Scaling Probabilistic Databases

Hernán Blanco
University of Antwerp, Belgium
hernan.blanco@uantwerpen.be

Supervised by Martin Theobald
University of Ulm, Germany
martin.theobald@uni-ulm.de

ABSTRACT

Probabilistic databases, which have been widely studied over the past years, lie at the expressive intersection of databases and probabilistic graphical models, thus aiming to provide efficient support for the evaluation of probabilistic queries over uncertain, relational data.

Several Machine Learning approaches, on the one hand, have recently investigated the issue of *distributed probabilistic inference* but do not support relational data and SQL. Conventional database engines, on the other hand, do not handle probabilistic data and queries, nor any form of uncertain data management. With this project, we aim to fill this prevalent gap between the two fields of Databases and Machine Learning by *scaling probabilistic databases* to a distributed setting, which is a topic that so far has not been addressed in the literature. The proposed PhD dissertation topic provides a number of intriguing and challenging aspects, both from a theoretical and a systems-engineering perspective.

1. INTRODUCTION

Managing uncertain data via probabilistic databases (PDBs) has evolved as an established field of research in recent years. The field meanwhile encompasses a plethora of applications, which are ranging from scientific data management, sensor networks, data integration, to knowledge management systems [13, 12]. PDBs lie at the intersection of databases and probabilistic graphical models. While classical database approaches benefit from a mature and scalable infrastructure for the management of relational data, probabilistic databases aim to further combine these well-studied data management strategies with inference techniques known from graphical models such as Bayesian Networks and Markov Random Fields. PDBs adopt powerful query languages from relational databases, including Relational Algebra, the Structured Query Language (SQL), and logical query languages such as Datalog.

Proceedings of the VLDB 2015 PhD Workshop

The Trio probabilistic database system [7] was the first system that explicitly addressed the integration of data management via SQL, provenance (aka. "lineage") management via Boolean formulas, and efficient probabilistic inference. The data model which was considered for Uncertainty and Lineage Databases (ULDBs) [1] was the first probabilistic database approach that was shown to provide a closed and complete probabilistic extension to the relational model which supports all of the core relational (i.e., SQL-based) operations over this kind of uncertain data.

A key in making probabilistic inference scalable to the extent that is needed for modern data management applications lies (without considering parallelization techniques yet) in the identification of tractable query classes and the adaptation of known inference techniques from graphical models to a relational data setting. While for specific classes of queries, probabilistic inference can directly be coupled with the relational operations [9, 12], the performance may very quickly degenerate when the underlying independence assumptions among the data objects do not hold. Despite the polynomial runtime complexity for the data computation step that is involved in finding answer candidates to probabilistic queries, the confidence computation step, i.e., the actual probabilistic inference, for these answers is known to be of exponential cost already for fairly simple SQL queries. Consequently, much of the recent research in PDBs has focused on establishing a classification of query plans for which confidence computations are either of polynomial runtime or are #P-hard [2, 12]. Thus, strategies for efficient confidence computations and early pruning of low-confidence query answers remain a key challenge also for the scalable management of uncertain data via PDBs.

2. STATE-OF-THE-ART

Recent work on efficient confidence computations in PDBs has addressed this problem mainly from two ends, namely by restricting the class of queries that are allowed, i.e., by focusing on so-called *safe query plans* [12], or by considering a specific class of tuple-dependencies, commonly referred to as *read-once functions* [11]. Intuitively, safe query plans denote a class of queries for which confidence computations can directly be coupled with the relational operators and thus be performed by an extensional query plan. On the other hand, read-once formulas denote a class of Boolean lineage formulas which can be factorized in polynomial time into a form where every variable in the formula (each representing a database tuple) appears at most once, thus again permitting efficient confidence computations.

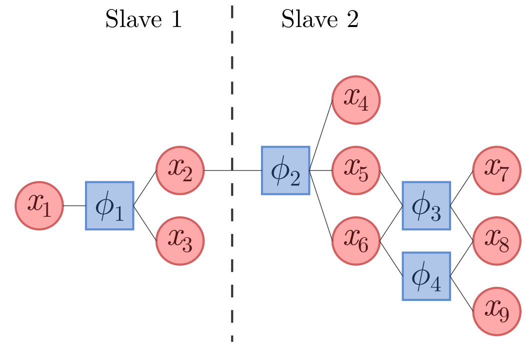
While safe plans clearly focus on the characteristics of the query structure, and read-once formulas focus on the logical dependencies among individual data objects, *top-k-style pruning approaches* [4, 8] have been proposed as an alternative way to address confidence computations in PDBs. These approaches aim to efficiently identify the top-k most probable answers, using lower and upper bounds for their marginal probabilities, without the need to compute the exact probabilities of these query answers. In [4], a form of first-order lineage formulas (based on a restricted class of first-order logic) is proposed, which allows to fully integrate the data and confidence computation steps in a probabilistic database setting. PrDB [10] is one of the most generic PDB approaches in the field, aiming to significantly widen the opportunities to include richer probabilistic models by allowing uncertainty at both tuple and attribute level as well as tuple correlations. It also achieves improvements in terms of fast inference by implementing a self-developed algorithm based on bisimulation. However, also PrDB does not address the problem of distribution and parallelization, where an enormous potential for scalability lies.

