Managing Consent for Data Access in Shared Databases

Osnat Drien
Bar-Ilan University

Antoine Amarilli
LTCI, Télécom Paris, Institut Polytechnique de Paris

Yael Amsterdamer
Bar-Ilan University

Abstract

Data sharing is commonplace on the cloud, in social networks and other platforms. When a peer shares data and the platform owners (or other peers) wish to use it, they need the consent of the data contributor (as per regulations such as GDPR). The standard solution is to require this consent in advance, when the data is provided to the system. However, platforms cannot always know ahead of time how they will use the data, so they ofter require coarse-grained and excessively broad consent. The problem is exacerbated because the data is transformed and queried internally in the platform, which makes it harder to identify whose consent is needed to use or share the query results. Motivated by this, we propose a novel framework for actively procuring consent in shared databases, focusing on the relational model and SPJU queries. The solution includes a consent model that is reminiscent of existing Access Control models, with the important distinction that the basic building blocks – consent for individual input tuples – are unknown. This yields the following problem: how to *probe* peers to ask for their consent regarding input tuples, in a way that determines whether there is sufficient consent to share the query output, while making as few probes as possible in expectation. We formalize the problem and analyze it for different query classes, both theoretically and experimentally.

1 Introduction

Data is routinely being shared online by peers on different platforms including social networks, cloud-based file sharing, and messaging applications. Using this data requires the consent of the data owner, according to regulations such as GDPR [1]. Typically, consent is obtained in advance, e.g., when the peer joins the platform (accepting the general platform policy) or when the data is originally uploaded to the platform. This has two main limitations:

1. The future ways in which data is used may be unknown and hard to anticipate. Consequently, a-priori consent is coarse-grained since we may

not know yet with which third parties it would make sense to share the data.

2. To share original data as-is, one may simply ask the data owner for their consent when the need arises. However, data is often transferred, processed and combined through complex queries. Now, if one wishes to use a query result, they are often not directly represent the data of any individual user; but on the other hand, the owners of the data from which the result has been derived do have rights with respect to it [1].

Similar questions are studied in the fields of data sharing, privacy and access control (see Section 5), but to our knowledge there has been no study of the combination of these two challenges, namely the need to actively obtain consent and to do so for derived data.

We therefore propose a *novel framework for managing consent in shared databases*, focusing on the relational setting and Select-Project-Join-Union (SPJU) queries. The main components of this framework are as follows.

Consent Model for Derived Data As mentioned above, it is unclear who "owns" pieces of data that are derived through data transformations. In the context of relational databases, we use a model based on a "possible-worlds-like" semantics: for a database D, a query Q and a query result $t \in Q(D)$, we say that consent is given with respect to t iff t appears in the result of evaluating Q over $D' \subseteq D$ containing the input tuples for which consent is given.

So far, this choice is in line with standard access control models (e.g., [2, 3, 4, 5]). However, in our setting, it is also unknown if we have consent with respect to the *input tuples* – see point (1) above. Thus, we cannot simply propagate known consent from the input to the output, but we need to actively *probe* peers to ask for their consent. We use the notion of probing in an abstract sense, and in practice it may correspond to questions posed to human users [6], requests through automated agents, etc. Data owners may give or deny consent based on social, business or other motives, such as reciprocal data sharing.

Now, whom should we probe? Of course, if we probe all peers with respect to all data in the shared database, we would know precisely the consent status with respect to the entire input database and thereby for any output as well. Probing, however, is a costly operation – in terms of latency and human effort. We thus wish to probe in a frugal manner, which leads to our problem (defined formally in Section 2): *how to minimize the expected number of probes required to determine consent with respect to a query result?* We say "expected" (in the probabilistic sense), since the number of probes needed in total will typically depend on the answers that we receive, which of course are not known in advance.

