Cache Me If You Can: Accuracy-Aware Inference Engine for Differentially Private Data Exploration

Miti Mazmudar
miti.mazmudar@uwaterloo.ca
University of Waterloo

Thomas Humphries
t3humphries@uwaterloo.ca
University of Waterloo

Matthew Rafuse
matthew.rafuse@uwaterloo.ca
University of Waterloo

Xi He
xi.he@uwaterloo.ca
University of Waterloo

ABSTRACT

Differential privacy (DP) allows data analysts to query databases that contain users' sensitive information while providing a provable privacy guarantee to the users. Recent interactive DP tools such as APEx provide accuracy guarantees over the query responses, but may fail to support a large number of queries with a limited total privacy budget, as they process new incoming queries independently from the historical queries. This paper proposes a DP accuracy-aware inference engine, that maintains a cache of past responses, and utilizes these responses to save privacy budget for future queries. To make use of the cache, we design a modified matrix mechanism and show that it saves privacy budget in our preliminary experiments.

1 INTRODUCTION

Organizations often collect large datasets that contain users' sensitive data and permit data analysts to query these datasets for aggregate statistics. However, responses to these queries may be used by a curious data analyst to learn an individual's record. The Differential Privacy (DP) formulation [4, 5] allows organizations to provide a guarantee to their users that the presence or absence of their record in the dataset will only change the distribution of the query response by a small factor, measured in terms of ϵ. Multiple query responses can be released and post-processed without breaking this privacy guarantee. Statistical organizations such as the US Census Bureau [13] and companies like Google and Microsoft have started to adopt differential privacy in their applications.

Existing deployments of differential privacy [1, 2, 10, 12–14] mainly consider a non-interactive setting, where the analyst provides queries in advance. Systems [6, 9, 15, 17] that support interactive settings for differentially private data exploration have been difficult to deploy as data analysts have often been left with choosing an appropriate privacy budget ϵ per query. Secondly, data analysts desire a level of query accuracy to be maintained, a constraint that traditional differentially private systems do not provide on their outputs. Ge et al.'s APEx system [7] eliminates these two drawbacks; data analysts need only specify native queries and accuracy bounds in the form of an error rate α and a probability of failure β without specifying the privacy budget per query.

However, most database systems that support interactive DP [7, 9, 15], may fail to support a large number of queries under a limited total privacy budget. As each query consumes some privacy budget, without proper planning of these queries, the privacy budget will be depleted quickly. Interactive systems such as APEx currently process each set of queries that have the same accuracy requirements, or a *workload*, independently, without taking into account the differentially private responses to previous workloads. We observe that in data exploration, workloads that arrive at different times can be highly correlated. For example, an analyst may ask a query that spans the entire domain of a given attribute and then ask queries that focus on successively smaller parts of the domain (see Example 1 in Figure 1). Alternatively, their queries may explore the domain of an attribute sequentially (Example 2). Moreover, after an analyst sees a noisy response for a given query, they may request a more accurate response for the *same* query (Example 3). Our key insight lies in the observation that processing new workloads while using previously cached responses saves privacy budget and thus we can answer more queries, under a given total privacy budget.

Example 1: Related queries: ($\alpha = 0.1|D|$, $\beta = 0.0005$).
Histogram: $Q_1 = [0, 5000)$, $Q_2 = \{[0, 2500), [2500, 5000)\}$,
$Q_3 = \{[0, 1250), [1250, 2500), [2500, 3750), [3750, 5000)\} \ldots$
Non-histogram: $Q_1 = [0, 5000)$, $Q_2 = \{[0, 2500), [0, 5000)\}$,
$Q_3 = \{[0, 1250), [0, 2500), [0, 3750), [0, 5000)\} \ldots$
Example 2: Disjoint queries: ($\alpha = 0.1|D|$, $\beta = 0.0005$)
$Q_1 = [0, 5000)$, $Q_2 = [0, 500)$, $Q_3 = [500, 1000)$,
$Q_4 = [500, 1000) \ldots Q_{11} = [4500, 5000)$
Example 3: Same query with higher accuracy requirements.
$Q_1 = [0, 5000)$, $\alpha = 0.25|D|$, $\beta = 0.0005$.
$Q_2 = [0, 5000)$, $\alpha = 0.125|D|$, $\beta = 0.0005$.

