Differentially Private Data Release over Multiple Tables

Badih Ghazi
Google Research
badihghazi@gmail.com

Xiao Hu*
University of Waterloo
xiaohu@uwaterloo.ca

Ravi Kumar
Google Research
ravi.k53@gmail.com

Pasin Manurangsi
Google Research
pasin@google.com

ABSTRACT

We study synthetic data release for answering multiple linear queries over a set of database tables in a differentially private way. Two special cases have been considered in the literature: how to release a synthetic dataset for answering multiple linear queries over a single table, and how to release the answer for a single counting (join size) query over a set of database tables. Compared to the single-table case, the join operator makes query answering challenging, since the sensitivity (i.e., by how much an individual data record can affect the answer) could be heavily amplified by complex join relationships.

We present an algorithm for the general problem, and prove a lower bound illustrating that our general algorithm achieves parameterized optimality (up to logarithmic factors) on some simple queries (e.g., two-table join queries) in the most commonly-used privacy parameter regimes. For the case of hierarchical joins, we present a data partition procedure that exploits the concept of *uniformized sensitivities* to further improve the utility.

CCS CONCEPTS

• **Information systems** → **Database query processing**; **Query optimization**; • **Security and privacy** → **Data anonymization and sanitization**; • **Theory of computation** → **Theory of database privacy and security**.

KEYWORDS

Differential privacy; Synthetic data; Linear query; Multi-table join

ACM Reference Format:
Badih Ghazi, Xiao Hu, Ravi Kumar, and Pasin Manurangsi. 2023. Differentially Private Data Release over Multiple Tables. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '23), June 18–23, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3584372.3588665

*This work was done while the author was visiting Google Research.

PODS '23, June 18–23, 2023, Seattle, WA, USA

ACM ISBN 979-8-4007-0127-6/23/06…$15.00
https://doi.org/10.1145/3584372.3588665

1 INTRODUCTION

Synthetic data release is a very useful objective in private data analysis. Differential privacy (DP) [17, 18] has emerged as a compelling model that enables us to formally study the tradeoff between utility of released information and the privacy of individuals. In the literature, there has been a large body of work that studies synthetic data release over a single table, e.g., the private multiplicative weight algorithm [25], the histogram-based algorithm [43], the matrix mechanism [33], the Bayesian network algorithm [45], and other works on geometric range queries [7, 9, 11, 19, 20, 28, 34, 35] and datacubes [13].

Data analysis over multiple private tables connected via join operators has been the subject of significant interest within the area of modern database systems. In particular, the challenging question of releasing the join size over a set of private tables has been studied in several recent works including the sensitivity-based framework [15, 16, 30], the truncation-based mechanism [14, 32, 42], as well as in works on one-to-one joins [37, 41], and on graph databases [6, 10]. In practice, multiple queries (as opposed to a single one) are typically issued for data analysis, for example, a large class of linear queries on top of join results with different weights on input tuples, as a generalization of the counting join size query. One might consider answering each query independently but the utility would be very low due to the limited privacy budget, implied by DP composition rules [17, 18]. Hence the question that we tackle in this paper is: how can we release synthetic data for accurately answering a large class of linear queries over multiple tables privately?

1.1 Problem Definition

Multi-way Join and Linear Queries. A (natural) join query can be represented as a hypergraph $\mathcal{H} = (\mathbf{x}, \{\mathbf{x}_1, \ldots, \mathbf{x}_m\})$ [1], where $\mathbf{x}$ is the set of vertices or attributes and each $\mathbf{x}_i \subseteq \mathbf{x}$ is a hyperedge or relation. Let $\mathrm{dom}(x)$ be the domain of attribute x. Define $\mathbb{D} = \mathrm{dom}(\mathbf{x}) = \prod_{x \in \mathbf{x}} \mathrm{dom}(x)$, and $\mathbb{D}_i = \mathrm{dom}(\mathbf{x}_i) = \prod_{x \in \mathbf{x}_i} \mathrm{dom}(x)$ for any $i \in [m]$. For attribute(s) $\mathbf{y}$, we use $\pi_{\mathbf{y}} t$ to denote the value(s) that tuple t displays on attribute(s) $\mathbf{y}$.

Given an instance $\mathbf{I} = \left(R_1^{\mathbf{I}}, \ldots, R_m^{\mathbf{I}}\right)$ of $\mathcal{H}$, each table $R_i^{\mathbf{I}} : \mathbb{D}_i \to \mathbb{Z}^{\geq 0}$ is defined as a function from domain $\mathbb{D}_i$ to a non-negative integer, indicating the frequency of each tuple in $\mathbb{D}_i$. This is more general than the setting where each tuple appears at most once in each relation, since it can capture annotated relations [24] with non-negative annotations. The *input size* of $\mathbf{I}$ is defined as the sum of frequencies of all data records, i.e., $n = \sum_{i \in [m]} \sum_{t \in \mathbb{D}_i} R_i^{\mathbf{I}}(t)$.

We encode the natural join as a function $\rho : \times_{i\in[m]}\mathbb{D}_i \to \{0,1\}$, such that for each combination $\vec{t} = (t_1, \ldots, t_m) \in \times_{i\in[m]}\mathbb{D}_i$ of tuples, $\rho(\vec{t}) = 1$ if and only if $\pi_{\mathbf{x}_i\cap\mathbf{x}_j} t_i = \pi_{\mathbf{x}_i\cap\mathbf{x}_j} t_j$ for each pair of $i, j \in [m]$, i.e., t_i, t_j share the same values on the common attributes between $\mathbf{x}_i$ and $\mathbf{x}_j$. Given a join query $\mathcal{H}$ and an instance $\mathbf{I}$, the join result is also represented as a function $\mathsf{Join}^{\mathbf{I}} : \mathbb{D} \to \mathbb{Z}^{\geq 0}$, such that for any combination $\vec{t} = (t_1, \ldots, t_m) \in \times_{i\in[m]}\mathbb{D}_i$,

$$\mathsf{Join}^{\mathbf{I}}\left(\vec{t}\right) = \rho(\vec{t}) \prod_{i\in[m]} R_i^{\mathbf{I}}(t_i).$$

The *join size* of instance $\mathbf{I}$ over join query $\mathcal{H}$ is therefore defined as

$$\mathrm{count}(\mathbf{I}) = \sum_{\vec{t}\in\times_{i\in[m]}\mathbb{D}_i} \mathsf{Join}^{\mathbf{I}}\left(\vec{t}\right).$$

For each $i \in [m]$, we have a family $\mathcal{Q}_i$ of linear queries defined over $\mathbb{D}_i$ such that for $q \in \mathcal{Q}_i$, $q : \mathbb{D}_i \to [-1, +1]$. Let $\mathcal{Q} = \times_{i\in[m]}\mathcal{Q}_i$. For each linear query $q = (q_1, \ldots, q_m) \in \mathcal{Q}$, the result over instance $\mathbf{I}$ is defined as

$$q(\mathbf{I}) = \sum_{\vec{t}=(t_1,\ldots,t_m)\in\times_{i\in[m]}\mathbb{D}_i} \rho\left(\vec{t}\right) \prod_{i\in[m]} q_i(t_i)\cdot R_i^{\mathbf{I}}(t_i).$$

Our goal is to release a function $\mathbb{F} : \times_{i\in[m]}\mathbb{D}_i \to \mathbb{N}$ such that all queries in $\mathcal{Q}$ can be answered over $\mathbb{F}$ as accurately as possible, i.e., minimizing the ℓ_∞-error $\alpha = \max_{q\in\mathcal{Q}} |q(\mathbf{I}) - q(\mathbb{F})|$, where

$$q(\mathbb{F}) = \sum_{\vec{t}=(t_1,\ldots,t_m)\in\times_{i\in[m]}\mathbb{D}_i} \mathbb{F}(\vec{t}) \prod_{i\in[m]} q_i(t_i).$$

We study the data complexity [44] of this problem and assume that the size of join query is a constant. When the context is clear, we just drop the superscript $\mathbf{I}$ from $R^{\mathbf{I}}$, $\mathsf{Join}^{\mathbf{I}}$.

DP Setting. Two DP settings have been studied in the relational model, depending on whether foreign key constraints are considered or not. The one considering foreign key constraints assumes the existence of a primary private table, and deleting a tuple in the primary private relation will delete all other tuples referencing it; see [14, 32, 42]. In this work, we adopt the other notion, which does not consider foreign key constrains, but defines instances to be neighboring if one can be converted into the other by adding/removing a single tuple; this is the same as the notion studied in some previous works [30, 31, 38, 41]. [1]

Definition 1.1 (Neighboring Instances). A pair $\mathbf{I} = (R_1, \ldots, R_m)$ and $\mathbf{I}' = (R'_1, \ldots, R'_m)$ of instances are *neighboring* if there exists some $i \in [m]$ and $t^* \in \mathbb{D}_i$ such that:

- for any $j \in [m] \setminus \{i\}$, $R_j(t) = R'_j(t)$ for every $t \in \mathbb{D}_j$;
- $R_i(t) = R'_i(t)$ for every $t \in \mathbb{D}_i \setminus \{t^*\}$ and $|R_i(t^*) - R'_i(t^*)| = 1$.

Definition 1.2 (Differential Privacy [17, 18]). For $\epsilon, \delta > 0$, an algorithm $\mathcal{A}$ is (ϵ, δ)-*differentially private* (denoted by (ϵ, δ)-DP) if for any pair $\mathbf{I}, \mathbf{I}'$ of neighboring instances and any subset $\mathcal{S}$ of outputs, $\Pr(\mathcal{A}(\mathbf{I}) \in \mathcal{S}) \leq e^\epsilon \cdot \Pr(\mathcal{A}(\mathbf{I}') \in \mathcal{S}) + \delta$.

[1] Our definition corresponds to the *add/remove* variant of DP where a record is added/removed from a single database R_i. Another possible definition is based on *substitution* DP. Add/remove DP implies substitution DP with only a factor of two increase in the privacy parameters; therefore, all of our results also apply to substitution DP.

Notation. For simplicity of presentation, we henceforth assume throughout that $0 < \epsilon \leq O(1)$ and $0 \leq \delta \leq 1/2$. Furthermore, all lower bounds below hold against an algorithm that achieves the stated error α *with probability* $1 - \beta$ *for some sufficiently small constant* $\beta > 0$; we will omit mentioning this probabilistic part for brevity. For notational ease, we define the following quantities:

$$f^{\mathrm{lower}}(\mathbb{D}, \mathcal{Q}, \epsilon) = \sqrt{\frac{1}{\epsilon}\cdot\sqrt{\log|\mathbb{D}|}}, \text{ and}$$
$$f^{\mathrm{upper}}(\mathbb{D}, \mathcal{Q}, \epsilon, \delta) = f^{\mathrm{lower}}(\mathbb{D}, \mathcal{Q}, \epsilon)\cdot\sqrt{\log|\mathcal{Q}|\cdot\log 1/\delta}.$$

When $\mathbb{D}, \mathcal{Q}, \epsilon, \delta$ are clear from context, we will omit them. Let $\lambda = \frac{1}{\epsilon}\log\frac{1}{\delta}$, which is a commonly-used parameter in this paper.

1.2 Prior Work

We first review the problem of releasing synthetic data for a single table, and mention two important results in the literature. In the single table case, nearly tight upper and lower bounds are known:

Theorem 1.3 ([25]). *For a single table R of at most n records, a family $\mathcal{Q}$ of linear queries, and $\epsilon, \delta > 0$, there exists an algorithm that is (ϵ, δ)-DP, and with probability at least $1 - 1/\mathrm{poly}(|\mathcal{Q}|)$ produces $\mathbb{F}$ such that all queries in $\mathcal{Q}$ can be answered to within error $\alpha = O\left(\sqrt{n}\cdot f^{\mathrm{upper}}\right)$ using $\mathbb{F}$.*

Theorem 1.4 ([8]). *For every sufficiently small $\epsilon > 0$, sufficiently large n, $n_D \geq n^{O(1)}$ and $n_Q \geq (n\cdot\log n_D)^{O(1)}$, there exists a family $\mathcal{Q}$ of queries of size n_Q on a domain $\mathbb{D}$ of size n_D such that any $(\epsilon, 1/n^{\omega(1)})$-DP algorithm that takes as input a database of size at most n, and outputs an approximate answer to each query in $\mathcal{Q}$ to within error α must satisfy $\alpha \geq \widetilde{\Omega}\left(\min\left\{n, \sqrt{n}\cdot f^{\mathrm{lower}}\right\}\right)$.*

Another related problem that has been widely studied by the database community is how to release the join size of a query privately, i.e., $\mathrm{count}(\mathbf{I})$ for every input instance $\mathbf{I}$. Note that $\mathrm{count}(\mathbf{I})$ is essentially a special linear query $q = (q_1, \ldots, q_m)$ with $q_i : \mathbb{D}_i \to \{+1\}$ for every $i \in [m]$. A popular approach of answering the counting join size query is based on the sensitivity framework, which first computes $\mathrm{count}(\mathbf{I})$, then computes the sensitivity of the query (measuring the difference between the query answers on neighboring database instances), and releases a noise-masked query answer, where the noise is drawn from some zero-mean distribution calibrated appropriately according to the sensitivity. It is known that the local sensitivity of counting join size query, defined as

$$\mathrm{LS}_{\mathrm{count}}(\mathbf{I}) = \max_{\mathbf{I}':(\mathbf{I},\mathbf{I}') \text{ are neighboring}} |\mathrm{count}(\mathbf{I}) - \mathrm{count}(\mathbf{I}')|,$$

where the maximum is over all neighboring instances $\mathbf{I}'$ of $\mathbf{I}$, cannot be directly used for calibrating noise, as $\mathrm{LS}_{\mathrm{count}}(\mathbf{I}), \mathrm{LS}_{\mathrm{count}}(\mathbf{I}')$ for a pair $\mathbf{I}, \mathbf{I}'$ of neighboring databases can be used to distinguish them. Global sensitivity (e.g., [18]) defined as

$$\mathrm{GS}_{\mathrm{count}}(\mathbf{I}) = \max_{\mathbf{I}} \mathrm{LS}_{\mathrm{count}}(\mathbf{I}),$$

where the maximum is over all instances $\mathbf{I}$, could be used but the utility is unsatisfactory in many instances. Smooth upper bounds on local sensitivity [40] have instead been considered; they offer much better utility than global sensitivity. Examples include *smooth sensitivity*, which is the smallest smooth upper bound but usually cannot be efficiently computed, and *residual sensitivity* [15], which

is a constant-approximation of smooth sensitivity but can be efficiently computed and used in practice. See Section 3 for formal details.

We note that the sensitivity-based framework can be adapted to answering any single linear query, as long as the sensitivity of the specific linear query is correctly computed. However, we are not interested in answering each linear query individually, since the privacy budget (implied by DP composition) would blow up with the number of queries to be answered, which would be too costly when the query space is extremely large. Instead, we aim to release a synthetic dataset such that any arbitrary linear query can be freely answered over it while preserving DP. In building our data release algorithm, we also resort to the existing sensitivities derived for the counting join size query but in a more careful way.

In the remaining of this paper, when we mention "sensitivity" without a specific function, it should be assumed that the function is counting join size of an input instance, i.e., count($\mathbf{I}$).

1.3 Our Results

We first present the basic *join-as-one* approach for releasing a synthetic dataset for multi-table queries, which computes the join results as a single table and uses existing algorithms for releasing synthetic dataset for a single table. The error will be a function of the join size and the residual sensitivity of count($\cdot$).

