Working Group: Lineage/Provenance

Anish Das Sarma[1], Amol Deshpande[2], Thomas Hubauer[3], Ihab Ilyas[4], Birgitta König-Ries[5], Matthias Renz[6], and Martin Theobald[7]

[1] Stanford University
[2] University of Maryland
[3] Siemens München
[4] University of Waterloo
[5] Universität Jena
[6] Ludwig-Maximilians-Universität München
[7] Max-Planck-Institut Informatik, Saarbrücken

Abstract. The following summary tries to capture a collection of state-of-the-art techniques and challenges for future work on lineage management in uncertain and probabilistic databases that we discussed in our working group. It was one half of a larger committee that we had initially formed, which then got split into two groups—one focusing on lineage as a means of explanation of data, and one focusing more on lineage usage in probabilistic databases (see also the "Explanation" working group report for more details on the first subgroup).

1 Why lineage?

Users want lineage as a basic functionality of the database or information system they use. Explicitly tracing lineage information along with the actual results of queries may help users to better understand complex workflows or to analyze results in huge data warehouses that are often derived from complicated queries and many levels of materialized views. Lineage, as a form of *annotation* or *metadata* over the actual data, can provide this information without the need to rerun the entire query or workflow and thus help to trace the origin and/or explain the derivation of individual results. Tracing lineage is not part of any commercially available DBMS today, and yet it is one of the top items on the wishlist of many database and data warehouse customers. Particularly challenging, large-scale applications are in domains like Biomedical or Genomics databases, where lineage may help to model and better understand various forms of evolutionary processes or workflows.

Systems need lineage for their representation models. Lineage recently underwent a renaissance in the context of uncertain and probabilistic databases, where constraints (typically in the form of *Boolean formulas* over the base data) are needed to capture the *possible instances* that the uncertain database may take. In systems like Trio [3, 10], already seemingly simple relational operations like joins over two or more uncertain relations may create situations, where the query result could not properly be captured without this explicit lineage information. Here, lineage is key for *closed* and *complete representation models*. Query processing in these systems then however becomes closely related to the evaluation of propositional logic, with the well-known complexity issues: NP-hardness for satisfiability tests of Boolean formulas in the general case, and even #P-completeness for confidence computations when all the possible worlds that the result tuple belongs to need to be enumerated. Thus, efficient approximation algorithms with tight guarantees are sought-after by all current systems [2, 4, 10, 12, 14]. *Scalability*, in particular for complex operations such as confidence computations or co-existence checks of results with respect to this possible-worlds semantics, will remain a challenging issue.

2 Lineage in Uncertain and Probabilistic Databases

In uncertain and probabilistic databases, closedness of operations means that any result of (relational) operations over the uncertain database can again be captured in the uncertain database correctly. Completeness of the representation model, on the other hand, means that the uncertain database can represent any finite set of uncertain data. Here, completeness implies closedness (with a respective implementation of relational operators), and both can be achieved via lineage. Interesting efficiency aspects lie in the different strategies on how to manage lineage. With respect to efficiency versus runtime and storage trade-offs, different approaches either follow an eager (sometimes even transitive) materialization of lineage, or some more lazy, recursive unfolding strategies [9]. Approximation and aggregation of lineage may also be of high interest, not only with respect to efficiency aspects but also in terms of privacy and presentation issues (see below).

In various current systems, lineage allows not only to trace the origin of results as plain sets of base tuples, but it can capture the exact *semantics* of relational operations (or workflows) with respect to the possible instances of the database, often referred to as the *intensional semantics* [8, 12] of operations. Opposed to this intensional semantics is the *extensional semantics* that aims to avoid the additional overhead of tracing these constraints altogether—whenever it is safe to do so. Both strategies provide different challenges and opportunities for efficient query processing over expressive representation models, while current approaches and implementations still remain highly specialized and differ widely in the functionality they can support efficiently. [1], for example, focuses on attribute-level decomposition techniques to compactly encode very large sets of possible worlds.

Extensional semantics: In the extensional case, one may try to identify so-called "safe query plans" [11], when queries "naturally" preserve the possible instances of the database correctly and there is no need to explicitly constrain the results. In classic probabilistic databases this is the case for a broad class of operations and queries, e.g., for joins over independent input relations, as well as duplicate-preserving projections and selections. The extensional case has the lowest-possible overhead for processing probabilistic data and allows for fast query response times in the order of standard SQL queries. In uncertain databases (e.g., when already having constraints among the base data, like mutual exclusiveness of tuple alternatives in Trio or MayBMS [1, 2]) this class of "safe-plans" is however further restricted.

Intensional semantics: Data in derived relations is typically no longer independent as multiple tuples may share the same input. Thus, queries involving derived or otherwise non-independent input data, including complex operations like self-joins or duplicate eliminations, require some notion of constraints to explicitly capture and encode the possible worlds that the database may take. The constraints are typically stored as Boolean lineage formulas among input tuples. In systems like Trio [10] or MayBMS [2], the structure of the Boolean formulas reflects the semantics of the relational operations applied in the query plan (e.g., using conjunction for joins, disjunction for duplicate elimination, and negation for set difference). Trio is currently the only uncertain database system that allows for a true form of data warehousing over uncertain data as it always traces the lineage back to the base data, possibly through multiple levels of materialized views. [12] shows that being able to dynamically choose between extensional and intensional query processing strategies based on the query plan can however drastically increase the performance.

