MSBD5014B Answering SQL Queries Under Differential Privacy Report

Name: Zixin MA
ID: 21001887
GitHub: https://github.com/ZixinMa27/MSBD5014B_DP

May 24, 2024

1. Introduction

The primary objective of this project is to apply and evaluate the concepts and algorithms presented in the research paper titled "R2T: Instance-optimal Truncation for Differentially Private Query Evaluation with Foreign Keys" [2]. The R2T mechanism is specifically designed to handle arbitrary SPJA (Selection-Projection-Join-Aggregation) queries in databases with foreign-key constraints and self-joins. To assess its effectiveness, we have selected a range of query types from the TPC-H benchmark and implemented the R2T algorithm on these queries. Through conducting experiments, we aim to compare the relative error between the actual result query and the result query after applying R2T. This evaluation will provide insights into the utility and efficacy of the R2T mechanism in preserving privacy while accurately answering queries in the presence of foreign-key constraints.

2. Background of the Project

Differential privacy. Differential privacy (DP)[1] is a rigorous privacy framework that ensures the protection of individual data while allowing valuable analysis and querying of datasets. It guarantees that the results of queries on a dataset remain statistically indistinguishable even when an individual's data is included or excluded. By applying DP techniques, we can strike a balance between data utility and privacy, making it suitable for various domains, including SQL-based systems. DP has gained significant attention in recent years due to its ability to provide privacy guarantees in data analysis tasks.
TPC-H Benchmark. TPC-H [4] is a widely used decision support benchmark that consists of a set of complex queries designed to evaluate the performance of database systems. In this project, we use it use a benchmark to evaluate the efficiency of R2T algorithms in handling complex queries with foreign-key constraints and self-joins.
Laplace Noise. Laplace noise is used in differential privacy to add random noise to query

results, ensuring privacy while allowing data analysis. It is based on the Laplace distribution, with the amount of noise controlled by the privacy parameter ϵ. Balancing privacy and utility, smaller ϵ values provide stronger privacy but more noise, while larger ϵ values offer more accurate results at the expense of some privacy.
CPLEX. CPLEX (ILOG CPLEX Optimization Studio) is a widely used optimization software package. In this project, CPLEX is used to solve optimization problems that arise when applying the R2T mechanism. The R2T algorithm involves determining an optimal truncation threshold to achieve the desired utility guarantees while preserving privacy.

3. Related Work

The challenge of answering queries under differential privacy (DP) in relational databases with foreign-key constraints and self-joins has been a topic of significant research. One notable contribution in this area is the Race-to-the-Top (R2T) mechanism proposed by Dong et al. [2]. The R2T mechanism addresses the challenge of answering queries under differential privacy (DP) in relational databases with foreign-key constraints and self-joins. It combines truncation and differential privacy to achieve optimal utility guarantees. The R2T mechanism extends truncation to handle self-joins, providing a comprehensive solution for both challenges. It has demonstrated superior utility guarantees compared to existing techniques and offers a practical solution for differentially private query evaluation in real-world databases. The R2T mechanism is efficient and easy to implement, making it a promising approach for preserving privacy in complex relational structures.

4. Description of the Query and the Algorithm

In this study, I have selected five queries and categorized them into three types based on the nature of their primary private relations and aggregation operations.

4.1 Query Type: Single Primary Private Relation

Q12: This query counts the number of line items, designating "Order" as the primary private relation. The original query is:

```
SELECT COUNT(*)
FROM orders, lineitem
WHERE o_orderkey = l_orderkey.
```

The rewritten version is:

```
SELECT o_orderkey
FROM orders, lineitem
WHERE o_orderkey = l_orderkey.
```

Q3: This query counts the number of line items where the associated orders were made by customers before January 1, 1997, and the corresponding line items were shipped after January 1, 1994. "Customer" is designated as the primary private relation. The original query is:

```
SELECT COUNT(*)
FROM customer, orders, lineitem
WHERE orders.O_CUSTKEY = customer.C_CUSTKEY
AND lineitem.L_ORDERKEY = orders.O_ORDERKEY
AND o_orderdate < DATE '1997-01-01'
AND l_shipdate > DATE '1994-01-01'.
```

