ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

Sum Query Evaluation over Uncertain Data

Roslinmary M[1], Saravana Kumar T[2] and Addlin Shinney R[3]

M.TECH / IT, Dr.Sivanthi Aditanar College of Engineering, Tiruchendur-628215, Tamilnadu, India[1,3]

Asst. Prof / IT, Dr.Sivanthi Aditanar College of Engineering, Tiruchendur-628215, Tamilnadu, India[2]

ABSTRACT*:* One of the important Queries in Many real time applications is SUM query, it deals with unpredictable data. In this paper, dealing with the query, called ALL_SUM Query. In general, the SUM query returns only the sum of the values. But the ALL_SUM Query returns all possible sum values together with their probabilities. There is no efficient solution for the problem of evaluating ALL_SUM queries used in many applications where the aggregate attribute values are real with small precision. In this paper, evaluating a pseudo - polynomial algorithm called DPSUM algorithm which is based on a recursive approach, it efficiently calculate ALL_SUM Query. The proposed DPSUM algorithm returns an efficient solution for determining the exact result of ALL_SUM queries. The results of an experimental evaluation over synthetic and real-world data sets show its effectiveness.

KEYWORDS*:* query processing, Database management systems

I. INTRODUCTION

Uncertain Database is "Membership of an item to the database" is a probabilistic event or the value of attributes is a probabilistic variable. Uncertain data streams are important in growing number of environments, such as traditional sensor networks, RFID networks for object tracking, radar networks for severe weather monitoring and etc.Aggregate query, in particular SUM queries, are essential for many applications that deal with uncertain data. Let us give a motivating example from the medical applications domain.

Patient	Sensor	Blood press.	Required human resources	Probability
P1	$S_{1,1}$	16	3	0.5
P2	$S_{1,2}$	13	1	0.4
P3	$S_{2,1}$	12	0	1

Table.1: Motivating Example

Example: E-health Management System. Consider a medical canter that monitors key biological parameters of remote patients at their homes, using sensors in their bodies. The sensors periodically send to the canter the patients health data, *e.g.* blood pressure, hydration levels, thermal signals, etc. For high availability, there are two or more sensors for each biological parameter. However, the data sent by sensors may be uncertain, and the sensors that monitor the same parameter may send inconsistent values as shown in table.1. There are approaches to estimate a confidence value for the data sent by each sensor, *e.g.* based on their precision. According to the data sent by the sensors, the medical application computes the number of required human resources, e.g. nurses, and equipments for each patient. One important query in this application is "return the sum of required nurses". Table.1 shows an example table of this application. The table shows the number of required nurses for each patient.

The table.2 shows the possible worlds, i.e., the possible database instances, their probabilities, and the result of the SUM query in each world. In this example, there are eight possible worlds and four possible sum values, i.e., 0 to 3.

Table.2 The possible worlds and the results of SUM query in each instances.

Database instances	Probability	Required resources
DI_1={}	0.28	0
DI_2={t1}	0.42	2
DI_3={t2}	0.12	2
DI_4={t1.t2}	0.18	4

A Navie algorithm for evaluating ALL_SUM queries is to return all possible worlds, i.e. .all possible database instances, compute sum in each world, and return the possible sum values and their probability. However, Search space can be exponential - Up to 2n answers for SUM queries, where n is the number of uncertain items.In this demonstration, we present a probabilistic database system for managing uncertain data.Our demonstration application is the E- health Management system application described in above Example.

The rest of the paper is organized as follows. In Section 2, we present some technical basis e.g. the probabilistic data models and some intuitions about our algorithms. In Section 3, we describe our Algorithm. In Section 4 we present performance evaluation of my paper. Section 5 concludes.

II. TECHNICAL BASIS

In this section we introduce the probabilistic data models that we consider. Main objective for using Probabilistic database is to extend data management tools to handle probabilistic data. Then, we define the problem that we address.

2.1 Probabilistic Models

we first introduce the two probabilistic data models that most frequently used in our community.

Tuple-level model:

In this model, each uncertain table T has an attribute that indicates the membership probability (also called existence probability) of each Tuple in T, i.e., the probability that the Tuple appears in a possible world. In this paper, the membership probability of a Tuple ti is denoted by p(ti). Thus, the probability that ti does not appear in a random possible world is 1-p(ti). The database shown in Table.3 is under Tuple-level model.

Table.3 Tuple level model

Tuple	**Probability**
t1	p1
t2	p2
……	……

Attribute-level model:

In this model, each tuple ti has at least one uncertain attributes, e.g. α, and the value of α in ti is chosen by a random variable X. assume that X has a discrete probability density function(pdf). This is a realistic assumption for many applications e.g., sensor readings.

