DProvSQL: Privacy Provenance Framework for Differentially Private SQL Engine

Shufan Zhang
University of Waterloo
Waterloo, Canada
shufan.zhang@uwaterloo.ca

Runchao Jiang
Meta Platforms
USA
runchaojiang@fb.com

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

ABSTRACT

Recent years have witnessed the adoption of differential privacy (DP) in practical database query systems. Such systems, like PrivateSQL and FLEX, allow data analysts to query sensitive data while providing a rigorous and provable privacy guarantee. However, existing systems may use more privacy budgets than necessary in certain cases where different data analysts with different privilege levels ask a similar or even the same query. In this work, we propose *DProvSQL*, a fine-grained privacy provenance framework that tracks the privacy loss to each single data analyst and we build algorithms that make use of this framework to maximize the number of queries that could be answered. Preliminary empirical results show that our approach can answer 6x more queries than the baseline approach on average meanwhile the answer from our approach is 1.5x to 5.6x more accurate.

1 INTRODUCTION

Data collected by companies and organizations can contain sensitive information. With the growing attention on data privacy and the development of privacy protection regulations such as GDPR [12], companies wish to allow external and internal data analysts to make use of the data without compromising the privacy of data contributors. To address the privacy issues, differential privacy (DP) [4] has been considered as a promising and the *de facto* standard for privacy-preserving data analysis these days. In the framework of DP theory, privacy is parameterized by a variable ϵ (or (ϵ, δ) in approximate-DP), called the *privacy budget*, which controls the privacy protection level over the data. By carefully injecting controllable noise, researchers or algorithm designers can prove the output of an algorithm (or mechanism, using DP terminology) can only reveal bounded information of any individual in the dataset, and thus this mechanism satisfies the notion of DP.

Recent years have witnessed the adoption of DP from a pure theoretical perspective to practical systems [10]. A plethora of systems are proposed and developed to enforce DP for database management and SQL queries in real-world, including PINQ [9], FLEX [7], PrivateSQL [8], GoogleDP [1], and Chorus [6]. Despite these significant efforts made in existing DP systems, these systems regard the data analysts as a unified entity querying and obtaining the results from the system. Thus the privacy analysis is stark and not personalized as per data analyst. We argue that data analysts can have different privilege levels in practical scenarios on accessing the query results, which requires a system to enforce a finer-grained privacy tracking. We illustrate this problem as below.

Motivating Example. We consider a protected database (Fig. 1) in a corporation that records the data of its employees. This database contains sensitive information about the employees, such as salary and age. Other attributes, like department and state, are less sensitive and considered public information. Two queriers, A, which is an *internal* application in the company, and B, who represents an *external* application outside the company, ask the following aggregation query about the average salary of employees with age less than 30 for each department:

eid	ename	age	state	role	salary	department
1	Alice Land	24	AZ	SDE	20k	Infrastructure
2	Bobby Hill	28	CA	HR	30k	Infrastructure
3	Carrie Sea	37	WA	Manager	40k	Sales
4	Danny Des	26	CA	SDE	32k	Cloud

Figure 1: The Employee Table

```
SELECT AVERAGE(salary) FROM employee
WHERE age < 30
GROUP BY department;
```

In this example, the average salary is sensitive information and the query result should be protected with DP. The queriers, application A and application B, cannot access the raw data in the employee table but they are able to learn some noisy answers of the aggregation query. These two applications differ in their privilege of accessing the database and the query results — the internal application (i.e., A) can have a higher privilege level than the external application (i.e., B). In terms of DP, the privacy leakages to A and B through the noisy answers are different, and the internal application A should be able to see more accurate results.

This use case is common in practice for tech companies who need to use sensitive data for internal applications like anomaly detection and also would like to invite external researchers to analyze their data but with more noise. However, the existing DP systems do not provide tools to distinguish these queriers and track their perspective privacy loss. A naive tracking and answering of each querier's queries independent of the others can waste privacy budgets, i.e., fewer queries can be answered accurately under a given total privacy budget. Additionally, if we assume (all or a subset of) data analysts can communicate with each other and collude, we would like to control the overall privacy loss. This is due to the underlying privacy implication, where the compromised data analysts asking the same query are able to infer a more accurate result of sensitive data, according to the sequential composition theorem of DP [4]. As one can see, this process is not convenient nor secure as those systems are not dedicatedly designed for the use case that we are showing.