Solution Framework Towards a solution, we first observe that a key ingredient is *provenance*. The provenance of a query result tells us which input tuples it depends on, and in what way. In settings where we know whether consent is granted for each input tuple, the Boolean provenance of the query output (à la c-tables [7]) tells us whether we have consent to share the query results. Furthermore, when we do not know the consent for the input, we may annotate each input tuple with a variable, and provenance is then Boolean expression over these variables. The problem then reduces to determining the truth value

for these Boolean expressions, by probing the truth values of the individual variables. We refer to this problem as *Interactive Boolean Evaluation*, which has been extensively studied in the literature in different contexts and under different names (e.g., [8, 9, 10, 11]). Our solution thus "marries" provenance and Boolean Evaluation, leading to different results for different query classes.

Complexity Analysis and Algorithms We analyze the complexity of our problem for subclasses of SPJU queries. For S/SP/SU queries and for SJU queries we show, respectively, an exact solution algorithm based on [9] and an approximate solution based on [10, 12], works on Interactive Boolean Evaluation which we adapt to SPJU query provenance.

Then, we consider the general class of SPJU queries, and show a solution algorithm adapted from [8]. Here, we can show that optimal solution is NP-hard in data complexity, even for sharing a single tuple and already for SPJs.

Experimental Results To our knowledge, there have been no practical implementations, experimental studies or benchmarking close to our setting, let alone for our problem formulation, which is new. We have thus implemented all algorithms for the different variants and developed a benchmark that allows examining their performance over provenance expressions of various shapes. Our algorithms exhibited superior performance beyond their theoretic guarantees and compared to the baseline algorithms, and were (near-)optimal where the optimal solution was known.

For lack of space, full proofs and additional results and experiments are deferred to [13].

2 Model

We assume familiarity with standard relational database terminology [14]. Let $\mathcal{C}$ be a set of Boolean variables called *consent variables*. A *shared database* is a relational database D along with a one-to-one labeling function $L : \text{tuples}(D) \mapsto \mathcal{C}$ where each tuple is annotated by a (unique) variable from $\mathcal{C}$, intuitively controlling its consent status.

The premise is that there is a hidden truth value to whether or not we are allowed to share each tuple (in a specific context, e.g., with a given third party).

Definition 2.1. *A* consent valuation *is a function* $\text{val} : \mathcal{C} \mapsto \{\mathit{True}, \mathit{False}\}$.

Next, instead of sharing the data as-is, we allow a *query* executed on the database to perform some analysis and consider the sharing of its results. Most aspects of the model may apply to any query language, but we will focus in particular on SPJU (Select-Project-Join-Union) queries [14].

The question is then: given a shared database and an SPJU query, are we allowed to share the result? In the spirit of previous work on incomplete databases [7], we define that a query answer is shareable iff it appears in the result of evaluating the query over the sub-database consisting only of the shareable input tuples.

Definition 2.2. *Given a shared database* $\bar{D} = (D, L)$*, an SPJU query* Q*, a tuple* $t \in Q(D)$*, and a consent valuation* val*,* t *is* shareable *iff* $t \in Q(D')$*, where* $D' \subseteq D$ *is the database consisting of every* $t' \in D$ *such that* $\text{val}(L(t')) = \text{True}$.

The challenge, of course, is that we may not know some or all of the consent values, which are initially known only to data owners. To find out which query results can be shared, we can pose questions or *probes* to the owners of relevant tuples, asking whether the tuple can be shared. Our goal is to optimize the number of questions posed in order to determine which query results can be shared. In order to measure the performance of an algorithm, which usually depends on the outcome, we look at the probability of the answer of a variable being set to True, assuming independence between variables. The performance of a probing algorithm is then the expected number of probes that it makes.

Problem Definition The problem of OPT-PEER-PROBE (for "optimizing peer probing") is defined as follows, where we use italics to highlight some problem design choices that deferred to [13]:

Definition 2.3 (OPT-PEER-PROBE)**.** *We are given a shared database $\bar{D}$, a probability distribution π over consent variables in $\bar{D}$, and an SPJU query Q. A consent valuation* val *is drawn at random according to π, but is not given to the algorithm. The problem* OPT-PEER-PROBE *is to define an algorithm that can decide for* every *tuple $t \in Q(D)$ whether it is shareable or not under* val*. To reveal that, at each step, the algorithm may probe a consent variable x of its choice and obtain* val(x) *as a result. We assume probes are* sequential*, i.e., posed one at a time so that their choice may depend on previous answers. For a (hidden) choice of* val*, the algorithm's total cost is the* total number of probes*. The optimization target is to minimize the algorithm's* expected cost over the random choice of val*.*

