Catch a Blowfish Alive: A Demonstration of Policy-Aware Differential Privacy for Interactive Data Exploration

Jiaxiang Liu
University of Waterloo
j632liu@uwaterloo.ca

Karl Knopf
University of Waterloo
kknopf@uwaterloo.ca

Yiqing Tan
University of Waterloo
y57tan@uwaterloo.ca

Bolin Ding
Alibaba Group
bolin.ding@alibaba-inc.com

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

ABSTRACT

Policy-aware differential privacy (DP) frameworks such as Blowfish privacy enable more accurate query answers than standard DP. In this work, we build the first policy-aware DP system for interactive data exploration, BlowfishDB, that aims to (i) provide bounded and flexible privacy guarantees to the data curators of sensitive data and (ii) support accurate and efficient data exploration by data analysts. However, the specification and processing of customized privacy policies incur additional performance cost, especially for datasets with a large domain. To address this challenge, we propose dynamic Blowfish privacy which allows for the dynamic generation of smaller privacy policies and their data representations at query time. BlowfishDB ensures same levels of accuracy and privacy as one would get working on the static privacy policy. In this demonstration of BlowfishDB, we show how a data curator can fine-tune privacy policies for a sensitive dataset and how a data analyst can retrieve accuracy-bounded query answers efficiently without being a privacy expert.

PVLDB Reference Format:
Jiaxiang Liu, Karl Knopf, Yiqing Tan, Bolin Ding, and Xi He. Catch a Blowfish Alive: A Demonstration of Policy-Aware Differential Privacy for Interactive Data Exploration . PVLDB, 14(12): 2859-2862, 2021.
doi:10.14778/3476311.3476363

1 INTRODUCTION

Differential privacy (DP) [3] has arisen as the standard approach for protecting data contributors from privacy violations that could be caused through data exploration. Systems that implement DP mechanisms [5, 8, 9, 12, 13] have been built to support private data analysis. Under DP, the privacy loss is measured by a parameter, ϵ, which guarantees that any neighbouring databases differing in a record have similar output distributions. The smaller the value of ϵ, the closer the output distributions, and hence the stronger the privacy guarantee, usually with a degradation to utility.

However, DP is often too strict to offer useful answers in many applications [1, 11], as relaxations of the privacy parameter ϵ do not give meaningful utility improvements. A general approach that relaxes DP [1, 2, 7, 14] is to modify the notion of neighbouring inputs by applying a distance metric over the database domain. This relaxation gains a better privacy for utility trade-off. For example, Blowfish privacy [7] includes a privacy "policy graph" over the record domain that specifies which information must be kept secret about individuals.

Proceedings of the VLDB Endowment, Vol. 14, No. 12 ISSN 2150-8097.
doi:10.14778/3476311.3476363

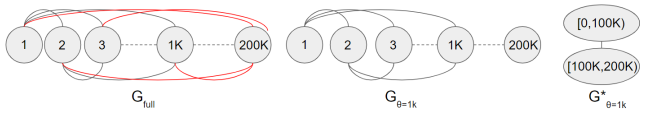


Figure 1: G_{full}: a fully connected policy graph for DP; $G_{\theta=1\text{k}}$: a θ-distance policy graph for Blowfish privacy, e.g., no edges between (1k, 200k) or (1,200k); and $G^*_{\theta=1\text{k}}$: a dynamic Blowfish graph based on a partition $\{[1, 100\mathbf{k}], [100k, 200\mathbf{k})\}$

Example 1: Consider a database of individuals' salary records. Each record takes one value in the domain $\mathcal{T} = \{1, 2, \ldots, 200\text{k}\}$. A DP algorithm ensures that an adversary cannot distinguish whether an individual has a capital gain of x or y, for any $x, y \in \mathcal{T}$. The corresponding policy graph is a complete graph G_{full} as shown in Figure 1. We also show a weaker privacy policy graph, a θ-distance graph $G_{\theta=1\text{k}}$ that connects all domain values that differ at most θ, where $\theta = 1\text{k}$. This specification intends to prevent the adversary from distinguishing close-by values, e.g. (1k vs. 1.2k), but not far-away values, e.g. (1k vs. 200k). By relaxing this privacy guarantee, the expected error per range query can be greatly improved from $O(\log^3 |\mathcal{T}|)$ to $O(\log^3 \theta)$.