Figure 1: Examples of query workloads over a capital gains attribute that would benefit from our cached approach.

In this work, we design *CacheDP*, an accuracy aware inference engine for differentially private data exploration. This engine consists of a cache algorithm that interacts with the private data and data analysts. We integrate this algorithm to an existing accuracy-aware DP tool, namely APEx, such that the accuracy requirements of the analyst are met and simultaneously, prior responses are used to save the privacy budget on a given workload.

Peng et al.'s Pioneer [16] uses historical query answers in order to obtain accurate responses to upcoming queries over the same or related predicates. However, our approach has three distinguishing features. First, we can optimize the privacy budget over multiple predicates, unlike Pioneer which handles a scalar response one by one. To achieve this, we adapt the matrix mechanism [12] from a single workload setting to capture both queries in the cache and the unanswered queries. In this way, we can spend a much smaller privacy budget than APEx to achieve the accuracy requirement. Second, if the same query is asked under a higher accuracy requirement, we add correlated noise to improve the accuracy [11]. Third, we proactively fetch certain query responses at a marginal cost so that if a user asks for responses to a disjoint query with similar accuracy requirements later, we do not need to spend any additional privacy budget.

Our main contributions are:

- We formulate the problem of interactively using historical query answers to minimize the privacy budget spent.
- We propose a cache-based inference engine and algorithm that satisfies accuracy requirements interactively.
- We integrate our engine into an existing accuracy-aware differential privacy tool, namely APEx.
- We provide a preliminary evaluation of our algorithm across various use-cases to illustrate the savings in the privacy budget compared to APEx.

2 BACKGROUND

We assume a single-table relational schema $\mathcal{R}(\mathcal{A}_1, \ldots \mathcal{A}_d)$. The domain of an attribute $\mathcal{A}_i$ is represented as $dom(\mathcal{A}_i)$. The full domain of $\mathcal{R}$ is $dom(\mathcal{R}) = dom(\mathcal{A}_1) \times \cdots \times dom(\mathcal{A}_d)$. A database instance D of relation $\mathcal{R}$ is a multiset whose elements are tuples in $dom(\mathcal{R})$. A predicate counting query takes a predicate $\phi : dom(\mathcal{R}) \to \{0, 1\}$ and returns the number of tuples in D that satisfy ϕ, i.e., $\phi(D) = \sum_{t \in D} \phi(t)$. Given a database instance D and $\Phi = \{\phi_1, \ldots, \phi_n\}$, the set of unit length predicates, we can represent D with a data vector x, where $x[i] = \phi_i(D)$. Given x, we can represent all linear counting queries as a length-n vector $w = [w_1, \ldots, w_n]$ with $w_i \in \{0, 1\}$. We say that two queries w and v are disjoint, if $w \cdot v = 0$. The answer to a linear counting query is thus $w \cdot x$. Hence, we can represent a workload of ℓ linear counting queries as a $\ell \times n$ matrix W over the domain of x and the response to this workload is Wx. We follow the standard definition of differential privacy (DP).

Definition 2.1 (ϵ-Differential Privacy [4])**.** A randomized mechanism $M : \mathcal{D} \to \mathcal{O}$ satisfies ϵ-differential privacy if

$$Pr[M(D) \in O] \leq e^{\epsilon} Pr[M(D') \in O] \tag{1}$$

for any set of outputs $O \subseteq \mathcal{O}$, and any pair of *neighboring* databases D, D' such that $|D \backslash D' \cup D' \backslash D| = 1$.