The rewritten version is:

```
SELECT c_custkey
FROM customer, orders, lineitem
WHERE orders.O_CUSTKEY = customer.C_CUSTKEY
AND lineitem.L_ORDERKEY = orders.O_ORDERKEY
AND o_orderdate < DATE '1997-01-01'
AND l_shipdate > DATE '1994-01-01'.
```

4.2 Query Type: Multiple Primary Private Relation

Q5: This query counts the number of line items where the customer and supplier are from the same nation and region. "Customer" and "Supplier" are designated as the primary private relations. The original query is:

```
SELECT COUNT(*)
FROM supplier, lineitem, orders, customer, nation, region
WHERE supplier.S_SUPPKEY = lineitem.L_SUPPKEY
AND lineitem.L_ORDERKEY = orders.O_ORDERKEY
AND orders.O_CUSTKEY = customer.C_CUSTKEY
AND customer.C_NATIONKEY = nation.N_NATIONKEY
AND nation.N_NATIONKEY = supplier.S_NATIONKEY
AND region.R_REGIONKEY = nation.N_REGIONKEY.
```

The rewritten version is:

```
SELECT C_CUSTKEY, S_SUPPKEY
FROM customer, orders, lineitem, supplier, nation, region
WHERE c_custkey = o_custkey
AND l_orderkey = o_orderkey
AND l_suppkey = s_suppkey
AND c_nationkey = s_nationkey
AND s_nationkey = n_nationkey
AND n_regionkey = r_regionkey.
```

4.3 Query Type: Aggregation

Q18: This query returns the total quantity of line items purchased by all customers. "Customer" is designated as the primary private relation. The original query is:

```
SELECT SUM(l_quantity)
FROM customer, orders, lineitem
WHERE c_custkey = o_custkey
AND o_orderkey = l_orderkey.
```

The rewritten version is:

```
SELECT c_custkey, l_quantity
FROM customer, orders, lineitem
WHERE c_custkey = o_custkey
AND l_orderkey = o_orderkey.
```

Q11: This query calculates the sum of the product of supply cost and available quantity (scaled) from the partsupplier table, based on supplier and nation keys. "Supplier" is designated as the primary private relation. The original query is:

```
SELECT SUM(ps_supplycost * ps_availqty / 1000000)
FROM nation, supplier, partsupp
WHERE ps_suppkey = s_suppkey
AND s_nationkey = n_nationkey.
```

The rewritten version is:

```
SELECT s_suppkey, ps_supplycost * ps_availqty / 1000000
FROM nation, supplier, partsupp
WHERE ps_suppkey = s_suppkey
AND s_nationkey = n_nationkey.
```

4.4 R2T Algorithm

$$\tilde{Q}(\mathbf{I},\tau^{(j)}) := Q(\mathbf{I},\tau^{(j)}) + Lap\left(\log(GS_Q)\frac{\tau^{(j)}}{\varepsilon}\right) - \log(GS_Q)\ln\left(\frac{\log(GS_Q)}{\beta}\right)\cdot\frac{\tau^{(j)}}{\varepsilon},$$

for $\tau^{(j)} = 2^j, j = 1, \ldots., \log(GS_Q)$. Then R2T outputs

$$\tilde{Q}(\mathbf{I}) := \max\left\{\max_j \tilde{Q}(\mathbf{I},\tau^{(j)}), Q(\mathbf{I},0)\right\}.$$

R2T algorithm is in the research paper[2]. First, the algorithm initializes key parameters, including the global sensitivity (GS_Q), the privacy budget (ϵ), and the confidence parameter (β). It then defines the truncation thresholds as $\tau^{(j)} = 2^j$ for $j = 1, \ldots, \log(GS_Q)$. Next, for each truncation threshold $\tau^{(j)}$, the algorithm computes the truncated query result $Q(I,\tau^{(j)})$.

