Qrlew: Differentially Privacte SQL Query Rewriting

Anonymous submission

Abstract

AAAI creates proceedings, working notes, and technical reports directly from electronic source furnished by the authors. To ensure that all papers in the publication have a uniform appearance, authors must adhere to the following instructions.

Introduction

In recent years, the importance of safeguarding privacy when dealing with personal data has continuously increased. Traditional anonymization techniques have proven vulnerable to re-identification, as demonstrated by numerous works (Archie et al. 2018; Dwork et al. 2017; Narayanan and Shmatikov 2008; Sweeney, Abu, and Winn 2013). The total cost of data breaches has also significantly increased (IBM 2023) and governments have introduced stricter data protection laws. Yet, the collection, sharing, and utilization of data hold the potential to generate significant value across various industries, including healthcare, finance, transportation, and energy distribution.

To realize these benefits while managing privacy risks, researchers have turned to *differential privacy (DP)* (Wood et al. 2018; Dwork, Roth et al. 2014), which has become the gold standard in academia since its introduction by Dwork et al. in 2006 (Dwork et al. 2006) due to its provable and automatic privacy guarantees.

Despite the availability of open-source tools, DP adoption remains limited. One of the reasons for this lack of adoption is the relative complexity of the existing tools considered the utility of the results. *Qrlew* has been designed to solve these problems by providing the following features:

***Qrlew* provides automatic output privacy guarantees** With *Qrlew* a *data owner* can let an analyst (*data practitionner*) with no expertise in privacy protection run arbitrary SQL queries with strong privacy garantees on the output.

***Qrlew* leverages existing infrastructures** *Qrlew* rewrites a SQL query into a *differentially private* SQL query that can be run on any data-store with a SQL interface from lightweight DB to big-data stores. This removes the need for a custom execution engine and enables *differentially private analytics with virtually no technical integration*.

***Qrlew* leverages synthetic data** . Synthetic data are an increasingly popular way of *privatizing* a dataset. Using jointly *differentially private* mechanisms and *differentially private* synthetic data can be a simple, yet powerful, way of managing a privacy budget and reaching better utility-privacy tradeoffs.

Assumptions and Design Goals

In this work, we assume the *central model of differential privacy* (Near 2020), where a trusted central organization: Hospital, Insurance Company, Utility Provider, called the *data owner*, collects and stores personal data in a secured database. and whishes to let untrusted *data-practitionners* run SQL queries on its data. Furthermore, the *Qrlew* was designed to ease the

General architecture

In this work, we assume the *central model of differential privacy* (Near 2020), where a trusted central organization: Hospital, Insurance Company, Utility Provider, called the *data owner*, collects and stores personal data in a secured database. and whishes to let untrusted *data-practitionners* run SQL queries on its data. Furthermore, the *Qrlew* was designed to ease the

Paul on compilation

Rewriting in *Qrlew* refers to the process of altering parts of an SQL query by substituting them with different components to alter the privacy properties of the result. This substitution aims to achieve specific objectives, such as ensuring privacy through the incorporation of differential privacy mechanisms. To facilitate this work, we decompose the SQL query in the form of a computation graph where each node (a `Relation`) is the result of transformations representing part of the SQL queries.

The main goals of the differential privacy rewriting are to modify SQL queries to ensure compliance with differential privacy frameworks, protecting sensitive data, and to guarantee that these modifications are consistent and deterministic, adhering to established privacy standards.

The challenge lies in accurately identifying sections for modification across a complex array of potential transformations, vigilantly tracking the integrity of data rows tied to protected entities during extensive aggregations, and adeptly applying the correct rewriting rules tailored to the relation dynamics of the query, such as Joins, Maps, or Reduces, while considering the original configuration of the sensitive data.

Rewriting

Rewriting is a recursive, two-step process: For any given relation, we begin by rewriting the parents to alter their privacy properties, and then we modify the transforms from which the relation is derived, ensuring the desired privacy property based on the privacy properties of the parents. The recursion's base step involves deciding which tables to utilize—either the private tables or the synthetic ones.