Solution Overview Our solution advantages techniques from two existing areas, namely relational database provenance and Interactive Boolean Evaluation. The first step is computing for each tuple in the query result a monotone Boolean formula over $\mathcal{C}$ that reflects the way it was derived from the input. The construction of these formulas follows the provenance semirings approach [15] (using the PosBool$[X]$ semiring where in our case $X = \mathcal{C}$). Intuitively, the union of two relations corresponds to taking the disjunction of the annotations of tuples appearing in both relations. Similarly, multiplication corresponds to joint derivation, thus, a tuple appearing in the result of a join will be annotated with the conjunction of the annotations of the two joined tuples. These formulas may be computed in PTIME and their truth value under any consent valuation correctly reflects if the output tuple can be shared or not. Our problem can now be rephrased as that of probing variables to infer the truth value of these Boolean expressions. As mentioned above, this is related to the area of *Interactive Boolean Evaluation*, whose results [8, 9, 10, 12] we will leverage. As explained in the Introduction, the crux lies in (1) characterizing query classes that yield particular shapes of Boolean expression for which evaluation may be done efficiently and (2) simultaneously evaluating multiple expressions corresponding to many output tuples.

3 Complexity and Algorithms

3.1 General Characterization

In this section, we consider different classes of queries, starting from a class for which we show an exact optimal solution, and ending with general SPJU queries in Section 3.4. Generally, the provenance of SPJU queries can be characterized based on the notion of *k-DNFs*.

Definition 3.1. *Let k be a constant. A Boolean formula is in k-DNF if (a) it is in a* Disjunctive Normal Form *(disjunction of conjunctions) and (b) the size of each term, i.e., the number of (distinct) variables in a conjunction, is bounded by k.*

In fact, we can show a two-way correspondence, meaning that every k-DNF formula is the provenance of some SPJU query result and vice versa [13]. For subclasses of SPJU, the provenance may thus be a particular subclass of k-DNF formulas, per tuple. For convenience, we extend the notion of valuation to unknown consent values.

Definition 3.2. *A* partial consent valuation *is an assignment of truth values to $\mathcal{C}$,* val : $\mathcal{C} \mapsto \{True, False, Unknown\}$*, where Unknown stands for unknown consent value. Any valuation can be extended to Boolean expressions over $\mathcal{C}$ via* Kleene three-valued logic*, e.g., $True \wedge Unknown = Unknown$, etc.*

3.2 Read-once

In Interactive Boolean Evaluation, the class of read-once DNF formulas, where each variable occurs only once, is known to have an exact (i.e., optimal) solution that minimizes the number of probes [9]. Algorithm RO (outlined in Algorithm 1) generalizes this technique to solve OPT-PEER-PROBE given multiple formulas and supporting variable repetitions, assuming the provenance of the query output is computed according to Section 2. I.e., the algorithm can be applied to *any* monotone DNF provenance, but optimality is only guaranteed for read-once DNF. The algorithm chooses the term (not evaluated yet) with the highest fraction of probability over size. It then probes the variables in increasing order of probability for an affirmative answer, until the term is evaluated (and if it is True so are the expressions containing it). Efficient min/max computation can be supported by efficient data structures for terms and their corresponding variables.

While previous work on provenance considered read-once provenance (e.g., [16, 17]), to our knowledge there has been no characterization of which queries always yield read-once *DNF*. Still, we can identify concrete practical subclasses of SPJU that have this guarantee.

Proposition 3.3. *Algorithm* RO *is a PTIME exact solution to* OPT-PEER-PROBE *for S/SP/SU queries (Selection/Selection-Projection/Selection-Union) over shared databases.*

By construction, the provenance of each output tuple for such queries is a disjunction of variables, disjoint between tuples – i.e., *overall read-once*.

