Better than Composition: How to Answer Multiple Relational Queries under Differential Privacy

Wei Dong Dajun Sun Ke Yi
Hong Kong University of Science and Technology
Hong Kong, China
{wdongac,dsunad,yike}@cse.ust.hk

ABSTRACT

Answering relational queries under differential privacy has attracted a lot of attention in recent years due to growing concerns on personal privacy, and instance-optimal mechanisms have been developed for a single query. However, most real-world data analytical tasks require multiple queries to be answered under a total privacy budget. The standard solution to extend the single-query mechanism to multiple queries is via privacy composition. However, we observe that this may yield an error bound that could be a $\sqrt{d}$-factor worse from the optimal, where d is the number of queries. In this paper, we present a different, more holistic approach that closes this gap. In addition to theoretical optimality, our new mechanism also significantly outperforms privacy composition in practice, especially on more skewed data and large d.

CCS CONCEPTS

• Information systems → Database query processing; **• Security and privacy → Database and storage security**; **• Theory of computation → Theory of database privacy and security**.

KEYWORDS

Differential privacy; SJA query; Multiple queries

ACM Reference Format:
Wei Dong Dajun Sun Ke Yi. 2018. Better than Composition: How to Answer Multiple Relational Queries under Differential Privacy. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Query answering under *differential privacy (DP)* has been studied extensively in the last 20 years. To date, most problems over a flat table have been relatively well solved. A single (counting or linear) query can be easily answered by the classical Laplace mechanism or the Gaussian mechanism [17], so most efforts have been devoted to the problem of answering a set of d queries. There are two general approaches to this multi-query problem. The first is *privacy composition*, i.e., we divide the privacy budget to the d queries and answer each query with the single-query mechanism. Using advanced composition [18], the utility suffers an $\tilde{O}(\sqrt{d})$-factor degradation[1], which is the best we can if the queries are arbitrary linear queries, unless d is larger than the size of the database [19]. The other approach is to exploit some special structures of the queries. For instance, if all queries are ranges queries, then the $\tilde{O}(\sqrt{d})$-factor degradation can be reduced to $\tilde{O}(1)$ [15, 29, 41].

1.1 DP in Relational Databases

The situation becomes more complicated in a relational database with multiple relations (tables), mostly due to two challenges.

Challenge 1: Unbounded global sensitivity. First, unlike a flat table, a relational schema models complicated relationships among different types of entities, not all of which are equally positioned in terms of privacy protection. Consider the TPC-H schema. We usually take `Customer` or `Supplier` (or both) as the private entities, often called "users", while `Nation` and `Region` are public. Furthermore, any tuple that has a foreign key (FK) reference (directly or indirectly) to a user, such as a lineitem in an order placed by a customer, is considered as data belonging to the user. This results in what is known as *user-level DP*, as opposed to *tuple-level DP* used over a flat table. The formal user-level DP definition in relational databases is deferred to Section 3.1; intuitively, this creates a difficulty to the Laplace/Gaussian mechanism, since two neighboring instances that differ by one user (e.g., a customer or a supplier) may differ by arbitrarily many tuples (e.g., lineitems), thus making the *global sensitivity* of the query, GS_Q, unbounded.

When GS_Q is unbounded, a common approach is to consider the *local sensitivity* [40], or more appropriately, the *downward sensitivity* [10]. Informally (formal definition given in Section 3.3), for a given query Q, the downward sensitivity of a user u, denoted $\mathrm{DS}_Q(u)$, is the contribution to Q from all the data belonging to u. The downward sensitivity of a database instance $\mathbf{I}$, denoted $\mathrm{DS}_Q(\mathbf{I})$, is the maximum $\mathrm{DS}_Q(u)$ over all $u \in \mathbf{I}$. Then, one may use the *truncation* mechanism [27]: For a truncation threshold r, delete all users u and their data where $\mathrm{DS}_Q(u) > r$, and then apply the Laplace/Gaussian mechanism with sensitivity r. The optimal value of r is $\mathrm{DS}_Q(\mathbf{I})$, which results in an instance-optimal error of $O(\mathrm{DS}_Q(\mathbf{I}))$. However, but this optimal truncation threshold cannot be used directly as it is sensitive to $\mathbf{I}$. Thus, the main challenge is to find a near-optimal r in a differentially private manner.

Challenge 2: Self-joins. When the query Q has no self-joins, the truncation mechanism would work (after finding a good r). However, self-joins create another challenge, as they introduce correlations among the users: Truncating one user's data may change the

[1]The $\tilde{O}$ notation suppresses the dependency on ε and polylogarithmic factors.

downward sensitivities of other users, which results in the truncation mechanism violating privacy. This issue has been identified in [10], who then propose a linear-program-based solution that fixes the problem while achieving the optimal error $\tilde{O}(\mathrm{DS}_Q(\mathbf{I}))$. Furthermore, as explained in Section 3.2, the issue with self-joins is more prominent under DP: many queries without explicit self-joins actually induce implicit self-joins due to the privacy requirement. In particular, self-joins are always introduced when the a join result references more than one user, which is a common scenario in relational databases.

1.2 From a Single Query to Multiple Queries

Answering a single query is not very useful in practice. Thus, as with the flat table case, 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 $\mathbf{Q} = (Q_1, \dots, Q_d)$ be the d 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 paper.

In this paper, we will allow the queries to contain arbitrary joins and selection predicates. As we do not assume any restricted form of the joins and predicates, the only known solution is to use privacy composition. By advanced composition, we can allocate a privacy budget of $\tilde{O}(1/\sqrt{d})$ to each query and invoke the single-query mechanism in [10]. This leads to an error of $\tilde{O}(\sqrt{d} \cdot \mathrm{DS}_{Q_k}(\mathbf{I}))$ for Q_k, hence an RMSE of

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

Challenge/opportunity 3: Better than composition. We make the crucial observation that the error bound in (1) is *not* optimal. In Section 4, we show that the lower bound for the multi-query problem is $\tilde{\Omega}\left(\sqrt{d} \cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})\right)$, where $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$, informally speaking, is the largest contribution of any user in $\mathbf{I}$ to the d query results measured in ℓ_2 norm. 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) 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.

1.3 Our Results

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 queries. We observe that 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 [5, 21, 23]. 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 [21] is

$$O\left(\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}) \cdot \left(\sqrt{d} + \sqrt{\log(\mathrm{GS}_{\mathbf{Q}}) \log\log(\mathrm{GS}_{\mathbf{Q}})}\right) \cdot \sqrt{\log(1/\delta)}/\varepsilon\right), \tag{2}$$

where ε, δ are the privacy parameters (see Section 3.3). Thus, they do not satisfactorily solve challenge 1. Our first result in this paper is the complete removal of the dependency on $\mathrm{GS}_{\mathbf{Q}}$, i.e., we do not impose any restrictions on the database instance $\mathbf{I}$, effectively allowing $\mathrm{GS}_{\mathbf{Q}} = \infty$. Specifically, in Section 4 we design an algorithm that achieves an error of

$$O\left(\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}) \cdot \left(\sqrt{d \log(1/\delta)} + \log\log(\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}))\right)/\varepsilon\right). \tag{3}$$

Note that even assuming a finite $\mathrm{GS}_{\mathbf{Q}}$, the error bound of (3) is better than (2) 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. Self-joins are difficult to handle, since they make the truncation mechanism fail. To tackle self-joins, R2T [10] uses a series of linear programs (LPs), which can be considered as the more general forms of the LPs used in private graph analysis [25]. However, as we explain in Section 5.1, 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

$$O\left(\sqrt{d} \cdot \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}) \cdot \sqrt{\log(e^{\varepsilon}/\delta)} \cdot \left(\log\log(\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})) + \log(e^{\varepsilon}/\delta)\right)/\varepsilon^2\right),$$

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 rewrites 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 [10], especially on more skewed data and large d.

$$\begin{tikzcd}
 & & \texttt{Lineitem(OK, SK, PtK)} \arrow[dl] \arrow[dr] & \\
 & \texttt{Partsupp(}\underline{\texttt{PtK}}\texttt{, }\underline{\texttt{SK}}\texttt{)} \arrow[dl] \arrow[d] & & \texttt{Orders(}\underline{\texttt{OK}}\texttt{, CK)} \arrow[d] \\
\texttt{Part(}\underline{\texttt{PtK}}\texttt{)} & \texttt{Supplier(}\underline{\texttt{SK}}\texttt{, NK)} \arrow[dr] & & \texttt{Customer(}\underline{\texttt{CK}}\texttt{, NK)} \arrow[dl] \\
 & & \texttt{Nation(}\underline{\texttt{NK}}\texttt{, RK)} \arrow[d] & \\
 & & \texttt{Region(}\underline{\texttt{RK}}\texttt{)} &
\end{tikzcd}$$

Figure 1: The TPC-H schema and its FK constraints. The underlined attributes are the PKs. Not all attributes are shown.

