Answering SQL Queries Under Differential Privacy

Abstract

Answering SPJA queries under differential privacy (DP), including graph pattern counting under node-DP as an important special case, has received considerable attention in recent years. The dual challenge of foreign-key constraints and self-joins, as well as traditional models fail to capture the complex user-tuple relationships in many real-world applications, presents a significant hurdle. The project implements two novel DP mechanisms to overcome these challenges. The first part of the project introduces Race-to-the-Top (R2T), a mechanism designed to answer arbitrary SPJA queries in databases with foreign-key constraints. This mechanism, which can be easily implemented on any Relational Database Management System (RDBMS) and Linear Programming (LP) solver, offers substantial improvements in utility over existing techniques, especially for graph pattern counting under Node-DP. The second part of the project presents Shifted Inverse, a general DP mechanism for any monotonic function under User-DP. This mechanism broadens the scope of User-DP beyond the sum estimation problem. Despite the potential for super-polynomial time complexity, the project demonstrates how to instantiate an approximate version of this mechanism in polynomial time for k-selection. This project is an implementation of the Race-to-the-Top Mechanism and the Shifted Inverse Mechanism mainly on TPC-H benchmarks.[1]

Index Terms

Differential privacy; Monotonic function; SPJA query; Foreign-key constraint

I. Introduction

Differential Privacy (DP) has emerged as a leading privacy standard, offering robust protection for individual users' data. It ensures that the outputs of a DP mechanism on neighboring datasets, which are two datasets differing only in one user's data, remain statistically indistinguishable. The standard Laplace mechanism operates by first determining the global sensitivity GS_Q of the query, which represents the maximum change in the query result if an individual's data is added or removed from the database. Subsequently, it introduces Laplace noise, calibrated based on the query result, to effectively mask this difference.

However, a notable challenge arises in the context of relational databases, where the GS_Q may potentially be infinite. This issue underscores the need for advanced mechanisms and strategies to effectively implement DP in relational databases, ensuring the privacy of user data while maintaining the utility of the database.

A. The Truncation Mechanism

The challenge above was first identified by Kotsogiannis et al. [3]. They formalized a DP policy specifically for relational databases with Foreign Key (FK) constraints. In their model, individuals and their private data are stored in separate relations, linked by FKs. This feature, while integral to the relational model, introduces significant complexity in the design of DP mechanisms.

To address this, they proposed the truncation mechanism. This approach involves deleting all customers with more than a certain threshold τ of orders before applying the Laplace mechanism. Post truncation, the query has a sensitivity of τ, and adding noise of scale τ effectively ensures DP. This solution, while simple, provides a practical approach to implementing DP in relational databases with FK constraints.

B. The Issue with Self-joins

While queries without self-joins are equivalent to mean (sum) estimation, the introduction of self-joins presents a unique challenge for relational queries. Specifically, all techniques from the statistics and machine learning literature [4]–[8] for selecting a threshold τ fundamentally depend on the independence of individuals. This means that adding or removing one individual does not impact the data associated with another individual.

However, this assumption of independence does not hold when the query involves self-joins. In such cases, even the truncation mechanism, which is designed to handle the sensitivity of queries, fails because the sensitivity of the query may not remain as τ after truncation. This highlights the complexity of implementing Differential Privacy in relational databases, particularly when dealing with self-joins.

[1]My implementation is publicly available at https://github.com/kentxusy/Differential-Privacy

C. The Race-to-the-Top Mechanism

The challenge posed by self-joins in the context of Differential Privacy (DP) was first acknowledged by Wei Dong et al. [1]. The essense of their model revolves around the selection of a near-optimal threshold τ in a DP-compliant manner, even in the presence of self-joins.

To address this, they proposed an instance-optimal truncation mechanism, termed the Race-to-the-Top(R2T) mechanism. This mechanism is designed to work in conjunction with any truncation method that satisfies specific properties. The introduction of R2T represents a significant stride in overcoming the complexities introduced by self-joins in the implementation of DP in relational databases.

D. The Shifted Inverse Mechanism

The recognition that a simplistic model cannot adequately represent the complex user-tuple relationships in numerous real-world applications has led to an increased focus on User Differential Privacy (User-DP).

While most existing research on User-DP has been confined to the sum estimation problem, KONG H et al. [2] has made significant strides by designing a general DP mechanism for any monotonic function under User-DP. This mechanism comes with robust optimality guarantees.