On the other hand, recent trends in relational databases clearly focus on "Big Data" management and "Cloud Computing". Distributed database engines like Apache's Cassandra, HIVE, or HBase aim to achieve an even better scalability to many petabytes of data by going for distributed file systems and by performing SQL queries via iterative and largely synchronous communication workflows based on Google's MapReduce or Apache's Hadoop frameworks.

There currently exists no distributed probabilistic database system. A number of systems brought up by the Machine Learning and Artificial Intelligence communities, such as CMU's GraphLab [6], MIT's FactorIE, Microsoft's Infer.NET (MSR Cambridge), and the recently released Amazon Machine Learning and AMPLab's KeystoneML platforms focus on purely graph-based and partly also on distributed approaches. However, all of these employ their own APIs and do not natively support relational data or SQL. From a more classical (i.e., deterministic) database perspective, we have recently seen distributed engines like SciDB or Berkeley's BOOM project, which were designed from scratch to be massively scalable (and which, in the first case, originally even had the support for uncertain data as part of its core design goals), but they ultimately do not support probabilistic or otherwise uncertain data.

3. NOVEL CONTRIBUTIONS

In this project, we advocate for the development of a radically new, scalable, and massively distributed infrastructure for probabilistic databases. The goal is to investigate how far the advantages of mature database technologies can be carried over to a distributed and probabilistic setting. We propose a probabilistic data(-base) model, seeking to combine both the generic semantics of a relational backend with parallelization opportunities for probabilistic inference based on graphical models. Our intended research will specifically focus on the core functionalities of a database in terms of efficient query processing over a distributed, persistent storage layer with parallel query routing whenever possible. We will further focus on adapting existing techniques for probabilistic inference based on variable elimination and asynchronous message passing.

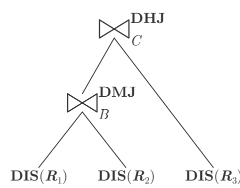


$\boldsymbol{R}_1$	A	B
x_1	a_1	b_1
x_2	a_2	b_2
x_3	a_3	b_3

$\boldsymbol{R}_2$	B	C
x_4	b_1	c_1
x_5	b_2	c_3
x_7	b_3	c_2

$\boldsymbol{R}_3$	C	D
x_6	c_1	d_2
x_8	c_1	d_1
x_9	c_2	d_1

Figure 1: A sample factor graph. The red circles represent variable nodes which are database items (instantiated in relations $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$); the blue squares represent factor nodes. The data is distributed across two cluster slaves by minimizing the cut between the two partitions of the factor graph.



(a)

lineage	A	B	C	D
$l_1 : x_1 \wedge x_4 \wedge x_6$	a_1	b_1	c_1	d_2
$l_2 : x_1 \wedge x_4 \wedge x_8$	a_1	b_1	c_1	d_1
$l_3 : x_3 \wedge x_7 \wedge x_9$	a_3	b_3	c_2	d_1

(b)

Figure 2: **(a)** A representation of a distributed query execution: *distributed index scans* over the tree base relations $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$ from Figure 1, a *distributed merge join* on B, and a *distributed hash join* on C. **(b)** The results retrieved from evaluating the query represented in Figure 2(a), each record holding its respective lineage expression.

3.1 A Unifying Data Model

Specifically, in this PhD project we advocate for the investigation of a distributed factor graph model as being the core data model for the intended (both distributed and probabilistic) database infrastructure. Factor graphs provide a generic data model for capturing various kinds of probabilistic graphical models and have successfully been applied to related (but non-distributed) probabilistic settings in the past. A *factor graph* (see Figure 1) is a bipartite graph $G(X, \Phi)$ that consists of a set of variable nodes $X = \{x_1, \ldots, x_m\}$ and a set of factor nodes $\Phi = \{\phi_1(X_1), \ldots, \phi_n(X_n)\}$, where each $X_s \subseteq X$ is a subset of variables which are associated with a factor function $\phi_s : X_s \to \mathbb{R}$, such that $G(X, \Phi) = \prod_i \phi_i(X_i)$.

Variable nodes will represent data items of mutable (usually binary) state, which are connected to factor nodes in order to model weighted dependencies between the possible states of the variable nodes. This generic graphical model provides a very flexible and scalable representation model, which can easily be serialized into relational data.

