city}=\text{"Boston"})}(\text{Sales}) = C_{\text{Restrict}} =$

$$\{ \langle \langle 1993, P1, Boston \rangle, \langle 100, 10, 0.5 \rangle \rangle, \langle \langle 1993, P1, Boston \rangle, \langle 125, 10, 0.2 \rangle \rangle, \langle \langle 1993, P1, Boston \rangle, \langle 150, 15, 0.1 \rangle \rangle \}$$

Metric Projection (Π^M): The *metric projection* operator restricts the output of a cube to include only a subset of the original set of measures. This is similar to the projection operator in the relational algebra. Let S be a set of project metric attributes such that $S \subseteq A_m$. Then the output of the resulting cube includes only those measures in S. Since our cell represents a joint distribution of the attributes and this operation results in a subset of the original attributes, we need to marginalize the probabilities. We use the coalescence-PLUS (\oplus) operator for this. Note that the value-equivalence is over the set of attributes S and projecting out the pS itself (i.e. pS \notin S) may yield meaningless result.

The algebra of metric projection is defined as follows:

Input: A cube $C_1 = \langle C, A, f, d, O, L \rangle$ and a set of projection attributes S

Output: A cube $C_0 = \langle C, A_0, f_0, d, O, L_0 \rangle$ where,

$$A_0 = S \cup A_d,$$

$$f_0: C \rightarrow 2^{A_0} \mid f_0(c) = f(c) \cap A_0,$$

$$L_0 = \{l_0 \mid \exists l \in L, l_0.AC = l.AC,$$

$$l_0.CC = \langle l.CC[s_1], l.CC[s_2], \dots, l.CC[s_3] \rangle$$

$$\text{where } \{s_1, s_2, \dots, s_3\} = S \quad \text{and} \quad \oplus \text{ over } S_{l \in L}$$

Mathematical Notation: $\Pi_S^M(C_1) = C_0$

A Simple Example: If we are interested in selecting only the quantity from the previous Restriction operator output

above, we use $\Pi_{\text{quantity}}^M(C_{\text{Restrict}}) = C_{\text{Project}} =$

$$\{ \langle \langle 1993, P1, Boston \rangle, \langle 10, 0.7 \rangle \rangle, \langle \langle 1993, P1, Boston \rangle, \langle 15, 0.1 \rangle \rangle \}$$

Note that the result contains only one cell with coalesced pS for the quantity=10 because there are two cells for that and they become value-equivalent when amount is projected out. Also, note that eliminating pS through this operation for this example would result in cells with identical address components, but different quantities (with belief strength implicitly 1), which is meaningless data.

Rename (A): The rename operator renames a set of elements. It is similar to the rename operator in relational algebra. Let S_i be some set of elements $\{s_{i1}, s_{i2}, \dots, s_{in}\}$. Then, $A_S(S_i) = \{S.s_{i1}, S.s_{i2}, \dots, S.s_{in}\}$. This operator can be used to eliminate duplicate names in the results of binary operations. For example, Renaming the attributes corresponding to the TIME dimension of the Sales cube can be expressed as follows: $A_{Sales}(A) = \{Sales.day, Sales.week, Sales.month, Sales.year\}$

Cubic Product (\otimes): The *Cubic Product* operator is a binary operator. It is used to relate two cubes. This operator joins the attributes of two cubes together. This operator is similar to the Cartesian Product operator in the relational algebra. The probability of the result is obtained by multiplying the probabilities of joined cells. By noting that the probability of each cell is the joint probability for that set of attributes, we can see that the result set is union of both tuples and the result cell's probability must be a joint probability of all the attributes together. We can also see that the resulting probability is meaningful only when all the attributes are mutually independent. The Cubic Product is defined as follows:

Input: A cube $C_1 = \langle C_1, A_1, f_1, d_1, O_1, L_1 \rangle$ and A cube $C_2 = \langle C_2, A_2, f_2, d_2, O_2, L_2 \rangle$.