Theorem 1.5. *For any join query $\mathcal{H}$, an instance $\mathbf{I}$, a family $\mathcal{Q}$ of linear queries, and $\epsilon > 0$, $\delta > 0$, there exists an (ϵ, δ)-DP algorithm that with probability at least $1 - 1/poly(|\mathcal{Q}|)$ produces $\mathbb{F}$ that can be used to answer all queries in $\mathcal{Q}$ within error:*

$$\alpha = O\left(\left(\sqrt{\text{count}(\mathbf{I}) \cdot \text{RS}^{\beta}_{\text{count}}(\mathbf{I})} + \text{RS}^{\beta}_{\text{count}}(\mathbf{I}) \cdot \sqrt{\lambda}\right) \cdot f^{\text{upper}}\right),$$

where count($\mathbf{I}$) *is the join size of $\mathcal{H}$ over $\mathbf{I}$ and* $\text{RS}^{\beta}_{\text{count}}(\mathbf{I})$ *is the β-residual sensitivity of $\mathbf{I}$ for $\beta = 1/\lambda$.*

We next present the parameterized optimality with respect to the join size and the local sensitivity of count($\cdot$).

Theorem 1.6. *Given arbitrary parameters* $\mathsf{OUT} \geq \Delta > 0$ *such that* OUT/Δ *is sufficiently large and a join query $\mathcal{H}$, for every sufficiently small $\epsilon > 0$, $n_D \geq \mathsf{OUT}^{O(1)}$ and $n_Q \geq (\mathsf{OUT} \cdot \log n_D)^{O(1)}$, there exists a family $\mathcal{Q}$ of queries of size n_Q on domain $\mathbb{D}$ of size n_D such that any $(\epsilon, 1/n^{\omega(1)})$-DP algorithm that takes as input a multi-table database over $\mathcal{H}$ of input size at most n, join size* OUT *and local sensitivity Δ, and outputs an approximate answer to each query in $\mathcal{Q}$ to within error α must satisfy*

$$\alpha \geq \widetilde{\Omega}\left(\min\left\{\mathsf{OUT}, \sqrt{\mathsf{OUT} \cdot \Delta} \cdot f^{\text{lower}}\right\}\right).$$

For the typical setting with $\epsilon = \Omega(1), \delta = O\left(\frac{1}{n^c}\right)$ for some constant c, $|\mathcal{Q}| \leq n^{O(1)}$ and $\lambda = O(\log n)$.

The gap between the upper bound in Theorem 1.5 and the lower bound in Theorem 1.6—ignoring poly-logarithmic factors—boils down to the gap between smooth sensitivity and local sensitivity used in answering count($\cdot$) in a private way, which appears in answering such a single query in a different form.

From the upper bound in terms of residual sensitivity, we exploit the idea of *uniformized sensitivity*[2] to further improve Theorem 1.5, by using a more fine-grained partition of input instances. For the class of hierarchical queries, which has been widely studied in the context of various database problems [4, 12, 21, 27], we obtain a more fine-grained parameterized upper bound in Theorem C.2 and lower bound in Theorem C.3. In the main text, we include Theorem 4.4 and Theorem 4.5 for the basic two-table join to illustrate the high-level idea. We defer all details to Section 4.

2 PRELIMINARIES

We introduce some commonly-used concepts, terminologies, and primitives used in this paper.

Notation. For random variables X, Y and scalars $\epsilon > 0, \delta \in [0, 1)$, we use $X \approx_{(\epsilon,\delta)} Y$ for short if $\Pr[X \in S] \leq e^{\epsilon} \cdot \Pr[Y \in S] + \delta$ and $\Pr[Y \in S] \leq e^{\epsilon} \cdot \Pr[X \in S] + \delta$ for all sets S of outcomes. For convenience, we sometimes write the distribution in place of a random variable drawn from that distribution. E.g., $a + \mathcal{D}$ is a shorthand for $a + X$, where X is a random variable with distribution $\mathcal{D}$ (denoted $X \sim \mathcal{D}$).

(Truncated) Laplace Distribution. The *Laplace distribution* with parameter b, denoted Lap_b, has probability density function (PDF) $\mathsf{Lap}_b(x) \propto e^{-|x|/b}$. The *shifted and truncated Laplace distribution* with parameters b and τ, denoted TLap^{τ}_b, is the distribution supported on $[0, 2\tau]$ whose PDF over the support satisfies $\mathsf{TLap}^{\tau}_b(x) \propto e^{-|x-\tau|/b}$. The DP guarantees of the Laplace and truncated Laplace distributions are well-known (e.g., [18, 22, 23]): for any pair u, v with $|u - v| \leq \Delta$, $u + \mathsf{Lap}_{\Delta/\epsilon} \approx_{(\epsilon,0)} v + \mathsf{Lap}_{\Delta/\epsilon}$ and $u + \mathsf{TLap}^{\tau}_{\Delta/\epsilon} \approx_{(\epsilon,\delta)} v + \mathsf{TLap}^{\tau}_{\Delta/\epsilon}$ for $\tau = \tau(\epsilon, \delta, \Delta) = \frac{\Delta}{\epsilon} \ln\left(1 + \frac{e^{\epsilon}-1}{\delta}\right)$. Note here that $\tau(\epsilon, \delta, \Delta) \leq O(\Delta \cdot \lambda)$ when ϵ is a constant.

Exponential Mechanism. The exponential mechanism (EM) [36] selects a "good" candidate from a set C of candidates. The goodness is defined by a *scoring function* $s(\mathbf{I}, c)$ for an input dataset $\mathbf{I}$ and a candidate $c \in C$ and is assumed to have sensitivity at most one. Then, the EM algorithm consists of sampling each candidate $c \in C$ with probability $\propto \exp(-0.5\epsilon \cdot s(\mathbf{I}, c))$; this algorithm is $(\epsilon, 0)$-DP.

3 JOIN-AS-ONE ALGORITHM

In this section, we present our *join-as-one* algorithm for general join queries, which computes the join results as a single function and then invokes the single-table *private multiplicative weights* (PMW) algorithm [25] (see Algorithm 2). While this is apparently simple, there are many challenging issues in putting everything together. (All missing proofs are in Appendix B.)

3.1 Algorithms for Two-Table Query

We start with the simplest two-table query. Assume the join query $\mathcal{H}$ is given by the hypergraph with vertices $\mathbf{x} = \{A, B, C\}$ and hyperedges $\mathbf{x}_1 = \{A, B\}$, $\mathbf{x}_2 = \{B, C\}$. For $i \in \{1, 2\}$, we define the *degree* of a join value $b \in \text{dom}(B)$ in R_i (i.e., the frequencies of data records displaying join value b in R_i) to be $\deg_{i,B}(b) = \sum_{t \in \mathbb{D}_i : \pi_B t = b} R_i(t)$.

[2] A similar idea of *uniformizing degrees* of join values, i.e., how many tuples a join value appears in a relation, has been used in query processing [29], but we use it in a more complicated scenario to achieve uniform sensitivity.

The *maximum degree* of any join value in $\mathrm{dom}(B)$ is denoted as $\Delta = \max_{b \in \mathrm{dom}(B)} \max\{\deg_{1,B}(b), \deg_{1,B}(b)\}$; note that $\Delta = \mathrm{LS}_{\mathrm{count}}(\mathbf{I})$.

A Natural (but Flawed) Idea. Consider the following algorithm:

- Compute the join result $J = \mathsf{Join}^{\mathbf{I}}$ and release a synthetic dataset $\tilde{J}_1$ for J by invoking the single-table PMW algorithm;

We now briefly explain why this algorithm violates DP. Let $\tilde{J}, \tilde{J}'$ be the synthetic dataset released by this algorithm for two neighboring databases $\mathbf{I}, \mathbf{I}'$ respectively. By the property of the single-table PMW algorithm, the total number of tuples over the whole domain stays unchanged, i.e., $\mathrm{count}(\mathbf{I}) = \sum_{\vec{t} \in \mathbb{D}_1 \times \mathbb{D}_2} \tilde{J}(\vec{t})$ and $\mathrm{count}(\mathbf{I}') = \sum_{\vec{t} \in \mathbb{D}_1 \times \mathbb{D}_2} \tilde{J}'(\vec{t})$. However, $\mathbf{I}, \mathbf{I}'$ could have dramatically different join sizes: Figure 1 shows two neighboring databases with join sizes n and 0 respectively. Thus, $\tilde{J}, \tilde{J}'$ can be used by an adversary to distinguish $\mathbf{I}, \mathbf{I}'$, which makes the algorithm not DP.

Another Natural (but Still Flawed) Idea. To remedy the leakage in the above algorithm, we seek to protect the total number of tuples in the released dataset, say, by padding with dummy tuples:

(1) Compute the join result $J = \mathsf{Join}^{\mathbf{I}}$ and release a synthetic dataset $\tilde{J}_1$ for J by invoking the single-table PMW algorithm;
(2) $\widetilde{\Delta} \leftarrow \Delta + \mathsf{TLap}^{\tau(\epsilon/2, \delta/2, 1)}_{2/\epsilon}$;
(3) $\tilde{J}_2 \leftarrow$ random sample of η records from $\mathbb{D}_1 \times \mathbb{D}_2$, where $\eta \sim \mathsf{TLap}^{\tau(\epsilon/2, \delta/2, \widetilde{\Delta})}_{2\widetilde{\Delta}/\epsilon}$;
(4) Release the union of these two datasets: $\mathbb{F} = \tilde{J}_1 \cup \tilde{J}_2$;

As noted, the local sensitivity cannot be directly used for privately releasing the join size, however, the global sensitivity of $\mathrm{LS}_{\mathrm{count}}(\cdot)$ is 1. Hence, the second step is to obtain an upper bound on the local sensitivity, which will be used to calibrate the noise needed to mask the join size of $\mathbf{I}$. More specifically, for any neighboring databases $\mathbf{I}, \mathbf{I}'$, let $\mathbb{F}, \mathbb{F}'$ be the synthetic datasets released for $\mathbf{I}, \mathbf{I}'$ respectively. As mentioned, $\sum_{t \in \mathbb{D}_1 \times \mathbb{D}_2} \mathbb{F}(t) \approx_{(\epsilon, \delta)} \sum_{t \in \mathbb{D}_1 \times \mathbb{D}_2} \mathbb{F}'(t)$, which resolves the information leakage of the join size. However, it turns out that this approach can still lead to the distinguishability of neighboring databases $\mathbf{I}, \mathbf{I}'$. Indeed, consider the following example.

Example 3.1. Consider neighboring instances $\mathbf{I}, \mathbf{I}'$ in Figure 1. Assume $\epsilon = \Theta(1)$ and $\delta = O(1/n^c)$ for some constant c. Let $\tilde{J}_1, \tilde{J}'_1$ be the synthetic datasets released for J, J' respectively. As $\tilde{J}(\vec{t}) = 0$ for each $\vec{t} \in \mathbb{D}_1 \times \mathbb{D}_2$, $\tilde{J}'(\vec{t}) = 0$ also holds for every $\vec{t} \in \mathbb{D}_1 \times \mathbb{D}_2$. Let $\mathbb{D}' = (\mathrm{dom}(A) \times \{b_1\}) \times (\{b_1, c_1\})$. In contrast, with probability at least $1 - 1/\mathrm{poly}(|\mathcal{Q}|)$, there exists some constant c_1 such that $\sum_{\vec{t} \in \mathbb{D}'} \tilde{J}_1(\vec{t}) \geq n - c_1 \cdot \sqrt{n} \cdot f^{\mathrm{upper}}$, for answering $\mathrm{count}(\cdot)$ within an error of $O(\sqrt{n} \cdot f^{\mathrm{upper}})$. Moreover, $\Delta = n$, hence $\widetilde{\Delta} = O(n \log n)$ and $\eta = O(n \log^2 n)$. The probability that no tuple in $\mathbb{D}'$ is sampled by $\tilde{J}'_2$ is $(1 - n/n^3)^{c_2 \cdot n \log^2 n} = (1 - 1/n^2)^{n^2 \cdot \frac{c_2 \log n^2}{n}} > 1/e$ for some constant c_2. Together, $\Pr[\sum_{\vec{t} \in \mathbb{D}'} \mathbb{F}(\vec{t}) = 0] < 1/\mathrm{poly}(|\mathcal{Q}|)$ and $\Pr[\sum_{\vec{t} \in \mathbb{D}'} \mathbb{F}'(\vec{t}) = 0] > 1/e$, thus violating the DP condition $1/e < 1/\mathrm{poly}(|\mathcal{Q}|) \cdot e^{\Theta(1)} + O(1/n^c)$.

Our Algorithm. The main insight in the idea that works is to change the order of the two steps in the previous flawed version. More specifically, we first pad dummy tuple to join results and then release the synthetic dataset for the "noisy" join results.

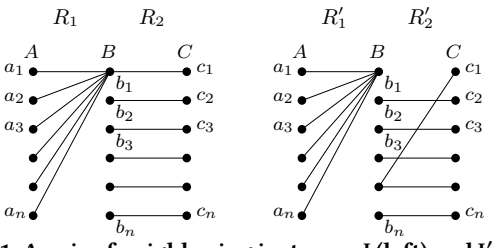

Figure 1: A pair of neighboring instances I (left) and I′ (right) for two-table join with $\mathbf{dom}(A) = \{a_1, \ldots, a_n\}$, $\mathbf{dom}(B) = \{b_1, \ldots, b_n\}$, and $\mathbf{dom}(C) = \{c_1, \ldots, c_n\}$.

Algorithm 1: $\mathrm{TwoTable}(\mathbf{I} = \{R_1, R_2\})$

$\widetilde{\Delta} \leftarrow \Delta + \mathsf{TLap}^{\tau(\epsilon/2, \delta/2, 1)}_{2/\epsilon}$;
return $\mathrm{PMW}_{\epsilon/2, \delta/2, \widetilde{\Delta}}(\mathbf{I})$; ▶ Algorithm 2;

Algorithm 2: $\mathrm{PMW}_{\epsilon, \delta, \widetilde{\Delta}}(\mathbf{I} = \{R_1, \ldots, R_m\})$

$\widehat{n} = \mathrm{count}(\mathbf{I}) + \mathsf{TLap}^{\tau(\epsilon/2, \delta/2, \widetilde{\Delta})}_{2\widetilde{\Delta}/\epsilon}$;
$\mathbb{F}_0 \leftarrow \widehat{n}$ times uniform distribution over $\mathbb{D}$: $\mathbb{F}_0(x) = \frac{\widehat{n}}{|\mathbb{D}|}$;
$\epsilon' \leftarrow \frac{\epsilon}{16\sqrt{k \cdot \log(1/\delta)}}$;
foreach $i = 1, \ldots, k$ **do**
  Sample a query $q_i \in \mathcal{Q}$ using the ϵ'-DP EM with score function $s_i(\mathbf{I}, q) = \frac{1}{\widetilde{\Delta}} \cdot |q(\mathbb{F}_{i-1}) - q(\mathbf{I})|$;
  $m_i \leftarrow q_i(\mathbf{I}) + \mathrm{Lap}_{\widetilde{\Delta}/\epsilon'}$;
  Update for each $x \in \mathbb{D}$:
  $\mathbb{F}_i(x) \propto \mathbb{F}_{i-1}(x) \times \exp\left(q_i(x) \cdot (m_i - q_i(\mathbb{F}_{i-1})) \cdot \frac{1}{2\widehat{n}}\right)$;
return $\mathrm{Avg}^k_{i=1} \mathbb{F}_i$;

(1) $\widetilde{\Delta} \leftarrow \Delta + \mathsf{TLap}^{\tau(\epsilon/2, \delta/2, 1)}_{2/\epsilon}$;
(2) $\tilde{J}_2 \leftarrow$ random sample of η records from $\mathbb{D}_1 \times \mathbb{D}_2$, where $\eta \sim \mathsf{TLap}^{\tau(\epsilon/2, \delta/2, \widetilde{\Delta})}_{2\widetilde{\Delta}/\epsilon}$;
(3) We compute the join result $J = \mathrm{Join}(\mathcal{H}, \mathbf{I})$ and release a synthetic dataset $\mathbb{F}$ for $J \cup \tilde{J}_2$ by invoking the single-table PMW algorithm;

We observe that this approach can be further simplified by releasing a synthetic dataset for the join result J directly by invoking the single-table PMW algorithm, but starting from an initial uniform distribution parameterized by $\hat{n} = \mathrm{count}(\mathbf{I}) + \eta$. Algorithm 1 contains a complete description for releasing synthetic dataset for two-table joins. Since $\widehat{n}$ is private and the single-table algorithm [25] releases a private synthetic dataset for J, Algorithm 1 also satisfies DP, as stated in Lemma 3.2.

