Query Evaluation under Differential Privacy

by

Wei Dong

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in the Department of Computer Science and Engineering

June 2023, Hong Kong

Authorization

Wei Dong

June 2023

Query Evaluation under Differential Privacy

by

Wei Dong

This is to certify that I have examined the above PhD thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

Prof. Ke Yi, Thesis Supervisor

Prof. Xiaofang Zhou, Head of Department

Department of Computer Science and Engineering
June 2023

Acknowledgments

I am deeply grateful to those who have supported me throughout my Ph.D. journey. Their guidance, encouragement, and help have been invaluable in enabling me to complete this thesis.

First and foremost, I extend my heartfelt thanks to my supervisor, Prof. Ke Yi. As a mentor, he has been exceptional. His guidance during my early years as a Ph.D. student has been crucial in shaping my research skills, from identifying potential problems to designing solutions and writing academic papers. I am grateful for his unwavering patience in correcting my mistakes and for his support, which has enabled me to grow and develop rapidly. As a pure researcher, his hard work and enthusiasm for research have been truly inspiring, setting an excellent example for me to follow. As a friend, he has offered invaluable suggestions and warm-hearted encouragement whenever I faced challenges. Without any exaggeration, I consider being supervised by Prof. Yi as one of the greatest blessings in my life.

I also extend my sincere appreciation to all committee members of my Ph.D. qualification exam, thesis proposal defense, and thesis defense, Prof. Dimitris Papadias, Prof. Dimitris Papadopoulos, Prof. Jiheng Zhang, Prof. Hongyu Yu, Prof. Xiaokui Xiao, Prof. Raymond Wong, and Prof. Qiong Luo. Their insightful comments and suggestions have been invaluable in shaping this thesis.

I am also grateful to my colleagues in my research group, including Dr. Xiao Hu, Dr. Bin Wu, Dr. Yu Chen, Dr. Yilei Wang, Dr. Yuan Qiu, Dr. Qichen Wang, Dr. Ziyue Huang, Haoqian Zhang, Qiyao Luo, Juanru Fang, Yuting Liang, Dajun Sun, Bingnan Chen, Zijun Chen, and Bingyang Dai. Their camaraderie, intellectual discussions, and willingness to lend a helping hand have made my Ph.D. journey more enriching.

Moreover, I want to thank my friends at HKUST, especially Xingbo Wang, Hongwei Liu, Zhuoyi Peng, Zhenhua Xu, Ruijie Ma, Zhiliang Tian, Zezheng Feng, Guanlan Zhang, Haoyang Li, Maocheng Li, Shizhe Diao, Yishuo Zhang, Keyou Chen, and Wuxian He, for their support and companionship. Their presence in my life has made each challenge more bearable and has added color to my Ph.D. journey.

Finally, I must express my gratitude to my parents and family members. Their unwavering support and encouragement have been a source of strength throughout my Ph.D. journey. I am deeply grateful for their love and care.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Query Evaluation under Differential Privacy

by Wei Dong

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

Abstract

Differential privacy (DP) has garnered significant attention from both academia and industry due to its potential in offering robust privacy protection for individual data during analysis. With the increasing volume of sensitive information being collected by organizations and analyzed through SQL queries, the development of a general-purpose query engine that is capable of supporting a broad range of SQLs while maintaining differential privacy has become the holy grail in privacy-preserving query release. Over the past several years, the database and security communities have made numerous efforts to achieve this objective. However, two major challenges still exist:

Challenge 1: Privacy guarantee in relational databases Informally speaking, DP requires indistinguishability of the query results whether any particular individual's data is included or not in the database. While this can be easily defined and well studied over a single flat table, the situation becomes more complex in a relational database with the presence of multiple relations, foreign keys, and the join operator.

Challenge 2: Optimal privacy-utility trade-off Noise injection is inherently necessary for privacy protection, but the central scientific question is how to achieve the lowest error (i.e., highest utility) under a given privacy budget. Unfortunately, for the problem of relational query evaluation under DP, traditional notions of optimality, i.e., instance optimality and worst-case optimality, are either unattainable or meaningless. Thus, a new notion of optimality is needed for answering this question.

In this thesis, we aim at tackling these challenges. To model the complex situation of relational databases, we study two different DP policies: tuple-DP, which protects the privacy of single tuples in each relation, and user-DP, which protects all data belonging to each user via foreign keys. Under each policy, we have designed DP mechanisms for answering a broad class of queries consisting of the selection, projection, aggregation, and join operators. To formalize their optimality, we introduce the notion of neighborhood optimality, which sits between instance optimality and worst-case optimality. Finally, based on the algorithms and theory developed, we have built a DP-SQL system that significantly outperforms existing systems in terms of both utility and efficiency.

CHAPTER 1

INTRODUCTION

Suppose a data analyst is interested in the total number of items sold this year where the customer and supplier are from the same nation in a given region, s/he would issue the following query (assuming the TPC-H schema):

```
SELECT count(*)
FROM Region, Nation, Customer, Orders, Supplier, Lineitem
WHERE Orders.Orderdate > 2023-01-01
  AND Region.Name = '[REGION]'
  AND Region.RK = Nation.RK AND Lineitem.SK = Supplier.SK
  AND Nation.NK = Customer.NK AND Customer.CK = Orders.CK
  AND Orders.OK = Lineitem.OK AND Nation.NK = Supplier.NK;
```

This type of queries can involve 4 basic relational operators: Selection, Projection, Join and Aggregation (like COUNT, SUM) and is called select-project-join-aggregation (SPJA) queries. In this thesis, we mainly discuss count and sum aggregations. Such queries are very common in today's data analytical tasks and are a central problem in databases that have been extensively studied in the literature. Sophisticated query processing algorithms and systems have been and are continually being developed and optimized throughout the years.

However, many datasets contain private information, and privacy concerns have become the main hurdle to making use of today's big data for knowledge discovery and decision making. In this example, while it might be safe to reveal `Region` and `Nation`, the other four relations contain private relationship information between various entities, such as which customer has placed a particular order, which items are contained in an order, which suppliers provide a certain item, so their privacy must be protected. Meanwhile, *Differential privacy (DP)*, already deployed by Apple [25], Google [39], Microsoft [26], and the US Census Bureau [61], has become the standard notion for private data release, due to its strong protection of individual information. It requires that any individual cannot largely change the query result thus his information can be masked. Formally speaking, for a query Q, let $Q(\mathbf{I})$ be the result of evaluating Q on database instance $\mathbf{I}$ and a mechanism $\mathcal{M}_Q : \mathcal{I} \to \mathcal{Y}$ is (ε, δ)-DP if

$$\mathsf{Pr}[\mathcal{M}_Q(\mathbf{I}) \in Y] \leq e^{\varepsilon} \cdot \mathsf{Pr}[\mathcal{M}_Q(\mathbf{I}') \in Y] + \delta \tag{1.1}$$

for any subset of output $Y \subseteq \mathcal{Y}$ and any pair of neighboring instances $\mathbf{I}, \mathbf{I}'$, i.e., $d(\mathbf{I}, \mathbf{I}') = 1$. Here, $d(\mathbf{I}, \mathbf{I}')$ denotes the minimum number of individual information needed to be changed to turn $\mathbf{I}$ into $\mathbf{I}'$. ε, δ are called privacy budget. Typically, ε is a constant ranging from 0.1 to 10, with smaller values corresponding to stronger privacy guarantees. On the other hand, δ should be much smaller than $1/N$ to ensure the privacy of individual tuples, where $N = |\mathbf{I}|$ is the instance size; in particular, the case where $\delta = 0$ is referred to as *pure differential privacy*, which is more desirable if achieved.

In the last 10 years, we have seen many efforts put into answering SQL queries under DP. In this problem, *tuple-DP* and *user-DP* are two popular policies. As the name suggests, the former aims at protecting tuples while the latter is for users, which can have multiple tuples. They are like protecting the privacy of edges and nodes in computing graph statistics, which is a special case of SQL queries. These two policies have different privacy-utility trade-offs. User-DP provides stronger privacy while tuple-DP usually yields a higher utility. Choosing the right policy depends on the privacy requirement of the application.

1.1 Tuple-DP

In the early stages of addressing SPJA queries, the predominant method was the tuple-DP approach, which was exclusively focused on queries involving count aggregation. On the other hand, sum aggregation, where each tuple can have an unbounded impact on the query result, was typically addressed using the user-DP technique. Unless otherwise specified, discussions surrounding tuple-DP and user-DP will refer to count aggregation and sum aggregation, respectively. For query answering under tupe-DP, a dominating approach is the following *sensitivity* framework.

1. Compute the query answer on the given database instance.

2. Compute some notions of sensitivity of the query $S_Q(\mathbf{I})$, which measure the difference between the query answers on two database instances that differ by one tuple (neighboring database instances).

3. Release a noise-masked query answer, where the noise is drawn from some zero-mean distribution, calibrated appropriately according to the sensitivity.

Step (1) and (3) are both well understood in the literature, so the key challenge is step (2). There are two main desiderata when designing a sensitivity measure. On one hand, the sensitivity should be as small as possible, as the noise level is proportional. More precisely, if we use the expected ℓ_2-error

$$\mathrm{Err}(\mathcal{M}_Q, \mathbf{I}) = \sqrt{\mathsf{E}\left[(\mathcal{M}_Q(\mathbf{I}) - |Q(\mathbf{I})|)^2\right]},$$

then various noise distributions (Section 3.1.3 gives more details) have been studied all achieve $\mathrm{Err}(\mathcal{M}_Q, \mathbf{I}) = \Theta(S_Q(\mathbf{I}))$. [1] Adding noise with a magnitude greater than the actual query answer would completely destroy its utility. On the other hand, the sensitivity should be computed efficiently.

Below, we review previous measures of sensitivity for answering SPJA queries under tuple-DP along the two desiderata, before presenting our proposal.

1.1.1 Previous Work

Global sensitivity The *global sensitivity* GS_Q is defined as the maximum difference between the query answers on *any* two neighboring database instances. Adding Laplace noise proportional to GS_Q preserves ε-DP and this mechanism is usually called the Laplace mechanism. Note that global sensitivity is a property of the query only, and does not depend on the actual instance. Equivalently speaking, global sensitivity considers the worst-case instance, and measures the amount of change in the query answer when a tuple is changed in that worst-case instance. It is worst-case optimal and works well for counting queries over a single relation, where the global sensitivity is just 1. However, the global sensitivity can be as large as N^{n-1} when (unrestricted) joins are present, where n is the number of relations involved in Q.

Local and smooth sensitivity When the global sensitivity is high or unbounded, it is tempting to use the sensitivity of the query on the particular given instance, which is

[1]Throughout the thesis, we adopt the convention of *data complexity* [2], i.e., all asymptotic notations suppress dependencies on the query size. We also take ε to be a constant, as with most work on differential privacy.

usually much lower, except on contrived instances. This is referred to as the *local sensitivity* $\mathrm{LS}_Q(\mathbf{I})$. However, as pointed out by Nissim et al. [67], using the local sensitivity to calibrate noise is not differentially private. This is because the local sensitivity can be very different on two neighboring databases, so the noise level may reveal information about an individual tuple. Essentially, the problem is that local sensitivity, when considered as a query, has high global sensitivity. To get around the problem, the idea is to use a smooth (i.e., having low global sensitivity) upper bound of the local sensitivity. Clearly, the smaller this upper bound is, the better utility we would obtain, and the tightest smooth upper bound is termed the *smooth sensitivity* $\mathrm{SS}_Q(\mathbf{I})$ [67] (see Section 3.1.3.1 for more precise definitions).

Although smooth sensitivity has met the first desideratum, the second one is less clear. In fact, it is shown that for certain problems, computing or even approximating the smooth sensitivity is NP-hard [67]. For multi-way join counting queries, so far, no known polynomial algorithm exists.

Elastic sensitivity Due to the lack of an efficient algorithm for computing the smooth sensitivity for SPJA queries under tuple-DP, Johnson et al. [48] proposed *elastic sensitivity*, which is also a smooth upper bound of local sensitivity, but not as tight as smooth sensitivity. Elastic sensitivity achieves the second desiderata: it can be computed in linear time, and can be implemented easily using a constant number of SQL queries, plus a user-defined function (UDF) that combines the answers of these queries using a certain formula. However, it does not really achieve the first desiderata. Theoretically, the gap between elastic sensitivity and smooth sensitivity can be as large as $O\left(N^{n-1}\right)$. In practice, it often yields noise levels that are orders-of-magnitude higher than the actual query answer, as demonstrated in the experiments of [48], as well as our own experiments.

Overall, there is still no known solution to this problem that can have a good tradeoff between efficiency and utility. Then, what is a good tradeoff? For efficiency, we hope the algorithm has a small computational complexity. But how do we quantify the utility of a DP mechanism? As mentioned before, the worst-case optimality loses its utility. Ideally, we would like the solution to be *instance-optimality* [40]. More formally, a DP mechanism $\mathcal{M}_Q(\cdot)$ is said to be c-instance-optimal if

$$\mathrm{Err}(\mathcal{M}_Q, \mathbf{I}) \leq c \cdot \mathrm{Err}(\mathcal{M}'_Q, \mathbf{I}),$$

for every instance $\mathbf{I}$ and every DP mechanism $\mathcal{M}'_Q(\cdot)$. Unfortunately, as pointed out by Asi and Duchi [10], instance-optimal DP mechanisms do not exist, unless the query is trivial (i.e., it returns the same count on all instances), because for an $\mathbf{I}$, one can always design a trivial DP mechanism $\mathcal{M}'_Q(\cdot) \equiv |Q(\mathbf{I})|$. It works perfectly on $\mathbf{I}$ but fails miserably on other instances. Yet, such a trivial $\mathcal{M}'_Q$ rules out instance-optimal mechanisms.

1.1.2 Neighborhood Optimality

To eliminate such a trivial $\mathcal{M}'_Q$, Asi and Duchi [10] propose the following natural relaxation of instance-optimality, which requires $\mathcal{M}'_Q$ to not just work well for $\mathbf{I}$, but also in its neighborhood.

Definition 1.1.1 ((r,c)-neighborhood optimal DP mechanisms). *An ε-DP mechanism $\mathcal{M}_Q(\cdot)$ is said to be (r,c)-neighborhood optimal if for any instance $\mathbf{I}$ and any ε-DP mechanism $\mathcal{M}'_Q(\cdot)$, there exists an instance $\mathbf{I}'$ with $d(\mathbf{I},\mathbf{I}') \le r$ such that*

$$\mathrm{Err}(\mathcal{M}_Q,\mathbf{I}) \le c \cdot \mathrm{Err}(\mathcal{M}'_Q,\mathbf{I}'). \tag{1.2}$$

Note that neighborhood optimality smoothly interpolates between instance optimality ($r=0$) and worst-case optimality ($r=N$). It is also called *local minimax optimality* in [10], as it minimizes (up to a factor of c) the maximum error in the local neighborhood of $\mathbf{I}$. As we are most interested in constant-factor approximations, we often use "r-neighborhood optimal" as a shorthand of "$(r,O(1))$-neighborhood optimal".

While more relaxed than instance optimality, neighborhood optimality is still not easy to achieve (for small r). For example, we can show that, with the expected ℓ_2 error metric, the Median query (i.e., returning the median of N elements in $[0,1]$, for odd N) does not have any (r,c)-neighborhood optimal mechanisms for $r \le \lfloor N/2 \rfloor$ and any c. To see this, consider $\mathbf{I} = (0,\dots,0)$ and $\mathcal{M}'_Q(\cdot) \equiv 0$. Note that all instances in the r-neighborhood of $\mathbf{I}$ have output 0, so $\mathrm{Err}(\mathcal{M}'_Q,\mathbf{I}') = 0$ for all $\mathbf{I}'$ with $d(\mathbf{I},\mathbf{I}') \le r$. Thus, we must set $\mathcal{M}_Q(\mathbf{I}) = 0$ to satisfy (1.2). We can apply the same argument on $\mathbf{I}'' = (1,\dots,1)$ and conclude that $\mathcal{M}_Q(\mathbf{I}'') = 1$. We have thus found two instances on which $\mathcal{M}(\cdot)$ returns different deterministic values, thus $\mathcal{M}$ cannot be DP by a standard argument [37].

Acute readers would realize that the negative result for Median relies on the fact that there are certain instances (like $(0,\dots,0)$) with a "flat" neighborhood, i.e., the query output is the same within the neighborhood. Fortunately, for SJA queries (without Projection), these flat neighborhoods do not exist. This is because in any instance $\mathbf{I}$, one can always add/remove a constant number (depending on Q) of tuples to change $Q(\mathbf{I})$. However, this is merely a necessary condition for a query to admit neighborhood-optimal DP mechanisms. To actually design such a mechanism $\mathcal{M}_Q(\cdot)$, we need an upper bound on $\mathrm{Err}(\mathcal{M}_Q, \mathbf{I})$, as well as a neighborhood lower bound on $\min_{\mathcal{M}'_Q} \max_{\mathbf{I}':d(\mathbf{I},\mathbf{I}')\leq r} \mathrm{Err}(\mathcal{M}'_Q, \mathbf{I}')$. Note that both the upper bound and the lower bound are instance specific (i.e., functions of $\mathbf{I}$), and the worst-case (over all $\mathbf{I}$) gap between the upper and lower bounds would become the optimality ratio c.

1.1.3 Our Proposal: Residual Sensitivity [29, 31]

1.1.3.1 JA Queries

To answer SPJA queries under tuple-DP, we propose *residual sensitivity* $\mathrm{RS}(\mathbf{I})$. Let's first consider JA queries and defer the discussion of selection and projection. Here, we relate both the upper and the lower bounds to the smooth sensitivity $\mathrm{SS}(\cdot)$. On the lower bound side, in Section 3.3.2, we show that $\mathrm{SS}(\cdot)$ is an $O(1)$-neighborhood lower bound. Thus, the smooth sensitivity based mechanism [67] is $O(1)$-neighborhood optimal. However, $\mathrm{SS}(\cdot)$ in general requires exponential time to compute. In Section 3.2, we define the $\mathrm{RS}(\cdot)$ and show it is a constant-factor upper bound of $\mathrm{SS}(\cdot)$ in Section 3.3.3. Besides, residual sensitivity can be computed in polynomial time, where the exponent depends only on the join structure, not the database. In practice, the running time appears to be linear, and is only slower than that for computing elastic sensitivity by a small constant factor. Together with the lower bound, this yields our first main result:

Theorem 1.1.1. *For any JA query Q under tuple-DP, there is an ε-DP mechanism $\mathcal{M}_Q(\cdot)$ that is $O(1)$-neighborhood optimal. For any instance $\mathbf{I}$, $\mathcal{M}_Q(\mathbf{I})$ can be computed in* $\mathrm{poly}(N)$ *time.*

Then, we extend our solution to support selection and projection operations.

1.1.3.2 SJA Queries

The selection operation involves some *predicates* $P_1(\mathbf{y}_1), \ldots, P_\kappa(\mathbf{y}_\kappa)$, each of which is a computable function $P : \mathbf{dom}(\mathbf{y}) \rightarrow \{\mathsf{True}, \mathsf{False}\}$ for a set of attributes $\mathbf{y}$. Predicates are important for expressing graph pattern counting queries. Suppose we would like to count the number of length-3 paths (no repeated vertices) in a directed graph whose edges are stored in a relation Edge. However, the JA query $|\mathsf{Edge}(x_1, x_2) \bowtie \mathsf{Edge}(x_2, x_3) \bowtie \mathsf{Edge}(x_3, x_4)|$ would not just count the number of length-3 paths, but also length-1 paths, length-2 paths, and triangles. We have to equip the query with inequalities, i.e., $x_i \neq x_j$ for all $i \neq j$, to exclude these false positives. Another type of predicates is comparison, i.e., $x_i \leq x_j$ or $x_i < x_j$, which are common in queries over spatiotemporal databases.

In Section 3.4 we show how to modify $\mathrm{RS}(\cdot)$ to take these predicates into consideration, while preserving its neighborhood optimality. The idea is conceptually easy: we just materialize each predicate $P(\mathbf{y})$ into a relation $\{t \in \mathbf{dom}(\mathbf{y}) \mid P(t)\}$, and then apply our DP mechanism in Theorem 1.1.1. However, this poses a computational issue, since this relation can be infinite (assuming infinite domains). To address this issue, we show that it is actually possible to compute $\mathrm{RS}(\cdot)$ without materializing the $P(\mathbf{y})$'s.

Theorem 1.1.2. *For any SJA query Q where all predicates are inequalities or comparisons, there is an $O(1)$-neighborhood optimal ε-DP mechanism $\mathcal{M}_Q(\cdot)$ and $\mathcal{M}_Q(\mathbf{I})$ can be computed in* $\mathrm{poly}(N)$ *time.*

1.1.3.3 SPJA Queries

To complete the picture, we finally study the SPJA queries in Section 3.5. Prior work simply ignores the projection, and uses the sensitivity of the query without projection to calibrate noise. We show how to extend $\mathrm{RS}(\cdot)$ to handle the projection more effectively so as to reduce the noise. Unfortunately, our lower bound no longer holds when projection is involved, hence losing neighborhood optimality. However, we show that this is unavoidable. In particular, even for the simple query $|\pi_{x_2}(R_1(x_1) \bowtie R_2(x_1, x_2))|$, we show that it does not admit any $o(\sqrt{N})$-neighborhood optimal DP mechanisms with the expected ℓ_2 error metric.

1.1.3.4 Integration to the relational database system

Similar to elastic sensitivity, residual sensitivity can also be computed by executing a constant number of SQL queries, and then combining their answers using a UDF, although the queries are slightly more complicated than those for elastic sensitivity. We have built a system prototype based on PostgreSQL that can answer any SPJA query under tuple-DP.

1.2 User-DP

Now, let's move towards the user-DP. Still with TPC-H schema, if a data analyst is interested in the total number of orders this year, s/he would release the following query,

```
SELECT count(*) FROM Orders WHERE Orders.Orderdate > 2023-01-01;
```

Tuple-DP protects the privacy of each order and requires only constant noise. However, in practice, we may hope to protect the privacy of users, i.e., customers in this case, who can own multiple orders. The issue above was first identified by Kotsogiannis et al. [57], who also formalized the *DP policy for relational databases with FK constraints*. The essence of their model (a rigorous definition is given in Section 6.1.2) is that the individuals and their private data are stored in separate relations that are linked by FKs. Adding/deleting one tuple will add/delete all tuples referencing it. The above query can be rewritten as

$$Q := |\mathsf{Customer}(\underline{\mathsf{CK}}) \bowtie \mathsf{Orders}(\mathsf{CK}, \mathsf{OD})|.$$

Here, the underlined attribute $\underline{\mathsf{CK}}$ is the primary key (PK), while $\mathsf{Orders.CK}$ is a foreign key (FK) referencing $\mathsf{Customer.CK}$. Our target is to protect the tuples in $\mathsf{Customer}$.

Without loss of generality, we assume there is only one relation storing individuals since if there are multiple ones, we can create a new relation to store all individuals and build FK constraints between it and the others. Therefore, the self-join query may refer the cases with multiple individual relations

1.2.1 Down-neighborhood Optimality

FK constraint is the most crucial feature of the relational model, yet it brings new challenges compared with tuple-DP. First, any sensitivity-based framework loses its utility. Give the above query as an example. What's the GS_Q for this query? It is, unfortunately, ∞. This is because a customer, theoretically, could have an unbounded number of orders, and adding such a customer to the database can cause an unbounded change in the query result. A simple fix is to assume a finite GS_Q, which can be justified in practice because we may never have a customer with, say, more than a million orders. However, as mentioned in Section 1.2.2, assuming such a GS_Q limits the allowable database instances, one tends to be conservative and sets a large GS_Q. Then, how about $\mathrm{LS}_Q(\mathbf{I})$ for some specific database instance $\mathbf{I}$. Unfortunately, $\mathrm{LS}_Q(\cdot) \equiv \mathrm{GS}_Q$ since for any $\mathbf{I}$, we can always add a customer with GS_Q number of orders to get one neighboring instance $\mathbf{I}'$. That will also lead to neighborhood optimality losing its utility. Recall Definition 1.1.1, we compare the $\mathrm{Err}(\mathcal{M}_Q, \mathbf{I})$ with the maximum between $\mathrm{Err}(\mathcal{M}'_Q, \mathbf{I})$ and $\mathrm{Err}(\mathcal{M}'_Q, \mathbf{I}')$. Since no mechanism can perform well in both $\mathbf{I}$ and $\mathbf{I}'$, that is a too low requirement for $\mathbf{I}$.

To address the issue, we first restrict the neighborhood by only considering *down-neighbors*, which can be obtained only by removing individuals. The new notation is called *down-neighborhood optimality*. Clearly, the smaller the neighborhood, the stronger the optimality notion and down-neighborhood optimality does not consider those "bad neighbors" formed by adding some heavy contributors. Similarly, the *downward sensitivity* $\mathrm{DS}_Q(\mathbf{I})$ is defined as how much the query answer changes if we delete one individual's information. Equivalently speaking, that is the largest contribution of any user in $\mathbf{I}$ to Q. $\mathrm{DS}_Q(\cdot)$ can be shown to be 1-down-neighborhood optimal lower bound thus under user-DP, our target is to design a DP mechanism with $O(\mathrm{DS}_Q(\cdot))$ error.

1.2.2 SA Queries

Let's first discuss SA queries. In fact, this type of queries is equivalent to the 1-dimensional mean (sum) estimation problem, which is important for many machine learning tasks. Therefore, this problem attracts lots of attention from both database community [57, 76] and statistics and machine learning community [1, 6, 7, 69, 62, 46].

The dominating solution is the *truncation mechanism*, which simply deletes all individuals with more than τ contributions to the query result before adding the noise, for some threshold τ. After truncation, the query has sensitivity τ, so adding a noise of scale τ is sufficient. A well-known issue for the truncation mechanism is the bias-variance trade-off: In one extreme $\tau = \mathrm{GS}_Q$, it degenerates into global sensitivity mechanism and has a large variance; in the other extreme $\tau = 0$, the truncation introduces a bias as large as the query answer.

The issue of how to choose a near-optimal τ under DP has been extensively studied [6, 7, 69, 62, 46]. However, all of them require a predefined GS_Q and their error bounds also depend on GS_Q even though the dependencies may be in the log factors.

1.2.2.1 Our Contribution [30]

We propose the first solution to remove that assumption for answering SA queries under user-DP. Besides, when the GS_Q is given, the state-of-art error bound [46] is $O(\mathrm{DS}_Q(\mathbf{I}) \log^{1.5}(\mathrm{GS}_Q))$.[2] We show this error can be improved.

Theorem 1.2.1. *For any SA query Q under user-DP, there is an ε-DP mechanism $\mathcal{M}_Q(\cdot)$ such that for any $\mathbf{I}$, it runs in linear time and returns a $\widetilde{Q}(\mathbf{I})$ such that $\left|\widetilde{Q}(\mathbf{I}) - \widetilde{Q}(\mathbf{I}')\right| = O\left(\mathrm{DS}_Q(\mathbf{I}) \log\log(\mathrm{DS}_Q(\mathbf{I}))\right)$.*

Thus, we obtain an exponential improvement even in the finite-domain case. Furthermore, we show that the optimality ratio cannot be better than $O(\log\log T)$ for all $\mathbf{I}$ with $\mathrm{DS}_Q(\mathbf{I}) = T$.

1.2.3 SPJA Queries

Then, let's move towards the SPJA queries. In this problem, all prior works focus on self-join-free SJA queries, which can be transformed to a SA queries (see Section 5.2 for a more formal statement). However, self-joins introduce another challenge. In particular, all techniques to answer SA queries [6, 7, 69, 62, 46] for choosing a truncation threshold τ critically rely on the fact that the individuals are independent, i.e., adding/removing one individual does not affect the data associated with another, which is not true when the query involves self-joins. In fact, when there are self-joins, even the truncation mechanism itself fails, as illustrated in the example below.

Example 1.2.1. Suppose we extend the query given at the beginning of Section 1.2 to the following one with a self-join:

$$Q := |\mathsf{Customer}(\underline{\mathsf{CK}}) \bowtie \mathsf{Supplier}(\underline{\mathsf{SK}}) \bowtie \mathsf{Orders}(\underline{\mathsf{OK}}, \mathsf{CK}) \bowtie \mathsf{Lineitems}(\mathsf{OK}, \mathsf{SK})|.$$

This query counts the total number of lineitems, each of which corresponds to one customer and supplier. This is like edge counting in a bipartite graph where the customers are on the left while the suppliers are on the right.

Let $\mathbf{I}$ be an τ-regular graph (i.e., every vertex has degree τ) with $N/2$ left vertices and right vertices. Let $\mathbf{I}'$ be the neighboring instance, where we add a left vertex that

[2]All results stated in Section 1.2 hold with constant success probability.

connects to every existing right vertex. Note that in $\mathbf{I}'$, all right vertices have degree $\tau+1$. Now truncating by τ fails DP: The query answer on $\mathbf{I}$ is $\tau N/2$, and that on $\mathbf{I}'$ is 0 (all right vertices are truncated). Adding noise of scale τ cannot mask this gap, violating the DP definition. $\square$

The reason why the truncation mechanism fails is that it requires after truncation, the query has the sensitivity bounded by τ, which does not hold in the presence of self-joins. More fundamentally, this is due to the correlation among the individuals introduced by self-joins. In the example above, we see that the addition of one node may cause the degrees of many others to increase. For the problem of graph pattern counting under node-DP, which can be formulated as a multi-way self-join counting query on the special schema $\mathbf{R} = \{\texttt{Node}(\underline{\texttt{ID}}), \texttt{Edge}(\texttt{src}, \texttt{dst})\}$, Kasiviswanathan et al. [55] propose an LP-based truncation mechanism (to differentiate, we will call the truncation mechanism above *naive truncation*) to fix the issue, but they do not study how to choose τ. As a result, while their mechanism satisfies DP, there is no optimality guarantee in terms of utility. In fact, if τ is chosen inappropriately, their error can be even larger than GS_Q.

1.2.3.1 Our Contribution [27]

We start by studying how to choose a near-optimal τ in a DP manner in the presence of self-joins. Here, we assume there is a predefined GS_Q. Since one tends to set a large GS_Q as argued before, we must try to minimize the dependency on GS_Q.

Our first contribution (Section 5.3) is a simple and general DP mechanism, called *Race-to-the-Top (R2T)*, which can be used to adaptively choose τ in combination with any valid DP truncation mechanism that satisfies certain properties. In fact, it does not choose τ per se; instead, it directly returns a privatized query answer with error at most $O(\log(\mathrm{GS}_Q)\log\log(\mathrm{GS}_Q)\cdot\mathrm{DS}_Q(\mathbf{I}))$ for any instance $\mathbf{I}$ with constant probability. As mentioned before, $\Omega(\mathrm{DS}_Q(\mathbf{I}))$ is down-neighborhood optimal lower bound. Thus, the error of R2T is optimal up to logarithmic factors in GS_Q.

However, as we see in Example 1.2.1, naive truncation is not a valid DP mechanism in the presence of self-joins. As our second contribution (Section 5.4), we extend the LP-based mechanism of [55], which only works for graph pattern counting queries, to general queries on an arbitrary relational schema that uses the 4 basic relational operators: Selection (with arbitrary predicates), Projection, Join (including self-join), and sum

Aggregation. When plugged into R2T, this yields the first DP mechanism for answering arbitrary SPJA queries in a database with FK constraints. For SJA queries, the utility is instance-optimal, while the optimality guarantee for SPJA queries (Section 5.5) is slightly weaker, but we argue that this is unavoidable.

Furthermore, the simplicity of our mechanism allows it to be built on top of any RDMBS and an LP solver. To demonstrate its practicality, we built a system prototype (Section 5.7) using PostgreSQL and CPLEX. Experimental results (Section 5.8) show that it can provide order-of-magnitude improvements in terms of utility over the state-of-the-art DP-SQL engines. We obtain similar improvements even over node-DP mechanisms that are specifically designed for graph pattern counting problems, which are just special SJA queries.

1.2.4 Multiple SJA Queries

Answering a single query is not very useful in practice. Thus it is natural to consider the multi-query problem in relational databases, which includes group-by queries as an important special case (i.e., each group corresponds to one query). Let $\mathbf{Q} = (Q_1, \ldots, Q_d)$ be the d SJA queries we wish to answer privately. We use the standard metric of root-mean-square error (RMSE) to measure the utility:

$$\|\widetilde{\mathbf{Q}}(\mathbf{I}) - \mathbf{Q}(\mathbf{I})\| = \sqrt{\sum_{k=1}^{d} (\widetilde{Q}_k(\mathbf{I}) - Q_k(\mathbf{I}))^2},$$

or equivalently, the ℓ_2 distance between the privatized query answers $\widetilde{\mathbf{Q}}(\mathbf{I})$ and the true answers $\mathbf{Q}(\mathbf{I})$, both taken as d-dimensional vectors. The notation $\|\cdot\|$ refers to the ℓ_2 norm of a vector throughout the thesis. Similarly, GS_Q and $\mathrm{DS}_{\mathbf{Q}}$ are also measured in ℓ_2 norm.

Here, we will allow the SJA queries to contain arbitrary joins and selection predicates. By advanced composition, we can allocate a privacy budget of $\tilde{O}(1/\sqrt{d})$ to each query and invoke our single-query mechanism R2T. This leads to an error of $\tilde{O}(\sqrt{d} \cdot \mathrm{DS}_{Q_k}(\mathbf{I}))$ for Q_k, hence an RMSE of

$$\tilde{O}\left(\sqrt{d} \cdot \sqrt{\sum_{k=1}^{d} \mathrm{DS}_{Q_k}(\mathbf{I})^2}\right). \tag{1.3}$$

Challenge/opportunity: Better than composition. We make the crucial observation that the error bound in (1.3) is *not* optimal. In Section 6.2, we show that the lower bound for the multi-query problem is $\tilde{\Omega}\left(\sqrt{d}\cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})\right)$. Note that we have the following relationship:

$$\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}) \leq \sqrt{\sum_{k=1}^{d} \mathrm{DS}_{Q_k}(\mathbf{I})^2} \leq \sqrt{d}\cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}).$$

Both inequalities are tight: The first inequality becomes an equality if the user with the maximum contribution to $\mathbf{Q}$ happens to be the maximum-contribution user to *every* $Q_k \in \mathbf{Q}$, and the second inequality becomes an equality if each user contributes to only one query in $\mathbf{Q}$. For typical database instances and queries (especially a group-by query), the situation will be more towards the latter, i.e., each user contributes to a small number of queries (groups), in which case the error bound of (1.3) can be a $\sqrt{d}$-factor away from optimal. This creates a third challenge, or rather, an opportunity for the multi-query problem, i.e., how to do better than privacy composition.

1.2.4.1 Our Contribution [28]

Our key insight is that answering all d queries as a whole can yield a much better result. We start by considering multiple self-join-free SJA queries. We observe that similar as 1 dimensional case, d such queries are equivalent to the sum (mean) estimation problem in d dimensions, a problem that has been extensively studied in the machine learning literature [46, 13, 49]. Restated in our terminology, their algorithms achieve the optimal error of $\tilde{O}\left(\sqrt{d}\cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})\right)$, modulo polylogarithmic factors. However, they are all restricted to instances $\mathbf{I}$ in which no user has contribution more than $\mathrm{GS}_{\mathbf{Q}}$ for some predefined $\mathrm{GS}_{\mathbf{Q}}$, and the hidden logarithmic factors depend on $\mathrm{GS}_{\mathbf{Q}}$. More precisely, the best error obtained so far [46] is

$$O\left(\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})\cdot\left(\sqrt{d}+\sqrt{\log(\mathrm{GS}_{\mathbf{Q}})\log\log(\mathrm{GS}_{\mathbf{Q}})}\right)\cdot\sqrt{\log(1/\delta)}\right). \tag{1.4}$$

Our first result is the complete removal of the dependency on $\mathrm{GS}_{\mathbf{Q}}$. Specifically, in Section 6.2 we design an algorithm that achieves an error of

$$O\left(\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})\cdot\left(\sqrt{d\log(1/\delta)}+\log\log(\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}))\right)\right). \tag{1.5}$$

Note that even assuming a finite $\mathrm{GS}_{\mathbf{Q}}$, the error bound of (1.5) is better than (1.4) since $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}) < \mathrm{GS}_{\mathbf{Q}}$ by definition. The key to obtaining this result is to find a near-optimal truncation threshold r under an unbounded $\mathrm{GS}_{\mathbf{Q}}$, and then the standard truncation mechanism can be applied.

Our main technical innovation is how to deal with self-joins. As mentioned in Section 1.2.3, self-joins are difficult to handle, since they make the truncation mechanism fail and to tackle self-joins, R2T [27] uses a series of linear programs (LPs). However, as we explain in Section 6.3.1, these LPs do not work for multiple queries due to fundamental reasons. Thereafter, we take a different approach to the multi-query problem, with the first version of the algorithm running in exponential time, which is subsequently reduced to polynomial using quadratically constrained quadratic programming (QCQP). We show that this algorithm achieves an error of

$$O\left(\sqrt{d} \cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}) \cdot \sqrt{\log(1/\delta)} \cdot (\log\log(\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})) + \log(1/\delta))\right),$$

matching the lower bound up to polylogarithmic factors.

Finally, we built a system prototype that can accept a set of SJA queries consisting of arbitrary joins, selection predicates, followed by aggregation. It can also automatically rewrite a group-by query into such a set of SJA queries and answer them with our query-answering mechanism. Experimental results demonstrate that our mechanism can significantly outperform privacy composition combined with the state-of-the-art single-query mechanism [27], especially on more skewed data and large d.

1.3 Organization

This thesis is organized as follows. The discussion of related work is given in Chapter 2. In Chapter 3, we talk about our algorithm to answer SPJA queries under tuple-DP. Then, our solutions for answering SA queries, SPJA queries, and multiple SJA queries under user-DP are given in Chapter 4, 5, and 6 respectively. Finally, we give a conclusion in Chapter 7.

CHAPTER 2

RELATED WORK

In the past several years, query answering under differential privacy [33, 32] has attracted a lot of attention [43, 12, 72, 71, 78, 82, 24, 63, 65, 68, 70, 8, 48, 57, 76]. Early works did not consider FK constraints, or equivalently, they adopt tuple-DP. Under tuple-DP, nearly all known work focus on answering counting queries, where as mentioned, if the query doesn't have joins, then the global sensitivity is 1, so the query answer can be released by just adding $O(1)$ noise. The problem becomes much more challenging when joins are present, because a single tuple may now affect many join results. A relatively easy approach is to add constraints so as to reduce the global sensitivity. McSherry [63] solves the problem by only restricting to one-to-one joins. Proserpio et al. [70] propose wPINQ to extend the work of McSherry to support general equijoins: by assigning weights to tuples and scaling down the weights, their algorithm ensures each tuple can at most affect one on final counting result. However, this only works well when one tuple affects a fixed number of results. Palamidessi and Stronati [68] add constraints on the attribute range. Arapinis et al. [8] and Narayan et al. [65] consider functional dependencies and cardinality constraints. In contrast, elastic sensitivity [48] and smooth sensitivity [67] do not require any constraints on the join structure or the database but have either the low utility or low efficiency. Our solution is the first one that has both high utility and high efficiency.

Starting from [57], people began to consider user-DP modeled by FK constraints. Under user-DP, self-join-free SJA queries [57, 76] are actually equivalent to SA queries, which further equal to the sum (mean) estimation problem [6, 46, 30]. Besides, people in statistic and statistics and machine learning community also studied mean estimation under statistical setting [75, 49, 51, 45, 54, 19, 17, 20, 13, 5, 46, 50, 16, 60, 9, 56], where data are drawn from some specific distributions. All these works are specifically designed for the given family and to achieve *pure*-DP (with $\delta = 0$) they must require a boundness assumption. Our solution can be further extended to give a pure-DP statistical mean estimator working on an arbitrary, unknown continuous distribution without any boundness assumption [30]. When self-joins appear, our solution is the first one to handle it. Besides, a consequent work [41] studies answering SJA queries with maximum aggregation. Their algorithm can also be used to answer sum aggregation query while the error bound is a little bit larger than us.

There are also a number works studying graph pattern counting queries under differential privacy [55, 15, 67, 81, 52, 22], which is an important special case of SJA queries. For graph data, there are two DP policies: edge-DP [67, 15, 81, 52] and node-DP [55, 15, 22]. They correspond to tuple-DP and user-DP applied to the special schema $\mathbf{R} = \{\texttt{Node}(\underline{\texttt{ID}}), \texttt{Edge}(\texttt{src}, \texttt{dst})\}$, respectively.

All the aforementioned works answer a single query at a time. The multi-query problem has been studied extensively on a flat table under tuple-DP[34, 44, 73, 43, 12, 72, 78, 82, 66, 59, 14, 79, 80, 64, 74]. For a set of d arbitrary linear queries, advanced composition or the d-dimensional Gaussian mechanism achieves $\tilde{O}(\sqrt{d})$ error for each query, which is the best we can achieve for $d < n$, where n is the size of the table. For $d > n$, the optimal error of each query is $\tilde{O}(\sqrt{n})$ [43]. For a set of queries with special structures, the error can be further reduced [78, 34, 64]. Furthermore, [66, 59, 14] design mechanisms that are optimal for any given query set.

Part I

Queries Answering under Tuple-level Differential Privacy

CHAPTER 3

ANSWERING SPJA QUERIES UNDER TUPLE-DP

As previously noted, the tuple-DP approaches focus on count aggregation. For the purpose of convenience, we formulate JA queries as full conjunctive counting queries. Subsequently, SJA queries and SPJA queries are expressed as full conjunctive counting queries with predicates and non-full conjunctive counting queries with predicates, respectively.

3.1 Preliminaries

3.1.1 Conjunctive Queries

Let $\mathbf{R}$ be a database schema. A *full conjunctive counting query (CQ)* has the form

$$Q := \left|R_1(\mathbf{x}_1) \bowtie \cdots \bowtie R_n(\mathbf{x}_n)\right|,$$

where $R_1, \dots, R_n$ are relation names in $\mathbf{R}$, and each $\mathbf{x}_i$ is a set of $arity(R_i)$ variables/attributes[1]. We call each $R_i(\mathbf{x}_i)$ an *atom*. We use $[n]$ to denote $\{1, \dots, n\}$, and $[i, j] = \{i, \dots, j\}$. For any $E \subseteq [n]$, $\bar{E} = [n] - E$. For a variable x, we use $\mathbf{dom}(x)$ to denote the domain of x. For $\mathbf{x} = (x_1, \dots, x_k)$, let $\mathbf{dom}(\mathbf{x}) = \mathbf{dom}(x_1) \times \cdots \times \mathbf{dom}(x_k)$. Let $var(Q)$ denote the set of variables in Q.

When considering self-joins, there can be repeats, i.e., $R_i = R_j$. In this case, we assume $\mathbf{x}_i \neq \mathbf{x}_j$; otherwise one of the two atoms is redundant. Let $\mathbf{I}$ be a database instance over $\mathbf{R}$. For a relation name $R \in \mathbf{R}$, let $\mathbf{I}(R)$ denote the relation instance of R. We use I_i as a shorthand for $\mathbf{I}(R_i)$. $\mathbf{I}$ and the I_i's are called *physical instances.* On the other hand, we use $I_i(\mathbf{x}_i)$ to denote I_i after renaming its attributes to $\mathbf{x}_i$. The $I_i(\mathbf{x}_i)$'s are called the *logical instances.* Note that if R_i and R_j are the same relation name, then $I_i = I_j$, but $I_i(\mathbf{x}_i) \neq I_j(\mathbf{x}_j)$, as $I_i(\mathbf{x}_i)$ and $I_j(\mathbf{x}_j)$ have different attributes. For a self-join-free query, we may without loss of generality assume that $\mathbf{x}_i = sort(R_i)$ for all $i \in [n]$

[1]If $\mathbf{x}_i$ has constants, we can preprocess $R_i(\mathbf{x}_i)$ in linear time so that only tuples that match these constants remain.

so the logical instances are the same as the physical instances, but for queries with self-joins, one physical relation instance may correspond to multiple logical relation instances. This distinction makes the problem more difficult under DP, as the distance between two logical instances may be larger than between the physical instances.

By rearranging the atoms, we may assume that all appearances of the same relation name are consecutive. Suppose m distinct relation names are mentioned in Q, and for $i = 1, \dots, m$, $R_{l_i}, \dots, R_{l_{i+1}-1}$ are the same relation name (set $l_{m+1} = n + 1$). Let $D_i = [l_i, l_{i+1} - 1]$ and $n_i = l_{i+1} - l_i$, which is the number of copies of R_{l_i} mentioned in Q.

3.1.2 Differential Privacy in Relational Databases under Tuple-DP

Differential privacy is already defined in (1.1). This notion can be applied to any problem by properly defining the distance function $d(\cdot, \cdot)$. As a database may contain both public and private relations, we use a more refined definition of $d(\cdot, \cdot)$. For two relation instances over the same relation name I, I', we use $d(I, I')$ to denote the distance between I and I', which is the minimum number of tuples to change I into I'. Note that $d(I_j, I'_j)$ is the same for all $j \in D_i$, for any $i \in [m]$, as they are the same physical relation instance.

We use $P^m \subseteq [m]$ to denote the set of private physical relations, while $P^n = \cup_{i \in P^m} D_i$ is the set of private logical relations. Let $m_P = |P^m|$, $n_P = |P^n|$. Two database instances can only differ in the private relations, i.e., $d(I_j, I'_j) = 0$ for every $j \in \bar{P}^n$. In the DP definition (1.1), we must use the distance between the physical database instances, i.e., $d(\mathbf{I}, \mathbf{I}') = \sum_{i \in [m]} d(I_{l_i}, I'_{l_i})$. Note that the distance between the logical instances, namely $\sum_{i \in [n]} d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i))$, can be larger than $d(\mathbf{I}, \mathbf{I}')$ when self-joins are present.

A simple but important observation is that a query with self-joins on instance $\{I_i\}_i$ can be considered as a query without self-joins on instance $\{I_i(\mathbf{x}_i)\}_i$. This allows us to reuse some of the technical results from self-join-free queries to support self-join queries. However, the critical difference is that this conversion enlarges the distance, while the DP guarantee must hold with respect to the distance on the original, physical instance.

3.1.3 Sensitivity-based DP Mechanisms

The most common technique of achieving differential privacy is to mask the query result by adding random noise drawn from a certain zero-mean probability distribution. The noise level (i.e., the standard deviation of the distribution) should depend on the difference between the query results on $\mathbf{I}$ and $\mathbf{I}'$, which is captured by the notion of *sensitivity.* The *local sensitivity* of Q at instance $\mathbf{I}$ is how much Q can change at most if one tuple in $\mathbf{I}$ is changed, i.e.,

$$\mathrm{LS}_Q(\mathbf{I}) = \max_{\mathbf{I}', d(\mathbf{I},\mathbf{I}')=1} \left| Q(\mathbf{I}) - Q(\mathbf{I}') \right|. \tag{3.1}$$

The *global sensitivity* of Q is

$$\mathrm{GS}_Q = \max_{\mathbf{I}} \mathrm{LS}_Q(\mathbf{I}).$$

It is well known that one can achieve ε-DP with $\mathrm{Err}(\mathcal{M}, \mathbf{I}) = O(\mathrm{GS}_Q)$ by drawing noise from the Laplace distribution calibrated to $\mathrm{GS}_Q/\varepsilon$ [33]. But unfortunately, the global sensitivity of many queries can be very high as the max is taken over all instances $\mathbf{I}$. For an n-way join, the global sensitivity can be as high as $O(N^{n-1})$. On the other hand, the local sensitivity is often much smaller in most real-world instances. However, as observed in [67], local sensitivity cannot be used to scale the noise directly, since $\mathrm{LS}_Q(\mathbf{I})$ and $\mathrm{LS}_Q(\mathbf{I}')$ can differ a lot on two neighboring instances $\mathbf{I}$ and $\mathbf{I}'$. Very different amounts of noise would be added to $Q(\mathbf{I})$ and $Q(\mathbf{I}')$, which breaches privacy.