To make the rewriting process more intelligible, we separate global privacy properties accounting for the computation graph and local rewriting rules concerning only the relation and the associated transforms to understand how the rewriting process could unfold recursively.

Global Privacy Properties Let's sum up the global privacy properties:

- **Protected Entity (PE)**: The definition of Differential Privacy (DP) is based on a notion of distance between datasets (see OpenDP work). Literature often formalizes datasets as multisets and uses the size of their symmetric difference as the distance between them (or equivalently, the L1 norm of the difference of their histograms). In our clients' datasets, an individual's data is often described by several rows of data. To adapt the distance to this requirement, Sarus labels each data row with an identifier: the Protected Entity ID (PEID) and defines the distance between datasets as the size of the symmetric difference of the set of PEIDs from each dataset.
- **Protected Entity Preserving (PEP)**: A dataset is PEP if each data row is associated with a single PE. A dataset transformation is said to be PEP if each row labeled with a PEID results from calculations that do not depend on rows labeled with another PEID.
- **Differential Privacy (DP)**: A result will be qualified as DP if it comes from a DP mechanism applied to a PEP relation. A DP relation can be published if the associated privacy loss has been accurately accounted for.
- **Synthetic (SD)**: A relation derived only from transformation from the synthetic table.
- **Public (Pub)**: A relation derived from public tables is labeled as such and does not require DP protections.
- **Published (Pubd)**: A relation is considered 'published' if the overall mechanism from the private table to the relation complies with the differential privacy framework, ensuring that the privacy of the private data is maintained provided the dataset is not inherently public. This is achieved when all parent relations of a given relation are either published, public, synthetic themselves, or use a differential privacy mechanism.

Rewriting Rules and Their Application

Rewriting rules are the main component to understand how to rewrite the computation graph. A rewriting rule is a local property explaining how to rewrite a relation to achieve global properties, being either DP, PEP, or synthetic, depending on some list of requirements on the privacy characteristics of the parents of the relation. In essence, a rewriting rule is a recursive step in the rewriting process. It explains how we could rewrite a relation given how we can rewrite the parents. The base case in the recursion is always: a table could either be private, public, or synthetic.

Relations can appear in various forms, including Join, Map, Reduce, Relation, Set, Table, and Values. Each type has its unique set of possible rewriting rules, outlining potential transformations and the associated global properties. An exhaustive list of the rewriting rules for each type of relation is provided in the annex.

Key Rewriting Rules Let's detail two crucial rewriting rules:

Relation with PEP $\rightarrow$ PEP and PEP $\rightarrow$ DP Rewriting Rules The first rule, PEP $\rightarrow$ PEP, concerns transferring the protected entity preserving property from one transformation stage to the next without mixing protected entities. The second rule, PEP $\rightarrow$ DP, means that if the parent of the relation can be rewritten as PEP, then we can rewrite the relation to be DP by applying a DP mechanism to publish the result. Both types of rewriting rules necessitate having parents that are capable of preserving protected entities. Some transformations, such as REDUCE, can aggregate data from multiple rows and thus can mix protected entities in the process. Such relations could be rewritten mostly only into published or synthetic, but it is possible that some REDUCE operations only aggregate rows related to the same protected entity and therefore could be rewritten into a PEP Relation.

Feasible Rewriting Rules A rewriting rule of a relation is said to be feasible if there exists a rewriting of the relation satisfying the rule. This is a global property of the relation, meaning that it depends on all the parents of the relation and not only on the transforms of the relation. Thus, the rewriting rule PEP $\rightarrow$ DP could be feasible for a relation if there exists a rewriting where the parent of the relation is in practice PEP. Then, we can change the transform and apply a DP mechanism to publish the relation. All rewriting rules about synthetic data are feasible because it is always possible in Sarus to build a synthetic variant of the relation using the synthetic table. The main goal of the DP rewriting is then to list for the relation that we want to obtain all the feasible rewriting rules and to select one where the outcome is DP. If not, it would mean that there does not exist a way to obtain this value using DP mechanisms, and we would need to use synthetic data.

