ISSN: 2319 – 8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 1, Issue 2, December 2012

Flexible Operations in Uncertain and Probabilistic Database

Ajit R Pradnyavant[1], Prof. G A Patil[2]

M. Tech. Scholar, Department of Technology, Shivaji University, Kolhapur, Maharashtra, India[1]

Head, Department of Computer Science, D Y Patil College of Engineering and Technology, Kolhapur, Maharashtra, India[2]

Abstract: Many real world applications need a database that stores probabilistic and uncertain database. Trio is a robust prototype build to store and retrieve uncertain and lineage data. It also supports some features of a relational DBMS. ULDB is an extension of relational databases with expressive construct for representing and manipulating both lineage and uncertainty. ULDB representation is complete and it permits straightforward implementation of many relational operations. Currently Trio performs only *select-project-join* queries and some set operations. Queries are expressed using TriQL query language. This paper highlights on how multiple aggregation can be handled in *select* clause in Trio system for uncertain and probabilistic data. It also highlights on how *distinct* clause can be used along with aggregation function. The results on the implementation of *minus* and *intersect all* clause in Trio system have been discussed. These operations allow users to use Trio system in a more flexible way.

Keywords: Aggregation, database management, query processing, set operation.

I. INTRODUCTION

In traditional database management systems (DBMS), we can store the data item with exact value. We can not store inexact (uncertain, probabilistic, fuzzy, approximate, incomplete and imprecise) data into a DBMS. Database research has primarily concentrated on how to store and query exact data. The development of techniques allows user to express and efficiently process complex queries over large collections of data. Unfortunately, many real world applications produce large amount of uncertain data. In such cases, database need to do more than simply store and retrieve. They have to help the user shift through the uncertainty and find the results most likely to be the answer.

Probabilistic databases have received attention recently due to need for storing uncertain data produced by many real world applications. In uncertain database, each data item instance has multiple possible instances, each corresponds to a single possible state of database [2]. Lineage identifies a data item's derivation, in terms of other data in the database or outside data sources. Lineage is also important for uncertainty within a single database. When a user writes queries against uncertain data, the result is uncertain too. Lineage facilitated the correlation and coordination of uncertainty in query results with uncertainty in the input data. The relation between uncertain database and lineage is that lineage can be used for understanding and resolving uncertainty [6].

In the new Trio Project at Stanford, a prototype management system is under development, in which data, uncertainty of the data and data lineage are all first-class citizens. The objective is to address the shortcomings of conventional DBMS's. By combining data, uncertainty and lineage it provides a data management platform that is useful for data integration, data cleaning, information extraction systems, Scientific and Sensor data management, approximate and hypothetical query processing and other modern applications. Trio's database is managed by ULDB's. ULDB extend the standard relational model. Queries are expressed using TriQL. TriQL is the query language used in the Trio for querying data [1] [3].

II. LITERATURE SURVEY

Trio is system for Integrated Management of Data, Accuracy and Lineage. Trio is a new database system that manages not only data, but also the accuracy and lineage of the data. The goals of Trio project are to combine and distill previous work into a simple and usable model, design a query language as an understandable extension to SQL and most importantly build a working System. ULDB has been implemented in the Trio project wherein uncertain data is captured by tuples that may include several *alternatives* and possible values for some (or all) of their attributes, with optional *confidence* values associated with each alternative. The TriQL query language specifies a precise generic semantics for any relational query over a ULDB. The result of

ISSN: 2319 – 8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 1, Issue 2, December 2012

