ATLANTIC: Making Database Differentially Private and Faster with Accuracy Guarantee

Lei Cao
MIT
Cambridge, MA
lcao@csail.mit.edu

Dongqing Xiao
Google
Cambridge, MA
xiaoguohaoke@gmail.com

Yizhou Yan
WPI
Worcester, MA
yyan2@wpi.edu

Samuel Madden
MIT
Cambridge, MA
madden@csail.mit.edu

Guoliang Li
Tsinghua University
Beijing, China
liguoliang@tsinghua.edu.cn

ABSTRACT

Differential privacy promises to enable data sharing and general data analytics while protecting individual privacy. Because the private data is often stored in the form of relational database that supports SQL queries, making SQL-based analytics differentially private is thus critical. However, the existing SQL-based differentially private systems either only focus on specific type of SQL queries such as COUNT or substantially modify the database engine, thus obstructing adoption in practice. Worse yet, these systems often do not guarantee the desired accuracy by the applications. In this demonstration, using the driving trace workload from Cambridge Mobile Telematics (CMT), we show that our ATLANTIC system, as a database middleware, enforces differential privacy for real-world SQL queries with provable accuracy guarantees and is compatible with existing databases. Moreover, using a sampling-based technique, ATLANTIC significantly speeds up the query execution, yet effectively amplifying the privacy guarantee.

PVLDB Reference Format:
Lei Cao, Dongqing Xiao, Yizhou Yan, Samuel Madden, and Guoliang Li. ATLANTIC: Making Database Differentially Private and Faster with Accuracy Guarantee. PVLDB, 14(12): 2755 - 2758, 2021.
doi:10.14778/3476311.3476337

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/yizhouyan/atlantic.

1 INTRODUCTION

As organizations increasingly collect sensitive information about individuals, these organizations are legally and ethically obligated to avoid privacy leaks. This sensitive information is often stored in relational database, allowing data analysts to easily conduct general-purpose data analytics through SQL queries. If the company does not want to disclose sensitive data to analysts, SQL-based analytics must be privacy-preserving.

Proceedings of the VLDB Endowment, Vol. 14, No. 12 ISSN 2150-8097.
doi:10.14778/3476311.3476337

Figure 1: A Middleware Between the User and Database

Differential privacy [5, 7] is a promising technique that admits general statistical analysis of data while protecting information about individuals with a strong guarantee of privacy. Because of its desirable formal guarantees, differential privacy has received growing attention from organizations including Google and Apple and from the database research community. However, differential privacy for general-purpose SQL queries remains an open challenge. First, many differential privacy techniques only focused on specific types of SQL queries, such as COUNT queries [9, 18], histograms [10, 17], equi-joins [8, 18], range queries, or special-purpose use cases such as data federation [3, 4]. Further, most of the existing systems [9, 13, 16] require substantial modification to the underlying database engine, complicating adoption in practice. Moreover, differential privacy protects the privacy of the data by injecting noise into the query results. Therefore, making SQL queries differentially private often produces imprecise query results with large error bounds that are not useful to some applications.

In this demonstration, we will show ATLANTIC, an approach that enables differential privacy for general purpose SQL-based analytics. ATLANTIC runs as a middleware layer on top of existing databases. It is compatible with real database systems, supports queries expressed in standard SQL, and thus easily integrates into existing data environments. Rather than introducing additional overhead to query execution, ATLANTIC runs queries even faster than the original databases by employing data sampling to enforce privacy guarantees. It uses a novel machine-learning method to provide the formal guarantees of differential privacy, while satisfying the user's requirements on the accuracy of query results.

As shown in Fig. 1, ATLANTIC sits between the user and the database. Given a user query, ATLANTIC uses query-rewriting to modify the query to collect the statistics required as input for the differential privacy algorithms. After query results come back, it runs the differential privacy algorithms on the results. In this way, the underlying database remains unchanged and any differential privacy mechanisms can be seamlessly plugged into ATLANTIC.

Moreover, we use a set of sample queries to learn a machine leaning model that is customized to the given data and query workload. It is thus more precise than the general purpose statistical methods in bounding the errors due to sampling and injected noise for protecting privacy. This model, that takes the sampling rate into consideration, provides ATLANTIC a way to automatically choose an *optimal sampling rate* for each category of queries, offering strong privacy and accuracy guarantees while minimizing the query execution time.

