IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 5, Oct-Nov, 2013
ISSN: 2320 - 8791
www.ijreat.org

ALL_SUM Query Evaluation over unpredictable data

Roslinmary M[1], SaravanaKumar T[2] and AddlinShinney R[3]

[1]Information Technology, Dr.Sivanthi Aditanar College of Engineering, Tiruchendur, 628215, India

[2]Information Technology, Dr.Sivanthi Aditanar College of Engineering, Tiruchendur, 628215, India

[3]Information Technology, Dr.Sivanthi Aditanar College of Engineering, Tiruchendur, 628215, India

Abstract

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 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 application where the aggregate attribute values are real with small precision. In this paper, evaluating a pseudo - polynomial algorithm called AgrQSUM algorithm which is based on a recursive approach, it efficiently calculate ALL_SUM Query. The proposed AgrQSUM 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*

1. Introduction

Uncertain Database is "Membership of an item to the database" is a probabilistic eventOr The value of attributes is a probabilistic variable .Uncertain data streams are important in growing number of environments, such as traditional sensor networks, GPS system for locationing, RFID networks for object tracking, radar networks for severe weather monitoring and the telescope surveys for astrophysical pattern.Aggregate query on those applications is very important.

For Example,E-healthManagementSystem. 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 patientshealth 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.

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".Fig. 1 shows an example table of this application. The table shows the number of required nurses for each patient

Fig.1: Motivating Example

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

The table.1 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.

A Q_PSUM 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, response time of these algorithms is high compared to our approach

In this demonstration, we present aprobabilistic database system for managing uncertain data. In particular, we show the efficiency of processing aggregatequeries such as ALL_SUM.Our demonstration application is the E-healthManagement system application described in above Example.

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

Table.1 The possible worlds and the results of SUM query in each instances.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. InSection 3, we describe our prototype. In Section 4 we present performance evaluation of my paper. Section 5 concludes.

2.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:

All attributes in a Tuple are known precisely, existence of the Tuple is uncertain. 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 Tupleti 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.2 is under Tuple-level model.

Table.2 Tuple level model

State	Event	pS
p	e	0.4
p	f	0.6
Q	e	0.5
Q	f	0.4

For example, in the above table, the probability that neither of the two tuples (p,e) and (p,f) exists in the database is given by (1-0.4) * (1-0.6) = 0.24

Attribute-level model:

Tuples (identified by *keys*) exist for certain; an attribute value is however uncertain for example Tomorrow temperature will be somewhere between 50F and 70F

Table.3. attribute level model

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

For example, in the above table Tuples exist with certainty. Temperature at time t1 at location 1 etc.But the attribute values (temperatures) are uncertain

2.2 Problem Definition

ALL_SUM Query:

It returns all possible sum results together with their probability.Our objective of our paper is to return the result of sum as follows

ALL_SUM(Q,D)=$\{(v,p)/v \in V_{D,f} \wedge p$=P(v,Q,D)$\}$

p(v,Q,D), is computed as follows:

$$\mathbf{p(v,Q,D)} = \sum_{\mathbf{w\epsilon W and f(w)=v}} \mathbf{p(w)}.$$

Where Dbea uncertain database, *W*the set of its possible worlds, and *P (w)* theProbability of a possible world *w*∈*W*. Let *Q*be a given aggregate query, *f* the aggregate function Stated in *Q* (i.e. SUM), f *(w)* the result of executing *Q* in a world *w*∈*W*, and *VD,fthe*set of all possible results of executing *Q*over D. Thecumulative probability of having a value *v*as the result of Q over D, denoted as P(v; Q; D).

2.3 Algorithmic Basis

In Probabilistic database, almost all aggregate functions are processed using recursive algorithms. Below, we discuss about the ALL_SUM algorithm that is in charge of executing SUM queries.