Input: $X = \{x_0, x_1 \ldots\}$ – a set of variables,
$\pi : X \to [0, 1]$ the probability of each variable to be True,
dnfs – m monotone DNF formulas over X.
val $\leftarrow$ partial valuation setting all $x \in X$ to Unknown;
while $\exists \psi \in$ dnfs, $\text{val}(\psi) = \textit{Unknown}$ **do**
  $T \leftarrow \{t \in \text{terms}(\psi) \mid \psi \in \text{dnfs} \wedge \text{val}(t) = \text{Unknown}\}$;
  $t \leftarrow \arg\max_{t' \in T} \frac{1}{|t'|} \prod_{x \in t' \wedge \text{val}(x) = \text{Unknown}} \pi(x)$;
  $x \leftarrow \arg\min_{x \in t} \pi(x)$;
  $b \leftarrow$ probe x;
  val $\leftarrow$ val setting $x := b$;
return $[\text{val}(\varphi_1), \ldots, \text{val}(\varphi_m)]$ *for* dnfs $= [\varphi_1, \ldots, \varphi_m]$.

Algorithm 1: Algorithm RO for OPT-PEER-PROBE

In the multi-expression setting, we distinguish the more restrictive case of overall RO from cases where each individual tuple has read-once DNF provenance, but variables may repeat across tuples, which we term *per-tuple RO*. In [13] we consider queries that guarantee per-tuple read-once provenance, and show OPT-PEER-PROBE is hard for this class.

3.3 Queries without Projection

The next fragment that we consider is that of *projection-free queries*, i.e., SJUs. Importantly, for such queries, the provenance size for each individual tuple is in fact *independent of the database size* – the number of terms is bounded by the number of unions, and the size of terms is bounded by the number of joins in a single conjunctive query.

We will use this property to design an approximation algorithm, Q-value, whose details appear in Algorithms 2 and 3, and which incorporates some techniques and takes its name from [10]. The main idea is as follows: given both the CNF and DNF of some formula, termed *CDNF* formulas, compute exactly how many DNF terms (conjunctions) and CNF clauses (disjunctions) are eliminated (evaluated to True/False) if a given variable value is True or False, respectively. A greedy selection of concepts as a specific function of the expected number of clauses/terms they eliminate, termed *Q-value*, yields an expected number of probes which approximates the optimal solution. This is shown in [10] based on [12]. Algorithm 3 is the overall solution using Algorithm 2, first translates the input DNF formulas into CNF.

Proposition 3.4. *Let Q be a projection-free query and $\bar{D}$ a shared database. The* Q-value *Algorithm is a* $\mathrm{O}(\left|\bar{D}\right|^{|Q|})$ *time* $\log \left|Q(\bar{D})\right|$*-approximation for* OPT-PEER-PROBE.

The approximation ratio follows from the shape and size of the computed provenance for this fragment, and from the logarithmic approximation ratio proved in [10, 12]. The time complexity is dominated by the cost of evaluating Q over $\bar{D}$.

Input: $X = \{x_0, x_1 \ldots\}$ – a set of variables,
val a valuation for X,
$\pi : X \to [0, 1]$ the probability of each variable to be True,
$[\varphi_1, \ldots, \varphi_m]$ – array of m monotone DNF formulas over X,
$[\mathrm{CNF}(\varphi_1), \ldots, \mathrm{CNF}(\varphi_m)]$ – (monotone) CNF of $\varphi_1, \ldots, \varphi_m$ respectively.
Output: $x \in X$ – the next variable to probe
terms, clauses $\leftarrow$ arrays containing at the jth index the number of terms and clauses in φ_j and $\mathrm{CNF}(\varphi_j)$ resp.;
QVal $\leftarrow$ array of size $|X|$;
for $x_i \in X$ *s.t.* $\mathrm{val}(x) =$ *Unknown* **do**
  for $b \in \{$*False*, *True*$\}$ **do**
    $Q_b \leftarrow 0$;
    $\mathrm{val}' \leftarrow$ val setting $x_i := b$;
    for $j \in 1 \ldots m$ **do**
      $t_j \leftarrow$ # terms in φ_j evaluated to Unknown by val';
      $c_j \leftarrow$ # clauses in $\mathrm{CNF}(\varphi_j)$ evaluated to Unknown by val';
      $Q_b \leftarrow Q_b + \mathrm{terms}[j] \cdot \mathrm{clauses}[j] - t_j \cdot c_j$;
    $\mathrm{QVal}[i] \leftarrow \Pr(x_i) \cdot Q_{\mathrm{True}} + (1 - \Pr(x_i)) \cdot Q_{\mathrm{False}}$;
return $\arg\max_i \mathrm{QVal}[i]$

Algorithm 2: Algorithm Q-value-next (pick next probe).

