Tractability in Probabilistic Databases

Dan Suciu
Department of Computer Science and Engineering
University of Washington
Seattle, WA
suciu@cs.washington.edu

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Query Processing*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic

General Terms

Algorithms, Theory

Keywords

Probabilistic database, query evaluation

Probabilistic databases are motivated by a large and diverse set of applications that need to query and process *uncertain data*. Uncertain and probabilistic data arises in RFID systems [22], information extraction [12], data cleaning [1], scientific data management [17], biomedical data integration [9], business intelligence [14], approximate schema mappings [10], data deduplication [13]. All these applications have large collections of data, where some, or most individual data items are uncertain.

In the simplest model, a probabilistic database is a database where each record is a random variable: it may either be present in the database, or absent from the database. More complex models allow individual attribute values [2], or individual XML elements to be random variables [15]. These random variables may be independent [6], disjoint and independent [3], may form a graphical model [24] or a Bayesian network [26]. The goal of a probabilistic database system is to allow probabilistic data to be stored natively, then processed with a general-purpose query language, like SQL [8]. Each answer to the query will be annotated with a probability, which represents the confidence in that answer based on the probabilities of the input tuples, and the system will rank the answers in decreasing order of this probability [21], or according to some combination of the probability score and a user-defined ranking function [27, 16].

Despite the intense recent research on probabilistic databases and its huge demand from applications, to date there are no usable systems available, besides a few still brittle research prototypes. At blame for this state of affair is that we do not have techniques that can deliver *predictable performance* for all queries on probabilistic databases, in the same way in which standard SQL engines can deliver predictable performance on traditional databases. The fundamental reason is that many relational queries have an intractable data complexity over probabilistic databases: they are hard for $FP^{\#P}$. Each such query can only be answered either by approximate probabilistic inference tools, which do not scale to volumes of data encountered in typical applications, or by heuristics (such as junction trees that exploit bounded tree widths), which often lead to exponential running times on real databases.

In this talk I will discuss tractable queries, and will describe some of the recent approaches in identifying queries with provably tractable complexity. It is important for us to identify the exact border between tractability and intractability, for two reasons. First, a database engine needs to identify whenever a query is tractable, in order to avoid a much more expensive computation; and, second, because of the interesting possibility of approximating intractable queries by tractable queries, which was suggested recently [11]. Several theoretical results characterize the class of tractable queries over probabilistic databases consisting of independent tuples, or of disjoint and independent tuples. This talk will present some of these results.

I will start by considering a simple case, when the query language is restricted to conjunctive queries without self-joins. In this simple case, the *hierarchical queries* have been shown to capture precisely the tractable queries [6, 8] and to also capture precisely the queries whose lineage is always a read-once Boolean expression [18]. Here we have a dichotomy into PTIME/#P based on the query's syntax: hierarchical queries are in PTIME and non-hierarchical queries are hard for $FP^{\#P}$. The former can be evaluated by a standard relational query plan, consisting of selection, projection, and join operators, where the projection and join operators are modified to compute the tuple probabilities. Various extensions and implementations exist for the tractable conjunctive queries without self-joins: extensions to functional dependencies [7, 20], improved in-engine query evaluation [20], extensions to queries with inequalities [19], to disjoint and independent tuples [4, 8], and to queries with a `having`-clause [23].

Next, I will discuss a richer query language, *unions of conjunctive queries*, and will present the class of tractable queries over this class, following [5]. Each tractable query in this case can be evaluated by an almost equally simple, but more surprising algorithm. The key new step in this algorithm consists of the inclusion/exclusion formula, which has to be applied not to the usual

ICDT 2011, March 21–23, 2011, Uppsala, Sweden.

union of conjunctive queries, but to a dual representation, *conjunction of disjunctive queries*. Another surprising feature of the algorithm is its key usage of Möbius' function on a lattice [25], whose role is indispensable in ensuring that the algorithm is complete: when the Möbious function of a sub-query is zero, then that sub-query does not contribute to the inclusion/exclusion formula, and therfore its evaluation can be omitted. Here, too, we have a dichotomy into PTIME/#P based on the query's syntax, but where the notion of "syntax" includes the Möbius function over a certain lattice of sub-queries, derived from the query's syntax.