This mechanism, termed the Shifted Inverse mechanism, is similar to the exponential mechanism but with a deliberate downward shift in the target. This strategic shift ensures a high probability of hitting a target that is close to optimal, though the likelihood of hitting the optimal target itself might be low. This innovative approach enhances the applicability and effectiveness of User-DP in complex real-world scenarios.

II. Related Work

Addressing arbitrary SQL queries under the framework of Differential Privacy (DP) poses a substantial challenge in the field of private query processing. Initial research efforts were primarily directed towards resolving a collection of counting queries over a single relation with diverse predicates, referred to as SA queries with count aggregation [9]–[18]. The mechanisms proposed by [10], [13], [14] are optimal for a specific set of queries, but this optimality is confined to the worst-case database scenario. If the set comprises a single query, the optimality of these mechanisms is reduced to worst-case optimality. The existing literature on join queries generally supports a limited range of joins, such as PK-PK joins [19]–[23] and joins with a predetermined join attribute [24]. Although recent studies have sought to expand support for joins, accurately handling features like self-joins remains a complex undertaking. For example, PrivateSQL [3] utilizes naive truncation to truncate tuples with a high degree, which does not entirely satisfy the DP requirement when self-joins are present. In their subsequent work, Tao et al. [25] employ naive truncation to truncate tuples with high sensitivity for queries free of self-joins and suggest a mechanism for selecting τ. The error associated with their mechanism is $\Omega(GS_Q/log(GS_Q))$ with constant probability, indicating that it is at most a logarithmic-factor superior to the naive Laplace mechanism that introduces noise of scale GS_Q. However, the Race-to-the-Top (R2T) method significantly improves this area by reducing the dependency on GS_Q from near-linear to logarithmic.

The User Differential Privacy (User-DP) model has been utilized by various researchers in the past [6], [10], [11], [13]. Their methodology, however, situates the model within the context of Structured Query Language with Join and Aggregation (SPJA) queries in a relational database that incorporates foreign-key constraints.

The Shifted Inverse formulation [2] aligns with these earlier models but provides a more mathematically succinct representation. Importantly, it accomplishes this without necessitating any specific background knowledge of relational databases.

The majority of these studies have concentrated on the sum/count function, which is a very particular instance of monotonic functions. While a few other specific monotonic functions, such as distinct count, have been investigated in [7], [10], [11], these studies do not encompass arbitrary monotonic functions. This underscores a potential area for further exploration and advancement in the User-DP field.

Transitioning from a scenario where $l = 1$ to one where $l \geq 2$ introduces additional complexity due to the interdependence of users. Specifically, the naive truncation mechanism, which was effective in the $l = 1$ scenario, is no longer viable as the data of users cannot be truncated independently.

III. Implementation

A. The Race-to-the-Top Mechanism

The instance-optimal truncation mechanism, Race-to-the-Top (R2T), can be used in combination with any truncation method $Q(I, \tau)$, which is a function $Q : \mathbb{I} \times \mathbb{N} \leftarrow \mathbb{N}$ with the following properties:

- For any τ, the global sensitivity of $Q(]\cdot, \tau)$ is at most τ.
- For any τ, $Q(I, \tau) \leq Q(I)$.
- For any I, there exists a non-negative integer $\tau^*(I) \leq GS_Q$ such that for any $\tau \geq \tau^*(I)$, $Q(I, \tau) = Q(I)$.

For self-join-free SJA queries, each join result q in join results $J(I)$ references only one tuple in primary relation R_p. Thus, the tuples in R_p are independent, i.e., removing one does not affect the sensitivities of others. This means that naive truncation is a valid $Q(I,\tau)$ that satisfies the 3 properties required by R2T with $\tau^*(I)$ equal to downward local sensitivity.

When there are self-joins, naive truncation does not satisfy property (1), where all tuple sensitivities in two neighboring instances may differ. A LP-based mechanism for graph pattern counting [26] can be generalized to deal with arbitrary SJA queries, and show that it satisfies the 3 properties with $\tau^*(I)$ equal to downward local sensitivity.