Definition 2.2. (Accuracy Bound [7]). Given a DP mechanism M that answers a workload counting query W over x, we say that M is (α, β)-accurate if for any output of M, $\tilde{y}$, we have

$$Pr[\|Wx - \tilde{y}\|_\infty \geq \alpha] \leq \beta$$

One way to obtain noisy workload response with (α, β)-accuracy is to use the Laplace mechanism with a specific choice of epsilon.

Definition 2.3 (Laplace Mechanism)**.** For a given $\ell \times n$ workload matrix W and accuracy requirement (α, β) the Laplace Mechanism $\mathcal{L}$ is defined as

$$\mathcal{L}(W, x, \alpha, \beta) = Wx + Lap(\|W\|_1/\epsilon)^{\ell}$$

where $\epsilon = \frac{\|W\|_1 \ln(1/(1-(1-\beta)^{1/\ell}))}{\alpha}$.

Algorithm 1 System Overview

Require: Dataset D, Total privacy budget $\mathcal{B}$, Proactive Threshold $\mathcal{T}$
Initialize privacy loss $B \leftarrow 0$, Cache $C \leftarrow \emptyset$
repeat
  Receive (Q, α, β) from analyst
  $W \leftarrow$ getWorkloadMatrix(Q, x)
  $A \leftarrow$ ChooseStrategyMatrix(W)
  $\epsilon \leftarrow$ EstMatrixCost(W, A, α, β, C)
  ▼ Get responses for disjoint predicates proactively.
  $W' \leftarrow$ GetProactive(W)
  $A' \leftarrow$ ChooseStrategyMatrix(W')
  $\epsilon' \leftarrow$ EstMatrixCost$(W', A', \alpha, \beta, C)$
  if $\epsilon' \leq \epsilon\mathcal{T}$ and $\epsilon + B < \mathcal{B}$ **then**
    $(W_i, A, \epsilon) \leftarrow (W', A', \epsilon')$,
  if $\epsilon + B > \mathcal{B}$ **then** break
  $\tilde{y}, \epsilon_Q \leftarrow$ ModifiedMatrixMechanism(W, A, ϵ, C, x)
  $B \leftarrow B + \epsilon_Q$
  return $\tilde{y}$
until No more queries sent by analyst

The Laplace Mechanism as defined above satisfies ϵ-DP whilst achieving (α, β)-accuracy as shown in A.1 of the APEx Paper [7]. Another way to obtain (α, β)-accurate responses is to use the Matrix Mechanism [12] using a Monte Carlo (MC) simulation to empirically bound the accuracy, as in APEx [7].

Definition 2.4 (Matrix Mechanism [12])**.** Given an $\ell \times n$ workload matrix W and a strategy matrix A, such that W is some linear combination of A, the matrix mechanism is defined as

$$M_{K,A}(W, x) = WA^{+}K(A, x)$$

where $K(A, x) = Ax + Lap(\|A\|_1/\epsilon)^{|A|}$, and $|A|$ denotes the number of rows in matrix A.

The flexibility of the matrix mechanism comes from different choices of the strategy matrix A. The Hierarchical Tree inferencer introduced by Hay et al. [8] can be represented as a strategy matrix H, as Li et al. point out. Mckenna et al. [14] provide a technique to optimize the choice of strategy matrix.

3 APPROACH

Given a database instance D and a total privacy budget set by data owner $\mathcal{B}$, our system receives a sequence of workloads with their accuracy requirements $[\ldots, (Q, \alpha, \beta), \ldots]$. The workload and its accuracy requirement will be translated to a differentially private algorithm that meets the accuracy requirement while minimizing the privacy budget spent. Our system, *CacheDP*, is different from prior work APEx [7] in that we interface *CacheDP* with a cache in order to exploit past responses. In this section, we describe how our system caches responses and uses them to answer each workload accurately, while saving the privacy budget for future workloads.

3.1 Overview

Our cache C is indexed by the query predicate in the *strategy* matrix. For each query predicate, we store a noisy response $\tilde{y}$ that was drawn from the Laplace distribution, as an intermediate step in the matrix mechanism ($K(A, x)$). We also store the noise parameter b, that was used to obtain $\tilde{y}$. We present our main algorithm in Algorithm 1.