Finally, I will conclude the talk with a list of open research questions in probabilistic databases.

Acknowledgment The author was partially supported by NSF IIS-0713576.

1. REFERENCES

[1] P. Andritsos, A. Fuxman, and R. J. Miller. Clean answers over dirty databases. In *ICDE*, 2006.

[2] D. Barbara, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Trans. Knowl. Data Eng.*, 4(5):487–502, 1992.

[3] O. Benjelloun, A. D. Sarma, A. Halevy, M. Theobald, and J. Widom. Databases with uncertainty and lineage. *VLDBJ*, 17(2):243–264, 2008.

[4] N. Dalvi, C. Re, and D. Suciu. Query evaluation on probabilistic databases. *IEEE Data Engineering Bulletin*, 29(1):25–31, 2006.

[5] N. Dalvi, K. Schnaitter, and D. Suciu. Computing query probability with incidence algebras. In *PODS*, pages 203–214, 2010.

[6] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, 2004.

[7] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDBJ*, 16(4):523–544, 2007.

[8] N. Dalvi and D. Suciu. Management of probabilistic data: Foundations and challenges. In *PODS*, pages 1–12, Beijing, China, 2007. (invited talk).

[9] L. Detwiler, W. Gatterbauer, B. Louie, D. Suciu, and P. Tarczy-Hornoch. Integrating and ranking uncertain scientific data. In *ICDE*, pages 1235–1238, 2009.

[10] X. Dong, A. Halevy, and C. Yu. Data integration with uncertainty. In *VLDB*, 2007.

[11] W. Gatterbauer, A. Jha, and D. Suciu. Dissociation and propagation for efficient query evaluation over probabilistic databases constraints. In *MUD*, 2010.

[12] R. Gupta and S. Sarawagi. Creating probabilistic databases from information extraction models. In *VLDB*, pages 965–976, 2006.

[13] O. Hassanzadeh and R. J. Miller. Creating probabilistic databases from duplicated data. *VLDB J.*, 18(5):1141–1166, 2009.

[14] R. Jampani, F. Xu, M. Wu, L. Perez, C. Jermaine, and P. Haas. MCDB: a Monte Carlo approach to managing uncertain data. In *SIGMOD*, pages 687–700, 2008.

[15] B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv. Query evaluation over probabilistic XML. *VLDB J.*, 18(5):1117–1140, 2009.

[16] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. *PVLDB*, 2(1):502–513, 2009.

[17] A. Nierman and H. Jagadish. ProTDB: Probabilistic data in XML. In *VLDB*, pages 646–657, 2002.

[18] D. Olteanu and J. Huang. Using OBDDs for efficient query evaluation on probabilistic databases. In *SUM*, pages 326–340, 2008.

[19] D. Olteanu and J. Huang. Secondary-storage confidence computation for conjunctive queries with inequalities. In *SIGMOD*, pages 389–402, 2009.

[20] D. Olteanu, J. Huang, and C. Koch. Sprout: Lazy vs. eager query plans for tuple-independent probabilistic databases. In *ICDE*, pages 640–651, 2009.

[21] C. Re, N. Dalvi, and D. Suciu. Efficient Top-k query evaluation on probabilistic data. In *ICDE*, 2007.

[22] C. Re, J. Letchner, M. Balazinska, and D. Suciu. Event queries on correlated probabilistic streams. In *SIGMOD*, Vancouver, Canada, 2008.

[23] C. Re and D. Suciu. The trichotomy of HAVING queries on a probabilistic database. *VLDB J.*, 18(5):1091–1116, 2009.

[24] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *ICDE*, 2007.

[25] R. P. Stanley. *Enumerative Combinatorics*. Cambridge University Press, 1997.

[26] D. Wang, E. .Michelakis, M. Garofalakis, and J. Hellerstein. BayesStore: managing large, uncertain data repositories with probabilistic graphical models. *PVLDB*, 1(1):340–351, 2008.

[27] X. Zhang and J. Chomicki. On the semantics and evaluation of top-k queries in probabilistic databases. In *International Workshop on Database Ranking (DBRank)*, pages 556–563, 2008.