ActivePDB: Active Probabilistic Databases

Osnat Drien
Bar-Ilan University

Matanya Freiman
Bar-Ilan University

Yael Amsterdamer
Bar-Ilan University

Abstract

We present a novel framework for uncertain data management, called ActivePDB. We are given a relational probabilistic database, where each tuple is correct with some probability; e.g., a database constructed from textual data using information extraction. We are now given a query and we want to determine the correctness of its results. Unlike probabilistic databases, we have an oracle that can resolve the uncertainty, such as a domain expert that can verify data against their sources. Since verification may be costly, our goal is to determine the correct output of the query, while asking the oracle to verify as few tuples as possible. ActivePDB provides an end-to-end solution to this problem. In a nutshell, we first track provenance to identify which input tuples contribute to the derivation of each output tuple, and in what ways. We then design an active learning solution to iteratively choose tuples to be verified based on the provenance structure and on an evolving estimation of the probability of the tuples correctness. We will demonstrate ActivePDB in the context of the NELL database of extracted facts, allowing participants to both pose queries and play the role of oracles.

1 Introduction

Many scenarios involve data whose correctness is uncertain. A common approach in these cases is to represent the information in a *probabilistic database* [12], namely a database in which every tuple is associated with a probability value reflecting the estimated likelihood that it is correct. An example for a large-scale probabilistic database is that of NELL (`http://rtw.ml.cmu.edu/rtw/`), consisting of 50M facts that were automatically extracted from different websites, each along with a confidence level.

There are multiple approaches for using uncertain data. One may attempt to *clean* the database, namely to retain only correct tuples, using fully automated techniques (e.g. [11]) or involving the help of experts or crowd workers (e.g., [4]). Cleaning large databases such as NELL can still be an infeasible task, unless certain conditions hold (e.g., all incorrect tuples violate certain types of constraints). Alternatively, one may work directly with the uncertain data, propagating the uncertainty to the query results [12, 13] and quantifying their uncertainty or seeking consistent answers across all possible worlds [10]. In some cases, however, we are required to resolve the uncertainty: e.g., when query results are mission-critical or when automated techniques are insufficiently certain. In such cases we would like to efficiently determine the correct query results while leveraging uncertainty information.

In this work, we lay the foundations for a novel hybrid approach, which we term *Active Probabilistic Databases*. We are given both an uncertain database and *an oracle* that is able to resolve the uncertainty for any given tuple, e.g., a domain expert or workers recruited through a crowdsourcing platform. For a query of interest, we aim to verify the correctness of *as few database tuples as possible* that would still allow us to *produce correct (non-probabilistic) query answers*. In this sense, we differ from works on query evaluation that work with a probabilistic database without aiming to resolve it [12, 13] or that clean data without a probabilistic underlying model [2]. In our framework, the tuples to verify are selected based on the connection between query input and output and the correctness probabilities. We iteratively select tuples and send verification requests to the oracle until the correct answers are determined.

As we send input tuples to be verified, we also gradually refine the probability estimation with respect to other, similar input tuples (e.g., extracted from the same source). We thus combine correctness verification with *active learning* of correctness probabilities.

Consider, for a simple example, a selection-projection (SP) query. In this case, to verify the correctness of a given output tuple, it suffices to identify one correct input tuple yielding it, or establish that all such tuples are incorrect. Strategically, it may thus be advisable to first verify a tuple with a high probability. This should be weighed along with (1) input tuples contribution to resolving correctness of other output tuples and (2) their contribution to the quality of probability estimation w.r.t. other relevant input tuples. Naturally, the analysis is more intricate for more complex queries.

Solution Overview We provide an end-to-end solution for Active Probabilistic Databases, focusing on the class of SPJU queries. The architecture of our solution is depicted in Figure 1. The user issues (step (1) in the Figure) a query over the database, and the *Query Evaluation* module yields the query results along with their *provenance* (step (2)). This provenance has the form of *Boolean expressions*, whose variables correspond to tuples have contributed to the derivation of each query result [7]. Determining the correctness of a query result now amounts to determining the truth value of the corresponding provenance expression, by (frugally) probing the oracle to reveal truth values of the Boolean variables occurring in it (see Section 3). To this end, the query results and their provenance are then passed to the *Manager Module*. The Manager also has access to past probes and answers from the *Probes Repository*. Such known probe answers are assigned to the relevant variables, and the expressions are simplified accordingly (step (3)). Then (Step (4)), the Manager performs *Active Query Evaluation*, to choose the next probe (see below). The oracle is probed to verify the selected tuples (step (5)). In line with human-in-the-loop data cleaning processes [4] and human-powered databases [2, 9], the oracle is considered costly and may introduce higher latency; yet, it performs a task that cannot be done automatically (or not with sufficient accuracy). Thus, carefully choosing the requests to the oracle is critical. The oracle answers are recorded in the Probes Repository. If the (in)correctness of some output tuples are now resolved, the user is notified about the (in)correctness of corresponding tuples. Steps 3-6 repeat to choose subsequent probes.