Given a SJA query Q and instance I, $Q(I) = \sum_{q\in J(I)} \psi(q)$, where $J(I)$ is the join results. For $k \in [|J(I)|]$, let $q_k(I)$ be the kth join result. For each $j \in [|I(R_p)|]$, let $t_j(I)$ be the jth tuple in $I(R_p)$. Wei Dong et al. [1] use $C_j(I)$ to denote (the indices of) the set of join results that reference $t_j(I)$. More precisely,

$$C_j(I) := k : q_k(I)\ references\ t_j(I)$$

For each $k \in [|J(I)|]$, introduce a variable u_k , which represents the weight assigned to the join result $q_k(I)$. Wei Dong et al. [1] return the optimal solution of the following LP as $Q(I,\tau)$:

$$\begin{aligned} maximize \quad & Q(I,\tau) = \sum_{k\in[|J(I)|]} u_k \\ subject\ to \quad & \sum_{k\in C_j(I)} u_k \le \tau, j \in [|I(R_p)|], \\ & 0 \le u_k \le \psi(q_k(i)), k \in [|J(I)|] \end{aligned}$$

The correctness of the LP-based truncation method relies on a key property of SJA queries, that removing t_p will always reduce sensitivity $Q(I)$ by $S_Q(I,t_p)$, which is the contribution of t_p to $Q(I)$. Unfortunately, the projection operator violates this property. For SJPA, Wei Dong et al. [1] define a new LP. For each $l \in [|\pi_y J(I)|]$, a new variable $v_l \in [0, \psi(p_l(I))]$ is introduced, which represents the weight assigned to the projected result $p_l(I)$. For each $k \in [|J(I)|]$, they still use a variable $u_k(I) \in [0, \psi(q_k(I))]$ to represent the weight assigned to $q_k(I)$. Keeping the same truncation constraints on the u_k's, while adding the constraint that the weight of a projected result should not exceed the total weights of all its corresponding join results. Then they try to maximize the projected results. More precisely, the new LP is

$$\begin{aligned} maximize \quad & Q(I,\tau) = \sum_{l\in[|\pi y J(I)|]} v_l \\ subject\ to \quad & v_l \le \sum_{k\in D_l(I)} u_k \\ & \sum_{k\in C_j(I)} u_k \le \tau, j \in [|I(RP)|] \\ & 0 \le u_k \le \psi(q_k(I)), k \in [|J(I)|] \\ & 0 \le v_l \le \psi(p_l(I)), l \in [|\pi y J(I)|] \end{aligned}$$

To implement in python with CPLEX, the LP can be rewritten as

$$\begin{aligned} maximize \quad & Q(I,\tau) = \sum_{l\in[|\pi y J(I)|]} 1\cdot v_l + \sum_{k\in[|J(I)|]} 0\cdot u_k \\ subject\ to \quad & 1\cdot v_l + \sum_{k\in D_l(I)} (-1)\cdot u_k \le 0 \\ & \sum_{k\in C_j(I)} u_k \le \tau, j \in [|I(RP)|] \\ & 0 \le u_k \le \psi(q_k(I)), k \in [|J(I)|] \\ & 0 \le v_l \le \psi(p_l(I)), l \in [|\pi y J(I)|] \end{aligned}$$

B. The Shifted Inverse Mechanism

The Exponential Mechanism Given an instance V and an output range R, the exponential mechanism assigns a utility score $s(V,r)$ for each possible output $r \in R$, and samples the output with probability proportional to $\exp\left(\frac{\varepsilon s(V,r)}{2\Delta s}\right)$, where Δs is the sensitivity of the utility score, i.e.,

$$\Delta s = \max_{r\in R}\max_{V\sim V'} |s(V,r) - s(V',r)|.$$

The Shifted Inverse Mechanism In the exponential mechanism, one usually assigns the highest score, e.g., 0, to the best answer $r = f(V)$. In the Shifted Inverse Mechanism, KONG H et al. [2] intentionally shift our target down, assigning the

highest score to an $r = f(\bar{V})$ such that $\bar{V} \preceq V$ and $d(V, \bar{V}) = \tau$, for an appropriately chosen τ. Then with $r = f(\bar{V})$ at the center, they reduce the scores gradually in both directions. While this makes sense for $r < f(\bar{V})$, for $r \in (f(\bar{V}), f(V)]$, it actually assigns smaller scores to better answers, and the best answer $r = f(V)$ has (almost) the lowest score. This counter-intuitive scoring mechanism is needed to ensure $\Delta\text{len} \leq 1$ and all $r \in (f(V), D]$ have low scores so that the exponential mechanism has good utility. Indeed, their analysis [2] shows that $\tau = O\left(\frac{1}{\varepsilon} \log \frac{D}{\beta}\right)$ is sufficient. This ensures that they hit a close-to-optimal target with high probability, although the probability of hitting the optimal target is low.