Table.4 attribute level model

Time	temp low	temp high
1	20	21
2	22	23
3	18	19

2.2 Problem Definition

ALL_SUM Query:

It returns all possible sum results together with their probability. Let us now formally define ALL_SUM queries. Let D be a given uncertain database, W the set of its possible worlds, and P(w) the probability of a possible world w ∈ W. Let Q be a given aggr query, f the aggr function stated in Q (i.e., SUM), f(w) the result of executing Q in a world w ∈ W, and $V_{D,f}$ the set of all possible results of executing Q over D, i.e., $V_{D,f} = \{ v / \exists w \in W \wedge f(w) = v \}$. The cumulative probability of having a value v as the result of Q over D, denoted as P (v, Q , D), is computed as follows:

$$P(v, Q, D) = \sum_{w \in W \text{ and } f(w)=v} p(w).$$

My objective in this paper is to return the results of ALL_SUM as follows:

$$\text{ALL\_SUM}(Q, D) = \{(v, p) / v \in V_{D,f} \wedge p = P(v,Q,D)\}.$$

III. Q_PSUM AND DPSUM ALGORITHM

In this section, proposed an efficient solution for evaluating ALL_SUM queries. Here first propose a new recursive approach for computing the results of ALL_SUM. Then, using the recursive approach propose the Q_PSUM algorithm and DPSUM algorithm. Here use that the database is under the tuple-level model and attribute-level model.

3.1 ALL_SUM Under Tuple - Level Model

In this section propose an efficient solution for evaluating ALL_SUM queries. first propose a new recursive approach for computing the results of ALL_SUM. Then, using the recursive approach propose an Q_PSUM algorithm. Assume that the database is under the tuple-level model defined in the previous section.

3.1.1 Recursive Approach

First develop a recursive approach that produces the results of ALL_SUM queries in a database with n tuples based on the results in a database with n - 1 tuples. The principle behind it is that the possible worlds of the database with n tuples can be constructed by adding/not adding the nth tuple to the possible worlds of the database with n -1 tuples. Let DB^n be a database involving the tuples t1, . . . , tn, and ps(i,n) be the probability of having sum = i in DB^n, then develop a recursive approach for computing ps(i; n).

Recursive Algorithm ps(i, n)

Input:
n: number of tuples ;
$t_{1...}t_{n:}$ the tuples of the database;
p (t) : a function that returns the probability of the tuple t;
Output:
Probability of sum = 1 in a database containing $t_1 \ldots t_n$;

Algorithm

```
Begin
If (n>1) then
Return PS (i, n-1) × (1-p(t_n)) + PS(i – val(t_n), n-1 ) × p(t_n);
Else If ((n=1) and (val (t_1) ≠ 0))then
Return p(t_1);
Else If ((n=1) and (i=0) and (val (t_1)≠0)) then
Return 1-p (t_1);
Else If ((n=1) and (i=val (t_1) =0)) then
Return 1;
Else return 0;
End;
```

3.1.2 DPSUM Algorithm

In this section, using the dynamic programming technique, propose an efficient algorithm, called DPSUM, designed for the applications where aggr values are integer or real numbers with small precisions. It is usually much more efficient than the Q_PSUM algorithm. Let MaxSum be the maximum possible sum value, e.g., for positive aggr values MaxSum is the sum of all values. DPSUM uses a 2D matrix, say PS, with (MaxSum + 1) rows and n columns. DPSUM is executed on PS, and when it ends, each entry PS[i, j] contains the probability of sum = i in a database involving tuples t_1 . . . . .t_j..

DPSUM proceeds in two steps as follows

1. In the first step, it initializes the first column of the matrix This column represents the probability of sum values for a database involving only the tuple t_1. DPSUM initializes this column using the base of recursive formula as follows:

if val(t_1) = 0, then PS[0, 1] = 1, Otherwise,
PS[0, 1] = (1 – p(t_1)) and PS[val(t_1), 1] = p(t_1).

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

The other entries of the first column should be set to zero, i.e.,

$$PS[i, 1] = 0 \text{ if } i \neq 0 \text{ and } i \neq val(t_1).$$

2. In the second step, in a loop, DPSUM sets the values of each column j (for j = 2 to n) by using recursive definition (i.e., (4)) and based on the values in column j -1 as follows:

$$PS[i; j] = PS[i, j-1] \times (1 - p(t_j)) + ps[i - val(t_j), j-1] \times p(t_j).$$

Notice that if (i < val(t_j)), then for the positive aggr values have PS[i – val(t_j), j – 1] = 0, i.e., because there is no possible sum value lower than zero. This is why, in the algorithm only if (i ≥ val(t_j)), consider PS[i –val(t_j) j – 1] × $p(t_j)$for computing PS[i, j].