3.1.3.1 Smooth Sensitivity

To address the issue, Nissim et al. [67] proposed *smooth sensitivity.* Similar with local sensitivity, smooth sensitivity is also instance-dependent and usually much smaller than global sensitivity. But different from local sensitivity, it eliminates abrupt changes between neighboring instances, hence the name “smooth sensitivity”.

The smooth sensitivity is based on the *local sensitivity at distance k*, $\mathrm{LS}_Q^{(k)}$, which is defined as

$$\mathrm{LS}_Q^{(k)}(\mathbf{I}) = \max_{\mathbf{I}', d(\mathbf{I},\mathbf{I}')\le k} \mathrm{LS}_Q(\mathbf{I}').$$

Note that $\mathrm{LS}_Q^{(0)}(\mathbf{I}) = \mathrm{LS}_Q(\mathbf{I})$ and $\forall k \ge 0, \mathrm{LS}_Q^{(k)}(\mathbf{I}) \le \mathrm{GS}_Q$. In fact, $\mathrm{LS}_Q^{(k)}$ can be equivalently defined as

$$\mathrm{LS}_Q^{(k)}(\mathbf{I}) = \max_{\mathbf{I}', d(\mathbf{I},\mathbf{I}')=k} \mathrm{LS}_Q(\mathbf{I}') \tag{3.2}$$

if dummy tuples are allowed in database.

Definition 3.1.1. *The β-smooth sensitivity of Q is*

$$\mathrm{SS}_Q^{\beta}(\mathbf{I}) = \max_{k\geq 0} e^{-\beta k}\mathrm{LS}_Q^{(k)}(\mathbf{I}). \tag{3.3}$$

An important property of $\mathrm{LS}_Q^{(k)}$ is that for any $\mathbf{I}, \mathbf{I}'$ such that $d(\mathbf{I},\mathbf{I}') = 1$, and any $k \geq 0$, $\mathrm{LS}_Q^{(k)}(\mathbf{I}) \leq \mathrm{LS}_Q^{(k+1)}(\mathbf{I}')$. This ensures the "smoothness" of $\mathrm{SS}_Q^{\beta}(\cdot)$: $\mathrm{SS}_Q^{\beta}(\mathbf{I})$ and $\mathrm{SS}_Q^{\beta}(\mathbf{I}')$ differ by at most a constant factor e^{β} on any two neighboring instances $\mathbf{I}$ and $\mathbf{I}'$.

Computing the smooth sensitivity by definition in general takes exponential time. Indeed, it has been shown that it is NP-hard to compute the smooth sensitivity for certain functions [67]. By exploiting special properties of the problem at hand, the running time can be reduced to polynomial; examples include the median, minimum spanning tree, and triangle counting [67]. However, it is still an open problem whether the smooth sensitivity of multi-way joins can be computed in polynomial time. In Section 3.2.4, we describe an algorithm with running time $N^{O(\log N)}$. Such a running time is said to be *quasi-polynomial*, which is super-polynomial but sub-exponential. This suggests that computing the smooth sensitivity for multi-way joins may not be NP-hard, although a polynomial-time algorithm still remains elusive.

To address this issue, Nissim et al. [67] show that any smooth upper bound of $\mathrm{SS}_Q^{\beta}(\cdot)$ can be used. Specifically, let

$$\widehat{\mathrm{SS}}_Q^{\beta}(\mathbf{I}) = \max_{k\geq 0} e^{-\beta k}\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}). \tag{3.4}$$

It has been shown that as long as $\widehat{\mathrm{LS}}_Q^{(k)}(\cdot)$ is an upper bound of $\mathrm{LS})_Q^{(k)}(\cdot)$ and satisfies the property, that for any neighbors $\mathbf{I}$ and $\mathbf{I}'$,

$$\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) \leq \widehat{\mathrm{LS}}_Q^{(k+1)}(\mathbf{I}'), \tag{3.5}$$

then, $\widehat{\mathrm{SS}}_Q^{\beta}(\cdot)$ follows the same "smoothness" property as $\mathrm{SS}_Q^{\beta}(\cdot)$.

After obtaining an $\widehat{\mathrm{SS}}_Q^{\beta}(\mathbf{I})/\mathrm{SS}_Q^{\beta}(\mathbf{I})$, adding noise to the query answer to achieve differential privacy is straightforward. In particular, Nissim et al. [67] describe the following two mechanisms.

General Cauchy The general Cauchy distribution has pdf $h(z) \propto \frac{1}{1+|z|^\gamma}$. It has bounded variance for $\gamma > 3$. It is shown [67] that by setting $\beta = \frac{\varepsilon}{2(\gamma+1)}$, adding noise $\frac{2(\gamma+1)}{\varepsilon} \cdot \widehat{\mathrm{SS}}^\beta_Q(\mathbf{I}) \cdot \eta$ to $Q(\mathbf{I})$ preserves ε-differential privacy, where η is drawn the general Cauchy distribution. We use $\gamma = 4$, for which $\mathrm{Var}[\eta] = 1$, and the noise level (i.e., the standard deviation of the noise distribution) is thus $\frac{10}{\varepsilon} \cdot \widehat{\mathrm{SS}}^\beta_Q(\mathbf{I})$.

Laplace The general Cauchy distribution has a heavy tail. Alternatively, one can use the Laplace distribution, which has a better concentration. However, adding noise according to the Laplace distribution only yields (ε, δ)-differential privacy. Specifically, one can set $\beta = \frac{\varepsilon}{2\ln(2/\delta)}$ and add noise $\frac{2}{\varepsilon} \cdot \widehat{\mathrm{SS}}^\beta_Q(\mathbf{I}) \cdot \eta$, where η is drawn from the Laplace distribution. Since $\mathrm{Var}[\eta] = 2$, the noise level is $\frac{2\sqrt{2}}{\varepsilon} \cdot \widehat{\mathrm{SS}}^\beta_Q(\mathbf{I})$. Note that the noise level of using the Laplace distribution may not be smaller than that of general Cauchy, because for the same ε, the Laplace mechanism requires a smaller β, which in turn leads to a larger $\widehat{\mathrm{SS}}^\beta_Q(\mathbf{I})$.

Note that the computation of $\widehat{\mathrm{SS}}^\beta_Q(\mathbf{I})$ and the subsequent noise calibration step are both done internally; only the noise-masked query result will be published in the end.

3.2 Residual Sensitivity

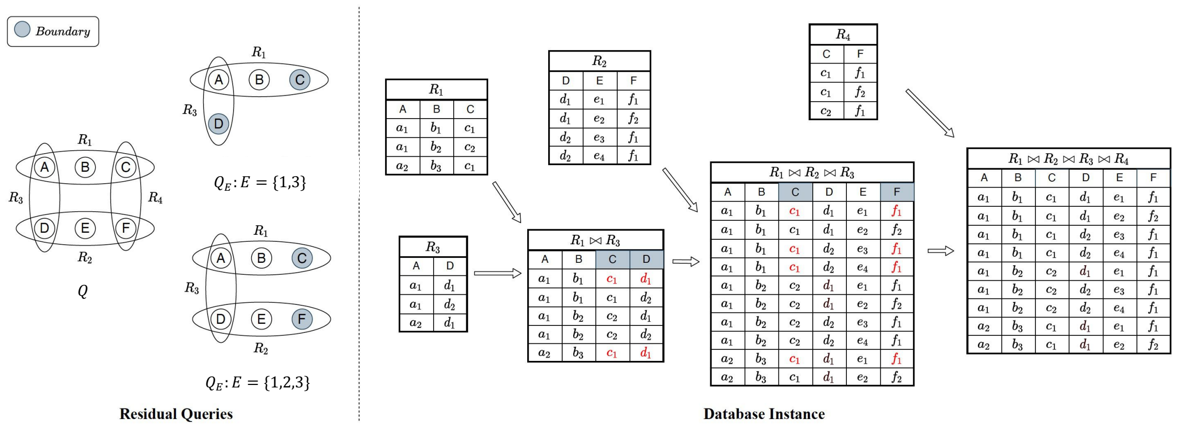
3.2.1 Residual Queries and Boundaries

Given a CQ Q, its *residual query* on a subset $E \subseteq [n]$ of relations is $Q_E := \Join_{i \in E} R_i(\mathbf{x}_i)$. Its *boundary*, denoted ∂Q_E, is the set of attributes that belong to atoms both in and out of E, i.e., $\partial Q_E = \{x \mid x \in \mathbf{x}_i \cap \mathbf{x}_j, i \in E, j \in \bar{E}\}$. The following notion plays an important role in our development.

Definition 3.2.1 (Maximum boundary and witness). *For a residual query Q_E on database instance $\mathbf{I}$, its maximum multiplicity over the boundary, or simply maximum boundary, is defined as*

$$T_E(\mathbf{I}) = \max_{t \in \mathbf{dom}(\partial Q_E)} |Q_E(\mathbf{I}) \ltimes t|.$$

A witness tuple of the maximum multiplicity over the boundary, or simply a witness, of



R_1		
A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_3	c_1

R_2		
D	E	F
d_1	e_1	f_1
d_1	e_2	f_2
d_2	e_3	f_1
d_2	e_4	f_1

R_3	
A	D
a_1	d_1
a_1	d_2
a_2	d_1

R_4	
C	F
c_1	f_1
c_1	f_2
c_2	f_1

$R_1 \bowtie R_3$			
A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_2
a_1	b_2	c_2	d_1
a_1	b_2	c_2	d_2
a_2	b_3	c_1	d_1

$R_1 \bowtie R_2 \bowtie R_3$					
A	B	C	D	E	F
a_1	b_1	c_1	d_1	e_1	f_1
a_1	b_1	c_1	d_1	e_2	f_2
a_1	b_1	c_1	d_2	e_3	f_1
a_1	b_1	c_1	d_2	e_4	f_1
a_1	b_2	c_2	d_1	e_1	f_1
a_1	b_2	c_2	d_1	e_2	f_2
a_1	b_2	c_2	d_2	e_3	f_1
a_1	b_2	c_2	d_2	e_4	f_1
a_2	b_3	c_1	d_1	e_1	f_1
a_2	b_3	c_1	d_1	e_2	f_2

$R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4$					
A	B	C	D	E	F
a_1	b_1	c_1	d_1	e_1	f_1
a_1	b_1	c_1	d_1	e_2	f_2
a_1	b_1	c_1	d_2	e_3	f_1
a_1	b_1	c_1	d_2	e_4	f_1
a_1	b_2	c_2	d_1	e_1	f_1
a_1	b_2	c_2	d_2	e_3	f_1
a_1	b_2	c_2	d_2	e_4	f_1
a_2	b_3	c_1	d_1	e_1	f_1
a_2	b_3	c_1	d_1	e_2	f_2

Figure 3.1: Residual query, boundary, maximum boundary, and witness.

Q_E *is*

$$t_E(\mathbf{I}) = \underset{t \in \mathbf{dom}(\partial Q_E)}{\arg\max} \; |Q_E(\mathbf{I}) \ltimes t|. \tag{3.6}$$

Per convention, when $E = \emptyset$, Q_E is an empty query and $Q_E(\mathbf{I}) = \{\langle\rangle\}$, where $\langle\rangle$ denotes the empty tuple, thus $T_\emptyset(\mathbf{I}) = 1$.

Example 3.2.1. Figure 3.1 illustrates these concepts using the query $Q = R_1(A, B, C) \bowtie R_2(D, E, F) \bowtie R_3(A, D) \bowtie R_4(C, F)$ on a specific database instance. The two residual queries shown are for $E = \{1, 3\}$ and $E = \{1, 2, 3\}$. For the first residual query, the boundary, maximum boundary, and witness tuple are $\{C, D\}$, 2, and (c_1, d_1) respectively. For the second one, those are $\{C, F\}$, 4, and (c_1, f_1). $\square$

Note that $T_E(\mathbf{I})$ can be computed by the following SQL query:

$$\texttt{SELECT MAX(Boundary) FROM} \tag{3.7}$$
$$\texttt{(SELECT COUNT(}*\texttt{) AS Boundary FROM } Q_E \texttt{ GROUP BY } \partial Q_E\texttt{)}$$

It is a special case of an *AJAR* query [47], and can be computed in $O(N^w)$ time, where w is a particular notion of the *width* of the query. The exact definition of w is a bit technical, but it is always a constant depending only on Q_E and ∂Q_E. Thus, $T_E(\mathbf{I})$ can be computed in polynomial time. Furthermore, various optimizations are possible; we discuss some of them in Section 3.6.2.

The following observations on the function $T_E(\cdot)$ are immediate.

Lemma 3.2.1. *For two instances* $\mathbf{I}, \mathbf{I}'$ *and any* $E \subseteq [n]$, *if* $I_i(\mathbf{x}_i) = I'_i(\mathbf{x}_i)$ *for all* $i \in E$, *then* $T_E(\mathbf{I}) = T_E(\mathbf{I}')$.

Lemma 3.2.2. *For* $\mathbf{I}, \mathbf{I}'$*, if* $I_i(\mathbf{x}_i) \subseteq I'_i(\mathbf{x}_i)$ *for all* $i \in E$*, then* $T_E(\mathbf{I}) \leq T_E(\mathbf{I}')$*.*

Notation	Meaning
$\mathbf{I}, \mathbf{I}'$	Database instances
$I_1, \dots, I_n$	Physical relation instances
$I_1(\mathbf{x}_1), \dots, I_n(\mathbf{x}_n)$	Logical relation instances
Q_E	Residual query
∂Q_E	Boundary of Q_E
T_E	Maximum boundary of Q_E
t_E	Witness of Q_E
$\mathcal{I}^k$	The set of instances having distance k to $\mathbf{I}$
$\mathbf{s}$	Distance vector used to describe the distance between two instances
$\mathbf{S}^k$	The set of distance vectors such that the total distance of all private relations is k
$\mathcal{I}^{\mathbf{s}}$	The set of instance differing from $\mathbf{I}$ with $\mathbf{s}$
$\widehat{T}_{E,\mathbf{s}}(\mathbf{I})$	An upper bound of $T_E(\mathbf{I}')$ for any $\mathbf{I}' \in \mathcal{I}^{\mathbf{s}}$
GS_Q	The global sensitivity of Q
$\mathrm{LS}_Q(\mathbf{I})$	The local sensitivity of instance $\mathbf{I}$
$\mathrm{LS}_Q^{(k)}(\mathbf{I})$	The local sensitivity of at distance k from $\mathbf{I}$
$\widehat{\mathrm{LS}}_{Q,\mathbf{s}}(\mathbf{I})$	An upper bound of $\mathrm{LS}_Q(\mathbf{I}')$ for any $\mathbf{I}' \in \mathcal{I}^{\mathbf{s}}$
$\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I})$	An upper bound of $\mathrm{LS}_Q^{(k)}(\mathbf{I}')$ for any $\mathbf{I}' \in \mathcal{I}^k$
$\mathrm{RS}_Q^{\beta}(\mathbf{I})$	Residual sensitivity of $\mathbf{I}$

Table 3.1: Notation used in the paper.

3.2.1.1 Sensitivity of T_E

The function $T_E(\mathbf{I})$ takes relation instances $I_i, i \in E$ as input and outputs a count, and its sensitivity is the maximum amount of change in $T_E(\mathbf{I})$ when $\mathbf{I}$ changes. Below, We first bound the sensitivity of $T_E(\mathbf{I})$ when only one tuple is changed. Then, we move onto the case where several tuples can change, but all the changes are in the same relation. Finally, we consider arbitrary changes.

Lemma 3.2.3. *Given any* $E \subseteq [n], i \in E$*, and two instances* $\mathbf{I}, \mathbf{I}'$ *such that* $d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i)) = 1$*,* $I_j(\mathbf{x}_j) = I'_j(\mathbf{x}_j)$ *for all* $j \in E - \{i\}$*, we have* $|T_E(\mathbf{I}) - T_E(\mathbf{I}')| \leq T_{E-\{i\}}(\mathbf{I})$*.*

Proof. $I'_i(\mathbf{x}_i)$ can be different from $I_i(\mathbf{x}_i)$ in three ways: insertion, deletion or change of a tuple.

For the first case, we have $T_E(\mathbf{I}') \geq T_E(\mathbf{I})$ by Lemma 3.2.2. Suppose the extra tuple in $I_i'(\mathbf{x}_i)$ is t'.

Let $t_E(\mathbf{I}')$ be a witness of $T_E(\mathbf{I}')$ as defined in (3.6), and let

$$\widetilde{T}_E(\mathbf{I}) = |Q_E(\mathbf{I}) \ltimes t_E(\mathbf{I}')|.$$

By definition, $T_E(\mathbf{I}) \geq \widetilde{T}_E(\mathbf{I})$, so

$$|T_E(\mathbf{I}) - T_E(\mathbf{I}')| = T_E(\mathbf{I}') - T_E(\mathbf{I}) \leq T_E(\mathbf{I}') - \widetilde{T}_E(\mathbf{I}).$$

Now we bound $T_E(\mathbf{I}') - \widetilde{T}_E(\mathbf{I})$:

$$\begin{aligned} & T_E(\mathbf{I}') - \widetilde{T}_E(\mathbf{I}) \\ =& |Q_E(\mathbf{I}') \ltimes t_E(\mathbf{I}')| - |Q_E(\mathbf{I}) \ltimes t_E(\mathbf{I}')| \\ =& |(\bowtie_{j \in E-\{i\}} I_j(\mathbf{x}_j)) \bowtie t' \ltimes t_E(\mathbf{I}')| \\ =& |(\bowtie_{j \in E-\{i\}} I_j(\mathbf{x}_j)) \ltimes (t' \bowtie t_E(\mathbf{I}')|. \end{aligned} \tag{3.8}$$

Note that $\bowtie_{j \in E-\{i\}} I_j$ is just $Q_{E-\{i\}}$ and $t' \bowtie t_E(\mathbf{I}')$ does not have any attribute interior to $Q_{E-\{i\}}$. So (3.8) is at most $T_{E-\{i\}}(\mathbf{I})$ by definition.

For the case where $\mathbf{I}'$ has one less tuple than $\mathbf{I}$ is symmetric, we have

$$|T_E(\mathbf{I}) - T_E(\mathbf{I}')| \leq T_{E-\{i\}}(\mathbf{I}') \leq T_{E-\{i\}}(\mathbf{I}),$$

where the last one is by Lemma 3.2.2.

For the third case where $I_i'(\mathbf{x}_i)$ is obtained from $I_i'(\mathbf{x}_i)$ by changing a tuple in $I_i'(\mathbf{x}_i)$, consider $I_i''(\mathbf{x}_i) = I_i(\mathbf{x}_i) \cap I_i'(\mathbf{x}_i)$. From previous work, we can derive

$$|T_E(\mathbf{I}) - T_E(\mathbf{I}')| \leq T_{E-\{i\}}(\mathbf{I}'') \leq T_{E-\{i\}}(\mathbf{I}),$$

where the last one is also by Lemma 3.2.2. □

Lemma 3.2.4. *Given any $E \subseteq [n], i \in E$ and two instances $\mathbf{I}, \mathbf{I}'$ such that $d(I_i(\mathbf{x}_i), I_i'(\mathbf{x}_i)) = k$, $I_j(\mathbf{x}_j) = I_j'(\mathbf{x}_j)$ for all $j \in E - \{i\}$, we have $|T_E(\mathbf{I}) - T_E(\mathbf{I}')| \leq k \cdot T_{E-\{i\}}(\mathbf{I})$.*

Proof. For the given $\mathbf{I}, \mathbf{I}'$, there is a sequence of instances $\mathbf{I}^0, \mathbf{I}^1, \ldots, \mathbf{I}^k$, such that $\mathbf{I}^0 = \mathbf{I}, \mathbf{I}^k = \mathbf{I}'$, while any two neighboring instances differ by one tuple in $I_i(\mathbf{x}_i)$.

Based on Lemma 3.2.1 and Lemma 3.2.3, for every $\ell \in [k]$,

$$|T_E(\mathbf{I}^{\ell-1}) - T_E(\mathbf{I}^{\ell})| \leq T_{E-\{i\}}(\mathbf{I}^{\ell-1}) = T_{E-\{i\}}(\mathbf{I}^0) = T_{E-\{i\}}(\mathbf{I}). \tag{3.9}$$

Summing (3.9) over all ℓ proves the lemma. □

Now, we consider the general case.

Lemma 3.2.5. *Given any $E \subseteq [n]$ and two instances $\mathbf{I}, \mathbf{I}'$, we have*

$$|T_E(\mathbf{I}) - T_E(\mathbf{I}')| \le \sum_{E' \subseteq E, E' \neq \emptyset} \left(T_{E-E'}(\mathbf{I}) \prod_{i \in E'} d(I_i(\mathbf{x}_i), I_i'(\mathbf{x}_i)) \right).$$

Proof. For any $\mathbf{I}, \mathbf{I}'$, there is a sequence of instances $\mathbf{I}^0, \mathbf{I}^1, \dots, \mathbf{I}^n$ such that $\mathbf{I}^0 = \mathbf{I}, \mathbf{I}^n = \mathbf{I}'$, while $\mathbf{I}^{\ell-1}$ and $\mathbf{I}^\ell$ differ only in I_ℓ, for $\ell \in [n]$. More precisely, $I_i^\ell(\mathbf{x}_i) = I_i'(\mathbf{x}_i)$ if $i \le \ell$, and $I_i^\ell(\mathbf{x}_i) = I_i(\mathbf{x}_i)$ if $i \ge \ell + 1$.

We will prove by induction that, for every $\ell = 0, 1, \dots, n$,

$$|T_E(\mathbf{I}^\ell) - T_E(\mathbf{I}^0)| \le \sum_{E' \subseteq E \cap [\ell], E' \neq \emptyset} \left(T_{E-E'}(\mathbf{I}) \prod_{i \in E'} d(I_i(\mathbf{x}_i), I_i'(\mathbf{x}_i)) \right), \tag{3.10}$$

for any $E \subseteq [n]$.

For the base case $\ell = 0$, both sides of (3.10) are 0. For the inductive step, assume (3.10) holds on $\ell - 1$ for any E. We will prove that it also holds on ℓ for any E. For any given E, we divide the set $\{E' : E' \subseteq E \cap [\ell], E' \neq \emptyset\}$ into two subsets $\mathcal{E}_1 = \{E' : E' \subseteq E \cap [\ell], \ell \in E'\}$ and $\mathcal{E}_2 = \{E' : E' \subseteq E \cap [\ell - 1], E' \neq \emptyset\}$, namely, $\mathcal{E}_1$ consists of E' that includes ℓ while $\mathcal{E}_2$ consists of those that do not. Consider the following two cases:

The easy case is when $\ell \notin E$. In this case, $\mathcal{E}_1$ is empty, and by Lemma 3.2.1, $T_E(\mathbf{I}^\ell) = T_E(\mathbf{I}^{\ell-1})$. Therefore,

$$\begin{aligned}
&|T_E(\mathbf{I}^\ell) - T_E(\mathbf{I}^0)| \\
=&|T_E(\mathbf{I}^{\ell-1}) - T_E(\mathbf{I}^0)| \\
\le& \sum_{E' \in \mathcal{E}_2} \left(T_{E-E'}(\mathbf{I}) \prod_{i \in E'} d(I_i(\mathbf{x}_i), I_i'(\mathbf{x}_i)) \right) \quad \text{(induction hypothesis)} \\
=& \sum_{E' \in \mathcal{E}_1 \cup \mathcal{E}_2} \left(T_{E-E'}(\mathbf{I}) \prod_{i \in E'} d(I_i(\mathbf{x}_i), I_i'(\mathbf{x}_i)) \right),
\end{aligned}$$

as desired.

The harder case is when $\ell \in E$. By Lemma 3.2.4, we have

$$|T_E(\mathbf{I}^\ell) - T_E(\mathbf{I}^{\ell-1})| \le d(I_\ell(\mathbf{x}_\ell), I_\ell'(\mathbf{x}_\ell)) \cdot T_{E-\{\ell\}}(\mathbf{I}^{\ell-1}). \tag{3.11}$$

Define $\prod_{i\in E'} d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i)) = 1$ if $E' = \emptyset$. From the induction hypothesis, we have

$$T_E(\mathbf{I}^{\ell-1}) \le \sum_{E'\subseteq E\cap[\ell-1]} \left(T_{E-E'}(\mathbf{I}) \prod_{j\in E'} d(I_j(\mathbf{x}_j), I'_j(\mathbf{x}_j)) \right). \tag{3.12}$$

Recall that the induction hypothesis holds for any E. In particular, we use $E - \{\ell\}$ in place of E in (3.12), and plug it into (3.11):

$$\begin{aligned}
&|T_E(\mathbf{I}^{\ell}) - T_E(\mathbf{I}^{\ell-1})| \\
\le& d(I_\ell(\mathbf{x}_\ell), I'_\ell(\mathbf{x}_\ell)) \sum_{E'\subseteq(E-\{\ell\})\cap[\ell-1]} \left(T_{E-\{\ell\}-E'}(\mathbf{I}) \prod_{i\in E'} d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i)) \right) \\
=& \sum_{E'\subseteq E\cap[\ell-1]} \left(T_{E-(E'\cup\{\ell\})}(\mathbf{I}) \prod_{i\in E'\cup\{\ell\}} d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i)) \right) \\
=& \sum_{E'\subseteq E\cap[\ell],\ell\in E'} \left(T_{E-E'}(\mathbf{I}) \prod_{i\in E'} d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i)) \right) \\
=& \sum_{E'\in\mathcal{E}_1} \left(T_{E-E'}(\mathbf{I}) \prod_{i\in E'} d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i)) \right).
\end{aligned} \tag{3.13}$$

Now we can finish the inductive step:

$$\begin{aligned}
&|T_E(\mathbf{I}^{\ell}) - T_E(\mathbf{I}^{0})| \\
\le& |T_E(\mathbf{I}^{\ell}) - T_E(\mathbf{I}^{\ell-1})| + |T_E(\mathbf{I}^{\ell-1}) - T_E(\mathbf{I}^{0})| \\
\le& \sum_{E'\in\mathcal{E}_1} \left(T_{E-E'}(\mathbf{I}) \prod_{i\in E'} d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i)) \right) \quad \text{(by (3.13))} \\
&+ \sum_{E'\in\mathcal{E}_2} \left(T_{E-E'}(\mathbf{I}) \prod_{i\in E'} d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i)) \right) \text{(induction hypothesis)} \\
=& \sum_{E'\in\mathcal{E}_1\cup\mathcal{E}_2} \left(T_{E-E'}(\mathbf{I}) \prod_{j\in E'} d(I_j(\mathbf{x}_i), I'_j(\mathbf{x}_i)) \right).
\end{aligned}$$

□

3.2.2 Local Sensitivity of CQs

Now, we show For any CQ Q, its local sensitivity is precisely characterized by the $T_E(\cdot)$'s. Let's first consider the self-join-free queries.

Lemma 3.2.6. *For a CQ without self-joins,*

$$\mathrm{LS}_Q(\mathbf{I}) = \max_{i \in P^n} T_{\overline{\{i\}}}(\mathbf{I}).$$

Proof. First, note that the definition of $\mathrm{LS}_Q(\mathbf{I})$ in (3.1) can be rewritten as

$$\mathrm{LS}_Q(\mathbf{I}) = \max_{i \in P^n} \max_{\mathbf{I}', d(\mathbf{I},\mathbf{I}')=1, d(I_i, I_i')=1} |Q(\mathbf{I}) - Q(\mathbf{I}')|.$$

Consider any $\mathbf{I}'$ with $d(\mathbf{I}, \mathbf{I}') = 1, d(I_i(\mathbf{x}_i), I_i'(\mathbf{x}_i)) = 1$. Similar as the proof of Lemma 3.2.3, $I_i'(\mathbf{x}_i)$ can be different from $I_i(\mathbf{x}_i)$ in three ways: insertion, deletion, or change of a tuple $t' \in \mathbf{dom}(\mathbf{x}_i)$. In the first case, $I_i'(\mathbf{x}_i) = I_i(\mathbf{x}_i) \cup \{t'\}, t' \notin I_i(\mathbf{x}_i)$, so

$$\begin{aligned}|Q(\mathbf{I}) - Q(\mathbf{I}')| &= Q(\mathbf{I}') - Q(\mathbf{I}) \\ &= | \bowtie_{j \in [n], j \neq i} (I_j(\mathbf{x}_j) \ltimes t')|.\end{aligned}$$

Similarly, for the second case, $I_i'(\mathbf{x}_i) = I_i(\mathbf{x}_i) - t', t' \in I_i(\mathbf{x}_i)$ and

$$|Q(\mathbf{I}) - Q(\mathbf{I}')| = | \bowtie_{j \in [n], j \neq i} (I_j(\mathbf{x}_j) \ltimes t')|.$$

Thus, over all $\mathbf{I}'$ that differs from $\mathbf{I}$ by the insertion or deletion of a tuple in $I_i(\mathbf{x}_i)$, we have

$$\begin{aligned}&\max_{\mathbf{I}'} |Q(\mathbf{I}) - Q(\mathbf{I}')| \\ =&\max \Bigg\{ \max_{t' \in \mathbf{I}} | \bowtie_{j \in [n], j \neq i} (I_j(\mathbf{x}_j) \ltimes t')|, \\ &\max_{t' \in \mathbf{dom}(\mathbf{x}_i), t' \notin \mathbf{I}} | \bowtie_{j \in [n], j \neq i} (I_j(\mathbf{x}_j) \ltimes t')| \Bigg\} \\ =& \max_{t \in \mathbf{dom}(\partial Q_{[n]-\{i\}})} | \bowtie_{j \in [n], j \neq i} (I_j(\mathbf{x}_j) \ltimes t)| \\ =& T_{[n]-\{i\}}(\mathbf{I}).\end{aligned}$$

Finally, the third case can be handled using a similar argument as in the proof of Lemma 3.2.3.

Summarizing the three cases, we have

$$\max_{\mathbf{I}', d(\mathbf{I},\mathbf{I}')=1, d(I_i(\mathbf{x}_i), I_i'(\mathbf{x}_i))=1} |Q(\mathbf{I}) - Q(\mathbf{I}')| = T_{[n]-\{i\}}(\mathbf{I}),$$

and

$$\begin{aligned}\mathrm{LS}_Q(\mathbf{I}) &= \max_{i \in P^n} \max_{\mathbf{I}', d(\mathbf{I},\mathbf{I}')=1, d(I_i(\mathbf{x}_i), I_i'(\mathbf{x}_i))=1} |Q(\mathbf{I}) - Q(\mathbf{I}')| \\ &= \max_{i \in P^n} T_{[n]-\{i\}}(\mathbf{I}).\end{aligned}$$

□

To extend this result to CQs with self-joins, we need to bound how much $Q(\mathbf{I})$ can change when multiple relations change simultaneously, as a change in one physical relation instance may correspond to changes in multiple logical relations when self-joins are present. We first draw the conclusion for the self-join-free queries.

Lemma 3.2.7. *Let Q be a CQ without self-joins, $B \subseteq [n]$, $B \neq \emptyset$, and let $\mathbf{I}, \mathbf{I}'$ be instances such that $d(I_j(\mathbf{x}_j), I'_j(\mathbf{x}_j)) = 1$ for all $j \in B$ while $d(I_j(\mathbf{x}_j), I'_j(\mathbf{x}_j)) = 0$ otherwise. Then*

$$|Q(\mathbf{I}) - Q(\mathbf{I}')| \leq \sum_{E \subseteq B, E \neq \emptyset} T_{\bar{E}}(\mathbf{I}).$$

Proof. For the given $\mathbf{I}, \mathbf{I}'$, there is a sequence of intermediate instances $\mathbf{I}^0, \mathbf{I}^1, \dots, \mathbf{I}^n$ such that, $\mathbf{I}^0 = \mathbf{I}$, $\mathbf{I}^n = \mathbf{I}'$, while $\mathbf{I}^{\ell-1}, \mathbf{I}^{\ell}$ can only possibly differ in $I_\ell(\mathbf{x}_\ell)$, $\ell = 1, 2, \dots, n$. More precisely, we set $I^\ell_j(\mathbf{x}_j) = I'_j(\mathbf{x}_j)$ if $j \leq \ell$; otherwise $I^\ell_j(\mathbf{x}_j) = I_j(\mathbf{x}_j)$.

Notice that, for all $j \notin B, \mathbf{I}^j = \mathbf{I}^{j-1}$, that is

$$\left|Q(\mathbf{I}^j)| - Q(\mathbf{I}^{j-1})\right| = 0.$$

For all $j \in B$, $d(\mathbf{I}^j, \mathbf{I}^{j-1}) = d(I^j_j(\mathbf{x}_j), I^{j-1}_j(\mathbf{x}_j)) = 1$. By Lemma 3.2.6,

$$\left|Q(\mathbf{I}^j) - Q(\mathbf{I}^{j-1})\right| \leq T_{[n]-\{j\}}(\mathbf{I}^{j-1}).$$

Therefore,

$$|Q(\mathbf{I}) - Q(\mathbf{I}')| = \left|Q(\mathbf{I}^n) - Q(\mathbf{I}^0)\right| \leq \sum_{j \in B} T_{[n]-\{j\}}(\mathbf{I}^{j-1}). \tag{3.14}$$

Note that for a self-join-free query, $I_i = I_i(\mathbf{x}_i)$. By Lemma 3.2.5, for all $j \in B$,

$$\begin{aligned}
& T_{[n]-\{j\}}(\mathbf{I}^{j-1}) \\
\leq & T_{[n]-\{j\}}(\mathbf{I}^0) + \sum_{E \subseteq [n]-\{j\}, E \neq \emptyset} \left(T_{[n]-\{j\}-E}(\mathbf{I}^0) \prod_{t \in E} d(I^{j-1}_t(\mathbf{x}_t), I^0_t(\mathbf{x}_t)) \right) \\
= & \sum_{E \subseteq [n]-\{j\}} \left(T_{[n]-\{j\}-E}(\mathbf{I}^0) \prod_{t \in E} d(I^{j-1}_t(\mathbf{x}_t), I^0_t(\mathbf{x}_t)) \right) \\
= & \sum_{E \subseteq B \cap [j-1]} T_{[n]-\{j\}-E}(\mathbf{I}^0) & (3.15) \\
= & \sum_{E \subseteq B \cap [j], j \in E} T_{\bar{E}}(\mathbf{I}^0). & (3.16)
\end{aligned}$$

(3.15) is because, by definition, for $\mathbf{I}^{j-1}, \mathbf{I}^0, j \in B$, $d(I_t^{j-1}(\mathbf{x}_t), I_t^0(\mathbf{x}_t)) = 1$ if $t \in B \cap [j-1]$; otherwise $d(I_t^{j-1}(\mathbf{x}_t), I_t^0(\mathbf{x}_t)) = 0$.

Plugging (3.16) into (3.14), we obtain

$$|Q(\mathbf{I}) - Q(\mathbf{I}')| \leq \sum_{j \in B} \sum_{E \subseteq B \cap [j], j \in E} T_{\bar{E}}(\mathbf{I}^0) = \sum_{E \subseteq B, E \neq \emptyset} T_{\bar{E}}(\mathbf{I}).$$

□

Based on this and (3.1), we can derive an upper bound on $\mathrm{LS}_Q(\mathbf{I})$ for CQs with self-joins.

Theorem 3.2.1. *For a CQ Q,*

$$\mathrm{LS}_Q(\mathbf{I}) \leq \max_{i \in P^m} \sum_{E \subseteq D_i, E \neq \emptyset} T_{\bar{E}}(\mathbf{I}).$$

Remark Note that when self-joins are present, we can no longer obtain an exact formula for $\mathrm{LS}_Q(\mathbf{I})$ like for self-join-free queries in Lemma 3.2.6. This is precisely due to the fact that self-joins induce changes in multiple logical relations that may interact in complex manners.

3.2.3 Global Sensitivity of CQs

Because $\mathrm{GS}_Q = \max_{\mathbf{I}} \mathrm{LS}_Q(\mathbf{I})$, a by-product of Theorem 3.2.1 is an upper bound on GS_Q under relaxed DP where the instance size N is public. This upper bound can be much smaller than the trivial upper bound $O(N^{n-1})$ mentioned in Section 3.1.3 for many CQs.

By Theorem 3.2.1, we have

$$\mathrm{GS}_Q \leq \max_{\mathbf{I}} \max_{i \in P^m} \sum_{E \subseteq D_i, E \neq \emptyset} T_{\bar{E}}(\mathbf{I}) \leq \max_{i \in P^m} \sum_{E \subseteq D_i, E \neq \emptyset} \max_{\mathbf{I}} T_{\bar{E}}(\mathbf{I}). \tag{3.17}$$

Observe that $\max_{\mathbf{I}} T_{\bar{E}}(\mathbf{I})$ is upper bounded by the maximum join size of the residual query $Q_{\bar{E}}$, when the logical relations of the same physical relation are allowed to be instantiated differently and the domain size of each variable in $\partial Q_E = \partial Q_{\bar{E}}$ is set to 1, which is equivalent to removing these variables. We can bound the maximum join size using the *AGM bound* [11]. Together with (3.17) this yields an upper bound on GS_Q.

Example 3.2.2. We illustrate how this is done on the triangle counting query $Q = |\texttt{Edge}(x_1, x_2) \bowtie \texttt{Edge}(x_2, x_3) \bowtie \texttt{Edge}(x_1, x_3)|$ on a single physical relation $\texttt{Edge}$. For $E = \{3\}$, i.e., $Q_{\bar{E}} = \texttt{Edge}(x_1, x_2) \bowtie \texttt{Edge}(x_2, x_3)$ and $\partial Q_{\bar{E}} = \{x_1, x_3\}$, we have

$$\begin{aligned}\max_{\mathbf{I}} T_{\bar{E}}(\mathbf{I}) =& \max_{\mathbf{I}} \max_{t \in \mathbf{dom}(x_1, x_3)} |\texttt{Edge}(x_1, x_2) \bowtie \texttt{Edge}(x_2, x_3)| \\ \le& \max_{\mathbf{I}} \max_{t \in \mathbf{dom}(x_1, x_3)} |\texttt{Edge}_1(x_1, x_2) \bowtie \texttt{Edge}_2(x_2, x_3)| \\ =& \max_{\mathbf{I}} (\texttt{Edge}_1(x_2) \bowtie \texttt{Edge}_2(x_2)). \\ =& \mathsf{AGM}(\texttt{Edge}_1(x_2) \bowtie \texttt{Edge}_2(x_2)).\end{aligned}$$

We can similarly derive a bound for other E's. Note that when E consists of 2 relations, $\max_{\mathbf{I}} T_{\bar{E}}(\mathbf{I}) \le 1$. Thus,

$$\begin{aligned}\mathrm{GS}_Q \le& \mathsf{AGM}(\texttt{Edge}_1(x_2) \bowtie \texttt{Edge}_2(x_2)) \\ &+ \mathsf{AGM}(\texttt{Edge}_2(x_3) \bowtie \texttt{Edge}_3(x_3)) \\ &+ \cdots \\ =& O(N).\end{aligned}$$

□

Example 3.2.3. As a more complicated example, consider the path-4 counting query

$$Q = |\texttt{Edge}(x_1, x_2) \bowtie \texttt{Edge}(x_2, x_3) \bowtie \texttt{Edge}(x_3, x_4) \bowtie \texttt{Edge}(x_4, x_5)|.$$

We have

$$\begin{aligned}\mathrm{GS}_Q \le& \mathsf{AGM}(\texttt{Edge}_2(x_3) \bowtie \texttt{Edge}_3(x_3, x_4) \bowtie \texttt{Edge}_4(x_4, x_5)) \\ &+ \mathsf{AGM}(\texttt{Edge}_1(x_1) \bowtie \texttt{Edge}_3(x_4) \bowtie \texttt{Edge}_4(x_4, x_5)) \\ &+ \mathsf{AGM}(\texttt{Edge}_1(x_1, x_2) \bowtie \texttt{Edge}_2(x_2) \bowtie \texttt{Edge}_4(x_5)) \\ &+ \cdots \\ =& O(N^2).\end{aligned}$$

□

In general, any join size upper bound can be plugged into (3.17). For example, when degree information or functional dependencies are publicly available, tighter upper bounds can be derived [42, 3]. Although DP mechanisms based on GS_Q are not as accurate as our $\mathrm{RS}_Q^{\beta}(\cdot)$-based mechanisms to be presented next, they can be computed in $O(1)$ time (excluding the time for computing $Q(\mathbf{I})$).

3.2.4 Smooth Sensitivity of CQs

In this section, we describe an $N^{O(\log N)}$-time algorithm for computing $\text{SS}_Q^\beta(\mathbf{I})$ for CQs. Recall the definition of $\text{SS}_Q^\beta(\mathbf{I})$:

$$\text{SS}_Q^\beta(\mathbf{I}) = \max_{k\geq 0} e^{-\beta k}\text{LS}_Q^{(k)}(\mathbf{I}).$$

First, we show that it is sufficient to consider $k = 0, 1, \ldots, \frac{2(n-1)}{\beta} \cdot \ln N$. It is trivial to see, $\text{LS}_Q^{(k)}(\mathbf{I}) \leq (N+k)^{n-1}$. When $k \geq \frac{2(n-1)}{\beta} \cdot \ln N$, we have $e^{-\beta k} \cdot \text{LS}_Q^{(k)}(\mathbf{I}) \leq e^{-\beta k} \cdot (N + k)^{n-1} \leq 1$, so it has no effects on the max.

Then, for each k, we need to compute

$$\text{LS}_Q^{(k)}(\mathbf{I}) = \max_{\mathbf{I}'\in\mathcal{I}^k} \text{LS}_Q(\mathbf{I}'),$$

where $\mathcal{I}^k = \{I' : d(\mathbf{I}, \mathbf{I}') = k\}$. The domains of the attributes can be infinite, thus $\mathcal{I}^k$ is also infinite. To resolve this issue, we show that we do not need to consider the entire domains of the attributes, while some finite sub-domains are sufficient for computing $\text{LS}_Q^{(k)}$.

For a given instance $\mathbf{I}$ and any attribute x, the *active domain* of x is $\mathbf{dom}_{act}(x) = \cup_{i\in[n]}\pi_x I_i$. For $\mathbf{x} = (x_1, \ldots, x_k)$, let $\mathbf{dom}_{act}(\mathbf{x}) = \mathbf{dom}_{act}(x_1) \times \cdots \times \mathbf{dom}_{act}(x_k)$. Tao et al. [76] show that when computing $\text{LS}_Q(\mathbf{I})$, only neighboring instances $\mathbf{I}, \mathbf{I}'$ that differ by a tuple $t \in \mathbf{dom}_{act}(\mathbf{x}_i), i \in P^n$ need to be considered. However, when computing $\text{LS}_Q^{(k)}$, we cannot only consider $\mathbf{I}'$ that differ from $\mathbf{I}$ by k tuples in $\mathbf{dom}(\mathbf{x}_i), i \in P^n$. The reason is that, these k tuples may correlate with each other through values not in the active domains. To solve this issue, we add k values to $\mathbf{dom}_{act}(x)$ to form the *k-extended active domain* for each attribute x:

$$\widehat{\mathbf{dom}}_{act}^{k}(x) = \mathbf{dom}_{act}(x) \cup \{a_1, \ldots, a_k\},$$

where $a_1, \ldots, a_k$ are k different values in $\mathbf{dom}(x) - \mathbf{dom}_{act}(x)$. Similarly, let $\mathbf{dom}_{act}(\mathbf{x}) = \mathbf{dom}_{act}(x_1) \times \cdots \times \mathbf{dom}_{act}(x_k)$ for $\mathbf{x} = (x_1, \ldots, x_k)$. We claim that when computing $\text{LS}_Q^{(k)}$, it is sufficient to consider

$$\mathcal{I}^k = \{\mathbf{I}' : \mathbf{I}, \mathbf{I}' \text{ differ by } k \text{ tuples in } \widehat{\mathbf{dom}}_{act}^{k}(\mathbf{x}_i)\}.$$

Indeed, suppose $\mathbf{I}'$ has tuples have value a' on attribute x and $a' \notin \widehat{\mathbf{dom}}^k_{act}(x)$. Because $\mathbf{I}'$ has at most k tuples not in $\mathbf{I}$, there must be an extra a_i in $\widehat{\mathbf{dom}}^k_{act}(x)$ that is not used. Then we can remap a' to a_i, which does not induce any structural change in $\mathbf{I}'$. Note that the size of $|\mathcal{I}^k|$ is at most $O((N+k)^{mk})$, where m is the number of attributes. This is $N^{O(\ln N)}$ in terms of data complexity.

Then, we can calculate the $\mathrm{SS}^\beta_Q(\mathbf{I})$ as follows. For each $k = 0, 1, \dots, \frac{2(n-1)}{\beta} \cdot \ln N$, we compute $\mathrm{LS}^{(k)}_Q(\mathbf{I})$. To compute $\mathrm{LS}^{(k)}_Q(\mathbf{I})$, we enumerate all $\mathbf{I}' \in \mathcal{I}^k$, and use Theorem 3.2.1 to calculate $\mathrm{LS}_Q(\mathbf{I}')$, hence $\mathrm{LS}^{(k)}_Q(\mathbf{I})$. Finally, we obtain $\mathrm{SS}^\beta_Q(\mathbf{I})$ by taking the maximum of $e^{\beta k}\mathrm{LS}^{(k)}_Q(\mathbf{I})$ over all k. The running time is still $N^{O(\ln N)}$.

3.2.5 Deriving $\widehat{\mathrm{LS}}^{(k)}_Q(\cdot)$

In the last section, we show computing $\mathrm{SS}^\beta_Q(\mathbf{I})$ for CQs is not NP hard problem but requires quasi-polynomial time. Here, we propose an efficient algorithm to approximate it. As mentioned in Section 3.1.3.1, the key here is to derive an upper bound of $\mathrm{LS}^{(k)}_Q(\mathbf{I})$ with smoothness property.

For any two instances $\mathbf{I}, \mathbf{I}'$, define their *distance vector* as

$$\mathbf{s} = (d(I_1, I'_1), d(I_2, I'_2), \dots, d(I_n, I'_n)).$$

For any $\mathbf{s} = (s_1, \dots, s_n)$, let $\mathcal{I}^{\mathbf{s}} = \{\mathbf{I}' : d(I_j, I'_j) = s_j, j \in [n]\}$ be the set of instances whose distance vectors are $\mathbf{s}$ from $\mathbf{I}$. Note that when self-joins are present, not any $\mathbf{s} \in \mathbb{N}^n$ is a valid distance vector. We must ensure $s_{l_i} = s_{l_i+1} = \cdots = s_{l_i+n_i-1}$, for any $i \in [m]$. Let $\mathbf{S}^k$ be the set of valid distance vectors such that the total distance of all private relations is k, i.e.,

$$\mathbf{S}^k = \left\{ \mathbf{s} : \sum_{i \in P^m} s_{l_i} = k; s_j = 0, j \in \bar{P}^n; \forall i \in [m], j \in D_i, s_j = s_{l_i} \right\}.$$

Denote the set of instances at k distance from $\mathbf{I}$ as $\mathcal{I}^k = \{\mathbf{I}' : d(\mathbf{I}, \mathbf{I}') = k\}$, i.e.,

$$\mathcal{I}^k = \cup_{\mathbf{s} \in \mathbf{S}^k} \mathcal{I}^{\mathbf{s}}.$$

We now derive an upper bound of $\mathrm{LS}^{(k)}_Q(\cdot)$ in terms of $T_E(\cdot)$.