3.4 General SPJU Queries

To conclude our analysis of the complexity of our problem, we consider the general problem setting, where the class of queries that is considered is SPJU, with no restrictions imposed. For this class, OPT-PEER-PROBE is hard.

Theorem 3.5. OPT-PEER-PROBE *is NP-hard (in data complexity) even for SPJ queries (without union), even on shared databases where all tuples have the same probability, and even for sharing a single output tuple.*

We next propose Algorithm General (outlined in Algorithm 4) as a heuristic solution for OPT-PEER-PROBE, by adapting the Interactive Boolean Evaluation algorithm of [8], which has approximation guarantees for k-DNF, but only for a single formula. The algorithm of [8] alternates between two sub-algorithms, alg0 and alg1, which respectively try to show that the formula is False and True; we halt as soon as one of them succeeds. To allow simultaneous evaluation of multiple formulas, we apply alg0 on the disjunction of tuple provenances in DNF (such that it tries to prove every DNF term in the entire provenance is False, each time discarding evaluated terms and formulas). We replace alg1 by the extension of RO to multiple formulas, as they operate on a similar principle (greedily selecting the term by which proving a formula is True is cheapest and sequentially probing its variables).

Input: $X = \{x_0, x_1 \ldots\}$ – a set of variables
$\pi : X \to [0,1]$ the probability of each variable to be True
dnfs – m monotone DNF formulas over X.
val $\leftarrow$ partial valuation setting all $x \in X$ to Unknown;
cnfs $\leftarrow$ monotone CNF for every $\varphi \in$ dnfs;
while $\exists \varphi \in$ dnfs, val$(\varphi) =$ *Unknown* **do**
  $x \leftarrow$ Q-value-next$(X, \pi,$ dnfs, cnfs$)$;
  $b \leftarrow$ probe x;
  val $\leftarrow$ val setting $x := b$;
return $[b_1, \ldots, b_m]$ *the consent values in* $\{True, False\}$ *for the formulas of* dnfs.

Algorithm 3: Algorithm Q-value for OPT-PEER-PROBE

Input: $X = \{x_0, x_1 \ldots\}$ – a set of variables,
$\pi : X \to [0,1]$ the probability of each variable to be True,
dnfs – m monotone DNF formulas over X.
val $\leftarrow$ partial valuation setting all $x \in X$ to Unknown;
cost1, cost2 $\leftarrow 0$;
while $\exists \varphi \in$ dnfs, val$(\varphi) =$ *Unknown* **do**
  if cost1 $\geq$ cost0 **then**
    $x \leftarrow$ choose probe using Alg0 from [8], Section 5.1 on $\bigvee_{\psi \in \text{dnfs} \mid \text{val}(\psi) = \text{Unknown}} \psi$;
  else
    $x \leftarrow$ choose probe using RO on dnfs, val;
  $b \leftarrow$ probe x;
  val $\leftarrow$ val setting $x := b$;
return $[b_1, \ldots, b_m]$ *the consent values in* $\{True, False\}$ *for the formulas of* dnfs.