Prior works [1, 6, 7, 14] only consider policy graphs over the full domain of a database record. General mechanisms for policy-aware DP [6] require the materialization of the policy graph in a matrix form and hence incur additional storage cost, $O(|\mathcal{T}|^3)$, and computation cost, $O(|\mathcal{T}|^3)$, where $|\mathcal{T}|$ is the domain size. These additional costs lead to challenges when extending policy-aware DP to explore queries over high-dimensional relational data. First, it is unclear how to allow data curators the ability to specify privacy policies over the full domain of a record. There is no tool that helps data curators to understand the privacy-utility trade-offs before setting the privacy policies. Second, though running a policy-aware mechanism offers more accurate answers than the standard DP, the additional storage and computation costs can hurt user experience in the interactive setting.

To address these concerns, we introduce BlowfishDB, a practical data exploration system for policy-aware differential privacy over high dimensional data. BlowfishDB implements a novel form of Blowfish privacy, called *dynamic Blowfish privacy*, a class of privacy policies that can be dynamically generated based on a new partition of the full domain (e.g., $G^*_{\theta=1\text{k}}$ in Figure 1). These privacy policies achieve the same privacy guarantee over the full domain while saving significant storage and computation cost. To use this, BlowfishDB includes a dynamic privacy policy generator that converts high-level semantic privacy polices specified by the data curator to lightweight representations for use in an ϵ-DP mechanism. The policy generator also allows a data curator to efficiently explore the privacy-utility trade-off of their chosen policy, so they are able to make more informed choices. Data analysts can also efficiently query the sensitive data sets with accuracy requirements and even make more queries using BlowfishDB than under a standard DP system, such as APEx[5].

Two demo scenarios are presented for the attendees. They each describe one aspect of the trade-off between privacy and utility that are managed by the BlowfishDB system. The first allows the attendee to act as a data curator and explore the effects of privacy policy selection on the accuracy of the queries. The second scenario allows the attendee to act as a data analyst, and explore how privacy polices affect the utility of the query answers.

2 BACKGROUND

Notation. Consider a relation $R = (att_1, \cdots, att_d)$ with d attributes. Let $dom(att_j)$ denote the domain of attribute att_j, and $\mathcal{T}$ denote the full domain of a record $dom(att_1) \times \cdots \times dom(att_d)$. Let the domain size $|\mathcal{T}| = k$, and D be a database instance consisting of n records $\{t_1, \cdots, t_n\}$. Each record t_i takes a value from $\mathcal{T}$. We can represent the database D by a histogram $x \in \mathbb{R}^k$ over the full domain $\mathcal{T}$. A linear query can be expressed as a linear combination of the counts in x, i.e., wx, where w is a k-dimensional row vector of real numbers. A *workload* of q linear queries can be then represented as $W = \left[w_1, w_2, \ldots w_q\right]^T \in \mathbb{R}^{q\times k}$.

Differential Privacy (DP). A randomized algorithm $\mathcal{M}$ satisfies ϵ-DP if for all possible output set, and any pair of *neighboring databases* (D, D') that differ in a record, we have $\mathcal{S} \in \text{Range}(\mathcal{M})$: $\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq e^\epsilon \Pr[\mathcal{M}(D') \in \mathcal{S}]$. This has become the golden standard privacy notion [8].

A common approach to achieve DP is to perturb the query answers directly using such techniques as the Laplace and Gaussian mechanisms [4]. However, this approach can cause in large errors in the final output. For linear queries, the error of the query answers can be optimized by using the matrix mechanism [10]. This approach first privately answers a representative strategy query workload A as Ax and then reconstructs a solution to the original workload W.

Formally, let A be a $p \times k$ matrix that supports the workload W [10] such that $Wx = WA^+Ax$, where A^+ be the Moore-Penrose pseudo-inverse of A. Let $Lap(\sigma)^p$ represents a p-dimensional vector of independent samples, where each sample is drawn from $\eta \sim \frac{1}{2\sigma}\exp(\frac{-|\eta|}{\sigma})$. Then the matrix mechanism can be written as:

$$\mathcal{M}_A(W, x) = WA^+(Ax + Lap(\frac{\Delta_A}{\epsilon})^p) \tag{1}$$

where Δ_A denotes the *sensitivity* of a workload A, which is the maximum difference in the answers to A between neighboring databases that differ in a record.