2 RELATED WORK

In the past several years, query answering under differential privacy [14] has attracted a lot of attention [3, 10, 11, 13, 22, 27, 30, 32, 35, 36, 40]. Early works did not consider FK constraints, or equivalently, they adopt a DP model that only protects the tuples, which is called tuple-DP. Starting from [27], people began to consider user-DP modeled by FK constraints. Under user-DP, self-join-free queries [27, 40] are actually equivalent to the sum (mean) estimation problem [2, 12, 21]. Very recently, R2T [10] is proposed to deal with queries with self-joins while achieving instance optimality.

There are also a number works studying graph pattern counting queries under differential privacy [7, 9, 24, 26, 34, 44], which is an important special case of SJA queries. For graph data, there are two DP policies: edge-DP [7, 24, 34, 44] and node-DP [7, 9, 26]. 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 [4, 6, 15, 19, 20, 29, 31, 33, 37–39, 41–43, 45]. 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 $\tilde{O}(\sqrt{n})$ [19]. For a set of queries with special structures, the error can be further reduced [15, 31, 41]. Furthermore, [6, 29, 33] design mechanisms that are optimal for any given query set.

3 PRELIMINARIES

3.1 Database with FK Constraints

We first review the DP definition in databases with FKs [27]. Let $\mathbf{R}$ be the database schema. When foreign-key constraints exist, $\mathbf{R}$ can be formalized as a directed acyclic graph, where each node corresponds to a relation $R \in \mathbf{R}$ and a directed edge from R' to R indicates an FK reference from an attribute of R' to the PK of R. One relation is designated as the *primary private relations* R_P (this is without loss of generality; see example below), while a relation having a direct or indirect FK referencing R_P is called a *secondary private relation*; relations having no FK references to R_P are public.

Example 3.1. Consider the TPC-H schema in Figure 1 : Suppose we want to protect the privacy of both the customers and the suppliers. Then we can add a virtual relation $\texttt{User}(\underline{\texttt{ID}})$, which includes all the PKs in `Customer` and `Supplier`, while adding FK constraints from the PKs of `Customer`, `Supplier` to $\underline{\texttt{ID}}$. Then, `User` becomes the only primary private relation, while `Customer`, `Supplier`, `Lineitem`, `Order`, `Partsupp` are secondary private relations, and `Part`, `Nation`, `Region` are public. □

Let $\mathbf{I}$ be a database instance over $\mathbf{R}$. For any $R \in \mathbf{R}$, let $\mathbf{I}(R)$ be the relation instance of R in $\mathbf{I}$. The reference relationships over tuples are defined naturally as follows: For $t \in \mathbf{I}(R)$ and $t' \in \mathbf{I}(R')$, we say t' references t if (1) R' references R and the FK of t' is same as the PK of t; or (2) there exists another t'' such that t' references t'' and t'' references t. Let $N = |\mathbf{I}(R_P)|$ be the number of private users. For $i \in [N]$, let $t_i(\mathbf{I})$ be the ith user in $\mathbf{I}(R_P)$. We write $\mathbf{I}' \subseteq \mathbf{I}$ if $\mathbf{I}'(R_P) \subseteq \mathbf{I}(R_P)$, and for each tuple t in a secondary relation, it is included in $\mathbf{I}'$ iff all tuples in $\mathbf{I}(R_P)$ that are referenced by t are included in $\mathbf{I}'(R_P)$. This way, it generalizes the notion of an "induced subgraph" and we can call $\mathbf{I}'$ an induced sub-instance of $\mathbf{I}$. Note that an induced sub-instance $\mathbf{I}'$ is completely specified by $\mathbf{I}'(R_P)$. Besides, let $[n] = \{1, 2, \dots, n\}$.

3.2 SJA Queries

We now define the class of queries considered. We start with a multi-way (natural) join:

$$J := R_1(\mathbf{x}_1) \bowtie \cdots \bowtie R_n(\mathbf{x}_n), \tag{4}$$

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. Let $\mathbf{x} := \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_n$. When self-joins are present, there can be repeats among $R_1, \dots, R_n$, i.e., $R_i = R_j$ (then we must have $\mathbf{x}_i \neq \mathbf{x}_j$). The join is required to be *complete* [27], i.e., if any R_i in J references another R', then R' must also be included in J, with its PK given a variable that is the same as the variable given to the corresponding FK of R_i.

Let $M = |J(\mathbf{I})|$ be the join size, let for $j \in [M]$, $q_j(\mathbf{I})$ denotes the jth join result. We say $q_j(\mathbf{I})$ references $t_i(\mathbf{I})$ iff $q_j(\mathbf{I}) \bowtie t_i(\mathbf{I}) \neq \emptyset$. To better describe the referencing relationships between tuples and join results, we introduce the following sets of indices $C_i(\mathbf{I})$ and $D_j(\mathbf{I})$ for each $i \in [N], j \in [M]$:

$$C_i(\mathbf{I}) := \{j : q_j(\mathbf{I}) \text{ references } t_i(\mathbf{I})\}, \tag{5}$$

$$D_j(\mathbf{I}) := \{i : q_j(\mathbf{I}) \text{ references } t_i(\mathbf{I})\}. \tag{6}$$

A JA or SJA query Q aggregates over the join results $J(\mathbf{I})$. They can both be captured by introducing a function $\psi : \mathbf{dom}(\mathbf{x}) \to \mathbb{N}$ and defining the query result on $\mathbf{I}$ as

$$Q(\mathbf{I}) := \sum_{j \in [M]} \psi(q_j(\mathbf{I})). \tag{7}$$

Note that if there is a selection predicate, we can simply set $\psi(q) = 0$ for any q that does not satisfy the predicate. We also use the shorthand $\psi_j(\mathbf{I}) := \psi(q_j(\mathbf{I}))$.

Example 3.2. Continuing with Example 3.1, suppose we ask the following query:

```
SELECT count(*) FROM Lineitem WHERE Lineitem.QTY > 10
```

Although this query does not have any explicit joins, we must make it complete, resulting in the following join:

$$J = \texttt{Orders(OK,CK)} \bowtie \texttt{Lineitem(OK,SK,PtK,QTY)} \bowtie \texttt{Customer(CK,NK)}$$
$$\bowtie \texttt{User(CK)} \bowtie \texttt{PartSupp(PtK,SK)} \bowtie \texttt{Supplier(SK,NK)} \bowtie \texttt{User(SK)}$$

Note how it introduces a self-join on User. Generally speaking, self-joins will be introduced as long as a join result references more than one private user. In fact, the join with the two copies of User has no effect on the join results; their purpose is to just mark CK and SK as variables representing the private users. To finish the query, we define $\psi(q) = 1$ for each lineitem q in the join result whose QTY is larger than 10, otherwise 0. Then $C_i(\mathbf{I})$ consists of all the lineitems belonging to the i-th user (a customer or a supplier), and $D_j(\mathbf{I})$ consists of all the users that the j-th lineitem references (exactly one customer and one supplier in this case). □

Example 3.3. Many graph pattern matching queries under node-DP [7, 9, 26] can be written as SJA queries on the schema $\mathbf{R} = \{\texttt{Node}(\underline{\texttt{ID}}), \texttt{Edge}(\texttt{src},\texttt{dst})\}$, where src, dst are both FKs referencing ID. For example, the triangle counting query uses the join

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

with $\psi(q) = 1$ for all $q \in J(\mathbf{I})$. Note that the 3 copies of Node are not really needed, but they are included (virtually) to signify that A, B, C all represent private users. For this query, $C_i(\mathbf{I})$ includes all the triangles incident to the i-th node, and $D_j(\mathbf{I})$ is the 3 nodes forming the j-th triangle.