Algorithm 4: Algorithm General for OPT-PEER-PROBE

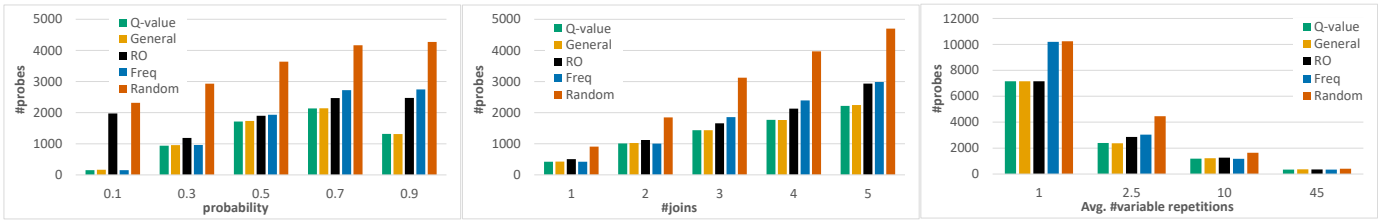
4 Experimental Study

We have implemented all algorithms described in Section 3 and examined their performance. We start by describing our experimental settings, and then present the results.

4.1 Experimental Settings

To our knowledge, the problem that we study has not been investigated before, so we are not aware of a standard benchmark or of existing implementations to use as competitors. We have thus designed a dedicated benchmark for the problem.

Dataset Our dataset, **skewed**, is a parametrized dataset that we generate randomly based on parameters such as the number of tuples, the number of joins, and the average number of repetitions per variable. The default parameters we have used are 1000 query output rows, 4 joins, 8 as projection limit, and where each variable repeats 2.6 times on average. The average total DNF/CNF



(a) # probes for varying probabilities (b) # probes for varying number of joins (c) # probes for varying # repetitions

Figure 1: Qualitative experiments for **skewed** dataset (less probes is better).

provenance sizes for the reported experiments (summed over all rows, averaged over experiment repetitions) were up to 0.4M and 2.2M, respectively.

Algorithms We have compared the following algorithms.

- Random. A baseline probing variables in random order.
- Freq. A baseline that greedily probes a variable with the maximal number of occurrences in the DNF provenance.
- RO. Algorithm 1 from Section 3.2
- Q-value. Algorithm 3 from Section 3.3.
- General. Algorithm 4 from Section 3.4.

Of course, all the algorithms that we benchmark will maximally simplify expressions after each probe answer, so that they never make useless probes. For fairness, all algorithms break ties by the same arbitrary criterion.

We have implemented all algorithms in node.js using Express, and in Java 13. Experiments were run on a Windows 10 machine using an Intel Core i7 5600U processor with 8 GB of DDR4 memory. Each experiment was executed 10 times (50 times for Random) and the reported results are the averages over these executions, each time drawing a valuation uniformly at random according for a fixed variable probability 0.7 (leading to $\sim 50\%$ True expressions), and executing all algorithms over this valuation.

4.2 Experimental Results

We next describe the results of our experiments. See further results in [13]. We evaluate our algorithms based on the *number of probes* that they issue, which is the criterion that we are trying to optimize; we discuss execution times at the end of the section. Note that all algorithms can be used even in cases where they do not have optimality guarantees – an algorithm with no optimality guarantees can still turn out to be efficient in practice.

Figure 1a shows the performance of the algorithms for varying probabilities that a probe is answered affirmatively. The advantage of our algorithms over Random is large (up to 93.5% deviation). Freq performs more poorly at proving an expression is True, intuitively since it does not account for the likelihood of terms to be True. Its comparative performance thus deteriorates when probability increases (up to a 52% deviation). RO performs poorly for both low and high probabilities, since term sizes are mostly equal and thus its choice of term

is almost arbitrary (up to 92% deviation). General and Q-value are the best for all tested probabilities, with a slight advantage to Q-value (up to 9% deviation).

Figure 1b shows the number of probes issued by each of the algorithms when varying the number of joins (corresponding to the DNF term sizes) from 1 to 5. The overall cost increases with the number of joins for all algorithms. When the provenance expressions are very simple (at most 2 variables per clause), choosing the most frequent variable performs as well as the more sophisticated solutions. As expressions become more complex, General and Q-value become significantly superior, also comparing to Random and RO. The reason is that they perform a finer analysis of the provenance structure, and in particular rely on techniques for choosing variables whose probing is effective for either proving True or False. They deviate by only up to 1.3% from the best performing algorithm for any probability.