The error of a DP mechanism $\mathcal{M}$ for a linear query w is commonly measured using mean squared error (MSE) per query [10], i.e., $\mathbb{E}((wx - \mathcal{M}(w, x)^2)$; or (α, β)-accuracy [5], i.e., the query error $|wx - \mathcal{M}(w, x)|$ is no more than α with a high probability $(1 - \beta)$.

Blowfish Privacy. Unlike DP that provides a blanket privacy guarantee for all values in the domain which can be too restrictive to offer sufficient utility, Blowfish privacy [6, 7] specifies pairs of domain values which the data curator wish to protect using a *policy graph.* A policy graph over $\mathcal{T}$ is a graph $G = (V, E)$, where $V = \mathcal{T}$ and $E \subseteq V \times V$. Based on a given policy graph, neighboring databases[1] and Blowfish privacy can be defined as follows.

Definition 2.1 (Blowfish Privacy)**.** Let $G = (V, E)$ be a policy graph. An algorithm $\mathcal{M}$ satisfies (ϵ, G)-Blowfish privacy if $\forall\, \mathcal{S} \in \text{Range}(\mathcal{M})$: $\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq e^\epsilon \Pr[\mathcal{M}(D') \in \mathcal{S}]$ for neighboring databases (D, D') that differ in the value of exactly one record, such that $(u, v) \in E$ where u is the record value in D and v is the corresponding record value in D'.

A broad class of policy graphs are known as θ-distance-threshold policy graphs [7], where the edges are defined as $E = \{(u, v) \mid d(u, v) \leq \theta\}$ for a given distance metric $d(\cdot, \cdot)$ (see Figure 1 for examples). When $\theta = 1$, it is a *line*-graph policy. More types of policy graphs can be found in [7].

Given an arbitrary graph G, rather than designing a (ϵ, G)-blowfish private mechanism from scratch, we can build such a mechanism from an ϵ-DP mechanism [6]. This requires the materialization of the policy graph as a matrix form using P_G of $(|V| - 1)$ rows and $|E|$ columns. Applying an optimal DP mechanism (e.g., matrix mechanism Eqn. (1)) to the transformed workload $W_G = WP_G$ on the transformed data vector $x_G = P_G^{-1}x$, will result in an optimal error for the given privacy policy graph G. Hence, the computation overhead directly relates to the graph size.

3 SYSTEM DESIGN

We design and build the first prototype, BlowfishDB, for policy-aware DP data exploration. This system ensures that (i) bounded and flexible privacy guarantees for data curators; and (ii) accurate and efficient query processing for data analysts. In this section, we first present the system overview and then introduce *dynamic Blowfish privacy*, a theoretical framework of Blowfish privacy that allows for better performance in the interactive setting.

System Overview. BlowfishDB is designed as a flexible software proxy to operate over any standard relational database system. Figure 2 shows the components of BlowfishDB and how it interacts with both the database servers and its users. It connects to the relational database through a SQL interface. Privacy policies are specified by the data curator and are controlled through a policy manager module. A privacy engine, which creates accuracy aware private queries, allows data analysts the ability to explore the sensitive data without requiring extensive privacy knowledge. Through

[1]This assumes known data size and no other database constraints. Variants of the definition can be found in [6].

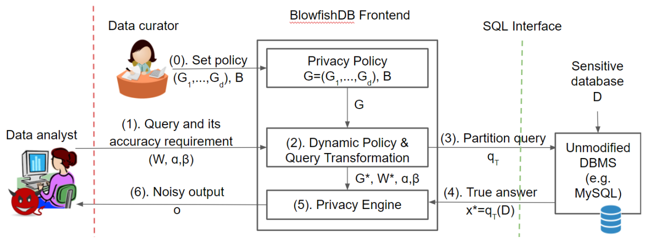
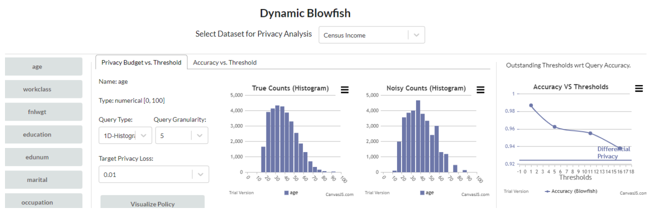


Figure 2: BlowfishDB overview. (0) set privacy policy G and budget B; (1-2) generate dynamic policy and query; (3-4) query over DBMS; (5-6) run DP mechanism that outputs o with (α, β)-accuracy guarantee.