Output: A cube $C_0 = \langle C_0, A_0, F_0, d_0, O_0, L_0 \rangle$, where (\bullet denotes concatenation)

$$\begin{aligned} C_0 &= A_{C_1}(C_1) \cup A_{C_2}(C_2), \\ A_0 &= A_{A_1}(A_1) \cup A_{A_2}(A_2), \\ L_0 &= \{l_0 \mid \exists l_1, \exists l_2, l_1 \in L_1, l_2 \in L_2, l_0.AC = l_1.AC \\ &\bullet l_2.AC, \\ l_0.CC &= l_1.CC - \{l_1.CC(pS)\} \bullet l_2.CC - \{l_2.CC(pS)\}, \\ l_0.CC(pS) &= \{l_1.CC(pS) * l_2.CC(pS)\} \end{aligned}$$

In addition,

$$\begin{aligned} \forall c_i &\in (C_1 \cup C_2), \\ f_0 &= f_1 \text{ when applied to } c_i \in C_1.c_i, f_2 \text{ when applied} \\ &\text{to } c_j \in C_2.c_j, \\ \forall c_i &\in (C_1 \cup C_2), \\ d_0 &= d_1 \text{ when applied to } c_i \in C_1.c_i, d_2 \text{ when applied} \\ &\text{to } c_j \in C_2.c_j \\ \forall a_i &\in (f(C_1) \cup f(C_2)), \\ O_0 &= O_1 \text{ when applied to } a_i \in f(C_1), O_2 \text{ when} \\ &\text{applied to } a_j \in f(C_2) \end{aligned}$$

Mathematical Notation: $C_1 \otimes C_2 = C_0$

A Simple Example: Suppose we have another cube, Discount, containing discount amounts for various combinations of product and city. The definition of Discount cube is characteristics $C = \{\text{PRODUCT, LOCATION, DISCOUNT, BELIEF}\}$, attributes $A = \{\text{product_name, city_ID, amount, pS}\}$, dimensions $D =$

$\{\text{PRODUCT, LOCATION}\}$, measures $M = \{\text{DISCOUNT, BELIEF}\}$, $f(\text{PRODUCT}) = \{\text{product_name}\}$, $f(\text{LOCATION}) = \{\text{city_ID}\}$, $f(\text{DISCOUNT}) = \{\text{amount}\}$, and $f(\text{BELIEF}) = \{\text{pS}\}$. If we want to assess how knowing Discount amounts will change the probability of Sales amounts, we can first use Cubic Product operation to the Sales and Discount cubes as follows: $Sales \otimes Discount = C_{\text{result}}$. This will result in the superset of the desired information. By using the Restriction and Metric Projection, we can extract the required answers. The Cubic Product operation does not place any restrictions on the domains of the attributes.

Join (\odot): The *join* operator relates two cubes having one or more dimensions in common, and having identical mappings from common dimensions to the respective attribute sets of these dimensions. This operation can be expressed using Cubic Product operation. Therefore, this is not a basic operator in our algebra. The description of this operator is as follows: two cubes $C_1 = \langle C_1, A_2, f_1, d_1, O_1, L_1 \rangle$ and $C_2 = \langle C_2, A_2, f_2, d_2, O_2, L_2 \rangle$ are *join-compatible* if $D_1 \cap D_2 \neq \emptyset$, and $\forall c_i \in D_1 \cup D_2, f_1(c_i) = f_2(c_i)$. Furthermore, let $cd = D_1 \cap D_2 = \{cd_1, cd_2, \dots, cd_m\}$ and $A_{cd} = \{a_{cd1}, a_{cd2}, \dots, a_{cdm}\}$ denote the set of dimensions and corresponding dimensional attributes respectively. Hence, $A_{cd} = \cup_{\forall cd_i \in cd} f(cd_i)$ and $A_{cd} \subseteq A_d$. The algebra of join can be represented in terms of Cubic Product as: $C_1 \odot C_2 = \sum_P (C_1 \otimes C_2)$ where P is a predicate of the form $[(C_1.a_{cd1} = C_2.a_{cd1}) \wedge (C_1.a_{cd2} = C_2.a_{cd2}) \wedge \dots \wedge (C_1.a_{cdm} = C_2.a_{cdm})]$.