In Figure 1c, we vary the average number of times that a variable is repeated. When this number is low (i.e., the expression is "close to" read-once), the advantage of our solutions is more significant. In particular, when there are no repetitions, provenance becomes overall read-once and RO is provably optimal; in contrast, Freq and Random perform equally badly (deviating by 42% from RO). As observed in the Figure, expressions that are close to read-once are "more difficult", since when variables are often repeated a single probe can eliminate many terms. This exacerbates the importance of our optimality results for read-once expressions.

Execution time We have thus far focused on measuring the number of probes performed by the algorithms, which is our main optimization goal. The algorithms' execution time, i.e., the time it took to choose the next probe, was typically a few milliseconds, and up to 1.3 seconds in all of our experiments – much less than the latency of obtaining probe answers in realistic scenarios, e.g., over the Web or with manual answers from peers.

5 Related Work

Provenance As described in Section 2, we use provenance to track the dependencies of derived data on the input data, and consequently decide on whose consent should be probed. Provenance has been extensively studied, with multiple models and applications, e.g., [18, 19, 7, 4, 20, 17, 21, 16]. Specifically, provenance has been used for access control, which is related to consent management [2, 22, 3, 5, 23]. In that work, the collection of atomic permissions is out of scope: they are fully given as input, either before or after the computation of provenance. Previous work on provenance considered read-once provenance (e.g., [16, 17]) – but to our knowledge, not read-once *DNF*. In contrast with read-once DNF, Interactive Boolean Evaluation for general read-once formulas is an open problem [24].

Interactive Boolean Evaluation We have adapted previous work to devise efficient algorithms for selecting probes and evaluating Boolean provenance expressions, and specifically [9, 10] and [8] for read-once DNF, DCNF and k-DNF provenance. This problem has been studied in other contexts and under other

names, including system testing, e.g., [9, 24] (where it is termed *Sequential System Testing*, active learning [12] and its connection to other problems such as Stochastic Set Cover [10, 11] (where it is termed *Stochastic Boolean Function Evaluation (SBFE)*). This line of work differs from the present work in several aspects: first, OPT-PEER-PROBE considers the simultaneous evaluation of multiple, possibly many expressions corresponding to the provenance of multiple tuples, whereas Interactive Boolean Evaluation is concerned with a single formula. The only exception to our knowledge is [10], which proposes constructions for simultaneous evaluation, but not in the context of query provenance. We have used a similar idea in Algorithm 3. Second, the works most related to ours are theoretical, and do not include an empirical study of algorithm performance.

Data sharing Studies on data sharing in social networks aim at studying how access policies [25], privacy [26], trust [27] and willingness to share data [28] can be defined over the network. Different cryptographic means and implementation designs have been proposed for this purpose [29, 30, 31, 32, 27]. While our work focuses on establishing whether consent for sharing is given or not, cryptographic techniques as in [30] may be employed to *enforce* these policies. Multiple ownership over data items has also been considered in this context [33, 31, 27], focusing on enforcing a policy that adheres to the individual policies of the involved peers. However, these studies do not consider data derivation/querying but rather the sharing of atomic items, which leads to technical problems different from ours.

6 Conclusion

We have proposed in this paper a new framework for managing consent in shared databases. Consent is managed at the tuple level, and we formalize the problem of determining consent for *query output tuples* via probing peers for their consent for relevant *input database tuples*. We have studied the complexity of the resulting optimization problem, showing intractability in general and identifying tractable sub-classes and approximate solutions. Our experimental study has validated the effectiveness of our algorithms, demonstrating their optimal or near-optimal performance in different cases and their superiority with respect to baseline alternatives.

Acknowledgements

This work was funded in part by the Israel Science Foundation (grant No. 1157/16).

References

[1] European Council and European Parliament. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). European Commission.

[2] S. Abiteboul, P. Bourhis, and V. Vianu, "A formal study of collaborative access control in distributed datalog," in *ICDT*, 2016.

[3] J. N. Foster, T. J. Green, and V. Tannen, "Annotated XML: queries and provenance," in *PODS*, 2008.

[4] G. Karvounarakis, Z. G. Ives, and V. Tannen, "Querying data provenance," in *SIGMOD*, 2010.