To tackle with these challenges, we propose *DProvSQL*, a new privacy provenance framework for DP SQL engine that fits in the

multi-analysts scenario. Instead of answering queries from each data analyst independently, *DProvSQL* generate DP synopses for a set of views, so that the queries can be answered based on these synopses and these synopses can be dynamically updated according to data analysts' requests. Furthermore, *DProvSQL* enables a privacy provenance table that enforces a fine-grained privacy provenance as per each data analyst and per view. The privacy provenance table is associated with privacy constraints so that constraint-violating queries will be rejected. Making use of this privacy provenance framework, we build the mechanisms, maintaining global (viz., as per view) and local (viz., as per analyst) DP synopses (i.e., materialized results for views) to answer as many query presented to the system as possible. Our preliminary empirical results show a significant improvement over the baseline method where queries are answered independently, in terms of the number of queries being answered and the minimum expected error among the answers.

2 BACKGROUND

We consider the database instance D that stores sensitive data with a set of schema/relations $\mathcal{R} = \{R_1, \ldots, R_l\}$. The domain of all database instances is denoted by $\mathcal{D}$. We introduce and summarize the related definitions about differential privacy as follows.

Definition 2.1 (Differential Privacy (DP)). We say that a randomized algorithm $M : \mathcal{D} \to \mathcal{O}$ satisfies (ϵ, δ)-differential privacy, if for any two databases D and D' that differ in only 1 tuple, and all $O \subseteq \mathcal{O}$, we have $\Pr[\mathcal{M}(D) \in O] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in O] + \delta$, where the probability is taken over the randomness used by the mechanism $\mathcal{M}$.

Definition 2.2 (l_2-Global Sensitivity). For a function $q : \mathcal{D} \to \mathbb{R}^d$ and all $D, D' \in \mathcal{D}$, the l_2 global sensitivity of this function is defined as $\Delta q = \max_{D,D':d(D,D')<=1} \|q(D) - q(D')\|_2$, where $d(\cdot,\cdot)$ denotes the number of tuples that D and D' differ.

Definition 2.3 (Analytic Gaussian Mechanism [2]). Let $q : \mathcal{D} \to \mathbb{R}^d$ be an arbitrary d-dimensional function. The analytic Gaussian mechanism $\mathcal{M}(x) = q(x) + \eta$ where $\eta \sim \mathcal{N}\left(0, \sigma^2\right)$ is (ϵ, δ)-DP if and only if $\Phi_{\mathcal{N}}\left(\frac{\Delta_2 q}{2\sigma} - \frac{\epsilon\sigma}{\Delta_2 q}\right) - e^{\epsilon}\Phi_{\mathcal{N}}\left(-\frac{\Delta_2 q}{2\sigma} - \frac{\epsilon\sigma}{\Delta_2 q}\right) \leq \delta$.

Definition 2.4 (Data Utility). For a query $q : \mathcal{D} \to \mathbb{R}^d$ and a mechanism $\mathcal{M} : \mathcal{D} \to \mathbb{R}^d$, the *data utility* of mechanism $\mathcal{M}$ is measured as the expected squared error, $v = \mathbb{E}[q(D) - \mathcal{M}(D)]^2$. For the (analytic) Gaussian mechanism, the expected squared error equals to its variance, that is, $v = \sigma^2$.

3 PROBLEM SETUP

We consider the **multi-analysts** setting, where there are multiple data analysts $\mathcal{A} = \{A_1, \ldots, A_m\}$ who want to ask queries on the database D. The data curator who manages the database wants to ensure that the sensitive data is properly and privately shared with the data analysts $A_1, \ldots, A_m$. In our threat model, the data analysts can adaptively select and submit arbitrary queries to the system to infer sensitive information about individuals in the protected database. In addition, in our multi-analysts model, data analysts may submit the same query and collude to leak more information about the sensitive data.

Differing from prior work [6, 8], these analysts have different privilege levels. We would like to define the privacy per analyst provenance framework as a DP variant that guarantees different levels of privacy loss to the analysts.

Definition 3.1 (Multi-analyst DP). We say a randomized mechanism $\mathcal{M} : \mathcal{D} \to (O_1, \ldots, O_m)$ satisfies $[(A_1, \epsilon_1, \delta_1), ..., (A_m, \epsilon_m, \delta_m)]$-multi-analyst-DP if for any two databases D and D' that differ in only 1 tuple, any $i \in [m]$, and all $O_i \subseteq \mathcal{O}_i$, we have

$$\Pr[\mathcal{M}(D) \in O_i] \leq e^{\epsilon_i} \Pr[\mathcal{M}(D') \in O_i] + \delta_i,$$

where O_i are the output released to the ith analyst.