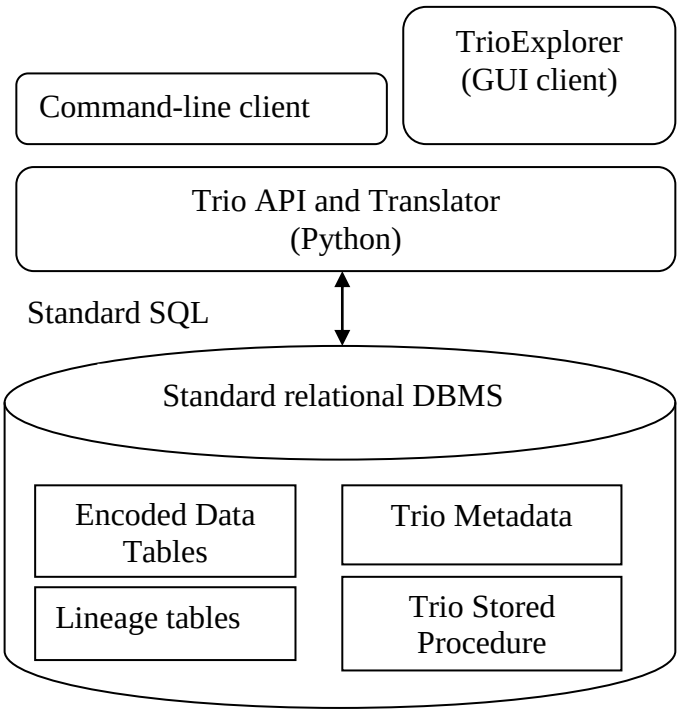
relational query *Q* on a ULDB *U* is a result *R* whose possible instances corresponds to applying *Q* to each possible-instance of *U*. TriQL includes a number of new features specific to uncertainty and lineage. TriQL allows construct for querying lineage, uncertainty, lineage and uncertainty together, special types of aggregation, extension to SQL's data modification, restructuring a ULDB relation. The Trio Prototype system is layered on top of the conventional relation DBMS. It is an implementation of ULDB model, TriQL query language and other features.

Initially, Trio system support *select-project* operations over uncertain database. After successful implementation of these basic operations they implemented *join* operation as we perform in SQL over two tables. As uncertain or probabilistic database based on possible-instances, for aggregation query the result size can grow exponentially with data size. There can be exponential number of possible instances, with different aggregation results in each one. To make the computation feasible, Trio offers several variants for aggregation function. A function returning the lowest possible value of the aggregate result *(low)*, the highest possible value *(high)* or the expected value *(expected)* [1]. Currently we can use only one aggregate function in *the select clause*. Some set operations like *minus, intersect all* are not implemented in the Trio. It also does not support *distinct* clause along with aggregate function [4].

Many certain databases allow users to use multiple aggregate function in *the select clause*. Computing aggregate over uncertain and probabilistic data is useful in situation where analytical processing is required over uncertain data. To make the user friendly database, we need the flexibility of queries over the uncertain and probabilistic database. If the user needs to add, count or perform basic statistical function, aggregate functions are helpful. These functions determine various statistics and values. Flexible operations reduce the amount of coding that the user needs to do in order to get information. Sometimes user need aggregation of distinct values. So to get an aggregation of distinct values, the uncertain database should support aggregation with *distinct* clause. For some queries we need *minus* set operation. So uncertain database should support use of *minus* set operator. These operations extend the flexibility of the Trio system [5].

III. THE TRIO SYSTEM

Figure 1 shows the basic three layer architecture of the Trio system. The core system is implemented in Python and it acts like a mediator between relational DBMS and Trio interfaces and applications. The Trio API accepts TriQL query and it modifies into regular SQL and query result may be ULDB tuples or regular tuples. It provides command line interactive client *(TrioPlus)* and *a TrioExplorer graphical user interface*.



ISSN: 2319 – 8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 1, Issue 2, December 2012

Figure 1: System Architecture

Trio DDL commands are translated via python to SQL DDL commands based on encoding. Processing of TriQL queries proceeds in two phases. In the *translation* phase, a TriQL parse tree is created and progressively transformed into a tree representing one or more standard SQL statement. In *the execution phase*, the SQL statement is executed against the relational database encoding. TriQL query results can either be *stored* or *transient*. Stored query results are placed in a new persistent table. Transient query results are accessed through the Trio API in a cursor-oriented fashion [6].