Let *t1…tn*be the tuples of the given uncertain database which is under the Tuple-level model. Let DBj be a database involving the tuples *t1…tj*, and Wj be the set of all possible worlds in DBj. Let *ps(i, j)* be the probability of having *sum* = *i*inDBj. We develop a recursive approach for computing *ps (i, j).*

2.3.1 Base

Let us consider DB1, *i.e.* the database that involves only the Tuple*t1*. Let *p(t1)* be the membership probability of *t1*, and *val(t1)* be the aggr value of *t1*. In DB1, there are two Possible worlds: 1) *w1={}*, in which *t1* does not exist, so its probability is *(1- p(t1))*; 2)*w2={t1}*, in which *t1* exists, so the probability is *p(t1)*. In *w1*, we have *sum=0*, and in *w2* we have *sum=val(t1)*. If *Val(t1) = 0*, then always we have *sum=0* because in both *w1* and *w2* sum is zero.

2.3.2 Recursion Step

Now consider DBn-1, *i.e.* a database involving the tuples *t1, …, tn-1*. Let Wn-1 be the set of possible worlds for DBn-1, i.e. set of possible instances for DBn-1. Let*ps(i, n-1)* be the probability of having *sum=i*in DBn-1, *i.e.* the aggregated probability of the DBn-1 worlds in
Which we have *sum=i*. Now, we construct DBn by adding *tn to* DBn-1. Notice that the set ofDBn possible worlds, denoted by Wn, are constructed by adding or not adding the Tuple*tn*toeach world of Wn-1. Thus, in Wn, there are two types of worlds: 1) the worlds that do notcontain*tn*, denoted as Wn1; 2) the worlds that contain *tn*, denoted as Wn2.For each world *w*∈ Wn1, we have the same world in DBn-1, say *w'*. Let *p(w)* and *p(w')* be theprobability of worlds *w* and *w'*. The probability of *w*, i.e. *p(w)*, is equal to *p(w')×(1 – p(tn))*,because *tn*does not exist in *w* even though it is involved in the database. Thus, in Wn
1 the sumvalues are the same as in DBn-1, but the probability of *sum=i*in Wn1 is equal to the probability of having *sum=i*in DBn-1 multiplied by the probability of non-existence of *tn*. In other words,wehave:In Wn1:

(Probability of sum=i) = ps(i, n-1)×(1 – p(tn)) **(1)**

Let us now consider Wn2. The worlds involved in Wn2 are constructed by adding *tn*toeachworld of DBn-1. Thus, for each sum value equal to *i*in DBn-1 we have a sum value equal to *(i+ val(tn))* in Wn2, where *val(tn)* is the aggr value of *tn*. The probability of *sum= i + val(tn)* in Wn2 is equal to the probability of *sum=i*inDBnmultipliedby the membership probability of*tn*. In other words, we have: In Wn2:

(Probability of sum=i) = ps(i - val(tn),n-1)×p(tn)(2)

Let *ps(i, n)* be the probability of *sum=i*inDBn. This probability is equal to the probability of*sum=i*inW$^n_1$ plus the probability of *sum=i*in W$^n_2$. Thus, by using the Equations 1 and 2, andusing the base of the recursion, we obtain the following recursive definition for theprobability of *sum=i*inDBn, i.e. *ps(i, n)* :

Ps(i,n)=

$$\begin{cases} ps(i,n-1) \times \big(1-p(t\,n)\big) + ps(i - val(tn), \\ \quad n-1 \times p(tn) & if\ n > 1 \\ 1-p(t1) & if\ n = 1\ and\ i = 0\ and\ val(t1)\ \neq 0 \\ p(t1) & if\ n = 1\ and\ i = val(t1) and\ val(t1) \neq 0 \\ 1 & if\ n = 1\ and\ i = val(t1) = 0 \\ 0 & otherwise \end{cases}$$

Based on the aboverecursive definition, we developed efficient algorithms for processingSUMQuery.

2.3.3 AgrQSUM Algorithm

In this section, using the dynamic programming technique, we propose an efficient algorithm, called AgrQSUM, 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.

AgrQSUM processed in two steps .In the first step,it initializes the first column of the matrix. This column represents the probability of sum values for a databaseinvolving only the Tuple t1. AgrQSUM initializes this column using the base of our recursive formula.