Let us now discuss the complexity of DPSUM. The first step of DPSUM is done in O(MaxSum), and its second step in O(n×MaxSum). Thus, the time complexity of DPSUM is O(n × MaxSum), where n is the number of uncertain tuples and MaxSum the sum of the aggr values of all tuples. Let avg be the average value of aggr values of tuples, then have MaxSum = n × avg. Thus, the complexity of DPSUM is O(n^2 × avg) where avg is the average of the aggr values in the database.

Algorithm : DP_PSUM()

Input:
n : number of tuples;
t1, …, tn : the tuples of the database;
MaxSum : maximum possible sum;
p(t) : a function that returns the probability of tuple t;
Output:
Possible sum values and their probability;
Algorithm

```
Begin
Let PS [0..MaxSum, 1..n] be a 2 dimensional matrix;
//Step 1 : initialization
For i=1 to MaxSum do
PS[i, 1] = 0;
If (val(t1) = 0) Then
PS[0, 1] = 1;
Else begin
PS[0, 1] = 1 - p(t1);
PS[val(t1), 1] = p(t1) ;
End ;
//Step 2 : filling the columns
For j=2 to n do
For i=0 to MaxSum do begin
PS[i, j] = PS[i, j-1] × (1 – p(tj))
If ( i – val(tj) ≥ 0) then
If ( PS[i – val(tj), j - 1] > 0) then
PS[i, j] = PS[i, j] + PS[i – val(tj), j - 1]×p(tj);
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