The three modules that combine to implement Active Query Evaluation

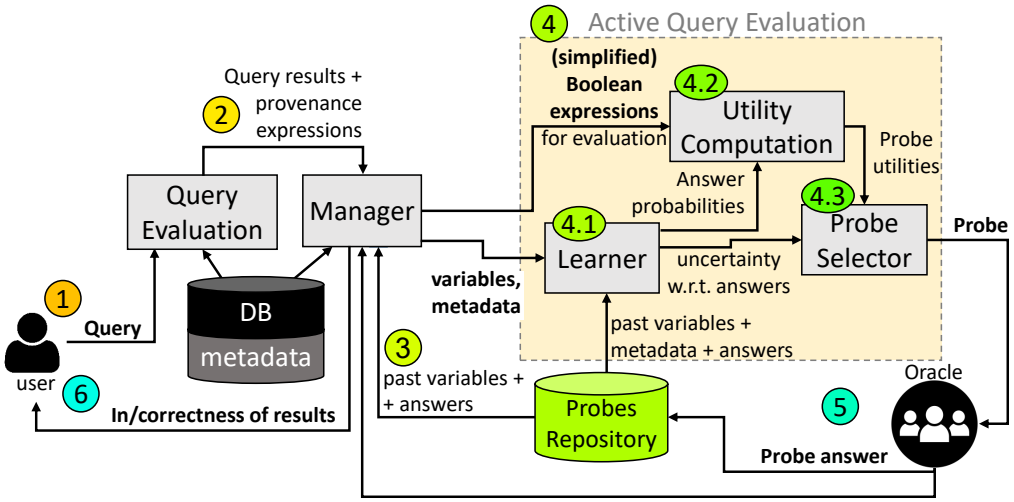

Figure 1: Framework architecture

(Step (4)) are our key technical contributions; they are described briefly next, and in more detail in the following sections.

Learner (Section 3.2) An informed selection of oracle probes (i.e. input tuples to be verified) relies on the estimation of probe answer probabilities. Our framework includes a Learner module trained for this task over past and incoming probe answers. For features, the learner uses metadata such as the website from which the tuple was extracted. We further use a novel approach [8] to quantify the expected reduction in the uncertainty of our estimation, yielded by a candidate probe. This quantity is used as one of the criteria for probe selection, see below.

Utility Computation (Section 3.3) Given (estimated) probe answer probabilities, we compute the expected utility of each possible probe towards deciding the truth value of the provenance expressions. For that, we adapt a suite of existing algorithms from the literature on *Interactive Boolean Evaluation*, originally intended to evaluate a single Boolean formula. Our adaptations include (1) quantifying the utility of all possible probes rather than choosing a single probe,since uniquely in our case, utility is not the only factor affecting probe selection; and (2) computing the joint utility with respect to multiple expressions (provenance for multiple tuples) affected by each probe.

Probe Selector (Section 3.4) Each possible probe is assigned a utility value and an uncertainty error reduction value (outputted by the Learner). The Probe Selector Module addresses this exploration-exploitation problem: exploitation corresponds here to progressing towards revealing whether tuples are correct, and exploration corresponds to improving the quality of our probability estimation with respect to future probe answers.