In this paper, we consider answering d such queries $\mathbf{Q} = (Q_1, \dots, Q_d)$. 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 3.2, 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.

3.3 Differential Privacy

Definition 3.4 (Differential privacy). For $\varepsilon, \delta > 0$, an algorithm $\mathcal{M} : \mathcal{I} \to \mathcal{Y}$ is (ε,δ)-differentially private if for any neighboring instances $\mathbf{I} \sim \mathbf{I}' \in \mathcal{I}$ and any subset of outputs $Y \subseteq \mathcal{Y}$,

$$\Pr[\mathcal{M}(\mathbf{I}) \in Y] \le e^{\varepsilon} \cdot \Pr[\mathcal{M}(\mathbf{I}') \in Y] + \delta.$$

The privacy parameter ε is usually a constant from 0.1 to 10, while $\delta \ll 1/N$. To apply the DP definition to a concrete problem, we need to define the neighboring relationship. We adopt the user-DP policy in [27] for relational database, which defines two instances $\mathbf{I}$ and $\mathbf{I}'$ as neighbors if $\mathbf{I}'$ can be obtained from $\mathbf{I}$ by deleting some $t_P \in \mathbf{I}(R_P)$ and all tuples referencing t_P.

Lemma 3.5 (Laplace Mechanism [14]). *Given a single query $Q : \mathcal{I} \to \mathbb{R}$, let $\mathrm{GS}_Q := \max_{\mathbf{I}\sim\mathbf{I}'} |Q(\mathbf{I}) - Q(\mathbf{I}')|$. The mechanism*

$$\mathcal{M}(\mathbf{I}) = Q(\mathbf{I}) + \frac{\mathrm{GS}_Q}{\varepsilon} \cdot Y,$$

where $Y \sim \mathrm{Lap}(1)$, preserves $(\varepsilon, 0)$-DP.

For a vectored-valued query, the *Gaussian mechanism* is more commonly used:

Lemma 3.6 (Gaussian Mechanism [8, 14]). *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. If it is large or unbounded, one may consider the *local sensitivity* $\mathrm{LS}_{\mathbf{Q}}(\mathbf{I}) = \max_{\mathbf{I}':\mathbf{I}'\sim\mathbf{I}} \|\mathbf{Q}(\mathbf{I}) - \mathbf{Q}(\mathbf{I}')\|$, and try to obtain a DP upper bound of $\mathrm{LS}_{\mathbf{Q}}(\mathbf{I})$ before applying the mechanism. Karwa et al. [25] show how this can be done for the Laplace mechanism. Here we obtain a similar result for the Gaussian mechanism:

Lemma 3.7 (Second-order Gaussian Mechanism). *Given a d-dimensional query $\mathbf{Q} : \mathcal{I} \to \mathbb{R}^d$, suppose there is an (ε_1,δ_1)-DP mechanism that outputs an $\widehat{\mathrm{LS}}_{\mathbf{Q}}(\mathbf{I}) \ge \mathrm{LS}_{\mathbf{Q}}(\mathbf{I})$ 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 [25]. □

However, in a relational database, $\mathrm{LS}_{\mathbf{Q}}(\mathbf{I})$ is also unbounded, since one can construct an $\mathbf{I}'$ by adding to $\mathbf{I}$ a user t_P with arbitrarily many tuples all referencing t_P. Thus, Dong et al. [10] propose to consider the *downward sensitivity*:

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

Note by adding the restriction $\mathbf{I}' \subseteq \mathbf{I}$, $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$ corresponds to the maximum contribution of any user in $\mathbf{I}$, which is always finite. Specifically, for each user $t_i(\mathbf{I}) \in \mathbf{I}(R_P)$, let

$$S_{k,i}(\mathbf{I}) = \sum_{j\in C_{k,i}(\mathbf{I})} \psi_{k,j}(\mathbf{I}) \tag{8}$$

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 $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}) = \max_{i\in[N]} \|\mathbf{S}_i(\mathbf{I})\|$.

3.4 Sparse Vector Technique

The *Sparse Vector Technique* (SVT) [16] has as input a (possibly infinite) sequence of 1-dimensional queries, $f_1(\mathbf{I}), f_2(\mathbf{I}), \dots$, where each has global sensitivity 1, and a threshold T. It targets to find the first query whose answer is above T. SVT has been shown to satisfy ε-DP with the following utility guarantee.

Lemma 3.8 ([12, 16]). *If there exists a k such that $f_k(D) \ge T + \frac{6}{\varepsilon}\log(2/\beta)$, then with probability at least $1-\beta$, SVT returns an $\ell \le k$ such that $f_\ell(D) \ge T - \frac{6}{\varepsilon}\log(2k/\beta)$.*

4 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 4.1 ([21, 23]). *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 [21][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 the 1-dimensional mechanism in [12] to d dimensions. Given a truncation threshold $r \ge 0$, the truncated query result $\mathbf{Q}(\mathbf{I}, r)$ is defined as

$$\mathbf{Q}(\mathbf{I}, r) := \sum_{i\in[N]}\left(\min\left(1, \frac{r}{\|\mathbf{S}_i(\mathbf{I})\|}\right)\cdot \mathbf{S}_i(\mathbf{I})\right).$$

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 $r \ge 0$, define

$$\mathrm{Count}(\mathbf{I}, r) = |\{i : \|\mathbf{S}_i(\mathbf{I})\| \le r\}|.$$

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 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 1.

[2][21] states the result under zCDP [8]; here we translate their result to (ε, δ)-DP.
[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.