Lemma 3.2.8.

$$\mathrm{LS}^{(k)}_Q(\mathbf{I}) \le \max_{\mathbf{s} \in \mathbf{S}^k} \max_{i \in P^m} \sum_{E \subseteq D_i, E \neq \emptyset} \max_{\mathbf{I}' \in \mathcal{I}^{\mathbf{s}}} T_{\bar{E}}(\mathbf{I}').$$

Proof. By (3.2), we have

$$\begin{aligned}
\mathrm{LS}_Q^{(k)}(\mathbf{I}) &= \max_{\mathbf{I}', d(\mathbf{I},\mathbf{I}')=k} \mathrm{LS}_Q(\mathbf{I}') \\
&= \max_{\mathbf{I}'\in\mathcal{I}^k} \max_{i\in P^m} \mathrm{LS}_Q(\mathbf{I}') \\
&\le \max_{\mathbf{I}'\in\mathcal{I}^k} \max_{i\in P^m} \sum_{E\subseteq D_i, E\neq\emptyset} T_{\bar{E}}(\mathbf{I}') \qquad (3.18) \\
&= \max_{\mathbf{s}\in\mathbf{S}^k} \max_{\mathbf{I}'\in\mathcal{I}^{\mathbf{s}}} \max_{i\in P^m} \sum_{E\subseteq D_i, E\neq\emptyset} T_{\bar{E}}(\mathbf{I}') \\
&= \max_{\mathbf{s}\in\mathbf{S}^k} \max_{i\in P^m} \max_{\mathbf{I}'\in\mathcal{I}^{\mathbf{s}}} \sum_{E\subseteq D_i, E\neq\emptyset} T_{\bar{E}}(\mathbf{I}') \\
&\le \max_{\mathbf{s}\in\mathbf{S}^k} \max_{i\in P^m} \sum_{E\subseteq D_i, E\neq\emptyset} \max_{\mathbf{I}'\in\mathcal{I}^{\mathbf{s}}} T_{\bar{E}}(\mathbf{I}').
\end{aligned}$$

(3.18) follows Theorem 3.2.1. □

Let $\widehat{T}_{E,\mathbf{s}}(\mathbf{I})$ be an upper bound of $\max_{\mathbf{I}'\in\mathcal{I}^{\mathbf{s}}} T_E(\mathbf{I}')$. Then

$$\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) := \max_{\mathbf{s}\in\mathbf{S}^k} \max_{i\in P^m} \sum_{E\subseteq D_i, E\neq\emptyset} \widehat{T}_{\bar{E},\mathbf{s}}(\mathbf{I}'), \qquad (3.19)$$

is clearly an upper bound of $\mathrm{LS}_Q^{(k)}(\mathbf{I})$.

Now, it remains to find a valid $\widehat{T}_{E,\mathbf{s}}(\mathbf{I})$. By Lemma 3.2.5, we have for any $E \subseteq [n]$ and any $\mathbf{I}' \in \mathcal{I}^{\mathbf{s}}$,

$$T_E(\mathbf{I}') \le T_E(\mathbf{I}) + \sum_{E'\subseteq E, E'\neq\emptyset} \left(T_{E-E'}(\mathbf{I}) \prod_{j\in E'} s_j \right).$$

So we set $\widehat{T}_{E,\mathbf{s}}(\mathbf{I})$ as

$$\widehat{T}_{E,\mathbf{s}}(\mathbf{I}) := \sum_{E'\subseteq E} \left(T_{E-E'}(\mathbf{I}) \prod_{j\in E'} s_j \right), \qquad (3.20)$$

where we define $\prod_{j\in\emptyset} s_j = 1$.

Finally, the residual sensitivity is defined as in (3.4) by setting $\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I})$ as in (3.19):

$$\mathrm{RS}_Q^{\beta}(\mathbf{I}) = \max_{k\ge 0} e^{-\beta k} \widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}). \qquad (3.21)$$

However, in order for $\mathrm{RS}_Q^{\beta}(\cdot)$ to be a valid DP mechanism, we need to show that $\widehat{\mathrm{LS}}_Q^{(k)}(\cdot)$ follows the smoothness property (3.5). To do so, we first derive a technical result:

Lemma 3.2.9. *Given any self-join-free CQ, any $E \subseteq [n]$, any $i \in [n]$, two instances $\mathbf{I}, \mathbf{I}'$ that differ by one tuple in $R_i(\mathbf{x}_i)$, any two distance vectors $\mathbf{s} = (s_1, \dots, s_n)$ and $\mathbf{s}' = (s'_1, \dots, s'_n)$ such that*

$$\mathbf{s}' = (s_1, \dots, s_{i-1}, s_i + 1, s_{i+1}, \dots, s_n),$$

we have

$$\sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}) \prod_{j \in E'} s_j \right) \leq \sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}') \prod_{j \in E'} s'_j \right).$$

Proof. If $i \notin E$, then for any $E' \subseteq E$, $j \in E - E'$, we have $I_j(\mathbf{x}_j) = I'_j(\mathbf{x}_j)$. Thus $T_{E-E'}(\mathbf{I}) = T_{E-E'}(\mathbf{I}')$ by Lemma 3.2.1. Meanwhile, $s_j = s'_j$ for all $j \in E'$. Therefore, $\sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}) \prod_{j \in E'} s_j \right) = \sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}') \prod_{j \in E'} s'_j \right)$ in this case.

If $i \in E$, we divide the set $\{E' \subseteq E\}$ into two subsets $\mathcal{E}_1 = \{E' : E' \subseteq E, i \in E'\}$ and $\mathcal{E}_2 = \{E' : E' \subseteq E, i \notin E'\}$. Note that there is one-to-one correspondence between the subsets in $\mathcal{E}_1$ and the subsets in $\mathcal{E}_2$, i.e., for any $E' \in \mathcal{E}_2$, $E' \cup \{i\} \in \mathcal{E}_1$, and vice versa.

For any $E' \in \mathcal{E}_1$, we have $i \notin E - E'$, so $T_{E-E'}(\mathbf{I}) = T_{E-E'}(\mathbf{I}')$ by Lemma 3.2.1. Thus,

$$\sum_{E' \in \mathcal{E}_1} \left(T_{E-E'}(\mathbf{I}) \prod_{j \in E'} s_j \right) = \sum_{E' \in \mathcal{E}_1} \left(T_{E-E'}(\mathbf{I}') \prod_{j \in E'} s_j \right). \tag{3.22}$$

For any $E' \in \mathcal{E}_2$, we have $i \in E - E'$. By Lemma 3.2.3, we have $T_{E-E'}(\mathbf{I}) \leq T_{E-E'}(\mathbf{I}') + T_{E-E'-\{i\}}(\mathbf{I}')$. Therefore,

$$\begin{aligned}
&\sum_{E' \in \mathcal{E}_2} \left(T_{E-E'}(\mathbf{I}) \prod_{j \in E'} s_j \right) \\
\leq &\sum_{E' \in \mathcal{E}_2} \left((T_{E-E'}(\mathbf{I}') + T_{E-E'-\{i\}}(\mathbf{I}')) \prod_{j \in E'} s_j \right) \\
= &\sum_{E' \in \mathcal{E}_2} \left(T_{E-E'}(\mathbf{I}') \prod_{j \in E'} s_j \right) + \sum_{E' \in \mathcal{E}_2} \left(T_{E-(E' \cup \{i\})}(\mathbf{I}') \prod_{j \in E'} s_j \right) \\
= &\sum_{E' \in \mathcal{E}_2} \left(T_{E-E'}(\mathbf{I}') \prod_{j \in E'} s_j \right) + \sum_{E' \in \mathcal{E}_1} \left(T_{E-E'}(\mathbf{I}') \prod_{j \in E'-\{i\}} s_j \right).
\end{aligned} \tag{3.23}$$

Note that the last step makes use of the one-to-one correspondence between $\mathcal{E}_1$ and $\mathcal{E}_2$.

Finally, based on (3.22) and (3.23), we have

$$\begin{aligned}
&\sum_{E'\subseteq E}\left(T_{E-E'}(\mathbf{I})\prod_{j\in E'}s_j\right)\\
\leq&\sum_{E'\in\mathcal{E}_1}\left(T_{E-E'}(\mathbf{I}')(\prod_{j\in E'}s_j+\prod_{j\in E'-\{i\}}s_j)\right)+\sum_{E'\in\mathcal{E}_2}\left(T_{E-E'}(\mathbf{I}')\prod_{j\in E'}s_j\right)\\
=&\sum_{E'\in\mathcal{E}_1}\left(T_{E-E'}(\mathbf{I}')(s_i+1)\prod_{j\in E'-\{i\}}s_j\right)+\sum_{E'\in\mathcal{E}_2}\left(T_{E-E'}(\mathbf{I}')\prod_{j\in E'}s_j\right)\\
=&\sum_{E'\in\mathcal{E}_1}\left(T_{E-E'}(\mathbf{I}')\prod_{j\in E'}s'_j\right)+\sum_{E'\in\mathcal{E}_2}\left(T_{E-E'}(\mathbf{I}')\prod_{j\in E'}s'_j\right)\\
=&\sum_{E'\subseteq E}\left(T_{E-E'}(\mathbf{I}')\prod_{j\in E'}s'_j\right),
\end{aligned}$$

□

In order to handle self-joins, we need to extend Lemma 3.2.9 to the case where multiple relations may differ.

Lemma 3.2.10. *Given any self-join-free CQ, any* $E\subseteq[n]$*, any* $B\subseteq[n]$*, two instances* $\mathbf{I},\mathbf{I}'$ *that differ by one tuple in every* $R_i(\mathbf{x}_i), i\in B$*, and two distance vectors* $\mathbf{s}=(s_1,\dots,s_n)$ *and* $\mathbf{s}'=(s'_1,\dots,s'_n)$ *such that*

$$s'_i=\begin{cases}s_i+1, & i\in B;\\ s_i, & i\notin B,\end{cases}$$

we have

$$\sum_{E'\subseteq E}\left(T_{E-E'}(\mathbf{I})\prod_{j\in E'}s_j\right)\leq\sum_{E'\subseteq E}\left(T_{E-E'}(\mathbf{I}')\prod_{j\in E'}s'_j\right).$$

Proof. Given $B\subseteq[n]$, two instances $\mathbf{I},\mathbf{I}'$ differing by one tuple in all $R_i(\mathbf{x}_i), i\in B$, we define $\mathbf{I}^0,\mathbf{I}^1,\dots,\mathbf{I}^n$ and $\mathbf{s}^0,\mathbf{s}^1,\dots,\mathbf{s}^n$: for $\ell\in[0,n]$,

$$I^\ell_j(\mathbf{x}_j)=\begin{cases}I'_j(\mathbf{x}_j) & j\in[\ell]\\ I_j(\mathbf{x}_j) & \textit{otherwise}\end{cases}$$

and

$$s^\ell_j=\begin{cases}s'_j & j\in[\ell]\\ s_j & \textit{otherwise}\end{cases}.$$

It is trivial to see $\mathbf{I}^0 = \mathbf{I}$, $\mathbf{I}^n = \mathbf{I}'$, $\mathbf{s} = \mathbf{s}^0$ and $\mathbf{s}' = \mathbf{s}^n$. Recall for $j \in [n]-B$, $I_j(\mathbf{x}_j) = I'_j(\mathbf{x}_j)$, $s_j = s'_j$ and for $j \in B$, $d(I_j(\mathbf{x}_j), I'_j(\mathbf{x}_j)) = 1$ and $s'_j = s_j + 1$.

Therefore, for $\ell \notin B$, $\mathbf{I}^{\ell-1} = \mathbf{I}^\ell$ and $\mathbf{s}^{\ell-1} = \mathbf{s}^\ell$. That is,

$$\sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}^{\ell-1}) \prod_{j \in E'} s_j^{\ell-1} \right) = \sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}^{\ell}) \prod_{j \in E'} s_j^{\ell} \right).$$

For $\ell \in B$, $\mathbf{I}^{\ell-1}, \mathbf{I}^\ell$ differ by one tuple in $R_\ell(\mathbf{x}_\ell)$ and $\mathbf{s}^{\ell-1}, \mathbf{s}^\ell$ only differ by $s^\ell = s^{\ell-1} + 1$. Based on Lemma 3.2.9, we have

$$\sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}^{\ell-1}) \prod_{j \in E'} s_j^{\ell-1} \right) \leq \sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}^{\ell}) \prod_{j \in E'} s_j^{\ell} \right).$$

Above all, for any $\ell \in [n]$,

$$\sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}^{\ell-1}) \prod_{j \in E'} s_j^{\ell-1} \right) \leq \sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}^{\ell}) \prod_{j \in E'} s_j^{\ell} \right).$$

And finally,

$$\begin{aligned} \sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}) \prod_{j \in E'} s_j \right) &= \sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}^0) \prod_{j \in E'} s_j^0 \right) \\ &\leq \sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}^n) \prod_{j \in E'} s_j^n \right) \\ &= \sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}') \prod_{j \in E'} s'_j \right) \end{aligned}$$

□

With Lemma 3.2.10, we can show the smoothness property of $\widehat{\mathrm{LS}}_Q^{(k)}(\cdot)$ for CQs.

Theorem 3.2.2. *For any CQ and any* $\mathbf{I}, \mathbf{I}'$ *such that* $d(\mathbf{I}, \mathbf{I}') = 1$, $\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) \leq \widehat{\mathrm{LS}}_Q^{(k+1)}(\mathbf{I}')$ *for any* $k \geq 0$.

Proof. Based on (3.19), we have

$$\begin{cases} \widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) = \max_{\mathbf{s} \in \mathbf{S}^k} \max_{i \in P^m} \sum_{E \subseteq D_i, E \neq \emptyset} \hat{T}_{[n]-E,\mathbf{s}}(\mathbf{I}); \\ \widehat{\mathrm{LS}}_Q^{(k+1)}(\mathbf{I}') = \max_{\mathbf{s} \in \mathbf{S}^{k+1}} \max_{i \in P^m} \sum_{E \subseteq D_i, E \neq \emptyset} \hat{T}_{[n]-E,\mathbf{s}}(\mathbf{I}'). \end{cases}$$

Recall formulation of $\hat{E}_{E,\mathbf{s}}(\mathbf{I})$ in (3.20). It is sufficient to show, for any $\mathbf{s} \in \mathbf{S}^k$, we can find a $\mathbf{s}' \in \mathbf{S}^{k+1}$ such that for any $E \subseteq [n]$

$$\sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}) \prod_{j \in E'} s_j \right) \leq \sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}') \prod_{j \in E'} s'_j \right) \tag{3.24}$$

Assume for $\mathbf{I}, \mathbf{I}'$ differs by tuple in relation $R_{l_i}, i \in P^m$. That is, for all $j \in D_i, d(I_j(\mathbf{x}_j), I'_j(\mathbf{x}_j)) = 1$. Given $\mathbf{s} \in \mathbf{S}^k$, we construct a $\mathbf{s}'$ such that

$$s'_j = \begin{cases} s_j + 1 & j \in D_i \\ s_j & otherwise \end{cases}.$$

It is clear $\mathbf{s}' \in \mathbf{S}^{k+1}$ and we show $\mathbf{s}'$ meets the requirement in (3.24).

Recall that a query with self-joins can be considered as a self-join-free query on the logical instance: Q as a self-join-free query on the logical instance and any $\mathbf{I}, \mathbf{I}'$ with $d(\mathbf{I}, \mathbf{I}') = 1, d(I_{l_i}, I'_{l_i}) = 1$ correspond to $\{I_j(\mathbf{x}_j)\}_j, \{I'_j(\mathbf{x}_j)\}_i$ with $d(I_j(\mathbf{x}_j), I_j(\mathbf{x}_j)) = 1$ for all $j \in D_i$. Besides, recall, $\mathbf{s}$ and $\mathbf{s}'$ only differ by $s'_j = s_j + 1$ for all $j \in D_i$.

Finally, we derive,

$$\begin{aligned} \sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}) \prod_{j \in E'} s_j \right) &= \sum_{E' \subseteq E} \left(T_{E-E'}(\{I_j(\mathbf{x}_j)\}_j) \prod_{j \in E'} s_j \right) \\ &\leq \sum_{E' \subseteq E} \left(T_{E-E'}(\{I'_j(\mathbf{x}_j)\}_j) \prod_{j \in E'} s'_j \right) \\ &= \sum_{E' \subseteq E} \left(T_{E-E'}(\mathbf{I}') \prod_{j \in E'} s'_j \right), \end{aligned}$$

where the inequality follows from Lemma 3.2.10. □

3.2.6 Computing $\mathrm{RS}^{\beta}_{Q}(\cdot)$

Recall that for any given k, $\widehat{\mathrm{LS}}^{(k)}_Q(\mathbf{I})$ can be computed in polynomial time since each $T_{\bar{E}}(\mathbf{I})$ is an AJAR/FAQ query [47, 4]. The last missing piece is to bound the range of k that one has to consider when computing $\mathrm{RS}^{\beta}_Q(\mathbf{I})$ using (3.21). The following lemma implies that we only need to consider $k = 0, \ldots, \hat{k} = O(1)$ when computing $\mathrm{RS}^{\beta}_Q(\mathbf{I})$.

Lemma 3.2.11. *For any* $k \geq \hat{k} = \frac{m_p}{1-\exp(-\beta/\max_{i\in[m]} n_i)}$,

$$e^{-\beta k}\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) \leq e^{-\beta(k-1)}\widehat{\mathrm{LS}}_Q^{(k-1)}(\mathbf{I}).$$

Proof. We expand $e^{-\beta k}\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I})$ with (3.19), (3.20)

$$e^{-\beta k}\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) = \max_{\mathbf{s}\in\mathbf{S}^k}\max_{i\in P^m} \sum_{E\subseteq D_i, E\neq\emptyset}\sum_{E'\subseteq[n]-E}\left(e^{-\beta k}T_{[n]-E-E'}\prod_{j\in E'} s_j\right).$$

Now, we show when $k \geq \frac{m_p}{1-\exp(-\beta/\max_{i\in[m]} n_i)}$, for any $\mathbf{s}\in\mathbf{S}^k$ there is an $\mathbf{s}'\in\mathbf{S}^{k-1}$ such that for any $i \in P^m$,

$$\sum_{E\subseteq D_i, E\neq\emptyset}\sum_{E'\subseteq[n]-E}\left(e^{-\beta k}T_{[n]-E-E'}\prod_{j\in E'} s_j\right) \leq \sum_{E\subseteq D_i, E\neq\emptyset}\sum_{E'\subseteq[n]-E}\left(e^{-\beta(k-1)}T_{[n]-E-E'}\prod_{j\in E'} s'_j\right). \tag{3.25}$$

Since $\mathbf{s}\in\mathbf{S}^k, k \geq \frac{m_p}{1-\exp(-\beta/\max_{i\in[m]} n_i)}$, there must exist one $i' \in [m]$, such that $s_{l_{i'}} \geq \frac{1}{1-\exp(-\beta/\max_{i\in[m]} n_i)}$. Then, we define the $\mathbf{s}'$ as

$$s'_j = \begin{cases} s_j - 1, & j \in D_{i'}; \\ s_j, & \textit{otherwise.}\end{cases}$$

Then, to prove (3.25), it is sufficient to show for any $E' \subseteq E \subset [n]$,

$$e^{-\beta k}T_{E-E'}(\mathbf{I})\prod_{j\in E'} s_j \leq e^{-\beta(k-1)}T_{E-E'}(\mathbf{I})\prod_{j\in E'} s'_j. \tag{3.26}$$

(3.26) can be simplified to

$$\prod_{j\in E'} s_j \leq e^{\beta}\prod_{j\in E'} s'_j,$$

which can be further reduced to

$$\prod_{j\in E'\cap D_{i'}} s_j \leq e^{\beta}\prod_{j\in E'\cap D_{i'}} s'_j,$$

since $s_j = s'_j$ for $j \notin D_{i'}$.

Since

$$\prod_{j\in E'\cap D_{i'}} \frac{s_j}{s'_j} \leq \left(\frac{1}{1-\frac{1}{k}}\right)^{\max_{i\in[m]} n_i} \leq (e^{\beta/\max_{i\in[m]} n_i})^{\max_{i\in[m]} n_i} = e^{\beta},$$

we prove the claim. □

Remark In actual implementation, we first compute $T_{\bar{E}}(\mathbf{I})$ for all $E \subseteq D_i, E \neq \emptyset$. After that, it only takes $O(1)$ time to compute $\mathrm{RS}_Q^\beta(\mathbf{I})$ using formulas (3.19), (3.20), and (3.21). Thus, the concrete computational complexity of $\mathrm{RS}_Q^\beta(\cdot)$ for a CQ Q is $O(N^{w_{\max}})$, where $w_{\max}$ is the maximum AJAR/FAQ width [47, 4] of the residual queries of Q.

3.3 Neighborhood Optimality

In this section we prove Theorem 1.1.1. This is done in three steps: We first derive a sufficient condition for $\mathrm{SS}_Q^\beta(\cdot)$ to be an r-neighborhood lower bound. Next, we show that this condition holds for full CQs with $r = O(1)$. Finally, we show that $\mathrm{RS}_Q^\beta(\cdot)$ is at most a constant factor larger than $\mathrm{SS}_Q^\beta(\cdot)$,

3.3.1 General Neighborhood Lower Bounds

We first develop two general neighborhood lower bounds that hold for arbitrary queries (not necessarily CQs), one based on $\mathrm{LS}_Q^{(k)}(\cdot)$ while the other based on $\mathrm{SS}_Q^\beta(\cdot)$. These lower bounds hold for an arbitrary query Q with vectored outputs. We start with an observation from [77]:

Lemma 3.3.1 ([77]). *For any ε-DP mechanism $\mathcal{M}'(\cdot)$ and any instance $\mathbf{I}$, there exists an $\mathbf{I}'$ with $d(\mathbf{I}, \mathbf{I}') \leq 1$ such that*

$$\Pr\left[|\mathcal{M}'(\mathbf{I}') - Q(\mathbf{I}')| \geq \frac{\mathrm{LS}_Q(\mathbf{I})}{2}\right] \geq \frac{1}{1+e^\varepsilon}.$$

This implies that $\mathrm{LS}_Q(\cdot)$ is an 1-neighborhood lower bound, i.e.,

$$\max_{\mathbf{I}': d(\mathbf{I},\mathbf{I}') \leq 1} \mathrm{Err}(\mathcal{M}', \mathbf{I}') \geq \frac{1}{2\sqrt{1+e^\varepsilon}} \cdot \mathrm{LS}_Q(\mathbf{I}), \tag{3.27}$$

for any $\mathbf{I}$ and any $\mathcal{M}'$. We generalize this result, showing that $\mathrm{LS}_Q^{(r-1)}(\cdot)$ is an r-neighborhood lower bound:

Lemma 3.3.2. *For any $\mathbf{I}$, any ε-DP mechanism $\mathcal{M}'$, and any $r \geq 1$,*

$$\max_{\mathbf{I}': d(\mathbf{I},\mathbf{I}') \leq r} \mathrm{Err}(\mathcal{M}', \mathbf{I}') \geq \frac{1}{2\sqrt{1+e^\varepsilon}} \cdot \mathrm{LS}_Q^{(r-1)}(\mathbf{I}). \tag{3.28}$$

Proof. For any $\mathbf{I}$, we need show that there exists an $\mathbf{I}'$ in its r-neighborhood such that

$$\mathrm{Err}(\mathcal{M}', \mathbf{I}') \geq \frac{1}{2\sqrt{1+e^{\varepsilon}}} \cdot \mathrm{LS}_Q^{(r-1)}(\mathbf{I}).$$

For any $r \geq 1$, let

$$\mathbf{I}^* = \underset{\mathbf{I}', d(\mathbf{I}, \mathbf{I}') \leq r-1}{\arg\max} \mathrm{LS}_Q(\mathbf{I}').$$

By Lemma 3.3.1, for any ε-DP mechanism $\mathcal{M}'(\cdot)$, there exists an $\mathbf{I}'$ with $d(\mathbf{I}^*, \mathbf{I}') \leq 1$ such that

$$\mathsf{E}\left[(\mathcal{M}'(\mathbf{I}') - Q(\mathbf{I}'))^2\right] \geq \frac{1}{1+e^{\varepsilon}} \cdot \left(\frac{\mathrm{LS}_Q(\mathbf{I}^*)}{2}\right)^2,$$

so

$$\mathrm{Err}(\mathcal{M}', \mathbf{I}') \geq \frac{1}{2\sqrt{1+e^{\varepsilon}}} \cdot \mathrm{LS}_Q(\mathbf{I}^*) = \frac{1}{2\sqrt{1+e^{\varepsilon}}} \cdot \mathrm{LS}_Q^{(r-1)}(\mathbf{I}).$$

Since $d(\mathbf{I}, \mathbf{I}^*) \leq r-1$ and $d(\mathbf{I}^*, \mathbf{I}') \leq 1$, we have $d(\mathbf{I}, \mathbf{I}') \leq r$, i.e., $\mathbf{I}'$ is in the r-neighborhood of $\mathbf{I}$, as desired. □

Note that (3.27) is the special case of (3.28) with $r = 1$.

Remark Previously, Asi and Duchi [10] also derive a neighborhood lower bound. Here, we show that our lower bound is always no worse than theirs for $\varepsilon = O(1)$, while can be polynomially better for certain queries. Furthermore, their lower bound requires a technical condition on Q while our lower bound holds for an arbitrary Q.

The r-neighborhood lower bound by Asi and Duchi (Lemma C.1 in [10]), when specialized to the 1-dimensional case, is as follows. Given a query Q, for any k and any instance $\mathbf{I}$, define

$$\omega(\mathbf{I}, r) := \max_{\mathbf{I}', d(\mathbf{I}, \mathbf{I}') \leq k} |Q(\mathbf{I}) - Q(\mathbf{I}')|.$$

If $\{Q(\mathbf{I}') : d(\mathbf{I}, \mathbf{I}') \leq k\}$ contains an interval of length $c \cdot \omega(\mathbf{I}, k)$ for some $c > 0$ and all $k \leq r$, then

$$\max_{\mathbf{I}': d(\mathbf{I}, \mathbf{I}') \leq r} \mathrm{Err}(\mathcal{M}', \mathbf{I}') \geq \frac{c}{16} \max_{k \leq r} e^{-\varepsilon k} \omega(\mathbf{I}, k). \tag{3.29}$$

Our lower bound, which does not require any condition on Q, is

$$\max_{\mathbf{I}': d(\mathbf{I}, \mathbf{I}') \leq r} \mathrm{Err}(\mathcal{M}', \mathbf{I}') \geq \frac{1}{2\sqrt{1+e^{\varepsilon}}} \cdot \mathrm{LS}_Q^{(r-1)}(\mathbf{I}). \tag{3.30}$$

Next we compare (3.29) and (3.30). By the definition of $\mathrm{LS}_Q^{(k)}(\mathbf{I})$ and $\omega(\mathbf{I},k)$, we have

$$\omega(\mathbf{I},k) \le \sum_{0\le k'\le k-1} \mathrm{LS}_Q^{(k')}(\mathbf{I}) \le k\cdot \mathrm{LS}_Q^{(k-1)}(\mathbf{I}).$$

Then,

$$(3.29) \le \frac{c}{16}\max_{k\le r}(e^{-\varepsilon k}\cdot k\cdot \mathrm{LS}_Q^{(k-1)}(\mathbf{I})) \le \frac{c}{16}\cdot \max_{k\le r}(ke^{-\varepsilon k})\cdot \mathrm{LS}_Q^{(r-1)}(\mathbf{I}),$$

which is asymptotically upper bounded by (3.30) as long as

$$ke^{-\varepsilon k} \le O\left(\frac{1}{\sqrt{1+e^{\varepsilon}}}\right),$$

which is true when $\varepsilon = O(1)$.

On the other hand, the gap between (3.29) and (3.30) can be poly(N). Consider the Median query with a constant ε. Let $\mathbf{I}$ consist of $\log N$ copies of 0.5, while the remaining entries are half 0 and half 1. For any $r \ge \log N$, our lower bound (3.30) is $\mathrm{LS}_Q^{(r)}(\mathbf{I}) = 0.5$, while their lower bound (3.29) is

$$\max_{k\le r} e^{-\varepsilon k}\omega(\mathbf{I},k) \le e^{-\varepsilon \log N}\cdot 0.5 = 1/N^{\Omega(1)}.$$

Nevertheless, Lemma C.1 in [10] yields better lower bounds for high-dimensional queries.

To show that $\mathrm{SS}_Q^{\beta}(\cdot)$ is an r-neighborhood lower bound, we need a condition on $\mathrm{LS}_Q^{(k)}(\cdot)$, that they do not grow more than exponentially quickly when $k \ge r$.

Lemma 3.3.3. *Given any $\varepsilon, \beta > 0$ and any instance $\mathbf{I}$, if for some $r \ge 1$ (possibly depending on β and $\mathbf{I}$),*

$$\mathrm{LS}_Q^{(k)}(\mathbf{I}) \le e^{\beta k}\mathrm{LS}_Q^{(r-1)}(\mathbf{I}), \tag{3.31}$$

for any $k \ge r$, for any ε-DP mechanism $\mathcal{M}'$,

$$\max_{\mathbf{I}':d(\mathbf{I},\mathbf{I}')\le r} \mathrm{Err}(\mathcal{M}',\mathbf{I}') \ge \frac{1}{2\sqrt{1+e^{\varepsilon}}}\cdot \mathrm{SS}_Q^{\beta}(\mathbf{I}).$$

Proof. Consider any $\mathbf{I}$. When $k < r$,

$$e^{-\beta k}\mathrm{LS}_Q^{(k)}(\mathbf{I}) \le \mathrm{LS}_Q^{(k)}(\mathbf{I}) \le \mathrm{LS}_Q^{(r-1)}(\mathbf{I}),$$

where the second inequality follows from the definition of $\mathrm{LS}_Q^{(k)}(\mathbf{I})$ in (3.2).

When $k \geq r$, from the premise of the lemma, we have

$$e^{-\beta k}\mathrm{LS}_Q^{(k)}(\mathbf{I}) \leq e^{-\beta k}e^{\beta k}\mathrm{LS}_Q^{(r-1)}(\mathbf{I}) = \mathrm{LS}_Q^{(r-1)}(\mathbf{I}).$$

Therefore, for any $k \geq 0$, $e^{-\beta k}\mathrm{LS}_Q^{(k)}(\mathbf{I}) \leq \mathrm{LS}_Q^{(r-1)}(\mathbf{I})$. So

$$\mathrm{SS}_Q^{\beta}(\mathbf{I}) = \max_{k\geq 0} e^{-\beta k}\mathrm{LS}_Q^{(k)}(\mathbf{I}) \leq \mathrm{LS}_Q^{(r-1)}(\mathbf{I}). \tag{3.32}$$

Finally, combine (3.32) and Lemma 3.3.2, we prove the lemma. □

Remark 1 Recall from Section 3.1.3 that β and ε are just a constant-factor apart, so β is also a constant if ε is considered a constant.

Remark 2 The restriction of the growth rate is very mild, except that it also forbids $\mathrm{LS}_Q^{(k)}(\cdot)$ to go from zero to nonzero. This is why we impose this restriction only for $k \geq r$. For certain problems like MEDIAN, this requires a large r, which is actually unavoidable since a large flat neighborhood will rule out r-neighborhood optimal mechanisms for small r anyway, as we argued in Section 1.1.2.

Before considering CQs, as a warm-up, we apply Lemma 3.3.3 to the triangle counting problem. Here, the instance $\mathbf{I}$ is a simple graph (i.e., no self-loops and multi-edges), and the query Q returns the number of triangles in $\mathbf{I}$.

Lemma 3.3.4. *For the triangle counting problem, the condition in Lemma 3.3.3 holds with* $r = \max\left\{3, \left\lceil 4\frac{\ln(1/\beta)}{\beta}\right\rceil\right\}$.

Proof. Let V be the domain of vertices. For $i, j \in V$, let $x_{i,j}(\mathbf{I}) = 1$ if there is an edge between vertex i and j in $\mathbf{I}$, and 0 otherwise. Then the number of common neighbors of vertices i, j is

$$a_{i,j}(\mathbf{I}) = \sum_{k\in V} x_{i,k}(\mathbf{I}) \cdot x_{j,k}(\mathbf{I}).$$

An exact formula for $\mathrm{LS}_Q^{(k)}(\mathbf{I})$, hence $\mathrm{SS}_Q^{\beta}(\mathbf{I})$, is given in [67] for the triangle counting problem, but we only need the following upper and lower bound on $\mathrm{LS}_Q^{(k)}$:

$$\max_{i,j\in V} a_{i,j}(\mathbf{I}) + \frac{k-1}{2} \leq \mathrm{LS}_Q^{(k)}(\mathbf{I}) \leq \max_{i,j\in V} a_{i,j}(\mathbf{I}) + k.$$

We will show that by setting $r = \max\left\{3, \left\lceil 4\frac{\ln(1/\beta)}{\beta}\right\rceil\right\}$, (3.31) holds for any $\mathbf{I}$ and $k \geq r$. Thus $\mathrm{SS}_Q^\beta(\cdot)$ is $O(1)$-neighborhood optimal.

For $k \geq r$, we have

$$\begin{aligned}
\mathrm{LS}_Q^{(r-1)}(\mathbf{I})e^{\beta k} \geq& \max_{i,j\in V} a_{i,j}(\mathbf{I})e^{\beta k} + \frac{r-1}{2}e^{\beta k}\\
\geq& \max_{i,j\in V} a_{i,j}(\mathbf{I}) + e^{\beta k}\\
\geq& \max_{i,j\in V} a_{i,j}(\mathbf{I}) + k\\
\geq& \mathrm{LS}_Q^{(k)}(\mathbf{I}).
\end{aligned}$$

The first equality is by the lower bound on $\mathrm{LS}_Q^{(k)}(\mathbf{I})$; the second inequality is because $e^{\beta k} \geq 1$ and $r \geq 3$; the third inequality $e^{\beta k} \geq k$ follows from $k \geq r \geq 4\frac{\ln(1/\beta)}{\beta}$ and some simple algebra; the last inequality is by the upper bound on $\mathrm{LS}_Q^{(k)}(\mathbf{I})$. □

Note that the r needed in the lemma above is independent of $\mathbf{I}$. Thus, $\mathrm{SS}_Q^\beta(\cdot)$ is an $O(1)$-neighborhood lower bound for the triangle counting problem, i.e., the previous SS-based DP-mechanism for triangle counting [67] is $O(1)$-neighborhood optimal. This is the first optimality guarantee for triangle counting under DP, while our main optimality result is a vast generalization of this.

3.3.2 Neighborhood Lower Bound for CQs

To show that $\mathrm{SS}_Q^\beta(\cdot)$ is an $O(1)$-neighborhood lower bound for CQs, we need to show that the condition in Lemma 3.3.3 holds with some constant r. This requires an upper bound on $\mathrm{LS}_Q^{(k)}(\cdot)$, as well as a lower bound on $\mathrm{LS}_Q^{(r-1)}(\cdot)$. For the upper bound on $\mathrm{LS}_Q^{(k)}(\cdot)$, we use the $\widehat{\mathrm{LS}}_Q^{(k)}(\cdot)$ developed in Section 3.2.5. For the lower bound, we first consider the case $r = n_P$. Recall that $n_P = |P^n|$ is the number of private logical relations.

Lemma 3.3.5. *For any CQ, any instance* $\mathbf{I}$*, and any* $E \subseteq P^n$*,* $E \neq \emptyset$*, we have* $\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I}) \geq T_{\bar{E}}(\mathbf{I})$*.*

Proof. We will construct an $\mathbf{I}'$ from $\mathbf{I}$ such that $d(\mathbf{I},\mathbf{I}') \leq n_P - 1$ and $\mathrm{LS}_Q(\mathbf{I}') \geq T_{\bar{E}}(\mathbf{I})$. Recall $t_{\bar{E}}(\mathbf{I})$ is the witness of $T_{\bar{E}}(\mathbf{I})$. Then, we write $T_{\bar{E}}(\mathbf{I})$ as

$$T_{\bar{E}}(\mathbf{I}) = |\bowtie_{i\in\bar{E}} (I_i(\mathbf{x}_i) \ltimes t_{\bar{E}}(\mathbf{I}))|.$$

We construct the $\mathbf{I}'$ as follows. First, we fix j_E and i_E such that $j_E \in E$ and $j_E \in D_{i_E}$. Then, we find a tuple $t' \in \mathbf{dom}(\cup_{i\in E}\mathbf{x}_i)$ such that $\pi_{\partial Q_E} t' = t_{\bar{E}}(\mathbf{I})$. Next, for each $j \in E - \{j_E\}$, we add $\pi_{\mathbf{x}_j} t'$ to I_j unless it already exists in I_j. Here, we at most add $|E| - 1$ tuples. Since $E \subseteq P^n$, $d(\mathbf{I}, \mathbf{I}') \le n_P - 1$ and $\mathrm{LS}_Q(\mathbf{I}') \le \mathrm{LS}_Q^{(n_P-1)}(\mathbf{I})$. Besides, we can ensure for each $j \in E - \{j_E\}$, $I'_j(\mathbf{x}_j)$ contains $\pi_{\mathbf{x}_j} t'$.

Therefore, it suffices to show $\mathrm{LS}_Q(\mathbf{I}') \ge T_{\bar{E}}(\mathbf{I})$. To do that, we will show by flipping the tuple $t_{j_E} = \pi_{\mathbf{x}_{j_E}} t'$ in I'_{j_E}, $Q(\mathbf{I}')$ will change by at least $T_{\bar{E}}(\mathbf{I})$. Suppose $t_{j_E} \notin I'_{j_E}$. The number of tuples changes by

$$
\begin{aligned}
&|(\Join_{j\in D_{i_E}} (I'_j(\mathbf{x}_j) \cup t_{j_E})) \Join (\Join_{j\in[n]-D_{i_E}} I'_j(\mathbf{x}_j)) \\
&\quad - (\Join_{j\in[n]} I'_j(\mathbf{x}_j))| \\
\ge &|(I'_{j_E}(\mathbf{x}_j) \cup t_{j_E}) \Join (\Join_{j\in[n]-j_E} I'_j(\mathbf{x}_j)) \\
&\quad - (\Join_{j\in[n]} I'_j(\mathbf{x}_j))| \\
= &|t_{j_E} \Join (\Join_{j\in[n]-j_E} I'_j(\mathbf{x}_j))| \\
= &|t_{j_E} \Join (\Join_{j\in E-j_E} I'_j(\mathbf{x}_j)) \Join (\Join_{[n]-E} I'_j(\mathbf{x}_j))| \\
\ge &|\pi_{\mathbf{x}_{j_E}} t' \Join (\Join_{j\in E-j_E} \pi_{\mathbf{x}_j} t') \Join (\Join_{j\in\bar{E}} I_j(\mathbf{x}_j))| \qquad (3.33)\\
= &|t' \Join (\Join_{\bar{E}} I_j(\mathbf{x}_j))| \\
= &| \Join_{i\in\bar{E}} (I_i(\mathbf{x}_i) \ltimes t_{\bar{E}}(\mathbf{I}))| = T_{\bar{E}}(\mathbf{I})
\end{aligned}
$$

The (3.33) is because for each $j \in E - \{j_E\}$, $I'_j(\mathbf{x}_j)$ contains $\pi_{\mathbf{x}_j} t'$ and for $j \in [n] - E$, $I_j(\mathbf{x}_j) = I'_j(\mathbf{x}_j)$. For the case $t_{j_E} \in I'_i$, we can draw a similar conclusion. □

Next, recall from equations (3.19) and (3.20) that $\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I})$ is also defined in terms of the $T_{\bar{E}}(\mathbf{I})$'s. Together with Lemma 3.3.5, this allows us to build a connection between $\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I})$ and $\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I})$:

Lemma 3.3.6. *For any CQ Q, any instance $\mathbf{I}$, and any $k \ge 1$, we have*

$$\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) \le (4k)^{n_P-1} \mathrm{LS}_Q^{(n_P-1)}(\mathbf{I}).$$

Proof. Based on (3.19) and (3.20), we have

$$
\begin{aligned}
\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) &= \max_{\mathbf{s}\in\mathbf{S}^k} \max_{i\in P^m} \sum_{E\subseteq D_i, E\neq\emptyset} \sum_{E'\subseteq[n]-E} \left(T_{[n]-E-E'}(\mathbf{I}) \prod_{j\in E'} s_j \right) \\
&= \max_{\mathbf{s}\in\mathbf{S}^k} \max_{i\in P^m} \sum_{E\subseteq D_i, E\neq\emptyset} \sum_{E'\subseteq P^n-E} \left(T_{[n]-E-E'}(\mathbf{I}) \prod_{j\in E'} s_j \right) \qquad (3.34)
\end{aligned}
$$

The (3.34) is because for any $\mathbf{s}$, $s_j = 0$ for any $j \notin P^n$.

For above E and E', $E \cup E' \subseteq P^n$. Based on Lemma 3.3.5, we have $T_{[n]-E-E'}(\mathbf{I}) \leq \mathrm{LS}_Q^{(n_P-1)}(\mathbf{I})$. Besides, for any $\mathbf{s} \in \mathbf{S}^k$, since $E' \subseteq P^n - E, E \neq \emptyset$, we have $\prod_{j \in E'} s_j \leq k^{n_P-1}$. Plug these into (3.34),

$$\begin{aligned}
\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) \leq & \max_{\mathbf{s}\in\mathbf{S}^k} \max_{i \in P^m} \sum_{E \subseteq D_i, E \neq \emptyset} \sum_{E' \subseteq P^n - E} \left(\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I}) k^{n_P-1}\right) \\
\leq & \max_{\mathbf{s}\in\mathbf{S}^k} \max_{i \in P^m} \sum_{E \subseteq D_i, E \neq \emptyset} \left(2^{n_P-1}\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I}) k^{n_P-1}\right) \\
\leq & \max_{\mathbf{s}\in\mathbf{S}^{(k)}} \max_{i \in [m]} (2^{n_P-1} 2^{n_P-1}\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I}) k^{n_P-1}) \\
= & (4k)^{n_P-1}\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I})
\end{aligned}$$

□

Lemma 3.3.6 almost meets the condition of Lemma 3.3.3, except that $(4k)^{n_P-1}$ is not necessarily smaller than $e^{\beta k}$. But as the former is a polynomial while the latter is exponential, this is not an issue as long as k is larger than a constant.

Theorem 3.3.1. *For any CQ Q, any $\varepsilon, \beta > 0$, there exist a constant $r > 0$ (depending on Q, ε, β) such that for any $\mathbf{I}$ and any ε-DP mechanism $\mathcal{M}'$,*

$$\max_{\mathbf{I}': d(\mathbf{I}, \mathbf{I}') \leq r} \mathrm{Err}(\mathcal{M}', \mathbf{I}') \geq \frac{1}{2\sqrt{1+e^{\varepsilon}}} \cdot \mathrm{SS}_Q^{\beta}(\mathbf{I}).$$

Proof. We will show that

$$\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) \leq e^{\beta k}\mathrm{LS}_Q^{(r-1)}(\mathbf{I}), \tag{3.35}$$

for all $k \geq \max\left\{4, n_P, \left\lceil \frac{2(n_P-1)}{\beta} \ln \frac{2(n_P-1)}{\beta} \right\rceil\right\}$.

By Lemma 3.3.6, for $k \geq 4$, we have

$$\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) \leq (4k)^{n_P-1}\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I}) \leq k^{2(n_P-1)}\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I}).$$

By setting

$$r := \max\left\{4, n_P, \left\lceil \frac{2(n_P-1)}{\beta} \ln \frac{2(n_P-1)}{\beta} \right\rceil\right\},$$

We can show that (3.35) holds for any $k \geq r$:

$$\begin{aligned}\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) \leq & k^{2(n_P-1)}\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I}) \\ \leq & e^{\beta k}\mathrm{LS}_Q^{(r-1)}(\mathbf{I}).\end{aligned}$$

The second inequality follow because when $r \geq n_P$, we have $\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I}) \leq \mathrm{LS}_Q^{(r-1)}(\mathbf{I})$, and when $k \geq \frac{2(n_P-1)}{\beta} \ln \frac{2(n_P-1)}{\beta}$, we have $k^{2(n_P-1)} \leq e^{\beta k}$. □

3.3.3 Optimality of $\mathrm{RS}_Q^\beta(\cdot)$

To complete the proof of Theorem 1.1.1, we show that $\mathrm{RS}_Q^\beta(\cdot)$ is at most a constant-factor larger than $\mathrm{SS}_Q^\beta(\cdot)$.

Lemma 3.3.7. *For any CQ and any* $\mathbf{I}$, $\mathrm{RS}_Q^\beta(\mathbf{I}) \leq \left(\frac{4(n_P-1)}{\beta e^{1-\beta}}\right)^{n_P-1}\mathrm{SS}_Q^\beta(\mathbf{I})$.

Proof. Recall the definition of $\mathrm{SS}_Q^\beta(\mathbf{I})$ and $\mathrm{RS}_Q^\beta(\mathbf{I})$ in (3.3) and (3.21),

$$\mathrm{SS}_Q^\beta(\mathbf{I}) = \max_{k\geq 0} e^{-\beta k}\mathrm{LS}_Q^{(k)}(\mathbf{I}),$$

$$\mathrm{RS}_Q^\beta(\mathbf{I}) = \max_{k\geq 0} e^{-\beta k}\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}).$$

Let $k^* = \arg\max e^{-\beta k}\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I})$, and define the function

$$g(k) = e^{-\beta k}(4k)^{n_P-1}\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I}).$$

Setting its derivative to 0, we see that $g(k)$ maximizes at $k_{\max} = \frac{n_P-1}{\beta}$ (even allowing k to take fractional values). Thus

$$g(k) \leq g(\frac{n_P-1}{\beta}). \tag{3.36}$$

Therefore,

$$\begin{aligned}\mathrm{RS}_Q^\beta(\mathbf{I}) =& e^{-\beta k^*}\widehat{\mathrm{LS}}_Q^{(k^*)}(\mathbf{I}) \\ \le& e^{-\beta k^*}(4k^*)^{n_P-1}\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I}) \\ \le& e^{-(n_P-1)}\left(\frac{4(n_P-1)}{\beta}\right)^{n_P-1}\mathrm{LS}_Q^{(n_P-1)}(\mathbf{I}) \\ \le& \left(\frac{4(n_P-1)}{\beta e^{1-\beta}}\right)^{n_P-1}\max_{k\ge 0} e^{-\beta k}\mathrm{LS}_Q^{(k)}(\mathbf{I}) \\ =& \left(\frac{4(n_P-1)}{\beta e^{1-\beta}}\right)^{n_P-1}\mathrm{SS}_Q^\beta(\mathbf{I}).\end{aligned}$$

The first inequality follows from Lemma 3.3.6, and the second inequality is by (3.36). □

3.3.4 Elastic Sensitivity

Elastic sensitivity [48], denoted as $\mathrm{ES}_Q^\beta(\cdot)$, is the only other DP mechanism for CQs with self-joins. It is also a version of $\hat{SS}(\cdot)$, but defined using a different $\widehat{\mathrm{LS}}_Q^{(k)}(\cdot)$. For $i \in [n], \mathbf{x} \subseteq \mathbf{x}_i$, let $mf(\mathbf{x}, I_i(\mathbf{x}_i))$ be the *maximum frequency* in $I_i(\mathbf{x}_i)$ on attributes $\mathbf{x}$, i.e., $mf(\mathbf{x}, I_i(\mathbf{x}_i)) = \max_{t\in\mathbf{dom}(\mathbf{x})}|I_i(\mathbf{x}_i) \ltimes t|$. For $\mathrm{ES}_Q^\beta(\cdot)$, $\widehat{\mathrm{LS}}_Q^{(k)}(\cdot)$ is defined as a product of a number of maximum frequencies; please see [48] for the exact formula.

We give an example below showing that $\mathrm{ES}_Q^\beta(\mathbf{I})$ can be asymptotically larger than GS. This means that $\mathrm{ES}_Q^\beta(\cdot)$ is not even worst-case optimal (i.e., not N-neighborhood optimal).