IV. IMPLEMENTATION

First we describe how relational tables are encoded in the Trio system to efficiently compute the queries. Consider a Trio relation $T(A_1....A_n)$. Relation *T* is stored in a conventional relational table with four additional attribute: *T_enc (xid, aid, conf, certain, $A_1....A_n$).* The Addition attributes in *T_enc* are as follows:

- *xid* identifies the x-tuple.
- *aid* identifies an alternative within the x-tuple.
- *conf* contains the confidence of the alternative.
- *certain* is flag to indicate whether the x-tuple has a ϕ alternative.

For Example, the Trio relation *Sightings(time, color, length)* is encoded into *Sighting_enc(xid, aid, conf, certain, time, color, length)* as shown in Table 1.

TABLE 1
Sighting_enc RELATION WITH TRIO SYSTEM ENCODING

xid	aid	conf	certain	time	color	length
101	1	0.5	0	1	gray	20
101	2	0.4	0	1	black	20
102	3	0.8	1	2	black	18
102	4	0.2	1	2	brown	16
103	5	1	1	2	brown	20

We have implemented the *minus* set operator in the Trio system. The *minus* operation returns unique rows that are returned by the first query but are *not* returned by the second query. Usually, *minus* is used to compare data from different data sources (tables). For example, differences in the same tables across test and production and/or actual copy and backup. Visually Query1 MINUS Query2 can be expressed as shown in Figure 2.

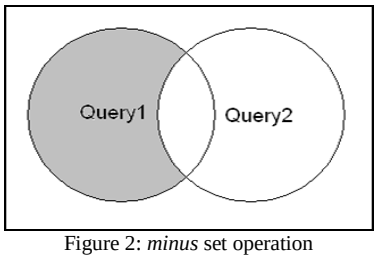


Figure 2: *minus* set operation

In Figure 2 shaded portion is the result of a query. The query is executed in the following way. The Trio system Python layer translates the TriQL query into the corresponding SQL query, sends it to the underlying DBMS and opens a cursor in the result. The translated query refers to virtual views. To use *minus* operator following conditions must be satisfied. (a) The result set of both the queries must have the same number of columns. (b) The data type of each column in the second result set must match the data type of the first result set.

Let *Tfetch* denote a cursor call to the Trio API for the original TriQL query and let *Dfetch* denote a cursor call to the underlying DBMS for the translated SQL query. Each call to the *Tfetch* must return a complete u-tuple, which may contain several calls to *Dfetch*. Each tuple returned from *Dfetch* on the SQL query corresponds to one alternative in the TriQL query result.

ISSN: 2319 – 8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 1, Issue 2, December 2012

A. Running Example: Squirrel Sighting

Trio application was inspired by the Christmas Bird Count [7], an original motivating example for Trio. Human volunteers observed, Squirrels on the Stanford campus and recorded its observations. Volunteer recorded the color (species) and length of each squirrel sighting along with the time of observation. For *minus* operation, we consider volunteer observation of two days. Observed data are stored in relation *SightingDay1 (time, color, length)* and *SightingDay2 (time, color, length)* as shown in Table 2 and Table 3 respectively.

TABLE 2
RELATION *SightingDay1 (time, color, length)*

time	(color, length)
1	(gray, 20) 0.5 || (black, 20) 0.4 || ϕ 0.1
2	(black, 18) 0.8 || (brown, 16) 0.2

TABLE 3
RELATION *SightingDay2 (time, color, length)*

time	(color, length)
1	(black, 18) 0.8 || (gray, 20) 0.2
2	(black, 20) 1.0

We run following query over two tables.

(SELECT color FROM SightingDay1) MINUS (SELECT color FROM SightingDay2) (1)

The result of the query is shown in Table 4.