[5] V. Z. Moffitt, J. Stoyanovich, S. Abiteboul, and G. Miklau, "Collaborative access control in Webdamlog," in *SIGMOD*, 2015.

[6] Y. Amsterdamer and O. Drien, "PePPer: Fine-grained personal access control via peer probing," in *ICDE*, 2019.

[7] T. Imielinski and W. L. Jr., "Incomplete information in relational databases," *J. ACM*, vol. 31, no. 4, 1984.

[8] S. R. Allen, L. Hellerstein, D. Kletenik, and T. Ünlüyurt, "Evaluation of monotone DNF formulas," *Algorithmica*, vol. 77, no. 3, 2017.

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

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

[11] H. Kaplan, E. Kushilevitz, and Y. Mansour, "Learning with attribute costs," in *STOC*, 2005.

[12] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *J. Artif. Intell. Res.*, vol. 42, 2011.

[13] O. Drien, A. Amarilli, and Y. Amsterdamer, "Managing consent for data access in shared databases (full version)," https://u.cs.biu.ac.il/~amstery/files/managingconsent.pdf, 2021.

[14] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995.

[15] T. J. Green, G. Karvounarakis, and V. Tannen, "Provenance semirings," in *SIGMOD*, 2007.

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

[17] S. Roy, V. Perduca, and V. Tannen, "Faster query answering in probabilistic databases using read-once functions," in *ICDT*, 2011.

[18] Y. Amsterdamer, D. Deutch, and V. Tannen, "Provenance for aggregate queries," in *PODS*, 2011.

[19] D. Deutch, T. Milo, S. Roy, and V. Tannen, "Circuits for datalog provenance," in *ICDT*, 2014.

[20] D. Olteanu and J. Zavodny, "Factorised representations of query results: size bounds and readability," in *ICDT*, 2012.

[21] P. Senellart, L. Jachiet, S. Maniu, and Y. Ramusat, "ProvSQL: Provenance and probability management in PostgreSQL," *PVLDB*, vol. 11, no. 12, 2018.

[22] A. Bates, B. Mood, M. Valafar, and K. R. B. Butler, "Towards secure provenance-based access control in cloud environments," in *CODASPY*, 2013.

[23] J. Park, D. Nguyen, and R. S. Sandhu, "A provenance-based access control model," in *PST*, 2012.

[24] T. Ünlüyurt, "Sequential testing of complex systems: a review," *Discrete Applied Mathematics*, vol. 142, no. 1-3, 2004.

[25] Y. Cheng, J. Park, and R. S. Sandhu, "An access control model for online social networks using user-to-user relationships," *IEEE Trans. Dependable Sec. Comput.*, vol. 13, no. 4, 2016.

[26] L. Yu, S. M. Motipalli, D. Lee, P. Liu, H. Xu, Q. Liu, J. Tan, and B. Luo, "My friend leaks my privacy: Modeling and analyzing privacy in social networks," in *SACMAT*, 2018.

[27] N. C. Rathore, P. Shaw, and S. Tripathy, "Collaborative access control mechanism for online social networks," in *ICDCIT*, 2016.

[28] E. Gudes and N. Voloch, "An information-flow control model for online social networks based on user-attribute credibility and connection-strength factors," in *CSCML*, 2018.

[29] A. K. Abdulla and S. Bakiras, "HITC: data privacy in online social networks with fine-grained access control," in *SACMAT*, 2019.

[30] M. Davidson, T. Tassa, and E. Gudes, "Content sharing schemes in DRM systems with enhanced performance and privacy preservation," *Journal of Computer Security*, vol. 24, no. 6, 2016.

[31] P. Ilia, B. Carminati, E. Ferrari, P. Fragopoulou, and S. Ioannidis, "SAMPAC: socially-aware collaborative multi-party access control," in *CODASPY*, 2017.

[32] I. Kayes and A. Iamnitchi, "Privacy and security in online social networks: A survey," *Online Social Networks and Media*, vol. 3-4, 2017.

[33] H. Hu, G. Ahn, Z. Zhao, and D. Yang, "Game theoretic analysis of multiparty access control in online social networks," in *SACMAT*, 2014.