A Simple Example: Consider the query in the Cubic Product example. It can be answered by joining the Sales and Discount cubes as follows: $Sales \odot Discount = C_{\text{Result}}$.

Union-Compatible Cubes: Two cubes are *union-compatible* if they have the same structure. i.e. $C_1 = \langle C_1, A_1, f_1, d_1, O_1, L_1 \rangle$ and $C_2 = \langle C_2, A_2, f_2, d_2, O_2, L_2 \rangle$ are union-compatible if $C_1 = C_2, A_1 = A_2, f_1 = f_2, d_1 = d_2$, and $O_1 = O_2$.

Union (U): The *union* operator is a binary operator that finds the union of two union-compatible cubes. When union of two cubes is performed, value-equivalent cells must be coalesced using the coalescence-MAX (\odot) operator. This is because when we have two statements with varying degrees of belief, we pick the one with higher degree of belief (or more certain about). The algebra of the union operator is defined as follows:

Input: A cube $C_1 = \langle C_1, A_1, f_1, d_1, O_1, L_1 \rangle$ and another cube $C_2 = \langle C_2, A_2, f_2, d_2, O_2, L_2 \rangle$ which is union-compatible with C_1 .

Output: A cube $C_0 = \langle C_0, A_0, F_0, d_0, O_0, L_0 \rangle$, where $C_0 = C_1 = C_2, A_0 = A_1 = A_2, f_0 = f_1 = f_2, d_0 = d_1 = d_2, O_0 = O_1 = O_2;$

$$\begin{aligned} l \in L_0 &\Leftrightarrow \{ ((l \in L_1 \vee l \in L_2) \wedge ((\forall k \in L_1 - \{l\}, \neg(k \cong l)) \wedge \\ &(\forall j \in L_2 - \{l\}, \neg(j \cong l))) \\ &\vee ((j \in L_1) \wedge (k \in L_2) \wedge (l \cong j \cong k) \wedge (l = j \odot k)) \} \end{aligned}$$

Mathematical Notation: $C_1 \cup C_2 = C_0$

A Simple Example: Consider two cubes, Sales_South and Sales_North, both having the same cube structure. Suppose that Sales_South represents the sales in the southern region and Sales_North represents the sales in the northern region. We want to know the sales for the entire north-south region. Then we can accomplish this by using the union operator as follows: Sales_South \cup Sales_North = Sales_North_South. The value-equivalent cells (in this example, those belonging to both the regions reported to have different probability measures with all the other attributes being identical) being coalesced.

Belief Difference (θ): The *belief difference* operator is a binary operator that finds the difference of belief measures for two union-compatible cubes. This operator can be used to find how a reporter of information differs with another in terms of belief strengths for the same object. This operator is non-commutative. Suppose we have two cubes: Cube1 and Cube2. It is only possible to find how much more confidence is represented by Cube1 compared to Cube2. By reversing the operands it is possible to find how much more confidence is represented by Cube2 compared to Cube1. Repetitive application of this operator will result in finding the objects for which both cubes have the same confidence as well. When belief difference operation is performed, the probability of value-equivalent cells in the result is calculated to reflect the difference in strengths of belief. If the difference between the probabilities of two value-equivalent cells is positive, then the cell assumes the new probability in the result. If not, it is not included in the result. The algebra of the belief difference operator is defined as follows:

Input: A cube $C_1 = \langle C_1, A_1, f_1, d_1, O_1, L_1 \rangle$ and another cube $C_2 = \langle C_2, A_2, f_2, d_2, O_2, L_2 \rangle$ which is union-compatible with C_1 .