Example 3.3.1. Consider the path-4 query:

$$Q = |\texttt{Edge}(x_1,x_2) \bowtie \texttt{Edge}(x_2,x_3) \bowtie \texttt{Edge}(x_3,x_4) \bowtie \texttt{Edge}(x_4,x_5)|.$$

We showed that $\mathrm{GS}_Q = O(N^2)$ in Example 3.2.3. Now consider the following instance $\mathbf{I}$ on `Edge` relation (assume the domain is $\mathbb{N}$):

$$\begin{aligned}\mathbf{I}(\texttt{Edge}) =& \{(0,1),(0,2),\dots,(0,\frac{N}{2}), \\ & (\frac{N}{2}+1,N+1),\dots,(N,N+1)\}.\end{aligned}$$

Note that $mf(x_i, E(x_i,x_{i+1})) = mf(x_{i+1}, E(x_i,x_{i+1})) = \frac{N}{2}$ for $i = 1,2,3,4$. By the formula in [48], we have $\widehat{\mathrm{LS}}_Q^{(0)}(\mathbf{I}) = 4(\frac{N}{2})^3 = \frac{N^3}{2}$, thus

$$\mathrm{ES}_Q^\beta(\mathbf{I}) = \max_{k\ge 0} e^{-\beta k}\widehat{\mathrm{LS}}_Q^{(k)}(\mathbf{I}) \ge \widehat{\mathrm{LS}}_Q^{(0)}(\mathbf{I}) = \Omega(N^3).$$

□

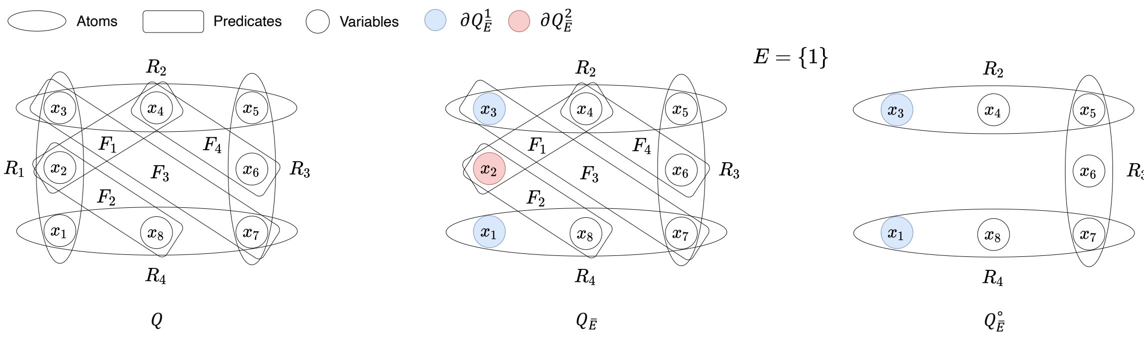


Figure 3.2: $Q_{\bar{E}}$, $Q^{\circ}_{\bar{E}}$, $\partial Q^1_{\bar{E}}$ and $\partial Q^2_{\bar{E}}$.

3.4 CQs with Predicates

A CQ with predicates (CQP) has the form

$$Q := |\sigma_{P_1(\mathbf{y}_1)\wedge\cdots\wedge P_\kappa(\mathbf{y}_\kappa)}(R_1(\mathbf{x}_1) \bowtie \cdots \bowtie R_n(\mathbf{x}_n))|,$$

where each $P_j : \mathbf{dom}(\mathbf{y}_j) \to \{\mathsf{True}, \mathsf{False}\}$ is a computable function for some $\mathbf{y}_j \subseteq var(Q) = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_n$. By a slight abuse of notation, we also use $P(\mathbf{y})$ to denote the (possibly infinite) relation $\{t \in \mathbf{dom}(\mathbf{y}) \mid P(t)\}$. This way, a CQP can be written as a normal CQ:

$$Q := |R_1(\mathbf{x}_1) \bowtie \cdots \bowtie R_n(\mathbf{x}_n) \bowtie P_1(\mathbf{y}_1) \bowtie \cdots \bowtie P_\kappa(\mathbf{y}_\kappa)|. \tag{3.37}$$

Note that the $P_j(\mathbf{y}_j)$'s are all public, since they only depend on the query and the domain, not on the instance.

The current approach to dealing with a CQP under DP [48, 57, 29] is to evaluate the CQP as given, but compute the sensitivity without considering the predicates. This yields a valid DP mechanism, but loses optimality. To see this, just consider an extreme case where a predicate always returns False. Then the query becomes a trivial query and the optimal (under any notion of optimality) mechanism is $\mathcal{M}(\cdot) \equiv 0$, i.e., $\mathrm{Err}(\mathcal{M}, \mathbf{I}) = 0$ for all $\mathbf{I}$, but the sensitivity of the query without the predicate must be nonzero.

In this section, we show how to extend $\mathrm{RS}^{\beta}_{Q}(\cdot)$ to CQPs while preserving its $O(1)$-neighborhood optimality. The idea is simple, we just consider a CQP as a CQ as defined in (3.37), so optimality immediately follows from Theorem 1.1.1. The issue, however, is how to compute $\mathrm{RS}^{\beta}_{Q}(\cdot)$ when some relations are infinite. In Section 3.4.1 we first give a general algorithm, which may take exponential time, to compute $\mathrm{RS}^{\beta}_{Q}(\cdot)$ for arbitrary predicates under the technical condition of Theorem 1.1.2; in Section 3.4.2 we give a polynomial-time algorithm for the case where all the predicates are inequalities or comparisons, which proves the second part of Theorem 1.1.2.

3.4.1 General Predicates

The first observation is that, when the $P(\mathbf{y}_j)$'s are arbitrary (but still computable), it is undecidable to check if a given CQP is a trivial query. Recall that if the query is trivial, the optimal DP mechanism $\mathcal{M}$ is deterministic and achieves $\mathrm{Err}(\mathcal{M}, \mathbf{I}) = 0$ for all $\mathbf{I}$; otherwise, the mechanism must be probabilistic. Since one cannot distinguish between the two cases, optimal (under any notion of optimality) DP mechanisms do not exist. For the undecidability result, just consider the simple CQP $Q_M = |R(x) \bowtie P_M(x)|$, where $P_M(x) = \mathsf{True}$ iff the Turing machine M terminates in less than x steps. Note that $P_M(x)$ is decidable. However, it is easy to see that $Q_M(\cdot) \equiv 0$ iff M does not halt.

However, the situation is not hopeless. Below we show how to compute $\mathrm{RS}^{\beta}_{Q}(\mathbf{I})$ for any CQP if for any $\mathbf{z} \subseteq var(Q)$, the satisfiability of $\varphi_1 \wedge \cdots \wedge \varphi_S$ is decidable, where each φ_i is $P_j(\mathbf{u}_i)$ for any j and $\mathbf{u}_i$ is $\mathbf{y}_j$ after replacing all variables not in $\mathbf{z}$ by any constants. This is a very mild condition; in fact, the entire literature on *constraint satisfaction problems (CSPs)* is devoted to designing efficient algorithms for determining the satisfiability of $\varphi_1 \wedge \cdots \wedge \varphi_S$ when the φ_i's take certain forms, and finding a satisfying valuation for $\mathbf{z}$, if one exists.

It suffices to show how to compute $T_{\bar{E}}(\mathbf{I})$ for any $E \subseteq P^n$. Since all the predicates correspond to public relations, the residual query has the form

$$Q_{\bar{E}} = (\bowtie_{i \in \bar{E}} R_i(\mathbf{x}_i)) \bowtie (\bowtie_{j \in [\kappa]} P_j(\mathbf{y}_j)).$$

We split the boundary variables as $\partial Q_{\bar{E}} = \partial q^1_{\bar{E}} \cup \partial Q^2_{\bar{E}}$, where

$$\partial Q^1_{\bar{E}} = \{x \mid x \in \mathbf{x}_i \cap \mathbf{x}_j, i \in E, j \in \bar{E}\},$$

and

$$\partial Q^2_{\bar{E}} = \{x \mid x \in \mathbf{x}_i \cap \mathbf{y}_j, i \in E, j \in [\kappa]\} - \partial q^1_{\bar{E}}.$$

Let

$$Q^{\circ}_{\bar{E}} = (\bowtie_{i \in \bar{E}} R_i(\mathbf{x}_i))$$

be the CQ part of $Q_{\bar{E}}$.

Example 3.4.1. Figure 3.2 illustrates these concepts with the query

$$\begin{aligned}Q =& |R_1(x_1,x_2,x_3) \bowtie R_2(x_3,x_4,x_5) \bowtie R_3(x_5,x_6,x_7) \bowtie R_4(x_1,x_7,x_8) \\ & \bowtie P_1(x_2,x_4) \bowtie P_2(x_2,x_8) \bowtie P_3(x_3,x_7) \bowtie P_4(x_4,x_6)|,\end{aligned}$$

where we set $E = \{1\}$. □

The following observations about the boundary variables are straightforward.

Lemma 3.4.1. *For any CQP Q and any $E \subseteq [n]$,*

1. $\partial Q^2_{\bar{E}} \subseteq \mathbf{y}_1 \cup \cdots \cup \mathbf{y}_\kappa \subseteq \partial Q^2_{\bar{E}} \cup var(Q^\circ_{\bar{E}})$;
2. $\partial Q^1_{\bar{E}} \subseteq var(Q^\circ_{\bar{E}})$;
3. $var(Q^\circ_{\bar{E}}) \cap \partial Q^2_{\bar{E}} = \emptyset$.

Now, we look at how to compute $T_{\bar{E}}(\mathbf{I})$:

$$\begin{aligned}T_{\bar{E}}(\mathbf{I}) &= \max_{t \in \mathbf{dom}(\partial Q_{\bar{E}})} |Q_{\bar{E}}(\mathbf{I}) \bowtie t| \\ &= \max_{t \in \mathbf{dom}(\partial Q_{\bar{E}})} |Q^\circ_{\bar{E}}(\mathbf{I}) \bowtie (\bowtie_{j\in[\kappa]} P_i(\mathbf{y}_i)) \bowtie t| \\ &= \max_{\substack{t_1 \in \mathbf{dom}(\partial Q^1_{\bar{E}}) \\ t_2 \in \mathbf{dom}(\partial Q^2_{\bar{E}})}} \left|Q^\circ_{\bar{E}}(\mathbf{I}) \bowtie (\bowtie_{j\in[\kappa]} P_j(\mathbf{y}_j)) \bowtie t_1 \bowtie t_2\right| \\ &= \max_{\substack{t_1 \in \pi_{\partial Q^1_{\bar{E}}} Q^\circ_{\bar{E}}(\mathbf{I}) \\ t_2 \in \mathbf{dom}(\partial Q^2_{\bar{E}})}} \left|Q^\circ_{\bar{E}}(\mathbf{I}) \bowtie (\bowtie_{j\in[\kappa]} P_j(\mathbf{y}_j)) \bowtie t_1 \bowtie t_2\right|.\end{aligned}$$

The last step is because only $t_1 \in \pi_{\partial Q^1_{\bar{E}}} Q^\circ_{\bar{E}}(\mathbf{I})$ can join with $Q^\circ_{\bar{E}}(\mathbf{I})$. Since $|Q^\circ_{\bar{E}}(\mathbf{I})|$ is bounded by $O(N^n)$, the choices of t_1 are limited. The difficulty is that $t_2 \in \mathbf{dom}(\partial Q^2_{\bar{E}})$ has infinitely many choices. The idea is to flip the problem around. For any $B \subseteq Q^\circ_{\bar{E}}(\mathbf{I})$, we check if there exist t_1, t_2 such that

$$|B \bowtie (\bowtie_{j\in[\kappa]} P_j(\mathbf{y}_j)) \bowtie t_1 \bowtie t_2| = |B|. \tag{3.38}$$

This is equivalent to checking if $t_B \bowtie t_1 \bowtie t_2$ can pass all predicates for every $t_B \in B$. Since $\partial Q^1_{\bar{E}} \subseteq var(Q^\circ_{\bar{E}})$, for each $t_B \in B$, t_1 must be $\pi_{\partial Q^1_{\bar{E}}} t_B$. Thus the problem boils down to deciding if

$$\bigwedge_{t_B \in B, j\in[\kappa]} P_j(\mathbf{y}_j(t_B)) \tag{3.39}$$

is satisfiable, where $\mathbf{y}_j(t_B)$ denotes $\mathbf{y}_j$ after replacing its variables by the corresponding constants if they appear in t_B. Note that free variables in (3.39) are $\mathbf{z} = \partial Q^2_{\bar{E}}$. This is precisely the technical condition we impose on the predicates. Finally, we enumerate all B, and return the maximum $|B|$ for which (3.39) is satisfiable. This proves the first part of Theorem 1.1.2. However, this algorithm runs in exponential time since there are $2^{|Q^{\circ}_{\bar{E}}(\mathbf{I})|} = 2^{\text{poly}(N)}$ B's that need to be considered.

3.4.2 Comparison and Inequality Predicates

For CQPs where the predicates are inequalities or comparisons, we may without loss of generality assume that the domain of all attributes in $\mathbf{y}_1 \cup \cdots \cup \mathbf{y}_\kappa$ is $\mathbb{Z}$. We show in this subsection how to reduce the running time of the algorithm to poly(N) in this case. Let $\rho = |\partial Q^2_{\bar{E}}|$. Then t_2 takes values from $\mathbb{Z}^\rho$. The key to an efficient algorithm is thus to reduce this domain, and then apply the algorithm in [47, 4].

To reduce the domain of t_2, one natural idea is to only consider the *active domain* [2]. Let $\mathbb{Z}^*(\mathbf{I})$ be the set of integers appearing in $\mathbf{I}$ on attributes $\mathbf{y}_1 \cup \cdots \cup \mathbf{y}_\kappa$, and let $\mathbb{Z}^*(Q)$ be the set of integers appearing in the predicates of q. Then the active domain is $\mathbb{Z}^*(Q, \mathbf{I}) = \mathbb{Z}^*(Q) \cup \mathbb{Z}^*(\mathbf{I}) \cup \{-\infty, \infty\}$. However, only considering $t_2 \in (\mathbb{Z}^*(Q, \mathbf{I}))^\rho$ is not enough as seen in the following example.

Example 3.4.2. Following Example 3.4.1, suppose $P_1(x_2, x_4)$ is $x_2 > x_4$, $P_2(x_2, x_8)$ is $x_8 > x_2$, while ignoring P_3 and P_4. Consider the following instance $\mathbf{I}$:

$$\begin{aligned} R_1(x_1, x_2, x_3) &= \{(0,3,0), (0,5,0)\}, \\ R_2(x_3, x_4, x_5) &= \{(0,1,0), (0,2,0), (0,3,0)\}, \\ R_3(x_5, x_6, x_7) &= \{(0,0,0)\}, \\ R_4(x_7, x_8, x_1) &= \{(0,5,0), (0,6,0), (0,7,0)\}. \end{aligned}$$

For $E = \{1\}$, $T_{\bar{E}}(\mathbf{I})$ attains its maximum at $x_2 = 4$, which is not included in $\mathbb{Z}^*(q, \mathbf{I})$. □

This example shows that $T_{\bar{E}}(\mathbf{I})$ may attain its maximum at some value between two consecutive values in the active domain. Thus, we augment the active domain to $\mathbb{Z}^+(q, \mathbf{I})$, as follows. Let $\mathbb{Z}^*(Q, \mathbf{I}, i)$ be the ith elements in $\mathbb{Z}^*(Q, \mathbf{I})$ in order. $\mathbb{Z}^+(Q, \mathbf{I})$ includes all elements in $\mathbb{Z}^*(Q, \mathbf{I}, i)$, plus 2κ arbitrary distinct elements between $\mathbb{Z}^*(Q, \mathbf{I}, i)$ and $\mathbb{Z}^*(Q, \mathbf{I}, i+1)$ for all $i \in [|\mathbb{Z}^*(Q, \mathbf{I})| - 1]$. If there are less than 2κ elements between $\mathbb{Z}^*(Q, \mathbf{I}, i)$ and $\mathbb{Z}^*(Q, \mathbf{I}, i+1)$, all elements in between are included.

We show that it suffices to use $\mathbb{Z}^+(Q, \mathbf{I})$ as the domain of t_2.

Lemma 3.4.2. *When all the predicates are inequalities and comparisons,*

$$T_{\bar{E}}(\mathbf{I}) = \max_{\substack{t_1 \in \pi_{\partial Q^1_{\bar{E}}} Q^\circ_{\bar{E}}(\mathbf{I}) \\ t_2 \in (\mathbb{Z}^+(Q,\mathbf{I}))^\rho}} \left| Q^\circ_{\bar{E}}(\mathbf{I}) \bowtie (\bowtie_{j\in[\kappa]} P_j(\mathbf{y}_j)) \bowtie t_1 \bowtie t_2 \right|. \tag{3.40}$$

Proof. It is sufficient to show that, for any $t_2 \in \mathbb{Z}^\rho$, we can find a $t_2' \in (\mathbb{Z}^+(Q, \mathbf{I}))^\rho$ such that

$$\begin{aligned} &|Q^\circ_{\bar{E}}(\mathbf{I}) \bowtie (\bowtie_{j\in[\kappa]} P_j(\mathbf{y}_j)) \bowtie t_1 \bowtie t_2| \\ =&|Q^\circ_{\bar{E}}(\mathbf{I}) \bowtie (\bowtie_{j\in[\kappa]} P_j(\mathbf{y}_j)) \bowtie t_1 \bowtie t_2'|. \end{aligned} \tag{3.41}$$

We order the at most ρ distinct values in t_2 as $v_1, v_2, \ldots$, and map them to values in $\mathbb{Z}^+(Q, \mathbf{I})$ to obtain t_2', as follows. If $v_i \in \mathbb{Z}^*(Q, \mathbf{I})$, we map v_i to itself. For values strictly between $\mathbb{Z}^*(Q, \mathbf{I}, i)$ and $\mathbb{Z}^*(Q, \mathbf{I}, i+1)$ for some i, we map them to the additional elements in $\mathbb{Z}^+(Q, \mathbf{I})$ between $\mathbb{Z}^*(Q, \mathbf{I}, i)$ and $\mathbb{Z}^*(Q, \mathbf{I}, i+1)$ in an order-preserving fashion. Since $\rho \le 2\kappa$, this is always possible.

It is easy to see that (3.41) holds after the above mapping. This is because the attributes of t_2 and t_2' only appear in the predicates and all the comparison/inequality relationships remain unchanged. □

Since $\mathbb{Z}^+(Q, \mathbf{I}) = O(N + \kappa) = O(N)$, we can simply materialize each $P_j(\mathbf{y}_j)$ into $\{t \in (\mathbb{Z}^+(Q, \mathbf{I}))^2 \mid P_j(t)\}$, which has size $O(N^2)$. Thus, evaluating (3.40) using the algorithm in [47, 4] also takes polynomial time, and we have concluded the proof of the second part of Theorem 1.1.2.

As a practical improvement, observe that if a variable $y \in \partial Q^2_{\bar{E}}$ is only involved in inequality predicates, then it can always take a value such that all these inequalities hold. Thus, there is no need to materialize these predicates. In particular, we arrive at a simpler formula for computing $T_{\bar{E}}(\mathbf{I})$ when all predicates are inequalities, e.g., graph pattern counting queries.

Corollary 3.4.3. *For a CQP Q where all predicates are inequalities,*

$$T_{\bar{E}}(\mathbf{I}) = \max_{t_1 \in \mathbf{dom}(\partial Q^1_{\bar{E}})} \left| Q^\circ_{\bar{E}}(\mathbf{I}) \bowtie (\bowtie_{j\in[\kappa], \mathbf{y}_j \subseteq var(Q^\circ_{\bar{E}})} P_j(\mathbf{y}_j)) \bowtie t_1 \right|.$$

To compute $T_{\bar{E}}(\mathbf{I})$, we compute $Q^\circ_{\bar{E}}(\mathbf{I})$, apply all predicates $P_j(\mathbf{y}_j))$ for $j \in [\kappa], \mathbf{y}_j \subseteq var(Q^\circ_{\bar{E}}))$, do a count group-by $\partial Q^1_{\bar{E}}$, and return the maximum count.

3.5 Non-full CQs

A non-full CQ has the form

$$Q := |\pi_{\mathbf{o}}\left(R_1(\mathbf{x}_1) \bowtie \cdots \bowtie R_n(\mathbf{x}_n)\right)|,$$

where $\mathbf{o} \subseteq \mathbf{x}$ denotes the set of output variables.

Similarly, the current approach [48, 57, 29] simply computes the noise ignoring the projection. This performs badly as the projection usually reduces the true count significantly, so the noise becomes relatively much larger. In this section, we show how to add projection into the residual sensitivity framework.

For any $E \subseteq [n]$, define

$$\mathbf{o}_E = \mathbf{o} \cap (\cup_{i \in E} \mathbf{x}_i).$$

Note that $\mathbf{o} = \mathbf{o}_{[n]}$.

Given $E \subseteq [n]$, the residual query with projection is

$$Q_E := \pi_{\mathbf{o}_E}(\bowtie_{i \in E} R_i(\mathbf{x}_i)).$$

The boundary variables $\partial Q_E = \{x | x \in \mathbf{x}_i \cap \mathbf{x}_j, i \in E, j \in \bar{E}\}$ remain unchanged, but the maximum boundary $T_E(\mathbf{I})$ and witness $t_E(\mathbf{I})$ are modified as

$$T_E(\mathbf{I}) = \max_{t \in \mathbf{dom}(\partial Q_E)} |\pi_{\mathbf{o}_E}(\bowtie_{i \in E} I_i(\mathbf{x}_i) \bowtie t)|$$

and

$$t_E(\mathbf{I}) = \arg\max_{t \in \mathbf{dom}(\partial Q_E)} |\pi_{\mathbf{o}_E}(\bowtie_{i \in E} I_i(\mathbf{x}_i) \bowtie t)|.$$

If $\mathbf{o}_E = \emptyset$, there is always a $t \in \mathbf{dom}(\partial Q_E)$ such that $(\bowtie_{i \in E} I_i(\mathbf{x}_i)) \bowtie t \neq \emptyset$, which becomes $\{\langle\rangle\}$ after the projection, so $T_E(\mathbf{I}) = 1$.

Note that these definitions degenerate into the full-CQ case when $\mathbf{o} = var(Q)$.

We as before compute $\widehat{\text{LS}}_Q^{(k)}(\mathbf{I})$ by (3.19), (3.20), and then $\text{RS}_Q^{\beta}(\mathbf{I})$ by (3.21), but using the new definition of $T_E(\mathbf{I})$ with projection. Below we show that $\text{RS}_Q^{\beta}(\cdot)$ is still a valid ε-DP mechanism and it can be computed efficiently. Recall that the validity of $\text{RS}_Q^{\beta}(\cdot)$ is based on (1) $\widehat{\text{LS}}_Q^{(k)}(\cdot)$ is an upper bound of $\text{LS})_Q^{(k)}(\cdot)$; and (2) $\widehat{\text{LS}}_Q^{(k)}(\cdot)$ satisfies the smoothness property (3.5). The first depends on Lemma 3.2.5 and Theorem 3.2.1 while the second depends on Lemma 3.2.5. One can verify that, as long as Lemma 3.2.5 and Theorem 3.2.1 hold for non-full CQs, the rest of the validity proof will go through. We thus focus on verifying Lemma 3.2.5 and Theorem 3.2.1 on non-full CQs.

Lemma 3.5.1. *For non-full CQs, Lemma 3.2.1, 3.2.2, and 3.2.5 still hold.*

Proof. First, it is trivial to see, Lemma 3.2.1 and 3.2.2 are still valid. The validness of Lemma 3.2.5 is based on the self-join-free version of Lemma 3.2.5. Therefore, it suffices to prove the self-join-free version of Lemma 3.2.5.

We first show, for non-full CQs without self-joins, given any $E \subseteq [n], i \in E$, and two instances $\mathbf{I}, \mathbf{I}'$ such that $d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i)) = 1$, $I_j(\mathbf{x}_j) = I'_j(\mathbf{x}_j)$ for all $j \in E - \{i\}$, we have $|T_E(\mathbf{I}) - T_E(\mathbf{I}')| \leq T_{E-\{i\}}(\mathbf{I})$.

There are three cases: I'_i is obtained from insertion, deletion or change a tuple t' from I_i. For the first case, if $\mathbf{o}_E = \emptyset$, $T_E(\mathbf{I}') = T_E(\mathbf{I}) = 1$ thus

$$|T_E(\mathbf{I}') - T_E(\mathbf{I})| = 0 \leq T_{E-\{i\}}(\mathbf{I}).$$

If $\mathbf{o}_E \neq \emptyset$, define

$$\widetilde{T}_E(\mathbf{I}) = |\pi_{\mathbf{y}_E}((\bowtie_{i\in E} I_i(\mathbf{x}_i)) \ltimes t_E(\mathbf{I}'))|$$

and have

$$|T_E(\mathbf{I}') - T_E(\mathbf{I})| = T_E(\mathbf{I}') - T_E(\mathbf{I}) \leq T_E(\mathbf{I}') - \widetilde{T}_E(\mathbf{I}).$$

Next, we bound $T_E(\mathbf{I}') - \widetilde{T}_E(\mathbf{I})$.

$$\begin{aligned}
&T_E(\mathbf{I}') - \widetilde{T}_E(\mathbf{I}) \\
=&|\pi_{\mathbf{o}_E}((\bowtie_{i\in E} I_i(\mathbf{x}_i)) \ltimes t_E(\mathbf{I}'))| - |\pi_{\mathbf{o}_E}((\bowtie_{i\in E} I'_i(\mathbf{x}_i)) \ltimes t_E(\mathbf{I}'))| \\
=&|\pi_{\mathbf{o}_E}((\bowtie_{j\in E-\{i\}} I_j(\mathbf{x}_j)) \bowtie t' \ltimes t_E(\mathbf{I}'))| \\
=&|\pi_{\mathbf{o}_{E-\{i\}}}((\bowtie_{j\in E-\{i\}} I_j(\mathbf{x}_j)) \bowtie t' \ltimes t_E(\mathbf{I}'))| && (3.42)\\
\leq&\max_{t\in\mathbf{dom}(\partial Q_{E-\{i\}})} |\pi_{\mathbf{o}_{E-\{i\}}}((\bowtie_{j\in E-\{i\}} I_j(\mathbf{x}_j)) \ltimes t)| && (3.43)\\
=&T_{E-\{i\}}(\mathbf{I}).
\end{aligned}$$

(3.42) is derived by fixing t' will fix the values of attributes in $\mathbf{x}_i$. (3.43) is because the attributes of $t' \bowtie t_E(\mathbf{I}')$ are divided into two parts, one is interior to $Q_{[n]-(E-\{i\})}$ while the other is $\partial Q_{E-\{i\}}$; the values of attributes in the first part will not affect the join between $t' \bowtie t_E(\mathbf{I}')$ and $\bowtie_{j\in E-\{i\}} I_j(\mathbf{x}_j)$.

For the second case, we can draw the conclusion with the same idea. For the third case, where $\mathbf{I}'$ is achieved by changing one tuple from $\mathbf{I}$, define their common part as $\mathbf{I}''$:

$$\begin{cases} |T_E(\mathbf{I}) - T_E(\mathbf{I}'')| \le T_{E-\{i\}}(\mathbf{I}'') \\ |T_E(\mathbf{I}') - T_E(\mathbf{I}'')| \le T_{E-\{i\}}(\mathbf{I}'') \\ T_E(\mathbf{I}'), T_E(\mathbf{I}) \ge T_E(\mathbf{I}'') \end{cases}$$

$$\Rightarrow |T_E(\mathbf{I})| - |T_E(\mathbf{I}')| \le T_{E-\{i\}}(\mathbf{I}'') \le T_{E-\{i\}}(\mathbf{I}). \tag{3.44}$$

By now, we prove for CQs without self-joins, any $\mathbf{I}, \mathbf{I}'$ such that they only differ by one tuple in $R_1(\mathbf{x}_i)$, the difference between $T_E(\mathbf{I})$ and $T_E(\mathbf{I}')$ is at most $T_{E-\{i\}}(\mathbf{I})$. Then, we can follow the idea from Lemma 4.6 to 4.8 in our prior work [29] to prove the self-join-free version of Lemma 3.2.5. □

Theorem 3.5.1. *For non-full CQs, Theorem 3.2.1 still holds.*

Proof. As shown in the remark after Theorem 3.2.1, the validity of Theorem 3.2.1 is based on Lemma 3.2.5 and 3.2.6. Since Lemma 3.2.5 has already been shown to be valid for non-full CQs, it suffices to only show Lemma 3.2.6 holds: for non-full CQs without self-joins, $\mathrm{LS}_Q(\mathbf{I}) \le \max_{i \in P^n} T_{[n]-\{i\}}(\mathbf{I})$. Recall

$$\mathrm{LS}_Q(\mathbf{I}) = \max_{i \in P^n} \max_{\mathbf{I}': d(\mathbf{I}, \mathbf{I}')=1, d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i))=1} |Q(\mathbf{I}) - Q(\mathbf{I}')|.$$

Given $\mathbf{I}, \mathbf{I}'$ differing by one tuple in $R_i(\mathbf{x}_i)$. $\mathbf{I}'$ can be different from $\mathbf{I}$ in three ways: insertion, deletion or change a tuple $t' \in \mathbf{dom}(\mathbf{x}_i)$. In the first two cases

$$\begin{aligned} &|Q(\mathbf{I}) - Q(\mathbf{I}')| \\ =&|\pi_{\mathbf{o}}(t' \bowtie (\bowtie_{j \in [n]-\{i\}} I_j(\mathbf{x}_j)))| \\ =&|\pi_{\mathbf{o}_{[n]-\{i\}}}(t' \bowtie (\bowtie_{j \in [n]-\{i\}} I_j(\mathbf{x}_j)))| && (3.45) \\ \le& \max_{t \in \mathbf{dom}(\partial Q_{[n]-\{i\}})} |\pi_{\mathbf{o}_{[n]-\{i\}}}((\bowtie_{j \in [n]-\{i\}} I_j(\mathbf{x}_j)) \ltimes t)| && (3.46) \\ =&T_{[n]-\{i\}}(\mathbf{I}). \end{aligned}$$

(3.45) is derived by fixing t' will fix the values of attributes for $\mathbf{x}_i$. (3.46) is because $\mathbf{x}_i$ can be divided into two parts: $\partial Q_{\{i\}}$ and $\mathbf{x}_i - \partial Q_{\{i\}}$; the first part is equal to $\partial Q_{[n]-\{i\}}$ while the second one does not affect the join.

In the case where $\mathbf{I}'$ is achieved by changing one tuple from $\mathbf{I}$, we can use a similar idea as Lemma 3.5.1 and get $|Q(\mathbf{I}) - Q(\mathbf{I}')| \le T_{[n]-\{i\}}(\mathbf{I})$. □

In terms of computation, we observe that $T_E(\mathbf{I})$ with projection is still an AJAR/FAQ query, but now with 3 semiring aggregations $(\max, +, \max)$, so it can still be computed by the algorithm in [47, 4] in polynomial time. Furthermore, one can verify that Lemma 3.2.11 still holds non-full queries, so it takes $O(1)$ time to compute $\mathrm{RS}_Q^\beta(\cdot)$ after all the $T_E(\mathbf{I})$'s have been computed.

Theorem 3.5.2. *For any non-full CQ Q, $\mathrm{RS}_Q^\beta(\cdot)$ is an ε-DP mechanism that can be computed in* $\mathrm{poly}(N)$ *time.*

Non-full CQs with predicates can be handled by combining the methods described in this and Section 3.4. More precisely, for a non-full CQ with general predicates, we add $\pi_{\mathbf{o}_{\bar{E}}}$ on both sides of (3.38); if the predicates are inequalities and comparisons, we materialize each predicate and then apply the algorithm above. The resulting $\mathrm{RS}_Q^\beta(\cdot)$ is still ε-DP, and can be much smaller than that on the full CQ. However, the lower bound Theorem 3.3.1 no longer holds for non-full CQs, thus $\mathrm{RS}_Q^\beta(\cdot)$ is not $O(1)$-neighborhood optimal. We complement this with the following negative result.

Theorem 3.5.3. *For any $\varepsilon > 0$, any (r, c)-neighborhood optimal ε-DP mechanism $\mathcal{M}(\cdot)$ for the query $Q := |\pi_{x_1}(R_1(x_1, x_2) \bowtie R_2(x_2))|$, where R_1 is the private relation, must have $cr^2 \geq N$.*

Proof. Let $\mathcal{M}(\cdot)$ be as given. We will construct two instances $\mathbf{I}$ and $\mathbf{I}'$ such that $\mathcal{M}(\mathbf{I})$ and $\mathcal{M}(\mathbf{I}')$ must differ a lot. Suppose $\mathbf{dom}(x_1) = \mathbf{dom}(x_2) = \mathbb{Z}$. The public relation R_2 takes the same instance $I_2 = I_2' = [r]$. For the private relation R_1, we set $I_1 = [N/r] \times [r]$ and $I_1' = [N] \times \{0\}$. Note that $Q(\mathbf{I}) = N/r$ and $Q(\mathbf{I}') = 0$. In addition, for any $\mathbf{I}''$ with $d(\mathbf{I}, \mathbf{I}'') \leq r$, $Q(\mathbf{I}'') = N/r$. For any $\mathbf{I}''$ with $d(\mathbf{I}', \mathbf{I}'') \leq r$, $Q(\mathbf{I}'') \leq r$.

First consider $\mathbf{I}$. The adversary sets $\mathcal{M}'(\cdot) \equiv N/r$, so $\mathrm{Err}(\mathcal{M}', \mathbf{I}'') = 0$ for all $\mathbf{I}''$ in the r-neighborhood of $\mathbf{I}$. Since $\mathcal{M}(\cdot)$ is (r, c)-neighborhood optimal, $\mathcal{M}(\mathbf{I})$ must output N/r deterministically. As $\mathcal{M}(\cdot)$ is ε-DP, $\mathcal{M}(\cdot)$ must output N/r deterministically at all instances. So its error on $\mathbf{I}'$ is $\mathrm{Err}(\mathcal{M}, \mathbf{I}') = N/r$.

At $\mathbf{I}'$, the adversary sets $\mathcal{M}'(\cdot) \equiv 0$. For any $\mathbf{I}''$ in the r-neighborhood of $\mathbf{I}'$, $\mathcal{M}'(\cdot)$ has error at most $\mathrm{Err}(\mathcal{M}', \mathbf{I}'') \leq r$. Thus, we conclude that $c \geq \frac{N/r}{r}$, or $cr^2 \geq N$. □

Thus, if one still desires $c = O(1)$, r must be at least $\Omega(\sqrt{N})$. We also remark that this negative result holds even under relaxed DP, since only substitutions are used when defining the neighborhoods in the proof.

3.6 Experiments

Our analysis in Section 3.3 shows that $\mathrm{RS}^{\beta}_{Q}(\cdot)$ is $O(1)$-neighborhood optimal. At the same time, $\mathrm{ES}^{\beta}_{Q}(\cdot)$ does not have any optimal guarantee. In this section, we conduct an experimental study on the actual gap on a collection of multi-way joins over both benchmark and real-world datasets. We also investigate its implications to the noise levels, as well as the computational overheads.

We have also tested wPINQ [70], the earliest work on multi-way join queries under the same DP policy as ours. Our results confirm those reported in [48], that wPINQ has worse utility than elastic sensitivity.

3.6.1 Setup

Datasets We use two datasets in our experiments: TPC-H and the Facebook ego-network dataset. The TPC-H schema has many foreign key constraints. As our DP policy does not consider foreign key constraints, when a tuple is deleted in one relation, say `Customer`, we might obtain a neighboring instance that violates the foreign key constraint from `Orders` to `Customer`. To resolve this inconsistency, the correct way to interpret our DP policy is the following. We conceptually create the following relations by projecting the original relations to the join attributes. This results in the following 8 projected relations: `Region(RK)`, `Nation(RK, NK)`, `Customer(NK, CK)`, `Orders(CK, OK)`, `Supplier(NK, SK)`, `Part(PK)`, `PartSupp(SK, PK)`, `Lineitem(SK, PK, OK)`. We abbreviate these relations as `R`, `N`, `C`, `O`, `S`, `P`, `PS`, `L`, respectively. All relations except `R` and `P` capture relationships, e.g., `Orders(CK, OK)` stores which custom placed which order. We treat `C, O, S, PS, L` as private relations. There are no foreign key constraints between these private relations, so our DP policy will be well defined. Note that the same interpretation is used in the prior work [48]. We generated datasets with scale factors ranging from 0.01 to 10; the one with scale factor 1 contains about 7.5 million tuples.

The Facebook ego-network dataset is from SNAP [58], which contains $4,039$ nodes and $176,467$ directed edges. The nodes are organized as 193 "social circles". We merged these social circles into 5 "mega-circles", and created five relations $R_i(x, y), i = 1, \ldots, 5$, where each $R_i(x, y)$ contains all edges (x, y) that originate in the i-th mega-circle. In addition, we create a relation $R_6(x, y, z)$ that consists of all triangles formed by edges in R_1 or R_2, i.e.,

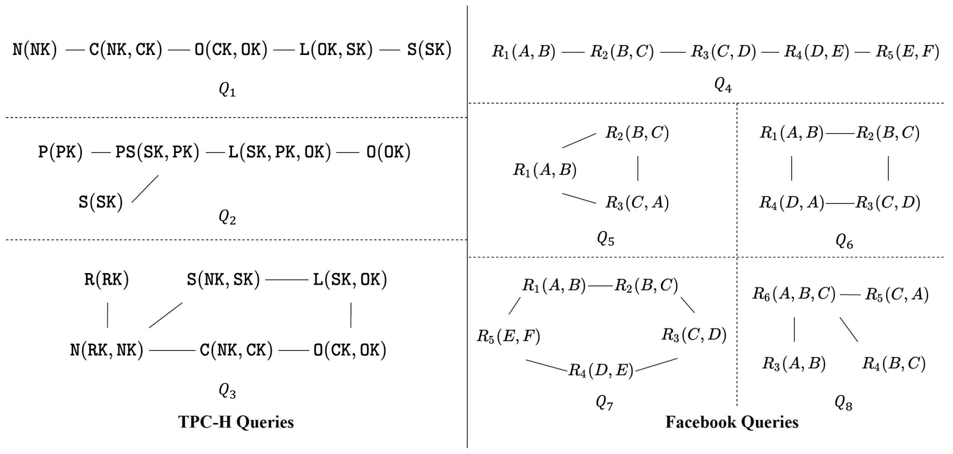


Figure 3.3: The join structure of queries.

$R_6(x, y, z) := (R_1(x, y) \bowtie R_1(y, z) \bowtie R_1(z, x)) \cup (R_2(x, y) \bowtie R_2(y, z) \bowtie R_2(z, x))$. The 6 relations have 40968, 55125, 28231, 22486, 19179, 6080904 tuples, respectively, and they are all regarded as private relations. This data set models the scenario where different types of relations exist among the entities, such as friendship, co-workers, co-authors, family members, etc.

Queries For TPC-H data, we used Q7, Q9, and Q5 from the benchmark, but removed projections, predicates, and group-by conditions, and their join structures are shown as Q_1, Q_2, Q_3 in Figure 3.3. For the Facebook ego-network dataset, we used 5 queries, shown as $Q_4, \ldots, Q_8$ in Figure 3.3. Note that all joins on the TPC-H data are foreign-key joins (i.e., many-to-one), while those on the Facebook data are many-to-many joins.

Dataset		TPC-H			Facebook				
Query		Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8
Query result		6,001,215	6,001,215	239,917	1,666,978,389	19,927	285,754	6,348,654	21,613
wPINQ output		8,680	9,770	385	54.2	131.4	44.1	23.8	183.6
Residual Sensitivity (RS_Q^β)	Min Value	694	694	49	77,100,000	203	2,790	86,800	50
	Max Value	51,900	52,000	51,800	301,000,000	1,410	54,500	2,050,000	51,300
	Running Time(s)	27.6	53.6	48.6	2.96	1.68	4.71	18.1	20.1
Elastic Sensitivity (ES_Q^β)	Min Value	1,740	2,140	175,000,000	25,500,000,000	219,000	110,000,000	55,000,000,000	561
	Max Value	4,950,000	1,870,000	263,000,000	25,500,000,000	219,000	110,000,000	55,000,000,000	2,440,000
	Running Time(s)	7.55	9.02	6.73	0.300	0.611	0.628	5.725	8.78
$\text{ES}_Q^\beta/\text{RS}_Q^\beta$	Min	2.51×	3.08×	5,069×	84.8×	156×	2,010×	26,900×	11.2×
	Max	273×	67×	3,580,000×	330×	1,080×	39,300×	634,000×	178×
	Avg	94.4×	27.8×	1,750,000×	286×	875×	27,300×	503,000×	80.6×
Running time: $\text{RS}_Q^\beta/\text{ES}_Q^\beta$		3.65×	5.94×	7.23×	9.86×	2.75×	7.50×	3.17×	2.29×

Table 3.2: Comparison among wPINQ, residual sensitivity and elastic sensitivity.

3.6.2 Implementation

Both residual sensitivity and elastic sensitivity can be computed easily by SQL. Elastic sensitivity is computed by a UDF, after collecting the maximum frequencies of each relation by SQL [48]. Similarly, for residual sensitivity, each $T_E(\mathbf{I})$ can be computed using query (3.7). Then $\mathrm{RS}_Q^\beta(\mathbf{I})$ can be computed by formula (3.21) using a UDF. Thus, residual sensitivity enjoys the same benefit of elastic sensitivity that it can be easily integrated into any database system without any modification to the kernel. To automate this process, we have built a system prototype on top of PostgreSQL.[2] In our experiments, we used PostgreSQL 11.5. For wPINQ, we follow the similar settings as [48]. All experiments are conducted on a machine equipped with a 2.7 GHz Intel Core i7 and 16GB of memory.

However, we notice that PostgreSQL is not able to find the optimal query plan for executing query (3.7). Thus, we wrote another query rewriter that uses the following rules to rewrite a query in the form of (3.7). Our current query rewriter has only implemented a subset of these rules (this should be the job of the query optimizer of the database!), so one may still need some manual rewriting for the best performance.

Disconnected queries If Q_E consists of a few connected components, then Q_E is the Cartesian product of the join results of each connected component. In this case, T_E is simply the product of the maximum boundaries of these connected components. It is more efficient to evaluate (3.7) for each connected component separately.

Example 3.6.1. Consider query Q_1 in Figure 3.3 with $E = \{\mathtt{N}, \mathtt{C}, \mathtt{L}, \mathtt{S}\}$. We have $T_E = T_{E_1} \cdot T_{E_2}$ where $E_1 = \{\mathtt{N}, \mathtt{C}\}, E_2 = \{\mathtt{L}, \mathtt{S}\}$. □

Exploiting dependencies Suppose there is a functional dependency $X \to Y$ and $X \subseteq \partial Q_E$, then it is clear that we can add all attributes in Y to the group-by attributes ∂Q_E without affecting the results of query (3.7). This step can be repeatedly applied. If all attributes of Q_E can be added to ∂Q_E, then the result of (3.7) must be either 1 (when Q_E is non-empty) or 0 (when Q_E is empty), provided there are no duplicated tuples in a relation.

Example 3.6.2. Consider query Q_3 in Figure 3.3 with $E = \{\mathtt{R}, \mathtt{N}, \mathtt{S}, \mathtt{C}\}$, and $\partial Q_E = \{\mathtt{SK}, \mathtt{CK}\}$. We have $\mathtt{SK} \to \mathtt{NK}$ (SK is the primary key of S) so we can add NK to ∂Q_E. Also,

[2] Code is available at `https://github.com/hkustDB/ResidualSensitivity`.

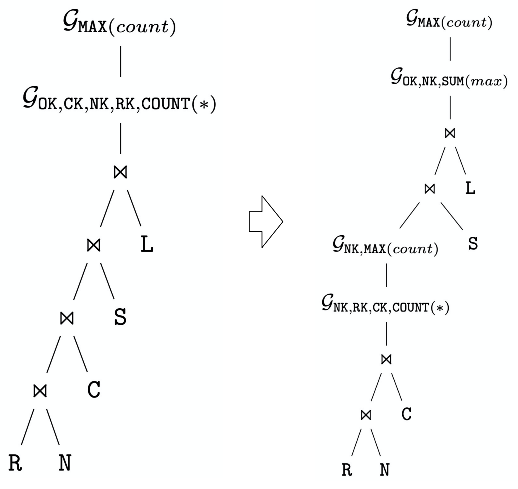

$\mathcal{G}_{\texttt{MAX}(count)}$
$\mathcal{G}_{\texttt{OK,CK,NK,RK,COUNT}(*)}$
⋈
L
S
C
R
N
$\mathcal{G}_{\texttt{MAX}(count)}$
$\mathcal{G}_{\texttt{OK,NK,SUM}(max)}$
$\mathcal{G}_{\texttt{NK,MAX}(count)}$
$\mathcal{G}_{\texttt{NK,RK,CK,COUNT}(*)}$

Figure 3.4: The rewriting of query of T_E for Q_3 with $E = \{\texttt{R}, \texttt{N}, \texttt{C}, \texttt{S}, \texttt{L}\}$.

$\texttt{NK} \to \texttt{RK}$ (NK is the primary key of N), so we can also add RK to ∂Q_E. Now all attributes of Q_E have been added to ∂Q_E and we only need to check whether Q_E is empty. □

Aggregation push-down When evaluating (3.7), PostgreSQL would first compute the join and then the aggregation, which is inefficient. We can push down the aggregation as far as possible to reduce the execution cost. Joglekar et al. [47] present a general framework for pushing down aggregations, which in turn defines the width w as mentioned in Section 3.2.1.

Example 3.6.3. Consider query Q_3 in Figure 3.3 with $E = \{\texttt{R}, \texttt{N}, \texttt{C}, \texttt{S}, \texttt{L}\}$, we can first add attributes $\texttt{NK}, \texttt{RK}$ to ∂Q_E by exploiting dependencies. Then, we rewrite (3.7) by pushing down both COUNT and MAX. The query plans before and after this rewrite are shown in Figure 3.4. The optimized query plan has $w = 1$ and it can be evaluated in linear time. In practice, this reduces PostgreSQL's execution time by roughly 100 times in our experiments. □

Incremental computation Finally, observe that the subquery in (3.7):

$$\texttt{SELECT COUNT}(*) \texttt{ AS Boundary FROM } Q_E \texttt{ GROUP BY } \partial Q_E \tag{3.47}$$

can be computed incrementally for E's that differ by one relation. Thus, we can order the computation of T_E's in a way so that previous results can be re-used.

Example 3.6.4. Consider query Q_6 in Figure 3.3, where we need to compute T_{E_1}, T_{E_2}, T_{E_3} for $E_1 = \{R_1\}$, $E_2 = \{R_1, R_2\}$, $E_3 = \{R_1, R_2, R_3\}$, among others. First, the result of executing (3.47) on Q_{E_1} is just R_1 itself, with an additional column `Boundary` whose values are all 1. Then we execute (3.47) on Q_{E_2}:

$$\begin{aligned}&\texttt{SELECT A, C COUNT}(*)\ \texttt{AS Boundary FROM}\ R_1, R_2\\&\texttt{WHERE}\ R_1.\texttt{B} = R_2.\texttt{B GROUP BY A, C}\end{aligned}\tag{3.48}$$

Next, we can run

$$\begin{aligned}&\texttt{SELECT A, D SUM}((3.48).\texttt{Boundary})\ \texttt{AS Boundary FROM}\ (3.48), R_2\\&\texttt{WHERE}\ (3.48).\texttt{C} = R_2.\texttt{C GROUP BY A, D}\end{aligned}$$

to obtain the result of (3.47) on Q_{E_3}. □

3.6.3 Experimental Results

Compare with wPINQ We test wPINQ on all queries and find it losses the utility in all cases: as shown in Table 3.2, wPINQ outputs a result much less than 1% of the real query result. That is because, the wPINQ ensures any tuple can at most effect one counting query result by scaling down the tuples' weights. Such operations can lead to the output result much smaller than the real one and that becomes more problem with the number of relations increase. The wPINQ performs well in the case where most tuples affect no more than one query result or the case where tuples in the same relation have the same degree(counting triangles incident on vertices with fixed degrees). However, in our experiments, tuples join with any number of tuples. Besides, it has been shown that wPINQ performs much worse than elastic sensitivity for non-histogram counting queries [48].

Compare with elastic sensitivity Since elastic sensitivity (ES_Q^β) and residual sensitivity (RS_Q^β) can be used in exactly the same manner to calibrate noise, where the noise level is proportional to the sensitivity value, we can compare their sensitivity values directly, given the same smoothing parameter β. We tried 7 different values $\beta = 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64$, and computed ES_Q^β and RS_Q^β for each of the queries in Figure 3.3. For TPC-H queries, we used the dataset with scale factor 1. In Table 3.2, we report the maximum, minimum, and the average value of the sensitivities over the β's, as well as their ratios. Note that both sensitivities decrease as β increases, but the ratio $\mathrm{ES}_Q^\beta/\mathrm{RS}_Q^\beta$ is not necessarily monotone.