The multi-analyst DP framework supports the composition across different algorithms, indicated by the following theorem.

Theorem 3.2. *Given two mechanisms $\mathcal{M}_1 : \mathcal{D} \to (O_1, \ldots, O_m)$ that satisfies $[(A_1, \epsilon_1, \delta_1), ..., (A_m, \epsilon_m, \delta_m)]$-multi-analyst-DP, and $\mathcal{M}_2 : \mathcal{D} \to (O'_1, \ldots, O'_m)$ that satisfies $[(A_1, \epsilon'_1, \delta'_1), ..., (A_m, \epsilon'_m, \delta'_m)]$-multi-analyst-DP, then the mechanism $\mathcal{M}_1 \circ \mathcal{M}_2$ gives the $[(A_1, \epsilon_1 + \epsilon'_1, \delta_1 + \delta'_1), ..., (A_m, \epsilon_m + \epsilon'_m, \delta_m + \delta'_m)]$-multi-analyst-DP guarantee.*

Under this new multi-analyst DP framework, several research questions are raised and motivate our work.

Question 1: worst-case privacy analysis across analysts. If the data analysts do not collude, we can use sequential composition (Theorem 3.2) to track the privacy loss to each data analyst for all queries this analyst asks. However, if all or a subset of data analysts collude or are compromised by an adversary, how to design algorithms to account the privacy loss to the colluded analysts?

When all the analysts are compromised by an adversary, the privacy loss to this adversary is upper bounded by $(\sum \epsilon_i, \sum \delta_i)$ and it is lower bounded by $(\max \epsilon_i, \max \delta_i)$, where (ϵ_i, δ_i) is the privacy loss to the ith analyst. In this paper, we would like to design an algorithm that achieves the lower privacy bound.

Question 2: dynamic budget allocation across views. Prior works for DP query answering [5, 8] often assume the availability of a representative workload that can capture the queries interested by the data analysts in the future. Given such a representative workload, a system [8] can select a set of views $V_1, V_2, \ldots, V_l$ such that each query in this workload can be answered with a linear query on a single view, and then generate a DP synopsis for each of the selected view. Given a total privacy budget, prior work [8] splits the privacy budget equally or proportional to the sensitivity of the view to achieve equal accuracy rate. However, some views may require higher accuracy than the others, depending on the requests of the data analysts. Therefore, it is important to design an algorithm that can dynamically allocate privacy budgets to the given views and update their corresponding DP synopses overtime.

Definition 3.3 (View and query answerability [8]). Given a database instance D, a materialized (histogram) view $\mathbf{V}(D)$ (or V) is a set of results of counting query about some specific domain values over some attributes in the database instance. For a query q over the database instance D, if there exists a query q' over the histogram view V such that $q(D) = q(\mathbf{V}(D))$, we say the query q is answerable over the view V.

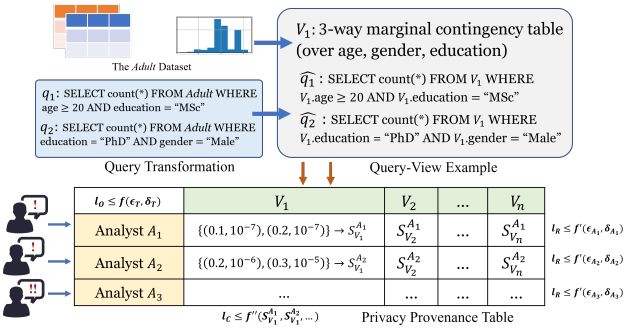

Figure 2: The Privacy Provenance Table: Data Structure

4 SYSTEM OVERVIEW

To support the multi-analysts use case and to answer the aforementioned research questions, we identify the following three principles for a differentially private SQL query system and propose a system *DProvSQL* that follow these principles.

Principle 1: fine-grained privacy provenance. The system should be able to track the privacy budget allocated as per each data analyst and per each view in a fine-grained way. The system should additionally enable a mechanism to compose privacy loss across data analysts and the queries they ask.

Principle 2: view-based privacy management. The queries are answered based on differentially private views/synopses in the system. View is the minimum data object that we keep track of its privacy loss and the views can be updated dynamically if higher data utility is required. The privacy budgets spent on different views during the updating process depend on the incoming queries.