Output: A cube $C_0 = \langle C_0, A_0, F_0, d_0, O_0, L_0 \rangle$, where $C_0 = C_1 - C_2$; $A_0 = A_1 = A_2$; $f_0 = f_1 = f_2$; $d_0 = d_1 = d_2$; $O_0 = O_1 = O_2$;

$$l \in L_0 \Leftrightarrow ((j \in L_1) \wedge (k \in L_2) \wedge (l \cong j \cong k) \wedge (j.CC(pS) > k.CC(pS)) \wedge (l.CC(pS) = j.CC(pS) - k.CC(pS)))$$

Mathematical Notation: $C_1 \theta C_2 = C_0$

A Simple Example: Consider two cubes, Sales_Report_By_John and Sales_Report_By_Jill, both having the same cube structure, representing the competitor's sales as reported by John and Jill respectively. We want to know how they differ in their beliefs for the competitor's sales. We can accomplish this by using the belief difference operator as follows: Sales_Report_By_John θ Sales_Report_By_Jill = Difference_Btwn_John_Jill. We also note that we can find the difference of belief strengths only when there are value-equivalent cells.

Cubic Difference ($-$): The *cubic difference* operator is a binary operator that finds the difference of two union-

compatible cubes ignoring the probability measures. When cubic difference operation is performed, the value-equivalent cells with second cube are eliminated from the first cube. The algebra of the cubic difference operator is defined as follows:

Input: A cube $C_1 = \langle C_1, A_1, f_1, d_1, O_1, L_1 \rangle$ and another cube $C_2 = \langle C_2, A_2, f_2, d_2, O_2, L_2 \rangle$ which is union-compatible with C_1 .

Output: A cube $C_0 = \langle C_0, A_0, F_0, d_0, O_0, L_0 \rangle$, where $C_0 = C_1 - C_2$; $A_0 = A_1 = A_2$; $f_0 = f_1 = f_2$; $d_0 = d_1 = d_2$; $O_0 = O_1 = O_2$; $l \in L_0 \Leftrightarrow \{ ((l \in L_1) \wedge (\forall_j \in L_2, -(j \cong l))) \}$

Mathematical Notation: $C_1 - C_2 = C_0$

A Simple Example: Consider two cubes, Sales_South and Sales_Dallas, both having the same cube structure. Suppose that Sales_South represents the sales in the southern region and Sales_Dallas represents the sales in the Dallas city. We want to know the sales for the entire southern region except the Dallas city. Then we can accomplish this by using the cubic difference operator as follows: Sales_South $-$ Sales_Dallas = Sales_South_without_Dallas.

The cubic intersection operator can be defined using the cubic difference operator. It is expressed as $C_1 - (C_1 - C_2) = C_0$. Intersection is not a fundamental operator since it can be expressed in terms of other operators.

The cubic difference and belief difference operators can be used to find several interesting features of the data such as regions where different agents reported different belief strengths. They also can be used to sanitize the data where conflicting reports are not allowed. By judiciously applying Cubic Difference and Belief Difference operators in combination with other cubic operators defined earlier, it is possible to find the difference in strengths of beliefs for cubes that are union-compatible within a subset of their characteristics.

Aggregation (J): The *aggregation* operator performs aggregation (MIN, MAX, SUM, AVG, COUNT, RANK, PERCENTILE, etc.) on one or more dimensional attributes. This operator in combination with fuzzy membership functions defined for pS can be used to answer queries such as "what is the most likely average sales?", "what is confidence level for average sales to be 25?", etc. The queries that do not contain uncertainty in their formulation may have an answer with uncertainty in it. For example, "what are the maximum sales?" can be answered by selecting the cell with maximum sales reported which may have strength of belief of 0.01 (or very unlikely).

Let μ be a metric attribute to aggregate where $\mu \in A_m$ and G be a set of grouping attributes such that $G \in A_d$. Let F be an aggregate function having the mapping

$$F \ 2^{\prod_{vg} \in G^{dom_{agg}}} \rightarrow dom_{agg},$$

where *agg* represents an user-specified attribute name given to the result, which is extracted from the domain *dom agg*.