In this demonstration, we focus on the set of common aggregation operators that are known to be amenable to differential privacy: COUNT, SUM, AVG, and MEDIAN/QUANTILE [7]. ATLANTIC also supports typical database operations such as equi-joins and selection predicates underneath the aggregation operators.

2 PRELIMINARIES

We begin by recapping the key concepts of differential privacy.

Differential Privacy. Differential privacy provides a formal guarantee that a differentially private result does not disclose much information about which of two neighboring databases was used in calculating the result.

Formally, differential privacy considers a database modeled as a vector $x \in D^n$, in which x_i represents the data contributed by user i. The distance between two databases $x, y \in D^n$ is $d(x, y) = |\{i \mid x_i \neq y_i\}|$. Two databases x, y are neighbors if $d(x, y) = 1$.

Definition 2.1. Differential privacy. A randomized mechanism $\mathcal{M} : D^n \rightarrow \mathbb{R}^d$ preserves (ϵ, δ)-differential privacy if for any pair of databases x,$y \in D^n$ such that $d(x, y) = 1$, and for all sets S of possible outputs:

$$Pr[\mathcal{M}(x) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(y) \in S] + \delta \quad (1)$$

Intuitively, differential privacy ensures that for all adjacent database x, y, the absolute value of the privacy loss will be bounded by ϵ with probability at least 1 - δ. Therefore, the smaller the parameters ϵ and δ are, the stronger the privacy guarantee is.

Sensitivity. The sensitivity of a query corresponds to the amount its results will change when the database changes. One measure of sensitivity is global sensitivity, which is the maximum difference in the query's result on any two neighboring databases.

Definition 2.2. Global Sensitivity. For $f : D^n \rightarrow \mathbb{R}^d$ and all x, $y \in D^n$, the global sensitivity of f is

$$GS_f = \max_{x,y:d(x,y)=1} \|f(x) - f(y)\| \quad (2)$$

Another definition of sensitivity is local sensitivity [22,44], which is the maximum difference between the query's results on the true database and any neighbor of it. Local sensitivity is often much lower than global sensitivity, since it is a property of the single true database rather than all possible databases.

Definition 2.3. Local Sensitivity. For $f : D^n \rightarrow \mathbb{R}^d$ and $x \in D^n$, the local sensitivity of f at x is

$$LS_f(x) = \max_{y:d(x,y)=1} \|f(x) - f(y)\| \quad (3)$$

Differential Privacy Mechanism. By default, ATLANTIC uses the Laplace output perturbation mechanism [6] to enforce differential privacy, although other mechanisms can be equally applied.

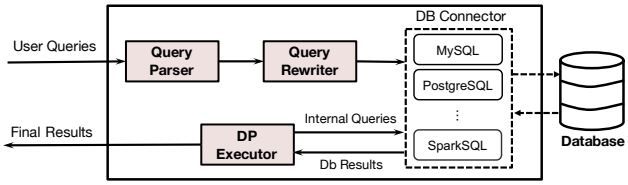

Figure 2: ATLANTIC Architecture

It constructs differentially private estimates of a function $f(x)$ by adding noise sampled from the Laplace distribution with magnitude proportional to the global sensitivity GS_f.

Definition 2.4. Laplace Distribution. The probability density function (PDF) of the *Laplace distribution* Lap(λ) is $h(z) = \frac{1}{2\lambda} e^{-|z|/\lambda}$. It has zero mean, and standard deviation $\sqrt{2}\lambda$.

Theorem 1. *$\forall f : D^n \rightarrow \mathbb{R}^d$, the database access mechanism $\mathcal{M}_f(x) = f(x) + (Z_1, \ldots, Z_d)$, where the Z_i are drawn i.i.d. from Lap(GS_f/ϵ), preserves ϵ-differential privacy [6].*

The original Laplace mechanism releases $f(x)$ with noise magnitude proportional to $GS_f(x)$. Because global sensitivity is often large, it tends to yield unacceptably high noise levels, hence low accuracy. If we release the result with noise magnitude proportional to $LS_f(x)$, the resulting algorithm would add significantly less noise for typical inputs. However, it does not satisfy the definition of differential privacy (Def. 2.1). In this work, we use smooth sensitivity [14] and its variation [8] to solve this problem.

3 ATLANTIC SYSTEM

In this paper we are focused on summarizing the functions and the main components of ATLANTIC. First, we briefly describe the main components of ATLANTIC in Sec. 3.1. We then explain how ATLANTIC interacts with users in Sec. 3.2. In Sec. 3.3 we discuss the types of SQL queries that benefit from ATLANTIC. Finally, we quantify the privacy amplification brought by sampling in Sec. 3.4.