First, RS_Q^β is always no more than ES_Q^β, while the gap can be *very* large for certain queries. More importantly, the results reveal two fundamental reasons why RS_Q^β offers a much tighter upper bound on the smooth sensitivity than ES_Q^β. The first one is the data skewness. ES_Q^β is calculated based on multiplying the maximum frequencies in the relations. In doing so, it makes the simple but very pessimistic assumption that these most frequent attribute values can join, thereby shooting up the sensitivity tremendously. If data is skewed, there are very few heavy hitters, and the chance that they can join is low. On the other hand, RS_Q^β computes the actual join of the residual query, so the presence of the heavy hitters will not affect the sensitivity, unless they can actually join. We see from Table 3.2 that the ratio $\mathrm{ES}_Q^\beta/\mathrm{RS}_Q^\beta$ is generally larger on the Facebook dataset, which is real network data with high skewness. On the other hand, TPC-H data is more uniform. The second situation where RS_Q^β is much smaller than ES_Q^β is on cyclic queries (Q_3, Q_5, Q_6, Q_7). This is because the approach towards cyclicity taken by ES_Q^β, which just removes some join conditions, is too simplistic. Removing these join conditions dramatically enlarges the sensitivity, because these join conditions put restrictions on how join results can be formed. On the other hand, RS_Q^β is defined in a unified manner over acyclic and cyclic queries.

In Table 3.2, we also report the running times of these queries, which include executing the query itself plus the time spent in computing the sensitivity. The time to add noise to the query answer is negligible. We repeated each query 10 times and the average running time is reported. We see that RS_Q^β indeed requires more time to evaluate, but considering the huge improvement in the accuracy of the released query answers (which will be more prominently compared next), the extra time is well spent.

Scalability To examine the effects as data scale changes, we used TPC-H datasets with scale factors ranging from 0.01 to 10. To get a more intuitive sense, we calculated the noise level from the computed sensitivity with both the Cauchy mechanism and Laplace mechanism as described in Section 3.1.3.1. We fix the privacy parameter $\varepsilon = 0.8$. For the Laplace mechanism, we set $\delta = 10^{-7}, 2 \times 10^{-8}, 10^{-8}, 2 \times 10^{-9}, 10^{-9}, 2 \times 10^{-10}, 10^{-10}$ for TPC-H dataset with different scale and set $\delta = 10^{-9}/10^{-7}$ for Facebook dataset with R_6 involved/uninvolved.

The noise levels for different data scales are plotted in the top row of Figure 3.5, in

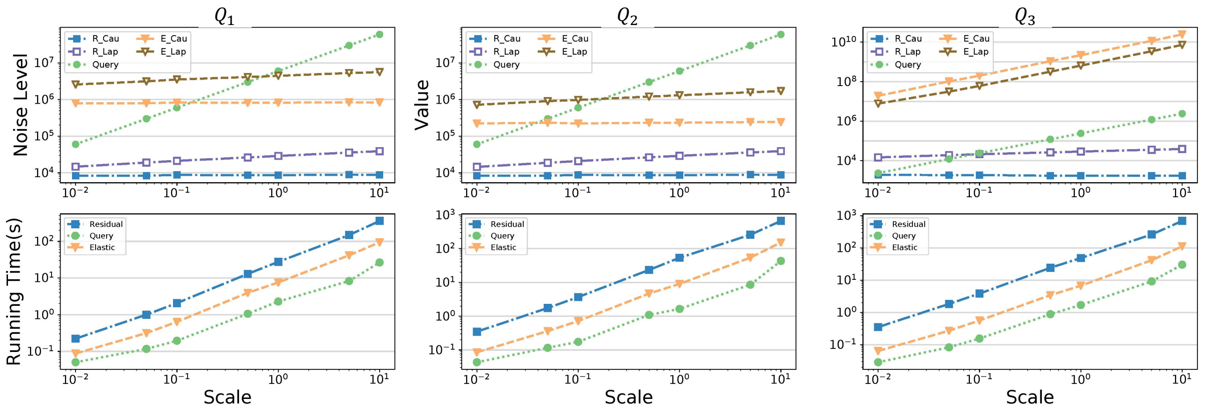


Figure 3.5: Running times and noise levels of residual sensitivity and elastic sensitivity with different noise mechanisms for different queries and data scales. R/E represents the noise level calculated from residual sensitivity or elastic sensitivity, respectively, while Cau/Lap denotes the respective noise mechanism.

which we also plot the actual query answer. Note that a noise level higher than the query answer means that the noise-masked result would be basically useless. We see from the results that the noise level from RS_Q^β is always below the query answer, while this is not the case for ES_Q^β. In particular, for Q_3, which is a cyclic query, the noise level of ES_Q^β is always (much) higher than the query answer.

Another observation is that the noise level does not increase much with the data scale. Intuitively, this is because sensitivity measures the impact of an individual tuple, which in some sense is a “local” measure. The implication is that the released query answer is more accurate, relatively speaking, on larger datasets, which is a nice property. We also observe that the noise level from the Laplace mechanism increases more than that from Cauchy. This is because we set smaller δ for larger dataset in the Laplace mechanism, which can further bring impact on β, while β is independent of data size for Cauchy.

The running times are plotted in the second row of Figure 3.5. There is not much surprise there, but it is nice to see that the growth rate of the time for computing the RS_Q^β is more or less the same as that computing the query itself. This means that the cost of evaluating query (3.7) (after appropriate rewriting) is in the same ballpark as that of executing the query itself, while RS_Q^β needs to evaluate a constant number of such queries.

Privacy parameter ε Lastly, we conducted experiments to see how the privacy parameter ε affects various mechanisms. Recall that a smaller ε means higher privacy protection, but also increases the noise level. In fact, it does so via two channels: (1) A smaller ε

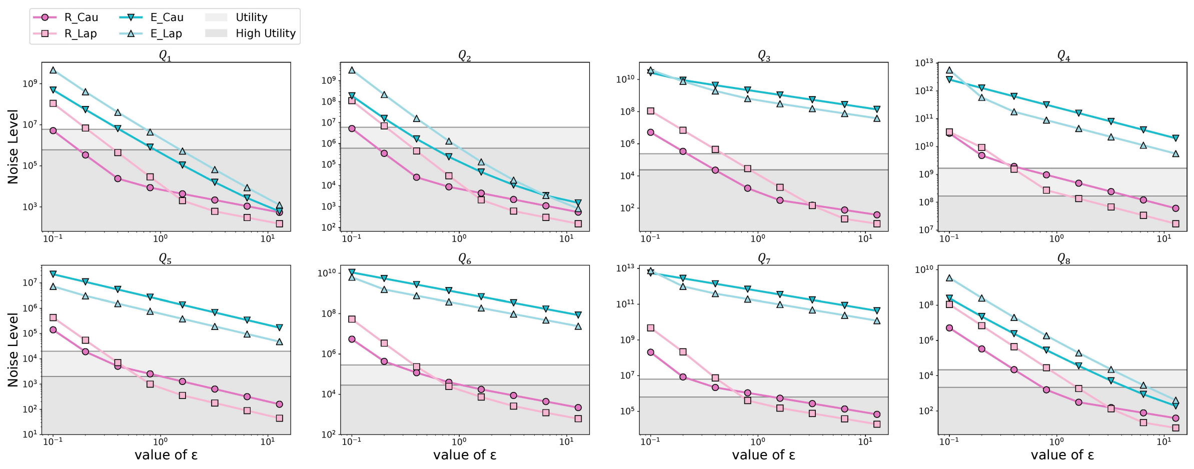


Figure 3.6: The noise level of residual sensitivity and elastic sensitivity for various values of ε.

increases the coefficient of the noise distribution; and (2) a smaller ε leads to a smaller β, hence a higher sensitivity. We tried various values of ε from 0.1 to 12 and tested the 8 queries. The results are plotted in Figure 3.6. In the figures, we also plot the query answer and 10% of that: A noise level below the query answer is considered to have utility while having high utility if it is below 10% of the query answer.

The first message the figures convey is nothing but a reconfirmation of the results in Table 3.2: The gap between the sensitivities directly translates to that between the noise levels. On cyclic queries and/or data with high skewness (Q_3, Q_4, Q_5, Q_6, Q_7), ES_Q^β does not have utility even with ε is as large as 12, which is usually considered too high. For easy queries (Q_1, Q_2, Q_8), RS_Q^β is able to obtain utility or high utility at a much smaller ε.

The more interesting observation is the comparison between the two noise mechanisms. We see that, when combined with RS_Q^β, the two mechanisms have a crossover on every query: the Cauchy mechanism seems to work better for smaller ε, while the Laplace mechanism favors larger ε. This is precisely due to the two channels through which ε affects the noise level. The coefficients of both noise distributions are inversely proportional to ε, so the noise levels both decrease as ε increases. Meanwhile, ε also affects β through the second channel, but its impact on β is larger for the Laplace mechanism. A larger β reduces RS_Q^β, although this relationship is a complicated one. Anyhow, because the Laplace mechanism is more sensitive to the change in ε, we see steeper curves in its noise levels as we vary ε. On the other hand, this phenomenon is not obvious for ES_Q^β.

This is because ES_Q^β is not very sensitive to β on many queries (as can be seen from Table 3.2), which reduces the effects of the second channel. In fact, being sensitive to β is a necessary property of any good upper bound on the smooth sensitivity, which itself is highly sensitive to β. In the extreme case, if an upper bound is completely insensitive to β, it just boils down to the global sensitivity.

Part II

Queries Answering under User-level Differential Privacy

CHAPTER 4

ANSWERING SA QUERIES UNDER USER-DP

As previously discussed in Section 1.2.2, the user-DP framework is concerned with sum aggregation queries. In this context, addressing SA queries is akin to solving a one-dimensional sum estimation problem. This chapter focuses on the sum estimation problem, while the discussion of the user-DP model in a relational database is deferred to the subsequent section.

4.1 Preliminaries

4.1.1 Notation

Given a multiset $\mathbf{I} = \{X_1, \ldots, X_N\} \in \mathbb{Z}^N$, and *radius* is $\text{rad}(\mathbf{I}) = \max_i |X_i|$. For any $\mathcal{S} \subseteq \mathbb{R}$, let $|\mathbf{I} \cap \mathcal{S}| = |\{1 \le i \le N \mid X_i \in \mathbf{I} \cap \mathcal{S}\}|$. Define $[S] := \{0, 1, \ldots, S\}$. The sum query is $\text{Sum}(\mathbf{I}) = \sum_{i=1}^{N} X_i$. We say $\mathbf{I}' \subseteq \mathbf{I}$ if $\mathbf{I}'$ can be obtained by deleting several elements from $\mathbf{I}$. In this chapter, we use e as the base of log and define $\log(x) = 1$ for any $x \le e$, unless stated otherwise.

In this chapter, we use high-probability error, which is defined as

$$\text{Err}(\mathcal{M}, \mathbf{I}) = \inf\left\{\lambda \in \mathbb{R} \mid \Pr\left[|\mathcal{M}(\mathbf{I}) - Q(\mathbf{I})| \le \lambda\right] \ge \frac{2}{3}\right\}.$$

4.1.2 More Knowledge of Differential Privacy

The DP definition has already been introduced in Section 1. The following two properties of DP are well-known:

Lemma 4.1.1 (Post Processing [33]). *If $\mathcal{M} : \mathcal{X}^n \to \mathcal{Y}$ satisfies ε-DP and $\mathcal{M}' : \mathcal{Y} \to \mathcal{Z}$ is any randomized mechanism, then $\mathcal{M}'(\mathcal{M}(\mathbf{I}))$ satisfies ε-DP.*

Lemma 4.1.2 (Basic Composition [33]). *If $\mathcal{M}_1 : \mathcal{X}^n \to \mathcal{Y}$ satisfies ε_1-DP and $\mathcal{M}_2 : \mathcal{X}^n \times \mathcal{Y} \to \mathcal{Z}$ satisfies ε_2-DP, then $\mathcal{M}_2(D, \mathcal{M}_1(\mathbf{I}))$ satisfies $(\varepsilon_1 + \varepsilon_2)$-DP.*

Algorithm 1: SVT.

```
  Input: $T$, $\varepsilon$, $Q_1(\mathbf{I}), Q_2(\mathbf{I}), \dots$
1 $\tilde{T} \leftarrow T + \mathrm{Lap}(2/\varepsilon)$;
2 for $i \leftarrow 1, 2, \dots$ do
3     $\tilde{Q}_i(\mathbf{I}) \leftarrow Q_i(\mathbf{I}) + \mathrm{Lap}(4/\varepsilon)$;
4     if $\tilde{Q}_i(\mathbf{I}) > \tilde{T}$ then
5         Break;
6     end
7 end
8 return $i$;
```

A basic pure DP mechanism is the Laplace mechanism, which has already been discussed in Section 3.1.3.

$$\mathrm{DS}_Q(\mathbf{I}) := \max_{\mathbf{I}', \mathbf{I} \sim \mathbf{I}, \mathbf{I}' \subseteq \mathbf{I}} |Q(\mathbf{I}) - Q(\mathbf{I}')| \tag{4.1}$$

is the *downward sensitivity* of $\mathbf{I}$. For sum estimation, $\mathrm{DS}_{\mathrm{Sum}}(\mathbf{I}) = \mathrm{rad}(\mathbf{I})$. $\mathrm{DS}_Q(\mathbf{I})$ can be shown to be a per-instance lower bound. We defer the discussion of this to the next section.

4.1.3 The Sparse Vector Technique

The Sparse Vector Technique (SVT) [36] has as input a (possibly infinite) sequence of queries, $Q_1, Q_2, \dots$, where each query has global sensitivity 1, and a threshold T. It aims to find the first query whose answer is above T. The detailed algorithm is given in Algorithm 8. The SVT has been shown to satisfy ε-DP and enjoy the following error guarantee, which says that it will not stop until it gets close to T.

Lemma 4.1.3 ([37])**.** *Suppose there exists a k_1 less than the length of the query sequence such that for all $i = 1, \dots, k_1$, $Q_i(\mathbf{I}) \le T - \frac{8}{\varepsilon}\log(2k_1/\beta)$. Then with probability at least $1 - \beta$,* SVT *returns an $i \ge k_1 + 1$.*

However, as will be clear later, we will actually need a complementary result that guarantees that SVT will stop in time. The following lemma gives such a result. More importantly, it also yields a utility guarantee on the returned query.

Lemma 4.1.4. *If there exists a k_2 such that $Q_{k_2}(\mathbf{I}) \ge T + \frac{6}{\varepsilon}\log(2/\beta)$, then with probability at least $1 - \beta$,* SVT *returns an $i \le k_2$ such that $Q_i(\mathbf{I}) \ge T - \frac{6}{\varepsilon}\log(2k_2/\beta)$.*

Proof. First, by the tail bound of the Laplace distribution, with probability at least $1-\frac{\beta}{2}$,

$$|\tilde{T}-T| < \frac{2}{\varepsilon}\log(2/\beta). \tag{4.2}$$

And with probability at least $1-\frac{\beta}{4}$,

$$\tilde{Q}_{k_2}(\mathbf{I}) > Q_{k_2}(\mathbf{I}) - \frac{4}{\varepsilon}\log(2/\beta). \tag{4.3}$$

By a union bound over (4.2) and (4.3), together with the given condition $Q_{k_2}(\mathbf{I}) \geq T + \frac{6}{\varepsilon}\log(2/\beta)$, we have that with probability at least $1-\frac{3}{4}\beta$, $\tilde{Q}_{k_2}(\mathbf{I}) > \tilde{T}$, which implies $i \leq k_2$.

To show $Q_i(\mathbf{I}) \geq T - \frac{6}{\varepsilon}\log(2k_2/\beta)$, we also require the following condition, which will be shown to hold with probability at least $1-\frac{\beta}{4}$. Consider each $j=1,\dots,k_2$. We have

$$\Pr\left[\tilde{Q}_j(\mathbf{I}) \geq Q_j(\mathbf{I}) + \frac{4}{\varepsilon}\log\frac{2k_2}{\beta}\right] = \Pr\left[\mathrm{Lap}\left(\frac{4}{\varepsilon}\right) \geq \frac{4}{\varepsilon}\log\frac{2k_2}{\beta}\right] \leq \frac{\beta}{4k_2}.$$

By a union bound over all j, we have that, with probability at least $1-\frac{\beta}{4}$, $\tilde{Q}_j(\mathbf{I}) < Q_j(\mathbf{I}) + \frac{4}{\varepsilon}\log(2k_2/\beta)$ for all j. By further combining with (4.2), we have $Q_i(\mathbf{I}) \geq T - \frac{6}{\varepsilon}\log(2k_2/\beta)$. □

4.1.4 The Clipped Sum Estimator

A standard idea for dealing with an unbounded domain is to clip all values into a bounded range $[l,r]$. Define

$$\mathrm{Clip}\left(X,[l,r]\right) = \begin{cases} l, & \text{if } X < l; \\ X, & \text{if } l \leq X \leq r; \\ r, & \text{if } X > r. \end{cases}$$

Let

$$\mathrm{Clip}(\mathbf{I},[l,r]) = \{\mathrm{Clip}\left(X_i,[l,r]\right) \mid X_i \in D\}.$$

Then the clipped sum estimator is

$$\mathrm{ClippedSum}(\mathbf{I},[l,r]) = \mathrm{Sum}(\mathrm{Clip}(\mathbf{I},[l,r])).$$

It is obvious that $\mathrm{ClippedSum}(\cdot,[l,r])$ has global sensitivity $r-l$. Thus, $\mathrm{ClippedSum}(\mathbf{I},[l,r]) + \mathrm{Lap}\left(\frac{r-l}{\varepsilon}\right)$ satisfies ε-DP.

4.2 Methodology

In this section, we design ε-DP mechanisms for estimating $\mathrm{Sum}(\mathbf{I})$. We will first obtain $\widetilde{\mathrm{rad}}(\mathbf{I})$, a privatized $\mathrm{rad}(\mathbf{I})$, and then invoke the clipped sum estimator. It turns out that the instance optimality ratio crucially depends on how well $\widetilde{\mathrm{rad}}(\mathbf{I})$ approximates $\mathrm{rad}(\mathbf{I})$.

4.2.1 Estimate Radius

We will show how to obtain a $\widetilde{\mathrm{rad}}(\mathbf{I})$ such that $\widetilde{\mathrm{rad}}(\mathbf{I}) \leq 2\cdot\mathrm{rad}(\mathbf{I})$ while $[-\widetilde{\mathrm{rad}}(\mathbf{I}), \widetilde{\mathrm{rad}}(\mathbf{I})]$ covers all but $O(\log\log(\mathrm{rad}(\mathbf{I})))$ elements of $\mathbf{I}$.

Let $\mathrm{Count}(D, x) = |D \cap [-x, x]|$. It is easy to see that $\mathrm{Count}(\mathbf{I}, x) - N$ has the global sensitivity 1 for any x, while $\mathrm{rad}(\mathbf{I})$ is exactly the smallest x such that $\mathrm{Count}(D, x) - N \geq 0$. Thus, a natural idea is to feed the query sequence $\mathrm{Count}(D, x) - N$ for $x = 0, 1, 2, 4, 8, \dots$ to SVT with a threshold of $T = n$. However, doing so suffers from the "late stop" problem, i.e., SVT may stop at a $\widetilde{\mathrm{rad}}(\mathbf{I})$ that is too large due to the exponential growth rate of x. On the other hand, reducing the growth rate increases the length of the query sequence, degrading the utility of SVT. Inspired by Lemma 4.1.4, we use $T = -6\log(2/\beta)/\varepsilon$ so that SVT will stop at the "right" place. The details are shown in Algorithm 7.

Algorithm 2: `EstimateRadius`.

```
  Input: $D$, $\varepsilon$, $\beta$
1 $\tilde{i} =$
    $\mathrm{SVT}\left(-\frac{6}{\varepsilon}\log(2/\beta), \varepsilon, \mathrm{Count}(D,0) - N, \mathrm{Count}(D,2^0) - N, \mathrm{Count}(D,2^1) - N, \dots\right)$;

2 if $\tilde{i} = 1$ then
3     $\widetilde{\mathrm{rad}}(\mathbf{I}) = 0$;
4 else
5     $\widetilde{\mathrm{rad}}(\mathbf{I}) = 2^{\tilde{i}-2}$;
6 end
7 return $\widetilde{\mathrm{rad}}(\mathbf{I})$;
```

The privacy of `EstimateRadius` follows from that of the SVT and the post-processing property of DP. We analyze its utility below:

Theorem 4.2.1. *For any $\mathbf{I} \in \mathbb{Z}^N$, with probability at least $1-\beta$,* `EstimateRadius` *returns*

Algorithm 3: `InfiniteDomainSum`.

```
Input: $\mathbf{I}$, $\varepsilon$, $\beta$
1 $\widetilde{\text{rad}}(\mathbf{I}) = \texttt{EstimateRadius}(D, \frac{\varepsilon}{2}, \frac{\beta}{2})$;
2 $\widetilde{\text{Sum}}(\mathbf{I}) = \text{ClippedSum}(\mathbf{I}, [-\widetilde{\text{rad}}(\mathbf{I}), \widetilde{\text{rad}}(\mathbf{I})]) + \text{Lap}\left(4 \cdot \widetilde{\text{rad}}(\mathbf{I})/\varepsilon\right)$;
3 return $\widetilde{\text{rad}}(\mathbf{I})$;
```

a $\widetilde{\text{rad}}(\mathbf{I})$ *such that* $\widetilde{\text{rad}}(\mathbf{I}) \leq 2 \cdot \text{rad}(\mathbf{I})$ *and*

$$\left| D \cap \overline{\left[-\widetilde{\text{rad}}(\mathbf{I}), \widetilde{\text{rad}}(\mathbf{I})\right]} \right| = O\left(\frac{1}{\varepsilon} \log\left(\log\left(\text{rad}(\mathbf{I})\right)/\beta\right)\right).$$

Proof. We consider two cases: $\text{rad}(\mathbf{I}) = 0$ and $\text{rad}(\mathbf{I}) \in [2^{j-1}, 2^j]$ for some $j \in \mathbb{N}$. In the first case, $\text{Count}(\mathbf{I}, 0) = N$. By Lemma 4.1.4, with probability at least $1 - \beta$, we have $\widetilde{\text{rad}}(\mathbf{I}) = 0$, and both conclusions hold.

In the second case, $\text{Count}(\mathbf{I}, 2^j) = N$. Plugging in Lemma 4.1.4 with $T = -\frac{6}{\varepsilon}\log(2/\beta)$ and $k_2 = \log_2(2^j) + 2$, we have, with probability at least $1 - \beta$,

$$\widetilde{\text{rad}}(\mathbf{I}) \leq 2^j \leq 2 \cdot \text{rad}(\mathbf{I}).$$

and

$$\text{Count}(\mathbf{I}, \widetilde{\text{rad}}(\mathbf{I})) \geq N - \frac{6}{\varepsilon}\log(2/\beta) - \frac{6}{\varepsilon}\log\left(2\left(\log_2(2^j) + 2\right)/\beta\right).$$

□

4.2.2 Sum Estimation

With a good $\widetilde{\text{rad}}(\mathbf{I})$, we can now do sum estimation over an infinite domain with clipped sum. The algorithm is shown in Algorithm 3. Its privacy follows from basic composition, while its utility guarantee directly follows Theorem 4.2.1 and tail bound of Laplace distribution:

Theorem 4.2.2. *Given* ε, β, *for any* $\mathbf{I} \in \mathbb{Z}^N$, *with probability at least* $1-\beta$, `InfiniteDomainSum` *returns a* $\widetilde{\text{Sum}}(\mathbf{I})$ *such that*

$$\left|\widetilde{\text{Sum}}(\mathbf{I}) - \text{Sum}(\mathbf{I})\right| = O\left(\frac{\text{rad}(\mathbf{I})}{\varepsilon} \log\left(\log\left(\text{rad}(\mathbf{I})\right)/\beta\right)\right).$$

Recall from Section 4.1.2, $\mathrm{rad}(\mathbf{I})$ is the instance-specific optimal lower bound. This means that `InfiniteDomainSum` is optimal with an optimality ratio of $c = O(\log\log(\mathrm{rad}(\mathbf{I}))/\varepsilon)$ for constant β. Below, we show that this c is worst-case optimal in the finite-domain case. In particular, it implies that the optimality ratio cannot be independent of $\mathbf{I}$.

Theorem 4.2.3. *For the empirical mean $\mu(\mathbf{I})$, given any ε, any integer $H \geq 1$, and any $N > \log\log_2(H)/\varepsilon$, for any ε-DP mechanism $\mathcal{M} : [H]^N \to \mathbb{R}$, there exists $\mathbf{I} \in [H]^N$, such that*

$$\mathrm{Err}(\mathcal{M}, \mathbf{I}) \geq \frac{\mathrm{rad}(\mathbf{I})}{3\varepsilon} \log\log_2(H).$$

Proof. We use a packing argument by constructing a sequence of $\log_2(H) + 1$ datasets: $\mathbf{I}^0$,$\mathbf{I}^1$,..., $\mathbf{I}^{\log_2(H)}$. $\mathbf{I}^0$ contains all 0's. For each $i = 1, \dots, \log_2(H)$, $\mathbf{I}^i$ is constructed by changing $\log\log_2(H)/\varepsilon$ number of 0's in $\mathbf{I}^0$ to 2^i. It can be verified that

$$\mathrm{Sum}(\mathbf{I}^i) = \frac{2^i}{\varepsilon} \log\log_2(H). \tag{4.4}$$

We argue that, for any ε-DP mechanism $\mathcal{M}$, there exists at least one $\mathbf{I}^i$ such that

$$\mathrm{Err}(\mathcal{M}, \mathbf{I}^i) \geq \frac{\mathrm{rad}(\mathbf{I}^i)}{3\varepsilon} \log\log_2(H) = \frac{2^i}{3\varepsilon} \log\log_2(H). \tag{4.5}$$

We prove this by contradiction. If (4.5) does not hold, then

$$\begin{aligned}
\frac{1}{3} \geq & \Pr[\mathcal{M}(\mathbf{I}^0) \neq 0] \\
\geq & \sum_{1 \leq i \leq \log_2(H)} \Pr\left[\mathcal{M}(\mathbf{I}^0) \in \left(\frac{2}{3} \cdot 2^i \cdot \frac{1}{\varepsilon} \log\log_2(H), \frac{4}{3} \cdot 2^i \cdot \frac{1}{\varepsilon} \log\log_2(H)\right)\right] \qquad (4.6) \\
\geq & \sum_{1 \leq i \leq \log_2(H)} \left(e^{-\varepsilon \log(\log_2(H))/\varepsilon} \Pr\left[\mathcal{M}(\mathbf{I}^i) \in \left(\frac{2}{3} \cdot 2^i \cdot \frac{1}{\varepsilon} \log\log_2(H), \frac{4}{3} \cdot 2^i \cdot \frac{1}{\varepsilon} \log\log_2(H)\right)\right]\right) \\
\geq & \log_2(H) \cdot \frac{1}{\log_2(H)} \cdot \frac{2}{3} \qquad (4.7) \\
= & \frac{2}{3},
\end{aligned}$$

which causes a contradiction. (4.6) is because the intervals $\left(\frac{2}{3} \cdot 2^i \cdot \frac{1}{\varepsilon} \log\log_2(H), \frac{4}{3} \cdot 2^i \cdot \frac{1}{\varepsilon} \log\log_2(H)\right)$ for all $i = 1, \dots, \log_2(H)$ are disjoint. (4.7) follows from (4.4) and the hypothesis. □

CHAPTER 5

ANSWERING SPJA QUERIES UNDER USER-DP

5.1 Preliminaries

5.1.1 Database Queries

Follow the notations defined in Section 3.1.1. $\mathbf{R}$ is a database schema and a multi-way join is

$$J := R_1(\mathbf{x}_1) \bowtie \cdots \bowtie R_n(\mathbf{x}_n), \tag{5.1}$$

Let $var(J) := \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_n$. $\mathbf{I}$ is a database instance over $\mathbf{R}$. Recall when self-joins appear, there can be repeats, i.e., $R_i = R_j$. To avoid the confusion, we define physical relation instance and logical relation instance (see Section 3.1.1 for more details). For convenience, in this chapter, for any $R \in \mathbf{R}$, the physical relation instance is $\mathbf{I}(R)$ and the logical relation instance is $\mathbf{I}(R, \mathbf{x})$, where we rename the attributes $\mathbf{I}(R)$ to $\mathbf{x}$. When there are self-joins, one physical relation instance may have multiple logical relation instances; they have the same rows but with different column (variable) names.

An JA or SJA query Q aggregates over the join results $J(\mathbf{I})$. More abstractly, let $\psi : \mathbf{dom}(var(J)) \to \mathbb{N}$ be a function that assigns non-negative integer weights to the join results, where $\mathbf{dom}(var(J))$ denotes the domain of $var(J)$. The result of evaluating Q on $\mathbf{I}$ is

$$Q(\mathbf{I}) := \sum_{q \in J(\mathbf{I})} \psi(q). \tag{5.2}$$

Note that the function ψ only depends on the query. For a counting query, $\psi(\cdot) \equiv 1$; for an aggregation query, e.g. `SUM`$(A * B)$, $\psi(q)$ is the value of $A * B$ for q. And an SJA query with an arbitrary predicate over $var(J)$ can be easily incorporated into this formulation: If some $q \in J(\mathbf{I})$ does not satisfy the predicate, we simply set $\psi(q) = 0$.

Example 5.1.1. Graph pattern counting queries can be formulated as SJA queries. Suppose we store a graph in a relational database by the schema $\mathbf{R}$ = {`Node`($\underline{\texttt{ID}}$),

$\texttt{Edge}(\texttt{src}, \texttt{dst})\}$ where `src` and `dst` are FKs referencing `ID`, then the number of length-3 paths can be counted by first computing the join

$$\texttt{Edge}(\texttt{A}, \texttt{B}) \bowtie \texttt{Edge}(\texttt{B}, \texttt{C}) \bowtie \texttt{Edge}(\texttt{C}, \texttt{D}),$$

followed by a count aggregation. Note that this also counts triangles and non-simple paths (e.g., x-y-x-z), which may or may not be considered as length-3 paths depending on the application. If not, they can be excluded by introducing a predicate (i.e., redefining ψ) $\texttt{A} \neq \texttt{C} \wedge \texttt{A} \neq \texttt{D} \wedge \texttt{B} \neq \texttt{D}$. If the graph is undirected, then the query counts every path twice, so we should divide the answer by 2. Alternatively, we may introduce the predicate $\texttt{A} < \texttt{D}$ to eliminate the double counting. □

Finally, for an SPJA query where the output variables are $\mathbf{o} \subset var(J)$, we simply replace $J(\mathbf{I})$ with $\pi_{\mathbf{o}} J(\mathbf{I})$ in (5.2). Note that we use the relational algebra semantics of a projection, where duplicates are removed. If not, the projection would not make any difference in the aggregate. In fact, it is precisely the duplicate-removal that makes SPJA queries more difficult than SJA queries in terms of optimality, as we argue in Section 5.5.

5.1.2 Differential Privacy in Relational Databases with Foreign Key Constraints (User-DP Model)

We adopt the DP policy in [57], which defines neighboring instances by taking foreign key (FK) constraints into consideration. We model all the FK relationships as a directed acyclic graph over $\mathbf{R}$ by adding a directed edge from R to R' if R has an FK referencing the PK of R'. There is a[1] designated *primary private relation* R_P and any relation that has a direct or indirect FK referencing R_P is called a *secondary private relation*. The *referencing* relationship over the tuples is defined recursively as follows: (1) any tuple $t_P \in \mathbf{I}(R_P)$ said to reference itself; (2) for $t_P \in \mathbf{I}(R_P)$, $t \in \mathbf{I}(R), t' \in \mathbf{I}(R')$, if t' references t_P, R has an FK referencing the PK of R', and the FK of t equals to the PK of t', then we say that t references t_P. Then two instances $\mathbf{I}$ and $\mathbf{I}'$ are considered neighbors if $\mathbf{I}'$ can be obtained from $\mathbf{I}$ by deleting a set of tuples, all of which reference the same tuple $t_P \in \mathbf{I}(R_P)$, or vice versa. In particular, t_P may also be deleted, in which case all tuples referencing t_P must be deleted in order to preserve the FK constraints. Finally, for a join result $q \in J(\mathbf{I})$, we say that q references $t_P \in \mathbf{I}(R_P)$ if $|t_P \bowtie q| = 1$.

[1]For most parts of the paper, we consider the case where there is only one *primary private relation* in $\mathbf{R}$; the case with multiple primary private relations is discussed in Section 5.6.

We use the notation $\mathbf{I} \sim \mathbf{I}'$ to denote two neighboring instances and $\mathbf{I} \sim_{t_P} \mathbf{I}'$ denotes that all tuples in the difference between $\mathbf{I}$ and $\mathbf{I}'$ reference the tuple $t_P \in R_P$.

In a relational database, user-DP model is equivalent to FK constraints: the users are stored in the user relation and we build FK constraints between user relation and others and add/delete one user will also add/delete all tuples referencing that.

Example 5.1.2. Consider the TPC-H schema:

$$\mathbf{R} = \{\texttt{Nation}(\underline{\texttt{NK}}), \texttt{Customer}(\underline{\texttt{CK}}, \texttt{NK}), \texttt{Order}(\underline{\texttt{OK}}, \texttt{CK}), \texttt{Lineitem}(\texttt{OK})\}.$$

If the customers are the individuals whose privacy we wish to protect, then we designate `Customer` as the primary private relation, which implies that `Order` and `Lineitem` will be secondary private relations, while `Nation` will be public. Note that once `Customer` is designated as a primary private relation, the information in `Order` and `Lineitem` is also protected since the privacy induced by `Customer` is stronger than that induced by `Order` and `Lineitem`. Alternatively, one may designate `Order` as the primary private relation, which implies that `Lineitem` will be a secondary private relation, while `Customer` and `Nation` will be public. This would result in weaker privacy protection but offer higher utility. □

Some queries, as given, may be *incomplete*, i.e., it has a variable that is an FK but its referenced PK does not appear in the query Q. The query in Example 5.1.1 is such an example. Following [57], we always make the query complete by iteratively adding those relations whose PKs are referenced to Q. The PKs will be given variables names matching the FKs. For example, for the query in Example 5.1.1, we add `Node(A)`, `Node(B)`, `Node(C)`, and `Node(D)`.

The DP policy above incorporates both edge-DP and node-DP, two commonly used DP policies for private graph analysis, as special cases. In Example 5.1.1, by designating `Edge` as the private relation (`Node` is thus public, and we may even assume it contains all possible vertex IDs), we obtain edge-DP; for node-DP, we add FK constraints from `src` and `dst` to `ID`, and designate `Node` as the primary private relation, while `Edge` becomes a secondary private relation.

After formulating the neighboring instances, we can feed this to (1.1) to define user-DP in a relational database.

5.2 Instance Optimality under User DP

Global sensitivity and worst-case optimality As mentioned in Section 1.1.1, the most standard DP mechanism is the Laplace mechanism [37], which adds $\mathrm{Lap}(\mathrm{GS}_Q)$ to the query answer. However, either the user DP or a sum aggregation makes GS_Q unbounded. The issue with the former is illustrated in the example given at the beginning of Section 1.2, where a customer may have unbounded orders.

A sum aggregation with an unbounded ψ results in the same situation. Thus, as with prior work [6, 7, 69, 62, 76, 46], we restrict to a set of instances $\mathcal{I}$ such that

$$\max_{\mathbf{I}\in\mathcal{I},\mathbf{I}'\in\mathcal{I},\mathbf{I}\sim\mathbf{I}'} |Q(\mathbf{I}) - Q(\mathbf{I}')| = \mathrm{GS}_Q, \tag{5.3}$$

where GS_Q is a parameter given in advance. For the query given at the beginning of Section 1.2, this is equivalent to assuming that a customer is allowed to have at most GS_Q orders in any instance.

For general queries, the situation is more complicated. We first consider SJA queries. Given an instance $\mathbf{I}$ and an SJA query Q, for a tuple $t_P \in \mathbf{I}(R_P)$, its *sensitivity* is

$$S_Q(\mathbf{I}, t_P) := \sum_{q\in J(\mathbf{I})} \psi(q)\mathbb{I}(q \text{ references } t_P), \tag{5.4}$$

where $\mathbb{I}(\cdot)$ is the indicator function. For SJA queries, (5.3) is equivalent to

$$\max_{\mathbf{I}\in\mathcal{I}} \max_{t_P\in\mathbf{I}(R_P)} S_Q(\mathbf{I}, t_P) = \mathrm{GS}_Q.$$

For self-join-free SJA queries, it is clear that

$$Q(\mathbf{I}) = \sum_{t_P\in R_P} S_Q(\mathbf{I}, t_P),$$

which turns the problem into a sum estimation problem (SA queries). However, when self-joins are present, this equality no longer holds since one join result q references multiple t_P's. This also implies that removing one tuple from $\mathbf{I}(R_P)$ may affect multiple $S_Q(\mathbf{I}, t_P)$'s, making the neighboring relationship more complicated than in the sum estimation problem, where two neighboring instances differ by only one datum [6, 7, 69, 62, 46].

What notion of optimality shall we use for sum aggregation queries over user-DP? The worst-case optimality is meaningless even in SA queries. Besides, as mentioned in

Section 1.1.1, instance optimality is unachievable under DP. In this chapter, we adopt the high probability error metric thus instance optimal lower bound should be reformulated as

$$\mathcal{L}_{\text{ins}}(\mathbf{I}) := \min_{M' \in \mathcal{M}} \min\{\xi : \Pr[|M'(\mathbf{I}) - Q(\mathbf{I})| \leq \xi] \geq 2/3\}.$$

Recall under tuple-DP, we adopt neighborhood optimality, a relaxed version of instance optimality where we compare M against any M' that is required to work well not just on $\mathbf{I}$, but also on its neighbors, i.e., we raise the target error from $\mathcal{L}_{\text{ins}}(\mathbf{I})$ to

$$\mathcal{L}_{\text{nbr}}(\mathbf{I}) := \min_{M' \in \mathcal{M}} \max_{\mathbf{I}' : \mathbf{I} \sim \mathbf{I}'} \min\{\xi : \Pr[|M'(\mathbf{I}') - Q(\mathbf{I}')| \leq \xi] \geq 2/3\},$$

and Vadhan [77] observes that $\mathcal{L}_{\text{nbr}}(\mathbf{I}) \geq \text{LS}_Q(\mathbf{I})/2$.

However, it has an issue for SJA queries in a database with FK constraints, namely, under user-DP: For any $\mathbf{I}$, we can add a t_P to $\mathbf{I}(R_P)$ together with tuples in the secondary private relations all referencing t_P, obtaining an $\mathbf{I}'$ such that $S_Q(\mathbf{I}', t_P) = \text{GS}_Q$, i.e., $\text{LS}_Q(\cdot) \equiv \text{GS}_Q$. This means that this relaxed instance optimality degenerates into worst-case optimality. This is also why smooth sensitivity, including all its efficiently computable versions [67, 48, 29, 31], will not have better utility than the naive Laplace mechanism on databases with FK constraints, since they are all no lower than the local sensitivity.

The reason why the above relaxation from instance optimality to neighborhood optimality is "too much" is that we require M' to work well on any neighbor $\mathbf{I}'$ of $\mathbf{I}$. Under the neighborhood definition with FK constraints, this means that $\mathbf{I}'$ can be any instance obtained from $\mathbf{I}$ by adding a tuple t_P and *arbitrary* tuples referencing t_P in the secondary private relations. This is too high a requirement for M', hence too low an optimality notion for M.

To address the issue, we revise $\mathcal{L}_{\text{nbr}}(\cdot)$ to

$$\mathcal{L}_{\text{d-nbr}}(\mathbf{I}) := \min_{M' \in \mathcal{M}} \max_{\mathbf{I}' : \mathbf{I} \sim \mathbf{I}', \mathbf{I}' \subseteq \mathbf{I}} \min\{\xi : \Pr[|M'(\mathbf{I}') - Q(\mathbf{I}')| \leq \xi] \geq 2/3\},$$

namely, we require M' to work well only on $\mathbf{I}'$ and its *down-neighbors*, which can be obtained only by removing a tuple t_P already in $\mathbf{I}(R_P)$ and all tuples referencing t_P. A mechanism M is *down-neighborhood optimal* if

$$\Pr[|M(\mathbf{I}) - Q(\mathbf{I})| \leq c \cdot \mathcal{L}_{\text{ins}}(\mathbf{I})] \geq 2/3$$

for every $\mathbf{I}$, where c is called the *optimality ratio*.

Using the same argument from [77], we have $\mathcal{L}_{\text{d-nbr}}(\mathbf{I}) \geq \text{DS}_Q(\mathbf{I})/2$, where

$$\text{DS}_Q(\mathbf{I}) := \max_{\mathbf{I}',\mathbf{I}\sim\mathbf{I},\mathbf{I}'\subseteq\mathbf{I}} |Q(\mathbf{I}) - Q(\mathbf{I}')| = \max_{t_P\in\mathbf{I}(R_P)} S_Q(\mathbf{I}, t_P) \tag{5.5}$$

is the *downward sensitivity* of $\mathbf{I}$. Thus, $\text{DS}_Q(\mathbf{I})$ is a per-instance lower bound, which can be used to replace $\mathcal{L}_{\text{inc}}(\mathbf{I})$ in (5.2) in the definition of instance-optimal DP mechanisms.

5.3 R2T: Instance-optimal Truncation

Our instance-optimal truncation mechanism, *Race-to-the-Top (R2T)*, can be used in combination with any truncation method $Q(\mathbf{I}, \tau)$, which is a function $Q : \mathcal{I} \times \mathbb{N} \to \mathbb{N}$ with the following properties:

(1) For any τ, the global sensitivity of $Q(\cdot, \tau)$ is at most τ.

(2) For any τ, $Q(\mathbf{I}, \tau) \leq Q(\mathbf{I})$.

(3) For any $\mathbf{I}$, there exists a non-negative integer $\tau^*(\mathbf{I}) \leq \text{GS}_Q$ such that for any $\tau \geq \tau^*(\mathbf{I})$, $Q(\mathbf{I}, \tau) = Q(\mathbf{I})$.

We describe various choices for $Q(\mathbf{I}, \tau)$ depending on the DP policy and whether the query contains self-joins and/or projections in the subsequent sections. Intuitively, such a $Q(\mathbf{I}, \tau)$ gives a stable (property (1)) underestimate (property (2)) of $Q(\mathbf{I})$, while reaches $Q(\mathbf{I})$ for τ sufficiently large (property (3)). Note that $Q(\mathbf{I}, \tau)$ itself is not DP. To make it DP, we can add $\text{Lap}(\tau/\varepsilon)$, which would turn it into an ε-DP mechanism by property (1). The issue, of course, is how to set τ. The basic idea of R2T is to try geometrically increasing values of τ and somehow pick the "winner" of the race.

Assuming such a $Q(\mathbf{I}, \tau)$, R2T is works as follows. For a probability[2] β, we first compute[3]

$$\begin{aligned}\tilde{Q}(\mathbf{I}, \tau^{(j)}) :=& Q(\mathbf{I}, \tau^{(j)}) + \text{Lap}\left(\log(\text{GS}_Q)\frac{\tau^{(j)}}{\varepsilon}\right) \\ & - \log(\text{GS}_Q)\ln\left(\frac{\log(\text{GS}_Q)}{\beta}\right)\cdot\frac{\tau^{(j)}}{\varepsilon},\end{aligned} \tag{5.6}$$

[2]The probability β only concerns about the utility but not privacy.

[3]log has base 2 and ln has base e.

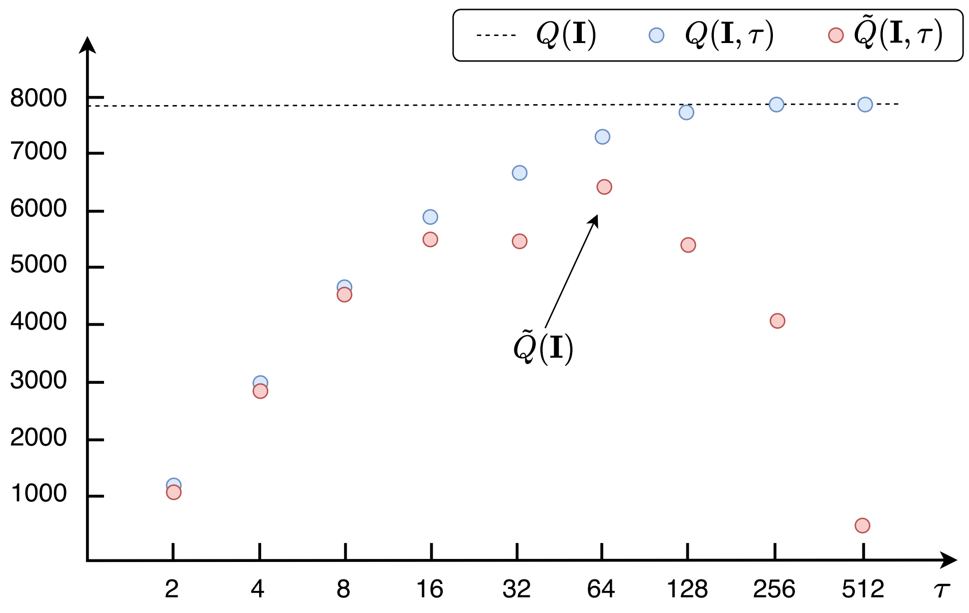


Figure 5.1: An illustration of R2T.

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

$$\tilde{Q}(\mathbf{I}) := \max\left\{\max_j \tilde{Q}(\mathbf{I}, \tau^{(j)}), Q(\mathbf{I}, 0)\right\}. \tag{5.7}$$

The privacy of R2T is straightforward: Since $Q(\mathbf{I}, \tau^{(j)})$ has global sensitivity at most $\tau^{(j)}$, and the third term of (5.6) is independent of $\mathbf{I}$, each $\tilde{Q}(\mathbf{I}, \tau^{(j)})$ satisfies $\frac{\varepsilon}{\log(\mathrm{GS}_Q)}$-DP by the standard Laplace mechanism. Collectively, all the $\tilde{Q}(\mathbf{I}, \tau^{(j)})$'s satisfy ε-DP by the basic composition theorem (defined in Section 4.1.2). Finally, returning the maximum preserves DP by the post-processing property of DP.

Utility analysis For some intuition on why R2T offers good utility, please see Figure 5.1. By property (2) and (3), as we increase τ, $Q(\mathbf{I}, \tau)$ gradually approaches the true answer $Q(\mathbf{I})$ from below and reaches $Q(\mathbf{I}, \tau) = Q(\mathbf{I})$ when $\tau \geq \tau^*(\mathbf{I})$. However, we cannot use $Q(\mathbf{I}, \tau)$ or $\tau^*(\mathbf{I})$ directly as this would violate DP. Instead, we only get to see $\tilde{Q}(\mathbf{I}, \tau)$, which is masked with the noise of scale proportional to τ. We thus face a dilemma, that the closer we get to $Q(\mathbf{I})$, the more uncertain we are about the estimate $\tilde{Q}(\mathbf{I}, \tau)$. To get out of the dilemma, we shift $Q(\mathbf{I}, \tau)$ down by an amount that equals to the scale of the noise (if ignoring the $\log\log$ factor). This penalty for $\tilde{Q}(\mathbf{I}, \hat{\tau})$, where $\hat{\tau}$ is the smallest power of 2 above $\tau^*(\mathbf{I})$, will be on the same order as $\tau^*(\mathbf{I})$, so it will not affect its error by more than a constant factor, while taking the maximum ensures that the winner is at least as good as $\tilde{Q}(\mathbf{I},\hat{\tau})$. Meanwhile, the extra $\log\log$ factor ensures that no $\tilde{Q}(\mathbf{I},\tau)$ overshoots the target. Below, we formalize the intuition.