Algorithm 2 Cached Matrix Mechanism

1: **function** ModifiedMatrixMechanism(W, A, ϵ_{est}, C, x)
2: F, $P \leftarrow$ SplitStrategy(A, C)
3: **if** $F = A$ **then** ▷ All predicates are in the cache.
4: ▼ Improve accuracy of cached responses.
5: $\tilde{y}_a, \epsilon \leftarrow$ RelaxPrivacy(C, Ax, ϵ_{est}) ▷ $\epsilon \leq \epsilon_{est}$
6: updateCache(A, $\|A\|_1/\epsilon$, $\tilde{y}_a$)
7: **else** ▷ At least one predicate is not in the cache.
8: ▼ Use as many cached predicate responses as possible.
9: $\tilde{y}_f \leftarrow$ getFromCache(F)
10: $\tilde{y}_p \leftarrow Px + Lap(\|P\|_1/\epsilon_{est})$
11: insertToCache(P, $\|P\|_1/\epsilon_{est}$, $\tilde{y}_p$)
12: $\tilde{y}_a \leftarrow \begin{bmatrix} \tilde{y}_f \\ \tilde{y}_p \end{bmatrix}, \epsilon \leftarrow \epsilon_{est}$
13: **return** $(WA^{+}\tilde{y}_a, \epsilon)$

14: **function** EstMatrixCost(W, A, α, β, C)
15: Set $u = \frac{\|A\|_1 \|WA^{+}\|_F}{\alpha\sqrt{\beta/2}}$ and $l = 0$
16: $\epsilon \leftarrow$ binarySearch(l,u,estimateBeta($\cdot$, A, WA^{+}, α, β, C))
17: **return** ϵ

18: **function** estimateBeta(ϵ, A, WA^{+}, α, β, C)
19: Sample size $N \leftarrow 10000$ and failure counter $n_f \leftarrow 0$
20: F, $P \leftarrow$ SplitStrategy(A, C)
21: **if** $F = A$ **then** ▷ All predicates are in the cache.
22: ▼ Estimate minimum target ϵ by treating F as if it was P.
23: $P \leftarrow F, F \leftarrow \emptyset$
24: **for** $i \in [1, \ldots, N]$ **do**
25: $\eta_{i,j} \sim \begin{cases} Lap(\|P\|_1/\epsilon) & \text{if } A[j] \in P \\ Lap(\text{getFromCache}(A[j])) & \text{otherwise} \end{cases}$
26: **if** $\|(WA^{+})\boldsymbol{\eta}_i\|_\infty > \alpha$ **then**
27: $n_f \leftarrow n_f + 1$
28: $\beta_e = n_f/N, p = \beta/100$
29: $\delta\beta = z_{1-p/2}\sqrt{\beta_e(1-\beta_e)/N}$
30: **return** $(\beta_e + \delta\beta + p/2) < \beta$

After receiving a workload Q with accuracy requirement (α, β), our system generates a corresponding workload matrix representation W (line 4), such that $W \cdot x$ computes the query response. We then obtain a strategy matrix A that can answer all queries in W, while using as many cached responses as possible (line 5). Developing such an optimally cache-aware strategy matrix is left to future work. We treat this function as a blackbox for our initial analysis and illustrate examples of such strategy matrices for our experiments in the evaluation. We then estimate the minimum privacy budget required to answer queries using this strategy matrix, along with the cache C (line 6).