3.1 The Components of ATLANTIC

Fig. 2 shows the main components of ATLANTIC including a *Query Parser*, a *Query Rewriter*, a *DB Connector*, and a *DP Executor*.

Given a user query, ATLANTIC uses an existing *Query Parser* to translate it into logical operators (e.g., projections, selections, joins, etc.). Then, *Query Rewriter* converts this logical expression into another logical expression, for example, automatically replace the original database table with the corresponding sample table, or to compute and retrieve the statistics needed by the differential privacy (DP) mechanism. *DB Connector* converts the rewritten logical expression into a SQL statement compatible with the underlying database. It then communicates with the database to get the query results.

Once the rewritten query is executed by the database, *DP Executor* uses SQL queries to collect the pre-computed statistics to compute *sensitivity*, runs DP mechanism to add noise to the query results, produces error estimates and confidence intervals, and returns the final results to users.

3.2 Workflow

Because ATLANTIC is a middleware as shown in Fig. 1, the user, either a data analyst or an app, does not directly interact with the database. Instead, they submit queries to and receive the results from ATLANTIC. The user interacts with ATLANTIC in two stages: offline preparation and online query processing.

During the **offline preparation** stage, the user decides on the tables and the attributes to be protected. In the meantime, the user has to specify the desired *privacy guarantee*, e.g. the parameters ϵ and δ in Def. 2.1 and the *accuracy requirement*, e.g., returning an answer that is at least 95% accurate. This information is then recorded in a specific schema inside the database catalog.

However, it can be hard even for the database and privacy experts to set an appropriate privacy guarantee and accuracy requirement, because they conflict with each other. If the user requests a very strong privacy guarantee by setting the parameter ϵ to a very small value, it will inevitably produce a noisy answer with large error bounds; and these error bounds vary across datasets and queries. Therefore, instead of requesting users to set these parameters at the online stage when submitting queries, ATLANTIC offers users a *tool* to setup these parameters at the offline stage. For each table, it first uses a set of sample queries to *profile* the relationship between the privacy guarantee and query accuracy w.r.t. different categories of queries. The queries in the same category could use different aggregate functions, but show the similar trend in the privacy and accuracy trade-off. ATLANTIC then plots the captured relationship into some privacy vs. accuracy charts, assisting the user to determine the parameters. For example, if the user desires a *hard* privacy guarantee, the charts will help them determine a realistic accuracy requirement, and vice versa. Note this process does not leak privacy, because it only uses the accuracy of the queries rather than the actual results.

The profiling also takes the *sampling rates* into consideration, because it plays a critical role in determining the amount of noise to be injected to the query results. One advantage of this sampling rate-aware profiling is that it naturally helps ATLANTIC determine the optimal sampling rate w.r.t. each category of queries and build the sample tables. The optimal sampling rate means the lowest one that promises the privacy and accuracy guarantee. Accordingly, for the same table ATLANTIC may construct multiple sample tables using different sampling rates and sampling methods such as random sampling or stratified sampling.

Moreover, using the accuracy data obtained via profiling, ATLANTIC trains a machine learning model that classifies each user query received at runtime to one query category and predicts its accuracy. Rather than using off-the-shelf statistical methods [12] to estimate the accuracy, we decide to employ machine learning, because the estimates produced by these statistical methods are known to be inaccurate in many cases [15].

At **runtime**, when a user issues a query, ATLANTIC identifies the set of sample tables to replace the corresponding base tables involved in the query. Because one single base table could correspond to multiple sample tables with different sampling rates, ATLANTIC uses the machine learning model trained during the offline stage to choose the optimal sampling rate and accordingly the sample table to use for query processing. ATLANTIC then rewrites the original query into another SQL statement and executes it with the underlying database. After the database returns the results, ATLANTIC computes the *sensitivity*. According to its value as well as the sampling rate and parameter ϵ, ATLANTIC adds noise to the result and estimates the accuracy using machine learning model, eventually returning the final result as well as the accuracy estimates.

3.3 Supported SQL Queries

ATLANTIC enables differential privacy on the analytic SQL queries that use common aggregate functions suitable for differential privacy [7]. In general, these aggregate functions correspond to queries with mean-like statistics, including COUNT, SUM, AVG, QUANTILE, VAR, and STDDEV.