Theorem 5.3.1. *On any instance* $\mathbf{I}$*, with probability at least* $1-\beta$*, we have*

$$Q(\mathbf{I}) - 4\log(\mathrm{GS}_Q)\ln\left(\frac{\log(\mathrm{GS}_Q)}{\beta}\right)\frac{\tau^*(\mathbf{I})}{\varepsilon} \le \tilde{Q}(\mathbf{I}) \le Q(\mathbf{I}).$$

Proof. It suffices to show that each inequality holds with probability at least $1-\frac{\beta}{2}$. For the second inequality, since $Q(\mathbf{I},0) \le Q(\mathbf{I})$, we just need to show that $\max_j \tilde{Q}(\mathbf{I},\tau^{(j)}) \le Q(\mathbf{I})$. By a union bound, it suffices to show that $\tilde{Q}(\mathbf{I},\tau) \le Q(\mathbf{I})$ with probability at most $\frac{\beta}{2\log(\mathrm{GS}_Q)}$ for each τ. This easily follows from property (2) of $Q(\mathbf{I},\tau)$ and the tail bound of the Laplace distribution:

$$\begin{aligned}
&\mathsf{Pr}[\tilde{Q}(\mathbf{I},\tau) > Q(\mathbf{I})]\\
\le&\mathsf{Pr}[\tilde{Q}(\mathbf{I},\tau) > Q(\mathbf{I},\tau)]\\
=&\mathsf{Pr}\left[\mathrm{Lap}\left(\log(\mathrm{GS}_Q)\frac{\tau}{\varepsilon}\right) > \log(\mathrm{GS}_Q)\ln\left(\frac{\log(\mathrm{GS}_Q)}{\beta}\right)\cdot\frac{\tau}{\varepsilon}\right]\\
=&\frac{\beta}{2\log(\mathrm{GS}_Q)}.
\end{aligned}$$

For the first inequality, we discuss two cases $\tau^*(\mathbf{I}) = 0$ and $\tau^*(\mathbf{I}) \in [2^{j-1}, 2^j]$ for some $j \ge 1$. For the first case, by property (3) of $Q(\mathbf{I},\tau)$, $Q(\mathbf{I},0) = Q(\mathbf{I})$. Therefore, $\tilde{Q}(\mathbf{I}) \ge Q(\mathbf{I},0) = Q(\mathbf{I})$. Below we discuss the second case where $\tau^*(\mathbf{I}) \in [2^{j-1}, 2^j]$. Note that $2^j \le 2\tau^*(\mathbf{I})$. Let $\hat{\tau} = 2^j$. By the tail bound on the Laplace distribution, with probability at least $1-\frac{\beta}{2}$, we have

$$\begin{aligned}
\tilde{Q}(\mathbf{I},\hat{\tau}) \ge& Q(\mathbf{I},2^j) - 2\log(\mathrm{GS}_Q)\ln\left(\frac{\log(\mathrm{GS}_Q)}{\beta}\right)\frac{2^j}{\varepsilon}\\
=& Q(\mathbf{I}) - 2\log(\mathrm{GS}_Q)\ln\left(\frac{\log(\mathrm{GS}_Q)}{\beta}\right)\frac{2^j}{\varepsilon} \qquad (5.8)\\
\ge& Q(\mathbf{I}) - 4\log(\mathrm{GS}_Q)\ln\left(\frac{\log(\mathrm{GS}_Q)}{\beta}\right)\frac{\tau^*(\mathbf{I})}{\varepsilon}. \qquad (5.9)
\end{aligned}$$

Note that (5.8) follows the third property of $Q(\mathbf{I},\tau)$, and (5.9) is because $2^j \le 2\tau^*(\mathbf{I})$. Finally, since $\tilde{Q}(\mathbf{I}) = \max_j \tilde{Q}(\mathbf{I},\tau^{(j)}) \ge \tilde{Q}(\mathbf{I},\hat{\tau})$, the first inequality also holds with probability at least $1-\frac{\beta}{2}$. □

5.4 Truncation for SJA Queries

In this section, we will design a $Q(\mathbf{I},\tau)$ with $\tau^*(\mathbf{I}) = \mathrm{DS}_Q(\mathbf{I})$ for SJA queries. Plugged into Theorem 5.3.1 with $\beta = 1/3$ and the definition of instance optimality, this turns R2T into an instance-optimal DP mechanism with an optimality ratio of $O(\log(\mathrm{GS}_Q)\log\log(\mathrm{GS}_Q)/\varepsilon)$.

For self-join-free SJA queries, each join result $q \in J(\mathbf{I})$ references only one tuple in R_P. Thus, the tuples in R_P are independent, i.e., removing one does not affect the sensitivities of others. This means that naive truncation (i.e., removing all $S_Q(\mathbf{I}, t_P) > \tau$ and then summing up the rest) is a valid $Q(\mathbf{I},\tau)$ that satisfies the 3 properties required by R2T with $\tau^*(\mathbf{I}) = \mathrm{DS}_Q(\mathbf{I})$.

When there are self-joins, naive truncation does not satisfy property (1), as illustrated in Example 1.2.1, where all $S_Q(\mathbf{I}, t_P)$'s in two neighboring instances may differ. Below we generalize the LP-based mechanism for graph pattern counting [55] to arbitrary SJA queries, and show that it satisfies the 3 properties with $\tau^*(\mathbf{I}) = \mathrm{DS}_Q(\mathbf{I})$.

Given a SJA query Q and instance $\mathbf{I}$, recall that $Q(\mathbf{I}) = \sum_{q\in J(\mathbf{I})} \psi(q)$, where $J(\mathbf{I})$ is the join results. For $k \in [|J(\mathbf{I})|]$, let $q_k(\mathbf{I})$ be the kth join result. For each $j \in [|\mathbf{I}(R_P)|]$, let $t_j(\mathbf{I})$ be the jth tuple in $\mathbf{I}(R_P)$. We use $C_j(\mathbf{I})$ to denote (the indices of) the set of join results that reference $t_j(\mathbf{I})$. More precisely,

$$C_j(\mathbf{I}) := \{k : q_k(\mathbf{I}) \text{ references } t_j(\mathbf{I})\}. \tag{5.10}$$

For each $k \in [|J(\mathbf{I})|]$, introduce a variable u_k, which represents the weight assigned to the join result $q_k(\mathbf{I})$. We return the optimal solution of the following LP as $Q(\mathbf{I},\tau)$:

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

ex:self-join-queries

Lemma 5.4.1. *For SJA queries, the $Q(\mathbf{I},\tau)$ defined above satisfies the 3 properties required by R2T with $\tau^*(\mathbf{I}) = DS_Q(\mathbf{I})$.*

Proof. Property (2) easily follows from the constraint $u_k \le \psi(q_k(\mathbf{I}))$. For property (3), observe that for SJA queries, for any $j \in [|\mathbf{I}(R_P)|]$, $S_Q(\mathbf{I}, t_j(\mathbf{I})) = \sum_{k\in C_j(\mathbf{I})} \psi(q_k(\mathbf{I}))$. So when $\tau \geq \mathrm{DS}_Q(\mathbf{I})$, all constraints $\sum_{k \in C_j(\mathbf{I})} u_k \leq \tau$ are satisfied automatically and we can set $u_k = \psi(q_k(\mathbf{I}))$ for all k.

Below, we prove property (1), i.e., for any $\mathbf{I} \sim \mathbf{I}'$, $Q(\mathbf{I}, \tau)$ and $Q(\mathbf{I}', \tau)$ differ by at most τ. W.l.o.g., assume $\mathbf{I} \subseteq \mathbf{I}'$. It is clear that $J(\mathbf{I}) \subseteq J(\mathbf{I}')$, and we order the join results in $J(\mathbf{I}')$ in such a way that the extra join results are at the end. This means that the two LPs on $\mathbf{I}$ and $\mathbf{I}'$ share common variables $u_1, \dots, u_{J(\mathbf{I})}$, while the latter has some extra variables $u_{J(\mathbf{I})+1}, \dots, u_{J(\mathbf{I}')}$. Each constraint $\sum_{k \in C_j(\mathbf{I})} u_k \leq \tau$ in the LP on $\mathbf{I}$ has a counterpart $\sum_{k \in C_j(\mathbf{I}')} u_k \leq \tau$ in the LP on $\mathbf{I}'$, where $C_j(\mathbf{I}) \subseteq C_j(\mathbf{I}')$. Let t_{j^*} be the tuple in $\mathbf{I}'(R_P)$ that all tuples in $\mathbf{I}' - \mathbf{I}$ reference. Note that t_{j^*} may or may not appear in $\mathbf{I}$. But in either case, the LP on $\mathbf{I}'$ has a constraint $\sum_{k \in C_{j^*}(\mathbf{I}')} u_k \leq \tau$ and $C_{j^*}(\mathbf{I}')$ contains all the extra variables in the LP on $\mathbf{I}'$.

Let $\{u_k^*(\mathbf{I})\}_k$ be the optimal solution of the LP on $\mathbf{I}$. We extend it to a solution $\{u_k(\mathbf{I}')\}_k$ of the LP on $\mathbf{I}'$, by setting $u_k(\mathbf{I}') = u_k^*(\mathbf{I})$ for $k \leq |J(\mathbf{I})|$ and $u_k(\mathbf{I}') = 0$ for all $k > |J(\mathbf{I})|$. It is clear that $\{u_k(\mathbf{I}')\}_k$ is a valid solution of the LP on $\mathbf{I}'$, so we have

$$Q(\mathbf{I}', \tau) \geq \sum_k u_k(\mathbf{I}') = \sum_k u^*(\mathbf{I}) = Q(\mathbf{I}, \tau).$$

For the other direction, let $\{u_k^*(\mathbf{I}')\}_k$ be an optimal solution of the LP on $\mathbf{I}'$. We cut it down to a solution $\{u_k(\mathbf{I})\}_k$ of the LP on $\mathbf{I}$, by setting $u_k(\mathbf{I}) = u_k^*(\mathbf{I}')$ for $k \leq |J(\mathbf{I})|$ while ignoring all $u_k^*(\mathbf{I}')$ for $k > |J(\mathbf{I})|$. It is clear that $\{u_k(\mathbf{I})\}_k$ is a valid solution of the LP on $\mathbf{I}$, so we have

$$Q(\mathbf{I}, \tau) \geq \sum_k u_k(\mathbf{I}) \geq \sum_k u^*(\mathbf{I}') - \tau = Q(\mathbf{I}', \tau) - \tau,$$

where the second inequality follows from the observation that the constraint $\sum_{k \in C_{j^*}(\mathbf{I}')} u_k \leq \tau$ in the LP on $\mathbf{I}'$ implies that the sum of the ignored $u_k^*(\mathbf{I}')$'s is at most τ. □

Example 5.4.1. We now give a step-by-step example to show how this truncation method works together with R2T. Consider the problem of edge counting under node-DP, which corresponds to the SJA query

$$Q := |\sigma_{\texttt{ID1}<\texttt{ID2}}(\texttt{Node(ID1)} \bowtie \texttt{Node(ID2)} \bowtie \texttt{Edge(ID1, ID2)})|$$

on the graph data schema introduced in Example 5.1.1. Note that in SQL, the query

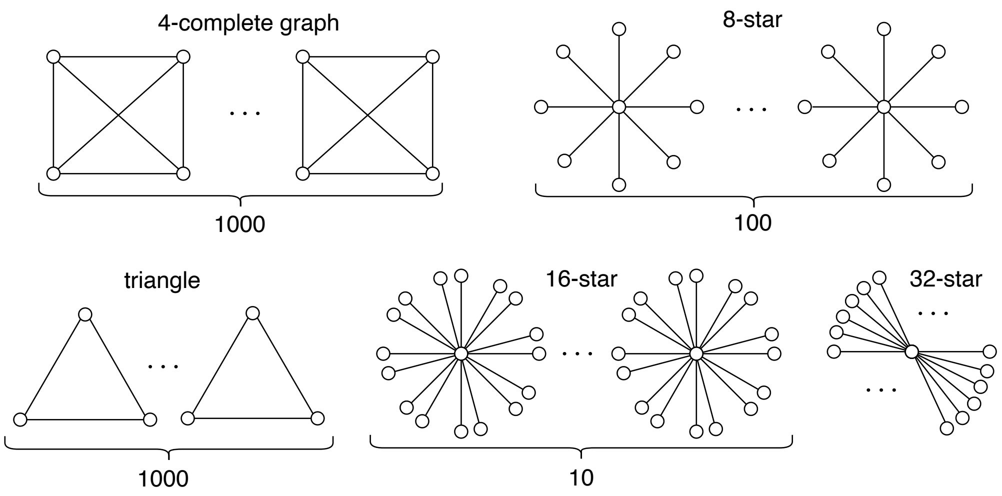


Figure 5.2: Example of edge counting.

would be written as

```
SELECT count(*) FROM Node AS Node1, Node AS Node2, Edge
WHERE Edge.src = Node1.ID AND Edge.dst = Node2.ID
          AND Node1.ID < Node2.ID
```

Suppose we set $\mathrm{GS}_Q = 2^8 = 256$. For this particular Q, this means the maximum degree of any node in any instance $\mathbf{I} \in \mathcal{I}$ is 256. We set $\beta = 0.1$ and $\varepsilon = 1$.

Now, suppose we are given an $\mathbf{I}$ containing 8103 nodes, which form 1000 triangles, 1000 4-cliques, 100 8-stars, 10 16-stars, and one 32-star as shown in Figure 5.2. The true query result is

$$Q(\mathbf{I}) = 3 \times 1000 + 6 \times 1000 + 8 \times 100 + 16 \times 10 + 32 = 9992.$$

We run R2T with $\tau^{(j)} = 2^j$ for $j = 1, \ldots, 8$. For each $\tau = \tau^{(j)}$, we assign a weight $u_k \in [0, 1]$ to each join result (i.e., an edge) that satisfies the predicate `ID1` $<$ `ID2`. To calculate $Q(\mathbf{I}, \tau)$, we can consider the LP on each clique/star separately. For a triangle, the optimal LP solution always assigns $u_k = 1$ for each edge. For each 4-clique, it assigns 2/3 to each edge for $\tau = 2$ and 1 for $\tau \geq 4$. For each k-star, the LP optimal solution is

$\min\{k, \tau\}$. Thus, the optimal LP solutions are

$$\begin{aligned}
Q(\mathbf{I}, 2) &= 1 \times 3000 + \frac{2}{3} \times 6000 + 2 \times 100 + 2 \times 10 + 2 \times 1 = 7222, \\
Q(\mathbf{I}, 4) &= 1 \times 3000 + 1 \times 6000 + 4 \times 100 + 4 \times 10 + 4 \times 1 = 9444, \\
Q(\mathbf{I}, 8) &= 1 \times 3000 + 1 \times 6000 + 8 \times 100 + 8 \times 10 + 8 \times 1 = 9888, \\
Q(\mathbf{I}, 16) &= 1 \times 3000 + 1 \times 6000 + 8 \times 100 + 16 \times 10 + 16 \times 1 = 9976.
\end{aligned}$$

In addition, we have $Q(\mathbf{I}, 0) = 0$ and $Q(\mathbf{I}, \tau) = 9992$ for $\tau \geq 32$. Finally, we plug all the $Q(\mathbf{I}, \tau)$'s into (5.6) and (5.7) to obtain the final output. $\square$

5.5 Truncation for SPJA Queries

A negative result The correctness of the LP-based truncation method relies on a key property of SJA queries, that removing t_P will always reduce $Q(\mathbf{I})$ by $S_Q(\mathbf{I}, t_P)$, which is the contribution of t_P to $Q(\mathbf{I})$. Unfortunately, the projection operator violates this property, as illustrated in the following example.

Example 5.5.1. Consider the query

$$Q := |\pi_{x_2}(R_1(x_1) \bowtie R_2(x_1, x_2))|,$$

where R_1 is the primary private relation, and R_2 is a secondary relation. Consider the following instance $\mathbf{I}$: Set $\mathbf{I}(R_1) = \{(a_1), (a_2)\}$, $\mathbf{I}(R_2) = \{(a_i, b_j) : i \in [2], j \in [m]\}$. Both (a_1) and (a_2) contribute m to $Q(\mathbf{I})$ but their contributions "overlap", thus removing either will not affect the query result, i.e., $\mathrm{DS}_Q(\mathbf{I}) = 0$. $\square$

Intuitively, a projection reduces the query answer, hence its sensitivity, so it requires less noise. However, it makes achieving instance optimality harder because the optimality target, $\mathrm{DS}_Q(\mathbf{I})$, may get a lot smaller, as illustrated in the example above. In particular, the second equality in (5.5) no longer holds (the first equality is the definition of $\mathrm{DS}_Q(\mathbf{I})$), and $\mathrm{DS}_Q(\mathbf{I})$ may be smaller than any $S_Q(\mathbf{I}, t_P)$. We formalize this intuition with the following negative result:

Theorem 5.5.1. *Let Q be the query in Example 5.5.1. For any GS_Q, there is a set of instances $\mathcal{I}$ with global sensitivity GS_Q such that, for any functions $M, f : \mathcal{I} \to \mathbb{R}$, if $\Pr[|M(\mathbf{I}) - Q(\mathbf{I})| \leq f(\mathbf{I}) \cdot \mathrm{DS}_Q(\mathbf{I})] \geq 2/3$, then M is not ε-DP for any $\varepsilon < \frac{1}{2}\ln(\mathrm{GS}_Q/2)$.*

Proof. We build the set of instances $\mathcal{I}$ as follows. First, put the empty instance $\mathbf{I}_0$ into $\mathcal{I}$. Then, for any $m \in [\mathrm{GS}_Q]$, construct an $\mathbf{I}_m$ with $\mathbf{I}_m(R_1) = \{(a_1), (a_2)\}$, $\mathbf{I}_m(R_2) = \{(a_i, b_j) : i \in [2], j \in [m]\}$. Note that $Q(\mathbf{I}_m) = m$, and $\mathrm{DS}_Q(\mathbf{I}_m) = 0$ since removing either (a_1) or (a_2) will not affect the query result. Finally, for each $\mathbf{I}_m$, remove (a_1) (and all referencing tuples) and add the resulting instance to $\mathcal{I}$. It can be verified that the global sensitivity of $\mathcal{I}$ is GS_Q. Meanwhile, for any $m \in [\mathrm{GS}_Q]$, $\mathbf{I}_m$ and $\mathbf{I}_0$ are 2-hop neighbors, so if M is ε-DP, then

$$\Pr[M(\mathbf{I}_m) = y] \le e^{2\varepsilon} \Pr[M(\mathbf{I}_0) = y],$$

for any y, by the *group privacy* property of DP [37].

The instance-optimality guarantee implies that for every $m \in [\mathrm{GS}_Q]$,

$$\Pr[M(\mathbf{I}_m) = m] \ge 2/3.$$

Consider $\mathbf{I}_0$. On the one hand,

$$\Pr[M(\mathbf{I}_0) \ne 0] \le 1/3. \tag{5.11}$$

On the other hand,

$$\begin{aligned}\Pr[M(\mathbf{I}_0) \ne 0] &\ge \Pr[M(\mathbf{I}_0) = 1] + \cdots + \Pr[M(\mathbf{I}_0) = \mathrm{GS}_Q] \\ &\ge \sum_{m=1}^{\mathrm{GS}_Q} e^{-2\varepsilon} \Pr[M(\mathbf{I}_m) = m] \\ &\ge \sum_{m=1}^{\mathrm{GS}_Q} e^{-2\varepsilon} \cdot \frac{2}{3} = \frac{2\mathrm{GS}_Q}{3e^{2\varepsilon}},\end{aligned}$$

which contradicts (5.11) when $\varepsilon < \frac{1}{2}\ln(\mathrm{GS}_Q/2)$. □

Indirect sensitivity Recall the definition of $S_Q(\mathbf{I}, t_P)$ as in (5.4). However, for an SPJA query, we have $Q(\mathbf{I}) = \sum_{q \in \pi_{\mathbf{o}} J(\mathbf{I})} \psi(q)$ instead of $Q(\mathbf{I}) = \sum_{q \in J(\mathbf{I})} \psi(q)$ thus (5.5) no longer holds. This means that, while $S_Q(\mathbf{I}, t_P)$ is still the contribution of t_P to $Q(\mathbf{I})$, it is "indirect": The overlapping contributions should be counted only once due to the projection operator removing duplicates.

We now define the *indirect sensitivity* for an instance $\mathbf{I}$:

$$\mathrm{IS}_Q(\mathbf{I}) = \max_{t_P \in \mathbf{I}(R_P)} S_Q(\mathbf{I}, t_P).$$

It should be clear that $\mathrm{IS}_Q(\mathbf{I}) \geq DS_Q(\mathbf{I})$ due to the overlapping; in the extreme case shown in Example 5.5.1, we have $\mathrm{IS}_Q(\mathbf{I}) = m$ but $\mathrm{DS}_Q(\mathbf{I}) = 0$. Below we give a truncation method for SPJA queries with $\tau^*(\mathbf{I}) = \mathrm{IS}_Q(\mathbf{I})$. When plugged into R2T, this yields a DP mechanism with error $O(\log(\mathrm{GS}_Q)\log\log(\mathrm{GS}_Q)\mathrm{IS}_Q(\mathbf{I})/\varepsilon)$. This is not instance-optimal, which is unachievable by Theorem 5.5.1 anyway. Note that for SJA queries, we have $\mathbf{o} = var(J)$, and $\mathrm{DS}_Q(\mathbf{I}) = \mathrm{IS}_Q(\mathbf{I})$ in this case.

Truncation method We modify the LP-based truncation method from Section 5.4 to handle SPJA queries. Let $p_l(\mathbf{I})$ be the l-th result in $\pi_{\mathbf{o}}J(\mathbf{I})$, $q_k(\mathbf{I})$ the k-th result in $J(\mathbf{I})$. To formalize the relationship of the query results before and after the projection, we use $D_l(\mathbf{I})$ to denote (the indices of) the join results corresponding to the projected result $p_l(\mathbf{I})$, i.e.,

$$D_l(\mathbf{I}) := \{j : p_l = \pi_{\mathbf{o}} q_j(\mathbf{I})\},$$

while $C_j(\mathbf{I})$ is still defined as in (5.10). Then $S_Q(\mathbf{I}, t_j)$ can be rewritten as

$$S_Q(\mathbf{I}, t_j) = \sum_{k \in C_j(\mathbf{I})} \psi(q_k(\mathbf{I})).$$

Now, we define a new LP. For each $l \in [|\pi_{\mathbf{o}}J(\mathbf{I})|]$, we introduce a new variable $v_l \in [0, \psi(p_l(\mathbf{I}))]$, which represents the weight assigned to the projected result $p_l(\mathbf{I})$. For each $k \in [|J(\mathbf{I})|]$, we still use a variable $u_k(\mathbf{I}) \in [0, \psi(q_k(\mathbf{I}))]$ to represent the weight assigned to $q_k(\mathbf{I})$. We keep the same truncation constraints on the u_k's, while adding the constraint that a the weight of a projected result should not exceed the total weights of all its corresponding join results. Then we try to maximize the projected results. More precisely, the new LP is

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

We can show that this modified LP yields a valid truncation method for SPJA queries:

Lemma 5.5.1. *For SPJA queries, the $Q(\mathbf{I}, \tau)$ defined above satisfies the 3 properties required by R2T with $\tau^*(\mathbf{I}) = \mathrm{IS}_Q(\mathbf{I})$.*

Proof. First, same as SJA queries, property (2) holds due to the constraint $v_l \leq \psi(p_l(\mathbf{I}))$. For property (3), we have $S_Q(\mathbf{I}, t_j) = \sum_{k \in C_j(\mathbf{I})} \psi(q_k(\mathbf{I}))$. Then with same argument as in the proof of Lemma 5.4.1, we can show that the property holds with $\tau^*(\mathbf{I}) = \mathrm{IS}_Q(\mathbf{I})$. Finally consider property (1). For any $\mathbf{I} \sim \mathbf{I}'$, $\mathbf{I} \subseteq \mathbf{I}'$, it is easy to see that $J(\mathbf{I}) \subseteq J(\mathbf{I}')$ and all different projection results are in C_{j^*} for some $j^* \in [|\mathbf{I}(R_P)|]$. Then the same line of reasoning as in the proof of Lemma 5.4.1 proves property (1). $\square$

5.6 Multiple Primary Private Relations

Now we consider the case with $k \geq 2$ primary private relations $R_P^1, \dots, R_P^k$. In this case, two instances are considered neighbors if one can be obtained from the other by deleting a set of tuples, all of which reference the same tuple that belongs to some $R_P^i, i \in [k]$. We reduce it to the case with only one primary private relation as follows. Add a new column `ID` to every $\mathbf{I}(R_P^i), i \in [k]$, and assign unique identifiers to all tuples in these relations. Next, we construct a new relation $R_P(\texttt{ID})$, whose physical instance $\mathbf{I}(R_P)$ consists of all these identifiers. For each R_P^i, we add an FK constraint from its `ID` column to reference the `ID` column of R_P. Note that this FK reference relationship is actually a bijection between the `ID` column in R_P and all the identifiers in the primary private relations. Now, we designate R_P as the only primary private relation, while $R_P^i, i \in [k]$ all become secondary private relations. The original secondary private relations, i.e., those having FK references to the R_P^i's directly or indirectly, are still secondary private relations.

It is not hard to see that (1) the query answer is not affected by this schema change; (2) two instances in the original schema are neighbors if and only if they are neighbors in the new schema; and (3) the join results that reference any tuple $t \in \mathbf{I}(R_P^i), i \in [k]$ are the same as those that reference $t_P \in \mathbf{I}(R_P)$, where t_P and t have the same identifier. Thus, both the privacy and utility guarantees of our algorithm continue to hold.

Finally, it is worth pointing out that the reduction above is conceptual; in the actual implementation, there is no need to construct the new primary private relation and the additional `ID` columns, as illustrated in Example 5.7.1 of the next section.

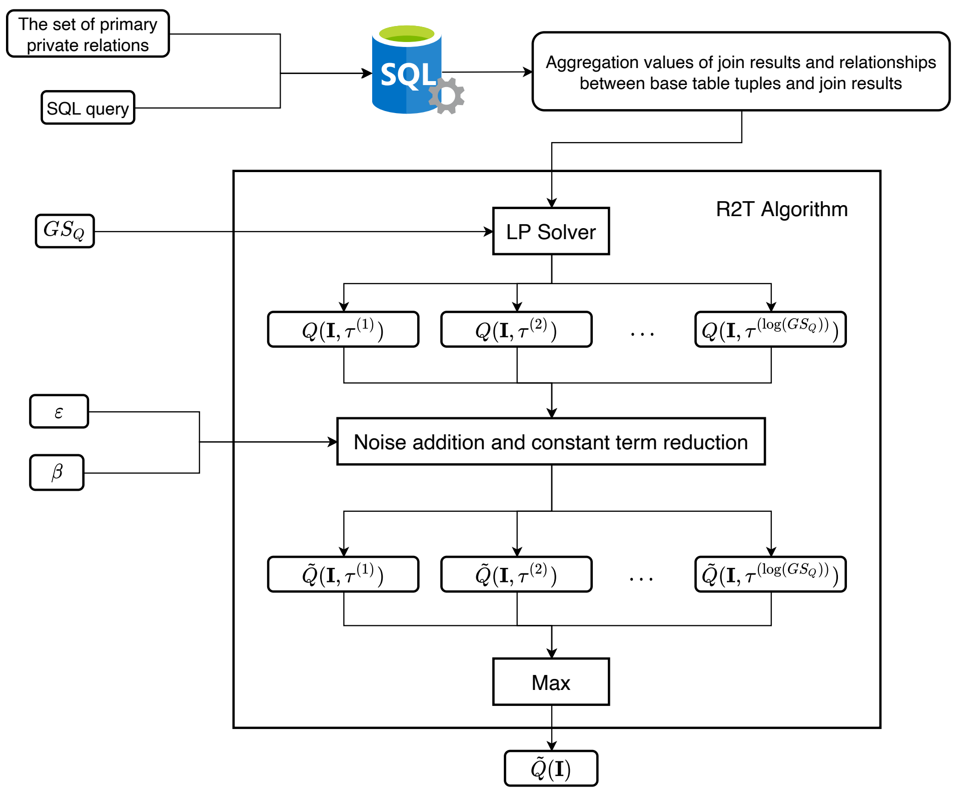

The set of primary private relations
SQL query
SQL
Aggregation values of join results and relationships between base table tuples and join results
R2T Algorithm
GS_Q
LP Solver
$Q(\mathbf{I}, \tau^{(1)})$
$Q(\mathbf{I}, \tau^{(2)})$
$\cdots$
$Q(\mathbf{I}, \tau^{(\log(GS_Q))})$
ε
β
Noise addition and constant term reduction
$\tilde{Q}(\mathbf{I}, \tau^{(1)})$
$\tilde{Q}(\mathbf{I}, \tau^{(2)})$
$\cdots$
$\tilde{Q}(\mathbf{I}, \tau^{(\log(GS_Q))})$
Max
$\tilde{Q}(\mathbf{I})$

Figure 5.3: System structure.

$$\begin{tikzcd}
 & & \text{lineitem} \arrow[dl] \arrow[dr] & & \\
 & \text{partsupp} \arrow[dl] \arrow[d] & & \text{orders} \arrow[d] & \\
\text{part} & \text{supplier} \arrow[dr] & & \text{customer} \arrow[dl] & \\
 & & \text{nation} \arrow[d] & & \\
 & & \text{region} & &
\end{tikzcd}$$

Figure 5.4: The foreign-key graph of TPC-H schema.

5.7 System Implementation

Based on the R2T algorithm, we have implemented a system on top of PostgreSQL and CPLEX. The system structure is shown in Figure 5.3. The input to our system is any SPJA query written in SQL, together with a designated primary private relation R_P (interestingly, while R2T satisfies the DP policy with FK constraints, the algorithm itself does not need to know the PK-FK constraints).

The system supports `SUM` and `COUNT` aggregation. Our SQL parser first unpacks the aggregation into a reporting query so as to find $\psi(q_k(\mathbf{I}))$ for each join result, as well as $C_j(\mathbf{I})$, which stores the referencing relationships between tuples in $\mathbf{I}(R_P)$ and $J(\mathbf{I})$.

Example 5.7.1. Suppose we use the TPC-H schema (shown in Figure 5.4), where we designate `Supplier` and `Customer` as primary private relations. Consider the following query:

```
SELECT SUM(price * (1 - discount))
FROM Supplier, Lineitem, Orders, Customer
WHERE Supplier.SK = Lineitem.SK AND Lineitem.OK = Orders.OK
  AND Orders.CK = Customer.CK
  AND Orders.orderdate >=' 2020 - 08 - 01'
```

We rewrite it as

```
SELECT Supplier.SK, Customer.CK, price * (1 - discount)
FROM Supplier, Lineitem, Orders, Customer
WHERE Supplier.SK = Lineitem.SK AND Lineitem.OK = Orders.OK
  AND Orders.CK = Customer.CK
  AND Orders.orderdate >=' 2020 - 08 - 01'
```

The `price * (1 - discount)` column in the query results gives all the $\psi(q_k(\mathbf{I}))$ values, while `Supplier.SK` and `Customer.CK` yield the referencing relationships from each supplier and customer to all the join results they contribute to. □

We execute the rewritten query in PostgreSQL, and export the query results to a file. Then, an external program is invoked to construct the $\log(\mathrm{GS}_Q)$ LPs from the query results, which are then solved by CPLEX. Finally, we use R2T to compute a privatized output.

The computation bottleneck is the $\log(\mathrm{GS}_Q)$ LPs, each of which contains $|J(\mathbf{I})|$ variables and $|J(\mathbf{I})| + |\mathbf{I}(R_P)|$ constraints. This takes polynomial time, but can still be very expensive in practice. One immediate optimization is to solve them in parallel. Below we present another effective technique to speed up the process.

Early stop The key observation is that R2T returns the maximum of $O(\log(\mathrm{GS}_Q))$ maximization LPs (masked by some noise and reduced by a factor), and most LP solvers

Algorithm 4: R2T with early stop

```
   Input: $\mathbf{I}$, $Q$,$R_P$, $\mathrm{GS}_Q$
1  $\tilde{Q}(\mathbf{I}) \leftarrow 0$;
2  for $\tau^{(j)} \leftarrow \mathrm{GS}_Q, \mathrm{GS}_Q/2, \dots, 1$ do in parallel
3      $T^{(j)} \leftarrow \mathrm{Lap}\left(\log(\mathrm{GS}_Q)\frac{\tau^{(j)}}{\varepsilon}\right) - \log(\mathrm{GS}_Q)\ln\left(\frac{\log(\mathrm{GS}_Q)}{\beta}\right) \cdot \frac{\tau^{(j)}}{\varepsilon}$;
4      for $t \leftarrow 1, 2, \dots$ do
5          if $\hat{Q}^{(t)}(\mathbf{I}, \tau^{(j)})$ achieves the optimal then
6              $\tilde{Q}(\mathbf{I}) \leftarrow \max(\tilde{Q}(\mathbf{I}), \hat{Q}^{(t)}(\mathbf{I}, \tau^{(j)}) + T^{(j)})$;
7              Break;
8          else if $\hat{Q}^{(t)}(\mathbf{I}, \tau^{(j)}) + T^{(j)} \leq \tilde{Q}(\mathbf{I})$ then
9              Break;
10         end
11     end
12 end
13 return $\tilde{Q}(\mathbf{I})$;
```

(e.g., CPLEX) for maximization problems use some iterative search technique to gradually approach the optimum from below, namely, these $O(\log(\mathrm{GS}_Q))$ LP solvers all "race to the top". Thus, we will not know the winner until they all stop.

To cut down the unnecessary search, the idea is to flip the problem around. Instead of solving the primal LPs, we solve their duals. By LP duality, the dual LP has the same optimal solution as the primal, but importantly, the LP solver will approach the optimal solution from above, namely, we have a gradually decreasing upper bound for the optimal solution of each LP. This allows us to terminate those LPs that have no hope to be the winner. The optimized R2T algorithm, shown in Algorithm 4, also uses the trick that the noises are generated before we start running the LP solvers, so that we know when to terminate.

In Algorithm 4, we use t to denote the iterations of the LP solver, and use $\hat{Q}^{(t)}(\mathbf{I}, \tau)$ to denote the solution to the dual LP in the t-th iteration. A technicality is that in line 1, we should initialize $\tilde{Q}(\mathbf{I})$ to $Q(\mathbf{I}, 0)$ to be consistent with the R2T algorithm, but $Q(\mathbf{I}, 0) = 0$ for all the truncation methods described in this paper.

When there are not enough CPU cores to solve all LPs in parallel, we choose to start with those with a larger τ in line 3 of Algorithm 4. This is based on our observation that those LPs tend to terminate faster. This is very intuitive: when τ is larger, the optimal solution is also higher, thus the LP solver for the dual can terminate earlier.

5.8 Experiments

We conducted experiments on two types of queries: graph pattern counting queries under node-DP and general SPJA queries with FK constraints, with the former being an important special case of the latter. For graph pattern counting queries, we compare R2T with naive truncation with smooth sensitivity (NT) [55], smooth distance estimator (SDE) [15], recursive mechanism (RM) [22], and the LP-based mechanism (LP) [55]. For general SPJA queries, we compare with the local sensitivity-based mechanism (LS) [76].

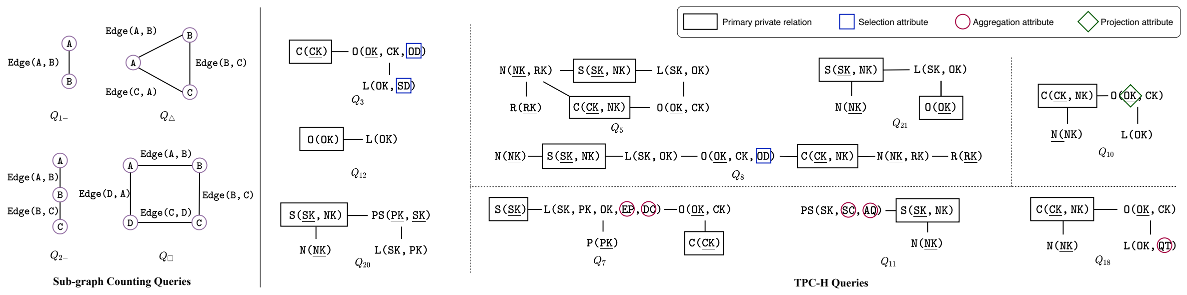


Figure 5.5: The structure of queries.

Dataset	**Deezer**	**Amazon1**	**Amazon2**	**RoadnetPA**	**RoadnetCA**
Nodes	144,000	262,000	335,000	1,090,000	1,970,000
Edges	847,000	900,000	926,000	1,540,000	2,770,000
Maximum degree	420	420	549	9	12
Degree upper bound D	1,024	1,024	1,024	16	16

Table 5.1: Graph datasets used in the experiments.

5.8.1 Setup

Queries For graph pattern counting queries, we used four queries: edge counting Q_{1-}, length-2 path counting Q_{2-}, triangle counting $Q_{\triangle}$, and rectangle counting $Q_{\square}$. For SPJA queries, we used 10 queries from the TPC-H benchmark, whose structures are shown in Figure 5.5. These queries involve a good mix of selection, projection, join, and aggregation. We removed all the group-by clauses from the queries — a brief discussion on this is provided at the end of the paper.

Datasets For graph pattern counting queries, we used 5 real world networks datasets: **Deezer**, **Amazon1**, **Amazon2**, **RoadnetPA** and **RoadnetCA**. **Deezer** collects the friendships of users from the music streaming service Deezer. **Amazon1** and **Amazon2**

are two Amazon co-purchasing networks. **RoadnetPA** and **RoadnetCA** are road networks of Pennsylvania and California, respectively. All these datasets are obtained from SNAP [58]. Table 5.1 shows the basic statistics of these datasets.

Most algorithms need to assume a GS_Q in advance. Note that the value of GS_Q should not depend on the instance, but may use some background knowledge for a particular class of instances. Thus, for the three social networks, we set a degree upper bound of $D = 1024$, while for the two road networks, we set $D = 16$. Then we set GS_Q as the maximum number of graph patterns containing any node. This means that $GS_{Q_{1-}} = D$, $GS_{Q_{2-}} = GS_{Q_{\triangle}} = D^2$, and $GS_{Q_{\square}} = D^3$. For TPC-H queries, we used datasets of scale $2^{-3}, 2^{-2}, \ldots, 2^3$. The one with scale 1 (default scale) has about 7.5 million tuples, and we set $\mathrm{GS}_Q = 10^6$.

The LP mechanism requires a truncation threshold τ, but [55] does not discuss how this should be set. Initially, we used a random threshold uniformly chosen from $[1, \mathrm{GS}_Q]$. This turned out to be very bad as with constant probability, the picked threshold is $\Omega(\mathrm{GS}_Q)$, which makes these mechanisms as bad as the naive mechanism that adds GS_Q noise. To achieve better results, as in R2T, we consider $\{2, 4, 8, \ldots, \mathrm{GS}_Q\}$ as the possible choices. Similarly, NT and SDE need a truncation threshold θ on the degree, and we choose one from $\{2, 4, 8, \ldots, D\}$ randomly.

Experimental environment All experiments were conducted on a Linux server with a 24-core 2.2GHz Intel Xeon CPU and 256GB of memory. Each program was allowed to use at most 10 threads and we set a time limit of 6 hours for each run. Each experiment was repeated 100 times and we report the average running time. The errors are less stable due to the random noise, so we remove the best 20 and worst 20 runs, and report the average error of the remaining 60 runs. The failure probability β in R2T is set to 0.1. The default DP parameter is $\varepsilon = 0.8$.

5.8.2 Graph Pattern Counting Queries

Utility and efficiency The errors and running times of all mechanisms over the graph pattern counting queries are shown in Table 5.2. These results indicate a clear superiority of R2T in terms of utility, offering order-of-magnitude improvements over other methods in many cases. What is more desirable is its robustness: In all the 20 query-dataset

Dataset		**Deezer**		**Amazon1**		**Amazon2**		**Roadnet – PA**		**Roadnet – CA**	
Result type		Relative error(%)	Time(s)	Relative error(%)	Time(s)	Relative error(%)	Time(s)	Relative error(%)	Time(s)	Relative error(%)	Time(s)
q_{1-}	Query result	847,000	1.28	900,000	1.52	926,000	1.62	1,540,000	1.51	2,770,000	2.64
	R2T	0.535	12.3	0.557	15.6	0.432	16.2	0.0114	26.8	0.00635	48.7
	NT	59.1	18.1	101	29.3	125	40.4	1,370	21.9	1,410	39.7
	SDE	548	9,870	363	4,570	286	1,130	55.2	105	81.8	292
	LP	14.3	16.9	5.72	14.7	6.75	14.4	3.6	28.3	3.02	54
q_{2-}	Query result	21,800,000	13.8	9,120,000	11.8	9,750,000	13.8	3,390,000	6.39	6,000,000	6.06
	R2T	6.64	356	12.2	170	9.06	196	0.0539	80.2	0.0352	145
	NT	116	21.0	398	28.4	390	41.0	6,160	23.2	6,530	44.2
	SDE	8,900	9,870	5,110	4,570	1,930	1,130	211	104	228	296
	LP	35.9	8,820	23.2	3,600	27.8	461	11.1	148	13.3	404
$q_{\triangle}$	Query result	794,000	4.53	718,000	5.03	667,000	4.20	67,200	2.96	121,000	5.17
	R2T	5.58	17.3	1.27	18.8	2.03	19.9	0.102	4.21	0.061	7.5
	NT	782	23.0	1,660	31.7	1,920	41.0	110,000	23.3	105,000	45.0
	SDE	67,300	9,880	26,000	4,570	9,600	1,130	4,150	106	3,830	297
	LP	24.6	131	12.8	18.2	14.2	18.3	0.104	3.95	0.0625	7.06
	RM	Over time limit						0.0388	1,280	0.0193	2,550
$q_{\square}$	Query result	11,900,000	74.3	2,480,000	21.6	3,130,000	15.6	158,000	4.50	262,000	10.1
	R2T	16.9	289	6.29	70.5	10.5	86.8	0.0729	8.18	0.0638	16.2
	NT	3,750	57.6	30,700	35.8	26,100	50.6	319,000	24.8	368,000	45.0
	SDE	6,970,000	9,930	11,400,000	4,580	202,000	1,140	10,300	108	9,130	300
	LP	92.6	2,530	94.8	70.4	77.8	81.2	0.223	7.83	0.165	14.2
	RM	Over time limit						0.0217	10,500	Over time limit	

Table 5.2: Comparison between R2T, naive truncation with smooth sensitivity (NT), smooth distance estimator (SDE), LP-based Mechanism (LP), and recursive mechanism (RM) on graph pattern counting queries.

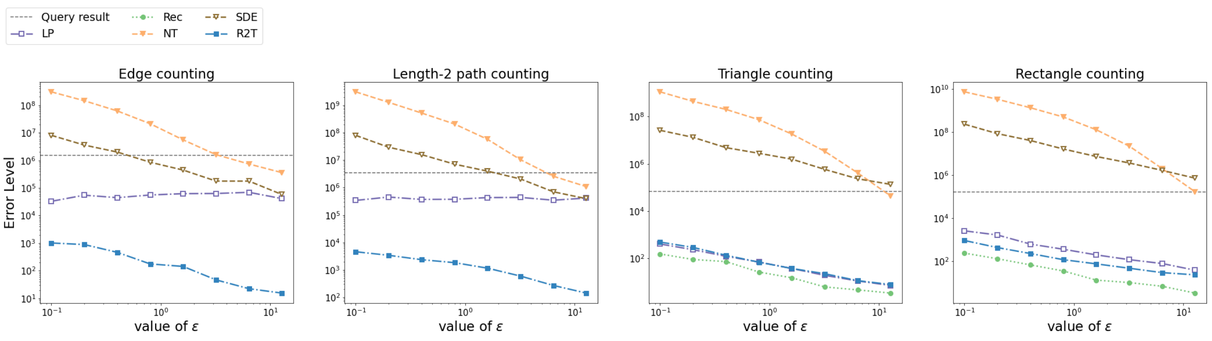


Figure 5.6: Error levels of various mechanisms on graph pattern counting queries various values of ε.

combinations, R2T consistently achieves an error below 20%, while the error is below 10% in all but 3 cases. We also notice that, given a query, R2T performs better in road networks than social networks. This is because the error of R2T is proportional to $DS_Q(\mathbf{I})$ by our theoretical analysis. Thus the relative error is proportional to $DS_Q(\mathbf{I})/|Q(\mathbf{I})|$. Therefore, larger and sparser graphs, such as road networks, lead to smaller relative errors.

In terms of running time, all mechanisms are reasonable, except for RM and SDE. RM can only complete within the 6-hour time limit on 3 cases, although it achieves very small errors on these 3 cases. SDE is faster than RM but runs a bit slower than others. It is also interesting to see that R2T sometimes even runs faster than LP, despite the fact that R2T needs to solve $O(\log \mathrm{GS}_Q)$ LPs. This is due to the early stop optimization: The running time of R2T is determined by the LP that corresponds to the near-optimal τ, which often happens to be one of the LPs that can be solved fastest.

Query		Q_{1-}	Q_{2-}	$Q_{\triangle}$	$Q_{\square}$
Query result		926,000	9,750,000	667,000	3,130,000
R2T		4,000	883,000	13,500	328,000
LP	$\tau = \mathrm{GS}_Q$	1,440	1,580,000	1,290,000	1,370,000,000
	$\tau = \mathrm{GS}_Q/8$	2,100	181,000	157,000	140,000,000
	$\tau = \mathrm{GS}_Q/64$	110,000	259,000	15,100	25,800,000
	$\tau = \mathrm{GS}_Q/512$	645,000	1,260,000	2,790	2,630,000
	$\tau = \mathrm{GS}_Q/4096$	810,000	3,950,000	2,090	274,000
	$\tau = \mathrm{GS}_Q/32768$	911,000	7,580,000	92,300	48,700
	$\tau = \mathrm{GS}_Q/262144$	924,000	9,340,000	459,000	76,400
	Average error	62,500	2,710,000	94,900	2,430,000

Table 5.3: Error levels of R2T and LP-based mechanism (LP) with different τ.

Dataset	**Deezer**	**Amazon1**	**Amazon2**	**RoadnetPA**	**RoadnetCA**
w early stop	289	70.5	86.8	8.18	16.2
w/o early stop	28,700	537	422	12.8	16.4
Speed up	99.3×	7.62 ×	4.86×	1.56×	1.01×

Table 5.4: Running times of R2T with and without early stop.

Privacy parameter ε Next, we conducted experiments to see how the privacy parameter ε affects various mechanisms. We tested different queries on **Roadnet – PA** where we vary ε from 0.1 to 12.8. We plot the results in Figure 5.6, where we also plot the query result to help see the utilities of the mechanisms. The first message from the plot is the same as before, that both R2T and RM achieve high utility (but RM spends 300x more time). NT and SDE lose utility (i.e., error larger than query result) except for very large ε. LP achieves similar utility as R2T on $Q_{\triangle}$ and $Q_{\square}$, but is much worse on Q_{1-} and Q_{2-}. In particular, a higher ε does not help LP on these two queries, because the bias (further controlled by a randomly selected τ) dominates the error for these two queries.

Selection of τ In the next set of experiments, we dive deeper and see how sensitive the utility is with respect to the truncation threshold τ. We tested the queries on **Amazon2** and measured the error of the LP-based mechanism [55] with different τ. For each query, we tried various τ from 2 to GS_Q and compare their errors with R2T. The results are shown in Table 5.3, where the optimal error is marked in gray. The results indicate that the error is highly sensitive to τ, and more importantly, the optimal choice of τ closely depends on the query, and there is no fixed τ that works for all cases. On the other hand, the error of R2T is within a small constant factor (around 6) to the optimal choice of τ, which is exactly the value of instance-optimality.

Early stop optimization We also did some experiments to compare the running time of R2T with and without the early stop optimization. Here, we ran $Q_{\square}$ over different datasets and the results are shown in Table 5.4. From this table, we can see the early stop is particularly useful in cases with long running times. In these cases, one or two LPs, which do not correspond to the optimal choice of τ, take a long time to run, and early stop is able to terminate these LPs as soon as possible.