Lemma 3.2. *Algorithm 1 is (ϵ, δ)-DP.*

Error Analysis. As shown in Appendix A, with probability $1 - 1/\mathrm{poly}(|\mathcal{Q}|)$, Algorithm 2 returns a synthetic dataset $\mathbb{F}$ such that every linear query in $\mathcal{Q}$ can be answered over $\mathbb{F}$ within error (omitting f^{upper}): $O\left(\sqrt{\mathrm{count}(\mathbf{I}) \cdot (\Delta + \lambda)} + (\Delta + \lambda) \cdot \sqrt{\lambda}\right)$. Then, we arrive at:

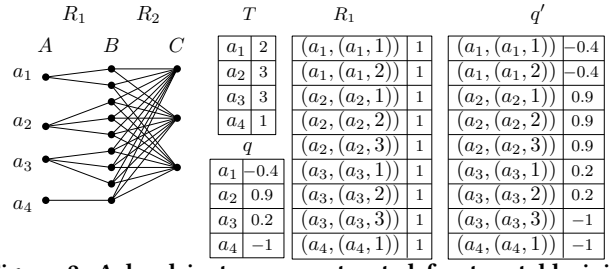

Figure 2: A hard instance constructed for two-table join, with $n = 9$, $\Delta = 3$ and $\mathsf{OUT} = 27$.

Theorem 3.3. *For any two-table instance $\mathbf{I}$, a family $\mathcal{Q}$ of linear queries, and $\epsilon > 0$, $\delta > 0$, there exists an algorithm that is (ϵ, δ)-DP, and with probability at least $1 - 1/poly(|\mathcal{Q}|)$ produces $\mathbb{F}$ such that all linear queries in $\mathcal{Q}$ can be answered to within error:*

$$\alpha = O\left((\sqrt{\text{count}(\mathbf{I}) \cdot (\Delta + \lambda)} + (\Delta + \lambda) \cdot \sqrt{\lambda}) \cdot f^{\text{upper}}\right),$$

where $\text{count}(\mathbf{I})$ *is the join size of* $\mathbf{I}$*, and* $\Delta = LS_{\text{count}}(\mathbf{I})$.

3.2 Lower Bounds for Two-Table Join

The first lower bound is based on the local sensitivity:

Theorem 3.4. *For any $\Delta > 0$, there exists a family $\mathcal{Q}$ of queries such that any (ϵ, δ)-DP algorithm that takes as input a database $\mathbf{I}$ with local sensitivity at most Δ and outputs an approximate answer to each query in $\mathcal{Q}$ to within error α, must satisfy $\alpha \geq \Omega(\Delta)$.*

Our second lower bound is via a reduction to the single-table case: we create a two-table instance where R_1 encodes the single-table and R_2 "amplifies" both the sensitivity and the join size by a factor of Δ. This eventually results in the following lower bound:

Theorem 3.5. *Given arbitrary parameters $\mathsf{OUT} \geq \Delta > 0$ such that OUT/Δ is sufficiently large, for every sufficiently small $\epsilon > 0$, $n_D \geq \mathsf{OUT}^{O(1)}$ and $n_Q \geq (\mathsf{OUT} \cdot \log n_D)^{O(1)}$, there exists a family $\mathcal{Q}$ of queries of size n_Q on domain $\mathbb{D}$ of size n_D such that any $(\epsilon, 1/n^{\omega(1)})$-DP algorithm that takes as input a two-table database over $\mathcal{H}$ of input size at most n, join size OUT and local sensitivity Δ, and outputs an approximate answer to each query in $\mathcal{Q}$ to within error α must satisfy*

$$\alpha \geq \widetilde{\Omega}\left(\min\left\{\mathsf{OUT}, \sqrt{\mathsf{OUT} \cdot \Delta} \cdot f^{\text{lower}}\right\}\right).$$

Proof. Let $n = \frac{\mathsf{OUT}}{\Delta}$. From Theorem 1.4, there exists a set $\mathcal{Q}_{\mathsf{one}}$ of queries on domain $\mathbb{D}$ for which any (ϵ, δ)-DP algorithm that takes as input a single-table database $T \in \mathbb{D}$ and outputs an approximate answer to each query in $\mathcal{Q}_{\mathsf{one}}$ within error α requires that $\alpha \geq \tilde{\Omega}\left(\min\left\{n, \sqrt{n} \cdot f^{\text{lower}}(\mathbb{D}, \mathcal{Q}_{\mathsf{one}}, \epsilon)\right\}\right)$. For an arbitrary single-table database $T : \mathbb{D} \to \mathbb{Z}^+$, we construct a two-table instance $\mathbf{I}$ of join size OUT, and local sensitivity Δ as follows:

- Set $\text{dom}(A) = \mathbb{D}$, $\text{dom}(B) = \mathbb{D} \times [n]$, and $\text{dom}(C) = [\Delta]$.
- Let $R_1(a, (b_1, b_2)) = \mathbf{1}[a = b_1 \wedge b_2 \leq T(a)]$ for all $a \in \text{dom}(A)$ and $(b_1, b_2) \in \text{dom}(B)$.
- Let $R_2(b, c) = 1$ for all $b \in \text{dom}(B)$ and $c \in \text{dom}(C)$.

Algorithm 3: MultiTable($\mathbf{I}$)

$\beta \leftarrow 1/\lambda$;
$\widetilde{\Delta} \leftarrow \text{RS}^{\beta}_{\text{count}}(\mathbf{I}) \cdot e^{\mathsf{TLap}^{\tau(\epsilon/2, \delta/2, \beta)}_{2\beta/\epsilon}}$;
return $\text{PMW}_{\epsilon/2, \delta/2, \widetilde{\Delta}}(\mathbf{I})$;

It can be easily checked that $\mathbf{I}$ has join size at most OUT and local sensitivity Δ, and that two neighboring databases T, T' result in neighboring instances $\mathbf{I}, \mathbf{I}'$. Finally, let $\mathcal{Q}_1$ contain queries from $\mathcal{Q}_{\mathsf{one}}$ applied on its first attribute (i.e., $\mathcal{Q}_1 := \{q \circ \pi_A \mid q \in \mathcal{Q}_{\mathsf{one}}\}$), and let $\mathcal{Q}_2$ contain only a single query $q_{\text{all-one}} : \mathbb{D}_2 \to \{+1\}$. An example is illustrated in Figure 2.

Our lower bound argument is a reduction to the single-table case. Let $\mathbb{I}(\mathsf{OUT}, \Delta)$ be the set of all instances of join size OUT and local sensitivity Δ. Let $\mathcal{A}$ be an algorithm that takes any two-table instance in $\mathbb{I}(\mathsf{OUT}, \Delta)$, and outputs an approximate answer to each query in $\mathcal{Q}$ within error α'. For each query $q \in \mathcal{Q}_{\mathsf{one}}$, let $q' = (q \circ \pi_A, q_{\text{all-one}})$ be its corresponding query in $\mathcal{Q}$. Let $\widetilde{q}'(\mathbf{I})$ be the approximate answer for query $\widetilde{q}'$. We then return $\widetilde{q}(T) = \widetilde{q}'(\mathbf{I})/\Delta$ as an approximate answer to $q(T)$.

We first note that this algorithm for answering queries in $\mathcal{Q}_{\mathsf{one}}$ is (ϵ, δ)-DP due to the post-processing property of DP. The error guarantee follows immediately from the observation that $q'(\mathbf{I}) = \Delta \cdot q(T)$. Therefore, from Theorem 1.4, we can conclude that

$$\begin{aligned}\alpha' &\geq \Delta \cdot \widetilde{\Omega}(\min\{n, \sqrt{n} \cdot f^{\text{lower}}(\mathbb{D}, \mathcal{Q}_{\mathsf{one}}, \epsilon)\}) \\ &= \widetilde{\Omega}\left(\min\{\mathsf{OUT}, \sqrt{\mathsf{OUT}} \cdot \sqrt{\Delta} \cdot f^{\text{lower}}\}\right).\end{aligned}$$

□

3.3 Multi-Table Join

We next extend the join-as-one approach to multi-table queries. First, notice that Algorithm 1 does not work in the multi-table setting because $\text{LS}_{\text{count}}(\cdot)$ may itself have a large global sensitivity (unlike the two-table case where the global sensitivity of $\text{LS}_{\text{count}}(\cdot)$ is 1.) Hence, we will have to find other ways to perturb the join size, and then use this noisy join size to parameterize the initial uniform distribution of the single-table PMW algorithm. As noted, several previous works have proposed various smooth upper bounds on the local sensitivity [40]: For $\beta > 0$, $S^{\beta}(\cdot)$ is a β-smooth upper bound on local sensitivity, if (1) $S^{\beta}(\mathbf{I}) \geq \text{LS}_{\text{count}}(\mathbf{I})$ for every instance $\mathbf{I}$; and (2) for any pair of neighboring instances $\mathbf{I}, \mathbf{I}'$, $S^{\beta}(\mathbf{I}') \leq e^{\beta} S^{\beta}(\mathbf{I})$.

The smallest smooth upper bound on local sensitivity is denoted as *smooth sensitivity*. However, as observed by [15], it takes $n^{O(\log n)}$ time to compute smooth sensitivity for count$(\cdot)$, which is very expensive especially when the input size n of underlying instance is large. Fortunately, we do not necessarily need smooth sensitivity: any smooth upper bound on local sensitivity can be used for perturbing the join size. In building our data release algorithm, we use *residual sensitivity*, which is a constant-approximation of smooth sensitivity, and more importantly can be computed in time polynomial in n.

To introduce the definition of residual sensitivity, we need the following terminology. Given a join query $\mathcal{H} = (\mathbf{x}, \{\mathbf{x}_1, \ldots, \mathbf{x}_m\})$ and a subset $E \subseteq [m]$ of relations, its *boundary*, denoted as ∂_E, is the set of attributes that belong to relations both in and out of E, i.e., $\partial E = \{x \mid x \in \mathbf{x}_i \cap \mathbf{x}_j, i \in E, j \notin E\}$. Correspondingly, its maximum

boundary query is defined as[3]:

$$T_E(\mathbf{I}) = \max_{t \in \mathsf{dom}(\partial E)} \sum_{t' \in \mathsf{dom}(\cup_{i \in E} \mathbf{x}_i): \pi_{\partial E} t' = t} \prod_{i \in E} R_i^{\mathbf{I}}(\pi_{\mathbf{x}_i} t'). \quad (1)$$

Definition 3.6 (Residual Sensitivity [15, 16]). Given $\beta > 0$, the *β-residual sensitivity* of count$(\cdot)$ over $\mathbf{I}$ is defined as

$$\mathrm{RS}^{\beta}_{\mathrm{count}}(\mathbf{I}) = \max_{k \geq 0} e^{-\beta k} \cdot \hat{\mathrm{LS}}^{k}_{\mathrm{count}}(\mathbf{I}),$$

where $\hat{\mathrm{LS}}^{k}_{\mathrm{count}}(\mathbf{I}) = \max_{\mathbf{s} \in \mathcal{S}^k} \max_{i \in [m]} \sum_{E \subseteq [m] \setminus \{i\}} T_{[m] \setminus \{i\} \setminus E}(\mathbf{I}) \cdot \prod_{j \in E} \mathbf{s}_j$, for $\mathcal{S}^k = \left\{\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_m) : \sum_{i=1}^m \mathbf{s}_i = k, \mathbf{s}_i \in \mathbb{Z}^{\geq 0}, \forall i \in [m]\right\}$.

It is well-known that $\mathrm{RS}^{\beta}_{\mathrm{count}}(\cdot)$ is a smooth upper bound on $\mathrm{LS}_{\mathrm{count}}(\cdot)$ [15]. However, we do not use it (as in [15]) to calibrate the noise drawn from a Laplace or Cauchy distribution. In our data release problem, we always find an upper bound $\widetilde{\Delta}$ for $\mathrm{RS}^{\beta}_{\mathrm{count}}(\mathbf{I})$, which is then passed to the single-table PMW algorithm. To compute $\widetilde{\Delta}$, we observe that $\ln\left(\mathrm{RS}^{\beta}_{\mathrm{count}}(\cdot)\right)$ has global sensitivity at most β. Therefore, adding to it an appropriately calibrated (truncated and shifted) Laplace noise provides an upper bound that is private. The idea is formalized in Algorithm 3. Its privacy guarantee is immediate:

Lemma 3.7. *Algorithm 3 is (ϵ, δ)-DP.*

Error analysis. By definition of TLap, we have $\left|\mathsf{TLap}^{\tau(\epsilon/2,\delta/2,\beta)}_{2\beta/\epsilon}\right| \leq 2\tau(\epsilon/2, \delta/2, \beta) \leq O\left(\frac{\beta}{\epsilon} \cdot \log(1/\delta)\right) = O(1)$ and therefore $\widetilde{\Delta}$ is a constant-approximation of $\mathrm{RS}^{\beta}_{\mathrm{count}}(\mathbf{I})$, i.e., $\widetilde{\Delta} = \Theta(\mathrm{RS}^{\beta}_{\mathrm{count}}(\mathbf{I}))$. Hence $\widehat{n} = O\left(\mathrm{count}(\mathbf{I}) + \mathrm{RS}^{\beta}_{\mathrm{count}}(\mathbf{I}) \cdot \lambda\right)$. Putting everything together, the total error (omitting f^{upper}) is:

$$O\left(\sqrt{\widehat{n}} \cdot \sqrt{\widetilde{\Delta}}\right) = O\left(\sqrt{\mathrm{count}(\mathbf{I}) \cdot \mathrm{RS}^{\beta}_{\mathrm{count}}(\mathbf{I})} + \mathrm{RS}^{\beta}_{\mathrm{count}}(\mathbf{I}) \cdot \sqrt{\lambda}\right).$$

Extending the previous lower bound argument on the two-table query, we can obtain the lower bound on the multi-table query in Theorem 1.6. Moreover, we give the worst-case analysis on the error achieved above in Appendix B.

4 UNIFORMIZED SENSITIVITY

So far, we have shown a parameterized algorithm for answering linear queries whose utility is in terms of the join size and the residual sensitivity. A natural question arises: can we achieve a more fine-grained parameterized algorithm with better utility?

Let us start with an instance of two-table join (see Figure 3), with input size $\Theta(n)$, join size $\Theta(n\sqrt{n})$, and local sensitivity $\sqrt{n}$. Algorithm 1 achieves an error of $O(n)$. However, this instance is beyond the scope of Theorem 3.5, as the degree distribution over join values is extremely non-uniform. Revisiting the error bound in Theorem 3.3, we can gain an intuition regarding why Algorithm 1 does not perform well on this instance. The costly term is $O(\sqrt{\mathrm{count}(\mathbf{I})} \cdot \sqrt{\Delta})$, where Δ is the largest degree of join values in the input instance $\mathbf{I}$. However, there are many join values whose

[3]The semi-join result of $R_i \ltimes t$ is defined as function $R'_i : \mathbb{D}_i \to \mathbb{Z}^+$ such that $R_t(t') = R(t')$ if $\pi_{\mathbf{x}_i} t' = \pi_{\mathbf{x}_i} t$ and 0 otherwise.

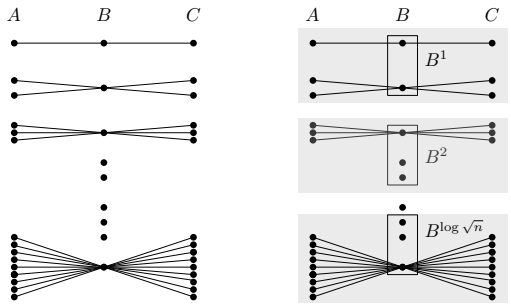

Figure 3: An instance beyond the scope of Theorem 3.5. There are $\sqrt{n}$ join values in attribute B, where there is exactly one join value with degree i in both R_1, R_2 for every $i \in [\sqrt{n}]$.