In the computation graph, while each node's multiple rewriting rules might suggest a combinatorial explosion in the number of possible paths, this is mitigated in practice. The pruning of infeasible rules, dictated by the requirement for most relations to have a PEP input for a DP or PEP outcome, significantly reduces the complexity. Hence, despite the theoretical breadth of possibilities, the actual number of feasible paths remains manageable, avoiding substantial computational problems.

The challenge lies in choosing the appropriate rewriting rule for each segment of the SQL query. Though the potential solutions are numerous, practical circumstances significantly narrow down the viable options.

Implementation of the Rewriting Algorithm

The goal is to have a deterministic algorithm that rewrites SQL queries into a privacy-preserving form. During the *Qrlew* rewriting process, the algorithm systematically goes through the computation graph, applying the following steps:

1. Transform the SQL query into a computation graph composed of relations.
2. Tagging rewriting rules: Initially tag all relations with their rewriting rules depending on the type of relation and the parameters of the relation from the list in the annex.
3. Filtering Out Infeasible Rules: Identify and remove recursively any rewriting rules that are not feasible, ensuring that only viable transformations are considered.
4. Applying Rewriting Rules: Starting from the last final node, for each node with multiple feasible rewriting rules, we select the best one based on a scoring system. The simple score system encodes the preferences among different privacy properties: we prefer to have a DP result than a synthetic one, that is why we give a better score for a feasible rule resulting in a DP result than a synthetic one.
5. Building the Final Query: Once all relations have been transformed, convert the computation graph back into an SQL query that adheres to differential privacy standards. The budget is split among the different DP mechanisms involved in the computation graph.

Victoria on DP mech and DP test

Privacy algorithms

At this stage, we examine a `Relation` that takes one or two protected `Relation` inputs, where the entity to be protected is specified. To secure the outcome of the SQL query, it is imperative to protect not only the results obtained from aggregation functions but also the grouping keys. We will briefly describe these two steps in the following section.

Protecting aggregation results The protection of aggregation functions is carried out in three sequential steps. Given that all currently supported aggregations (`COUNT`, `SUM`, `AVG`, `VARIANCE STDDEV`) can be expressed as compositions of sums, our focus will be on the `SUM` aggregation. Let's consider the scenario where we aim to compute the sum of a column.

1. **Limit the contribution per user**: For each user, we calculate the ℓ_2 norm of the observations associated with that user. Subsequently, we constrain the contribution of a specific user by scaling $\mathbf{x}$ with a factor chosen to maintain the original observations if the ℓ_2 norm of the user's observations does not exceed the clipping factor c. Conversely, if the ℓ_2 norm surpasses c, the ℓ_2 norm of the rescaled observations is restricted to c. More technical details can be found in the appendix .
2. **Add random noise**: The sum of the original data is substituted with the sum of the scaled data with the addition of Gaussian noise. The level of noise is parameterized by the clipping and privacy parameters.
3. **Restrict differentially private aggregation**: The final operation involves confining the differentially private aggregation within the bounds automatically computed for the `SUM`($\mathbf{x}$).

Protecting grouping keys As explained by Wilson et al., making grouping keys public could result in privacy breaches. The treatment of grouping keys varies depending on whether the analyst knows them.

- **Public Grouping Keys.**We categorize grouping keys as public when the analyst specifies them in the `WHERE` clause. In this scenario, all user-specified keys must be disclosed, even if they do not exist in the database. If a key is absent in the database, differentially private results will include these missing keys, and the corresponding aggregations will be random noise.
- **Revealing Private Keys through τ-Thresholding** As introduced by Wilson et al., tau-thresholding involves releasing keys whose differentially private noise surpasses a threshold determined by privacy parameters.