Dynamic Blowfish

Select Dataset for Privacy Analysis: Census Income

age
workclass
fnlwgt
education
edunum
marital
occupation

Privacy Budget vs. Threshold | Accuracy vs. Threshold

Name: age

Type: numerical [0, 100]

Query Type: 1D-Histogr

Query Granularity: 5

Target Privacy Loss: 0.01

Visualize Policy

Outstanding Thresholds wrt Query Accuracy.

Figure 3: A snapshot of data owner interface that displays data schema, query template, true query answer plot, noisy answer plot, utility-privacy tradeoff plot, and policy visualization feature.

its interfaces, BlowfishDB provides its users with an complete private data exploration system.

Frontend Interfaces. When a data curator first saves sensitive data D, the privacy policy will default to DP, the strongest privacy level. The data curator can then adjust the privacy policies at the attribute level to a desired policy graph G and set the total privacy budget B for the data exploration. BlowfishDB will save this information to control the privacy loss when the data analyst queries the sensitive data. In addition, the data curator is allowed to query the databases and test the accuracy-privacy tradeoffs (Figure 3) at different privacy policies and privacy budgets, so that they can make informed decision on the policies based on the explorations.

A data analyst interface is provided, with a set of exploration query templates to choose from. They can select a query W with its accuracy requirement (α, β). BlowfishDB will generate a projected workload, data vector query, and privacy policy (W^*, x^*, G^*) based on W and G which allows more efficient processing than the static version (W, x, G). Then BlowfishDB will construct and send a SQL query $q_T(D)$ that corresponds to answers for the partition on the database x^*. The accuracy translation engine then determines an (ϵ, G^*)-blowfish private algorithm that runs on (W^*, x^*) with an (α, β)-accuracy guarantee. In the prototype, BlowfishDB also offers noisy plots of the query answers and provides a log of the previously executed queries for comparison.

Dynamic Blowfish Policy Projection. To optimize performance, BlowfishDB dynamically generates a policy graph based on the query workload W and the static policy graph G. For example, given a policy graph G on a high-dimensional domain $\mathcal{T} = dom(att_1) \times \cdots \times dom(att_d)$, if a workload W only conditions on a single attribute att_i, we can project the workload, the data, and the policy graph onto a partitioned domain based on att_i. Thus, when storing the policy graph, we only incur an additional $O(|dom(att_i)|^3)$ cost versus the $O(|T|^3)$ cost of a complete policy graph.

Definition 3.1. [Projected Privacy Policy] Let $\mathcal{T}^*$ be a partition over $\mathcal{T}$. Given a Blowfish privacy policy graph $G = (V, E)$ over the full domain $\mathcal{T}$, the projected privacy policy G based on $\mathcal{T}^*$ is defined as $G^* = (V^*, E^*)$, where the node set V^* is the partitioned domain $\mathcal{T}^*$, and the edge set E^* includes an edge (v_1^*, v_2^*) if exists an edge $(v_{i_1}, v_{i_2}) \in E$ such that $v_{i_1} \in v_1^*$ and $v_{i_2} \in v_2^*$.

We can represent a partition $\mathcal{T}^*$ by a partition matrix T, where $T[i, j] = 1$ if $v_j \in \mathcal{T}$ is a value in the ith bin of the partition $\mathcal{T}^*$. Then the projected data vector $x^* = Tx$ represents the counts for the bins in the partition, where x is the histogram of the full domain $\mathcal{T}$. We say a partition $\mathcal{T}^*$ *supports* a workload query W if each query in W can be expressed as a linear combination of the counts in Tx, in particular $WT^+Tx = Wx$. In practice, we can directly construct the projected workload $W^* = WT^+$ and the projected data vector $x^* = Tx$ based on the new partition without materializing any of W, x, T. Under Dynamic Blowfish, we are able to incur a smaller performance overhead of $O(|x^*|^3)$ when compared to $O(|\mathcal{T}|^3)$ under Blowfish, where $\frac{|x^*|^3}{|\mathcal{T}|^3} \leq 1$.