Algorithm 4: $\textsc{Uniformize}_{\epsilon,\delta}(\mathbf{I})$

```
𝕀 ← Partition_{ε/2,δ/2}(I);
foreach I' ∈ 𝕀 do
    𝔽(I') ← MultiTable_{ε/2,δ/2}(I');        ▶Algorithm 3;
return ⋃_{I'∈𝕀} 𝔽(I');
```

degree is much smaller than Δ. Therefore, a natural idea is to *uniformize sensitivities*, i.e., partition the input instance into a set of sub-instances by join values, where join values with similar sensitivities are in the same sub-instance. We then invoke our previous join-as-one algorithm as a primitive on each sub-instance independently, and return the union of the synthetic datasets generated for all sub-instances. Our uniformization framework is illustrated in Algorithm 4. (All missing proofs are given in Appendix C).

4.1 Uniformized Two-Table Join

As mentioned, there is quite an intuitive way to uniformize a two-table join. As described in Algorithm 5, the high-level idea is to bucket join values in $\mathsf{dom}(B)$ by their maximum degree in R_1 and R_2. To protect the privacy of individual tuples, we draw a random sample from $\mathsf{TLap}^{\tau(\epsilon/2,\delta/2,1)}_{2/\epsilon}$ and add it to a join value's degree, before determining to which bucket it should go. Recall that $\lambda = \frac{1}{\epsilon}\log\frac{1}{\delta}$. Let $\gamma_0 = 0$ and $\gamma_i = 2^i\lambda$ for all $i \in \mathbb{N}$. Conceptually, we divide $[0, n+\lambda]$ into $\ell = \lceil\log(\frac{n}{\lambda}+1)\rceil$ buckets, where the ith bucket is associated with $(\gamma_{i-1}, \gamma_i]$ for $i \in [\ell]$. The set of values from $\mathsf{dom}(B)$ whose maximum noisy degree in R_1 and R_2 falls into bucket i is denoted as B^i. For each i, we identify tuples in R_1, R_2 whose join value falls into B^i as R_1^i, R_2^i, which forms the sub-instance (R_1^i, R_2^i). More specifically, $R_1^i : B^i \to \mathbb{Z}^{\geq 0}$, such that $R_1^i(t) = R_1(t)$ if $\pi_B t \in B^i$ and $R_1^i(t) = R_1(t)$ otherwise. R_2^i is defined similarly. Finally, we return all the sub-instances as the partition.

Lemma 4.1. *Algorithm 4 on two-table join is (ϵ, δ)-DP.*

The key insight in Lemma 4.1 is that adding or removing one input tuple can increase or decrease the degree of one join value $b \in \mathsf{dom}(B)$ by at most one. Hence, Algorithm 5 satisfies (ϵ, δ)-DP by parallel composition [37]. Moreover, since each input tuple participates in exactly one sub-instance, and Algorithm 2 preserves (ϵ, δ)-DP for each sub-instance by Lemma 3.2, Algorithm 4 preserves $(2\epsilon, 2\delta)$-DP by basic composition [18].

Algorithm 5: PARTITION-TWOTABLE$_{\epsilon,\delta}(\mathbf{I} = \{R_1, R_2\})$

1 **foreach** $i \in \mathbb{N}$ **do** $B^i \leftarrow \emptyset$;
2 **foreach** $b \in dom(B)$ **do**
3 $\quad \widetilde{\deg}_B(b) = \max\{\deg_{1,B}(b), \deg_{2,B}(b)\} + \mathsf{TLap}^{\tau(\epsilon,\delta,1)}_{1/\epsilon}$;
4 $\quad i \leftarrow \max\left\{1, \left\lceil \log \frac{1}{\lambda} \cdot \widetilde{\deg}_B(b) \right\rceil\right\}$;
5 $\quad B^i \leftarrow B^i \cup \{b\}$;
6 **foreach** i *with* $B^i \neq \emptyset$ **do**
7 $\quad$ **foreach** $j \in \{1, 2\}$ **do**
8 $\quad\quad R^i_j : \mathrm{dom}(\mathbb{D}_1) \to \mathbb{Z}^{\geq 0}$ such that for any $t \in \mathbb{D}_1$,
$\quad\quad\quad R^i_j(t) = R_j(t)$ if $\pi_B t \in B^i$ and $R^i_j(t) = 0$ otherwise;
9 **return** $\bigcup_{i:B^i\neq\emptyset}\{(R^i_1, R^i_2)\}$;

Error Analysis. Note that n is not explicitly used in Algorithm 5 as this is not public, but it is easy to check that there exists no join value in $\mathrm{dom}(B)$ with degree larger than n under the input size constraint. Hence, it is safe to consider $i \in [\ell]$ in our analysis.

Given an instance $\mathbf{I}$ over the two-table join, let $\pi = \{B^1_\pi, \ldots, B^\ell_\pi\}$ be the partition of $\mathrm{dom}(B)$ generated by Algorithm 5. Let $\mathbf{I}^i_\pi = (R^i_1, R^i_2)$ be the sub-instance induced by B^i_π. Let $\mathbb{F}^i$ be the synthetic dataset generated for $\mathbf{I}^i_\pi$. From Theorem 3.3, with probability $1 - 1/\mathrm{poly}(|\boldsymbol{Q}|)$, the error for answering any linear query defined on $\left(\mathrm{dom}(A) \times \mathrm{dom}(B^i_\pi)\right) \times \left(\mathrm{dom}(B^i_\pi) \times \mathrm{dom}(C)\right)$ with $\mathbb{F}^i$ is (omitting f^{upper}) $\alpha_i = O\left(\sqrt{\mathrm{count}(\mathbf{I}^i_\pi) \cdot 2^i \cdot \lambda} + 2^i \cdot \lambda^{3/2}\right)$. By a union bound, with probability $1 - 1/\mathrm{poly}(|\boldsymbol{Q}|)$, the error for answering any linear query in $\boldsymbol{Q}$ with $\cup_i \mathbb{F}^i$ is (omitting f^{upper}):

$$\alpha \leq \sum_{i\in[\ell]} \alpha_i \leq \lambda^{3/2} \cdot (\Delta + \lambda) + \sqrt{\lambda} \cdot \sum_{i\in[\ell]} \sqrt{\mathrm{count}(\mathbf{I}^i_\pi)} \cdot \sqrt{2^i}, \quad (2)$$

since $\sum_{i\in[\ell]} 2^i \leq \sum_{i\in[\lceil\log(\Delta+\lambda)\rceil]} 2^i = O(\Delta + \lambda)$. Moreover, we observe that Algorithm 4 achieves better (or at least not worse than) error than Algorithm 1 (if ignoring λ), since

$$\begin{aligned}(2) &\leq \lambda^{3/2} \cdot (\Delta + \lambda) + \sqrt{\lambda} \cdot \sqrt{\sum_{i\in[\ell]} \mathrm{count}(\mathbf{I}^i_\pi)} \cdot \sqrt{\sum_{i\in[\ell]} 2^i} \\ &= \lambda^{3/2} \cdot (\Delta + \lambda) + \sqrt{\lambda} \cdot \sqrt{\mathrm{count}(\mathbf{I})} \cdot \sqrt{\Delta + \lambda},\end{aligned}$$

where the first inequality is implied by the Cauchy–Schwarz inequality. Furthermore, we observe that the gap between the error achieved by Algorithm 4 and Algorithm 1 can be polynomially large in terms of the data size; see Example 4.2.

Example 4.2. Consider an instance $\mathbf{I}$ of two-table join, which further amplifies the non-uniformity of instance in Figure 3. For $i \in \{0, 1, \cdots, \frac{2}{3}\log_2 k\}$, there are $k^2/8^i$ distinct join values in $\mathrm{dom}(B)$ with $\deg_{1,B}(b) = \deg_{2,B}(b) = 2^i$. Obviously, $\Delta = k^{2/3}$. It can be easily checked that the input size is $n \leq 2k^2$ and join size is $\mathrm{count}(\mathbf{I}) = \frac{2}{3}k^2 \log_2 k$. For simplicity, we assume $\epsilon = \Theta(1)$ and $\delta = 1/n^c$ for some constant c, therefore $\lambda = \Theta(1)$. Algorithm 1 achieves an error of (omitting f^{upper}) $\alpha = O(\sqrt{\mathrm{count}(\mathbf{I}) \cdot \Delta}) = O(k^{4/3})$. By uniformization, join values with the same degree will be put into the same bucket (even after adding a small noise). In this way, Algorithm 5 achieves an error of (omitting f^{upper}): $\alpha = O(k^{2/3} + \sum_{i\in[\frac{2}{3}\log_2 k]} \sqrt{k^2/8^i \cdot 2^i \cdot 2^i} \cdot \sqrt{2^i}) = O(k \log_2 k)$, improving Algorithm 1 by a factor of $k^{1/3} = O(n^{1/6})$.

Although the partition generated by Algorithm 5 is randomized due to noisy degrees (line 3), it is not far away from a fixed partition based on true degrees. As shown in Appendix C, Algorithm 5 can have its error bounded by what can be achieved through the *uniform partition* below (recall that $\ell = \lceil \log \frac{n}{\lambda} \rceil$ and $\gamma_i = \lambda \cdot 2^i$ for $i \in [\ell]$):

Definition 4.3 (Uniform Partition). For an instance $\mathbf{I} = (R_1, R_2)$, a uniform partition of $\mathrm{dom}(B)$ is $\pi^* = \left\{B^1_{\pi^*}, \ldots, B^\ell_{\pi^*}\right\}$ such that for any $i \in [\ell]$, $b \in B^i_{\pi^*}$ if $\max\left\{\deg_{1,B}(b), \deg_{2,B}(b)\right\} \in (\gamma_{i-1}, \gamma_i]$.

THEOREM 4.4. *For any two-table instance $\mathbf{I}$, a family $\boldsymbol{Q}$ of linear queries, and $\epsilon > 0$, $\delta > 0$, there exists an algorithm that is (ϵ, δ)-DP, and with probability at least $1 - 1/poly(|\boldsymbol{Q}|)$ produces $\mathbb{F}$ such that all linear queries in $\boldsymbol{Q}$ can be answered to within error:*

$$\alpha = O\left((\lambda^{\frac{3}{2}}(\Delta + \lambda) + \sum_{i\in[\ell]} \sqrt{\mathrm{count}\left(\mathbf{I}^i_{\pi^*}\right)} \cdot \sqrt{2^i \cdot \lambda}) \cdot f^{\mathrm{upper}}\right),$$

where $\ell = \lceil \log \frac{n}{\lambda} \rceil$ and $\mathbf{I}^i_{\pi^*}$ is the sub-instance of $\mathbf{I}$ induced by $B^i_{\pi^*}$.

Let $\overrightarrow{\mathsf{OUT}} = \left\langle \mathsf{OUT}^i \in \mathbb{Z}^{\geq 0} : i \in [\ell], \sum_i \mathsf{OUT}^i \in [0, n^2] \right\rangle$ be a join size vector. An instance $\mathbf{I} = (R_1, R_2)$ *conforms* to $\overrightarrow{\mathsf{OUT}}$ if

- for every $i \in [\ell]$, $\mathrm{count}\left(\mathbf{I}^i_{\pi^*}\right) = \Theta\left(\mathsf{OUT}^i\right)$;
- $\mathrm{count}(\mathbf{I}) = \Theta\left(\sum_{i\in[\ell]} \mathsf{OUT}^i\right)$.

Then, we introduce a more fine-grained lower bound parameterized by the join size distribution under the uniform partition, and show that Algorithm 4 is optimal up to poly-logarithmic factors. The proof is given in Appendix C.

THEOREM 4.5. *Given a join size vector $\overrightarrow{\mathsf{OUT}}$, for every sufficiently small $\epsilon > 0$, $n_D \geq (\log \mathsf{OUT})^{O(1)}$ and $n_Q \geq (\mathsf{OUT} \cdot \log n_D)^{O(1)}$, there exists a family $\boldsymbol{Q}$ of queries of size n_Q on a domain $\mathbb{D}$ of size n_D such that any $(\epsilon, o(1/n))$-DP algorithm that takes as input a two-table instance of input size at most n while conforming to $\overrightarrow{\mathsf{OUT}}$, and outputs an approximate answer to each query in $\boldsymbol{Q}$ to within error α must satisfy $\alpha \geq \widetilde{\Omega}\left(\max_{i\in[\ell]} \min\left\{\mathsf{OUT}^i, \sqrt{\mathsf{OUT}^i} \cdot \sqrt{2^i \cdot \lambda} \cdot f^{\mathrm{lower}}\right\}\right)$, for $\ell = \lceil \log \frac{n}{\lambda} \rceil$.*

4.2 Uniformized Hierarchical Join

This uniformization technique, surprisingly, can be further extended beyond two-table queries to the class of *hierarchical queries*. A join query $\mathcal{H} = \{\mathbf{x}, \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}\}$ is *hierarchical*, if for any pair x, y of attributes, either $\mathrm{atom}(x) \subseteq \mathrm{atom}(y)$, or $\mathrm{atom}(y) \subseteq \mathrm{atom}(x)$, or $\mathrm{atom}(x) \cap \mathrm{atom}(y) = \emptyset$, where $\mathrm{atom}(x) = \{i \in [m] : x \in \mathbf{x}_i\}$ is the set of relations containing attribute x. One can always organize the attributes of a hierarchical join into a tree such that every relation corresponds to a root-to-node path (see Figure 4).

We show how to exploit this nice property to improve Algorithm 3 on hierarchical joins with uniformization. Recall that the essence of uniformization is to decompose the instance into a set of sub-instances so that we can upper bound the residual sensitivity as tight as possible, as implied by the error expression in Theorem 1.5.

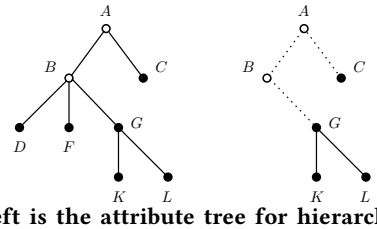

Figure 4: Left is the attribute tree for hierarchical join $\mathcal{H}$ with $\mathbf{x} = \{A, B, C, D, F, G, K, L\}$, $\mathbf{x}_1 = \{A, B, D\}$, $\mathbf{x}_2 = \{A, B, F\}$, $\mathbf{x}_3 = \{A, B, G, K\}$, $\mathbf{x}_4 = \{A, B, G, L\}$, and $\mathbf{x}_5 = \{A, C\}$. Right is the residual query defined on $E = \{3, 4, 5\}$ with $\partial E = \{A, B\}$. Moreover, $\bigwedge_{345} = \{A\}$ and $\bigvee_{345} = \{A, B, C, G, K, L\}$. $\mathcal{H}_{E,\partial E}$ is disconnected with $C_E = \{\{3, 4\}, \{5\}\}$. $T_{345}(\mathbf{I})$ can be upper bounded by $\mathrm{mdeg}_5(A) \cdot \mathrm{mdeg}_{34}(AB) \cdot \mathrm{mdeg}_3(ABG) \cdot \mathrm{mdeg}_4(ABG)$.

In Definition 3.6, the residual sensitivity is built on the join sizes of a set of maximum boundary queries $T_E(\mathbf{I})$'s, but these statistics are too far away from being the partition criteria. Instead, we find an upper bound of residual sensitivity in terms of degrees.

4.2.1 **An Upper Bound on T_E.** In a hierarchical join $\mathcal{H}$, we denote $\mathcal{T}$ as the attribute tree for $\mathcal{H}$. For any $E \subseteq [m]$, the partial attributes ∂E forms a connected subtree of $\mathcal{T}$ including the root (see Figure 4). To derive an upper bound on T_E, we identify a broader class of *q-aggregate queries* by generalizing aggregate attributes from ∂_E to any subset of attributes that form a connected subtree of $\mathcal{T}$ including the root (i.e., sitting on top of $\mathcal{T}$). For simplicity, we define $\bigwedge_E = \cap_{i \in E} \mathbf{x}_i$ and $\bigvee_E = \cup_{i \in E} \mathbf{x}_i$.