When the SQL query involves both public and private grouping keys, we perform cross joins on the grouping keys obtained through the two algorithms.

Comparison to other systems

Known limitations

Qrlew relies on the random number generator of the SQL engine used. It is usually not a cryptographic noise.

Qrlew uses the floating-point numbers of the host SQL engine, therefore our system is liable to the vulnerabilities described in

Useful links

PPAI

Last year papers: https://aaai-ppai23.github.io/#sp2 This year program: https://ppai-workshop.github.io/

Comparable open-source projects

- Paszke et al. 2017 - Automatic differentiation in PyTorch https://openreview.net/pdf?id=BJJsrmfCZ
- Frostig et al. 2018 - Compiling machine learning programs via high-level tracing https://mlsys.org/Conferences/2019/doc/2018/146.pdf

Comparable DP SQL papers

- Lessons Learned: Surveying the Practicality of Differential Privacy in the Industry (Garrido et al. 2022)
- Tumult Analytics: a robust, easy-to-use, scalable, and expressive framework for differential privacy (Berghel et al. 2022)
- Differentially Private SQL with Bounded User Contribution (Wilson et al. 2019)

- CHORUS: a Programming Framework for Building Scalable Differential Privacy Mechanisms (Johnson et al. 2020)
- Towards Practical Differential Privacy for SQL Queries (Johnson, Near, and Song 2018)

Appendix: Limit the contribution per user within the aggregations

We consider the **x** columns that contains n observations belonging to p users:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_{n_1} \\ x_{n_1+1} \\ \vdots \\ x_{n_2} \\ \vdots \\ x_{n_{p-1}+1} \\ \vdots \\ x_n \end{pmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{User 1} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{User 2} \\ \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{User } p \end{matrix}$$

For each user k, we compute the scale factor that constrains their to a maximum value of c:

$$s_k = \frac{1}{\max(1, \frac{1}{c}\sqrt{\sum_{i=n_{k-1}+1}^{n_k} x_i^2})}. \quad (1)$$

The original **x** vector is then rescaled by multiplying each coordinate with the corresponding user's scale factor. The resulting rescaled vector has coordinates given by $s_{u_i} x_i$, where u_i represents the user ID associated with observation i. This process ensures that the ℓ_2 contribution of each user is restricted to c within the rescaled vector.

In our algorithm, the clipping value c is given by:

$$c = \max(|\min \mathbf{x}|, |\max \mathbf{x}|), \quad (2)$$

where $\min \mathbf{x}$ and $\max \mathbf{x}$ are the automatically computed bounds of **x**.

Thank you for reading these instructions carefully. We look forward to receiving your electronic files!

References

Archie, M.; Gershon, S.; Katcoff, A.; and Zeng, A. 2018. Who's watching? de-anonymization of netflix reviews using amazon reviews.

Berghel, S.; Bohannon, P.; Desfontaines, D.; Estes, C.; Haney, S.; Hartman, L.; Hay, M.; Machanavajjhala, A.; Magerlein, T.; Miklau, G.; et al. 2022. Tumult Analytics: a robust, easy-to-use, scalable, and expressive framework for differential privacy. *arXiv preprint arXiv:2212.04133*.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, 265–284. Springer.

Dwork, C.; Roth, A.; et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4): 211–407.

Dwork, C.; Smith, A.; Steinke, T.; and Ullman, J. 2017. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application*, 4: 61–84.

Garrido, G. M.; Liu, X.; Matthes, F.; and Song, D. 2022. Lessons learned: Surveying the practicality of differential privacy in the industry. *arXiv preprint arXiv:2211.03898*.

IBM. 2023. Cost of a Data Breach Report 2023.

Johnson, N.; Near, J. P.; Hellerstein, J. M.; and Song, D. 2020. Chorus: a programming framework for building scalable differential privacy mechanisms. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, 535–551. IEEE.