Our system then attempts to proactively fetch responses for queries whose predicates are disjoint with the new workload (lines 8–12), using the minimal privacy budget that is needed to answer them accurately. For example, if the data analyst queries the number of people with capital gain from 0 to 2500, we may expect that counts over other disjoint ranges at the same granularity, such as [2500,5000), [5000,7500) etc. would be relevant for the analyst in future queries. Hence, we modify the workload matrix W to W' as if the analyst also requested these other disjoint predicates, through the GetProactive function (line 8). We then again choose a strategy matrix that optimally uses the cache to answer the predicates in W' (line 9). Under our accuracy definition 2.2, when the number of predicates increases, the cost of obtaining noisy responses to these predicates increases slightly. Thus, we ensure that the privacy budget for the proactive W' is within a multiplicative factor $\mathcal{T}$ of the original budget estimate ϵ (lines 10–12).

We check that the privacy budget spent is less than the remaining privacy budget irrespective of whether we proactively cache query responses or not (line 13). We thus provide $\mathcal{B}$-differential privacy, just as APEx does. We finally execute our *modified matrix mechanism*, with the chosen strategy matrix A under the privacy budget ϵ, which was estimated to provide the (α, β) guarantee. We obtain the response $\tilde{y}$, which answers W, and the actual privacy budget spent ϵ_Q (line 14) and we update the cumulative privacy budget (line 15). We now proceed to discuss the modified matrix mechanism function, which is presented in Algorithm 2. We remark that the cost estimation function for the matrix mechanism is similar to the original function in APEx. Only the implementation of the Monte Carlo simulation, encapsulated in the call to the estimateBeta function, is modified to be cache-aware and is also discussed below.

3.2 Cached Matrix Mechanism

Unlike the matrix mechanism in prior work [7, 12], the new mechanism and its cost estimation function, take in the cache C as input. We partition the strategy matrix A into two parts based on whether the predicate exists in the cache. That is, if the predicate belongs to the cache, it is in the free matrix F, otherwise it is in the matrix to be fetched from the database P (line 2). If noisy responses for *all* of the predicates in the strategy matrix are present in the cache (line 3), then our modified mechanism improves the accuracy of these noisy responses to satisfy the analyst's accuracy bounds. Otherwise, we use as many cached responses as possible to answer the query and use a lower privacy budget to noise all non-cached predicates, P. For each case, we describe the relevant parts of the modifiedMatrixMechanism and estimateBeta functions below.

In the first case, we estimate the least privacy cost ϵ_{est} required to answer these queries without the cache. This is achieved by treating all the queries found in the cache F as new queries P (lines 21–23 and second case of line 25) in the function estimateBeta. That is, all cached responses should meet the target epsilon ϵ_{est} in order to be sufficiently accurate. To improve the accuracy of cached responses while minimizing the privacy budget spent, we apply a similar technique as the relaxPrivacy function in APEx, which was based on Koufogiannis et al.'s work [11]. The new noisy response $\tilde{y}_p$ is correlated with the old response in the cache and the new privacy budget spent here, ϵ, is less than ϵ_{est}. However, in contrast with APEx, each of these cached responses may be obtained across different workloads, and hence the privacy budget for each response may be different. For such cases, we leave a careful analysis of the new privacy loss to the full paper. We also update the cache with these more accurate responses (lines 5–6).

On the other hand, if the noisy response for at least *one* of the predicates in the strategy matrix is absent from the cache ($F \subset A$), then our mechanism fetches the noisy responses $\tilde{y}_f$ for the cached

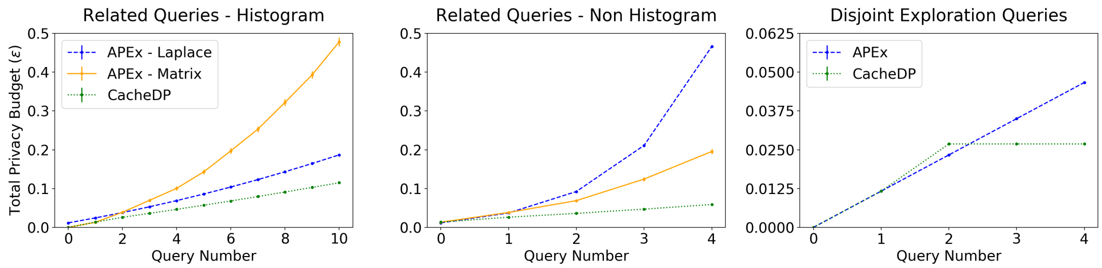

Figure 2: We consider the histogram workload sequence shown in Example 1 of Figure 1, the non-histogram sequence in that example and the sequence with disjoint predicates in Example 2, in the first, second and third plots respectively. We plot the average cumulative privacy budget for CacheDP and for the best mechanism in APEx, over 50 runs, versus the query number in the workload sequence.