Laplace noise, scaled by $\frac{\log(GS_Q)\tau^{(j)}}{\epsilon}$, is added to this result. Additionally, a correction term, $\log(GS_Q)\ln\left(\frac{\log(GS_Q)}{\beta}\right)\cdot\frac{\tau^{(j)}}{\epsilon}$, is subtracted to adjust for truncation effects. Then, the adjusted query result $\tilde{Q}(I,\tau^{(j)})$ is calculated using the formula:

$$\tilde{Q}(I,\tau^{(j)}) := Q(I,\tau^{(j)}) + \text{Lap}\left(\frac{\log(GS_Q)\tau^{(j)}}{\epsilon}\right) - \log(GS_Q)\ln\left(\frac{\log(GS_Q)}{\beta}\right)\cdot\frac{\tau^{(j)}}{\epsilon}$$

Finally, the algorithm selects the final differentially private query result $\tilde{Q}(I)$ as the maximum of the adjusted results $\tilde{Q}(I,\tau^{(j)})$ for all j, and the untruncated result $Q(I,0)$:

$$\tilde{Q}(I) := \max\left\{\max_j \tilde{Q}(I,\tau^{(j)}), Q(I,0)\right\}$$

4.5 New LP

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

In the research paper[2], the author defined a new LP. The objective is to maximize the function $Q(I,\tau)$, which represents the sum of the variables v_l. The constraints ensure that each v_l is bounded by the sum of u_k over the set $D_l(I)$. Additionally, the sum of u_k over the set $C_j(I)$ must be less than or equal to the truncation threshold τ. The variables u_k and v_l are further bounded by their respective functions $\psi(q_k(I))$ and $\psi(p_l(I))$. This LP problem formulation ensures that the query results are optimized while maintaining the constraints necessary for differential privacy.

5. Implementation details

The implementation process involved several key steps to ensure the effective application of the R2T mechanism for differential privacy while maintaining query accuracy and performance.

Data Retrieval. The first step was to execute SQL queries to fetch data from a PostgreSQL database [5]. This involved connecting to the database, running the queries, and retrieving the results as a list of tuples.

LP Problem Construction. The next step was to construct the Linear Programming (LP) problem based on the types of queries. This included setting up the objective function, defining the variables, and establishing the constraints needed for the LP formulation. In this project, we focused on counting for single primary private relation and multiple primary relation queries, as well as aggregation summing.

Optimal Solution with CPLEX. To solve the LP problem, we utilized the CPLEX solver [7]. This step involved configuring the solver, adding variables and constraints, and solving the LP to obtain the objective value.

R2T Mechanism for Differential Privacy. The final step was to implement and apply the R2T Mechanism as described in the "Description of Algorithm" section to ensure differential privacy. This step involved adding Laplace noise to the query results to protect sensitive information. We recorded both the real query output and the differentially private query output after applying the R2T mechanism. Additionally, we calculated the solve time and relative error for each query to evaluate the performance and accuracy of the differential privacy mechanism.

6. Experiments

Datasets. The experiments were conducted using the TPC-H benchmark, with datasets generated [6] at various scales: 0.125, 0.25, 0.5, and 1. These varying scales provide a comprehensive analysis of the algorithm's performance across different dataset sizes. The TPC-H benchmark comprises eight relations: Region (RK), Nation (RK, NK), Customer (NK, CK), Orders (CK, OK), Supplier (NK, SK), Part (PK), PartSupp (SK, PK), and Lineitem (SK, PK, OK, LN). The datasets were converted to CSV format and stored in the /data folder. For the default dataset used in the experiments, generated at scale 0.125, the record counts were as follows:

- Region: 5 records
- Nation: 25 records
- Customer: 150,000 records
- Orders: 1,500,000 records
- Supplier: 10,000 records
- Part: 200,000 records
- PartSupp: 800,000 records
- Lineitem: 6,000,000 records

The global sensitivity was set to 10^5.

Database. An empty PostgreSQL database named "tpch" was created, into which the generated datasets were imported. This process involved setting up the database schema and bulk-loading the CSV files into the corresponding tables, ensuring that the data was correctly structured and indexed for efficient querying.

Queries. The R2T algorithm was executed on three types of queries as described in the "Description of the Query" section. These queries were designed to test different aspects of the algorithm, including single primary private relation queries, multiple primary relation queries, and aggregation queries. This comprehensive set of queries ensured a thorough evaluation of the algorithm's performance and privacy guarantees.