Algorithm 1: 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 $\widetilde{\mathbf{Q}}(\mathbf{I}) = \mathbf{Q}(\mathbf{I}, \tilde{r}) + \tilde{r}\cdot\sigma(\frac{9\varepsilon}{10}, \delta)\cdot \mathrm{Y}, \mathrm{Y} \sim \mathcal{N}(0, \mathbf{I}_{d\times d})$;
4 return $\widetilde{\mathbf{Q}}(\mathbf{I})$;
```

Theorem 4.2. *Algorithm 1 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 3.8, with probability at least $1-\frac{\beta}{2}$, we have (1) $\mathrm{Count}(\mathbf{I}, \tilde{r}) \ge N - O\left(\frac{1}{\varepsilon}\cdot\log\frac{\log(\mathrm{DS}_{\mathbf{Q}})}{\beta}\right)$; and (2) $\tilde{r} \le \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$. 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 $O\left(\frac{\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})}{\varepsilon}\cdot\sqrt{d\log(1/\delta)\log(1/\beta)}\right)$. □

5 MULTIPLE QUERIES WITH SELF-JOINS

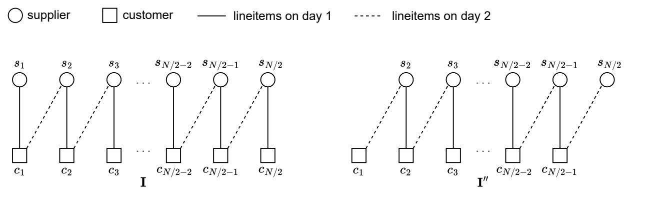
5.1 Why Self-joins are Hard

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 [10], 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, [10] replaces this "hard" truncation with a "soft" truncation [26] 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, [10] 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.



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 $c_{N/2-2}$ $c_{N/2-1}$ $c_{N/2}$
I
s_2 s_3 $s_{N/2-2}$ $s_{N/2-1}$ $s_{N/2}$
c_1 c_2 c_3 $c_{N/2-2}$ $c_{N/2-1}$
I''

Figure 2: An example showing $\mathbf{Q}(\mathbf{I}, r)$ has large sensitivity.

Example 5.1. Consider the query in Example 3.2 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 2, where solid lines denote the lineitems on day 1 and 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 $\|\mathbf{Q}(\mathbf{I}, r)\|$. □

5.2 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

$$E(\mathbf{I}, r) := \max_{\mathbf{I}'' \subseteq \mathbf{I}, \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \le r} |\mathbf{I}''(R_P)|,$$

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 $E(\mathbf{I}, r) \ge \mathrm{Count}(\mathbf{I}, r)$, 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 $J = \mathsf{Edge}(\mathsf{A}, \mathsf{B}) \bowtie \mathsf{Node}(\mathsf{A}) \bowtie \mathsf{Node}(\mathsf{B})$ 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 5.3, 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 5.2. *For any r and any $\mathbf{I}$, $F(\mathbf{I}, r) \le 0$. If $r \ge \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$, then $F(\mathbf{I}, r) = 0$.*

More importantly, we show that $F(\mathbf{I}, r)$ has sensitivity 1:

Lemma 5.3. *For any r and any $\mathbf{I} \sim \mathbf{I}'$, $\mathbf{I}' \subseteq \mathbf{I}$, we have*

$$F(\mathbf{I}', r) - 1 \le F(\mathbf{I}, r) \le F(\mathbf{I}', r).$$

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 $r \ge 0$, $E(\mathbf{I}, r) \ge E(\mathbf{I}', r)$ since any $\mathbf{I}'' \subseteq \mathbf{I}'$ also has $\mathbf{I}'' \subseteq \mathbf{I}$.

On the other hand, given any $r \ge 0$, let

$$\mathbf{I}^* = \operatorname*{arg\,max}_{\mathbf{I}'' \subseteq \mathbf{I}, \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \le r} |\mathbf{I}''|.$$

Then, we can construct a $\mathbf{I}^{*\prime}$ from $\mathbf{I}^*$ by deleting all tuples referencing $t_N(\mathbf{I})$. Then, $\mathbf{I}^{*\prime} \subseteq \mathbf{I}'$, $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}^{*\prime}) \le \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}^*) \le r$ and $|\mathbf{I}^{*\prime}(R_P)| = |\mathbf{I}^*(R_P)| - 1$. And this further means, $E(\mathbf{I}', r) \ge E(\mathbf{I}, r) - 1$.

Finally, combining $N = N' + 1$, the lemma follows. □

Because $F(\mathbf{I}, r)$ has sensitivity 1, we can feed $F(\mathbf{I}, 1), F(\mathbf{I}, 2), F(\mathbf{I}, 4), \ldots$ 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 $r \ge 0$, define

$$\mathbf{Q}(\mathbf{I}, r) := \mathbf{Q}(\mathbf{I}^*(r)), \text{ where } \mathbf{I}^*(r) = \operatorname*{arg\,max}_{\mathbf{I}'' \subseteq \mathbf{I}, \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \le r} \left|\mathbf{I}''(R_P)\right|. \quad (9)$$

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 5.4. Following the Example 5.1 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 $r \ge 2$, $E(\mathbf{I}, r) = N$, $\mathbf{I}^*(r) = \mathbf{I}$, 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 $r \ge 2$, $E(\mathbf{I}'', r) = N - 2$, $\mathbf{I}^{*''}(r) = \mathbf{I}''$, 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 5.5. *Given $r \ge 0$, for any $\mathbf{I} \sim \mathbf{I}'$,*

$$\left\|\mathbf{Q}(\mathbf{I}, r) - \mathbf{Q}(\mathbf{I}', r)\right\| \le (-2F(\mathbf{I}, r) + 2) \cdot r.$$

Proof. Here, we first consider the case $\mathbf{I}(R_P) = \mathbf{I}'(R_P) \cup \{t_N(\mathbf{I})\}$. For convenience, let

$$\mathbf{I}^* = \operatorname*{arg\,max}_{\mathbf{I}'' \subseteq \mathbf{I}, \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \le r} \|\mathbf{I}''(R_P)\|,$$

$$\mathbf{I}^{*'} = \operatorname*{arg\,max}_{\mathbf{I}'' \subseteq \mathbf{I}', \mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \le r} \|\mathbf{I}''(R_P)\|.$$

And define

$$\mathbf{I}^*_{\cap} = \mathbf{I}^* \cap \mathbf{I}^{*'}.$$

Algorithm 2: Exponential-time mechanism for multiple queries with self-joins.