3 Approximating Lineage

Approximating lineage information may be of interest when the lineage information itself becomes huge (e.g., for aggregations like in "WHERE COUNT(*)=3", or negations over subqueries like in "WHERE NOT EXISTS (...)"), or when the amount of possible instances becomes intractable (e.g., when dealing with/eliminating large amounts of duplicates). This may lead to a new notion of *expensive predicates*, namely those that involve high processing complexity not only in the actual data but also in the lineage. Considering *different granularities* of lineage may involve schema-level, record-level, and even different granularities of external lineage (pointing to external sources). One valid goal may be to avoid the intensional semantics for the expensive cases like aggregations or duplicate eliminations, which involve high combinatorial complexity in processing and/or computing confidences. A possible form of approximating or summarizing lineage could be to show distributions that capture the impact of the input data at different granularities in the form of (multi-dimensional) histograms, using different groupings/aggregations over the lineage information itself. *Convolution-like summaries* may be used for specific aggregation operations like summation.

4 Uncertain Lineage

Sometimes there may be uncertainty on the lineage information itself, i.e., one might not be sure where the actual information comes from (maybe an external source), and thus also attach confidences or alternatives to the lineage itself. A similarity to "Probabilistic Rules" [7] was pointed out, which may be modeled as Datalog-like Horn clauses that in turn are weighted by a confidence. Furthermore, formally specifying *updates* to the lineage will involve interesting semantic issues.

5 Privacy and Presentation Issues

Storing and exposing lineage information to users obviously involves lots of privacy issues. Privacy thus affects the way lineage can (and should) be presented to the user and/or how lineage can be used to explain results. Summaries or approximations (see above) will be of high interest also with respect to privacy issues. Appropriate *user interfaces* should allow the user to navigate through different lineage granularities and provide aggregations and summarizations on demand and at the same time need to consider these privacy issues.

6 Non-Independent Input Data

Lineage, in a more general sense, can be considered as pointers between data objects—not necessarily only between base and derived data objects, but also directly between different base data. Thus, modeling non-independent or otherwise correlated base data [13] (beyond just considering special cases like mutually exclusive alternatives or so-called OR-sets as in [2, 3]) will pose major challenges to the existing probabilistic-database approaches. Moreover, investigating the relationship and/or synergies between probabilistic databases and more sophisticated graphical models like Bayesian Nets [5, 6, 15] has become an emerging research topic lately.

References

1. L. Antova, C. Koch, and D. Olteanu. 10^{10^6} worlds and beyond: Efficient representation and processing of incomplete information. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE)*, pages 606–615, 2007.
2. L. Antova, C. Koch, and D. Olteanu. MayBMS: Managing incomplete information with probabilistic world-set decompositions. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE)*, pages 1479–1480, 2007.
3. O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, and J. Widom. Databases with uncertainty and lineage. *VLDB J.*, 17(2):243–264, 2008.
4. J. Boulos, N. N. Dalvi, B. Mandhani, S. Mathur, C. Ré, and D. Suciu. MYSTIQ: a system for finding more answers by using probabilities. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 891–893, 2005.
5. H. C. Bravo and R. Ramakrishnan. Optimizing MPF queries: decision support and probabilistic inference. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 701–712, 2007.
6. A. Deshpande and S. Sarawagi. Probabilistic graphical models and their role in databases. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*, pages 1435–1436, 2007.
7. N. Fuhr. Probabilistic datalog - a logic for powerful retrieval methods. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 282–290, 1995.
8. N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

9. T. Heinis and G. Alonso. Efficient lineage tracking for scientific workflows. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1007–1018, New York, NY, USA, 2008. ACM.
10. M. Mutsuzaki, M. Theobald, A. de Keijzer, J. Widom, P. Agrawal, O. Benjelloun, A. D. Sarma, R. Murthy, and T. Sugihara. Trio-One: Layering uncertainty and lineage on a conventional dbms (demo). In *Third Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 269–274, 2007.
11. C. Re, N. N. Dalvi, and D. Suciu. Query evaluation on probabilistic databases. *IEEE Data Eng. Bull.*, 29(1):25–31, 2006.
12. C. Re, N. N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE)*, pages 886–895, 2007.
13. P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE)*, pages 596–605, 2007.
14. M. A. Soliman, I. F. Ilyas, and K. C.-C. Chang. URank: formulation and efficient evaluation of top-k queries in uncertain databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data (SIGMOD)*, pages 1082–1084, New York, NY, USA, 2007. ACM.
15. D. Z. Wang, E. Michelakis, M. N. Garofalakis, and J. M. Hellerstein. BayesStore: managing large, uncertain data repositories with probabilistic graphical models. *PVLDB*, 1(1):340–351, 2008.