Definition 4.6 (q-aggregate Query). For a hierarchical join $\mathcal{H} = (\mathbf{x}, \{\mathbf{x}_1, \ldots, \mathbf{x}_m\})$, an instance $\mathbf{I}$, and a subset $E \subseteq [m]$ of relations, assume that a subset $\mathbf{y} \subseteq \bigvee_E$ of attributes satisfies the following property[4]: for any $x_1, x_2 \in \mathbf{x}$, if $\mathrm{atom}(x_1) \subseteq \mathrm{atom}(x_2)$ and $x_1 \in \mathbf{y}$, then $x_2 \in \mathbf{y}$. A *q-aggregate query* defined over E on $\mathbf{y}$ is defined as

$$T_{E,\mathbf{y}}(\mathbf{I}) = \max_{t \in \mathrm{dom}(\mathbf{y})} \sum_{t' \in \mathrm{dom}(\bigvee_E): \pi_{\mathbf{y}} t' = t} \prod_{i \in E} R_i(\pi_{\mathbf{x}_i} t').$$

It is not hard to see $T_E(\mathbf{I}) = T_{E,\partial E}(\mathbf{I})$ from (1). Below, we focus on an upper bound for $T_{E,\mathbf{y}}(\mathbf{I})$ with general $\mathbf{y}$ characterized by Definition 4.6, which boils down to the product of *maximum degrees*:

Definition 4.7 (Maximum Degree). For a hierarchical join $\mathcal{H} = (\mathbf{x}, \{\mathbf{x}_1, \ldots, \mathbf{x}_m\})$, an instance $\mathbf{I}$, $E \subseteq [m]$ and $\mathbf{y} \subseteq \bigwedge_E$, the degree of tuple $t \in \mathrm{dom}(\mathbf{y})$ is:

$$\deg^{\mathbf{I}}_{E,\mathbf{y}}(t) = \begin{cases} \sum_{t' \in \mathrm{dom}(\mathbf{x}_i): \pi_{\mathbf{y}} t' = t} R_i(t') & \text{if } |E| = 1, \text{say } E = \{i\} \\ \left|\{t' \in \Psi_E(\mathbf{I}) : \pi_{\mathbf{y}} t' = t\}\right| & \text{otherwise,} \end{cases}$$

where $\Psi_E(\mathbf{I}) = \left\{\pi_{\bigwedge_E} t' : t' \in \mathrm{dom}(\bigvee_E), \prod_{i \in E} R_i(\pi_{\mathbf{x}_i} t') > 0\right\}$. The *maximum degree* is defined as $\mathrm{mdeg}^{\mathbf{I}}_E(\mathbf{y}) = \max_{t \in \mathrm{dom}(\mathbf{y})} \deg^{\mathbf{I}}_{E,\mathbf{y}}(t)$.

We next show an upper bound on $T_{E,\mathbf{y}}(\mathbf{I})$ using $\mathrm{mdeg}^{\mathbf{I}}_E(\mathbf{y})$'s. When the context is clear, we drop the superscript $\mathbf{I}$ from $\mathrm{mdeg}^{\mathbf{I}}_E(\mathbf{y})$.

[4]The same property on $\mathbf{y}$ has been characterized by *q-hierarchical queries* [4], where $\mathbf{y}$ serves as the set of output attributes there.

Case (1). When $|E| = 1$, say $E = \{i\}$, we note that $\mathbf{y} \subseteq \mathbf{x}_i$ and $T_{E,\mathbf{y}}(\mathbf{I})$ is essentially equivalent to $\mathrm{mdeg}_i(\mathbf{y})$, since

$$T_{E,\mathbf{y}}(\mathbf{I}) = \max_{t \in \mathrm{dom}(\mathbf{y})} \sum_{t' \in \mathrm{dom}(\mathbf{x}_i): \pi_{\mathbf{y}} t' = t} R_i(t') = \mathrm{mdeg}_E(\mathbf{y}).$$

Case (2). In general, let $\mathcal{H}_{E,\mathbf{y}} = (\bigvee_E - \mathbf{y}, \{\mathbf{x}_i - \mathbf{y} : i \in E\})$ be the residual join defined on relations in E after removing attributes $\mathbf{y}$. We next distinguish two cases based on the connectivity[5] of $\mathcal{H}_{E,\mathbf{y}}$:

Case (2.1): $\mathcal{H}_{E,\mathbf{y}}$ is disconnected. Let C_E be the set of connected subqueries of $\mathcal{H}_{E,\mathbf{y}}$. We can further decompose $T_{E,\mathbf{y}}(\mathbf{I})$ as:

$$\begin{aligned} T_{E,\mathbf{y}}(\mathbf{I}) &= \max_{t \in \mathrm{dom}(\mathbf{y})} \prod_{E' \in C_E} \sum_{t' \in \mathrm{dom}(\bigvee_{E'}): \pi_{\mathbf{y} \cap (\bigvee_{E'})} t' = t} \prod_{i \in E'} R_i(\pi_{\mathbf{x}_i} t') \\ &\le \prod_{E' \in C_E} \max_{t \in \mathrm{dom}(\mathbf{y})} \sum_{t' \in \mathrm{dom}(\bigvee_{E'}): \pi_{\mathbf{y} \cap (\bigvee_{E'})} t' = t} \prod_{i \in E'} R_i(\pi_{\mathbf{x}_i} t') \\ &= \prod_{E' \in C_E} T_{E', \mathbf{y} \cap (\bigvee_{E'})}(\mathbf{I}), \end{aligned}$$

since $\mathbf{y} \cap (\bigvee_{E'}) \subseteq \bigvee_{E'}$ and $\mathbf{y} \cap (\bigvee_{E'})$ satisfies the property in Definition 4.7, if $\mathbf{y}$ satisfies the same property.

Case (2.2): $\mathcal{H}_{E,\mathbf{y}}$ is connected. In this case, we have $\mathbf{y} \subsetneq \bigwedge_E$.

$$\begin{aligned} T_{E,\mathbf{y}}(\mathbf{I}) &\le \mathrm{mdeg}_E(\mathbf{y}) \cdot \max_{t \in \mathrm{dom}(\bigwedge_E)} \sum_{t' \in \mathrm{dom}(\bigvee_E): \pi_{\bigwedge_E} t' = t} \prod_{i \in E} R_i(\pi_{\mathbf{x}_i} t') \\ &= \mathrm{mdeg}_E(\mathbf{y}) \cdot T_{E, \bigwedge_E}(\mathbf{I}), \end{aligned}$$

since $\bigwedge_E \subseteq \bigvee_E$ and $\bigwedge_E$ satisfies the property in Definition 4.7.

Hence, $T_{E,\mathbf{y}}(\mathbf{I})$ is eventually upper bounded by a product chain of maximum degrees (see Figure 4). A careful inspection reveals that the maximum degrees participating in $T_E(\mathbf{I})$ for $\mathbf{y} = \partial E$ are not arbitrary; instead, they display rather special structures captured by Lemma 4.8, which is critical to our partition procedure.

Lemma 4.8. *Every maximum degree* $\mathrm{mdeg}_{E'}(\mathbf{y})$ *participating in the upper bound of* $T_E(\mathbf{I})$ *corresponds to a distinct attribute* $x \in \mathbf{x}$ *such that* $E' = \mathrm{atom}(x)$ *and* $\mathbf{y}$ *corresponds to the ancestors of* x *in* $\mathcal{T}$.

4.2.2 **Partition with Maximum Degrees**. After getting an upper bound on $T_E(\mathbf{I})$, we now define *degree configuration* for hierarchical joins, similarly to the two-table join. Our target is to decompose the input instance into a set of sub-instances (which may not be tuple-disjoint), such that each sub-instance is characterized by one distinct *degree configuration*, and join results of all sub-instances form a partition of the final join result.

Definition 4.9 (Degree Configuration). For a hierarchical join $\mathcal{H} = (\mathbf{x}, \{\mathbf{x}_1, \ldots, \mathbf{x}_m\})$, a *degree configuration* is defined as $\sigma : 2^{[m]} \times 2^{\mathbf{x}} \to \mathbb{Z}^{\ge 0} \cup \{\perp\}$ such that for any $E \subseteq [m]$ and $\mathbf{y} \subseteq \mathbf{x}$, $\sigma(E, \mathbf{y}) \ne \perp$ if and only if there exists an attribute $x \in \mathbf{x}$ such that $E = \mathrm{atom}(x)$ and $\mathbf{y}$ is the set of ancestors of x in the attribute tree $\mathcal{T}$ of $\mathcal{H}$.

Algorithm 6 recursively decomposes the input instance by attributes in a bottom-up way on $\mathcal{T}$, and invokes Algorithm 7 as a primitive to further decompose every sub-instance. Algorithm 7

[5]A multi-way join $\mathcal{H} = (\mathbf{x}, \{\mathbf{x}_i : i \in [m]\})$ can be modeled it as a graph $G_{\mathcal{H}}$, where each $\mathbf{x}_i$ is a vertex and an edge exists between $\mathbf{x}_i, \mathbf{x}_j$ if $\mathbf{x}_i \cap \mathbf{x}_j \ne \emptyset$. $\mathcal{H}$ is *connected* if $G_{\mathcal{H}}$ is connected, and *disconnected* otherwise. For disconnected $\mathcal{H}$, we can decompose it into multiple connected subqueries by finding all connected components for $G_{\mathcal{H}}$ via graph search algorithm, where each component indicates a connected subquery of $\mathcal{H}$.

Algorithm 6: PARTITION-HIERARCHICAL$_{\epsilon,\delta}(\mathcal{H}, \mathbf{I})$

$\mathbb{I} \leftarrow \{\mathbf{I}\}$, $\mathcal{T} \leftarrow$ an attribute tree of $\mathcal{H}$;
while *there exists a non-visited node in* $\mathcal{T}$ **do**
  $\mathbb{I}' \leftarrow \emptyset$;
  $x \leftarrow$ a leaf or any node whose children are all visited;
  foreach $\mathbf{I}' \in \mathbb{I}$ **do**
    $\mathbb{I}' \leftarrow \mathbb{I}' \cup$ DECOMPOSE$_{\epsilon,\delta}(\mathbf{I}', x)$; ▶Algorithm 7 ;
  $\mathbb{I} \leftarrow \mathbb{I}'$, Mark x as visited;
return $\mathbb{I}$;

Algorithm 7: DECOMPOSE$_{\epsilon,\delta}(\mathbf{I}, x)$

$\mathbf{y} \leftarrow \{y \in \mathbf{x} : \text{atom}(x) \subsetneq \text{atom}(y)\}$, $E \leftarrow \text{atom}(x)$;
foreach $i \in \mathbb{N}$ **do** $\mathbf{y}^i \leftarrow \emptyset$;
foreach $t \in dom(\mathbf{y})$ **do**
  $\widetilde{\deg}^{\mathbf{I}}_{E,\mathbf{y}}(t) = \deg^{\mathbf{I}}_{E,\mathbf{y}}(t) + \mathsf{TLap}^{\tau(\epsilon,\delta,1)}_{1/\epsilon}$;
  $i \leftarrow \max\left\{1, \left\lceil \log \frac{1}{\lambda} \cdot \widetilde{\deg}^{\mathbf{I}}_{E,\mathbf{y}}(t) \right\rceil\right\}$;
  $\mathbf{y}^i \leftarrow \mathbf{y}^i \cup \{t\}$;
foreach i *with* $\mathbf{y}^i \neq \emptyset$ **do**
  foreach $j \in E$ **do**
    $R^{\mathbf{I}}_{j,i} : \text{dom}(\mathbb{D}_j) \to \mathbb{Z}$ such that for $t \in \mathbb{D}_j$, $R^{\mathbf{I}}_{j,i}(t) = R^{\mathbf{I}}_j(t)$ if $\pi_{\mathbf{y}} t \in \mathbf{y}^i$ and $R^{\mathbf{I}}_{j,i}(t) = 0$ otherwise;
return $\bigcup_{i:\mathbf{y}^i \neq \emptyset} \left\{\{R^{\mathbf{I}}_{j,i} : j \in E\} \cup \{R^{\mathbf{I}}_j : j \notin E\}\right\}$;

takes as input an attribute x and an instance $\mathbf{I}$ that has been decomposed by all descendants of x, and outputs a set of sub-instances such that their join results form a partition of join result of $\mathbf{I}$, and for every sub-instance $\mathbf{I}'$, $\deg^{\mathbf{I}'}_{\text{atom}(x),\mathbf{y}}(t)$ is roughly the same for every tuple $t \in \text{dom}(\mathbf{y})$, where $\mathbf{y}$ is the set of ancestors of x in $\mathcal{T}$.

LEMMA 4.10. *For input instance* $\mathbf{I}$*, let* $\mathbb{I}$ *be the set of sub-instances returned by Algorithm 7.* $\mathbb{I}$ *satisfies the following properties:*

- *For any* $\vec{t} \in \times_{i\in[m]}\mathbb{D}_i$*, there exists some* $\mathbf{I}' \in \mathbb{I}$ *such that* $\text{Join}^{\mathbf{I}}(\vec{t}) = \text{Join}^{\mathbf{I}'}(\vec{t})$ *and* $\text{Join}^{\mathbf{I}''}(\vec{t}) = 0$ *for any* $\mathbf{I}'' \in \mathbb{I} - \{\mathbf{I}'\}$.
- *Each input tuple appears in* $O(\log^c n)$ *sub-instances of* $\mathbb{I}$*, where* c *is a constant depending on* m*;*
- *Each sub-instance* $\mathbf{I} \in \mathbb{I}$ *corresponds to a distinct degree configuration* σ *such that for any* $E \in [m]$ *and* $\mathbf{y} \subseteq \mathbf{x}$ *with* $\sigma(E, \mathbf{y}) \neq \perp$*:*

$$\widetilde{\deg}^{\mathbf{I}}_{E,\mathbf{y}}(t) \in \left(\lambda \cdot 2^{\sigma(E,\mathbf{y})-1}, \lambda \cdot 2^{\sigma(E,\mathbf{y})}\right]$$

holds for any $t \in dom(\mathbf{y})$.

LEMMA 4.11. *Algorithm 4 is* $(O(\log^c n)\cdot\epsilon, O(\log^c n)\cdot\delta)$*-DP, where* c *is a constant depending on* m.

The logarithmic factor in Lemma 4.11 arises since each input tuple participates in $O(\log^c n)$ sub-instances, implied by Lemma 4.10. We present the error analysis of Algorithm 4 for hierarchical joins and a parameterized lower bound in Appendix C.

5 CONCLUSIONS AND DISCUSSION

In this paper, we proposed algorithms for releasing synthetic data for answering linear queries over multi-table joins. Our work opens up several interesting directions listed below.

Non-Hierarchical Queries. Perhaps the most immediate question is if uniformization can benefit the non-hierarchical case, even for the simplest join $\mathcal{H}$ defined on $\mathbf{x} = \{A, B, C\}$ with $\mathbf{x}_1 = \{A, B\}$, $\mathbf{x}_2 = \{B, C\}$, and $\mathbf{x}_3 = \{C, D\}$. In determining the residual sensitivity in Definition 3.6, we observe that $T_{23}(\mathbf{I}) \leq \text{mdeg}_2(B) \cdot \text{mdeg}_3(C)$, $T_{12}(\mathbf{I}) \leq \text{mdeg}_1(B) \cdot \text{mdeg}_2(C)$, and $T_{13} \leq \text{mdeg}_1(B) \cdot \text{mdeg}_3(C)$. It is easy to uniformize $\text{mdeg}_1(B), \text{mdeg}_3(C)$ by partitioning R_1, R_3 by attributes B, C respectively. However, it is challenging to uniformize $\text{mdeg}_2(B), \text{mdeg}_2(C)$ by partitioning R_2, while keeping the number of sub-instances small. For example, a trivial strategy simply puts every individual tuple $t \in \mathbb{D}_2$ with $R_2(t) > 0$, together with tuples in R_1, R_3 that can be joined with t, as one sub-instance. In this case, it is great to have small $\text{mdeg}_2(B)$ and $\text{mdeg}_2(C)$, but each tuple from R_1 or R_3 may participate in $\text{mdeg}_1(B)$ or $\text{mdeg}_3(C)$ sub-instances, hence the privacy consumption increases linearly when applying the parallel decomposition! Alternatively, one may uniformize $\text{mdeg}_2(B)$ and $\text{mdeg}_2(C)$ independently, say with partitions π_1 of $\text{dom}(B)$ and π_2 of $\text{dom}(C)$. However, the degrees $\deg_{2,B}$ as well as $\deg_{2,C}$ are defined on the whole relation R_2, hence we may still end up with very non-uniform distribution of $\deg_{2,B}, \deg_{2,C}$ when restricting to a sub-instance induced by $B^i_{\pi_1}, C^j_{\pi_2}$ together. We leave this interesting question for future work.