Principle 3: maximum query answering. The system should be tuned to answer as many queries as possible, without violating the privacy constraint specified by the data curators as per data analyst and per view based on their privilege levels.

4.1 Privacy Provenance Table

To meet the first two principles, we propose a privacy provenance table for *DProvSQL*, which is inspired by the access matrix model in access control literature [11], to track the privacy loss per analyst and per view, and further bound the privacy loss. Particularly, in our model, the state of the overall privacy loss of the system is defined as a triplet $(\mathcal{A}, \mathcal{V}, P)$, where $\mathcal{A}$ denotes the set of data analysts and $\mathcal{V}$ represents the list of query-views maintained by the system. We denote by P the privacy provenance table, defined as follows.

Definition 4.1 (Privacy Provenance Table). The privacy provenance table P is a matrix that tracks the privacy loss of the database as per each data analyst in $\mathcal{A}$ and each query-view in $\mathcal{V}$. Each row of P corresponds to a data analyst and each column of P corresponds to a query-view. Each entry of the matrix $P[A_i, V_j]$. records the current cumulative privacy loss, on view V_j to analyst A_i.

The table also includes a set of row/column/table privacy constraints, Φ. A *row constraint* for ith row, denoted by ϕ_{A_i}, refers to the total privacy loss to an particular data analyst A_i (according to his/her privilege level) while a *column constraint* for the jth column, denoted by ϕ_{V_j} refers to as the allowed maximum privacy loss to a specific view V_j. We use the *table constraint* over P, denoted by ϕ_P, to specify the overall privacy loss that is allowed for the protected database. The privacy constraints can be correlated. The internal restriction over these constraints indicates that the entry privacy loss cannot exceed row and column constraints while the row/column constraints cannot exceed the overall table constraint.

Example 4.2. Figure 2 shows an example of the privacy provenance table. We consider a histogram view V_1 is a 3-way marginal table over attributes (age, gender, and education). The queries q_1 and q_2 can be transformed into $\hat{q}_1$ and $\hat{q}_2$ that are answerable using V_1. Three data analysts A_1, A_2 and A_3 with different privilege levels are recorded in privacy provenance table and we track every privacy budget spent over time on the views.

To meet the third principle, we formulate the *maximum query answering problem* based on the privacy provenance table.

Problem 1. *Given a privacy provenance table $(\mathcal{A}, \mathcal{V}, P)$, at each time, a data analyst $A_i \in \mathcal{A}$ submits the query with a utility requirement (q_i, v_i), where the transformed $\hat{q}_i \in \mathcal{V}$, how to answer as many queries as possible without violating the row/column/table privacy constraints in P while meeting the utility requirement per query?*

WLOG, we assume the utility requirements per query requested by the same data analyst never decreases. That is, for the same query, the data analyst is only interested in a more accurate result. We will discuss the other cases in our full paper.

4.2 DP Mechanism Design

DP Synopses. To solve the maximum query answering problem, for each view $V \in \mathcal{V}$, *DProvSQL* maintains a *global DP synopsis* with a cost of (ϵ, δ), denoted by $V^{\epsilon,\delta}(D)$ or V^{ϵ}, where D is the database instance. For simplicity, we drop δ by considering the same value for all δ and D. For this veiw, *DProvSQL* also maintains a *local DP synopsis* for each analyst $A_i \in \mathcal{A}$, denoted by $V^{\epsilon'}_{A_i}$, where the local synopsis is always generated from the global synopsis V^{ϵ} of the view V by adding more noise. Hence, we would like to ensure $\epsilon > \epsilon'$. This local DP synopsis $V^{\epsilon'}_{A_i}$ will be used to answer the queries asked by the data analyst A_i.

First, we introduce our *additive Gaussian Mechanism* (additive GM) that releases a local DP synopsis $V^{\epsilon'}_{A_i}$ from a given global synopsis V^{ϵ}, where V^{ϵ} is generated by a Gaussian mechanism. Given the privacy guarantee ϵ (and δ) and the sensitivity of the view, the Gaussian mechanism can calculate a proper variance σ for adding noise and ensuring DP. The additive GM calculates σ and σ' based on ϵ and ϵ' respectively, and then generates the local synopsis $V^{\epsilon'}_{A_i}$ by injecting independent noise drawn from $\mathcal{N}(0, \sigma'^2 - \sigma^2)$ to the global synopsis V^{ϵ}. As the global synopsis is hidden from all the analysts, the privacy loss to the analyst A_i is ϵ'. Even if all the analysts collude, the maximum privacy loss is bounded by the budget spent on the global synopsis.