F is assumed to be a first-order definable function including

the standard arithmetic functions + (addition), - (subtraction), * (multiplication), and / (division), the standard SQL aggregate functions, and a RANK function. The RANK function takes a group of cells as input and returns an attribute agg corresponding to the ordinal number of the cell. The aggregation operator is defined as follows:
Input: A cube $C_1 = \langle C, A, f, d, O, L \rangle$, a set of grouping attributes G , a metric attribute μ , and an aggregate function F .

Output: A cube $C_0 = \langle C_0, A_0, F_0, d_0, O_0, L_0 \rangle$,
 Where

$$c_0 = \{c \mid c \in C, \exists x, x \in f(c) \cup \{AGG\} \text{ and } \{AGG\} \text{ is a new characteristic name defined specifically for the aggregated metrics, } A_0 = G \cup \{agg\} \text{ and } \{agg\} \text{ represents the computed aggregate attribute, } L_0 = \{l \mid \exists l \in L, l_0.AC = \langle l.AC[g_1], l.AC[g_2], \dots, l.AC[g_n] \rangle, l_0.CC = \langle l.CC[agg] \rangle\},$$

$$f_0 = \begin{cases} \{(x, y) \mid x \in C_0, (\exists(x, z) \in f, x \neq \{AGG\}, y = \{a \mid a \in (z \cap A_0)\})\} \\ \vee (\exists(x, z) \in f, x = \{AGG\}, y = \{a \mid a \in (z \cap A_0) \cup \{agg\}\}) \} \text{ if } \exists(\{AGG\}, z) \in f, \\ \{(x, y) \mid x \in C_0, (\exists(x, z) \in f, y = \{a \mid a \in (z \cap A_0)\})\} \\ \cup \{(\{AGG\}, \{agg\})\} \end{cases} \quad \text{otherwise}$$

$$\forall x \in C, d_0(x) = \begin{cases} d(x) & \text{if } \{AGG\} \in C \\ d(x) \cup \langle AGG, 0 \rangle & \text{otherwise} \end{cases}$$

Mathematical Notation: $\Gamma_{F,G,\mu}(C_1) = C_0$.

A Simple Example: Consider the Sales cube. Suppose the user wants to see the total annual sales for each product. Then, $F = \text{SUM}$, $G = \{\text{product_name, year}\}$, and $\mu = \text{amount}$. Therefore, the query to get the total annual sales for each product is written as $I[\text{SUM}, \{\text{product_name, year}\}, \text{amount}] (\text{Sales}) = C_{\text{Result}}$.

Note that prior to applying the aggregation operator, the Sales cube can be operated upon by various operations depending on what strengths of probabilities need to remain. If we want annual sales corresponding to the highest confidence level objects, then we first apply $\text{MAX}(pS)$ aggregation operator to the Sales cube and then apply aggregation operator for SUM . Applying aggregation operators without first applying a meaningful transformation on the pS may result in meaningless data. In this example, if we apply SUM on all the objects, then the result contains sum of several amounts for the same PRODUCT , TIME , and LOCATION , which is not a meaningful total. Instead, we could have applied a transformation that picks object with maximum pS or a more complex operation that combines all objects with differing pS but have the address component to obtain a probability distribution for that object and then selects an attribute value with say 95% confidence level. We can even ask for object attribute value that lies

between σ and $-\sigma$ thus utilizing all the concepts of statistical distributions.

Furthermore, Force and Extract operations defined for Thomas and Datta's model are defined for our model in identical manner. However, applying these operations on pS result in pS losing its special meaning and BELIEF becoming a regular characteristic.