Query-Specific Optimality. Throughout this work, we consider *worst-case* set $\mathcal{Q}$ of queries parameterized by its size. Although this is a reasonable starting point, it is also plausible to hope for an algorithm that is nearly optimal for *all* query sets $\mathcal{Q}$. In the single-table case, this has been achieved in [5, 26, 39]. However, the situation is more complicated in the multi-table setting since we have to take the local sensitivity into account, whereas, in the single-table case, the query family already dictates the possible change resulting from moving to a neighboring dataset. This is also an interesting open question for future research.

Instance-Specific Optimality. We have considered worst-case instances in this work. One might instead prefer to achieve finer-grained instance-optimal errors. For the single-table case, [2] observed that an instance-optimal DP algorithm is not achievable, since a trivial algorithm can return the same answer(s) for all input instances, which could work perfectly on one specific instance but poorly on all remaining instances. To overcome this, a notion of "neighborhood optimality" has been proposed [2, 16], where we consider not only a single instance but also its neighbors at some constant distance. We note, however, that this would not work in our setting when there are a large number of queries. Specifically, if we again consider an algorithm that always returns the true answer for this instance, then its error with respect to the entire neighborhood set is still quite small—at most the distance times the maximum local sensitivity in the set. This is independent of the table size n, whereas our lower bounds show that the dependence on n is inevitable. As such, the question of how to define and prove instance-optimal errors remains open for the multi-query case.

REFERENCES

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley Reading.

[2] Hilal Asi and John C Duchi. 2020. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. *NeurIPS*, 14106–14117.

[3] Albert Atserias, Martin Grohe, and Dániel Marx. 2008. Size bounds and query plans for relational joins. In *FOCS*. 739–748.

[4] Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. 2017. Answering conjunctive queries under updates. In *PODS*. 303–318.

[5] Aditya Bhaskara, Daniel Dadush, Ravishankar Krishnaswamy, and Kunal Talwar. 2012. Unconditional differentially private mechanisms for linear queries. In *STOC*. 1269–1284.

[6] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2013. Differentially private data analysis of social networks via restricted sensitivity. In *ITCS*. 87–96.

[7] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil Vadhan. 2015. Differentially private release and learning of threshold functions. In *FOCS*. 634–649.

[8] Mark Bun, Jonathan Ullman, and Salil Vadhan. 2018. Fingerprinting codes and the price of approximate differential privacy. *SICOMP* 47, 5 (2018), 1888–1938.

[9] T-H Hubert Chan, Elaine Shi, and Dawn Song. 2011. Private and continual release of statistics. *ACM TISSEC* 14, 3 (2011), 1–24.

[10] Shixi Chen and Shuigeng Zhou. 2013. Recursive mechanism: towards node differential privacy and unrestricted joins. In *SIGMOD*. 653–664.

[11] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. 2012. Differentially private spatial decompositions. In *ICDE*. 20–31.

[12] Nilesh Dalvi and Dan Suciu. 2007. Efficient query evaluation on probabilistic databases. *The VLDB Journal* 16, 4 (2007), 523–544.

[13] Bolin Ding, Marianne Winslett, Jiawei Han, and Zhenhui Li. 2011. Differentially private data cubes: optimizing noise sources and consistency. In *SIGMOD*. 217–228.

[14] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. 2022. R2T: Instance-optimal Truncation for Differentially Private Query Evaluation with Foreign Keys. In *SIGMOD*. 759–772.

[15] Wei Dong and Ke Yi. 2021. Residual Sensitivity for Differentially Private Multi-Way Joins. In *SIGMOD*. 432–444.

[16] Wei Dong and Ke Yi. 2022. A Nearly Instance-optimal Differentially Private Mechanism for Conjunctive Queries. In *PODS*. 213–225.

[17] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*. 486–503.

[18] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *TCC*. 265–284.

[19] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. 2010. Differential privacy under continual observation. In *STOC*. 715–724.

[20] Cynthia Dwork, Moni Naor, Omer Reingold, and Guy N Rothblum. 2015. Pure differential privacy for rectangle queries via private partitions. In *ASIACRYPT*. 735–751.

[21] Robert Fink and Dan Olteanu. 2016. Dichotomies for queries with negation in probabilistic databases. *ACM TODS* 41, 1 (2016), 1–47.

[22] Quan Geng, Wei Ding, Ruiqi Guo, and Sanjiv Kumar. 2020. Tight Analysis of Privacy and Utility Tradeoff in Approximate Differential Privacy. In *AISTATS*. 89–99.

[23] Badih Ghazi, Neel Kamal, Ravi Kumar, Pasin Manurangsi, and Annika Zhang. 2022. Private Aggregation of Trajectories. *PoPETS* 2022, 4 (2022), 626–644.

[24] Todd J Green, Grigoris Karvounarakis, and Val Tannen. 2007. Provenance semirings. In *PODS*. 31–40.

[25] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A simple and practical algorithm for differentially private data release. In *NIPS*. 2348–2356.

[26] Moritz Hardt and Kunal Talwar. 2010. On the geometry of differential privacy. In *STOC*. 705–714.

[27] Xiao Hu, Stavros Sintos, Junyang Gao, K. Pankaj Agarwal, and Jun Yang. 2022. Computing Complex Temporal Join Queries Efficiently. In *SIGMOD*. 2076–2090.

[28] Ziyue Huang and Ke Yi. 2021. Approximate Range Counting Under Differential Privacy. In *SoCG*. 45:1–45:14.

[29] Manas Joglekar and Christopher Ré. 2018. It's all a matter of degree: Using degree information to optimize multiway joins. *TCS* 62(4) (2018), 810–853.

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

[31] Daniel Kifer and Ashwin Machanavajjhala. 2011. No free lunch in data privacy. In *SIGMOD*. 193–204.

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

[33] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. 2010. Optimizing linear counting queries under differential privacy. In *PODS*. 123–134.

[34] Chao Li and Gerome Miklau. 2011. Efficient batch query answering under differential privacy. *arXiv preprint:1103.1367* (2011).

[35] Chao Li and Gerome Miklau. 2012. An adaptive mechanism for accurate query answering under differential privacy. *VLDB* 5(6) (2012), 514–525.

[36] Frank McSherry and Kunal Talwar. 2007. Mechanism Design via Differential Privacy. In *FOCS*. 94–103.

[37] Frank D McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*. 19–30.

[38] Arjun Narayan and Andreas Haeberlen. 2012. DJoin: Differentially Private Join Queries over Distributed Databases. In *OSDI*. 149–162.

[39] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. 2016. The Geometry of Differential Privacy: The Small Database and Approximate Cases. *SICOMP* 45, 2 (2016), 575–616.

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

[41] Davide Proserpio, Sharon Goldberg, and Frank McSherry. 2014. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *VLDB* 7, 8 (2014), 637–648.

[42] Yuchao Tao, Xi He, Ashwin Machanavajjhala, and Sudeepa Roy. 2020. Computing local sensitivities of counting queries with joins. In *SIGMOD*. 479–494.

[43] Salil Vadhan. 2017. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*. Springer, 347–450.

[44] Moshe Y Vardi. 1982. The complexity of relational query languages. In *STOC*. 137–146.

[45] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. Privbayes: Private data release via bayesian networks. *ACM TODS* 42, 4 (2017), 1–41.

A GUARANTEES OF PMW

In this section, we prove the guarantees of the single-table PMW algorithm (Algorithm 2). We stress that this is essentially the same as the proof in [25], but we reprove it for completeness.

Theorem A.1. *Let* $\mathbf{I}, \mathbf{I}'$ *be neighboring instances such that* $|\text{count}(\mathbf{I}) - \text{count}(\mathbf{I}')| \le \Delta$. *Then, Algorithm 2 satisfies the following:*

- *(Privacy)* $PMW(\mathbf{I}) \approx_{(\epsilon,\delta)} PMW(\mathbf{I}')$.
- *(Utility) With probability at least* $1 - 1/\text{poly}(|\boldsymbol{Q}|)$, $PMW(\mathbf{I})$ *produces a dataset such that all queries in* $\boldsymbol{Q}$ *can be answered to within error* $\alpha = O\left((\sqrt{\text{count}(\mathbf{I}) \cdot \widetilde{\Delta}} + \widetilde{\Delta} \cdot \sqrt{\lambda}) \cdot f^{\text{upper}}\right)$.

Privacy Guarantee. Let $\mathbf{I} = (R_1, \ldots, R_m)$ and $\mathbf{I}' = (R'_1, \ldots, R'_m)$, with $|\text{count}(\mathbf{I}) - \text{count}(\mathbf{I}')| \le \Delta$. By the privacy guarantee of the truncated Laplace mechanism, we have $\widehat{n} + \mathsf{TLap}^{\tau(\epsilon/2,\delta/2,\widetilde{\Delta})}_{2\widetilde{\Delta}/\epsilon} \approx_{(\epsilon/2,\delta/2)} \widehat{n}' + \mathsf{TLap}^{\tau(\epsilon/2,\delta/2,\widetilde{\Delta})}_{2\widetilde{\Delta}/\epsilon}$. Let us now condition on $\widehat{n} = \widehat{n}'$. Let $\mathbb{F}, \mathbb{F}'$ be the synthetic data generated for $\mathbf{I}, \mathbf{I}'$ correspondingly. The algorithm starts with the same uniform distribution. Furthermore, in each update, the guarantees of the EM and the Laplace mechanism ensure that (conditioned on results from iteration $1, \ldots, i-1$ being the same), we have $(i, m_i) \approx_{2\epsilon'} (i', m'_{i'})$. By applying advanced composition, we can conclude that, conditioned on $\widehat{n} = \widehat{n}'$, we have that $\mathbb{F} \approx_{(\epsilon/2,\delta/2)} \mathbb{F}'$. Finally, by basic composition (over the $\widehat{n}'$ part and the computation of $\mathbb{F}$ part), we can conclude that $\mathbb{F} \approx_{(\epsilon,\delta)} \mathbb{F}'$.

Utility Guarantee. For convenience, let $\mathbb{J}$ denote the join result of $\mathbf{I}$, and let $n = |\text{count}(\mathbf{I})|$. Define $\xi_i = \max_q |q(\mathbb{F}_{i-1}) - q(\mathbb{J})|$ as the maximum error over all queries in terms of $\mathbb{F}_{i-1}$ and $\mathbb{J}$. Now we are going to bound the maximum error of the PMW algorithm:

$$\max_q |q(\text{avg}_{i \le k} \mathbb{F}_i) - q(\mathbb{J})| = \max_q |\text{avg}_{i \le k} q(\mathbb{F}_i) - q(\mathbb{J})|,$$

which can be further upper bounded by

$$\le \max_q \text{avg}_{i \le k} |q(\mathbb{F}_i) - q(\mathbb{J})| \le \text{avg}_{i \le k} \max_q |q(\mathbb{F}_i) - q(\mathbb{J})| \le \text{avg}_{i \le k} \xi_i.$$

We next state the guarantees from the exponential mechanism and the Laplace mechanism. Let $\gamma = \widetilde{\Delta} \cdot \log |\boldsymbol{Q}|/\epsilon'$.

Lemma A.2. *With probability at least* $1 - 2k/|\mathcal{Q}|^c$ *for any* $c \geq 0$, *for all* $1 \leq i \leq k$, *we have:*

$$|q_i(\mathbb{F}_{i-1}) - q_i(\mathbb{J})| \geq \xi_i - (2c+2) \times \gamma, \text{ and } |m_i - q_i(\mathbb{J})| \leq c \times \gamma.$$

Proof. For the first inequality, we note that the probability EM with parameter ϵ' and sensitivity $\widetilde{\Delta}$ selects a query with quality score at least r less than the optimal is bounded by

$$\Pr[|q_i(\mathbb{F}_{i-1}) - q_i(\mathbb{J})| < \xi_i - (2c+2)/\epsilon'] \leq |\mathcal{Q}| \times \exp\left(-\frac{\gamma \cdot \epsilon'}{\widetilde{\Delta}}\right) \leq \frac{1}{|\mathcal{Q}|^c}.$$

For the second inequality, we note that $|m_i - q_i(\mathbb{J})| \leq c \cdot \gamma$ if and only if $\left|\mathsf{Lap}^{-}_{\widetilde{\Delta}/\epsilon'}\right| \leq c \cdot \gamma$. From Laplace distribution, we have:

$$\Pr\left[\left|\mathsf{Lap}^{-}_{\widetilde{\Delta}/\epsilon'}\right| > c \cdot (\widetilde{\Delta}/\epsilon') \cdot \log|\mathcal{Q}|\right] \leq \exp(-c \cdot \log|\mathcal{Q}|) = 1/|\mathcal{Q}|^c.$$

A union bound over $2k$ events completes the proof. □

We consider how the PMW mechanism can improve the approximation in each round where $q_i(\mathbb{F}_i) - q_i(\mathbb{J})$ has large magnitude. To capture the improvement, we use the relative entropy function:

$$\Psi'_i = \frac{1}{n} \sum_{x \in \mathbb{D}} \mathbb{J}(x) \log\left(\frac{\mathbb{J}(x)}{\mathbb{F}_i(x)}\right), \ \Psi_i = \frac{n}{\widehat{n}} \cdot \Psi_i.$$

Here, we can only show that $\Psi_0 \leq \log|\mathbb{D}|$ and $\Psi_i \geq -1$.

We next consider how Ψ changes in each iteration:

$$\Psi_i - \Psi_{i-1} = \frac{1}{\widehat{n}} \cdot \sum_{x \in \mathbb{D}} \mathbb{J}(x) \cdot \log\left(\frac{\mathbb{F}_i(x)}{\mathbb{F}_{i-1}(x)}\right) = \frac{1}{\widehat{n}} \cdot q_i(\mathbb{J}) \cdot \eta_i - \frac{n}{\widehat{n}} \log \beta_i,$$

where $\eta_i = \frac{1}{2\widehat{n}} \cdot (m_i - q_i(\mathbb{F}_{i-1}))$ and $\beta_i = \frac{1}{\widehat{n}} \cdot \sum_{x \in \mathbb{D}} \exp(q_i(x)\eta_i)\mathbb{F}_{i-1}(x)$.