ATLANTIC supports equi-join and selection predicates for differentially private aggregation. Because running the build-in join algorithms of databases on samples are known to be not effective in estimating the aggregates, we implement the popular sampling-based Wander join [11] in the middleware as a series of SQL queries.

ATLANTIC does not support aggregate functions that collect extreme statistics (i.e., min and max). For such aggregate functions, DP mechanism has to add large amount of noise to the query results [7] to protect the privacy, hence in practice often not satisfying the accuracy requirement. Currently ATLANTIC will reject these queries if they require privacy protection. For the queries that do not request protection, ATLANTIC will simply pass them down to the underlying database without any change.

3.4 Sampling and Differential Privacy

In this section, we first quantify the privacy amplification brought by sampling. We then introduce smooth sensitivity for minimizing the noise magnitude added to the query result and show how to efficiently compute it.

Privacy Amplification. We provide explicit privacy amplification bounds for the uniform sampling and stratified sampling, which are the most common sampling methods used in databases [1, 15]. The theoretical results apply to both single table and multi-table queries (equi-join). The results include the existing amplification bounds [2] in the literature as well as the novel bounds on join that we contribute.

Intuitively applying the differential privacy mechanism $\mathcal{M}$ to a random sample of the input database rather than on the database itself decreases the chances of leaking information about a particular individual, because nothing about that individual can be leaked in the cases where the individual is not included in the sample. More formally, if $\mathcal{M}$ is (ϵ, δ)-DP, then the sample mechanism $\mathcal{M}_S$ is (ϵ', δ')-DP for some $\epsilon' \leq \epsilon$ and $\delta' \leq \delta$. This is a privacy amplification, because the new mechanism has better privacy parameters than the original one. Eq. 4 quantifies this amplification.

$$e^{\epsilon'} = 1 + \eta(e^{\epsilon} - 1); \quad \delta' = \eta\delta \tag{4}$$

In Eq. 4, the parameter η represents the factor of privacy amplification. The larger the η is, the smaller the amplification is. Essentially, η corresponds to the sampling rate. In stratified sampling, because it uses different sampling rates for different subsets of the data, η corresponds to the largest one. For equi-join, if we

consider the sample-based join results as *one single table*, each distinct value is randomly sampled with certain rate from the complete join results on the original tables. Therefore, sample-based join can be treated as a special stratified sampling. Each distinct value has a unique sampling rate; and η corresponds to the largest one.

Smooth Sensitivity. As discussed in Sec. 2, releasing the query result with noise magnitude proportional to *global sensitivity* GS_f tends to yield unacceptably high noise levels. However, releasing the query result with noise magnitude proportional to *local sensitivity* LS_f does not satisfy the definition of differential privacy, as the noise magnitude itself reveals information about the database [14].

To solve this problem, ATLANTIC uses *smooth sensitivity* S_f which is much smaller than GS_f, while still satisfying Definition 2.1. The key idea is to make the noise magnitude insensitive to the real database by deriving a smooth upper bound on the local sensitivity.

$$\beta = \frac{\epsilon}{2\ln 2/\delta};\ \ S_f(x) = \max_{k=0,1,2,\ldots,n} e^{-\beta k} LS_f^k(x) \tag{5}$$

In Eq. 5, n represents the cardinality of x. $LS_f^k(x)$ is a generalization of the local sensitivity, representing how much the sensitivity can change when up to k entries of the database x are modified, called the sensitivity of f at distance k.

$$LS_f^k(x) = \max_{y \in D^n : d(x,y)=k} LS_f(y) \tag{6}$$

To support equi-join for aggregation, we use *elastic sensitivity* [8] which is a variation of smooth sensitivity.

4 DEMONSTRATION

ATLANTIC demo will feature an end-to-end implementation of the system. We will demonstrate ATLANTIC largely *speeds up* a range of ad-hoc exploratory queries with *provable privacy and accuracy guarantees*, using real-world data and analytics queries.

4.1 Setup Details

Our demo setup would consist of a Java open source implementation of ATLANTIC deployed to a Google GCP machine. In order to effectively demonstrate the system, our demo will feature an interactive JavaScript/HTML-based web console. Users can leverage this console to rapidly query across a range of parameters. To demonstrate a real-world analysis scenario, we will pre-store data in PostgreSQL and MySQL independently.