In the second step, in a loop, AgrQSUM sets the values of each column j (for j = 2 to n) by using our recursive definition and based on the values in column j – 1as follows:

PS[i; j]= PS[i, j _ 1]×(1 – p(tj))+
PS [i–val (tj), j – 1] ×p(tj).

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

Execution of DP-SUM over the database works correctly if the database is under Tuple-level model and the aggr attribute are positive integers, and their sum is less than or equal to MaxSum.

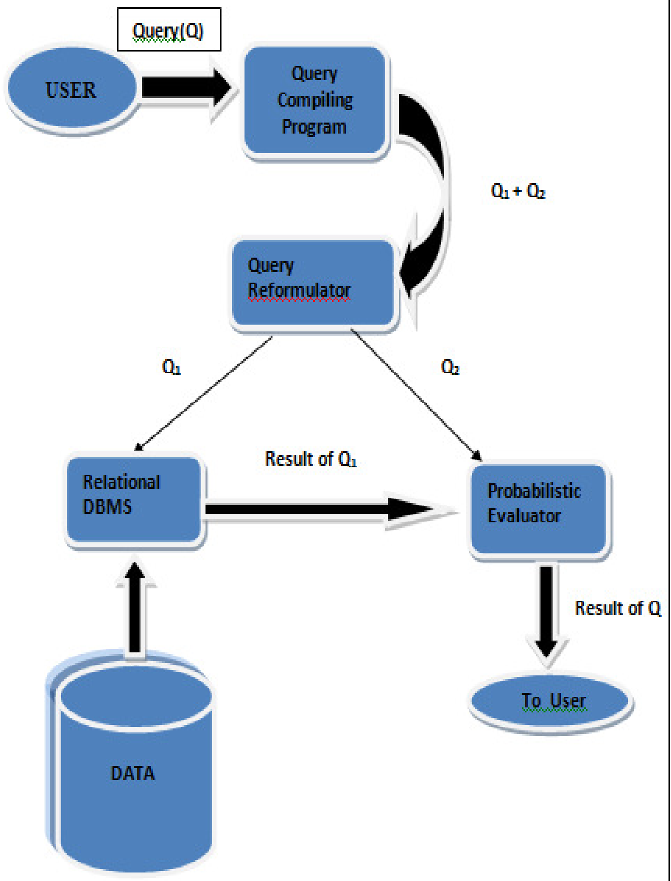
Pseudo code of AgrQSUM Algorithm

```
1.LetMaxSum= [n ×avg]
2.LetPS [MaxSum + 1, n] : be a two
Dimensional matrix
3. Step1: initializing the first row of the matrix
        • PS[val(t1), 1] = p(t1)
        • PS[0, 1] = (1-p(t1))
        • PS[i, 1] = 0 for  i≠0 and i≠val(t1)
4. Step2: compute other rows of the matrix
        • For j=2 to n do
        • For i=0 to MaxSum do
        • PS[i, j] = PS[i, j-1] × (1 - p(tj)) +
          PS[i-val(tj), j-1] × p(tj)
```

3. Prototype

ProbDB is built on top of a classical Database Management System (DBMS). It addsprobabilistic capabilities to the DBMS that are transparent to the user. Instead of directlymodifying the DBMS and adding "native" primitives to it, we have chosen to implementProbDB on top of the DBMS, and thus to be able to change the underlying DBMS with aslight programming effort. In its current version, the prototype is built atop PosgreSQL, butcould easily be adapted to work on a MySQL database for instance.

When the user sends a query to ProbDB, the query is analyzed and probabilistic keywords areextracted. Then classical (non probabilistic) sub-queries are sent to the DBMS that processthem and returns intermediate results. Then, probabilistic functions are applied to theintermediateresults, and the final results are returned to the user.ProbDB is composed of the following components (see the architecture)



Query Compiling Program: it is responsible for Separating the probabilistic parts of thequery from the ordinary ones. Let *Q* be thequery given by the user. The parser divides Query *Q* to two parts: 1) Q_1: the sub querythatcan be evaluated by a deterministicDBMS ; 2) Q_2: the parts of the query thatneed special probabilistic algorithms forbeing evaluated.