Force (ψ): The force operator converts dimensions to measures. Let a_i be a dimensional attribute to transform such that $g(a_i) \in D$. Let c_i be the corresponding characteristic name for a_i such that $c_i \notin D$ and either $c_i \in M$ or c_i is a new characteristic name. The force operator is defined as follows:

Input: A cube $C_1 = \langle C, A, f, d, O, L \rangle$, a dimensional attribute to transform a_i , and a corresponding characteristic name c_i .
Output: A cube $C_0 = \langle C_0, A_0, F_0, d_0, O_0, L_0 \rangle$
 Where

$$C_0 = C \cup \{c_i\}, \\ f_0 = f - f(g(a_i)) + [g(a_i) \rightarrow f(g(a_i) - a_i)] + [c_i \rightarrow a_i], \\ O_0 = O_{\text{prev}} \cup O_{\text{new}} \text{ where } O_{\text{prev}} \text{ is obtained by removing ordered pairs containing } a_i \text{ from } O \text{ and } O_{\text{new}} \text{ represents a user specified set of ordering relations between } a_i \text{ and the elements of } f(c_i) \text{ if } c_i \in M, \\ L_0 = \{l_0 \mid \exists l \in L, l_0.AC = l.AC - \langle l.AC[a_i] \rangle, l_0.CC = l.CC \bullet \langle l.AC[a_i] \rangle\},$$

$$\forall c_i \in C, d_0(c_i) = \begin{cases} d(c_i) & \text{if } c_i \neq c_i, \\ 0 & \text{otherwise} \end{cases}$$

Mathematical Notation: $\Psi_{a_i, c_i}(C_1) = C_0$

A Simple Example: Converting *store_name* from dimension to a measure in the Sales cube. This can be expressed as:
 $\Psi_{\text{store_name, sales}}(\text{Sales}) = C_{\text{Result}}$

Extract (Φ): The extract operator converts measures to dimensions. Since we assigned a special meaning for the BELIEF characteristic and made it a measure, it cannot be extracted to a dimension without losing its special meaning. Even forcing it back to a measure after extracting pS may not restore its meaning after certain operations. Let a_i be a metric attribute to transform such that $g(a_i) \in M$. Let c_i be the corresponding characteristic name for a_i such that $c_i \notin M$ and either $c_i \in D$ or c_i is a new characteristic name. The extract operator is defined as follows:

Input: A cube $C_1 = \langle C, A, f, d, O, L \rangle$, a metric attribute to transform a_i , and a corresponding characteristic name c_i .
Output: A cube $C_0 = \langle C_0, A_0, F_0, d_0, O_0, L_0 \rangle$
 Where

$$C_0 = C \cup \{c_i\}, \\ f_0 = f - f(g(a_i)) + [g(a_i) \rightarrow f(g(a_i) - a_i)] + [c_i \rightarrow a_i], \\ O_0 = O_{\text{prev}} \cup O_{\text{new}} \text{ where } O_{\text{prev}} \text{ is obtained by removing ordered pairs containing } a_i \text{ from } O \text{ and } O_{\text{new}} \text{ represents a user specified set of ordering}$$

relations between a_i and the elements of $f(c_i)$ if $c_i \in D$,

$$L_O = \{l_o \mid \exists l \in L, l_o.AC = l.AC \bullet (l.AC[a_i]), \\ l_o.CC = l.CC - (l.AC[a_i])\},$$

$$\forall c_i \in C, d_o(c_i) = \begin{cases} d(c_i) & \text{if } c_i \neq c_i, \\ 0 & \text{otherwise} \end{cases}$$

Mathematical Notation: $\Phi_{at,ct}(C_i) = C_O$

A Simple Example: Converting store_name from a measure to a dimension in the Sales cube. This can be expressed as:

$$\Phi_{store_name, sales}(Sales) = C_{Result}$$

Properties of the Model

Our probabilistic multidimensional data model is very similar to Thomas and Datta's model and preserves all the properties of that model. When BELIEF is not a characteristic of the cube, this model is reduced to the deterministic model of Thomas and Datta. In this section, we prove that the algebra defined for our model is closed, at least as expressive as relational model, and relationally complete. We start by showing that our data model is reducible to relational model and content-wise equivalent.