predicates (line 9) and obtains new noisy responses $\tilde{y}_p$ for the non-cached predicates (line 10). We also insert these new responses into our cache (line 11). Within the ESTIMATEBETA function, we estimate the privacy budget required for the new responses using MC as follows. For each run of MC, for predicates in the non-cached strategy matrix P, we set the noise parameter for the Laplace noise as the sensitivity of this matrix divided by the ϵ from the binary search (first case in line 25). However, for predicates that are in the cached strategy matrix F, the noise parameter is simply the one that was used to generate the cached response (second case in line 25). We then compute the L_∞ norm of the noise vector formed with these different noise parameters, and test the error bound (line 26).

4 EVALUATION

We integrated our cached-based approach *CacheDP* into APEx and compared its privacy cost with that of APEx over the first three sequences of workloads that are described in Figure 1. We simulate 50 runs of the ESTMATRIXCOST function in Algorithm 2. We plot the average cumulative privacy budget versus the sequence number of each workload query, over 50 runs, in Figure 3.2. We use the Adult dataset of the 1994 US census data [3]. We describe each workload sequence below, and then proceed to discuss our results.

In our first experiment, we consider the case when a data analyst asks increasingly fine-grained range counting queries for an attribute; that is, the histogram and non-histogram query workloads presented in Example 1 of Figure 1. In both workloads, although the cache does not contain the responses for any predicates of later workloads, it contains responses that are *related* to them. In our CHOOSESTRATEGYMATRIX function, we aim to automatically create a strategy matrix that exploits as many past related responses that are currently in the cache as possible. Thus, for this example, we manually chose the binary, hierarchical H_2 strategy matrix [12]. The predicates in the strategy matrix for both workloads can be arranged into a binary tree, under an inclusion relationship. Each workload in the workload sequence consists of all nodes in the tree at that level. With APEx as our baseline, we observe that its Laplace mechanism performs better for the histogram workload sequence and the matrix mechanism performs better for the non-histogram sequence. In the plots for both sequences in Figure 3.2, we can see that the cumulative privacy budget spent for our cached approach grows linearly with the query sequence number, and with a significantly reduced slope than the best mechanism offered by APEx. If the analyst continues to ask more fine-grained workload queries, then the cumulative privacy budget savings only increase.

In our second experiment, we focus on our proactive approach, which would benefit the workload sequences over disjoint predicates shown in Example 2 of Figure 1. That is, we consider the case where an analyst first asks a single large range counting query workload, and follows it up with a sequence of smaller workloads, all of which are at the same granularity and same accuracy requirements. Our baseline, APEx, uses the Laplace Mechanism to answer each workload query, as this mechanism is the optimal choice for a single query predicate. Whereas, when we receive Q_2, our GETPROACTIVE function modifies the workload matrix to fetch responses for all disjoint predicates that are at the same granularity as this query. We use the matrix mechanism with the hierarchical strategy matrix H_k, and thereby exploit the cache for the response of Q_1, by building a k-ary tree of height 1.

Within our approach, only a very small privacy budget needs to be spent in proactively fetching disjoint predicate responses for the second workload. In return, we get significant savings in the future, which will occur for even a single query over one of the disjoint predicates. These observations favor incorporating a proactive caching method. We note that benefits of the proactive approach are unique to the interactive setting as in the non-interactive one a more optimal workload could have been derived.