Example 2: Continuing from Example 1. A cumulative workload W at granularity of $100k$, i.e., {[1,100k), [1, 200k)}, can be supported by a partition {[1,100k) and [100k, 200k)}. We can directly construct the projected data vector x^* using 2 selection queries and materialize the corresponding projected workload $W^* = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$. The projected policy graph $G^*_{\theta=1k}$ for $G_{\theta=1k}$ will involve only 2 nodes as shown in Figure 1. It is not hard to verify that $Wx = WT^+Tx = W^*x^*$, where $W = \begin{pmatrix} 1\cdots1 & 0\cdots0 \\ 1\cdots1 & 1\cdots1 \end{pmatrix}$ and $T = \begin{pmatrix} 1\cdots1 & 0\cdots0 \\ 0\cdots0 & 1\cdots1 \end{pmatrix}$.

When the partition size is much smaller than the full domain size, the projected set of data vector, workload, and graph (x^*, W^*, G^*) will have a smaller representation than (x, W, G), and results in a much faster computation time. In addition, directly applying an optimal ϵ-DP matrix mechanism (e.g., Eqn. (1)) on (x^*, W^*, G^*) achieves (ϵ, G)-Blowfish privacy with the same accuracy guarantee as on (x, W, G) (See Theorem 3 in the full version[2]).

Dynamic Blowfish Policy Composition. To allow flexible specifications of privacy policies for data curators, BlowfishDB considers a class of Blowfish privacy policies that can be composed from a set of attribute-based privacy policies.

Definition 3.2 (Attribute Composability)**.** Given a policy $G(V, E)$, let $G_{att_1}(V_1, E_1), \ldots, G_{att_d}(V_d, E_d)$ denote the projected privacy policies from G onto the respective attribute partitions. We say G is *attribute composable* if $G(V, E) = G_c(V_c, E_c)$, where $V_c = V_1 \times \cdots \times V_d$ and E_c includes edge $(u, v) \in V_c \times V_c$ if $\sum_{j=1}^{d} dist_{G_{att_j}}(u.Aj, v.Aj) = 1$, and the distance function $dist_{G_{att_j}}(u.Aj, v.Aj)$ denotes the shortest distance between u and v in G_{att_j}.

An example for attribute composition is illustrated in Figure 4. Note this is not the only possible composition function. For this

[2]https://github.com/BlowfishDB/DemoPaper

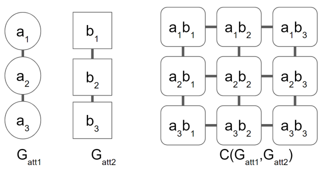


Figure 4: Compositing attribute-based policies G_{att_1} and G_{att_2} gives $C(G_{att_1}, G_{att_2})$.

class of attribute composable privacy policies, the data curator can customize the privacy policy of each attribute. For a given workload query W supported by the partition over a subset of attributes $T^* = \{att_{i_1}, \ldots, att_{i_j}\}$, we can project the workload and data over T^* and form the corresponding policy graph G_{T^*} by composing the relevant attribute policies $C(G_{att_{i_1}}, \ldots, G_{att_{i_j}})$. In an interactive setting, each query workload works with different dynamic privacy policies, but the total privacy loss can be bounded as follows.

Theorem 1. If G is attribute composable, let $\mathcal{M}_1, \ldots, \mathcal{M}_l$ be a sequence of algorithms, if each $\mathcal{M}_i$ satisfies $(\epsilon_i, G_{T_i^*})$-Blowfish privacy for a partition T_i^*, then the overall privacy loss is $(\sum_{i=1}^{l} \epsilon_i, G)$-Blowfish privacy.

Limitations and Future Work. The prototype of BlowfishDB provided for the demonstration only supports (i) 1-D or 2-D histogram queries and cumulative histogram queries and (ii) threshold policy graphs. The query templates supported by BlowfishDB also suggest a good partition choice. Besides expanding the scope of the queries and privacy policies, we will optimize the performance and accuracy over different choices of partitions and mechanisms.