```
  Input: $\varepsilon, \delta, \beta$
1 $\tilde{i} \leftarrow \mathrm{SVT}\left(-\frac{30}{\varepsilon}\log(6/\beta), \frac{\varepsilon}{5}, F(\mathbf{I}, 2^0), F(\mathbf{I}, 2^1), \dots\right)$;
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 $\widetilde{\mathbf{Q}}(\mathbf{I})$;
```

First, by definition,

$$\begin{cases} |\mathbf{I}(R_P) \setminus \mathbf{I}'(R_P)| \le 1 \\ |\mathbf{I}'(R_P) \setminus \mathbf{I}^{*'}(R_P)| \le -F(\mathbf{I}', r) \end{cases} \Rightarrow |\mathbf{I}(R_P) \setminus \mathbf{I}^{*'}(R_P)| \le -F(\mathbf{I}', r) + 1.$$

Further recall $\mathbf{I}^* \subseteq \mathbf{I}$, we have

$$|\mathbf{I}^*(R_P) \setminus \mathbf{I}^{*'}(R_P)| \le -F(\mathbf{I}', r) + 1,$$

which means

$$|\mathbf{I}^*(R_P) \setminus \mathbf{I}^*_{\cap}(R_P)| \le -F(\mathbf{I}', r) + 1.$$

Recall $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}^*) \le r$,

$$\|\mathbf{Q}(\mathbf{I}^*) - \mathbf{Q}(\mathbf{I}^*_{\cap})\| \le \left(-F(\mathbf{I}', r) + 1\right) \cdot r. \tag{10}$$

Symmetrically, we have

$$\|\mathbf{Q}(\mathbf{I}^{*'}) - \mathbf{Q}(\mathbf{I}^*_{\cap})\| \le -F(\mathbf{I}, r) \cdot r. \tag{11}$$

Combining Lemma 5.3, (10), (11), 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,

$$\|\mathbf{Q}(\mathbf{I}^*) - \mathbf{Q}(\mathbf{I}^*_{\cap})\| \le -F(\mathbf{I}', r) \cdot r,$$
$$\|\mathbf{Q}(\mathbf{I}^{*'}) - \mathbf{Q}(\mathbf{I}^*_{\cap})\| \le (-F(\mathbf{I}, r) + 1) \cdot r.$$

With a similar argument, the claim also follows. □

Note that Lemma 5.5 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 5.1. 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 2.

Theorem 5.6. *For any $\varepsilon, \delta, \beta > 0$, and any $\mathbf{I}$, Algorithm 2 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 5.3, 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 $1 - \frac{\delta}{2e^{\varepsilon/5}}$, $\widehat{T} \ge T$. By Lemma 3.7, $\widetilde{Q}(\mathbf{I})$ preserves (ε, δ)-DP.

Below we prove the utility bound. First, by Lemma 3.8, with probability at least $1 - \frac{\beta}{3}$,

$$-F(\mathbf{I}, \tilde{r}) = O\left(\frac{1}{\varepsilon}\log\frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta}\right), \tag{12}$$

and

$$\tilde{r} \le 2 \cdot \mathrm{DS}(\mathbf{I}). \tag{13}$$

Second, by (12) and the definitions of $\mathrm{DS}(\mathbf{I})$ and $\mathbf{Q}(\mathbf{I}, \tilde{r})$,

$$\|\mathbf{Q}(\mathbf{I}) - \mathbf{Q}(\mathbf{I}, \tilde{r})\| = O\left(\frac{\mathrm{DS}(\mathbf{I})}{\varepsilon}\log\frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta}\right). \tag{14}$$

Third, by (12) and the tail bound of Laplace distribution, with probability at least $1 - \frac{\beta}{3}$,

$$\widehat{T} = O\left(\frac{1}{\varepsilon}\log\frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta} + \frac{1}{\varepsilon}\log(e^{\varepsilon}/\delta)\right). \tag{15}$$

Then, combine (13), (15) and the tail bound of Gaussian distribution, we have with probability at least $1 - \frac{\beta}{3}$,

$$\left\|\widetilde{\mathbf{Q}}(\mathbf{I}) - \mathbf{Q}(\mathbf{I}^*)\right\| = O\left(\frac{\sqrt{d\log(1/\beta)\log(e^{\varepsilon}/\delta)}}{\varepsilon^2} \cdot \mathrm{DS}(\mathbf{I}) \cdot \left(\log\frac{\log(\mathrm{DS})}{\beta} + \log(e^{\varepsilon}/\delta)\right)\right). \tag{16}$$

Finally, the theorem follows by combining (14) and (16). □

5.3 A Polynomial-time Algorithm

In this section, we show how to reduce the running time of Algorithm 2 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:

$$z_{k,j} \ge \sum_{i \in D_{k,j}(\mathbf{I})} y_i - |D_{k,j}(\mathbf{I})| + 1, k \in [d], j \in [M_k].$$

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 $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \le r$. 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 (8) into the constraint $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}'') \le r$ turns it into

$$\sum_{k \in [d]} \left(\sum_{j \in C_{k,i}(\mathbf{I})} \left(\psi_{k,j}(\mathbf{I}) \cdot z_{k,j}\right)\right)^2 \le r^2, i \in [N].$$

Therefore, $E(\mathbf{I}, r)$ is the optimal solution of the following integer program:

$$\begin{aligned}
\max \quad & \sum_{i \in [N]} y_i, \\
\text{s.t.} \quad & z_{k,j} \geq \sum_{i \in D_{k,j}(\mathbf{I})} y_i - |D_{k,j}(\mathbf{I})| + 1, && k \in [d], j \in [M_k] \\
& \sum_{k \in [d]} \left(\sum_{j \in C_{k,i}(\mathbf{I})} \left(\psi_{k,j}(\mathbf{I}) \cdot z_{k,j} \right) \right)^2 \leq r^2, && i \in [N], \\
& y_i \in \{0, 1\} && i \in [N], \\
& z_{k,j} \in \{0, 1\} && k \in [d], j \in [M_k].
\end{aligned}$$

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 5.2–5.5.

Letting $\widehat{F}(\mathbf{I}, r) = \widehat{E}(\mathbf{I}, r) - |\mathbf{I}(R_P)|$. The first property is trivial:

Lemma 5.7. *For any r and any $\mathbf{I}$, $\widehat{F}(\mathbf{I}, r) \leq 0$. If $r \geq \mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$, then $\widehat{F}(\mathbf{I}, r) = 0$.*

Now we prove that $\widehat{F}(\mathbf{I}, r)$ also has sensitivity 1.

Lemma 5.8. *For any r and any $\mathbf{I} \sim \mathbf{I}', \mathbf{I}' \subseteq \mathbf{I}$, we have*

$$\widehat{F}(\mathbf{I}', r) - 1 \leq \widehat{F}(\mathbf{I}, r) \leq \widehat{F}(\mathbf{I}', r).$$

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 $k \in [d]$, $C_{k,i}(\mathbf{I}) = \{M_k - |C_{k,i}(\mathbf{I})| + 1, \ldots, M_k - 1, M_k\}$.

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 $\sum_{i \in [N-1]]} y_i^* = E(\mathbf{I}, r) - y_N^* \geq E(\mathbf{I}, r) - 1$, which implies that $\widehat{E}(\mathbf{I}', r) \geq \widehat{E}(\mathbf{I}, r) - 1$.

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 5.5. 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.,

$$z_{k,j}^* = \max\left(0, \sum_{i \in D_{k,j}(\mathbf{I})} y_i^* - |D_{k,j}(\mathbf{I})| + 1\right). \tag{17}$$