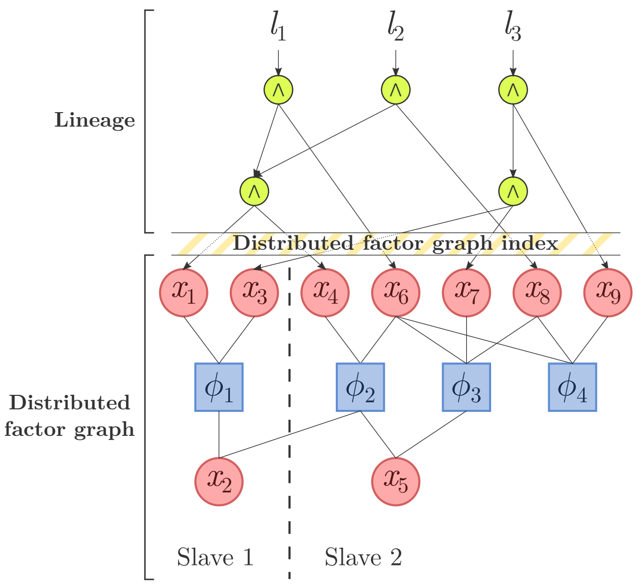


Figure 3: A representation of the complete, both distributed and probabilistic, setting obtained from the query results in Figure 2(b). It illustrates the combined data model: the *directed acyclic graph* (DAG) structures at the top represents the three lineage formulas obtained from the query evaluation, l_1, l_2 and l_3; the bottom part shows their connection to the factor graph from Figure 1.

Factor graphs subsume both tuple-independent probabilistic database approaches (where a factor node is connected to exactly one variable node, representing its marginal probability) and current state-of-the-art probabilistic database [10] approaches based on either undirected graphical models, such as Markov Random Fields, or directed ones, like Bayesian Networks (where a factor node may be connected to multiple variable nodes in order to also express tuple correlations or conditional probabilities).

While such a factor graph is the basis of our data model, thus encapsulating the static probabilistic interactions between all data items, one of the novelties that we pursue on our work is to combine it with a data lineage model, in order to include a dynamic uncertainty layer on top of the factor graph. This way, we would provide a very generic uncertainty model in which we consider the probability distribution obtained from coupling both the intrinsic distribution of the given dataset (i.e., the factor graph), and the distribution generated dynamically during query processing with respect to the query expression (i.e., the lineage).

More formally, we will define *lineage* as a formula which captures the logical dependencies between a query answer and the facts from which it was derived (see Figure 3). As it was previously mentioned, first-order logic formulas have already been explored to express lineage; however, in this paper, for the sake of simplicity and following the majority of literature, we consider propositional formulas to represent lineage. A number of lineage models have been proposed in the database community in order to track data that is transformed by means of relational operations, such as sequences of updates and queries. These models vary in the kind of data (relational data, complex objects, XML), the class of queries (Relational Algebra, SQL, Datalog, XQuery), and the level of granularity of the recorded information. The underlying commonality in all these models is that the lineage information is typically generated via annotations of the individual data items.

A major goal to be accomplished in our project is to explore and develop a compilation technique to represent the kinds of formulas that we obtain through the lineage calculation in a succinct way. To this end, data structures like OBDDs (*Ordered Binary Decision Diagrams*) and related knowledge-compilation approaches such as SDDs (*Sentential Decision Diagrams*) [3] are state-of-art representation models for propositional formulas. However, they might not be directly adaptable to the lineage expressions obtained from query evaluations, and potentially expensive formula transformation costs must be taken into account.

3.2 Distributed Database Setting

However, as emphasized already in previous sections, efficient computation of marginal probabilties over a set of random variables remains a major computational challenge. For this reason, our goal is to implement such a probabilistic setting in a distributed fashion. Thus, we expect to obtain the benefits from a highly parallelized query execution and from applying distributed inference algorithms to achieve an improved performance on the marginalization process.

Given this overview, both data partitioning and indexing are clearly the first two issues that are to be addressed in order to enable query evaluations with asynchronous communication patterns and execution of the inference algorithms. In this sense, it must be taken into account that the factor graph adds another degree of complexity to the partitioning task. The partitioning algorithm must consider not only the data items but also their underlying correlations (i.e., factors connecting the variables in the factor graph). Otherwise, the communication among the compute nodes might heavily increase during inference computation time.

Several approaches to address these problems can be followed; currently, the focus of our research is on extending TriAD [5] – a fast, distributed RDF engine – in order to support the combined probabilistic data model that we are proposing. TriAD implements a replicated index with all possible triplet permutations (as many other RDF stores), and a hash-based, horizontal partitioning. In our adaptation, the factor graph is currently partitioned based on a horizontal form of data partitioning: each factor node is located at the slave where there are more variable nodes (i.e., data items) connected to it. Thus, the factor nodes are distributed by minimizing the amount of cut edges, thus decreasing the communication between slaves while performing probabilistic inference calculations.