4 DEMO EXPERIENCE

An attendee can choose from two demo scenarios for the BlowfishDB system, which are presented as tabs on the interface. The first takes the perspective of a data curator, who wishes to explore the accuracy vs privacy policy trade-off. They will be able to explore the affect of private mechanisms on query outputs, and will be able to explore the dataset so they can set an appropriate privacy policy. The second scenario takes the perspective of a data analyst, who wishes to explore the privacy vs utility trade-off. They will be able to run a variety of queries to see how a fixed privacy policy is able to affect the quality of their data exploration.

Demo Scenario 1: Data Curator Interface. We will provide a set of datasets in .csv format that attendees can explore. The attendee, acting as a data curator will select one of these datasets, prompting a menu listing the data set attributes to appear. The attendee will then select one of these attributes, as well as a query type. The true answers to the chosen query will be displayed as a bar chart as shown in Figure 3.

The attendee can then begin the trade-off exploration by choosing a privacy parameter ϵ. This will cause a noisy answer under ϵ-differential privacy to be calculated. The result will be plotted besides the true answer plot, so a direct comparison can be made.

The data curator can then choose to display the current privacy policy, to allow for visual exploration. By default, the policy is equivalent to DP so a fully connect graph over the chosen attribute is displayed. It is then possible to relax this policy by specifying a threshold value, which will update the display. The attendee can then select a set of possible threshold values, for which the total privacy loss will be calculated. These will be displayed as a line chart, where the attendee can then select a node representing a specific threshold value to display a noisy answer under that threshold policy as a histogram plot. By carrying out these steps, the attendee will be able to interactively explore the affect of privacy policies on the queries accuracy, and will be able to make an informed decision on how they would set an appropriate policy to protect their data.

Demo Scenario 2: Data Analyst Interface. Similarly to the previous demo, the attendee, acting as a data analyst, will select one of these datasets, prompting a menu listing the data set attributes to appear. The attendee will then select one of these attributes, as well as a query type and granularity of the query. The analyst will also be asked to specify a set of accuracy requirements, α and β, for their queries. A plot of the noisy answer to the chosen query will be displayed.

The analyst can then see their remaining privacy budget and chose to make another query if there is sufficient budget to do so. This process can be repeated using different attributes or query types. The analyst is restricted to 1-D histogram, 1-D cumulative histogram and 2-D histogram queries in the prototype. All prior query results are stored in a table that is visible to the data analyst. They are able to display the results of up to two previous noisy queries, so they are able to compare the results.

If a policy was specified for an attribute using the data curator demo scenario, then it will be applied in this demo. Otherwise, a default differential privacy policy will be used. An attendee can then run this scenario under different policies to be able to see the difference in privacy consumption, as well as the difference in the accuracy of the noisy answers.

REFERENCES

[1] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geo-indistinguishability: Differential privacy for location-based systems. In *SIGSAC*.
[2] Konstantinos Chatzikokolakis, Miguel E Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. 2013. Broadening the scope of differential privacy using metrics. In *PoPETS*.
[3] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *TCC*.
[4] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* (2014).
[5] Chang Ge, Xi He, Ihab F Ilyas, and Ashwin Machanavajjhala. 2019. APEx: Accuracy-Aware Differentially Private Data Exploration. In *SIGMOD*.
[6] Samuel Haney, Ashwin Machanavajjhala, and Bolin Ding. 2015. Design of policy-aware differentially private algorithms. *VLDB* (2015).
[7] Xi He, Ashwin Machanavajjhala, and Bolin Ding. 2014. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *SIGMOD*.
[8] Noah Johnson, Joseph P Near, and Dawn Song. 2018. Towards practical differential privacy for SQL queries. *VLDB* (2018).
[9] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2019. Privatesql: a differentially private sql query engine. *VLDB* (2019).
[10] 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* (2015).
[11] Ashwin Machanavajjhala, Aleksandra Korolova, and Atish Das Sarma. 2011. Personalized social recommendations-accurate or private? *VLDB* (2011).
[12] Frank D McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*.
[13] Prashanth Mohan, Abhradeep Thakurta, Elaine Shi, Dawn Song, and David Culler. 2012. GUPT: privacy preserving data analysis made easy. In *SIGMOD*.
[14] Zhuolun Xiang, Bolin Ding, Xi He, and Jingren Zhou. 2020. Linear and range counting under metric-based local differential privacy. In *ISIT*.