Experimental Environment. All experiments were conducted on an Apple M1 Pro machine, providing a modern and efficient environment for the computations. Each experiment was repeated 100 times to ensure the results were statistically significant. Due to the random noise introduced by the differential privacy mechanism, the errors exhibited some variability. To address this, the best 20 and worst 20 runs were removed, and the average error was calculated based on the remaining 60 runs. This approach helped in obtaining a more accurate measure of the algorithm's performance. The default values used for the experiments were a beta value of 0.1 and an epsilon value of 0.8, which are standard parameters for balancing privacy and accuracy in differential privacy mechanisms.

7. Experimental results and discussion

The results of the experiments are summarized in Table 7.1. The relative error for all queries is less than 1%, which indicates that the R2T mechanism is highly effective in maintaining the accuracy of the results while ensuring differential privacy. In terms of running time, the R2T algorithm performed efficiently. The running times observed in our experiments were consistently less than those reported in the paper "R2T: Instance-optimal Truncation for Differentially Private Query Evaluation with Foreign Keys" [2]. This discrepancy is primarily due to the different data sizes used in the experiments. While the referenced paper conducted experiments with a data scale of 1, our experiments were conducted with a data scale of 0.125. This reduction in data size naturally leads to faster processing times. Moreover, although the relative results obtained in our experiments are similar to those reported in the referenced paper, there are slight differences. These variations can be attributed to differences in data scale and the global sensitivity values used. In our experiments, the global sensitivity was set to 10^5, which differ from the settings in the original study 10^6. These differences highlight the importance of context-specific parameter tuning in differential privacy implementations.

Query Type	**Single Primary Private Relation**		**Multiple Primary Private Relation**	**Aggregation**	
Query Name	Q3	Q12	Q5	Q11	Q18
Real Query Result	362146	750594	29756	250269.788	19170865.0
R2T Relative Error (%)	0.2240	0.112	0.750	0.6657	0.0348
Time (s)	6.5821	13.6838	1.1504	3.3286	29.4196

Table 7.1: Experimental results summarizing real query results, relative errors, and computation times.

8. Future directions

The paper [2] introduces a projection type of query, which I did not include in this project. However, incorporating this query type is a potential avenue for future work. Furthermore, I attempted to test the scalability of the R2T algorithm by generating data at different scales. Unfortunately, due to limitations with the computer's GPU, I was unable to achieve my intended goals in this regard.

Additionally, another innovative method of differential privacy called shifted-inverse, presented in the paper "Shifted Inverse: A General Mechanism for Monotonic Functions under User Differential Privacy" [3], could be explored in the future. This mechanism introduces a novel differential privacy approach specifically designed for handling monotonic functions under user-level differential privacy (user-DP). Investigating and implementing the shifted-inverse algorithm and comparing its performance with R2T would provide valuable insights and contribute to the advancement of differential privacy techniques.

9. Acknowledgments

I appreciate the guidance and support provided by Professor Yike and TA Yu Jianzhe throughout this project. I would like to acknowledge the use of ChatGPT for refining and correcting the English grammar in this report.

References

[1] Aaron Roth and Cynthia Dwork. *The Algorithmic Foundations of Differential Privacy.* Available at: `https://www.cis.upenn.edu/~aaroth/Papers/privacybook.pdf`

[2] Yike He, Ke Yi, and Graham Cormode. *R2T: Instance-optimal Truncation for Differentially Private Query Evaluation with Foreign Keys.* Available at: `https://cse.hkust.edu.hk/~yike/R2T.pdf`

[3] Yike He, Ke Yi, and Graham Cormode. *Shifted Inverse: A General Mechanism for Monotonic Functions under User Differential Privacy.* Available at: `https://cse.hkust.edu.hk/~yike/ShiftedInverse.pdf`

[4] *TPC Benchmark.* Available at: `https://www.tpc.org/tpc_documents_current_versions/current_specifications5.asp`

[5] *Dajianshi Blog on Python Set up PostgreSQL.* Available at: `https://www.cnblogs.com/dajianshi/archive/2012/06/06/2827093.html`

[6] *Gist on Generating TPCH Dataset.* Available at: `https://gist.github.com/yunpengn/6220ffc1b69cee5c861d93754e759d08`

[7] *CSDN Blog on Python Cplex Tutorial*. Available at: https://blog.csdn.net/kobeyu652453/article/details/118784161