Acquisitions

Acquired	Acquiring	Date	
A2Bdone	Zazzer	7/11/2020	a_0
microBarg	Fiffer	1/5/2017	a_1
fPharm	Fiffer	1/2/2016	a_2
Optobest	microBarg	8/8/2015	a_3

Education

Alumni	Institute	Year	
Usha Koirala	U. Melbourne	2017	e_0
Pavel Lebedev	U. Melbourne	2017	e_1
Nana Alvi	U. Sau Paolo	2010	e_2
Nana Alvi	U. Melbourne	2017	e_3
Gao Yawen	U. Sau Paolo	2010	e_4
Amaal Kader	U. Cape Town	2005	e_5

Roles

Organization	Role	Member	
A2Bdone	Founder	Usha Koirala	r_0
A2Bdone	Founding member	Pavel Lebedev	r_1
A2Bdone	Founding member	Nana Alvi	r_2
microBarg	Co-founder	Nana Alvi	r_3
microBarg	Co-founder	Gao Yawen	r_4
microBarg	CTO	Amaal Kader	r_5

Table 1: Example annotated database.

Related Work Our work relates to Probabilistic Databases (e.g., [12, 13]) in the probabilistic relational model, as well as the use of provenance [7]. Otherwise, our techniques and analysis are quite different: resolving truth values enables active probability learning and expression simplification that are not possible when uncertainty persists. Consistent Query Answering (CQA) (e.g., [10]) is similar to our approach in attempting to resolve the uncertainty, but differs in looking for answers that appear in the query result with respect to *all* possible worlds, rather than on a ground-truth world as we do (and so CQA results are a subset of ours). The involvement of oracles connects our work to human-powered databases (e.g., [9]) and data cleaning (e.g., [2, 4]). In particular, [2] uses crowd feedback to fix a given query's output by editing the input database; this differs from our focus on query output validation. To our knowledge, ActivePDB is the first general-purpose framework that allows to start with an uncertain database and get to certain query answers by actively procuring tuple verification, alongside actively learning correctness probabilities.

1.1 Probabilistic Databases and Possible Worlds

A probabilistic database [12] is represented as $\bar{D} = (D, X, L, \pi)$ where D is a relational database, X is a set of Boolean variables, $L : \text{tuples}(D) \to X$ maps tuples to variables that annotate them, and $\pi : X \to [0, 1]$ reflects the (marginal) probability that the corresponding tuple is correct. Unlike standard probabilistic databases, π may initially be unknown, in which case it is learned (see below).

2 Model

Example 2.1. *Table 1 outlines a probabilistic database with three relations:* Acquisitions*, including data on companies acquired by other companies;* Roles*,*

```
SELECT DISTINCT a.Acquired, e.Institute
FROM Acquisitions AS a, Roles AS r, Education AS e
WHERE a.Acquired = r.Organization AND
      r.Member = e.Alumni AND a.Date >= 2017.01.01 AND
      r.Role LIKE '%found%' AND e.YEAR <= year(a.Date)
```

Figure 2: Query over the example database

including data on roles of different organization members; and Education*, including data on university alumni. The right-most column shows the variable X annotating the tuple, which π associates with some correctness probability.*

A truth valuation $\text{val} : X \to \{\text{True}, \text{False}\}$ yields a *possible world* $D_{\text{val}} \subseteq D$, namely, the subset of database tuples whose variables have been mapped to True. Given also a query Q over $\bar{D}$, a tuple t is said to be correct w.r.t. val if t is in the output of $Q(D_{\text{val}})$.

Example 2.2. *As part of seeking promising entrepreneurs, an analyst may issue the SPJU query in Figure 2 over the DB in Table 1, which returns companies acquired since 2017 along with institutes in which founders of these companies had studied. In the event that the first two tuples in the Acquisitions relation are incorrect (*$\text{val}(a_0) = \text{val}(a_1) =$ *False) the query has no correct results. If, alternatively, the first tuple of each of the three relations is correct (i.e.,* $\text{val}(a_0) = \text{val}(r_0) = \text{val}(e_0) =$ *True), the query result tuple* `(A2Bdone, U. Melbourne)`*, derived from these input tuples, is correct.*

2.1 The Active Model

Next, we diverge from the standard probabilistic databases model in that we aim to reveal the correct query results by probing an oracle. We assume a ground truth valuation val^* reflecting the real correctness of input tuples; val^* may be (fully or partially) unknown and its assignments can be discovered by oracle probes, where each probe reveals $\text{val}^*(x)$ for a specific $x \in X$. When val^* is partially known, we may use the known values and *meta-data attributes* to estimate correctness probability for the remaining tuples. We use $\mathcal{A}(\bar{D})$ to denote meta-data which may, e.g., include schema information, properties of the data source, etc.