Johnson, N.; Near, J. P.; and Song, D. 2018. Towards Practical Differential Privacy for SQL Queries. *Proc. VLDB Endow.*, 11(5): 526–539.

Narayanan, A.; and Shmatikov, V. 2008. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, 111–125. IEEE.

Near, J. 2020. Threat Models for Differential Privacy.

Sweeney, L.; Abu, A.; and Winn, J. 2013. Identifying participants in the personal genome project by name (a re-identification experiment). *arXiv preprint arXiv:1304.7605*.

Wilson, R. J.; Zhang, C. Y.; Lam, W.; Desfontaines, D.; Simmons-Marengo, D.; and Gipson, B. 2019. Differentially private SQL with bounded user contribution. *arXiv preprint arXiv:1909.01917*.

Wood, A.; Altman, M.; Bembenek, A.; Bun, M.; Gaboardi, M.; Honaker, J.; Nissim, K.; O'Brien, D. R.; Steinke, T.; and Vadhan, S. 2018. Differential privacy: A primer for a non-technical audience. *Vand. J. Ent. & Tech. L.*, 21: 209.

Annex

Description of the Rewriting Rules for Each Type of Relation

Map Function: Transforming a Single Relation

- Pub $\to$ Pub: A relation could stay public if it starts off public.
- Pubd $\to$ Pubd: A relation could stay published if it starts off published.
- PEP $\to$ DP (with specific budget): A relation could become differentially private, adopting a specific budget, if it starts off preserving protected entities.
- SD $\to$ SD: A relation could remain as synthetic data if it starts off as synthetic data.
- SD $\to$ Pubd (with specific synthetic data parameters): A relation could transform into published if it starts off as synthetic data, depending on specific synthetic data parameters.

Reduce Function: Transforming a Single Relation

- Pub $\rightarrow$ Pub: A relation could stay public if its parent is public.
- Pubd $\rightarrow$ Pubd: A relation could stay published if its parent is published.
- PEP $\rightarrow$ DP (with specific budget): A relation could become differentially private, adopting a specific budget, if it is preserving protected entities.
- SD $\rightarrow$ SD: A relation could remain as synthetic data if its parent is synthetic data.
- SD $\rightarrow$ Pubd (with specific synthetic data parameters): A relation could transform into published if it is synthetic data, depending on specific synthetic data parameters.

Join Function: Combining Two Relations

- Pub + Pub $\rightarrow$ Pub: Combining two public relations could result in a public relation.
- Pubd + Pubd $\rightarrow$ Pubd: Combining two published relations could result in a published relation.
- Pubd + PEP $\rightarrow$ PEP (with specific parameters): Combining a published relation and a relation that preserves protected entities could result in a relation that preserves protected entities, adopting specific parameters.
- PEP + PEP $\rightarrow$ PEP (with specific parameters): Combining two relations that preserve protected entities could result in a relation that preserves protected entities, adopting specific parameters.
- DP + PEP $\rightarrow$ PEP (with specific parameters): Combining a differentially private relation and a relation that preserves protected entities could result in a relation that preserves protected entities, adopting specific parameters.
- SD + SD $\rightarrow$ SD: Combining two synthetic data relations could result in a synthetic data relation.

Set Function: Merging Two Relations

- Pub + Pub $\rightarrow$ Pub: Combining two public relations could result in a public relation.
- Pubd + Pubd $\rightarrow$ Pubd: Combining two published relations could result in a published relation.
- PEP + PEP $\rightarrow$ PEP (with specific parameters): Combining two relations that preserve protected entities could result in a relation that preserves protected entities, adopting specific parameters.
- SD + SD $\rightarrow$ SD (with specific parameters): Combining two synthetic data relations could result in a synthetic data relation, adopting specific parameters.

Values Function: Creating a New Relation

- $\rightarrow$ SD (with specific parameters): A new relation could be created as synthetic data, depending on specific parameters.
- $\rightarrow$ Pub: A new relation could be created as public.