5 FUTURE WORK

Our prototype system can be extended and developed in several ways. First, we will integrate a cache within McKenna et al.'s [14] technique of identifying an optimal strategy matrix for a given workload query. Second, we will develop a detailed proof that *CacheDP* preserves $\mathcal{B}$-differential privacy, with particular attention to the RELAXPRIVACY algorithm. Third, we will also test our approach for queries over multiple dimensions, and optimize it for such queries. We plan to implement an efficient cache structure and test our algorithm under multiple different query workload sequences with different datasets. Although we simply use constraints over the noisy responses to improve the accuracy of our output, our work opens up the question of consistency constraints for the interactive setting. For instance, an analyst may prefer knowing more accurate versions of noisy answers to historical queries, as new and related queries are answered. Our algorithm could potentially be modified to support this setting.

REFERENCES

[1] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. 2017. Prochlo: Strong Privacy for Analytics in the Crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*. Association for Computing Machinery, New York, NY, USA, 441–459. https://doi.org/10.1145/3132747.3132769

[2] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 3574–3583.

[3] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[4] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, Shai Halevi and Tal Rabin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 265–284.

[5] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (Aug. 2014), 211–407. https://doi.org/10.1561/0400000042

[6] Marco Gaboardi, Michael Hay, and Salil Vadhan. 2019. A Programming Framework for OpenDP. https://projects.iq.harvard.edu/opendp

[7] Chang Ge, Xi He, Ihab F. Ilyas, and Ashwin Machanavajjhala. 2019. APEx: Accuracy-Aware Differentially Private Data Exploration. In *Proceedings of the 2019 International Conference on Management of Data (SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 177–194. https://doi.org/10.1145/3299869.3300092

[8] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the Accuracy of Differentially Private Histograms through Consistency. *Proc. VLDB Endow.* 3, 1–2 (Sept. 2010), 1021–1032. https://doi.org/10.14778/1920841.1920970

[9] Noah Johnson, Joseph P. Near, and Dawn Song. 2018. Towards Practical Differential Privacy for SQL Queries. *Proc. VLDB Endow.* 11, 5 (Jan. 2018), 526–539. https://doi.org/10.1145/3177732.3177733

[10] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2019. PrivateSQL: A Differentially Private SQL Query Engine. *Proc. VLDB Endow.* 12, 11 (July 2019), 1371–1384. https://doi.org/10.14778/3342263.3342274

[11] Fragkiskos Koufogiannis, Shuo Han, and George J. Pappas. 2016. Gradual Release of Sensitive Data under Differential Privacy. *J. Priv. Confidentiality* 7, 2 (2016). https://doi.org/10.29012/jpc.v7i2.649

[12] Chao Li, Gerome Miklau, Michael Hay, Andrew Mcgregor, and Vibhor Rastogi. 2015. The Matrix Mechanism: Optimizing Linear Counting Queries under Differential Privacy. *The VLDB Journal* 24, 6 (Dec. 2015), 757–781. https://doi.org/10.1007/s00778-015-0398-x

[13] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. 2008. Privacy: Theory meets practice on the map. In *2008 IEEE 24th international conference on data engineering*. IEEE, IEEE, Cancun, Mexico, 277–286. https://doi.org/10.1109/ICDE.2008.4497436

[14] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2018. Optimizing Error of High-Dimensional Statistical Queries under Differential Privacy. *Proc. VLDB Endow.* 11, 10 (June 2018), 1206–1219. https://doi.org/10.14778/3231751.3231769

[15] Frank McSherry. 2010. Privacy Integrated Queries: An Extensible Platform for Privacy-Preserving Data Analysis. *Commun. ACM* 53, 9 (sep 2010), 89–97. https://doi.org/10.1145/1810891.1810916

[16] S. Peng, Y. Yang, Z. Zhang, M. Winslett, and Y. Yu. 2013. Query optimization for differentially private data management systems. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, Brisbane, QLD, Australia, 1093–1104.

[17] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. 2020. Differentially private sql with bounded user contribution. *Proceedings on Privacy Enhancing Technologies* 2020, 2 (2020), 230–250.