Problem definition We are given a probabilistic database $\bar{D}$, (possibly without the probability distribution π), metadata $\mathcal{A}(\bar{D})$ about $\bar{D}$'s tuples and an SPJU query Q over $\bar{D}$. We assume an unknown ground-truth valuation val^* drawn from π. We then iteratively select a tuple $t \in \bar{D}$ and verify its correctness, i.e., discover $\text{val}^*(x)$ for $x = L(t)$, through an oracle call. Our goal is to verify the correctness of output tuples in $Q(\bar{D})$ while making a minimal number of probes to the oracle in expectancy w.r.t. π. If π is unknown, we may estimate it based on the accumulating probe answers along with their metadata in $\mathcal{A}(\bar{D})$.

3 Solution Architecture

We next elaborate on the components of ActivePDB.

Acquired	Institute	
A2Bdone	U. Melbourne	$(a_0 \wedge r_0 \wedge e_0) \vee (a_0 \wedge r_1 \wedge e_1) \vee (a_0 \wedge r_2 \wedge e_3)$
A2Bdone	U. Sau Paolo	$(a_0 \wedge r_2 \wedge e_2)$
microBarg	U. Melbourne	$(a_1 \wedge r_3 \wedge e_3)$
microBarg	U. Sau Paolo	$(a_1 \wedge r_3 \wedge e_2) \vee (a_1 \wedge r_4 \wedge e_4)$

Table 2: Result of the example query.

3.1 Provenance

A key component in our solution is that of Boolean provenance [7], computed alongside query evaluation (step 2 in Figure 1). Briefly, given a probabilistic database $\bar{D} = (D, X, L, \pi)$ and a query Q, we compute the representation $Q(\bar{D}) = (Q(D), \text{Bool}[X], L')$ that consists of the "standard" query result and a labeling function L' that maps each output tuple t to a Boolean expression over the input annotations. Crucially, L' has the property that a valuation val satisfies $L'(t)$ if and only if t is correct w.r.t. val. We focus on SPJU queries, whose provenance can be computed in polynomial time as monotone k-DNF expressions, i.e., without negation and where terms (conjunctions) include at most k variables.

Example 3.1. *The result of applying the query from Figure 2 to the DB in Table 1 is shown in Table 2, along with its provenance in 3-DNF. In line with Example 2.2, since every provenance term contains either a_0 or a_1, if* $\text{val}(a_0) = \text{val}(a_1) =$ *False then all four expressions are False, i.e., every output tuple is incorrect.*

After provenance is computed, ActivePDB substitutes variables with known probe answers recorded in the Probes Repository, if exist, and simplifies the expressions accordingly (step 3 in Figure 1). Next, we explain the use of such expressions in selecting the next probe, by the Active Query Evaluation modules (step 4 in Figure 1).

3.2 Learner

Each answer from the oracle reveals the correctness of a specific input tuple, but, as explained above, we also use it to estimate the correctness of other tuples. Our Learner (step 4.1 in Figure 1) is trained over each such incoming answer along with metadata of the resolved tuple, and has two types of output, per candidate probe: (1) the *estimated probability* for a True probe answer, needed for estimating its effect on correctness computation; and (2) the *expected uncertainty reduction* if we train the Learner with this probe's answer, leading to improved probability estimation in the next probe selection. The first output depends on the concrete Learner used. For instance, in the demonstration we train a Random Forest Classifier, where the probability of that t is correct is the fraction of trees which classify t as *correct*. The second output follows an *active learning* paradigm, where training data is actively chosen by the Learner to improve the model's accuracy, and is used as one of the criteria in selecting probes. For quantifying uncertainty reduction, we use an advanced approach called Learning Active Learning (LAL) [8]. Briefly, LAL uses a regressor that is

trained on an annotated dataset (possibly from another domain). This regressor is transferred to predict the expected error reduction for an instance in a specific learning state, in our case, a candidate probe and a trained classifier, treating it as a regression problem.

3.3 Probe Utility Computation

Using the probe answer probabilities, the Utility Computation module (step 4.2 in Figure 1) estimates the utility of each candidate probe towards deciding the correctness of query output tuples. This problem is closely related to *Interactive Boolean Evaluation*, namely, revealing the truth value of a given Boolean formula by probing truth values of its variables. Work in this area generally considers Boolean Evaluation *in isolation*, namely, assumes fixed and known variable probabilities and a single Boolean formula. We thus cannot directly use them. In [6], by the present authors, such algorithms were adapted to the context of consent management and in particular to support multiple formulas in a query output's provenance. In contrast, in the present work Boolean Evaluation is not the only consideration in choosing among candidate probes. To allow combining it with other factors we cast these algorithms through a utility function assigning a score to *every* candidate probe.