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.,

$$\begin{aligned}
&\widehat{\mathbf{Q}}(\mathbf{I}, r) = \mathbf{Q}(\mathbf{I}^*) = \left(Q_1(\mathbf{I}^*), Q_2(\mathbf{I}^*), \ldots, Q_d(\mathbf{I}^*)\right), \\
&\text{where } Q_k(\mathbf{I}^*) = \sum_{j \in [M_k]} \left(z_{k,j}^* \cdot \psi_{k,j}(\mathbf{I})\right), k \in [d].
\end{aligned} \tag{18}$$

Example 5.9. Following the Example 5.4, 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 5.10. *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 5.5. And similarly as the proof of Lemma 5.8, 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 $\widehat{E}(\mathbf{I}, r)$, (17), and (18). 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

$$N - \sum_{i \in [N]} y_i^* = -\widehat{F}(\mathbf{I}, r). \tag{19}$$

And similarly, we have

$$N - \sum_{i \in [N]} y_i^{*'} = -\widehat{F}(\mathbf{I}', r) + 1 \leq -\widehat{F}(\mathbf{I}, r) + 2, \tag{20}$$

where the inequality is by Lemma 5.8.

By this setting, we have

$$\begin{aligned}
&\left\| \mathbf{Q}(\mathbf{I}^*) - \mathbf{Q}(\mathbf{I}^{*'}) \right\| \\
=&\sqrt{\sum_{k \in [d]} \left(\sum_{j \in [M_k]} \left(z_{k,j}^{*'} - z_{k,j}^*\right) \cdot \psi_{k,j}(\mathbf{I}) \right)^2} \\
\leq&\sqrt{\sum_{k \in [d]} \left(\sum_{j \in [M_k]} \left(z_{k,j}^{*'} - z_{k,j}^*\right) \cdot \psi_{k,j}(\mathbf{I}) \cdot \mathbb{I}\left(z_{k,j}^* < z_{k,j}^{*'}\right) \right)^2}
\end{aligned}$$

$$+\sqrt{\sum_{k\in[d]}\left(\sum_{j\in[M_k]}\left(z^*_{k,j}-z^{*'}_{k,j}\right)\cdot\psi_{k,j}(\mathbf{I})\cdot\mathbb{I}\left(z^*_{k,j}>z^{*'}_{k,j}\right)\right)^2}. \quad (21)$$

For any $k\in[d]$, $j\in[M_k]$ such that $z^*_{k,j}<z^{*'}_{k,j}$, we have

$$\begin{aligned}z^{*'}_{k,j}-z^*_{k,j} &\le z^{*'}_{k,j}-z^*_{k,j}\cdot z^{*'}_{k,j}\\ &=\left(1-z^*_{k,j}\right)\cdot z^{*'}_{k,j}\\ &\le\sum_{i\in D_{k,j}(\mathbf{I})}\left(1-y^*_i\right)\cdot z^{*'}_{k,j}. \end{aligned}\quad (22)$$

The first line is by $z^{*'}_{k,j}\le 1$. The last line is by (17).

Then, we have

$$\begin{aligned}&\sqrt{\sum_{k\in[d]}\left(\sum_{j\in[M_k]}\left(z^{*'}_{k,j}-z^*_{k,j}\right)\cdot\psi_{k,j}(\mathbf{I})\cdot\mathbb{I}\left(z^*_{k,j}<z^{*'}_{k,j}\right)\right)^2}\\ \le&\sqrt{\sum_{k\in[d]}\left(\sum_{j\in[M_k]}\sum_{i\in D_{k,j}(\mathbf{I})}\left(1-y^*_i\right)\cdot z^{*'}_{k,j}\cdot\psi_{k,j}(\mathbf{I})\cdot\mathbb{I}\left(z^*_{k,j}<z^{*'}_{k,j}\right)\right)^2}\\ \le&\sqrt{\sum_{k\in[d]}\left(\sum_{j\in[M_k]}\sum_{i\in D_{k,j}(\mathbf{I})}\left(1-y^*_i\right)\cdot z^{*'}_{k,j}\cdot\psi_{k,j}(\mathbf{I})\right)^2}\\ \le&\sum_{i\in[N]}\left(\left(1-y^*_i\right)\cdot\sqrt{\sum_{k\in[d]}\left(\sum_{j\in C_{k,i}(\mathbf{I})}z^{*'}_{k,j}\cdot\psi_{k,j}(\mathbf{I})\right)^2}\right)\\ \le&\sum_{i\in[N]}\left(1-y^*_i\right)\cdot r\\ \le&-\widehat{F}(\mathbf{I},r)\cdot r\end{aligned}\quad (23)$$

The second line is by (22). The fourth line is by the triangle inequality under ℓ_2 distance metric. The last line is by (19).

Similarly, with (20), we can show,

$$\begin{aligned}&\sqrt{\sum_{k\in[d]}\left(\sum_{j\in[M_k]}\left(z^*_{k,j}-z^{*'}_{k,j}\right)\cdot\psi_{k,j}(\mathbf{I})\cdot\mathbb{I}\left(z^*_{k,j}>z^{*'}_{k,j}\right)\right)^2}\\ &\le\left(-\widehat{F}(\mathbf{I},r)+2\right)\cdot r\end{aligned}\quad (24)$$

Finally, combining (21), (23), and (24), the lemma follows. □

Our polynomial-time algorithm is thus the same as Algorithm 2, 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 5.11. *For any $\varepsilon,\delta,\beta>0$ and any $\mathbf{I}$, the polynomial-time version of Algorithm 2 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 5.6, 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 3.8 and 5.2, with probability at least $1-\frac{\beta}{3}$,

$$-\widehat{F}(\mathbf{I},\tilde{r})=O\left(\frac{1}{\varepsilon}\log\frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta}\right), \quad (25)$$

and $\tilde{r}\le 2\cdot\mathrm{DS}(\mathbf{I})$.

By the difference between ExpPrivMultiSJA and PolyPrivMultiSJA, we only need to bound the bias

$$\left\|\mathbf{Q}(\mathbf{I})-\widehat{\mathbf{Q}}(\mathbf{I},\tilde{r})\right\|=O\left(\frac{\mathrm{DS}(\mathbf{I})}{\varepsilon}\log\frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta}\right). \quad (26)$$

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 (17) and (18). By (25),

$$N-\sum_i y^*_i=O\left(\frac{1}{\varepsilon}\log\frac{\log(\mathrm{DS}(\mathbf{I}))}{\beta}\right). \quad (27)$$

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, $\{y^{*(0)}_i\}_i=\{y^*_i\}_i$, $\{z^{*(0)}_{k,j}\}_{k,j}=\{z^*_{k,j}\}_{k,j}$, and

$$\mathbf{Q}(\mathbf{I}^{*(0)})=\mathbf{Q}(\mathbf{I}^*). \quad (28)$$

At iteration $i\in[N]$, we increment $y^{*(i)}_i$ from y^*_i to 1. Then, we set $\{z^{*(i)}_{k,j}\}_{k,j}$ and $\mathbf{Q}(\mathbf{I}^{*(i)})$ as (17) and (18). With setting, we have

$$\begin{aligned}&\|\mathbf{Q}(\mathbf{I}^{*(i)})-\mathbf{Q}(\mathbf{I}^{*(i-1)})\|\\ =&\sqrt{\sum_{k=1}^{d}\left(\sum_{j=1}^{M_k}\left(z^{*(i)}_{k,j}-z^{*(i-1)}_{k,j}\right)\cdot\psi_{k,j}(\mathbf{I})\right)^2}\\ =&\sqrt{\sum_{k=1}^{d}\left(\sum_{j\in C_{k,i}(\mathbf{I})}\left(z^{*(i)}_{k,j}-z^{*(i-1)}_{k,j}\right)\cdot\psi_{k,j}(\mathbf{I})\right)^2}\\ \le&\sqrt{\sum_{k=1}^{d}\left(\sum_{j\in C_{k,i}(\mathbf{I})}\left(1-y^*_i\right)\cdot\psi_{k,j}(\mathbf{I})\right)^2}\\ =&\left(1-y^*_i\right)\cdot\sqrt{\sum_{k=1}^{d}\left(\sum_{j\in C_{k,i}(\mathbf{I})}\psi_{k,j}(\mathbf{I})\right)^2}\\ \le&\left(1-y^*_i\right)\cdot\mathrm{DS}_{\mathbf{Q}}(\mathbf{I}).\end{aligned}\quad (29)$$

The third line is because, at ith iteration, for any $k\in[d]$, $z^{*(i)}_{k,j}$ will not be updated if $j\notin C_{k,i}(\mathbf{I})$. The fourth line is by (17). The last line is by the definition of $\mathrm{DS}_{\mathbf{Q}}(\mathbf{I})$.

After all iterations, we have all $y^{*(N)}_i=1$ thus

$$\mathbf{Q}(\mathbf{I}^{*(N)})=\mathbf{Q}(\mathbf{I}). \quad (30)$$