Algorithm 1: Shifted Inverse

Input : The instance V, a monotonic function $f : V \to [D]$, the privacy budget ϵ, and failure probability β
Output: A privatized $f(V)$
$\tau \leftarrow \lceil 2\epsilon \ln\left(\frac{D+1}{\beta}\right) \rceil$
for $j \in [2\tau]$ **do**
  Compute $\hat{f}(V, j) = \min_{\bar{V}, \bar{V} \preceq V, d(V,\bar{V}) \leq j} f(\bar{V})$
end
For $r \in [D]$ set $s(V, r) =$

$$\begin{cases} \tau - j, & \text{if } r \in [\hat{f}(V,j), \hat{f}(V,j-1)) \text{ for some } \tau < j \leq 2\tau \\ 0, & \text{if } r = \hat{f}(V,\tau) \\ -\tau + j - 1, & \text{if } r \in (\hat{f}(V,j), \hat{f}(V,j-1)] \text{ for some } 0 < j \leq \tau \\ -\tau - 1, & \text{otherwise} \end{cases}$$

Sample an r from $[D]$ with probability proportional to $\exp\left(\frac{\epsilon}{2} s(V, r)\right)$, denoted by $\tilde{r}$
return $M(V) = \tilde{r}$

IV. Experiment

A. The Race-to-the-Top Mechanism

I conduct experiments on general SPJA queries with FK constraints on TPC-H benchmark.

1) Setup: Datasets. The TPC-H benchmark consists on eight relations: Region(RK), Nation(RK,NK), Customer(NK,CK), Orders(CK,OK), Supplier(NK,SK), Part(PK), PartSupp(SK,PK), Lineitem(SK,PK,OK,LN). I used datasets with scale factors 2^{-1}. The one with scale factor 2^{-1} has 5×10^3 suppliers, 7.5×10^4 customers, and around 3×10^6 lineitems.

Queries. For TPC-H data, I chose the following four queries:

- Q12 counts the number of lineitems. For this SJA counting query, I only consider the customers as primary relation and $\psi(q_k(t)) = 1$.
- Q18 returns the total quantity of lineitems purchased by all customers. This is a SJA aggregation query similar to Q12. The customer is considered as primary relationship and $\psi(q_k(t)) = quantity\ of\ q_k(t)$.
- Q5 counts the number of lineitems where the customer and supplier are from the same nation. For this SJA counting query, both customers and suppliers are considered primary relationship, who jointly contribute the lineitems. The $\psi(q_k(t)) = 1$.
- Q10 returns the order key of certain order made by customers. For this SPJA query, the customer is considered as primary relationship and $\psi(q_k(t)) = 1$ due to the distinct count.

The datasets with scale 2^{-1} (default scale) has about 3.75 million tuples, and I set $GS_Q = 5 \times 10^5$ in advance. As for the truncation threshold τ needed for LP mechanism is selected from $\{2, 4, 8, \cdots GS_Q\}$ to achieve better results.

Implementation details. For the queries mentioned above, I rewrite the queries to keep only the join structure and corresponding count/aggregation/projection operators and add primary relation as select attributes.

Example. Consider the following query Q5:
select n_name, sum(l_extendedprice * (1 - l_discount)) as revenue from customer, orders, lineitem, supplier, nation, region
where c_custkey = o_custkey and l_orderkey = o_orderkey and l_suppkey = s_suppkey and c_nationkey = s_nationkey
and s_nationkey = n_nationkey and n_regionkey = r_regionkey and r_name = '[REGION]' and o_orderdate $\geq$ date '[DATE]'
and o_orderdate ¡ date '[DATE]' + interval '1' year group by n_name order by revenue desc;

I rewrite it as:
select c_custkey, s_suppkey from customer, orders, lineitem, supplier, nation, region
where c_custkey = o_custkey and l_orderkey = o_orderkey and l_suppkey = s_suppkey and c_nationkey = s_nationkey
and s_nationkey = n_nationkey and n_regionkey = r_regionkey

Experimental environment. All experiments were conducted on a laptop with Windows server. Each experiment was repeated 100 times and I report the average running time. The errors are less stable due to the random noise, so I remove the best 20 and worst 20 runs, and report the average error of the remaining 60 runs. The failure probability β in R2T is set to 0.1. The default privacy budget is $\epsilon = 1$.

Query type		Single primary private relation	Multiple primary private Relation	Aggregation	Projection
Query		Q12	Q5	Q18	Q10
Query result		2,999,671	120,257	76,520,242	750,000
R2T	Relative Error(%)	0.0263	0.1625	0.5235	0.4461
	Time(s)	25.6	2.1	33.3	47.6

TABLE I Result of R2T mechanism on SQL queries.