5.8.3 SPJA Queries

Query type		Single primary private relation			Multiple primary private relations			Aggregation			Projection
Query		Q_3	Q_{12}	Q_{20}	Q_5	Q_8	Q_{21}	Q_7	Q_{11}	Q_{18}	Q_{10}
Query result	Value	2,890,000	6,000,000	6,000,000	240,000	1,830,000	6,000,000	218,000,000	2,000,000	153,000,000	1,500,000
	Time(s)	1.6	1.24	1.25	2.51	1.41	2.32	3.22	0.29	2.21	0.32
R2T	Relative error(%)	0.254	0.0229	0.579	1.626	1.92	0.654	0.607	1.82	0.132	0.174
	Time(s)	18.9	28.2	24.5	8.42	39.6	124	140	4.41	42.7	8.77
LS	Relative error(%)	38.8	16.3	15.4	Not supported						
	Time(s)	19.2	25.8	24.4							

Table 5.5: Comparison between R2T and local-sensitivity based mechanism (LS) on SQL queries.

Utility and efficiency We tested 10 queries from the TPC-H benchmark comparing R2T and LS, and the results are shown in Table 5.5. We see that R2T achieves order-of-magnitude improvements over LS in terms of utility, with similar running times. More importantly, R2T supports a variety of SPJA queries that are not supported by LS, with robust performance across the board.

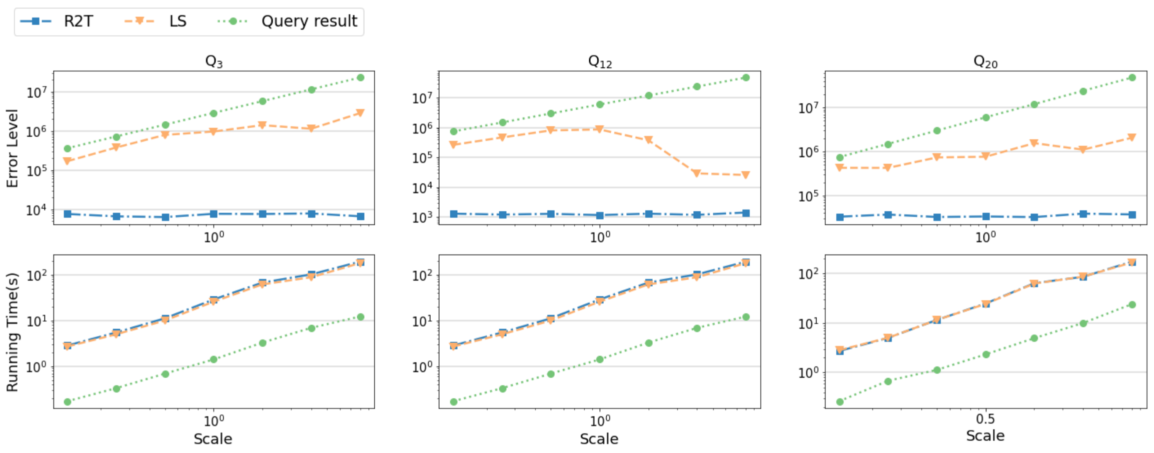


Figure 5.7: Running times and error levels of R2T and local-sensitivity based mechanism (LS) for different data scales.

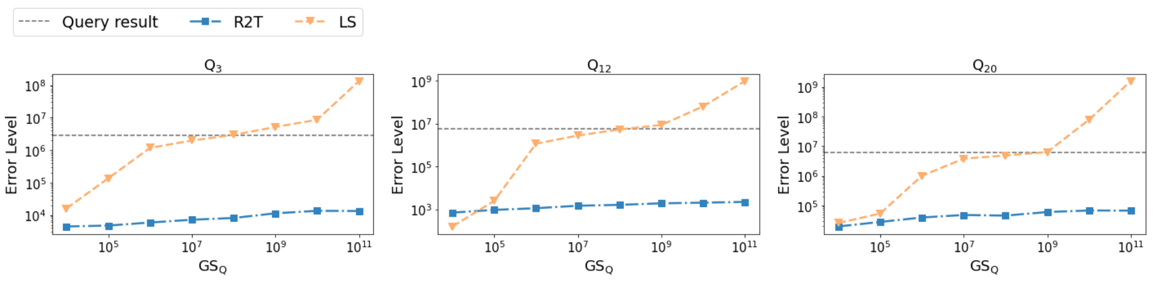


Figure 5.8: Error levels of R2T and local-sensitivity based mechanism (LS) with different GS_Q.

Scalability To examine the effects as the data scale changes, we used TPC-H datasets with scale factors ranging from 2^{-3} to 2^3 with Q_3, Q_{12} and Q_{20}. We compare both the errors and running times of R2T and LS. The results are shown in Figure 5.7. From the results, we see that the error of R2T barely increases with the data size. The reason is that our error only depends on $DS_Q(\mathbf{I})$, which does not change much by the scale of TPC-H data. On the other hand, the behavior of LS is more complicated. For Q_3 and Q_{20}, its error increases with the data size; for Q_{12}, its error increases first but then decreases later. This is because LS runs an SVT on the sensitivities of tuples to choose τ, which is closely related to the distribution of tuples' sensitivities. This is another indication that selecting a near-optimal τ is not an easy task. In terms of running time, both mechanisms have the running time linearly increase with the data size, which is expected.

Dependency on GS_Q Our last set of experiments examine the effect GS_Q brings to the utilities of R2T and LS. We conducted experiments using Q_3, Q_{12}, Q_{20} with different values of GS_Q. The results are shown in Figure 5.8. When GS_Q is small, the errors of these two mechanisms are very close. When GS_Q increases, the error of LS increases rapidly, and loses the utility (error larger than query result) very soon. Meanwhile, the error of R2T increases very slowly with GS_Q. This confirms our analysis that the error of LS grows near linearly as GS_Q, while that of R2T grows logarithmically. The important consequence is that, with R2T, one can be very conservative in setting the value of GS_Q. This gives the DBA a much easier job, in case s/he has little idea on what datasets the database is expected to receive. Meanwhile, recall that GS_Q is public information, so using a larger GS_Q reveals less information about the private dataset.

CHAPTER 6

ANSWERING MULTIPLE SPA QUERIES UNDER USER-DP

6.1 Preliminaries

6.1.1 Notation

The definition of user-DP in a database has already been defined in Section 5.1, where we study answering single SJA query. We list some key notations used in this chapter and please refer to Section 5.1 for more details.

The users are stored in primary private relation R_P with size N and $t_i(\mathbf{I})$ is its ith tuple. The multi-way join is denoted as J. $q_j(\mathbf{I})$ denotes the jth join result. Here, we additionally define $M = |J(\mathbf{I})|$. A SJA query Q aggregates over the join results $J(\mathbf{I})$ and we use the shorthand $\psi_j(\mathbf{I}) := \psi(q_j(\mathbf{I}))$. To better describe the referencing relationships between tuples and join results, for each $i \in [N], j \in [M]$, we define

$$C_i(\mathbf{I}) := \{j : q_j(\mathbf{I}) \text{ references } t_i(\mathbf{I})\}, \tag{6.1}$$

$$D_j(\mathbf{I}) := \{i : q_j(\mathbf{I}) \text{ references } t_i(\mathbf{I})\}. \tag{6.2}$$

In this chapter, we consider answering d such queries $\mathbf{Q} = (Q_1, \ldots, Q_d)$. We subscript them by k, and generalize the notation above as $N_k, M_k, J_k(\mathbf{I}), q_{k,j}(\mathbf{I}), \psi_{k,j}(\mathbf{I})$, etc. An important special case is group-by queries. Continuing with Example 1.2.1, suppose we add a `GROUP BY OrderDate` clause. Then $d = |\mathbf{dom}(\texttt{OrderDate})|$; $N_k, M_k, J_k(\mathbf{I})$, $q_{k,j}(\mathbf{I})$ are the same for all k, while $\psi_{k,j}(\mathbf{I})$ has a different predicate `OrderDate` $=x$, where x ranges over all the dates in $\mathbf{dom}(\texttt{OrderDate})$. Nevertheless, all developments below will assume the general case where the d queries can be completely different.

6.1.2 More Differential Privacy

For 1D query, the most commonly used DP mechanism is the Laplace mechanism (see Section 1.1.1). For a vectored-valued query, the *Gaussian mechanism* is more commonly used:

Lemma 6.1.1 (Gaussian Mechanism [33, 18])**.** *Given a d-dimensional query* $\mathbf{Q} : \mathcal{I} \to \mathbb{R}^d$*, let* $\mathrm{GS}_{\mathbf{Q}} := \max_{\mathbf{I}\sim\mathbf{I}'} \|\mathbf{Q}(\mathbf{I}) - \mathbf{Q}(\mathbf{I}')\|$*. Then the mechanism*

$$\mathcal{M}(\mathbf{I}) = \mathbf{Q}(\mathbf{I}) + \sigma \cdot \mathbf{Y},$$

where $\mathbf{Y} \sim \mathcal{N}(0, \mathbf{I}_{d\times d})$*, preserves* $\left(\frac{\mathrm{GS}_{\mathbf{Q}}^2}{2\sigma^2} + \frac{\mathrm{GS}_{\mathbf{Q}}}{\sigma}\sqrt{2\log(1/\delta)}, \delta\right)$*-DP for any* $\delta > 0$*.*

Thus, given ε, δ, one can set $\frac{\mathrm{GS}_{\mathbf{Q}}^2}{2\sigma^2} + \frac{\mathrm{GS}_{\mathbf{Q}}}{\sigma}\sqrt{2\log(1/\delta)} = \varepsilon$ and solve for σ, and we denote the solution as $\sigma(\varepsilon, \delta)$. For $\varepsilon = O(1)$, $\sigma(\varepsilon, \delta) = O\left(\sqrt{\log(1/\delta)}/\varepsilon \cdot \mathrm{GS}_{\mathbf{Q}}\right)$, so the Gaussian mechanism achieves an ℓ_2 error of $\tilde{O}\left(\sqrt{d} \cdot \mathrm{GS}_{\mathbf{Q}}\right)$.

Both the Laplace and the Gaussian mechanism require a small GS_Q.[1] If it is large or unbounded, one may consider the local sensitivity $\mathrm{LS}_{\mathbf{Q}}(\mathbf{I})$, and try to obtain a DP upper bound of $\mathrm{LS}_{\mathbf{Q}}(\mathbf{I})$ before applying the mechanism. Karwa et al. [53] show how this can be done for the Laplace mechanism. Here we obtain a similar result for the Gaussian mechanism:

Lemma 6.1.2 (Second-order Gaussian Mechanism)**.** *Given a d-dimensional query* $\mathbf{Q} : \mathcal{I} \to \mathbb{R}^d$*, suppose there is an* $(\varepsilon_1, \delta_1)$*-DP mechanism that outputs an* $\widehat{\mathrm{LS}}_{\mathbf{Q}}(\mathbf{I}) \geq \mathrm{LS}_{\mathbf{Q}}(\mathbf{I})$ *with probability at least* $1 - \delta_2$*, then the mechanism*

$$\mathcal{M}(\mathbf{I}) = \mathbf{Q}(\mathbf{I}) + \widehat{\mathrm{LS}}_{\mathbf{Q}}(\mathbf{I}) \cdot \sigma(\varepsilon_2, \delta_3) \cdot \mathbf{Y},$$

where $\mathbf{Y} \sim \mathcal{N}(0, \mathbf{I}_{d\times d})$*, preserves* $(\varepsilon_1 + \varepsilon_2, \delta_1 + e^{\varepsilon_1}\delta_2 + e^{\varepsilon_1}\delta_3)$*-DP.*

Proof. Similar to the proof of Lemma 4.4 in [53]. □

However, in a relational database, $\mathrm{LS}_{\mathbf{Q}}(\mathbf{I})$ is also unbounded as mentioned in Section 5.2 and we want to achieve $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$ error. Recall $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$ corresponds to the maximum contribution of any user in $\mathbf{I}$ and has already been discussed for 1D query in Section 5.2. For multiple queries,

$$\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}) = \max_{\mathbf{I}'\subseteq\mathbf{I}, \mathbf{I}'\sim\mathbf{I}} \|\mathbf{Q}(\mathbf{I}') - \mathbf{Q}(\mathbf{I})\|.$$

For each user $t_i(\mathbf{I}) \in \mathbf{I}(R_P)$, let

$$S_{k,i}(\mathbf{I}) = \sum_{j\in C_{k,i}(\mathbf{I})} \psi_{k,j}(\mathbf{I}) \tag{6.3}$$

[1]In this chapter, we adopt ℓ_2 distance metric thus the sensitivities like GS_Q, $\mathrm{LS}_Q(\mathbf{I})$ are also defined in ℓ_2 distance.

be the contribution of $t_i(\mathbf{I})$ to Q_k. The contributions of $t_i(\mathbf{I})$ to all queries are thus a d-dimensional vector $\mathbf{S}_i(\mathbf{I}) = (S_{1,i}(\mathbf{I}), \dots, S_{d,i}(\mathbf{I}))$. Then we have $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}) = \max_{i \in [N]} \|\mathbf{S}_i(\mathbf{I})\|$.

6.2 Multiple Self-join-free Queries

We start with the simpler case of d self-join-free queries. We observe that it is equivalent to the sum estimation problem in d dimensions: Given N vectors in d dimensions $\mathbf{x}_1, \dots, \mathbf{x}_N$, we wish to estimate $\sum_{i \in [N]} \mathbf{x}_i$ under DP where neighboring instances differ by one vector. For the forward direction, we just set $\mathbf{x}_i := \mathbf{S}_i(\mathbf{I})$. Since there is no self-join, each join result only references one user, then adding/removing one user in $\mathbf{I}$ is the same as adding/removing vector. For the backward direction, we simply construct a single table with d columns storing these vectors, and the k-th query asks for the sum on the k-th column.

This equivalence has two immediate consequences. First, the lower bound on the sum estimation problem is also a lower bound for the multi-query problem, which of course also holds for the more difficult case of self-joins:

Theorem 6.2.1 ([46, 49]). *For the multi-query problem, no DP mechanism can achieve an error smaller than $\tilde{\Omega}\left(\sqrt{d} \cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})\right)$ for all $\mathbf{I}$.*

Secondly, any sum estimation mechanism can also be used for self-join-free queries. However, all existing mechanisms assume that $\|\mathbf{S}_i(\mathbf{I})\| \le \mathrm{GS}_{\mathbf{Q}}$ for a predefined $\mathrm{GS}_{\mathbf{Q}}$. Under this assumption, the best result is [46][2]

$$O\left(\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}) \cdot \left(\sqrt{d} + \sqrt{\log(\mathrm{GS}_{\mathbf{Q}}) \log\log(\mathrm{GS}_{\mathbf{Q}})}\right) \cdot \sqrt{\log(1/\delta)}/\varepsilon\right).$$

Below, we show how to remove this assumption, i.e., we allow $\mathrm{GS}_{\mathbf{Q}} = \infty$. Meanwhile, we also improve the error bound to

$$O\left(\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}) \cdot \left(\sqrt{d \log(1/\delta)} + \log\log(\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}))\right) / \varepsilon\right).$$

Our idea is to extend our 1-dimensional mechanism given in Chapter 4 to d dimensions. Given a truncation threshold $r \ge 0$, the truncated query result $\mathbf{Q}(\mathbf{I}, r)$ is defined as

$$\mathbf{Q}(\mathbf{I}, r) := \sum_{i \in [N]} \left(\min\left(1, \frac{r}{\|\mathbf{S}_i(\mathbf{I})\|}\right) \cdot \mathbf{S}_i(\mathbf{I})\right).$$

[2] [46] states the result under zCDP [18]; here we translate their result to (ε, δ)-DP.

It is easy to see that $\mathbf{Q}(\mathbf{I}, r)$ has the global sensitivity r, so the Gaussian mechanism can be applied. Note that by using $r = \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$, no data is truncated and the Gaussian mechanism achieves the optimal error $\tilde{O}\left(\sqrt{d} \cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})\right)$. However, since $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$ depends on $\mathbf{I}$, using it directly will breach privacy. Then the idea is to find a privatized r that is as close to $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}) = \max_{i \in [N]} \|\mathbf{S}_i(\mathbf{I})\|$ as possible.

For any $r \geq 0$, define

$$\mathrm{Count}(\mathbf{I}, r) = \left|\{i : \|\mathbf{S}_i(\mathbf{I})\| \leq r\}\right|.$$

Then consider the sequence of queries $\mathrm{Count}(\mathbf{I}, r) - N$ for $r = 1, 2, 4, \dots$. It is clear that each such query has global sensitivity 1. We use SVT (introduced in Section 4.1.3) with privacy budget $\varepsilon/10$ to find the first $\tilde{r}$ such that $\mathrm{Count}(\mathbf{I}, \tilde{r}) - N > -\frac{60}{\varepsilon} \log \frac{4}{\beta}$, where β will be failure probability. After finding such an $\tilde{r}$, we invoke the Gaussian mechanism with global sensitivity $\tilde{r}$ and privacy budget $9\varepsilon/10$.[3] The detailed algorithm is shown in Algorithm 5.

Algorithm 5: Multiple self-join-free queries.

```
  Input: $\mathbf{I}$, $\varepsilon$, $\beta$
1 $\tilde{i} \leftarrow \mathrm{SVT}(-\frac{60}{\varepsilon}\log\frac{4}{\beta}, \frac{\varepsilon}{10}, \mathrm{Count}(\mathbf{I}, 2^0) - N, \mathrm{Count}(\mathbf{I}, 2^1) - N, \dots)$;
2 $\tilde{r} \leftarrow 2^{\tilde{i}-1}$;
3 $\widetilde{\mathbf{Q}}(\mathbf{I}) = \mathbf{Q}(\mathbf{I}, \tilde{r}) + \tilde{r} \cdot \sigma(\frac{9\varepsilon}{10}, \delta) \cdot \mathbf{Y}$, $\mathbf{Y} \sim \mathcal{N}(0, \mathbf{I}_{d \times d})$;
4 return $\widetilde{\mathbf{Q}}(\mathbf{I})$;
```

Theorem 6.2.2. *Algorithm 5 satisfies (ε, δ)-DP, and with probability at least $1 - \beta$,*

$$\|\widetilde{\mathbf{Q}}(\mathbf{I}) - \mathbf{Q}(\mathbf{I})\| = O\left(\frac{\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})}{\varepsilon} \cdot \left(\log \frac{\log(\mathrm{DS}_{\mathbf{Q}})}{\beta} + \sqrt{d \log \frac{1}{\delta} \log \frac{1}{\beta}}\right)\right).$$

Proof. The privacy guarantee follows directly from basic composition. For the utility, by Lemma 4.1.3, with probability at least $1 - \frac{\beta}{2}$, we have (1) $\mathrm{Count}(\mathbf{I}, \tilde{r}) \geq N - O\left(\frac{1}{\varepsilon} \cdot \log \frac{\log(\mathrm{DS}_{\mathbf{Q}})}{\beta}\right)$; and (2) $\tilde{r} \leq \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$. The first result implies the introduced bias is $O\left(\frac{\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})}{\varepsilon} \cdot \log \frac{\log(\mathrm{DS}_{\mathbf{Q}})}{\beta}\right)$. By combining the second result and the tail bound of Gaussian distribution, with probability at least $1 - \frac{\beta}{2}$, the added noise is $O\left(\frac{\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})}{\varepsilon} \cdot \sqrt{d \log(1/\delta) \log(1/\beta)}\right)$. □

[3]The ε is split into $\varepsilon/10$ and $9\varepsilon/10$ since the error is dominated by the noise instead of bias and we want to use more privacy budget when adding the noise.

6.3 Multiple Queries with Self-joins

6.3.1 Why Self-joins are Hard

When self-joins are present, or more fundamentally, when a join result references more than one user, a number of difficulties arise. First, since the contributions of the users overlap, the problem is no longer equivalent to sum estimation. Second, as pointed out in [27], the truncation mechanism fails: The query on all users i with $S_i(\mathbf{I}) \le r$ still has unbounded global sensitivity. To address this issue, [27] replaces this "hard" truncation with a "soft" truncation [55] for the case of a single query. The idea is that each join result may contribute a part of its full $\psi_j(\mathbf{I})$, so that the total contribution from any user is bounded by r. This can be formulated as an LP, where the variable z_j denotes the partial contribution from the j-th join result:

$$\begin{aligned} \max \quad & Q(\mathbf{I}, r) = \sum_{j \in [M]} z_j \\ \text{s.t.} \quad & \sum_{j \in C_i(\mathbf{I})} z_j \le r, \qquad i \in [N], \\ & 0 \le z_j \le \psi_j(\mathbf{I}), \qquad j \in [M]. \end{aligned}$$

The truncated query answer, denoted $Q(\mathbf{I}, r)$, is set to be the optimal solution of this LP, which can be shown to have global sensitivity r, so the Laplace mechanism can be applied. Then, [27] further proposes a mechanism to privately select an r to achieve the optimal error $\tilde{O}(\mathrm{DS}_Q(\mathbf{I}))$.

This LP can be naturally extended to multiple queries: z_j and $\psi_j(\mathbf{I})$ both become d-dimensional vectors, the first constraint imposes a bound r on the ℓ_2 norm of $\sum z_j$, and the second constraint becomes an element-wise inequality. Meanwhile, $\mathbf{Q}(\mathbf{I}, r) = \sum z_j$ also becomes a vector, and we may try to maximize its norm. This turns the LP in a quadratic program (QP), which is still efficiently solvable. But the critical issue is that $\mathbf{Q}(\mathbf{I}, r)$ has high local sensitivity, as illustrated in the following example.

Example 6.3.1. Consider the query in Example 1.2.1 with `GROUP BY OrderDate`. Suppose there are only two different dates on `OrderDate`, so we have just $d = 2$ queries. Consider an instance $\mathbf{I}$ that has $N/2$ suppliers $s_1, s_2, \cdots s_{N/2}$ and $N/2$ customers $c_1, c_2, \cdots c_{N/2}$. Each lineitem is purchased by one customer and supplied by a supplier, and the detailed construction is shown in Figure 6.1, where solid lines denote the lineitems on day 1 and

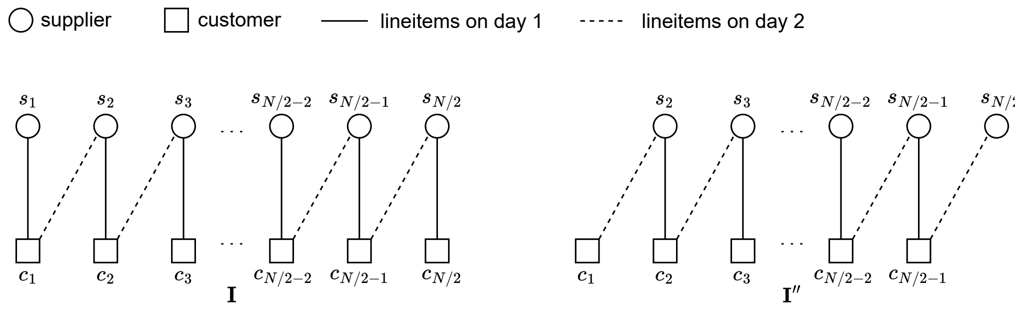

supplier
customer
lineitems on day 1
lineitems on day 2
s_1 s_2 s_3 $s_{N/2-2}$ $s_{N/2-1}$ $s_{N/2}$
c_1 c_2 c_3 $c_{N/2-2}$ $c_{N/2-1}$ $c_{N/2}$
$\mathbf{I}$
$\mathbf{I}''$

Figure 6.1: An example showing $\mathbf{Q}(\mathbf{I}, r)$ has large sensitivity.

dashed lines for day 2. Next, we construct $\mathbf{I}'$ by deleting s_1 (and the associated lineitem) and construct $\mathbf{I}''$ by further deleting $c_{N/2}$. Note that we have $\mathbf{I} \sim \mathbf{I}' \sim \mathbf{I}''$.

Suppose we set $r = 1$. We see that $\|\mathbf{Q}(\mathbf{I}, 1)\|$ is maximized by keeping all lineitems on day 1, yielding $\mathbf{Q}(\mathbf{I}, 1) = (N/2, 0)$, while $\|\mathbf{Q}(\mathbf{I}'', 1)\|$ is maximized by keeping all lineitems on day 2, yielding $\mathbf{Q}(\mathbf{I}'', 1) = (0, N/2 - 1)$. We see that although $\|\mathbf{Q}(\mathbf{I}, 1)\| - \|\mathbf{Q}(\mathbf{I}'', 1)\|$ is small, $\|\mathbf{Q}(\mathbf{I}, 1) - \mathbf{Q}(\mathbf{I}'', 1)\|$ is large, which means that one of $\|\mathbf{Q}(\mathbf{I}, 1) - \mathbf{Q}(\mathbf{I}', 1)\|$ and $\|\mathbf{Q}(\mathbf{I}', 1) - \mathbf{Q}(\mathbf{I}'', 1)\|$ must be large. Fundamentally, the reason is that although the LP (or QP) has low sensitivity in its optimal value $\|\mathbf{Q}(\mathbf{I}, r)\|$, it does not necessarily imply a low sensitivity on the optimal vector solution $\mathbf{Q}(\mathbf{I}, r)$, except in one dimension. However, the Gaussian mechanism needs a low sensitivity on $\mathbf{Q}(\mathbf{I}, r)$, not $\|\mathbf{Q}(\mathbf{I}, r)\|$. □

6.3.2 An Exponential-time Algorithm

To address the issue above, we take a different approach to defining $\mathbf{Q}(\mathbf{I}, r)$ so that it has bounded sensitivity. First, define

$$E(\mathbf{I}, r) := \max_{\mathbf{I}'' \subseteq \mathbf{I}, \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \leq r} |\mathbf{I}''(R_P)|,$$

i.e., the maximum number of users in any induced sub-instance of $\mathbf{I}$ such that no user's contribution is more than r in ℓ_2 norm. Note that for self-join-free queries, $E(\mathbf{I}, r) = \mathrm{Count}(\mathbf{I}, r)$. When there are self-joins, we have $E(\mathbf{I}, r) \geq \mathrm{Count}(\mathbf{I}, r)$, since removing one user may also reduce the contributions of other users. Exactly due to this reason, $\mathrm{Count}(\mathbf{I}, r)$ has unbounded sensitivity for self-joins, which is why our self-join-free algorithm no longer works. On the other hand, we will show that $E(\mathbf{I}, r)$ has sensitivity 1, as desired. However, computing $E(\mathbf{I}, r)$ can take exponential time: Even for the simple query $J = \texttt{Edge(A, B)} \bowtie \texttt{Node(A)} \bowtie \texttt{Node(B)}$ with $\psi(q) = 1$ for all $q \in J(\mathbf{I})$ (i.e., counting the number of edges in a graph under node-DP), $E(\mathbf{I}, r)$ is exactly the size of the maximum induced subgraph with degree constraint r, which is a classical NP-hard problem. We will leave the computational issue to Section 6.3.3, while focusing on privacy for now.

Next, define

$$F(\mathbf{I}, r) := E(\mathbf{I}, r) - |\mathbf{I}(R_P)|,$$

i.e., $-F(\mathbf{I}, r)$ is the minimum number of users that need to be removed so that the contribution from any user is at most r. The following lemma is obvious:

Lemma 6.3.1. *For any r and any $\mathbf{I}$, $F(\mathbf{I}, r) \leq 0$. If $r \geq \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$, then $F(\mathbf{I}, r) = 0$.*

More importantly, we show that $F(\mathbf{I}, r)$ has sensitivity 1:

Lemma 6.3.2. *For any r and any $\mathbf{I} \sim \mathbf{I}'$, $\mathbf{I}' \subseteq \mathbf{I}$, we have*

$$F(\mathbf{I}', r) - 1 \leq F(\mathbf{I}, r) \leq F(\mathbf{I}', r).$$

Proof. Let $N' = |\mathbf{I}'(R_P)|$ and $\mathbf{I}(R_P) = \mathbf{I}'(R_P) \cup t_N(\mathbf{I})$. On one hand, it is trivial to see for any $r \geq 0$, $E(\mathbf{I}, r) \geq E(\mathbf{I}', r)$ since any $\mathbf{I}'' \subseteq \mathbf{I}'$ also has $\mathbf{I}'' \subseteq \mathbf{I}$.

On the other hand, given any $r \geq 0$, let

$$\mathbf{I}^* = \operatorname*{arg\,max}_{\mathbf{I}'' \subseteq \mathbf{I}, \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \leq r} |\mathbf{I}''|.$$

Then, we can construct a $\mathbf{I}^{*\prime}$ from $\mathbf{I}^*$ by deleting all tuples referencing $t_N(\mathbf{I})$. Then, $\mathbf{I}^{*\prime} \subseteq \mathbf{I}'$, $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}^{*\prime}) \leq \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}^*) \leq r$ and $|\mathbf{I}^{*\prime}(R_P)| = |\mathbf{I}^*(R_P)| - 1$. And this further means, $E(\mathbf{I}', r) \geq E(\mathbf{I}, r) - 1$.

Finally, combining $N = N' + 1$, the lemma follows. □

Because $F(\mathbf{I}, r)$ has sensitivity 1, we can feed $F(\mathbf{I}, 1), F(\mathbf{I}, 2), F(\mathbf{I}, 4), \dots$ into SVT with threshold $-O(\log(1/\beta)/\varepsilon)$. Using a similar argument as for our self-join-free algorithm, we can show that this will return an $\tilde{r}$ that is close to $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$.

It turns out that $E(\mathbf{I}, r)$ is not only useful for finding the truncation threshold $\tilde{r}$, it also yields a new definition of the truncated query answer $\mathbf{Q}(\mathbf{I}, r)$ with bounded sensitivity. For any $r \geq 0$, define

$$\mathbf{Q}(\mathbf{I}, r) := \mathbf{Q}(\mathbf{I}^*(r)), \text{ where } \mathbf{I}^*(r) = \operatorname*{arg\,max}_{\mathbf{I}'' \subseteq \mathbf{I}, \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \leq r} |\mathbf{I}''(R_P)|. \tag{6.4}$$

Note that $\mathbf{I}^*(r)$ may not be unique. In this case, we use an arbitrary tie-breaker (e.g., the $\mathbf{I}''$ with the lexicographically smallest user IDs) to fix an $\mathbf{I}^*(r)$.

Example 6.3.2. Following the Example 6.3.1 and assuming $N = 6c$ for some $c \in \mathbb{N}$, for $\mathbf{I}$, when $r = 1$, we have $E(\mathbf{I}, 1) = 2N/3$, $\mathbf{I}^*(1)$ is obtained by deleting s_{3*i-1}'s and c_{3*i}'s for all $i \in [N/6]$, and $\mathbf{Q}(\mathbf{I}, 1) = (N/6, N/6)$. When $r \geq 2$, $E(\mathbf{I}, r) = N$, $\mathbf{I}^*(r) = \mathbf{I}$, and $\mathbf{Q}(\mathbf{I}, r) = (N/2, N/2 - 1)$. Meanwhile, for $\mathbf{I}''$, when $r = 1$, we have $E(\mathbf{I}'', 1) = 2N/3 - 1$, $\mathbf{I}^{*''}(1)$ is obtained by deleting c_{3*i-1}'s for all $i \in [N/6]$ and s_{3*j+1}'s for all $j \in [N/6 - 1]$, and $\mathbf{Q}(\mathbf{I}'', 1) = (N/6 - 1, N/6)$. When $r \geq 2$, $E(\mathbf{I}'', r) = N - 2$, $\mathbf{I}^{*''}(r) = \mathbf{I}''$, and $\mathbf{Q}(\mathbf{I}'', r) = (N/2 - 2, N/2 - 1)$. □

We bound the local sensitivity of $\mathbf{Q}(\mathbf{I}, r)$ as follows.

Lemma 6.3.3. *Given $r \geq 0$, for any $\mathbf{I} \sim \mathbf{I}'$,*

$$\|\mathbf{Q}(\mathbf{I}, r) - \mathbf{Q}(\mathbf{I}', r)\| \leq (-2F(\mathbf{I}, r) + 2) \cdot r.$$

Proof. Here, we first consider the case $\mathbf{I}(R_P) = \mathbf{I}'(R_P) \cup \{t_N(\mathbf{I})\}$.

For convenience, let

$$\mathbf{I}^* = \operatorname*{arg\,max}_{\mathbf{I}'' \subseteq \mathbf{I}, \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \leq r} \|\mathbf{I}''(R_P)\|,$$

$$\mathbf{I}^{*'} = \operatorname*{arg\,max}_{\mathbf{I}'' \subseteq \mathbf{I}', \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \leq r} \|\mathbf{I}''(R_P)\|.$$

And define

$$\mathbf{I}^*_\cap = \mathbf{I}^* \cap \mathbf{I}^{*'}.$$

First, by definition,

$$\begin{cases} |\mathbf{I}(R_P) \setminus \mathbf{I}'(R_P)| \leq 1 \\ |\mathbf{I}'(R_P) \setminus \mathbf{I}^{*'}(R_P)| \leq -F(\mathbf{I}', r) \end{cases} \Rightarrow |\mathbf{I}(R_P) \setminus \mathbf{I}^{*'}(R_P)| \leq -F(\mathbf{I}', r) + 1.$$

Further recall $\mathbf{I}^* \subseteq \mathbf{I}$, we have

$$|\mathbf{I}^*(R_P) \setminus \mathbf{I}^{*'}(R_P)| \leq -F(\mathbf{I}', r) + 1,$$

which means

$$|\mathbf{I}^*(R_P) \setminus \mathbf{I}^*_\cap(R_P)| \leq -F(\mathbf{I}', r) + 1.$$

Recall $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}^*) \leq r$,

$$\|\mathbf{Q}(\mathbf{I}^*) - \mathbf{Q}(\mathbf{I}^*_\cap)\| \leq (-F(\mathbf{I}', r) + 1) \cdot r. \tag{6.5}$$

Algorithm 6: Exponential-time mechanism for multiple queries with self-joins.

```
Input: $\varepsilon$, $\delta$, $\beta$
1 $\tilde{i} \leftarrow \mathrm{SVT}\left(-\frac{30}{\varepsilon}\log(6/\beta), \frac{\varepsilon}{5}, F(\mathbf{I}, 2^0), F(\mathbf{I}, 2^1), \ldots\right)$;
2 $\tilde{r} \leftarrow 2^{\tilde{i}-1}$;
3 $T \leftarrow -2F(\mathbf{I}, \tilde{r}) + 2$;
4 $\widehat{T} \leftarrow T + \mathrm{Lap}(\frac{5}{\varepsilon}) + \frac{5}{\varepsilon}\log(e^{3\varepsilon/5}/\delta)$;
5 $\widetilde{\mathbf{Q}}(\mathbf{I}) \leftarrow \mathbf{Q}(\mathbf{I}, \tilde{r}) + \widehat{T} \cdot \sigma\left(\frac{2\varepsilon}{5}, \frac{\delta}{2e^{3\varepsilon/5}}\right) \cdot \tilde{r} \cdot \mathbf{Y}, \mathbf{Y} \sim \mathcal{N}(0, \mathbf{I}_{d\times d})$;
6 return $\widetilde{\mathbf{Q}}(\mathbf{I})$;
```

Symmetrically, we have

$$\|\mathbf{Q}(\mathbf{I}^{*'}) - \mathbf{Q}(\mathbf{I}^*_\cap)\| \leq -F(\mathbf{I}, r) \cdot r. \tag{6.6}$$

Combining Lemma 6.3.2, (6.5), (6.6), the claim follows.

Similarly, for the other case, where $\mathbf{I}'(R_P) = \mathbf{I}(R_P) \cup \{t_{N+1}(\mathbf{I}')\}$, we have,

$$\|\mathbf{Q}(\mathbf{I}^*) - \mathbf{Q}(\mathbf{I}^*_\cap)\| \leq -F(\mathbf{I}', r) \cdot r,$$
$$\|\mathbf{Q}(\mathbf{I}^{*'}) - \mathbf{Q}(\mathbf{I}^*_\cap)\| \leq (-F(\mathbf{I}, r) + 1) \cdot r.$$

With a similar argument, the claim also follows. □

Note that Lemma 6.3.3 does not imply a small sensitivity for all r. In particular, if r is very small, $-F(\mathbf{I}, r)$ can be as large as N, resulting in the same issue as in Example 6.3.1. However, the crucial difference here is that we will only use an $\tilde{r}$ returned by the SVT, which has $-F(\mathbf{I}, \tilde{r}) = \tilde{O}(1)$ with high probability. We can further add a Laplace noise to it so that it becomes a high-probability DP upper bound on the local sensitivity, and then apply the second-order Gaussian mechanism. The detailed algorithm is shown in Algorithm 6.

Theorem 6.3.1. *For any ε, δ, $\beta > 0$, and any $\mathbf{I}$, Algorithm 6 satisfies (ε, δ)-DP and returns a $\widetilde{\mathbf{Q}}(\mathbf{I})$ such that with probability at least $1 - \beta$,*

$$\|\widetilde{\mathbf{Q}}(\mathbf{I}) - \mathbf{Q}(\mathbf{I})\|$$
$$= O\left(\frac{\sqrt{d\log(1/\beta)\log(e^\varepsilon/\delta)}}{\varepsilon^2} \cdot \mathrm{DS}(\mathbf{I}) \cdot \left(\log\frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta} + \log(e^\varepsilon/\delta)\right)\right).$$

Proof. Privacy: By Lemma 6.3.2, each $F(\mathbf{I}, \cdot)$ has the sensitivity 1 thus line 1 and line 4 consume $\frac{\varepsilon}{5}$ and $\frac{2\varepsilon}{5}$ privacy budget respectively. By composition theory, $\widehat{T}$ preserves $\frac{3\varepsilon}{5}$-DP and by the tail bound of Laplace distribution, with probability at least $1 - \frac{\delta}{2e^{\varepsilon/5}}$, $\widehat{T} \geq T$. By Lemma 6.1.2, $\widetilde{Q}(\mathbf{I})$ preserves (ε, δ)-DP.

Below we prove the utility bound. First, by Lemma 4.1.3, with probability at least $1 - \frac{\beta}{3}$,

$$-F(\mathbf{I}, \tilde{r}) = O\left(\frac{1}{\varepsilon} \log \frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta}\right), \tag{6.7}$$

and

$$\tilde{r} \leq 2 \cdot \mathrm{DS}(\mathbf{I}). \tag{6.8}$$

Second, by (6.7) and the definitions of $\mathrm{DS}(\mathbf{I})$ and $\mathbf{Q}(\mathbf{I}, \tilde{r})$,

$$\|\mathbf{Q}(\mathbf{I}) - \mathbf{Q}(\mathbf{I}, \tilde{r})\| = O\left(\frac{\mathrm{DS}(\mathbf{I})}{\varepsilon} \log \frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta}\right). \tag{6.9}$$

Third, by (6.7) and the tail bound of Laplace distribution, with probability at least $1 - \frac{\beta}{3}$,

$$\widehat{T} = O\left(\frac{1}{\varepsilon} \log \frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta} + \frac{1}{\varepsilon} \log(e^{\varepsilon}/\delta)\right). \tag{6.10}$$

Then, combine (6.8), (6.10) and the tail bound of Gaussian distribution, we have with probability at least $1 - \frac{\beta}{3}$,

$$\begin{aligned} &\left\|\widetilde{\mathbf{Q}}(\mathbf{I}) - \mathbf{Q}(\mathbf{I}^*)\right\| \\ =&O\left(\frac{\sqrt{d \log(1/\beta) \log(e^{\varepsilon}/\delta)}}{\varepsilon^2} \cdot \mathrm{DS}(\mathbf{I}) \cdot \left(\log \frac{\log(\mathrm{DS})}{\beta} + \log(e^{\varepsilon}/\delta)\right)\right). \end{aligned} \tag{6.11}$$

Finally, the theorem follows by combining (6.9) and (6.11). □

6.3.3 A Polynomial-time Algorithm

In this section, we show how to reduce the running time of Algorithm 6 to polynomial without affecting its utility bound. The computational bottleneck is $E(\mathbf{I}, r)$. We borrow a popular technique from approximation algorithms: formulate $E(\mathbf{I}, r)$ as an integer program, and solve its relaxed version.

Observe that any $\mathbf{I}'' \subseteq \mathbf{I}$ is specified by the users in $\mathbf{I}''(R_P)$. We introduce a variable $y_i \in \{0, 1\}$ to indicate whether the i-th user is included in $\mathbf{I}''(R_P)$. The objective is thus to maximize $\sum_i y_i$. To link the y_i's with the join results, we introduce a variable $z_{k,j} \in \{0,1\}$ to indicate whether the join result $q_{k,j}(\mathbf{I}) \in J_k(\mathbf{I}'')$, for $k \in [d]$, $j \in [M_k(\mathbf{I})]$. Recall $q_{k,j}(\mathbf{I}) \in J_k(\mathbf{I}'')$ if and only if for any $i \in D_{k,j}(\mathbf{I})$, $t_i(\mathbf{I}) \in \mathbf{I}''(R_P)$. To capture this requirement, we add the following constraint:

$$z_{k,j} \geq \sum_{i \in D_{k,j}(\mathbf{I})} y_i - |D_{k,j}(\mathbf{I})| + 1, k \in [d], j \in [M_k].$$

Note that the RHS is equal to 1 if $y_i = 1$ for all $i \in D_{k,j}(\mathbf{I})$ and 0 otherwise.

Next, we need to express the constraint $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \leq r$. Recall $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') = \max_i \|\mathbf{S}_i(\mathbf{I}'')\|$, where $\mathbf{S}_i(\mathbf{I}'') = (S_{1,i}(\mathbf{I}''), \dots, S_{d,i}(\mathbf{I}''))$. Plugging (6.3) into the constraint $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \leq r$ turns it into

$$\sum_{k \in [d]} \left(\sum_{j \in C_{k,i}(\mathbf{I})} (\psi_{k,j}(\mathbf{I}) \cdot z_{k,j}) \right)^2 \leq r^2, i \in [N].$$

Therefore, $E(\mathbf{I}, r)$ is the optimal solution of the following integer program:

$$\begin{aligned}
\max \quad & \sum_{i \in [N]} y_i, & \\
\text{s.t.} \quad & z_{k,j} \geq \sum_{i \in D_{k,j}(\mathbf{I})} y_i - |D_{k,j}(\mathbf{I})| + 1, & k \in [d], j \in [M_k] \\
& \sum_{k \in [d]} \left(\sum_{j \in C_{k,i}(\mathbf{I})} (\psi_{k,j}(\mathbf{I}) \cdot z_{k,j}) \right)^2 \leq r^2, & i \in [N], \\
& y_i \in \{0,1\} & i \in [N], \\
& z_{k,j} \in \{0,1\} & k \in [d], j \in [M_k].
\end{aligned}$$

By relaxing the integral constraint to $y_i \in [0,1], z_{k,j} \in [0,1]$, this program turns into a QCQP. Note that only convex QCQPs can be solved efficiently, which is indeed the case for our QCQP, by observing that the quadratic constraint is positive semi-definite.

Let $\{y_i^*\}_i, \{z_{k,j}^*\}_{k,j}$ be the optimal fractional solution of this QCQP, and let $\widehat{E}(\mathbf{I}, r) = \sum_i y_i^*$. In approximation algorithms, one would then try to round $\{y_i^*\}_i, \{z_{k,j}^*\}_{k,j}$ into integers, and show that the rounded solution is not too far away from $\widehat{E}(\mathbf{I}, r)$. However, for the private query answering problem, there is no need to do the rounding as we will not return the join results anyway. Instead, we only need to return the privatized aggregated query answer. On the other hand, we must show that $\widehat{E}(\mathbf{I}, r)$ preserves the three important sensitivity properties of $E(\mathbf{I}, r)$, namely, Lemma 6.3.1–6.3.3.

Letting $\widehat{F}(\mathbf{I}, r) = \widehat{E}(\mathbf{I}, r) - |\mathbf{I}(R_P)|$. The first property is trivial:

Lemma 6.3.4. *For any r and any $\mathbf{I}$, $\widehat{F}(\mathbf{I}, r) \le 0$. If $r \ge \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$, then $\widehat{F}(\mathbf{I}, r) = 0$.*

Now we prove that $\widehat{F}(\mathbf{I}, r)$ also has sensitivity 1.

Lemma 6.3.5. *For any r and any $\mathbf{I} \sim \mathbf{I}'$, $\mathbf{I}' \subseteq \mathbf{I}$, we have*

$$\widehat{F}(\mathbf{I}', r) - 1 \le \widehat{F}(\mathbf{I}, r) \le \widehat{F}(\mathbf{I}', r).$$

Proof. Let $\mathbf{I}(R_P) = \mathbf{I}'(R_P) \cup t_N(\mathbf{I})$. Assume that the join results referencing $t_N(\mathbf{I})$ are put at the end of $J(\mathbf{I})$, i.e., for every $k \in [d]$, $C_{k,i}(\mathbf{I}) = \{M_k - |C_{k,i}(\mathbf{I})| + 1, \dots, M_k - 1, M_k\}$.

Let $\{y_i^*\}_i, \{z_{k,j}^*\}_{k,j}$ and $\{y_i^{*'}\}_i, \{z_{k,j}^{*'}\}_{k,j}$ be the optimal fractional solutions of the QCQP on $\mathbf{I}$ and $\mathbf{I}'$, respectively. On one hand, from $\{y_i^{*'}\}_i, \{z_{k,j}^{*'}\}_{k,j}$, we can construct a valid solution of the QCQP for $\mathbf{I}$ by setting $y_N^{*'} = 0$, $z_{k.j}^{*'} = 0$ for any $k \in [d]$, $j \in C_{k,N}(\mathbf{I})$. Thus, the optimal QCQP solution on $\mathbf{I}$ can only be higher.

On the other hand, by removing y_N^*, $z_{k,j}^*$ for any $k \in [k]$, $j \in C_{k,N}(\mathbf{I})$ for all $k \in [k]$, from $\{y_i^*\}_i, \{z_{k,j}^*\}_{k,j}$, we can obtain a valid solution of the QCQP on $\mathbf{I}'$. On this solution, we have $\sum_{i \in [N-1]]} y_i^* = E(\mathbf{I}, r) - y_N^* \ge E(\mathbf{I}, r) - 1$, which implies that $\widehat{E}(\mathbf{I}', r) \ge \widehat{E}(\mathbf{I}, r) - 1$.

Finally, combining with $N = N' + 1$, the lemma follows. □

Lastly, we show how the optimal fractional solution $\{y_i^*\}_i, \{z_{k,j}^*\}_{k,j}$ also leads to a $\widehat{\mathbf{Q}}(\mathbf{I}, r)$ with bounded local sensitivity as in Lemma 6.3.3. First, it is easy to see that there must exist an optimal solution in which the constraint on each $z_{k,j}^*$ is tight, i.e.,

$$z_{k,j}^* = \max\left(0, \sum_{i \in D_{k,j}(\mathbf{I})} y_i^* - |D_{k,j}(\mathbf{I})| + 1\right). \tag{6.12}$$

If not, we could lower $z_{k,j}^*$ to make it tight without violating the quadratic constraint or changing the objective. If there are still multiple optimal fractional solutions, we pick one using an arbitrary tie-breaker. Then, we define $\widehat{\mathbf{Q}}(\mathbf{I}, r)$ as the query answers using the optimal fractional solution, i.e.,

$$\widehat{\mathbf{Q}}(\mathbf{I}, r) = \mathbf{Q}(\mathbf{I}^*) = (Q_1(\mathbf{I}^*), Q_2(\mathbf{I}^*), \dots, Q_d(\mathbf{I}^*)),$$

$$\text{where } Q_k(\mathbf{I}^*) = \sum_{j \in [M_k]} \left(z_{k,j}^* \cdot \psi_{k,j}(\mathbf{I})\right), k \in [d]. \tag{6.13}$$

Example 6.3.3. Following the Example 6.3.2, when $r = 1$, for both $\mathbf{I}$ and $\mathbf{I}'$, we have all $z_{k,j} = \sqrt{2}/2$, and all $y_i = \sqrt{2}/4 + 1/2$, thus $\widehat{E}(\mathbf{I}, 1) = \sqrt{2}N/4 + N/2$, $\widehat{Q}(\mathbf{I}, 1) = (\sqrt{2}N/4, \sqrt{2}N/4 - \sqrt{2}/2)$, $\widehat{E}(\mathbf{I}'', 1) = \sqrt{2}N/4 + N/2$, and $\widehat{Q}(\mathbf{I}'', 1) = (\sqrt{2}N/4 - \sqrt{2}, \sqrt{2}N/4 - \sqrt{2}/2)$. When $r \geq 2$, for both $\mathbf{I}$ and $\mathbf{I}'$, we have all $z_{k,j} = 1$, and all $y_i = 1$, thus $\widehat{E}(\mathbf{I}, r) = N$, $\widehat{Q}(\mathbf{I}, 1) = (N/2, N/2 - 1)$, $\widehat{E}(\mathbf{I}'', 1) = N - 2$, and $\widehat{Q}(\mathbf{I}'', 1) = (N/2 - 2, N/2 - 1)$. □

We now bound the local sensitivity of $\widehat{\mathbf{Q}}(\mathbf{I}, r)$:

Lemma 6.3.6. *Given any $r \geq 0$, for any $\mathbf{I} \sim \mathbf{I}'$,*

$$\left\|\widehat{\mathbf{Q}}(\mathbf{I}, r) - \widehat{\mathbf{Q}}(\mathbf{I}', r)\right\| \leq \left(-2\widehat{F}(\mathbf{I}, r) + 2\right) \cdot r.$$

Proof. We consider the case $\mathbf{I}(R_P) = \mathbf{I}'(R_P) \cup \{t_N(\mathbf{I})\}$ and the other case can been shown similarly as the proof of Lemma 6.3.3. And similarly as the proof of Lemma 6.3.5, we assume the join results corresponding the $t_N(\mathbf{I})$ are put at the end.