Summing up (29) for all i and combining (28) and (30), we get (26). Finally, with a similar process as the proof of Theorem 5.6, the claim follows. □

6 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 6.1. 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 $\widehat{\mathrm{Q}}(\mathbf{I}, \tilde{r})$. Finally, we add Gaussian noise to $\widehat{\mathrm{Q}}(\mathbf{I}, \tilde{r})$.

Algorithm 3: 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 5.6, the SVT is likely to stop around $r = 2^k$. Our idea is then to first generate v_ℓ for all $\ell = 1, \dots, k$ 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 3. Note that Algorithm 3 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 $f_1(\mathbf{I}), f_2(\mathbf{I}), \dots$, 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 3 in a more generic form.

7 EXPERIMENTS

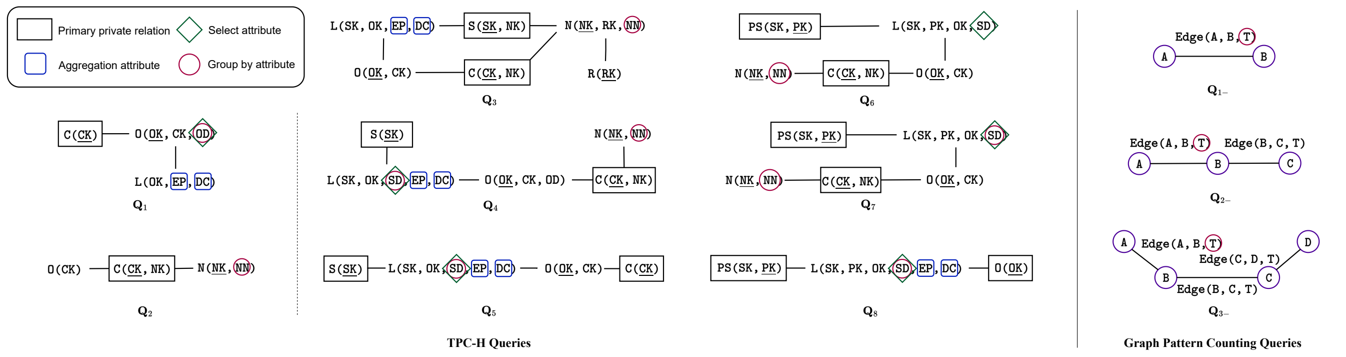
In this section, we report our experimental results comparing our algorithm (denoted PMSJA) with state-of-the-art algorithms for answering SJA queries with group-by over both benchmark and real-world datasets. For self-join-free queries, we compare with OptMean [21]; for queries with (implicit) self-joins, we compare with R2T [10] combined with advanced composition [18].

7.1 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 1. We use datasets with scale 0.125, 0.25, . . . , 8. The default scale is 2, where there are about 15 million tuples.

The Stack Overflow network dataset is from SNAP [28] 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 **stackoverflow** − **a2q** and **stackoverflow** − **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 3. Some of the queries are taken directly from TPC-H benchmark while others are designed



Primary private relation
Select attribute
Aggregation attribute
Group by attribute
C(CK) — O(OK, CK, OD)
L(OK, EP, DC)
Q1
O(CK) — C(CK, NK) — N(NK, NN)
Q2
L(SK, OK, EP, DC) — S(SK, NK) — N(NK, RK, NN)
O(OK, CK) — C(CK, NK)
R(RK)
Q3
S(SK)
N(NK, NN)
L(SK, OK, SD, EP, DC) — O(OK, CK, OD) — C(CK, NK)
Q4
S(SK) — L(SK, OK, SD, EP, DC) — O(OK, CK) — C(CK)
Q5
TPC-H Queries
PS(SK, PK) — L(SK, PK, OK, SD)
N(NK, NN) — C(CK, NK) — O(OK, CK)
Q6
PS(SK, PK) — L(SK, PK, OK, SD)
N(NK, NN) — C(CK, NK) — O(OK, CK)
Q7
PS(SK, PK) — L(SK, PK, OK, SD, EP, DC) — O(OK)
Q8
Edge(A, B, T)
A
B
Q1−
Edge(A, B, T) Edge(B, C, T)
A
B
C
Q2−
A
Edge(A, B, T)
D
Edge(C, D, T)
B
C
Edge(B, C, T)
Q3−
Graph Pattern Counting Queries

Figure 3: The structure of queries.

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: {PartSupp, Customer} and {PartSupp, 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 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 [10, 11, 13, 22, 40], 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 counting queries, we set a degree upper bound of $D = 1,000,000$ and set $GS_\mathbf{Q}$ as the maximum number of graph patterns containing any node, i.e., $GS_{\mathbf{Q}_{1-}} = D$, $GS_{\mathbf{Q}_{2-}} = D^2$, and $GS_{\mathbf{Q}_{3-}} = D^3$.

7.2 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 1 and Table 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, 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 $\sqrt{d}$

Query type		Self-join-free queries		Self-join queries					
Query		Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	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
PMSJA	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 1: Comparison between PMSJA and state-of-the-art algorithms (OptMean [21] for self-join-free queries and R2T [10] 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	Q_{1-}		Q_{2-}		Q_{3-}		Q_{1-}		Q_{2-}		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 2: Comparison between PMSJA and R2T [10] on graph pattern counting queries with $d = 10$ on different networks. We use $\varepsilon = 4$ and report relative error.

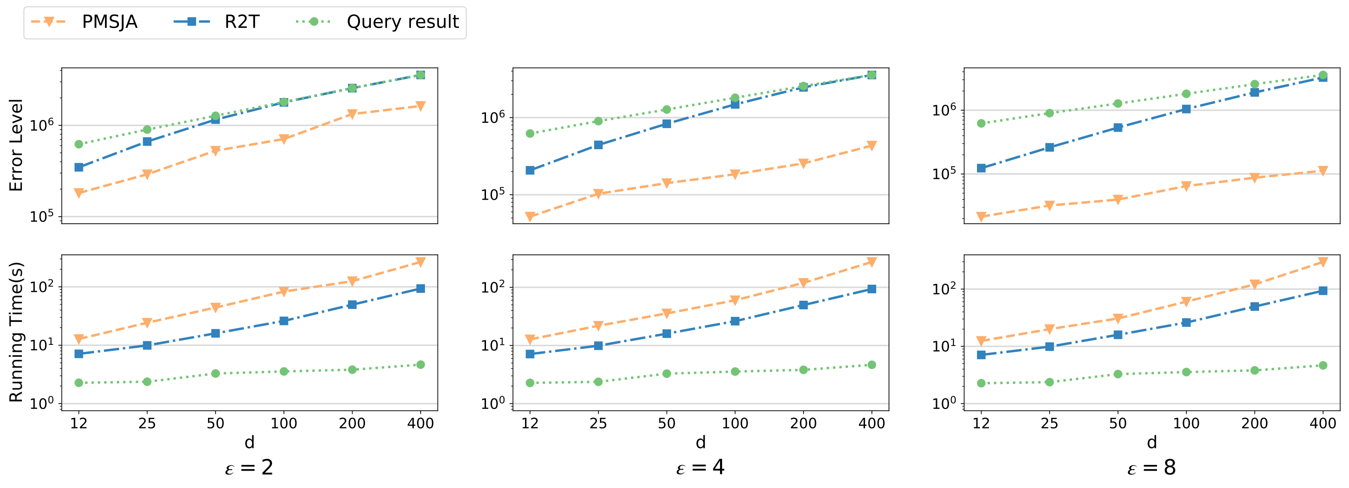
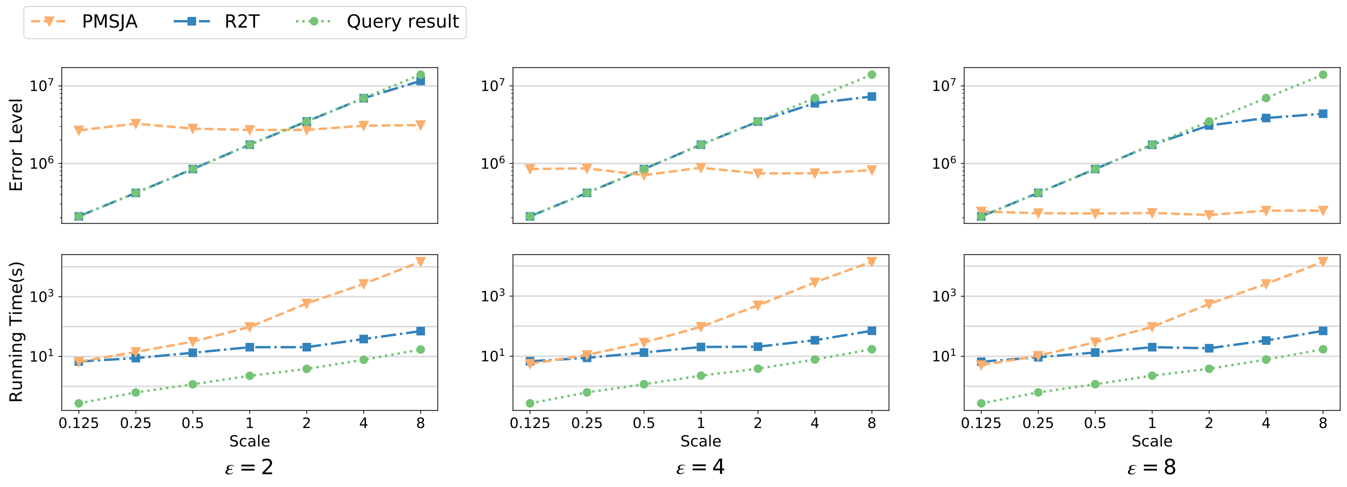


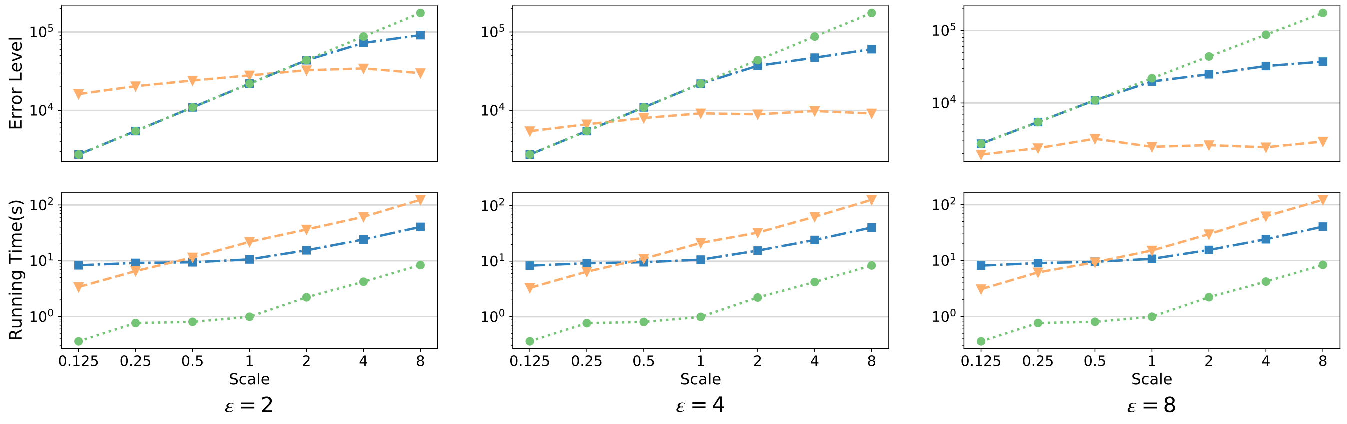
Figure 4: Running times and error levels of PMSJA and R2T for Q_3 with different d and ε.

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×), 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 Q_3 but more/fewer dates so we can have different d's. More precisely, we run it with $d = 12, 25, 50, \dots, 400$ 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 4, 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 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.



(a) Q_3



(b) Q_7

Figure 5: Running times and error levels of PMSJA and R2T with different queries, data scales and ε.

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 5a and 5b.

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.

ACKNOWLEDGMENTS

This work has been supported by HKRGC under grants 16201819, 16205420, and 16205422. We would also like to thank the anonymous reviewers who have made valuable suggestions on improving the presentation of the paper.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Kareem Amin, Alex Kulesza, Andres Munoz, and Sergei Vassilvtiskii. 2019. Bounding user contributions: A bias-variance trade-off in differential privacy. In *International Conference on Machine Learning*. PMLR, 263–271.

[3] Myrto Arapinis, Diego Figueira, and Marco Gaboardi. 2016. Sensitivity of Counting Queries. In *International Colloquium on Automata, Languages, and Programming (ICALP)*.

[4] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. 2007. 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*. 273–282.

[5] Sourav Biswas, Yihe Dong, Gautam Kamath, and Jonathan Ullman. 2020. CoinPress: Practical Private Mean and Covariance Estimation. *Advances in Neural Information Processing Systems* 33 (2020).