2) Experiment Result: I tested all four queries mentioned above and the results are shown in Table I. From the results, we can see that relative error and running time increases as the query result increase and R2K mechanism runs more time on SPJA queries compared to SJA queries. But the relative error and time is still small enough in terms of utility. Overall, R2K mechanism has a robust performance on SPJA queries. The time cost is bigger than the results from Wei Dong et al. [1] as the implementation of early stop is missing. Future work can be done to implement early stop to reduce running time.

B. The Shifted Inverse Mechanism

I conduct experiments on k-selection function on TPC-H benchmark.

1) Setup: Datasets. It's the same with the one used in the Race-to-the-Top Mechanism with scale factors 2^{-1}. The one with scale factor 2^{-1} has 5×10^3 suppliers, 7.5×10^4 customers, and around 3×10^6 lineitems.

Queries. For TPC-H data, I chose the following one queries:

- Q18 returns the total quantity of lineitems purchased by all customers. For this query, I only consider the customers as users, hence $l = 1$. $V(\{u\})$ is a multiset of quantities, and $f(V) = \sum_{t \in S(V)} t$.

Implementation details. For Q18, I rewrite the queries to only keep the join structure and selection attributes, but modify $f(V)$ to return the kth largest t. Therefore, the data are actually the value of t instead of the total aggregation result.

For Q18, the l is equal to 1. In this case, the $\hat{f}_k(V, j)$ needed for the Shifted Inverse mechanism can be computed by the following algorithm:

Algorithm 2: Computing $\hat{f}(V, j)$'s for k-selection with $\ell = 1$

```
Input  : The instance V, and the parameters k and τ
Output: f̂_k(V, j)'s for j ∈ [2τ]
count(u) ← 0 for all u ∈ U_V, f̂_k(V, j) ← 0 for j ∈ [2τ]
j ← 0
for i ← 1, ..., |V| do
    if sum of all but the largest j counters count(u) ≤ k − 1 then
        f̂_k(V, j) ← t(i)
    else
        j ← j + 1
        if j > 2τ then
            return f̂_k(V, j), j ∈ [2τ]
        else
            f̂_k(V, j) ← t(i)
        end
    end
    Increment count(u) for the user u contributing t(i)
end
```

The Shifted Inverse algorithm are time-consuming when computing $s(V, r)$. Since the function is monotonic, the $\hat{f}_k(V, j)$ computed from algorithm 2 is a sequence in a descending order and binary search can be used to reduce the running time of computing $s(V, r)$.

Experimental environment. All experiments were conducted on a laptop with Windows server.The failure probability β in the Shifted Inverse mechanism is set to $1/3$. The default privacy budget is $\epsilon = 1$. I set $D = 10^5$ for Q18.

2) Experiment Results: The errors (both absolute errors and rank errors) of (Approx)ShiftedInverse are shown in Table II. From the results, we can see that relative error and rank error is very small. In terms of utility, the Shifted Inverse mechanism has a robust performance on k-selection on TPC-H benchmark. The fact that $V(\{u\})$ is a multiset of quantities may also lead to low error. Future work are needed to implement quantile query on different queries. monotonic functions in more complex scenarios, particularly when $l \geq 2$, could yield valuable insights and enhance the performance.

Query		Q18			
$\alpha - Quantile(\%)$		100	75	50	25
Query result		50	38	26	13
(Approx)ShiftedInverse	Relative Error(%)	0	0	0.0384	0
	Rank Error	0	0	171	0

TABLE II Result of the (Approx)ShiftedInverse mechanism on on quantile queries on TPC-H benchmark.

V. Conclusion

This project focuses on the implementation of two primary mechanisms, namely the R2T and the Shifted Inverse mechanisms, with a particular emphasis on TPC-H benchmarks. The R2T mechanism was executed on four distinct queries, specifically Q5, Q10, Q12, and Q18. The outcomes consistently demonstrate that the R2T mechanism exhibits robust performance in terms of both utility and execution time, particularly in scenarios involving foreign key constraints and the projection operator.

In the case of the Shifted Inverse mechanism, the k-selection function was employed as the monotonic function in the implementation of the approximate Shifted Inverse mechanism. The error (both relative error and rank error) observed across four different quantiles were nearly zero, thereby underscoring the exceptional performance of the Shifted Inverse mechanism. This evidence strongly suggests the efficacy of these mechanisms in handling queries from TPC-H benchmarks.