Query reformulator: this componentreformulates the sub queryQ_1 aquerythat can be executed by the underlyingDeterministic DBMS over the data storedin the database. It needs the metadata ofthe probabilistic tables in order to translateeach relation of Q_1*into* one or moreprobabilistic relations in the database.

Relational DBMS: this is an ordinary (deterministic) relational databasemanagement system that given the ReformulatedQ_1, executes it over the probabilistic tables, and returns the resultsto the component that evaluates the Probabilistic parts of the query.

Probabilistic Evaluator. The inputs of this component are the intermediate data generated bythe DBMS and the query Q_2. According to the probabilistic expressions in Q_2, the componentchooses the appropriate algorithms and runs them over the intermediate results, and returns the final results to the user.

4. Performance Evaluation

Performance of ALL_SUM Query Evaluated over real-world as well as synthetic datasets.

Based on Uncertain Tuple

Based on the response time of the algorithms vs. theNumber of uncertain tuples, i.e. *n*,the best algorithm is AgrQSUM, compared to Q_PSUMalgorithm.The response timeofAgrQSUM is at least four times lowerthan that of Q_PSUM

Overall, the problem which we consideredin this report, i.e. returning the exact results of ALL_SUM queries, there is no efficient solution in the related work. In this report, we proposed AgrQSUM algorithms that allow us to efficiently evaluate ALL_SUM queries in many practical cases, e.g. where the aggregate attribute values are small integers, or real numbers with limited precisions.

5.Conclusions

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. 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 AgrQSUM 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 AgrQSUM is better than Q_PSUM.

References

1. T.S. Jayram, A. McGregor, S. Muthukrishnan, E. Vee.Estimating statistical aggregates on probabilistic data streams. PODS Conf., 243-252, 2007.
2. T.S. Jayram, S. Kale, E. Vee. Efficient aggregation algorithms for probabilistic data. SODA Conf., 346-355, 2007.
3. T. Ge, S. Zdonik and S. Madden. Top-k Queries on Uncertain Data: On Score Distribution and Typical Answers. Sigmod Conf., 2009.
4. N. Tatbul, M. Buller, R. Hoyt, S. Mullen, S. Zdonik. Confidence-based Data Management for Personal Area Sensor Networks. In DMSN, 2004.
5. M. Ha-Duonga, R. Swartb, L. Bernsteinc and A. Petersenb.Uncertainty management in the IPCC: Agreeing to disagree. J. Global Environmental Change, 17(1), 2007
6. 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
7. C. Re´ and D. Suciu, “The Trichotomy of HAVING Queries on aProbabilistic Database,” VLDB J.—Int’l J. Very Large Data Bases,vol. 18, no. 5, pp. 1091-1116, 2009.
8. A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W.Hong, “Model-Driven Data Acquisition in Sensor Networks,”Proc. 30th Int’l Conf. Very Large Data Bases (VLDB), 2004.
9. D. Deutch and T. Milo, “TOP-K Projection Queries for ProbabilisticBusiness Processes,” Proc. 12th Int’l Conf. Database Theory(ICDT), 2009
10. 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.
11. T. Ge, S.B. Zdonik, and S. Madden, “Top-k Queries on UncertainData: On Score Distribution and Typical Answers,” Proc. ACMSIGMOD Int’l Conf. Management of Data (SIGMOD), 2009.
12. T.J. Green and V. Tannen, “Models for Incomplete and ProbabilisticInformation,” IEEEData Eng. Bull., vol. 29, no. 1, pp.17-24 Mar.2006.

13. R. Gupta and S. Sarawagi, “Creating Probabilistic Databases fromInformation Extraction Models,” Proc. 32nd Int’l Conf. Very Large Data Bases (VLDB), 2006

14. M. Hua, J. Pei, W. Zhang, and X. Lin, “Ranking Queries onUncertain Data: A Probabilistic Threshold Approach,” Proc. ACMSIGMOD Int’l Conf. Management of Data (SIGMOD), 2008.