4.2 Demonstration Scenario

Our demo will use simulated data and workloads from an IoT company. It provides mobile applications to improve driver safety by measuring risky driving behavior and providing feedback to drivers. Because some of the metadata recorded about drivers, such as the times of day when they drive or the maximal speed they travel at, is sensitive, it's important to protect the identify of an individual driver from being discoverable by an analyst, while still making it possible to perform aggregate computations of driving behavior in, for example, particular locations or time periods. Specifically, we use a collection of aggregate queries on a synthetic database with a similar schema to that used by the company. Queries compute a variety of aggregate metrics, such as the total or average mileages and driving durations, count of the frequency of hard brakes or speeding by customer ID, city, or date, etc. These queries typically involve multiple filtering and equi-join operations. Differential privacy is used to ensure that the attributes of any particular individual cannot be discovered from the aggregate results of running the workload.

In this demonstration, we will show:

- How our ATLANTIC performs in offering the required privacy/accuracy guarantees and speeding up the query execution, in contrast to (1) FLEX [8] that enables differential privacy for COUNT query; (2) running differential privacy mechanism directly on database without sampling; (3) the state-of-the-art approximate query processing system VerdictDB [15] which runs queries on samples, but does not support differential privacy. We will show that ATLANTIC is (1) much faster and even more accurate than FLEX or running DP directly on database, while providing the same level privacy protection; (2) almost as fast as VerdictDB which always uses the same sampling rate to ATLANTIC.
- How ATLANTIC guides the users to quickly set the appropriate privacy and accuracy parameters.
- With its ML model, ATLANTIC is able to precisely predicate the accuracy of each query compared to the statistical methods.
- ATLANTIC supports different databases including PostgreSQL, MySQL, and SparkSQL.

REFERENCES

[1] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Eurosys*. 29–42.

[2] Borja Balle, Gilles Barthe, and Marco Gaboardi. 2018. Privacy Amplifica. by Subsampling: Tight Analyses via Couplings and Divergences. In *NeurIPS*. 6280–6290.

[3] Johes Bater, Xi He, William Ehrich, Ashwin Machanavajjhala, and Jennie Rogers. 2018. ShrinkWrap: Efficient SQL Query Processing in Differentially Private Data Federations. *Proc. VLDB Endow.* 12, 3 (2018), 307–320.

[4] Johes Bater, Yongjoo Park, Xi He, Xiao Wang, and Jennie Rogers. 2020. SAQE: Practical Privacy-Preserving Approximate Query Processing for Data Federations. *Proc. VLDB Endow.* 13, 11 (2020), 2691–2705.

[5] Cynthia Dwork and Jing Lei. 2009. Differential Privacy and Robust Statistics. In *STOC* (Bethesda, MD, USA). 371–380.

[6] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*. 265–284.

[7] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407.

[8] Noah M. Johnson, Joseph P. Near, and Dawn Song. 2018. Towards Practical Differential Privacy for SQL Queries. *Proc. VLDB Endow.* 11, 5 (2018), 526–539.

[9] 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 (2019), 1371–1384.

[10] Yu-Hsuan Kuo, Cho-Chun Chiu, Daniel Kifer, Michael Hay, and Ashwin Machanavajjhala. 2018. Differentially Private Hierarchical Count-of-Counts Histograms. *PVLDB* 11, 11 (2018), 1509–1521.

[11] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. 2016. Wander Join: Online Aggregation via Random Walks. In *SIGMOD*. 615–629.

[12] S. Lohr. 2009. Sampling: Design and Analysis, Seoncd Edition.

[13] Frank McSherry. 2010. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Commun. ACM* 53, 9 (2010), 89–97.

[14] Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *STOC*. 75–84.

[15] Yongjoo Park, Barzan Mozafari, Joseph Sorenson, and Junhao Wang. 2018. VerdictDB: Universalizing Approximate Query Processing. In *SIGMOD*. 1461–1476.

[16] Davide Proserpio, Sharon Goldberg, and Frank McSherry. 2014. Calibrating Data to Sensitivity in Private Data Analysis. *Proc. VLDB Endow.* 7, 8 (2014), 637–648.

[17] Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding Hierarchical Methods for Differentially Private Histograms. *Proc. VLDB Endow.* 6, 14 (2013), 1954–1965.

[18] Yuchao Tao, Xi He, Ashwin Machanavajjhala, and Sudeepa Roy. 2020. Computing Local Sensitivities of Counting Queries with Joins. In *SIGMOD*. 479–494.