TABLE 4
RESULT OF QUERY 1

color
brown:0.2

V. CONCLUSION

Trio supports *select-project-join* query. It also supports aggregation functions with variants. Many databases allow users to use multiple aggregate functions in the *select* clause. To make database user friendly, we need the flexibility of queries over the database. Flexible operations reduce the amount of coding that the user needs to do in order to get information. We have built new operations in the Trio system, which helps the user to use the system effectively. We have implemented the *minus* operation that used to calculate the difference between two resources.

The work is to implement following operations is in progress which will make the Trio system more flexible:

- Implementation of multiple aggregate function in *select* clause.
- Implementation of *intersect all* clause.
- Working with the Lineage data.

ACKNOWLEDGMENT

We are grateful to Prof. G. S. Kulkarni, Director and Ms. A. A. Manjrekar, M. Tech Computer Science and Technology course coordinator, Department of Technology, Shivaji University, Kolhapur for providing kind support and laboratory facility for carrying out research work.

REFERENCES

[1] R. Murthy, R. Ikeda, and J. Widom, "Making Aggregation Work in Uncertain and Probabilistic Databases," *IEEE Trans. Knowledge and Data Eng.*, vol. 23, Aug. 2011.
[2] R. Cavallo and M. Pittarelli, "The Theory of Probabilistic Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 71-81, 1987.

[3] O. Benjelloun, A. Das Sarma, A.Y. Halevy, and J. Widom "ULDBs: Databases with Uncertainty and Lineage," Conf. Very Large Data Bases (VLDB), pp. 953-964, 2006
[4] T.S. Jayram, S. Kale, and E. Vee, "Efficient Aggregation Algorithms for Probabilistic Data," Proc. ACM-SIAM Symp. Discrete Algorithms (SODA), pp. 896-905,2007.
[5] J. Widom, "Trio: A System for Integrated Management of Data, Accuracy, and Lineage," Proc. Conf. Innovative Data Systems Research (CIDR), pp. 262-276, 2005.
[6] J. Widom, *TRIO: A SYSTEM FOR DATA, UNCERTAINTY, AND LINEAGE*, Dept. of Computer Science Stanford University Verlag, 2008.
[7] Christmas Bird Count, Home Page, Available: http://www.audubon.org/bird/cbc, 2011.
[8] N. N. Davy and D. Suciu, "Making Aggregation Work in Uncertain and Probabilistic Databases," Proc. Workshop Management of Uncertain Data at Int'l conf. Very Large Data Bases (VLDB), pp. 76-90, 2007.
[9] R. Murthy and J. Widom "Making Aggregation Work in Uncertain and Probabilistic Databases," Proc. Workshop Management of Uncertain Data at Int'l Conf. Very Large Data Bases (VLDB), pp. 76-90, 2007.
[10] M. Mutsuzaki, M. Theobald, A. De Keijzer, J. Widom, P. Agrawal, O. Benjelloun, A. Das Sarma, R. Murthy, and T. Sugihara, "Trio-One: Layering Uncertainty and Lineage on a Conventional DBMS (Demo)," Proc. Conf. Innovative Data Systems Research (CIDR), pp. 269-274, 2007.
[11] L. Chen and A. Dobra, "Efficient Processing of Aggregates in Probabilistic Databases," Technical Report REP-2008-454, Univ. Of Florida, 2008.
[12] PL/pgSQL—SQL Procedural Language, Manual Available: http://www.postgresql.org/docs/7.4/interactive/plpgsql.html, 2009.

Biography

Ajit R Pradnyavant received BE degree in Information Technology from D Y Patil College of Engineering and Technology. He is pursuing M. Tech in Computer Science and Technology at Department of Technology, Shivaji University, Kolhapur. His research interest include database systems.

G A Patil is presently working as Associate Professor and Head in Department of Computer Science and Engineering, D Y Patil college of Engineering and Technology, Kolhapur. He has obtained a Bachelor's degree in Computer Science and Engineering from PES college of engineering, Mandya and post graduate degree in CSE from Walchand college of engineering, Sangli. His research interests are Cloud computing, Information security and Network Management.