Data Equivalence ($\cong_{\mathfrak{R}}$): A relation instance (a row in a table) is a set of n-tuples. Each tuple can correspond to an individual cell in the cube (in fact, our example of cells in the Sales cube are shown as tuples of dimensional attributes and metric attributes). When there are no tuples in the relational instance it is equivalent to an empty cube. A formal definition of Data Equivalence between relational instance r and a cube C is as follows:

An instance r of a relation R and cube C are data equivalent, denoted $r \cong_{\mathfrak{R}} c$,

iff $C = \langle C, A, f, d, O, L \rangle$ such that

$C = \{M\}$, where M is an arbitrary characteristic

$A = R$, i.e., the relation and the cube have the same set of attributes

$f = \{\{M, A\}\}$, i.e., f maps all attributes to the arbitrary characteristic M

$d = \{\{M, 0\}\}$, i.e., characteristic M is a measure

$O = \emptyset$, i.e., no partial ordering is present

$L = \{l \mid \exists t \in r, l.CC = t, l.AC = \emptyset\}$, i.e., for every tuple t in r , there exists a single cell l in C which has the tuple as its content component and no address component.

THEOREM 1: *Our algebra is closed.*

To prove this theorem, we must show that all the basic operations defined in our algebra result in Cube as defined in the model. The Cube must satisfy the following three criteria: (1) the values of cells must come from an appropriate domain, (2) not two cells in result cube are value-equivalent, and (3) The result cube is finite collection

of cells. Considering the definitions of operators, every basic operator produces a result Cube. The domain of cells other than pS are defined to be the same as those in the original cube. The pS has the domain of $(0, 1]$. To prove that domain of pS will be $(0, 1]$ after the application of basic operators, we examine each operator except Cubic Difference and find they all result in pS greater than zero. Looking at the coalescence operators which are part of Union, Metric Projection, etc. explicitly prevent the value of pS to be greater than 1. Therefore, we conclude that (1) is satisfied. Second criteria is trivially proved by noting that coalescence operators always coalesce value-equivalent cells and produce a single cell. The coalescence operators always produce the same number cells as there are in the input or less. We also can see that the number of cells in the result can be at most $|C_1| \times |C_2|$ for Cubic Product operation $C_1 \otimes C_2$. All the other operators result in less number of cells than Cubic Product. Since an empty cube also satisfy the definition, the operators resulting in empty cube are still closed. This algebra is a consistent extension of algebra defined for Thomas and Datta's model.

THEOREM 2: *Our algebra is at least as expressive as the relational algebra.*

We show that all five basic relational algebra operators (Restriction, Projection, Union, Difference, Product) can be expressed in our algebra. Consequently, it follows that other derived operators can be expressed as well.

Restriction: Given a relation instance r and a cube C such that $r \cong_{\mathfrak{R}} C$. Suppose we have a selection predicate P . Since $\sigma_P(r)$ returns relation instance r' containing tuples of r that satisfy P and $\Sigma_P(C)$ returns cube C' containing cells of C that satisfy P , we conclude that $\sigma_P(r) \cong_{\mathfrak{R}} \Sigma_P(C)$.

This line of argument can also be used to substantiate the claims below.

Metric Projection: Given relation instance r and a cube C such that $r \cong_{\mathfrak{R}} C$,

$$\pi_S(r) \cong_{\mathfrak{R}} \prod_S^M(C)$$

Union: Given relation instances r_1 and r_2 and cubes C_1 and C_2 such that $r_1 \cong_{\mathfrak{R}} C_1$ and $r_2 \cong_{\mathfrak{R}} C_2$, $r_1 \cup r_2 \cong_{\mathfrak{R}} C_1 \cup C_2$.

Difference: Given relation instances r_1 and r_2 and cubes C_1 and C_2 such that $r_1 \cong_{\mathfrak{R}} C_1$ and $r_2 \cong_{\mathfrak{R}} C_2$, $r_1 - r_2 \cong_{\mathfrak{R}} C_1 - C_2$.