Using the fact that $\exp(x) \leq 1 + x + x^2$ for $|x| \leq 1$ and $|q_i(x)\eta_i| \leq 1$,

$$\begin{aligned}\beta_i \leq & \frac{1}{\widehat{n}} \cdot \sum_{x \in \mathbb{D}} (1 + q_i(x)\eta_i + q_i^2(x)\eta_i^2)\mathbb{F}_{i-1}(x) \\ \leq & \frac{1}{\widehat{n}} \cdot \sum_{x \in \mathbb{D}} (1 + q_i(x)\eta_i + \eta_i^2)\mathbb{F}_{i-1}(x) \leq 1 + \frac{1}{\widehat{n}} \cdot \eta_i q_i(\mathbb{F}_{i-1}) + \eta_i^2.\end{aligned}$$

Then, we can rewrite $\Psi_i - \Psi_{i-1} = \frac{1}{\widehat{n}} \cdot q_i(\mathbb{J}) \cdot \eta_i - \log \beta_i$. Plugging the upper bound on β_i, we have

$$\Psi_i - \Psi_{i-1} \geq \frac{\eta_i}{\widehat{n}} \cdot q_i(\mathbb{J}) - \left(\frac{\eta_i}{\widehat{n}} \cdot q_i(\mathbb{F}_{i-1}) + \eta_i^2\right) \geq \frac{1}{4\widehat{n}^2}\left((\xi_i - 4\gamma)^2 - \gamma^2\right),$$

By rewriting the last inequality, we have

$$\xi_i \leq \sqrt{4\widehat{n}^2 \cdot (\Psi_i - \Psi_{i-1}) + \gamma^2} + 4 \times \gamma.$$

Then, $\text{avg}_{i \leq k} \xi_i \leq \sqrt{4\widehat{n}^2 \cdot \text{avg}_{i \leq k}(\Psi_i - \Psi_{i-1}) + \gamma^2} + 4\gamma \leq 2\widehat{n} \cdot \sqrt{\frac{\log|\mathbb{D}|}{k}}$ $+5\gamma$, which can be bounded by $O\left(\widehat{n} \cdot \sqrt{\frac{\log|\mathbb{D}|}{k}} + \frac{\log|\mathcal{Q}| \cdot \widetilde{\Delta}\sqrt{k \cdot \log(1/\delta)}}{\epsilon}\right)$.

By taking $k = \frac{\widehat{n} \cdot \epsilon \cdot \sqrt{\log|\mathbb{D}|}}{\widetilde{\Delta} \cdot \log|\mathcal{Q}| \cdot \sqrt{\log(1/\delta)}}$, we can obtain the minimized error as $O(\sqrt{\widehat{n} \cdot \widetilde{\Delta}} \cdot f^{\text{upper}})$. Finally, recall that $\widehat{n} \leq n + O(\widetilde{\Delta} \cdot \lambda)$. This means that the error is at most $O((\sqrt{n \cdot \widetilde{\Delta}} + \widetilde{\Delta} \cdot \sqrt{\lambda}) \cdot f^{\text{upper}})$.

B MISSING PROOFS IN SECTION 3

B.1 Upper Bound Proofs

For neighboring instances $\mathbf{I}, \mathbf{I}'$, $\left|\text{count}(\mathbf{I}) - \text{count}(\mathbf{I}')\right| \leq \tilde{\Delta}$, which follows the definition of $\text{LS}_{\text{count}}(\mathbf{I})$ and the non-negativity of TLap.

Proof of Lemma 3.2. It can be checked that $\text{LS}_{\text{count}}(\cdot)$ has global sensitivity of 1. Therefore, the DP guarantee of the truncated Laplace mechanism implies that $\widetilde{\Delta}$ (computed on Line 1) is $(\epsilon/2, \delta/2)$-DP. Applying the basic composition over this guarantee and the privacy guarantee in Theorem A.1, we can conclude that the entire algorithm is (ϵ, δ)-DP. □

Proof of Lemma 3.7. It follows from the definition that the global sensitivity of $\ln(RS^{\beta}_{\text{count}}(\mathbf{I}))$ is at most β. Furthermore, observe that Line 2 can be rewritten as $\widetilde{\Delta} \leftarrow \exp\left(\ln(RS^{\beta}_{\text{count}}(\mathbf{I})) + \mathsf{TLap}^{\tau(\epsilon/2,\delta/2,\beta)}_{2\beta/\epsilon}\right)$. Therefore, by the guarantee of truncated Laplace mechanism and the post-processing property of DP, we can conclude that $\widetilde{\Delta}$ is $(\epsilon/2, \delta/2)$-DP. Again, applying the basic composition of this guarantee and the privacy guarantee in Theorem A.1, we can conclude that the entire algorithm is (ϵ, δ)-DP. □

B.2 Lower Bound Proofs

The main idea behind proving Theorem 1.6 is similar to that of Theorem 3.5, where one relation encodes the single table, and remaining tables "amplify" both sensitivity and join size by a Δ factor.

Proof of Theorem 1.6. Let $n = \frac{\text{OUT}}{\Delta}$. From Theorem 1.4, there exists a set $\mathcal{Q}_{\text{one}}$ of queries on domain $\mathbb{D}$ for which any (ϵ, δ)-DP algorithm that takes as input a single-table instance $T \in \mathbb{D}$ and outputs an approximate answer to each query in $\mathcal{Q}_{\text{one}}$ within ℓ_∞-error α requires that $\alpha \geq \tilde{\Omega}\left(\min\left\{n, \sqrt{n} \cdot f^{\text{lower}}(\mathbb{D}, \mathcal{Q}_{\text{one}}, \epsilon)\right\}\right)$. For an arbitrary single-table instance $T : \mathbb{D} \to \mathbb{Z}^+$, we construct a multi-table instance $\mathbf{I}$ for $\mathcal{H}$ of input size m, join size OUT, and local sensitivity Δ as follows. We pick the relation with the smallest number of attributes to encode T. W.l.o.g., we assume that $|\mathbf{x}_1| = \min_i |\mathbf{x}_i|$. As $m \geq 2$, there must exist an attribute $y \in \mathbf{x} - \mathbf{x}_1$. Let $x \in \mathbf{x}_1$ be an arbitrary attribute. Let $k = |\mathbf{x} - \mathbf{x}_1|$.

- Set $\text{dom}(x) = \mathbb{D} \times [n]$ for each $x \in \mathbf{x}_1$, and $\text{dom}(y) = [\Delta^{1/k}]$ for each $y \in \mathbf{x} - \mathbf{x}_1$;
- Let $R_1((a,b), \ldots (a,b)) = \mathbf{1}[b \leq T(a)]$ for all $a \in \mathbb{D}$ and $b \in [n]$.
- For $i > 1$, let $R_i(t) = 1$ for all $t \in \mathbb{D}_i$;

It can be easily checked that $\mathbf{I}$ has join size OUT and local sensitivity Δ, and that two neighboring instances T, T' result in neighboring instances $\mathbf{I}, \mathbf{I}'$. Finally, let $\mathcal{Q}_1$ contain queries from $\mathcal{Q}_{\text{one}}$ applied on its first value of every attribute (i.e., $\mathcal{Q}_1 := \{q \circ \pi_{x,1} \mid q \in \mathcal{Q}_{\text{one}}\}$) for $x \in \mathbf{x}_1$, and let $\mathcal{Q}^i$ contain only a single query $q_{\text{all-one}} : \mathbb{D}_i \to \{+1\}$ for every $i \geq 2$. The remaining argument follows exactly the same for the two-table case as in the proof of Theorem 3.5. □

Theorem B.1. *For any* $\Delta > 0$, $\epsilon > 0$, $\delta > 0$, *and* $\gamma > 0$ *such that* $e^{\epsilon} \cdot \gamma + \delta < 1 - \gamma$, *there exists a family* $\mathcal{Q}$ *of linear queries such that any* (ϵ, δ)*-DP algorithm that takes as input an instance* $\mathbf{I}$ *with local sensitivity at most* Δ *and answers each query in* $\mathcal{Q}$ *to within error* α *with probability* $1 - \gamma$, *must satisfy* $\alpha \geq \Omega(\Delta)$.

Proof. Let $\mathcal{Q} = \{\text{count}\}$. Let $\mathbf{I}, \mathbf{I}'$ be neighboring instances of local sensitivity at most Δ such that $|\text{count}(\mathbf{I}) - \text{count}(\mathbf{I}')| \geq \Omega(\Delta)$. Suppose there is an (ϵ, δ)-DP algorithm $\mathcal{A}$ that achieves error $\alpha < |\text{count}(\mathbf{I}) - \text{count}(\mathbf{I}')|/2$ with probability at least $1 - \gamma$. Let $\tilde{\text{count}}(\mathbf{I}), \tilde{\text{count}}(\mathbf{I}')$ be the join sizes released for $\mathbf{I}, \mathbf{I}'$ respectively. Then, as $\mathcal{A}$ preserves (ϵ, δ)-DP, it must be that

$$\begin{aligned}
1 - \gamma &\leq \Pr\left(\tilde{\text{count}}(\mathbf{I}) \in [\text{count}(\mathbf{I}) - \alpha, \text{count}(\mathbf{I}) + \alpha]\right) \\
&\leq e^{\epsilon} \cdot \Pr\left(\tilde{\text{count}}(\mathbf{I}') \in [\text{count}(\mathbf{I}) - \alpha, \text{count}(\mathbf{I}) + \alpha]\right) + \delta \\
&< e^{\epsilon} \cdot \left(1 - \Pr\left(\tilde{\text{count}}(\mathbf{I}') \in [\text{count}(\mathbf{I}') - \alpha, \text{count}(\mathbf{I}') + \alpha]\right)\right) + \delta \\
&\leq e^{\epsilon} \cdot \gamma + \delta < 1 - \gamma,
\end{aligned}$$

a contradiction. □

B.3 Worst-Case Error Bound

We distinguish two cases below: (1) $R_i : \mathbb{D}_i \to \{0, 1\}$; (2) $R_i : \mathbb{D}_i \to \mathbb{Z}^{\geq 0}$. For simplicity, we assume $\epsilon = \Theta(1)$ and $\delta = 1/n^c$ for some constant $c > 0$. Therefore, $\lambda = \Theta(1)$ and $\beta = \Theta(1)$.

In the first case, we recall the AGM bound [3] on the maximum join size of a multi-way join. More specifically, let $\rho(\mathcal{H})$ be the fractional edge covering number of $\mathcal{H}$, which is the minimum value of $\sum_{i \in [m]} W_i$ subject to (1) $\sum_{i : x \in \mathbf{x}_i} W_i \geq 1$ for each $x \in \mathbf{x}$ and (2) $W_i \in [0, 1]$ for each $i \in [m]$. Then, $\text{count}(\mathbf{I}) \leq n^{\rho(\mathcal{H})}$ for any instance $\mathbf{I}$ of input size n. Now, we give an upper bound on the worst-case error. The residual sensitivity $RS^{\beta}_{\text{count}}(\mathbf{I})$ simply degenerates to [15]: $O\left(\max_{i \in [m]} \max_{E \subseteq [m] \setminus \{i\}} T_{[m] \setminus \{i\} \setminus E}(\mathbf{I})\right) = O\left(\max_{E \subsetneq [m]} T_E(\mathbf{I})\right)$. So, $T_E(\mathbf{I})$ is bounded by the maximum join size of $\mathcal{H}_{E, \partial E} = (\cup_{e \in E} e \setminus \partial E, \{\mathbf{x}_i \setminus \partial E : i \in E\})$. From the AGM bound, we have $T_E(\mathbf{I}) \leq n^{\rho(\mathcal{H}_{E, \partial E})}$. The worst-case error in Theorem 1.5 has a closed-form of $O_{\lambda, f^{\text{upper}}}\left(\sqrt{n^{\rho(\mathcal{H})} \cdot \max_{E \subsetneq [m]} n^{\rho(\mathcal{H}_{E, \partial E})}}\right)$. On the other hand, this is always smaller (at least not larger) than $O_{\lambda, f^{\text{upper}}}(n^{\rho(\mathcal{H})})$, i.e., the maximum join size of the input join query.

In the second case, the AGM bound does not hold any more. A simpler tight bound on the maximum join size is $\Theta(n^m)$. Correspondingly, $\max_{E \subsetneq [m]} T_E(\mathbf{I}) \leq n^{m-1}$. Putting everything together, the worst-case error in Theorem 1.5 has a closed-form of $O_{\lambda, f^{\text{upper}}}\left(n^{m - \frac{1}{2}}\right)$.

C MISSING PROOFS IN SECTION 4

C.1 Two-Table Join

Lemma C.1. *Algorithm 5 is (ϵ, δ)-DP.*

Proof. Define $\text{LS}_{\text{count}}(\mathbf{I}) = \max_{b \in \text{dom}(B)} \{\deg^{\mathbf{I}}_{1, B}(b), \deg^{\mathbf{I}}_{2, B}(b)\}$. It is simple to observe that the global sensitivity of LS_{count} is one. Furthermore, the output partition is simply a post-processing of the truncated Laplace mechanism, which satisfies (ϵ, δ)-DP. □

Proof of Lemma 4.1. From Lemma C.1 the partition $\mathbb{I}$ is $(\epsilon/2, \delta/2)$-DP. Furthermore, TwoTable (Algorithm 1) is $(\epsilon/2, \delta/2)$-DP and is applied on disjoint parts of input data. Thus, by the parallel composition theorem and the basic composition theorem, we can conclude that the entire algorithm is (ϵ, δ)-DP. □

Proof of Theorem 4.4. Given an input instance $\mathbf{I}$ of the two-table join, let $\pi_1 = \left\{B^1_1, \ldots, B^{\ell}_1\right\}$ be a fixed partition of $\text{dom}(B)$, such that $b \in B^i_1$ if and only if $\max\{\deg_{1, B}(b), \deg_{2, B}(b)\} \in (\gamma_{i-1}, \gamma_i]$. Let $\pi_2 = \left\{B^1_2, \ldots, B^{\ell}_2\right\}$ be the partition of $\text{dom}(B)$ returned by Algorithm 5. If $b \in B^i_2$, $\max\{\deg_{1, B}(b), \deg_{2, B}(b)\} \in (\gamma_{i-1} - \lambda, \gamma_i]$. It is easy to see that $B^i_1 \subseteq B^i_2 \cup B^{i+1}_2$. For simplicity, we denote the join size contributed by values in $B^i_1 \cap B^i_2$ and in $B^i_1 \setminus B^i_2$ as x_i, y_i respectively. Then, the cost of Algorithm 5 under partition π_2 is $\sum_{i \in [\ell]} \sqrt{x_i + y_{i-1}} \cdot \sqrt{\lambda \cdot 2^i} \leq \sum_{i \in [\ell]} (\sqrt{x_i} + \sqrt{y_{i-1}}) \cdot \sqrt{2^i \cdot \lambda} \leq 2 \sum_{i \in [\ell]} \sqrt{x_i + y_i} \cdot \sqrt{2^i \cdot \lambda}$, thus can be bounded by that under π_1. □

Proof of Theorem 4.5. Our proof consists of two steps:

- **Step (1):** There exists a family $\mathcal{Q}^i$ of queries such that any (ϵ, δ)-DP algorithm that takes as input an instance of join size OUT^i and local sensitivity $\Delta_i = \Theta(2^i \cdot \lambda)$, and outputs an approximate answer to each query in $\mathcal{Q}^i$ to within error α^i must require $\alpha_i \geq \widetilde{\Omega}\left(\min\left\{\text{OUT}^i, \sqrt{\text{OUT}^i \cdot 2^i \cdot \lambda} \cdot f^{\text{lower}}\right\}\right)$.
- **Step (2):** There exists a family $\mathcal{Q}$ of linear queries such that any (ϵ, δ)-DP algorithm that takes as input an instance $\mathbf{I}$ that conforms to $\overrightarrow{\text{OUT}}$ and answers each query in $\mathcal{Q}$ to within error α must require $\alpha \geq \widetilde{\Omega}\left(\max_i \min\left\{\text{OUT}^i, \sqrt{\text{OUT}^i \cdot 2^i \cdot \lambda} \cdot f^{\text{lower}}\right\}\right)$.