Second, when the global DP synopsis V^{ϵ} is not sufficiently accurate to handle a local synopsis, *DProvSQL* spends additional privacy budget $\Delta\epsilon$ to update the global DP synopsis to $V^{\epsilon+\Delta\epsilon}$. We still consider Gaussian mechanism, which generates an intermediate DP synopsis $V^{\Delta\epsilon}$ with a budget $\Delta\epsilon$. Then we combine the previous synopses with this intermediate synopsis with a weight proportional to their budget. That is, for the n-th release, $V' = \sum_{i=1}^{n-1} w_i V^{\epsilon_i} + w_n V^{\Delta\epsilon}$

where $\sum_{i=1}^{n} w_i = 1$ and $v = \sum_{i=1}^{n-1} w_i^2 \sigma_i^2 + w_n^2 \sigma^2$ (whose closed-form solution is $w_i = v/\sigma_i^2$, i.e. $w_i \propto \epsilon_i$).

When a local DP synopsis $V_{A_i}^{\epsilon'}$ is not sufficiently accurate to handle a query, but the budget ϵ' is still smaller than the budget for the global synopsis, *DProvSQL* generates an intermediate local synopsis $V_{A_i}^{\Delta\epsilon}$ from the global synopsis using additive GM. Then it combines $V_{A_i}^{\Delta\epsilon}$ with the previous local synopsis in a similar way for the global synopses, which leads to a new local synopsis $V_{A_i}^{\epsilon'+\Delta\epsilon}$.

Algorithm Overview. Algorithm 1 summarizes how *DProvSQL* uses the DP synopses to answer incoming queries. At the system setup phase (line 1-2), the system (or data curator) initializes the privacy provenance table by setting the privacy budget as per entry as 0 and the row/column/table constraints Φ. The system initializes empty global/local synopses for each view. The data analyst specifies a query q_i with its desired utility requirement v_i (line 4). Once the system receives the request, it selects the suitable view to answer this query (line 5) and uses the function privacyTranslate() to find the minimum privacy budget ϵ_i for V to meet the utility requirement of q_i (line 6). Then, *DProvSQL* checks if answering q_i with budget ϵ_i will violate the privacy constraints Φ (Line 7).

If this sanity check passes and the required privacy budget ϵ_i is greater than the current privacy budget ϵ used by the global synopsis V^{ϵ} (line 8), there is no way to update the local synopsis from the global synopsis to answer this query accurately. Thus, *DProvSQL* will update the global synopsis to the larger privacy budget V^{ϵ_i} (line 9). Then, *DProvSQL* will update the local synopsis based on the present global synopsis to $V_{A_i}^{\epsilon_i}$ (line 10) and the corresponding entry in the privacy provenance table, $P[A_i, V] = \epsilon_i$ (line 11). Last, *DProvSQL* uses the updated local synopsis to answer query q_i and returns the answer to the data analyst (line 12). If the sanity check fails, *DProvSQL* rejects the query (line 14). We will elaborate the details in our full paper, but the privacy guarantees are as follows.

Theorem 4.3. *Given the privacy provenance table and its constraint specifications, $\Phi = \{\phi_{A_i} | A_i \in \mathcal{A}\} \cup \{\phi_{V_j} | V_j \in \mathcal{V}\} \cup \{\phi_P\}$, Algorithm 1 ensures $[\ldots, (A_i, \phi_{A_i}, \delta), \ldots]$-multi-analyst-DP; it also ensures ϕ_{V_j}-DP for view $V_j \in \mathcal{V}$ and overall ϕ_P-DP if all the data analysts collude.*

Algorithm 1: System Overview

Input: Analysts $\mathcal{A} := A_1, \ldots, A_n$; Database instance D; Privacy provenance table P.