[6] Jaroslaw Błasiok, Mark Bun, Aleksandar Nikolov, and Thomas Steinke. 2019. Towards instance-optimal private query release. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2480–2497.

[7] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2013. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. 87–96.

[8] Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*. Springer, 635–658.

[9] Shixi Chen and Shuigeng Zhou. 2013. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 653–664.

[10] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. 2022. R2T: Instance-optimal Truncation for Differentially PrivateQuery Evaluation with Foreign Keys. In *Proc. ACM SIGMOD International Conference on Management of Data*.

[11] Wei Dong and Ke Yi. 2021. Residual Sensitivity for Deferentially Private Multi-Way Joins. In *Proc. ACM SIGMOD International Conference on Management of Data*.

[12] Wei Dong and Ke Yi. 2021. Universal Private Estimators. *arXiv preprint arXiv:2111.02598* (2021).

[13] Wei Dong and Ke Yi. 2022. A Nearly Instance-optimal Differentially Private Mechanism for Conjunctive Queries. In *Proc. ACM Symposium on Principles of Database Systems*.

[14] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[15] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. 2010. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*. 715–724.

[16] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. 2009. 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*. 381–390.

[17] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[18] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. 2010. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 51–60.

[19] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems*. 2339–2347.

[20] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the Accuracy of Differentially Private Histograms Through Consistency. *Proceedings of the VLDB Endowment* 3, 1 (2010).

[21] Ziyue Huang, Yuting Liang, and Ke Yi. 2021. Instance-optimal Mean Estimation Under Differential Privacy. *Advances in Neural Information Processing Systems* (2021).

[22] Noah Johnson, Joseph P Near, and Dawn Song. 2018. Towards practical differential privacy for SQL queries. *Proceedings of the VLDB Endowment* 11, 5 (2018), 526–539.

[23] Gautam Kamath, Jerry Li, Vikrant Singhal, and Jonathan Ullman. 2019. Privately Learning High-Dimensional Distributions. In *Proceedings of the 32nd Annual Conference on Learning Theory (COLT '19)*. 1853–1902.

[24] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2011. Private analysis of graph structure. *Proceedings of the VLDB Endowment* 4, 11 (2011), 1146–1157.

[25] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2014. Private analysis of graph structure. *ACM Transactions on Database Systems (TODS)* 39, 3 (2014), 1–33.

[26] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*. Springer, 457–476.

[27] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2019. PrivateSQL: a differentially private SQL query engine. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1371–1384.

[28] Jure Leskovec and Andrej Krevl. 2016. SNAP datasets: Stanford large network dataset collection (2014). *URL http://snap. stanford. edu/data* (2016), 49.

[29] Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. 2015. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal* 24, 6 (2015), 757–781.

[30] Frank D McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 19–30.

[31] Shanmugavelayutham Muthukrishnan and Aleksandar Nikolov. 2012. Optimal private halfspace counting via discrepancy. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. 1285–1292.

[32] Arjun Narayan and Andreas Haeberlen. 2012. DJoin: Differentially private join queries over distributed databases. In *USENIX Symposium on Operating Systems Design and Implementation*. 149–162.

[33] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. 2013. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. 351–360.

[34] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 75–84.

[35] Catuscia Palamidessi and Marco Stronati. 2012. Differential Privacy for Relational Algebra: Improving the Sensitivity Bounds via Constraint Systems. In *QAPL*.

[36] Davide Proserpio, Sharon Goldberg, and Frank McSherry. 2014. Calibrating Data to Sensitivity in Private Data Analysis. *Proceedings of the VLDB Endowment* 7, 8 (2014).

[37] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1954–1965.

[38] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2014. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1435–1446.

[39] Yuan Qiu, Wei Dong, Ke Yi, Bin Wu, and Feifei Li. 2022. Releasing Private Data for Numerical Queries. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1410–1419.

[40] Yuchao Tao, Xi He, Ashwin Machanavajjhala, and Sudeepa Roy. 2020. Computing Local Sensitivities of Counting Queries with Joins. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 479–494.

[41] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2010. Differential privacy via wavelet transforms. *IEEE Transactions on knowledge and data engineering* 23, 8 (2010), 1200–1214.

[42] Ganzhao Yuan, Yin Yang, Zhenjie Zhang, and Zhifeng Hao. 2016. 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*. 2005–2014.

[43] Ganzhao Yuan, Zhenjie Zhang, Marianne Winslett, Xiaokui Xiao, Yin Yang, and Zhifeng Hao. 2015. Optimizing batch linear queries under exact and approximate differential privacy. *ACM Transactions on Database Systems (TODS)* 40, 2 (2015), 1–47.

[44] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2015. Private release of graph statistics using ladder functions. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 731–745.

[45] Xiaojian Zhang, Rui Chen, Jianliang Xu, Xiaofeng Meng, and Yingtao Xie. 2014. Towards accurate histogram publication under differential privacy. In *Proceedings of the 2014 SIAM international conference on data mining*. SIAM, 587–595.