Cubic Product: Given relation instances r_1 and r_2 and cubes C_1 and C_2 such that $r_1 \cong_{\mathfrak{R}} C_1$ and $r_2 \cong_{\mathfrak{R}} C_2$, $r_1 \times r_2 \cong_{\mathfrak{R}} C_1 \otimes C_2$. This holds since (a) $r_1 \times r_2$ returns a relation r having $|r_1| \times |r_2|$ number of tuples representing all possible combinations of both relational instances and (b) $C_1 \otimes C_2$ returns a cube C having all characteristics and attributes of both C_1 and C_2 and cells representing all possible combinations of the cells of both cubes.

By showing that every relational algebra operator can be expressed in our algebra, we conclude that our algebra is at least as expressive as relational algebra and possibly more expressive since we can perform several additional operations in our algebra. Intuitively our algebra is relationally complete.

Further Research

This research can be extended in several directions. The modification of uncertainty measures when new information arrives can be defined. Implementation issues and performance analysis (space and time complexities) can be researched. These topics are very big in scope and hence not tackled in this research.

Conclusions

In this research report we defined a probabilistic multidimensional data model and associated algebra with several basic operations. All the operators are used in simple examples. Areas for further research are identified.

References

1. Ralph Kimball, Laura Reeves, Margy Ross, and Warren Thornthwaite, "The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses", John Wiley & Sons, Inc. 1998
2. Codd, E.F., Codd, S.B., and Sally, C.T., "Providing OLAP to User-Analysts: An IT Mandate", E.F. Codd & Associates (<http://www.arborsoft.com/papers/coddTOC.html>)
3. Helen Thomas and Anindya Datta, "A Conceptual Model and Algebra for On-Line Analytical Processing in Decision Support Databases", Information Systems Research, Vol. 12, No. 1, March 2001, pp. 83-102 (<http://137.52.54.32/Faculty/sumitra/articlelist/thomas2001.pdf>).
4. George J. Klir, and Mark J. Wierman, "Uncertainty-Based Information – Elements of Generalized Information Theory", Physica-Verlag Heidelberg 1998
5. Panos Vassiliadis, and Timos Sellis, "A Survey On Logical Models for OLAP Databases", National Technical University of Athens, Dept. Electrical and Computer Engineering, Computer Science Division, Knowledge and Database Systems Laboratory, Zografou 15773, Athens, Greece
6. Li, C., X. S. Wang. 1996. A data model for supporting on-line analytical processing. Proc. Conf. Inform. Knowledge Management, Baltimore, MD, USA.
7. Agarwal, R., A. Gupta, S. Saragawi, "Modeling multidimensional databases", Proc. 13th ICDE Conference, IEEE, Birmingham, U.K., 1997
8. Gyssens, M., and L.V.S. Lakshmanan, "A foundation for multidimensional databases", Proc. 23rd VLDB Conf., Athens, Greece, 1997
9. Debabrata Dey and Sumit Sarkar, "A probabilistic relational model and algebra", ACM Trans. Database Systems, 4 (4), Year 1996, pp. 397-434
10. Debabrata Dey and Sumit Sarkar, "Modifications of Uncertain Data: A Bayesian Framework for Belief Revision", Information Systems Research, Vol. 11, No. 1, March 2000, pp. 1-16 (<http://137.52.54.32/Faculty/sumitra/articlelist/baysianbelief.pdf>)
11. Amihai Motro and Philippe Smets, "Uncertainty Management in Information Systems: From needs to solutions", Kluwer Academic Publishers 1997
12. Zhengxin Chen, "Data Mining and Uncertain Reasoning: An Integrated Approach", John Wiley & Sons, Inc. 2001
13. Judea Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference", Morgan Kaufman Publishers 1988
14. Judea Pearl, "Causality: Models, Reasoning, and Inference", Cambridge University Press 2000