We first focus on **step (1)** for an arbitrary $i \in [\lceil \log(\frac{n}{\lambda}) \rceil]$. Let n_i be an arbitrary integer such that $n_i \cdot \lambda \cdot 2^{i-1} \leq \text{OUT}^i \leq n_i \cdot \lambda \cdot 2^i$. Set $\Delta_i = \frac{\text{OUT}^i}{n_i}$, where $\Delta_i \in (\lambda \cdot 2^{i-1}, \lambda \cdot 2^i]$. From Theorem 1.4, let $\mathcal{Q}^i_{\text{one}}$ be the set of hard queries on which any (ϵ, δ)-DP algorithm takes as input any single-table $T^i \in (\{0, 1\}^d)^{n_i}$, and outputs an approximate answer to each query in $\mathcal{Q}^i_{\text{one}}$ to within error α must require $\alpha \geq \widetilde{\Omega}\left(\min\{n_i, \sqrt{n_i} \cdot f^{\text{lower}}(\mathbb{D}, \mathcal{Q}^i_{\text{one}}, \epsilon)\}\right)$. For an arbitrary single-table $T^i \in (\{0, 1\}^d)^{n_i}$, we can construct a two-table instance (R^i_1, R^i_2) as follows:

- Set $\text{dom}(A) = \mathbb{D}$, $\text{dom}(B) = \mathbb{D} \times [n]$ and $\text{dom}(C) = [\Delta_i]$;
- Let $R^i_1(a, (b_1, b_2)) = \mathbf{1}[a = b_1 \cap b_2 \leq T(a)]$ for all $a \in \text{dom}(A)$ and $(b_1, b_2) \in \text{dom}(B)$.
- Let $R^i_2 = 1$ for all $b \in \text{dom}(B)$ and $c \in \text{dom}(C)$.

It can be easily checked that (R^i_1, R^i_2) has join size OUT^i and local sensitivity Δ_i, and that two neighboring instances T^i, T'^i result in neighboring instances $(R^i_1, R^i_2), (R'^i_1, R'^i_2)$ for the two-table join. Finally, let $\mathcal{Q}^i_1$ contain queries from $\mathcal{Q}^i_{\text{one}}$ applied on its first attribute (i.e., $\mathcal{Q}^i_1 := \{q \circ \pi_A \mid q \in \mathcal{Q}_{\text{one}}\}$), and let $\mathcal{Q}_2$ contain only a single query $q_{\text{all-one}} : \mathbb{D}_2 \to \{+1\}$.

We use an argument similar to Theorem 3.5, showing that if there exists an (ϵ, δ)-DP algorithm that takes in a two-table instance of join size OUT^i and local sensitivity Δ_i, and outputs an approximate answer to each query in $\mathcal{Q}^i_{\text{two}}$ to within error α, there exists an (ϵ, δ)-DP algorithm that takes an arbitrary single-table $T_i \in (\{0, 1\})^{n_i}$, and outputs an approximate answer to each query in $\mathcal{Q}^i_{\text{one}}$ to within error $\frac{\alpha^i}{\Delta^i} \geq \widetilde{\Omega}\left(\min\{n_i, \sqrt{n_i} \cdot f^{\text{lower}}(\mathbb{D}, \mathcal{Q}^i_{\text{one}}, \epsilon)\}\right)$; hence $\alpha^i \geq \widetilde{\Omega}\left(\min\left\{\text{OUT}^i, \sqrt{\text{OUT}^i \cdot 2^i \cdot \lambda} \cdot f^{\text{lower}}\right\}\right)$.

Step (2). From Theorem 1.4, let $\mathcal{Q}_{\text{two}}$ be the family of linear queries over $\mathbb{D}_1 \times \mathbb{D}_2$, such that $\mathcal{Q}_{\text{two}} = \left\{\cup_{i \in [m]} q_i : q_i \in \mathcal{Q}^i_{\text{two}}, \forall i \in [\ell]\right\}$. Consider an (ϵ, δ)-DP algorithm $\mathcal{A}$ takes as input an instance that

conforms to $\overrightarrow{\mathsf{OUT}}$ and answers each query in $\mathcal{Q}_{\text{two}}$ to within error α. If $\alpha \leq \widetilde{O}\left(\max_i \min\left\{\mathsf{OUT}^i, \sqrt{\mathsf{OUT}^i \cdot 2^i \cdot \lambda} \cdot f^{\text{lower}}\right\}\right)$, there exists an (ϵ, δ)-DP algorithm that takes an input an instance of join size OUT^i and local sensitivity Δ_i, and answers each query in $\mathcal{Q}^i_{\text{two}}$ for some i within error α_i, contradicting **Step (1)**. □

C.2 Hierarchical Join

Proof of Lemma 4.8. Let r be the root of $\mathcal{T}$ and let $\text{path}(r, x)$ be the set of attributes on the path from r to x. The proof has three steps.

Step 1. We show that for any $T_{E,\mathbf{y}}$ involved, if $T_{E,\mathbf{y}}$ falls into case (2.1), then $\partial E \subseteq \mathbf{y}$, and if $T_{E,\mathbf{y}}$ falls into case (2.2), then $\partial E = \mathbf{y}$. Initially, either case holds for $T_{E,\partial E}$. Moreover, we observe that the cases (2.1) and (2.2) are invoked exchangeably, i.e., $T_{E',\mathbf{y}\cap(\bigvee_{E'})}$ will fall into case (2.2), and $T_{E,\bigwedge_E}$ will fall into case (2.1). Next, we show that this property is preserved in each recursive step:

- $\mathcal{H}_{E,\mathbf{y}}$ is disconnected. By hypothesis, assume $\partial E \subseteq \mathbf{y}$. Consider an arbitrary connected subquery $E'' \in \mathcal{C}_E$. For any pair of $i \in E''$ and $j \notin E$, $\mathbf{x}_i \cap \mathbf{x}_j \in \mathbf{y}$. For any pair of $i \in E''$ and $j \in E \setminus E''$, $\mathbf{x}_i \cap \mathbf{x}_j \in \mathbf{y}$. This way, $\partial E'' \subseteq \mathbf{y}$. Together with $\partial E'' \subseteq \bigwedge_{E''}$, we obtain $\partial E'' \subseteq \mathbf{y} \cap (\bigwedge_{E''})$. On the other hand, for each attribute $x \in \mathbf{y} \cap (\bigwedge_{E''})$, there must exist some $i \in E''$ with $x \in \mathbf{x}_i$ and some $j \notin E$ such that $x \in \mathbf{x}_j$. This way, $\mathbf{y} \cap (\bigwedge_{E''}) \subseteq \partial E''$. Hence, $\partial E'' = \mathbf{y} \cap (\bigwedge_{E''})$ for $T_{E'',\mathbf{y}\cap(\bigwedge_{E''})}$.
- $\mathcal{H}_{E,\mathbf{y}}$ is connected. By hypothesis, assume $\partial E = \mathbf{y}$. As $\mathbf{y} \subsetneq \bigwedge_E$, $\partial E \subseteq \bigwedge_E$. Hence, $T_{E,\bigwedge_E}$ falls into case (2.1) and has $\partial E \subseteq \bigwedge_E$.

Note that $\text{mdeg}_E(\mathbf{y})$ is only introduced in case (2.2), hence $\partial E = \mathbf{y}$.

Step 2. From Definition 4.7, $\mathbf{y} \subseteq \bigwedge_E$. In Case (1) with $|E| = 1$, we must have $\mathbf{y} \neq \mathbf{x}_i$, thus $\mathbf{y} \subsetneq \mathbf{x}_i$. In Case (2.2), $\mathbf{y} \subsetneq \bigwedge_E$ follows the fact that $\mathcal{H}_{E,\mathbf{y}}$ is disconnected. Together, $\mathbf{y} \subsetneq \bigwedge_E$.

Step 3. We first note that $E \subseteq \text{atom}(x)$. Suppose $i \in \text{atom}(x) \setminus E$. We have $\text{path}(r, x) \in \partial E$, hence $\text{path}(r, x) \in \mathbf{y}$ from **Step 1**. This way, $\mathbf{y} = \bigwedge_E$ contradicts **Step 2**. Hence, $\text{atom}(x) \subseteq E$. Together, we have $\text{atom}(x) = E$. Moreover, as $\mathbf{y} \subseteq \bigwedge_E$, there must be $\mathbf{y} \subseteq \text{path}(r, x')$. Suppose $x' \notin \mathbf{y}$. Then, there exists some relation $i \notin E$ with $\text{path}(r, x') \in \mathbf{x}_i$. In this way, $x' \in \partial E$ and therefore $x' \in \mathbf{y}$, yielding a contradiction of $x' \notin \mathbf{y}$. Hence, $\mathbf{y} = \text{path}(r, x')$. □

Proof of Lemma 4.10. We prove the first property by induction. In the base case with $|\mathbb{I}| = 1$, these properties hold. Consider an arbitrary iteration of **while** loop in Algorithm 6. By hypothesis, all sub-instances in $\mathbb{I}$ have disjoint join results, and their union is $\mathsf{Join}^{\mathbb{I}}$. Then, it suffices to show that Decompose$(\mathbf{I}, x)$ generates a set of sub-instances of $\mathbf{I}$, such that they have disjoint join results, and their union is $\mathsf{Join}^{\mathbf{I}}$. The disjointness is easy to show since $(R_{j,i})_{i\in[\ell]}$ forms a partition of R_j, for every $j \in \text{atom}(x)$. The completeness follows the partition of $\text{dom}(\mathbf{y})$ and definition of $R_{j,i}$.

For the second property, we show that each tuple $t \in R_i$ appears in $O(\ell^c)$ sub-instances, where $c = \sum_{x\in\mathbf{x}} |\text{atom}(x)|$. In an invocation of Partition-Hierarchical$(\mathbf{I}, x)$, t appears in $O(\ell)$ sub-instances if $i \notin \text{atom}(x)$, and only one sub-instance otherwise. Overall, the procedure Decompose will be invoked on every non-leaf node in $\mathcal{T}$, thus tuple $t \in R_i$ appears in $O\left(\prod_{x\in\mathbf{x}} \ell\right) = O\left(\ell^{|\mathbf{x}|}\right)$ sub-instances.

Finally, we show that each sub-instance corresponds to a degree characterization. Let us focus on an arbitrary relation $\mathbf{x}_j$ in Algorithm 6. For simplicity, let $\langle x_1, \ldots, x_k\rangle$ be the root-to-node path corresponding to $\mathbf{x}_j$. Also, let $\mathbf{y}_1, \ldots, \mathbf{y}_k$ be the set of ancestors of $x_1, \ldots, x_k$ respectively. When Decompose$(\mathbf{I}, x_1)$ is invoked, the sub-instance $\mathbf{I}'$ with $\text{deg}^{\mathbf{I}'}_{\text{atom}(x_1),\mathbf{y}_1}(t) \in \left(\lambda \cdot 2^{i-1}, \lambda \cdot 2^i\right]$ for any $t \in \text{dom}(\mathbf{y}_1)$, corresponds to $\sigma(\text{atom}(x_1), \mathbf{y}_1) = i$. In the subsequent invocations of Decompose, tuples with the same value $t \in \text{dom}(\mathbf{y}_1)$ always fall into the same sub-instance, hence this property holds. Thus, each sub-instance returned by Algorithm 6 corresponds to a distinct degree characterization. □

Proof of Lemma 4.11. Consider two neighboring instances $\mathbf{I}$ and $\mathbf{I}'$. Note that $|\text{deg}^{\mathbf{I}}_{E,\mathbf{y}}(t) - \text{deg}^{\mathbf{I}'}_{E,\mathbf{y}}(t)| \leq 1$ holds for any $E \subseteq [m]$, $\mathbf{y} \subseteq \bigwedge_E$, and $t \in \text{dom}(\mathbf{y})$. Each tuple in R_i contributes to at most $|\mathbf{x}_i|$ degrees, i.e., $\text{deg}_{\text{atom}(x_i),\mathbf{y}}(\cdot)$. Hence, $\text{deg}^{\mathbf{I}} \approx_{(c'\cdot\epsilon, c'\cdot\delta)} \text{deg}^{\mathbf{I}'}$, where $c' = \max_{i\in[m]} |\mathbf{x}_i|$ is a join-query-dependent quantity.

Moreover, for each sub-instance $\mathbf{I}^\sigma$ returned, $\mathbb{F}^\sigma \approx_{(3\epsilon,3\delta)} \mathbb{F}'^\sigma$ implied by Lemma 3.2. As each tuple participates in at most $O(\ell^c)$ sub-instances, $\bigcup_\sigma \mathbb{F}^\sigma \approx_{(3\ell^c\cdot\epsilon, 3\ell^c\cdot\delta)} \bigcup_\sigma \mathbb{F}'^\sigma$ implied by the group privacy. Putting everything together and using basic composition, we conclude that Algorithm 4 is $(O(\ell^c \cdot \epsilon), O(\ell^c \cdot \delta))$-DP. □

Given a degree configuration σ, we can upper bound $T_E(\mathbf{I})$ as T^σ_E and $\text{RS}_{\text{count}}(\mathbf{I})$ as $\text{RS}^\sigma_{\text{count}}$, using Definition 3.6. Similar to the two-table case, we define the *uniform partition* of an instance $\mathbf{I}$ using true degrees as $\pi^* = \left\{\mathbf{I}^\sigma_{\pi^*} : \sigma \text{ is a degree configuration}\right\}$. Similarly, the error achieved by the partition based on noisy degrees can be bounded by the uniform one:

Theorem C.2. *For any hierarchical join $\mathcal{H}$ and an instance $\mathbf{I}$, a family $\mathcal{Q}$ of linear queries, and $\epsilon > 0, \delta > 0$, there exists an algorithm that is (ϵ, δ)-DP, and with probability at least $1 - 1/poly(|\mathcal{Q}|)$ produces $\mathbb{F}$ such that all queries in $\mathcal{Q}$ can be answered to within error:*

$$\alpha = O\left(\left(\sum_\sigma \sqrt{\text{count}\left(\mathbf{I}^\sigma_{\pi^*}\right) \cdot RS^\sigma_{\text{count}} \cdot \lambda} + RS^\sigma_{\text{count}} \cdot \lambda\right) \cdot f^{\text{upper}}\right),$$

where σ is over all degree configurations of $\mathcal{H}$, $\mathbf{I}^\sigma_{\pi^*}$ is the sub-instance characterized by σ under the uniform partition π^*, and $\text{RS}^\sigma_{\text{count}}$ be the residual sensitivity derived for instance characterized by σ.

Let $\overrightarrow{\mathsf{OUT}} = \left\{\mathsf{OUT}^\sigma \in \mathbb{Z}^{\geq}0 : \sigma \text{ is a degree configuration}\right\}$ be the join size distribution over the uniform partition π^*. An instance $\mathbf{I}$ conforms to $\overrightarrow{\mathsf{OUT}}$ if $\text{count}(\mathbf{I}^\sigma_{\pi^*}) = \Theta(\mathsf{OUT}^\sigma)$, for every degree configuration σ, where $\mathbf{I}^\sigma_{\pi^*}$ is the sub-instance of $\mathbf{I}$ under σ, π^*. Extending the lower bound argument of Theorem 4.5, we obtain the following parameterized lower bound for hierarchical queries.

Theorem C.3. *Given a hierarchical join $\mathcal{H}$ and an arbitrary parameter $\overrightarrow{\mathsf{OUT}}$, for every sufficiently small $\epsilon > 0$, $n_D \geq \mathsf{OUT}^{O(1)}$ and $n_Q \geq (\mathsf{OUT} \cdot \log n_D)^{O(1)}$, there exists a family $\mathcal{Q}$ of queries of size n_Q on domain $\mathbb{D}$ of size n_D such that any $(\epsilon, 1/n^{\omega(1)})$-DP algorithm that takes as input a multi-table instance over $\mathcal{H}$ of input size at most n while conforming to $\overrightarrow{\mathsf{OUT}}$, and outputs an approximate answer to each query in $\mathcal{Q}$ to within error α, must satisfy*

$$\alpha \geq \widetilde{\Omega}\left(\max_\sigma \min\left\{\mathsf{OUT}^\sigma, \sqrt{\mathsf{OUT}^\sigma} \cdot \sqrt{\text{LS}^\sigma_{\text{count}}} \cdot f^{\text{lower}}\right\}\right),$$

where the maximum is over all degree configurations σ of $\mathcal{H}$ and $\text{LS}^\sigma_{\text{count}} = \max_{i\in[m]} T^\sigma_{[m]\setminus\{i\}}$ is the local sensitivity of $\text{count}(\cdot)$ under σ.