For $\mathbf{I}$, we have $\{y_i^*\}_i$, $\{z_{k,j}^*\}_{k,j}$, and $\mathbf{Q}(\mathbf{I}^*)$ constructed as $\widehat{E}(\mathbf{I}, r)$, (6.12), and (6.13). For $\mathbf{I}'$, we have $\{y_i^{*'}\}_i$, $\{z_{k,j}^{*'}\}_{k,j}$, and $\mathbf{Q}(\mathbf{I}^{*'})$. To unify the size of $\{y_i^*\}_i$ and $\{y_i^{*'}\}_i$, we append one zero at the end of $\{y_i^{*'}\}_i$. And we process similarly for $\{z_{k,j}^{*'}\}_{k,j}$.

By definition of $\widehat{F}(\mathbf{I}, r)$ and $\{y_i^*\}_i$, we have

$$N - \sum_{i \in [N]} y_i^* = -\widehat{F}(\mathbf{I}, r). \tag{6.14}$$

And similarly, we have

$$N - \sum_{i \in [N]} y_i^{*'} = -\widehat{F}(\mathbf{I}', r) + 1 \leq -\widehat{F}(\mathbf{I}, r) + 2, \tag{6.15}$$

where the inequality is by Lemma 6.3.5.

By this setting, we have

$$\begin{aligned}
&\left\|\mathbf{Q}(\mathbf{I}^*) - \mathbf{Q}(\mathbf{I}^{*'})\right\| \\
=&\sqrt{\sum_{k\in[d]}\left(\sum_{j\in[M_k]}\left(z_{k,j}^{*'} - z_{k,j}^*\right)\cdot\psi_{k,j}(\mathbf{I})\right)^2} \\
\leq&\sqrt{\sum_{k\in[d]}\left(\sum_{j\in[M_k]}\left(z_{k,j}^{*'} - z_{k,j}^*\right)\cdot\psi_{k,j}(\mathbf{I})\cdot\mathbb{I}\left(z_{k,j}^* < z_{k,j}^{*'}\right)\right)^2} \\
&+\sqrt{\sum_{k\in[d]}\left(\sum_{j\in[M_k]}\left(z_{k,j}^* - z_{k,j}^{*'}\right)\cdot\psi_{k,j}(\mathbf{I})\cdot\mathbb{I}\left(z_{k,j}^* > z_{k,j}^{*'}\right)\right)^2}.
\end{aligned} \tag{6.16}$$

For any $k \in [d]$, $j \in [M_k]$ such that $z_{k,j}^* < z_{k,j}^{*'}$, we have

$$\begin{aligned}
z_{k,j}^{*'} - z_{k,j}^* \leq& z_{k,j}^{*'} - z_{k,j}^*\cdot z_{k,j}^{*'} \\
=&\left(1 - z_{k,j}^*\right)\cdot z_{k,j}^{*'} \\
\leq&\sum_{i\in D_{k,j}(\mathbf{I})}\left(1 - y_i^*\right)\cdot z_{k,j}^{*'}.
\end{aligned} \tag{6.17}$$

The first line is by $z_{k,j}^{*'} \leq 1$. The last line is by (6.12).

Then, we have

$$
\begin{aligned}
&\sqrt{\sum_{k\in[d]}\left(\sum_{j\in[M_k]}\left(z_{k,j}^{*'}-z_{k,j}^{*}\right)\cdot\psi_{k,j}(\mathbf{I})\cdot\mathbb{I}\left(z_{k,j}^{*}<z_{k,j}^{*'}\right)\right)^2}\\
\le&\sqrt{\sum_{k\in[d]}\left(\sum_{j\in[M_k]}\sum_{i\in D_{k,j}(\mathbf{I})}(1-y_i^*)\cdot z_{k,j}^{*'}\cdot\psi_{k,j}(\mathbf{I})\cdot\mathbb{I}\left(z_{k,j}^{*}<z_{k,j}^{*'}\right)\right)^2}\\
\le&\sqrt{\sum_{k\in[d]}\left(\sum_{j\in[M_k]}\sum_{i\in D_{k,j}(\mathbf{I})}(1-y_i^*)\cdot z_{k,j}^{*'}\cdot\psi_{k,j}(\mathbf{I})\right)^2}\\
\le&\sum_{i\in[N]}\left((1-y_i^*)\cdot\sqrt{\sum_{k\in[d]}\left(\sum_{j\in C_{k,i}(\mathbf{I})}z_{k,j}^{*'}\cdot\psi_{k,j}(\mathbf{I})\right)^2}\right)\\
\le&\sum_{i\in[N]}(1-y_i^*)\cdot r\\
\le&-\widehat{F}(\mathbf{I},r)\cdot r
\end{aligned}
\tag{6.18}
$$

The second line is by (6.17). The fourth line is by the triangle inequality under ℓ_2 distance metric. The last line is by (6.14).

Similarly, with (6.15), we can show,

$$
\begin{aligned}
&\sqrt{\sum_{k\in[d]}\left(\sum_{j\in[M_k]}\left(z_{k,j}^{*}-z_{k,j}^{*'}\right)\cdot\psi_{k,j}(\mathbf{I})\cdot\mathbb{I}\left(z_{k,j}^{*}>z_{k,j}^{*'}\right)\right)^2}\\
\le&\left(-\widehat{F}(\mathbf{I},r)+2\right)\cdot r
\end{aligned}
\tag{6.19}
$$

Finally, combining (6.16), (6.18), and (6.19), the lemma follows. □

Our polynomial-time algorithm is thus the same as Algorithm 6, except that $F(\mathbf{I},r)$ and $\mathbf{Q}(\mathbf{I},r)$ are replaced by $\widehat{F}(\mathbf{I},r)$ and $\widehat{\mathbf{Q}}(\mathbf{I},r)$, respectively. Below we show that this replacement does not affect its privacy or utility:

Theorem 6.3.2. *For any $\varepsilon,\delta,\beta>0$ and any $\mathbf{I}$, the polynomial-time version of Algorithm*

6 preserves (ε, δ)-DP, and returns a $\widetilde{\mathbf{Q}}(\mathbf{I})$ such that with probability at least $1-\beta$,

$$\begin{aligned}&\left\|\widetilde{\mathbf{Q}}(\mathbf{I})-\mathbf{Q}(\mathbf{I})\right\| \\ =&O\left(\frac{\sqrt{d\log(1/\beta)\log(e^{\varepsilon}/\delta)}}{\varepsilon^2}\cdot \mathrm{DS}(\mathbf{I})\cdot\left(\log\frac{\log(\mathrm{DS})}{\beta}+\log(e^{\varepsilon}/\delta)\right)\right).\end{aligned}$$

Proof. The privacy analysis remains the same as in the proof of Theorem 6.3.1, since $\widehat{F}(\mathbf{I},r)$ and $\widehat{\mathbf{Q}}(\mathbf{I},r)$ have the same sensitivity properties as $F(\mathbf{I},r)$ and $\mathbb{Q}(\mathbf{I},r)$. Below we analyze the utility.

First, by Lemma 4.1.3 and 6.3.1, with probability at least $1-\frac{\beta}{3}$,

$$-\widehat{F}(\mathbf{I},\tilde{r})=O\left(\frac{1}{\varepsilon}\log\frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta}\right), \tag{6.20}$$

and $\tilde{r}\le 2\cdot\mathrm{DS}(\mathbf{I})$.

By the difference between ExpPrivMultiSJA and PolyPrivMultiSJA, we only need to bound the bias

$$\left\|\mathbf{Q}(\mathbf{I})-\widehat{\mathbf{Q}}(\mathbf{I},\tilde{r})\right\|=O\left(\frac{\mathrm{DS}(\mathbf{I})}{\varepsilon}\log\frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta}\right). \tag{6.21}$$

Let $\{y_i^*\}_i=\mathrm{Sol}\left(\widehat{E}(\mathbf{I},\tilde{r})\right)$. And we construct $\{z_{k,j}^*\}_{k,j}$ and $\mathbf{Q}(\mathbf{I}^*)$ as (6.12) and (6.13). By (6.20),

$$N-\sum_i y_i^*=O\left(\frac{1}{\varepsilon}\log\frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta}\right). \tag{6.22}$$

Then, we increment $\{y_i^*\}_i$ to one vector, i.e. all elements equal to 1, and update the corresponding $\{z_{k,j}^*\}_{k,j}$, $\mathbf{Q}(\mathbf{I}^*)$ iteratively. For convenience, we use the subscript to denote the iteration and let $\{y_i^*\}_i$, $\{z_{k,j}^*\}_{k,j}$ and, $\mathbf{Q}(\mathbf{I}^*)$ be the ones for iteration 0, i.e, $\{y_i^{*(0)}\}_i=\{y_i^*\}_i$, $\{z_{k,j}^{*(0)}\}_{k,j}=\{z_{k,j}^*\}_{k,j}$, and

$$\mathbf{Q}(\mathbf{I}^{*(0)})=\mathbf{Q}(\mathbf{I}^*). \tag{6.23}$$

At iteration $i\in[N]$, we increment $y_i^{*(i)}$ from y_i^* to 1. Then, we set $\{z_{k,j}^{*(i)}\}_{k,j}$ and $\mathbf{Q}(\mathbf{I}^{*(i)})$ as (6.12) and (6.13). With setting, we have

$$\begin{aligned}
&\|\mathbf{Q}(\mathbf{I}^{*(i)}) - \mathbf{Q}(\mathbf{I}^{*(i-1)})\| \\
=&\sqrt{\sum_{k=1}^{d}\left(\sum_{j=1}^{M_k}\left(z_{k,j}^{*(i)} - z_{k,j}^{*(i-1)}\right)\cdot\psi_{k,j}(\mathbf{I})\right)^2} \\
=&\sqrt{\sum_{k=1}^{d}\left(\sum_{j\in C_{k,i}(\mathbf{I})}\left(z_{k,j}^{*(i)} - z_{k,j}^{*(i-1)}\right)\cdot\psi_{k,j}(\mathbf{I})\right)^2} \\
\leq&\sqrt{\sum_{k=1}^{d}\left(\sum_{j\in C_{k,i}(\mathbf{I})}(1-y_i^*)\cdot\psi_{k,j}(\mathbf{I})\right)^2} \\
=&(1-y_i^*)\cdot\sqrt{\sum_{k=1}^{d}\left(\sum_{j\in C_{k,i}(\mathbf{I})}\psi_{k,j}(\mathbf{I})\right)^2} \\
\leq&(1-y_i^*)\cdot\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}). \qquad (6.24)
\end{aligned}$$

The third line is because, at ith iteration, for any $k \in [d]$, $z_{k,j}^{*(i)}$ will not be updated if $j \notin C_{k,i}(\mathbf{I})$. The fourth line is by (6.12). The last line is by the definition of $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$.

After all iterations, we have all $y_i^{*(N)} = 1$ thus

$$\mathbf{Q}(\mathbf{I}^{*(N)}) = \mathbf{Q}(\mathbf{I}). \qquad (6.25)$$

Summing up (6.24) for all i and combining (6.23) and (6.25), we get (6.21). Finally, with a similar process as the proof of Theorem 6.3.1, the claim follows. □

6.4 System Implementation

Our algorithm can be implemented on top of any SQL engine and a QCQP solver. For our system prototype, we use PostgreSQL and MOSEK.

The first step is to extract $\{C_{k,i}\}_{k,i}$, $\{D_{k,j}\}_{k,j}$ from the join results. Note that the original query does not output this information, so the first step is to rewrite the query so that it also includes the PKs of the private entities, as illustrated in the following example.

Example 6.4.1. Consider Q5 of TPC-H benchmark:

```
SELECT nation_name, SUM(price * (1 - discount))
FROM Supplier, Lineitem, Orders, Customer, Nation
WHERE ... GROUP BY nation_name
```

We rewrite it as

```
SELECT nation_name, Supplier.SK, Customer.CK,
   price * (1 - discount)
FROM Supplier, Lineitem, Orders, Customer, Nation
WHERE ...
```

From the results of the rewritten query, we then construct a series of QCQPs and feed them into the SVT. Note that these QCQPs only differ in the value of r. The SVT returns an $\tilde{r}$, $\widehat{F}(\mathbf{I}, \tilde{r})$, and the corresponding optimal fractional solution, from which we construct $\widehat{\mathbf{Q}}(\mathbf{I}, \tilde{r})$. Finally, we add Gaussian noise to $\widehat{\mathbf{Q}}(\mathbf{I}, \tilde{r})$.

Algorithm 7: SVT with jump start

```
Input: $T$, $\varepsilon$, $k$, and a sequence of sensitivity-1 queries $f_1(\mathbf{I}), f_2(\mathbf{I}), \dots$
$\widetilde{T} \leftarrow T + \mathrm{Lap}(2/\varepsilon)$;
$\widehat{f'} = \infty$;
for $\ell \leftarrow k, k-1, \dots, 1$ do
    $v_\ell \leftarrow \mathrm{Lap}(4/\varepsilon)$;
    if $f' + v_\ell < \widetilde{T}$ then
        $f_\ell(\mathbf{I}) \leftarrow f'$;
    else
        Compute $f_\ell(\mathbf{I})$;
        $f' \leftarrow f_\ell(\mathbf{I})$;
    end
end
for $\ell \leftarrow 1, 2, \dots$ do
    if $\ell > k$ then
        $v_\ell \leftarrow \mathrm{Lap}(4/\varepsilon)$;
        Compute $f_\ell(\mathbf{I})$;
    end
    if $f_\ell(\mathbf{I}) + v_\ell \geq \widetilde{T}$ then
        return $\ell$;
    end
end
```

Optimizations We observe that the computational bottleneck is to solve the series of QCQPs. To make them more efficient, we use the following two techniques. First, for each QCQP, we rewrite it into a conic programming, which is then solved by MOSEK with a homogeneous primal-dual algorithm. It turns out that solving the QCQP this way is much more efficient in practice.

The second technique reduces the number of QCQPs we have to solve. We observe that as we increase r, the optimal solutions of the QCQPs, hence the $\widehat{F}(\mathbf{I}, r)$'s, are monotonically increasing. Recall that SVT returns the first $r = 2^\ell$ such that $\widehat{F}(\mathbf{I}, 2^\ell) + v_\ell \geq \widetilde{T}$, where v_ℓ is a Laplace noise. Since the first few $\widehat{F}(\mathbf{I}, 2^\ell)$'s are unlikely to go above the threshold, our idea is to give a "jump start" to the SVT by skipping those QCQPs. Let 2^k be the smallest power of 2 no less than $\mathrm{DS}(\mathbf{I})$. By our analysis in the proof of Theorem 6.3.1, the SVT is likely to stop around $r = 2^k$. Our idea is then to first generate v_ℓ for all $\ell = 1, \ldots, k$ in advance, but only compute an $\widehat{F}(\mathbf{I}, 2^\ell)$ if it has a chance to be above $\widetilde{T}$ after adding v_ℓ. By going backward from 2^k to 1 and exploiting the monotonicity of $\widehat{F}(\mathbf{I}, 2^\ell)$, this can eliminate many of them from having to be computed. The detailed algorithm is shown in Algorithm 7. Note that Algorithm 7 is nothing but a more efficient execution of the original SVT, so its privacy and utility guarantees remain the same. Furthermore, this technique can be applied to any SVT instantiation with a sequence of monotonic queries $f_1(\mathbf{I}), f_2(\mathbf{I}), \ldots$, as long as there is a good guess k on the likely stopping position (note that k can depend on private information), so we present Algorithm 7 in a more generic form.

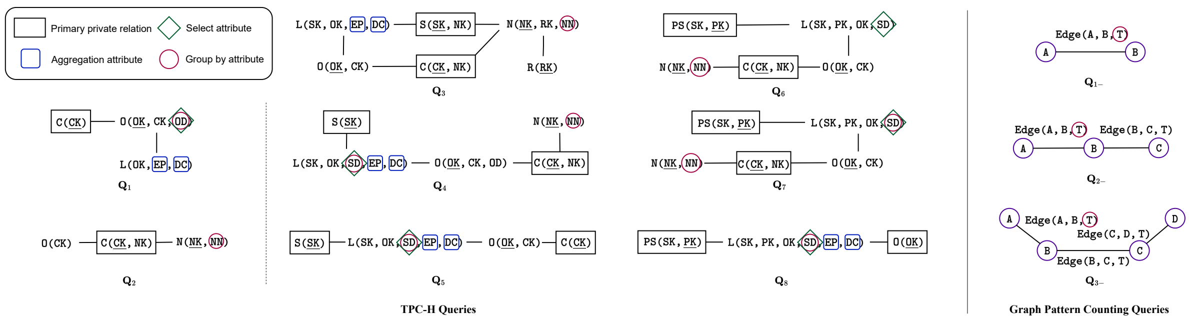


Figure 6.2: The structure of queries.

6.5 Experiments

In this section, we report our experimental results comparing our algorithm (denoted PMSJA) with state-of-the-art algorithms for answering SJA queries with group-by over both benchmark and real-world datasets. For self-join-free queries, we compare with OptMean [46]; for queries with (implicit) self-joins, we compare with R2T [27] combined with advanced composition [38].

6.5.1 Setup

Datasets We use two types of datasets: TPC-H and Stack Overflow network dataset. The TPC-H schema has been discussed before and is shown in Figure 5.4. We use datasets with scale $0.125, 0.25, \ldots, 8$. The default scale is 2, where there are about 15 million tuples.

The Stack Overflow network dataset is from SNAP [58] and records users' interactions on the Stack Overflow website. Here, each node represents one user and the interactions are stored as edges with a timestamp. We use two graphs, corresponding to answer-to-question (**a2q**) and comment-to-answer (**c2a**), respectively. They contain 2,464,606 nodes, 17,823,525 edges, and 1,646,338 nodes, 25,405,374 edges, respectively. We have deleted the top 10% nodes with the highest degrees, as protecting their privacy would introduce too much error. The number of edges of $\textbf{stackoverflow} - \textbf{a2q}$ and $\textbf{stackoverflow} - \textbf{c2a}$ are then reduced to 1,468,092 and 1,425,352.

Queries We use 8 queries over TPC-H schema. The first two are self-join-free queries while the others are self-join queries. The query structures are shown in Figure 6.2. Some of the queries are taken directly from TPC-H benchmark while others are designed to test the algorithms under various settings, in particular different combinations of primary private relations and counting queries are used. Furthermore, for the same TPC-H query, we may use different group-by attributes. That is also to test algorithm under more settings: Different group-by attributes lead to different group sizes d and different data distributions in groups. Another reason is that TPC-H queries have too many or too few groups , i.e., for TPC-H query $Q7$, if adding predicates, there are only 8 groups while removing the predicates will lead to thousands of groups. Therefore, we removed the predicates and used a subset of group-by attributes to get a good d. More precisely, both $\mathbf{Q}_1$, $\mathbf{Q}_2$ correspond to TPC-H query $Q10$ where `Customer` is assigned as the primary private relation but use different *group-by* attributes. $\mathbf{Q}_3$ corresponds to TPC-H query $Q5$. $\mathbf{Q}_4$ and $\mathbf{Q}_5$ correspond to TPC-H query $Q7$ but with different *group-by* attributes. All these three queries have `Customer` and `Supplier` as the primary private relations. Finally, $\mathbf{Q}_6$, $\mathbf{Q}_7$, and $\mathbf{Q}_8$ provide more primary private relation combinations: $\{\text{PartSupp}, \text{Customer}\}$ and $\{\text{PartSupp}, \text{Orders}\}$.

For the *group-by* attributes, we use date attribute or/and nation attribute. There are three cases. In the first case, we use the nation attribute and it has 25 groups. In the second case, we use the date, where we select 100 dates with each one corresponding to one group. We also conduct experiments with various grouping, which will be discussed later. Furthermore, we also use their combinations to make the group: we group the query results by nation and month. To avoid the heavy computations brought by the large group size, we only select two months and the group size here is 50. The group size for each query is shown in the head of Table 6.1.

In another dimension, $\mathbf{Q}_2$, $\mathbf{Q}_6$, $\mathbf{Q}_7$ are counting queries while the others do the sum aggregation over `extendedprice` and `discount` attributes in Lineitem relation. The numbers shown are in thousands.

For graph pattern counting queries, we use edge counting query $\mathbf{Q}_{1-}$, length-2 path counting query $\mathbf{Q}_{2-}$, and length-3 path counting query $\mathbf{Q}_{3-}$. We take the groups on the time attribute on the first edge. Here, we have 10 groups and each group has several dates. Each group in $\mathbf{Q}_{1-}$, $\mathbf{Q}_{2-}$, and $\mathbf{Q}_{3-}$ has 40, 80, 220 dates respectively.

Experimental parameters We conduct all experiments on a Linux server with a 24-core 48-thread 2.2GHz Intel Xeon CPU and 256GB memory. Each program is allowed to use at most 24 threads. We use ℓ_2 metric in the report of query result and the error and we call one mechanism has utility and high utility if the relative error is below 50% and 30% respectively. Each experiment is repeated 20 times and we remove 4 largest errors and 4 smallest errors and report the average error for the rest 12 runs. For the privacy budgets, we use $\varepsilon = 2, 4, 8$ and the default value is set to 4. Compared with the work of answering single query [48, 76, 29, 31, 27], we use larger ε. That is because answering multiple queries is much more complex and we need larger ε to guarantee the utility [1]. We set δ to 1e-7 and the failure probability β to 0.1. Both OptMean and R2T require an $GS_{\mathbf{Q}}$ as the input parameter. For TPC-H queries, we set $GS_{\mathbf{Q}}$ to 1e6. For graph pattern

Query type		Self-join-free queries		Self-join queries					
Query		$\mathbf{Q}_1$	$\mathbf{Q}_2$	$\mathbf{Q}_3$	$\mathbf{Q}_4$	$\mathbf{Q}_5$	$\mathbf{Q}_6$	$\mathbf{Q}_7$	$\mathbf{Q}_8$
Group size d		100	25	25	50	100	25	50	100
Query result	ℓ_2 norm	1,820,000	2,400,000	3,480,000	1,830,000	1,820,000	59,000	43,800	1,820,000
	Time(s)	1.33	4.58	3.85	2.62	1.66	2.12	2.21	3.56
JMSJA	Error(%)	0.504	0.0644	24.6	18.5	20.0	12.4	21.9	10.2
	Time(s)	12.5	75.4	562	4683.8	3056.8	36.3	35.8	68.5
R2T/OptMean	Error(%)	1.2	0.138	99.6	83.7	87.6	56.1	85.7	82.5
	Time(s)	6	65.2	20.7	30.1	24.1	12.0	15.3	26.1

Table 6.1: Comparison between PMSJA and state-of-the-art algorithms (OptMean [46] for self-join-free queries and R2T [27] for self-join queries) on TPC-H queries with *group-by* operator. We use data scale equal to 2, $\varepsilon = 4$ and report the relative error.

Dataset	**stackoverflow – a2q**						**stackoverflow – c2a**					
Query	$\mathbf{Q}_{1-}$		$\mathbf{Q}_{2-}$		$\mathbf{Q}_{3-}$		$\mathbf{Q}_{1-}$		$\mathbf{Q}_{2-}$		$\mathbf{Q}_{3-}$	
	Error(%)	Time(s)	Error(%)	Time(s)	Error(%)	Time(s)	Error(%)	Time(s)	Error(%)	Time(s)	Error(%)	Time(s)
Query Result	48,000	0.735	41,400	1.3	49,500	1.78	34,900	0.61	50,100	1.11	106,000	1.57
PMSJA	5.56	27.3	23	41.4	35.5	81.3	11.7	16	24.8	60.8	36.7	1,943
R2T	13.9	10.7	58.5	10.6	81	12.8	22.3	8.67	60.6	11	77.5	19.5

Table 6.2: Comparison between PMSJA and R2T [27] on graph pattern counting queries with $d = 10$ on different networks. We use $\varepsilon = 4$ and report relative error.

counting queries, we set a degree upper bound of $D = 1,000,000$ and set $\mathrm{GS}_{\mathbf{Q}}$ as the maximum number of graph patterns containing any node, i.e., $\mathrm{GS}_{\mathbf{Q}_{1-}} = D$, $\mathrm{GS}_{\mathbf{Q}_{2-}} = D^2$, and $\mathrm{GS}_{\mathbf{Q}_{3-}} = D^3$.

6.5.2 Experimental Results

Compare with state-of-the-art algorithms We conduct the experiments on the TPC-H dataset and graph data to compare PMSJA with OptMean and R2T and the results are shown in Table 6.1 and Table 6.2 where we report both error level (relative error) and running time. For self-join-free cases, OptMean already has very high utility while PMSJA further reduces this error by more than 50%. This matches our theoretical analyses: PMSJA improves OptMean by a log factor and both of them match the lower bound up to log factors. For efficiency, PMSJA uses a bit more time than OptMean but they are nearly at the same level. That is because, for self-join-free queries, they have very similar processes: extracting the relationship between users and join results first and then finding some threshold to do the clipping with sub-linear time.

Then, we talk about the results of 12 self-join queries and make a comparison between PMSJA with R2T. For the utility, PMSJA has high utility (relative error below 30%) in all 12 queries except the two $\mathbf{Q}_{3-}$, in which PMSJA still has the utility (relative error below 50%). Meanwhile, R2T only has utility in two $\mathbf{Q}_{1-}$ queries, where the group size is small,

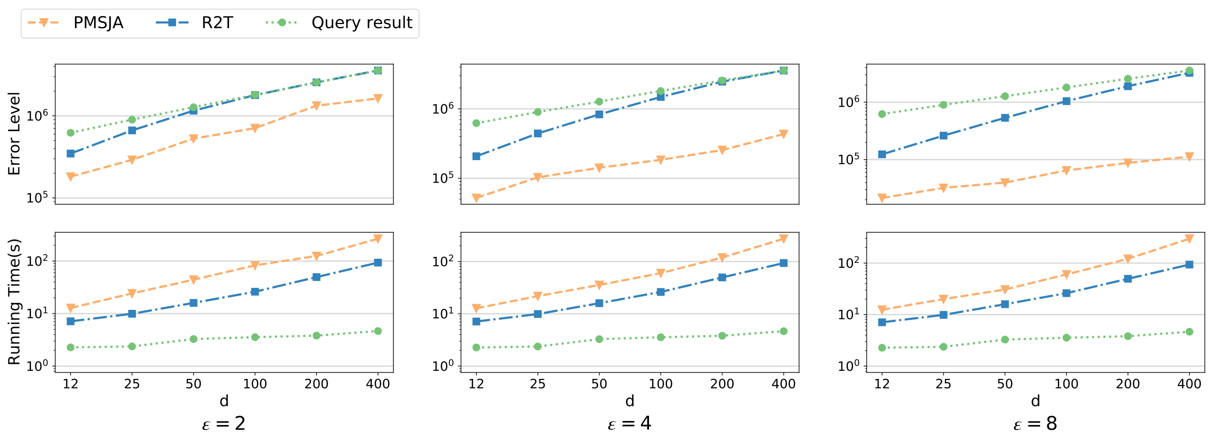
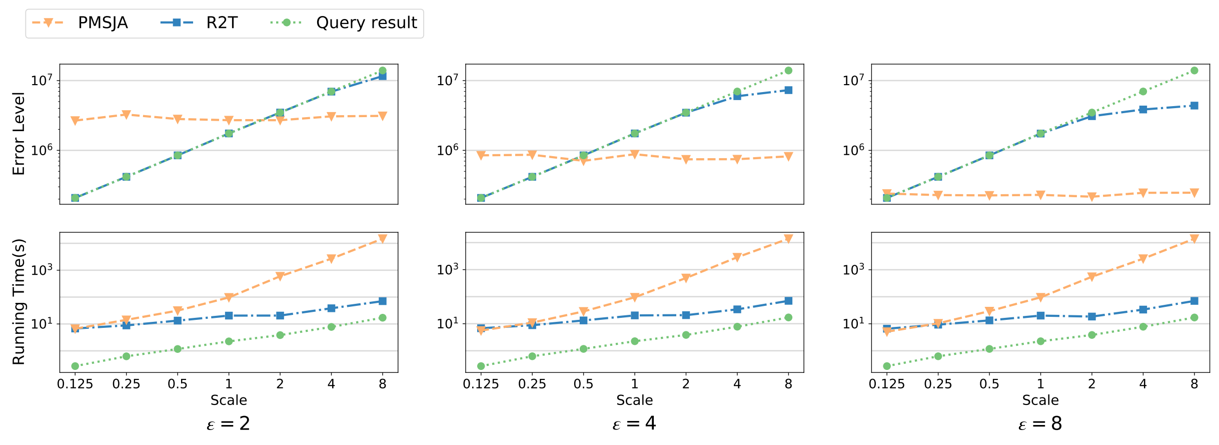


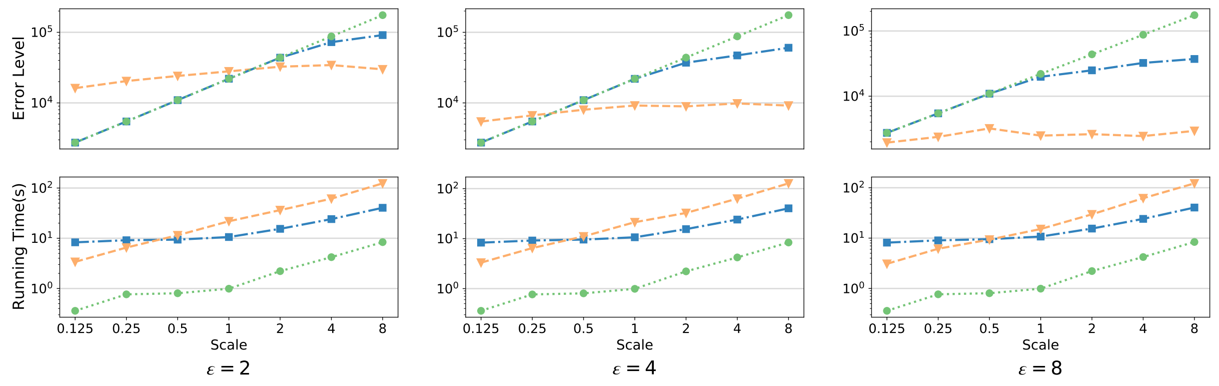
Figure 6.3: Running times and error levels of PMSJA and R2T for $\mathbf{Q}_3$ with different d and ε.

i.e., $d = 10$. More precisely, for TPC-H queries, the error level of R2T is $3.91\times \sim 8.1\times$ of PMSJA and a larger d is more likely to lead to a larger gap. For the graph pattern counting queries where $d = 10$, that ratio is reduced to $1.91\times \sim 2.53\times$. That confirms our theoretical analysis that PMSJA has $\sqrt{d}$ improvement over R2T and matches the lower bound up to log factors. For efficiency, as mentioned before, PMSJA solves several convex QCQP's and needs much more running time than R2T, which only solves a number of LPs. However, as the experiments show, in more than half of the cases (8/12), the running time of PMSJA is not much larger than R2T (less than $8\times$), although it can be much longer in the worst case.

Number of queries To further examine the effects of the change of number of queries d, we use $\mathbf{Q}_3$ but more/fewer dates so we can have different d's. More precisely, we run it with $d = 12, 25, 50, \ldots, 400$ on the TPC-H dataset with scale 2 and ε is set to $2, 4, 8$. We plot both error levels and running times in Figure 6.3, where we also plot the ℓ_2 norm of the query result and its running time. For the error level, first, it is not surprising to see, PMSJA always has an error lower than R2T. As d increases, the error of PMSJA decreases at the same rate as that of the query result: both of them increase with $\sqrt{d}$. Meanwhile, for R2T, its error level increases linearly with d before it catches up with the query result (see the figure with $\varepsilon = 8$). One interesting finding is, when R2T's catches up with the query result, it increases at the same rate as the query result and will not surpass that (see the figures with $\varepsilon = 2$ and $\varepsilon = 4$). That is because R2T will always return a noised value between 0 and the real query result thus its error is at most the



(a) $\mathbf{Q}_3$



(b) $\mathbf{Q}_7$

Figure 6.4: Running times and error levels of PMSJA and R2T with different queries, data scales and ε.

query result. However, this does not really have any benefit since we have already lost all utility when the error level reaches the query result.

On the other hand, we see the running time of real query sub-linearly increases with d. By contrast, R2T and PMSJA have linear and sup-linear speeds respectively. That is because R2T runs each query independently and for every single query, increasing d will only affect the assigned ε, which just brings a minor effect on the running time thus the running time has a linear dependency on d. Meanwhile, for PMSJA, increasing d will complicate the convex QCQP's it solves thus leading to a super-linear effect on the running time.

Scalability Lastly, we conduct the experiments to see the effects of data scale changes. We use TPC-H datasets with scale $0.125, 0.2, \ldots, 8$ and run $\mathbf{Q}_3$ and $\mathbf{Q}_7$ with $\varepsilon = 2, 4, 8$. The results are shown in Figure 6.4a and 6.4b.

First, the error level of DPSJA barely changes with the data scale. That is because theoretically, it only depends on $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$, which does not change much by the scale of TPC-H data. On the other hand, the error level of R2T first increases with query result but will then stay at some level. That is because its error guarantee also depends on $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$ and as mentioned before, its error is also bounded by the query result.

For efficiency, both query and R2T have running time that increases linearly with the data scale while the running time of PMSJA has a super-linear dependency on the data scale.

CHAPTER 7

CONCLUSION

This thesis presents my contributions to answering SQL queries under both the tuple-DP and user-DP models. Under the tuple-DP framework, I adopt the notion of neighborhood optimality, a fairly strong instance-specific notation, and propose the residual sensitivity mechanism for answering SPJA queries with an $O(1)$-neighborhood optimal error. The residual sensitivity approach exhibits desirable efficiency and can easily be integrated into any SQL engine. In the user-DP framework, I highlight the limitations of neighborhood optimality and propose a stronger optimality notation referred to as down-neighborhood optimality. I first address SA queries and propose a solution that removes the boundness assumption for the user contribution and leads to an improvement in the error bound by several logarithmic factors. Then, I extend my work to the general SPJA queries and propose R2T, the first algorithm for this type of queries. R2T has been shown to achieve the down-neighborhood optimal error. Finally, I explore the problem of answering d SJA queries and present a solution that improves the state-of-the-art error by a factor of $\sqrt{d}$.

Looking forward, two promising avenues for research are outlined. First, most existing works in answering SQL queries under DP have focused on static data, whereas in practice, the data are continually updated and the adversary draws a continuous observation of the query results, presenting new challenges. To date, works [34, 21, 35, 23] have been limited to addressing counting queries over a single relation under tuple-DP, achieving error bounds that match the static setting up to logarithmic factors. However, no known work exists for more complex queries, and while privacy composition is a trivial solution, it leads to a $\sqrt{T}$-factor reduction in utility compared to the static setting, where T is the number of time units.

Second, combining DP with secure computation in answering SQL queries is another interesting direction. Secure computation requires full obliviousness, meaning intermediate computations should not leak any information about the input data except for negligible probabilities. On the other hand, differential privacy can mask the final result, but does not protect intermediate computations. Combining these two notions has the potential to provide stronger privacy protection, by fully obliviously computing a query and then running a differential privacy mechanism in a secure environment to mask the query result. The first step of this combination has been well-researched, but the second step, especially for answering SQL queries, still requires further exploration.

Bibliography

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*, volume 8. Addison-Wesley Reading, 1995.

[3] Mahmoud Abo Khamis, Hung Ngo, and Dan Suciu. What do shannon-type inequalities, submodular width, and disjunctive datalog have to do with one another? In *Proc. ACM Symposium on Principles of Database Systems*, 2017.

[4] Mahmoud Abo Khamis, Hung Q Ngo, and Atri Rudra. Faq: questions asked frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 13–28, 2016.

[5] Ishaq Aden-Ali, Hassan Ashtiani, and Gautam Kamath. On the sample complexity of privately learning unbounded high-dimensional gaussians. In *Algorithmic Learning Theory*, pages 185–216. PMLR, 2021.

[6] Kareem Amin, Alex Kulesza, Andres Munoz, and Sergei Vassilvtiskii. Bounding user contributions: A bias-variance trade-off in differential privacy. In *International Conference on Machine Learning*, pages 263–271. PMLR, 2019.

[7] Galen Andrew, Om Thakkar, H Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871*, 2019.

[8] Myrto Arapinis, Diego Figueira, and Marco Gaboardi. Sensitivity of counting queries. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2016.

[9] Hassan Ashtiani and Christopher Liaw. Private and polynomial time algorithms for learning gaussians and beyond. *arXiv preprint arXiv:2111.11320*, 2021.

[10] Hilal Asi and John C Duchi. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. *Advances in neural information processing systems*, 33, 2020.

[11] Albert Atserias, Martin Grohe, and Dániel Marx. Size bounds and query plans for relational joins. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 739–748. IEEE, 2008.

[12] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 273–282, 2007.

[13] Sourav Biswas, Yihe Dong, Gautam Kamath, and Jonathan Ullman. Coinpress: Practical private mean and covariance estimation. *Advances in Neural Information Processing Systems*, 33, 2020.

[14] Jaroslaw Błasiok, Mark Bun, Aleksandar Nikolov, and Thomas Steinke. Towards instance-optimal private query release. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2480–2497. SIAM, 2019.

[15] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96, 2013.

[16] Gavin Brown, Marco Gaboardi, Adam Smith, Jonathan Ullman, and Lydia Zakynthinou. Covariance-aware private mean estimation without private covariance estimation. *Advances in Neural Information Processing Systems*, 34, 2021.

[17] Mark Bun, Gautam Kamath, Thomas Steinke, and Steven Z Wu. Private hypothesis selection. *Advances in Neural Information Processing Systems*, 32, 2019.

[18] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[19] Mark Bun and Thomas Steinke. Average-case averages: Private algorithms for smooth sensitivity and mean estimation. In *Advances in Neural Information Processing Systems 32*, NeurIPS '19, pages 181–191. Curran Associates, Inc., 2019.

[20] T Tony Cai, Yichen Wang, and Linjun Zhang. The cost of privacy: Optimal rates of convergence for parameter estimation with differential privacy. *arXiv preprint arXiv:1902.04495*, 2019.

[21] T-H Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)*, 14(3):1–24, 2011.

[22] Shixi Chen and Shuigeng Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 653–664, 2013.

[23] Rachel Cummings, Sara Krehbiel, Kevin A Lai, and Uthaipon Tantipongpipat. Differential privacy for growing databases. *Advances in Neural Information Processing Systems*, 31, 2018.

[24] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*, pages 123–138, 2016.

[25] Apple Differential Privacy Team. Learning with privacy at scale. `https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf`. December 2017.

[26] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *NIPS*, 2017.

[27] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. R2T: Instance-optimal truncation for differentially privatequery evaluation with foreign keys. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2022.

[28] Wei Dong, Dajun Sun, and Ke Yi. Better than composition: How to answer multiple relational queries under differential privacy. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2023.

[29] Wei Dong and Ke Yi. Residual sensitivity for differentially private multi-way joins. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2021.

[30] Wei Dong and Ke Yi. Universal private estimators. *arXiv preprint arXiv:2111.02598*, 2021.

[31] Wei Dong and Ke Yi. A nearly instance-optimal differentially private mechanism for conjunctive queries. In *Proc. ACM Symposium on Principles of Database Systems*, 2022.

[32] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 371–380, 2009.

[33] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[34] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724, 2010.

[35] Cynthia Dwork, Moni Naor, Omer Reingold, and Guy N Rothblum. Pure differential privacy for rectangle queries via private partitions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 735–751. Springer, 2015.

[36] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390, 2009.

[37] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[38] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[39] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM*

SIGSAC conference on computer and communications security, pages 1054–1067, 2014.

[40] R. Fagin, A. Lotem, and M. Noar. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, 66:614–656, 2003.

[41] Juanru Fang, Wei Dong, and Ke Yi. Shifted inverse: A general mechanism for monotonic functions under user differential privacy. 2022.

[42] Georg Gottlob, Stephanietien Lee, Gregory Valiant, and Paul Valiant. Size and treewidth bounds for conjunctive queries. *Journal of the ACM*, 59(3), 2012.

[43] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems*, pages 2339–2347, 2012.

[44] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1), 2010.

[45] Samuel B Hopkins, Gautam Kamath, and Mahbod Majid. Efficient mean estimation with pure differential privacy via a sum-of-squares exponential mechanism. *arXiv preprint arXiv:2111.12981*, 2021.

[46] Ziyue Huang, Yuting Liang, and Ke Yi. Instance-optimal mean estimation under differential privacy. *Advances in Neural Information Processing Systems*, 2021.

[47] Manas R Joglekar, Rohan Puttagunta, and Christopher Ré. Ajar: Aggregations and joins over annotated relations. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 91–106, 2016.

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

[49] Gautam Kamath, Jerry Li, Vikrant Singhal, and Jonathan Ullman. Privately learning high-dimensional distributions. In *Proceedings of the 32nd Annual Conference on Learning Theory*, COLT '19, pages 1853–1902, 2019.

[50] Gautam Kamath, Argyris Mouzakis, Vikrant Singhal, Thomas Steinke, and Jonathan Ullman. A private and computationally-efficient estimator for unbounded gaussians. *arXiv preprint arXiv:2111.04609*, 2021.

[51] Gautam Kamath, Vikrant Singhal, and Jonathan Ullman. Private mean estimation of heavy-tailed distributions. In *Conference on Learning Theory*, pages 2204–2235. PMLR, 2020.

[52] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11):1146–1157, 2011.

[53] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *ACM Transactions on Database Systems (TODS)*, 39(3):1–33, 2014.

[54] Vishesh Karwa and Salil Vadhan. Finite sample differentially private confidence intervals. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[55] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*, pages 457–476. Springer, 2013.

[56] Pravesh K Kothari, Pasin Manurangsi, and Ameya Velingker. Private robust estimation by stabilizing convex relaxations. *arXiv preprint arXiv:2112.03548*, 2021.

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

[58] Jure Leskovec and Andrej Krevl. Snap datasets: Stanford large network dataset collection (2014). *URL http://snap. stanford. edu/data*, page 49, 2016.

[59] Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal*, 24(6):757–781, 2015.

[60] Xiyang Liu, Weihao Kong, and Sewoong Oh. Differential privacy and robust statistics in high dimensions. *arXiv preprint arXiv:2111.06578*, 2021.

[61] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In *2008 IEEE 24th international conference on data engineering*, pages 277–286. IEEE, 2008.

[62] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

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

[64] Shanmugavelayutham Muthukrishnan and Aleksandar Nikolov. Optimal private half-space counting via discrepancy. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1285–1292, 2012.

[65] Arjun Narayan and Andreas Haeberlen. Djoin: Differentially private join queries over distributed databases. In *USENIX Symposium on Operating Systems Design and Implementation*, pages 149–162, 2012.

[66] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 351–360, 2013.

[67] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84, 2007.

[68] Catuscia Palamidessi and Marco Stronati. Differential privacy for relational algebra: Improving the sensitivity bounds via constraint systems. In *QAPL*, 2012.

[69] Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X Yu, Sashank J Reddi, and Sanjiv Kumar. Adaclip: Adaptive clipping for private sgd. *arXiv preprint arXiv:1908.07643*, 2019.

[70] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating data to sensitivity in private data analysis. *Proceedings of the VLDB Endowment*, 7(8), 2014.

[71] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1435–1446.

[72] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment*, 6(14):1954–1965, 2013.

[73] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1435–1446, 2014.

[74] Yuan Qiu, Wei Dong, Ke Yi, Bin Wu, and Feifei Li. Releasing private data for numerical queries. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1410–1419, 2022.

[75] Adam Smith. Privacy-preserving statistical estimation with optimal convergence rates. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 813–822, 2011.

[76] Yuchao Tao, Xi He, Ashwin Machanavajjhala, and Sudeepa Roy. Computing local sensitivities of counting queries with joins. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 479–494, 2020.

[77] Salil Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017.

[78] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on knowledge and data engineering*, 23(8):1200–1214, 2010.

[79] Ganzhao Yuan, Yin Yang, Zhenjie Zhang, and Zhifeng Hao. Convex optimization for linear query processing under approximate differential privacy. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2005–2014, 2016.

[80] Ganzhao Yuan, Zhenjie Zhang, Marianne Winslett, Xiaokui Xiao, Yin Yang, and Zhifeng Hao. Optimizing batch linear queries under exact and approximate differential privacy. *ACM Transactions on Database Systems (TODS)*, 40(2):1–47, 2015.

[81] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Private release of graph statistics using ladder functions. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 731–745, 2015.

[82] Xiaojian Zhang, Rui Chen, Jianliang Xu, Xiaofeng Meng, and Yingtao Xie. Towards accurate histogram publication under differential privacy. In *Proceedings of the 2014 SIAM international conference on data mining*, pages 587–595. SIAM, 2014.

APPENDIX A

LIST OF PUBLICATIONS

1. **Wei Dong**, Dajun Sun, and Ke Yi. "Better than Composition: How to Answer Multiple Relational Queries under Differential Privacy." ACM SIGMOD International Conference on Management of Data (SIGMOD), June 2023.

2. **Wei Dong**, Ke Yi. "Universal Private Estimators." ACM Symposium on Principles of Database Systems (PODS), June 2023.

3. **Wei Dong**, Qiyao Luo, and Ke Yi. "Continual Observation under User-level Differential Privacy." IEEE Symposium on Security and Privacy (S&P), May 2023.

4. **Wei Dong**, Yuting Liang, Ke Yi. "Differentially Private Covariance Revisited." Advances in Neural Information Processing Systems (NeurIPS), November 2022.

5. Juanru Fang, **Wei Dong**, Ke Yi. "Shifted Inverse: A General Mechanism for Monotonic Functions under User Differential Privacy." ACM Conference on Computer and Communications Security (CCS), November 2022.

6. Yuan Qiu, **Wei Dong**, Ke Yi, Bin Wu and Feifei Li. "Releasing Private Data for Numerical Queries." ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), August 2022.

7. **Wei Dong**, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. "R2T: Instance-optimal Truncation for Differentially Private Query Evaluation with Foreign Keys." ACM SIGMOD International Conference on Management of Data (SIGMOD), June 2022.

8. **Wei Dong**, Ke Yi. "A Nearly Instance-optimal Differentially Private Mechanism for Conjunctive Queries." ACM Symposium on Principles of Database Systems (PODS), June 2022.

9. **Wei Dong**, Ke Yi. "Residual Sensitivity for Differentially Private Multi-Way Joins." ACM SIGMOD International Conference on Management of Data (SIGMOD), June 2021.