We briefly and intuitively mention three utility functions we derived, where each is an adaptation of an algorithm compatible with the provenance shape of a specific query class.

1. A function greedily selecting the term and variable within it to probe, inspired by [3], which allows us to optimize probe selection for queries tyielding read-once provenance, e.g., S/SP/SU queries (details omitted).
2. A function based on the `Q-Value` approach of [5], which typically yields near-optimal probe selection but requires both the DNF and CNF representations of Boolean formulas. Hence, it is suitable e.g. for projection-free (SJU) queries, where each output tuple depends on a bounded number of input tuples (see more details below).
3. A function adapted from [1], which is approximately-optimal for a single output tuple of any SPJU query, and empirically performs well for multiple results (details omitted).

We elaborate, for example, on the `Q-value` utility function, intuitively based on the number of DNF terms and CNF clauses that each probe is expected to resolve. At each point, we have a partial valuation val capturing the answers we got so far. For a variable x, let $\text{val}_{\text{x=True}}$ be the valuation identical to val except that it also assigns True to x (and symmetrically for False). Let Φ be a set of Boolean formulas and given a formula $\varphi \in \Phi$, denote by $\text{nt}(\varphi)$ and $\text{nc}(\varphi)$, respectively, the number of DNF terms and CNF clauses in φ. Denote by $\text{val}(\varphi)$ the formula obtained by replacing variables in φ with their assignment in val. The Q-Value utility function can then be written as: $\text{util}(\Phi, \pi, \text{val}, x) = \sum_{\varphi \in \Phi} \{\text{nt}(\varphi) \cdot \text{nc}(\varphi) - \pi(x) \cdot \text{nt}\big(\text{val}_{\text{x=True}}(\varphi)\big) \cdot \text{nc}\big(\text{val}_{\text{x=True}}(\varphi)\big) - (1 - \pi(x)) \cdot \text{nt}\big(\text{val}_{\text{x=False}}(\varphi)\big) \cdot \text{nc}\big(\text{val}_{\text{x=False}}(\varphi)\big)$.

3.4 Probe Selector

The utility and the uncertainty reduction scores outputted by the above two modules, are fed into the Probe Selector (step 4.3 in Figure 1), whose role is

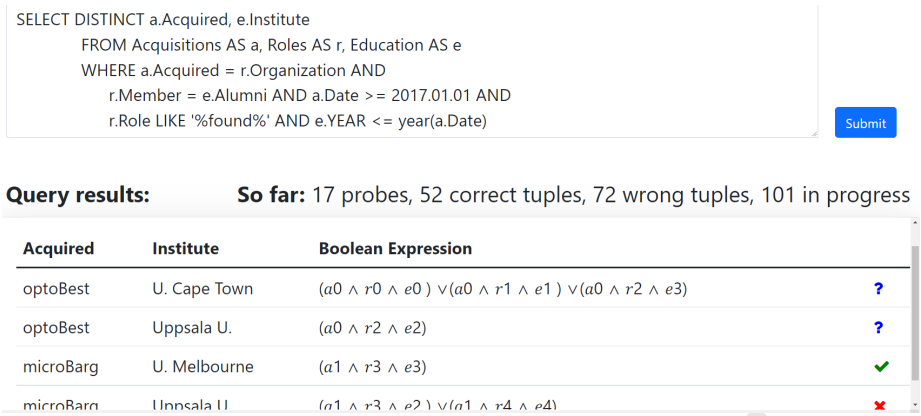

Figure 3: Query results screen.

to balance these two factors. Unlike classic exploration-exploitation problems such as multi-armed bandit, exploitation in our case has a different flavor, since each probe eliminates some candidate probes (in particular, the same probe is never issued twice). Our solution is to assign a score to each possible probe that combines the probe's estimated utility (denoted x) and its effect on uncertainty (denoted y). An example commonly used to combine rankings in IR is $f(x, y) := x \cdot (y + 1)$.