1 Data curator sets up the privacy provenance table P with row/column/table constraints Φ.
2 Initialize all the synopses for $V \in \mathcal{V}$
3 **repeat**
4 Receive (q_i, v_i) from data analyst A_i
5 Select $V \in \mathcal{V}$ to answer query q_i
6 $\epsilon_i \leftarrow$ privacyTranslate(q_i, v_i, V)
7 **if** constraintCheck$(P, A_i, V, \epsilon_i, \Phi)$ **then**
8 **if** $\epsilon_i > \epsilon$ *for global synopsis* V^{ϵ} **then**
9 Update global synopsis to $V^{\epsilon \leftarrow \epsilon_i}$ with D
10 Update local synopsis to $V_{A_i}^{\epsilon' \leftarrow \epsilon_i}$ with V^{ϵ}
11 Update privacy provenance table $P[A_i, V] \leftarrow \epsilon_i$
12 Answer q_i with $V_{A_i}^{\epsilon'}$ and return answer r_i to A_i
13 **else**
14 reject the query q_i
15 **end**
16 **until** *No more queries sent by analysts*

5 PRELIMINARY RESULTS

As a proof of concept, we implement our approach and present the preliminary empirical results on Adult dataset [3]. We set up three data analysts, two of which has low privilege level (row constraints $\phi_{A_1} = \phi_{A_2}$=1,) and the other one with high privilege level (ϕ_{A_3}=4). For simplicity, we assume all data analysts ask the same query:

```
SELECT COUNT(*) FROM adult
WHERE age >= 39 AND education = “Bachelors”
GROUP BY age, education, gender;
```

The data analysts keep on submitting this query with higher accuracy requirement (specified by the expected squared error, starting from 40, each time decreasing by 1) over time. In our approach, we build a 3-way contingency table over attributes age, gender, and education as a view to answer the query, with a column constraint $\phi_V = 4$. We enable the column constraints over different analysts while the row constraint is naturally enforced since we only have a single view in the preliminary experiments. We compare our approach to a baseline, which regard each query from analysts as a separated query and answer it independently. We measure the number of queries that the system could support for each data analyst until no more queries can be handled without violating the privacy constraints, and the minimum expected error among all queries returned to each data analyst. As shown in Table 1, our approach can answer 6x more queries than the baseline on average meanwhile the answer from our approach is 1.5x to 5.6x more accurate. Our approach performs significantly better than the baseline, because in the baseline, every query-answering is independent, whereas the usage and management of global/local DP synopses in our approach enables correlated DP noise to the query results. Our mechanism can therefore avoid wasting budget and hence answer more queries accurately.

Table 1: The comparison between our approach and baseline approach (in terms of the number of queries being answered and the minimum expected error of answers).

	Analyst 1	Analyst 2	Analyst 3
Baseline	2 (v=39)	2 (v=39)	7 (v=34)
Our Approach	15 (v=26)	15 (v=26)	35 (v=6)

6 CONCLUDING REMARKS

We propose *DProvSQL*, a privacy provenance framework for differentially private SQL engine that tracks the privacy loss to each supported data analyst. *DProvSQL* can avoid wasting privacy budgets on the same query asked by different data analysts and prevent risky queries that exceed the privilege level as per data analyst. The prototype of *DProvSQL* can be extended and further developed to support a variety of queries including (cross-relation) JOINs, and different types of complicated DP mechanisms, so that the fully-functioning system can be useful in the industry and in other real-world scenarios.

REFERENCES

[1] Kareem Amin, Jennifer Gillenwater, Matthew Joseph, Alex Kulesza, and Sergei Vassilvitskii. 2022. Plume: Differential Privacy at Scale. *arXiv preprint arXiv:2201.11603* (2022).

[2] Borja Balle and Yu-Xiang Wang. 2018. Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*. PMLR, 394–403.

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

[4] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407.

[5] Moritz Hardt and Guy N Rothblum. 2010. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 61–70.

[6] Noah Johnson, Joseph P Near, Joseph M Hellerstein, and Dawn Song. 2020. Chorus: a programming framework for building scalable differential privacy mechanisms. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 535–551.

[7] Noah Johnson, Joseph P Near, and Dawn Song. 2018. Towards practical differential privacy for SQL queries. *Proceedings of the VLDB Endowment* 11, 5 (2018), 526–539.

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

[9] Frank D McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 19–30.

[10] Joseph P Near and Xi He. 2021. Differential Privacy for Databases. *Foundations and Trends® in Databases* 11, 2 (2021), 109–225.

[11] Pierangela Samarati and Sabrina Capitani de Vimercati. 2000. Access control: Policies, models, and mechanisms. In *International School on Foundations of Security Analysis and Design*. Springer, 137–196.

[12] Paul Voigt and Axel Von dem Bussche. 2017. The EU general data protection regulation (GDPR). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10 (2017), 3152676.