15. T.S. Jayram, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan,and H. Zhu, “Avatar Information Extraction System,” IEEE DataEng. Bull., vol. 29, no. 1, pp. 40-48, Mar. 2006.

16. C. Jin, K. Yi, L. Chen, J.X. Yu, and X. Lin, “SlidingWindowTopkQueries on Uncertain Streams,” Proc. Int’l Conf. Very Large DataBases (VLDB), 2008.

17. B. Kanagal and A. Deshpande, “Online Filtering, Smoothing andProbabilisticModelling of Streaming Data,” Proc. IEEE 24th Int’lConf. Data Eng. (ICDE), 2008.

18. B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv, “Query Evaluationover Probabilistic XML,” VLDB J.—Int’l J. Very Large Data Bases,vol. 18, no. 5, pp. 1117-1140, 2009.

19. A. Nierman and H.V. Jagadish, “ProTDB: Probabilistic Data inXML,” Proc. 28th Int’l Conf. Very Large Data Bases (VLDB), 2002.

20. J. Pei, B. Jiang, X. Lin, and Y. Yuan, “Probabilistic Skylines onUncertain Data,” Proc. 33rd Int’l Conf. Very Large Data Bases(VLDB), 2007.

21. G. Rempala and J. Wesolowski, “Asymptotics of Products of Sumsand U-statistics,” Electronic Comm. in Probability, vol. 7, pp. 47-542002.

22. A.D. Sarma, O. Benjelloun, A. Halevy, and J. Widom, “WorkingModels for Uncertain Data,” Proc. 22nd Int’l Conf. Data Eng.(ICDE), 2006.

23. M.A. Soliman, I.F. Ilyas, and K.C.-C. Chang, “Top-k QueryProcessing in Uncertain Databases,” Proc. 23rd Int’l Conf. DataEng. (ICDE), 2007.

24. G. Trajcevski, R. Tamassia, H. Ding, P. Scheuermann, and I.F.Cruz“Continuous Probabilistic Nearest-Neighbor Queries forUncertain Trajectories,” Proc. 12th Int’l Conf. Extending DatabaseTechnology: Advances in Database Technology (EDBT), 2009.

25. T. Tran, A. McGregor, Y. Diao, L. Peng, and A. Liu, “Conditioningand Aggregating Uncertain Data Streams: Going Beyond Expectations,Proc. Int’l Conf. Very Large Data Bases (VLDB), 2010.

26. B. Yang, H. Lu, and C.S. Jensen, “Probabilistic Threshold k nearestNeighbor Queries over Moving Objects in Symbolic IndoorSpace,” Proc. 13th Int’l Conf. Extending Database Technology:Advances in Database Technology (EDBT), 2010.

27. M.L. Yiu, N. Mamoulis, X. Dai, Y. Tao, and M. Vaitis, “EfficientEvaluation of Probabilistic Advanced Spatial Queries on ExistentiallUncertain Data,” IEEE Trans. Knowledge Data Eng., vol. 21,no. 1, pp. 108-122, Jan. 2009.

28. S.M. Yuen, Y. Tao, X. Xiao, J. Pei, and D. Zhang, “SupersedingNearestNeighbor Search on Uncertain Spatial Databases,” IEEETrans. Knowledge Data Eng., vol. 22, no. 7, pp. 1041-1055, July 2010.

29. P. Sen, A. Deshpande, and L. Getoor. Exploiting sharedcorrelations in probabilistic databases. PVLDB,1(1):809{820, 2008.

30. P. Sen, A. Deshpande, and L. Getoor. Read-once functionsand query evaluation in probabilistic databases. PVLDB, 3(1):1068{1079, 2010.

31. M. A. Soliman, I. F. Ilyas, and S. Ben-David. Supporting ranking queries on uncertain and incomplete data. VLDB J., 19(4):477{501, 2010.

32. L. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian. Probview: A flexible probabilisticdatabase system. ACM Trans. Database Syst., 22(3), 1997.

33. I. Mansuri and S. Sarawagi. A system for integrating unstructured data into relationaldatabases. In Proc. of the 22nd IEEE Int’l Conference on Data Engineering (ICDE), 2006.