```
End;
// returning the results to the user
For i=0 to MaxSum do
If (PS[i, n] $\neq$ 0) then
Return i, PS[i, n];
 End;
```

3.2 ALL_SUM Under Attribute-Level Model

Due to the significant differences between the tuple-level and the attribute-level models it is not trivial to adapt yet proposed algorithms for the attribute-level model. In this section, proposed solution for computing ALL_SUM results under the attribute-level model.

Recursion Step

Now consider DB^{n-1}, i.e., a database involving the tuples $t_1 \ldots .t_{n-1}$. Let W^{n-1} be the set of possible worlds for DB^{n-1}. Let ps(i, n + 1) be the probability of having sum = i in DB^{n-1}, i.e., the aggregated probability of the DB^{n-1} worlds in which have sum = i. Let $v_{n,1}, \ldots v_{n,m}$ be the possible values of t_n's aggr attribute, and $p_{n,1} \ldots p_{n,m}$ their probabilities. construct DB^n by adding t_n to DB^{n-1}. The worlds of DB^n are constructed by adding to each w ∪by adding the possible value $v_{n,k}$ of t_n to the worlds of W^{n-1}. Indeed, for each world $w \in W^{n-1}$ there is a corresponding world $w' \in W^{nk}$ such that $w' = w + \{t_n\}$ where the possible aggr value of t_n is equal to $v_{n,k}$. This implies that for each possible sum = i with probability p in W^{n-1}, there is a possible sum = i + $v_{n,k}$ with probability p × $p_{n,k}$ in w^n_k. Recall that ps(i, n – 1) is the aggregate probability of the DB^{n-1} worlds in which sum = i. Then have Probability of sum = i in

$$w^n_k = ps(i - v_{n,k}, n-1) \times p_{n,k} \text{ for } k = 1, \ldots, m. \qquad (1)$$

Let ps(i.n) be the probability of sum = i in DB^n. Since have $W^n = W^n_1 \cup W^n_2 \cup \ldots \cup W^n_m$, the probability of sum = i in W^n is equal to the sum of the probabilities of sum = i in $W^n_1, W^n_2, \ldots, and\ W^n_m$. Thus, using (1) and the recursion base of , the probability of sum = i in DB^n, i.e., ps(i, n),can be stated as follows: ps(i, n) ,

$$ps(i,n) = \begin{cases} \sum_{k=1}^{m} ps(i = v_{n,k},\ n-1) \times p_{n,k}, & if\ n > 1, \\ p_{l,k}, & if\ n = 1\ and\ \exists\ k\ such\ that\ v_{l,k} = i, \\ 0 & otherwise \end{cases}$$

DPSUM2 Algorithm

Now, describe a dynamic programming algorithm, called DPSUM2, for computing ALL_SUM under the attribute-level model. Here, similar to the basic version of DPSUM algorithm, assume integer values. However, in a way similar to that of DPSUM, this assumption can be relaxed. Let PS be a 2D matrix with (MaxSum + 1) rows and n columns, where n is the number of tuples and MaxSum is the maximum possible sum, i.e., the sum of the greatest values of aggr attribute in the tuples. At the end of DPSUM2 execution, PS[i ,j] contains the probability of sum = i in a database involving tuples $t_i, \ldots .t_j$.

DPSUM2 works in two steps.

1. In the first step, it initializes the first column of the matrix, by using the base of the recursive definition, as follows: for each possible aggr value of tuple t_1, e.g., $v_{l,k}$, it sets the corresponding entry equal to the probability of the possible value, i.e., it sets $P[v_{l,k}, 1] = P_{l,k}$ for $1 <= k <= m$.
2. In the second step, DPSUM2 sets the entry values of each column j (for $2 \leq j \leq n$) by using the recursion step of the recursive definition, and based on the values yet set in precedent column. Formally, for each column j and row i it sets $PS[i, j] = \sum Ps([i - v1, k , j - 1] \times p_{j,k}$ such that $i \geq v_{1,k}$.

Let us now analyze the complexity of the algorithm. Let avg = MaxSum/n, i.e., the average of the aggregate attribute values. The space complexity of DPSUM2 is exactly the same as that of DPSUM, i.e., O (n2 × avg). The time complexity of DPSUM2 is O(MaxSum × n × m). In other words its time complexity is O (m × n × avg).

IV. CONCLUSION

One of the important Query in Many real time applications is SUM query, it deals with unpredictable data. In this paper, dealing with the query, called ALL_SUM Query. In this paper, evaluating a pseudo - polynomial algorithm called DPSUM algorithm which is based on a recursive approach, it efficiently calculate ALL_SUM Query. It returns exact result of ALL_SUM queries. The results of an experimental evaluation over synthetic and real-world data sets show its effectiveness of our solution. The performance of DPSUM is better than Navie Algorithm.

REFERENCES

[1] T.S. Jay ram, A. McGregor, S. Muthukrishnan, and E. Vee, "Estimating Statistical Aggregates on Probabilistic Data Streams," Proc. 26th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS), 2007.
[2] C. Re´ and D. Suciu, "Efficient Evaluation of HAVING Queries on a Probabilistic Database ," VLDB J.—Int'l J. Very Large Data Bases, vol. 18, no. 5, pp. 1091-1116, 2009.
[3] N. Dalvi and D. Suciu, "Efficient Query Evaluation on Probabilistic Databases," VLDB J.—Int'l J. Very Large Data Bases, vol. 16, no. 4, pp. 523-544, 2007.
[4] A. Gal, M.V. Martinez, G.I. Simari, and V. Subrahmanian, "Aggregate Query Answering under Uncertain Schema Mappings," Proc. Int'l Conf. Data Eng. (ICDE), 2009.
[5] T. Tran, A. McGregor, Y. Diao, L. Peng, and A. Liu, "Conditioning and Aggregating Uncertain Data Streams: Going Beyond Expectations," Proc. Int'l Conf. Very Large Data Bases (VLDB), 2010.
[6] G. Rempala and J. Wesolowski, "Asymptotics of Products of Sums and U-statistics," Electronic Comm. in Probability, vol. 7, pp. 47-54, 2002.
[7] B. Ross, V.S. Subrahmanian, and J. Grant, "Aggregate Operators in Probabilistic Databases," J. ACM, vol. 52, no. 1, pp. 54-101, 2005.
[8] R. Gupta and S. Sarawagi, "Creating Probabilistic Databases from Information Extraction Models," Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB), 2006.
[9] L. Antova, T. Jansen, C. Koch, and D. Olteanu, "Fast and Simple Relational Processing of Uncertain Data," Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE), 2008.
[10] G. Cormode, F. Li, and K. Yi, "Semantics of Ranking Queries for Probabilistic Data and Expected Ranks," Proc. 22nd Int'l Conf. Data Eng. (ICDE), 2009.
[11] D. Burdick, P. Deshpande, T.S. Jayram, R. Ramakrishnan, and S.Vaithyanathan, "OLAP over Uncertain and Imprecise Data," Proc. 31st Int'l Conf. Very Large Data Bases (VLDB), 2005.
[12] G. Cormode and M.N. Garofalakis, "Sketching Probabilistic Data Streams," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD), 2007.