Potential avenues for future research could include the incorporation of an early stop feature for the R2T mechanism, which could significantly reduce execution time. Additionally, the exploration of other monotonic functions and other queries in more complex scenarios, particularly when $l \geq 2$, could yield valuable insights and enhance the performance. These enhancements could further optimize the efficiency and effectiveness of the R2T mechanism and the Shift Inverse Mechanism in handling complex scenarios.

Acknowledgment

I would like to express my sincere gratitude to Prof Yi Ke for his invaluable guidance, profound patience, and constant encouragement throughout the course of this research work. Help from Yu Jianzhe and Fang Juanru is also greatly appreciated for clearing my doubts.

References

[1] DONG W, FANG J, YI K, et al. R2T: Instance-optimal Truncation for Differentially Private Query Evaluation with Foreign Keys[C/OL]//Proceedings of the 2022 International Conference on Management of Data. 2022. http://dx.doi.org/10.1145/3514221.3517844. DOI:10.1145/3514221.3517844.

[2] KONG H, DONG W, YI K. Shifted Inverse: A General Mechanism for Monotonic Functions under User Differential Privacy Juanru Fang[J].

[3] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Privatesql: a differentially private sql query engine. Proceedings of the VLDB Endowment, 12(11):1371–1384, 2019.

[4] Kareem Amin, Alex Kulesza, Andres Munoz, and Sergei Vassilvtiskii. Bounding user contributions: A bias-variance trade-off in differential privacy. In International Conference on Machine Learning, pages 263–271. PMLR, 2019.

[5] Galen Andrew, Om Thakkar, H Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. arXiv preprint arXiv:1905.03871, 2019.

[6] Ziyue Huang, Yuting Liang, and Ke Yi. Instance-optimal mean estimation under differential privacy. In NeurIPS, 2021.

[7] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. arXiv preprint arXiv:1710.06963, 2017.

[8] Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X Yu, Sashank J Reddi, and Sanjiv Kumar. Adaclip: Adaptive clipping for private sgd. arXiv preprint arXiv:1908.07643, 2019.

[9] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In Proceedings of the twenty-sixth ACM SIGMODSIGACT-SIGART symposium on Principles of database systems, pages 273–282, 2007.

[10] Jaroslaw Błasiok, Mark Bun, Aleksandar Nikolov, and Thomas Steinke. Towards instance-optimal private query release. In Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 2480–2497. SIAM, 2019.

[11] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In Proceedings of the 2016 International Conference on Management of Data, pages 123–138, 2016.

[12] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In Advances in Neural Information Processing Systems, pages 2339–2347, 2012.

[13] Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. The VLDB journal, 24(6):757–781, 2015.

[14] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In Proceedings of the forty-fifth annual ACM symposium on Theory of computing, pages 351–360, 2013.

[15] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Practical differentially private release of marginal contingency tables. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pages 1435–1446.

[16] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Understanding hierarchical methods for differentially private histograms. Proceedings of the VLDB Endowment, 6(14):1954–1965, 2013.

[17] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. IEEE Transactions on knowledge and data engineering, 23(8): 1200–1214, 2010.

[18] Xiaojian Zhang, Rui Chen, Jianliang Xu, Xiaofeng Meng, and Yingtao Xie. Towards accurate histogram publication under differential privacy. In Proceedings of the 2014 SIAM international conference on data mining, pages 587–595. SIAM, 2014.

[19] Myrto Arapinis, Diego Figueira, and Marco Gaboardi. Sensitivity of counting queries. In International Colloquium on Automata, Languages, and Programming (ICALP), 2016.

[20] Frank D McSherry. Privacy integrated queries: an extensible platform for privacypreserving data analysis. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pages 19–30, 2009.
[21] Arjun Narayan and Andreas Haeberlen. Djoin: Differentially private join queries over distributed databases. In USENIX Symposium on Operating Systems Design and Implementation, pages 149–162, 2012.
[22] Catuscia Palamidessi and Marco Stronati. Differential privacy for relational algebra: Improving the sensitivity bounds via constraint systems. In QAPL, 2012.
[23] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating data to sensitivity in private data analysis. Proceedings of the VLDB Endowment, 7(8), 2014.
[24] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. Differentially private sql with bounded user contribution. Proceedings on privacy enhancing technologies, 2020(2):230–250, 2020.
[25] Yuchao Tao, Xi He, Ashwin Machanavajjhala, and Sudeepa Roy. Computing local sensitivities of counting queries with joins. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pages 479–494, 2020.
[26] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In Theory of Cryptography Conference, pages 457–476. Springer, 2013.