4 Implementation and Demo Scenario

ActivePDB is implemented using the Scikit-learn package (`https://scikit-learn.org`) in the Learner, boolean.py (`https://pypi.org/project/boolean.py/`) to handle Boolean operations, and MongoDB (`https://www.mongodb.com/`) for the database. We support multiple implementations for each of its modules, and in particular, support the three utility functions described in Section 3.3. Our implementation also incorporates several optimizations, including parallel implementation of the utility computation and efficient CNF computation needed for the utility function Q-Value. We will demonstrate ActivePDB in the context of the NELL database mentioned above. We will issue different queries over the data, on different topics such as company acquisition and sports. We will execute the evaluation process step-by-step, and inspect the gradual change of underlying model parameters, as follows.

First, we will ask the audience to select a query to be issued over the data, track the progress of verifying query results (as in Figure 3) while a simulated expert answers probes about the query. Initially, for all the output tuples, correctness is unknown (marked by blue question marks); but as probe answers accumulate, more tuples are resolved as (in)correct (green checkmark /red "X").

Next, we will change the system configuration (e.g., to use a different utility function), and switch off automated probe answering. We will use a split screen to give both insights on the underlying model and the probe answering interface: the first frame will show, as before, the output tuples, this time along with their Boolean expressions (right of Figure 3). The second frame will show, for each unresolved variable, properties such as its current probability estimated by our Learner and its current uncertainty reduction estimation. We will be able

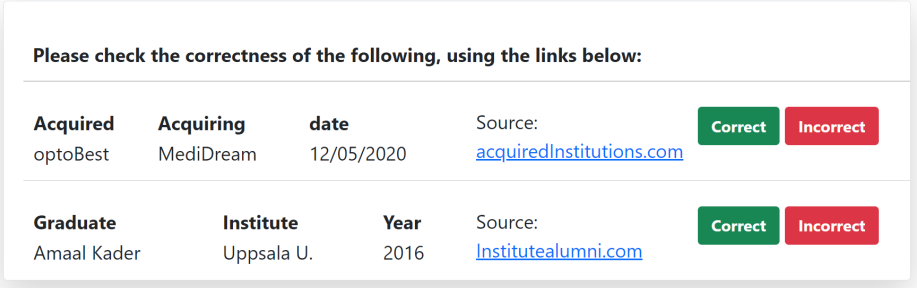

Figure 4: Manual correctness verification screen.

to observe the gradual convergence of probability estimation and uncertainty reduction. The third frame will sequentially display the oracle questions in the form of Figure 4, where tuple correctness can be verified by referring to the source from which they were extracted. We will feed in different answers and show their effect on the progress of determining correctness.

References

[1] S. R. Allen, L. Hellerstein, D. Kletenik, and T. Ünlüyurt. Evaluation of monotone DNF formulas. *Algorithmica*, 77(3), 2017.

[2] M. Bergman, T. Milo, S. Novgorodov, and W. C. Tan. Query-oriented data cleaning with oracles. In *SIGMOD*, 2015.

[3] E. Boros and T. Ünlüyurt. Sequential testing of series-parallel systems of small depth. In *Computing tools for modeling, optimization and simulation*. Springer, 2000.

[4] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. KATARA: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD*, 2015.

[5] A. Deshpande, L. Hellerstein, and D. Kletenik. Approximation algorithms for stochastic Boolean function evaluation and stochastic submodular set cover. In *SODA*, 2014.

[6] O. Drien, A. Amarilli, and Y. Amsterdamer. Managing Consent for Data Access in Shared Databases. In *ICDE*, 2021.

[7] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, 2007.

[8] K. Konyushkova, R. Sznitman, and P. Fua. Learning active learning from data. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *NIPS*, 2017.

[9] G. Li, C. Chai, J. Fan, X. Weng, J. Li, Y. Zheng, Y. Li, X. Yu, X. Zhang, and H. Yuan. CDB: A crowd-powered database system. *PVLDB*, 11(12), 2018.

[10] X. Lian, L. Chen, and S. Song. Consistent query answers in inconsistent probabilistic databases. In *SIGMOD*, 2010.

[11] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. HoloClean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11), 2017.

[12] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Morgan & Claypool, 2011.

[13] G. Van den Broeck and D. Suciu. Query processing on probabilistic data: A survey. *Found. Trends DBs*, 7(3-4), 2017.