On the other hand, and in order to solve the problem of the factor graph indexing, our purpose is to manage an in-memory distributed index based on distributed pointers, keeping direct references to the data items and factors, in such a way that unnecessary index look-ups can be avoided.

Regarding the query evaluation in a distributed probabilistic setting, besides the implementation of distributed relational operators (TriAD performs *distributed index scans*, *distributed merge* and *hash joins* and *projections*; a basic example is shown in Figure 2(a)), a key challenge consists of compactly encoding and propagating the lineage information along with the intermediate and final query answers through the compute network, and to embed this into a flexible communication protocol. To accomplish this, new augmented relational operators are to be implemented, allowing us to trace and construct the lineage formula of each individual query answer at query execution time.

3.3 Distributed Inference Algorithms

In order to take advantage of such a distributed system, we intend to develop probabilistic inference algorithms that specially take advantage of a decoupled, asynchronous execution layer. Concerning this, several implementations have been already discussed in previous sections. However, query processing and inference in probabilistic databases requires much more fine-grained, asynchronous communication patterns than what is available via the MapReduce paradigm. To this end, the Message Passing Interface (MPI) is an alternative framework for the exchange of messages among compute nodes – it is also used as basis for the GraphLab communication protocols [6], among others. The communication layer of MPI is tightly embedded into TCP/IP network protocol, allowing for the direct communication between two or more compute nodes via socket connections. Although it implies a higher level of effort, many graph algorithms and probabilistic inference techniques based on the variable elimination or the belief propagation algorithms are all much more suitable to such asynchronous communication patterns than to synchronous and long-running MapReduce iterations. Moreover, since we pursue a hybrid graphical model, we need distributed inference solutions for the combination between the *undirected factor graph* and the lineage structure, which conforms, as shown in Figure 3, to a *directed acyclic graph* (DAG).

4. CONCLUSION

We propose a project that aims to be highly innovative as it attempts to bridge the gap among PDBs and – and specially in a distributed setting – current Machine Learning approaches, which we believe has not been adequately investigated in the literature so far. To achieve our goals, we present, schematically, the following three main frontiers we intend to study.

- A combined, graph-oriented probabilistic model is proposed, which combines a static factor graph with a dynamic form of data lineage. We argue that this combined data model allows us to apply our approach to very generic problem settings, but at the same time it even more challenges us to investigate scalable solutions for query evaluation and probabilistic inference.
- Fitting such a probabilistic model into a distributed database system is presented as a core matter in our project. By employing main-memory-based, distributed index structures for the underlying factor graph, we intend to provide an appropriate distributed setup to scale out query evaluations under this data model.
- We consider several lines of research for the implementation of fast, distributed, asynchronous inference algorithms under our data model. Based on the Message Passing Interface, we aim to adapt and enhance the performance of existing inference techniques such as variable elimination and belief propagation.

5. REFERENCES

[1] 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.

[2] J. Cheney, L. Chiticariu, and W. C. Tan. Provenance in databases: Why, How, and Where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

[3] A. Darwiche. SDD: A new canonical representation of propositional knowledge bases. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 819–826, 2011.

[4] M. Dylla, I. Miliaraki, and M. Theobald. Top-k query processing in probabilistic databases with non-materialized views. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 122–133, 2013.

[5] S. Gurajada, S. Seufert, I. Miliaraki, and M. Theobald. TriAD: a distributed shared-nothing RDF engine based on asynchronous message passing. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 289–300, 2014.

[6] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Distributed GraphLab: A framework for Machine Learning in the cloud. *PVLDB*, 5(8):716–727, 2012.

[7] 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 *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*, pages 269–274, 2007.

[8] D. Olteanu and H. Wen. Ranking query answers in probabilistic databases: Complexity and efficient algorithms. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, pages 282–293, 2012.

[9] A. D. Sarma, M. Theobald, and J. Widom. Exploiting lineage for confidence computation in uncertain and probabilistic databases. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, México*, pages 1023–1032, 2008.

[10] P. Sen, A. Deshpande, and L. Getoor. PrDB: managing and exploiting rich correlations in probabilistic databases. *VLDB J.*, 18(5):1065–1090, 2009.

[11] P. Sen, A. Deshpande, and L. Getoor. Read-once functions and query evaluation in probabilistic databases. *PVLDB*, 3(1):1068–1079, 2010.

[12] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[13] G. Weikum and M. Theobald. From information to knowledge: harvesting entities and relationships from web sources. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 65–76, 2010.