Journal of Machine Learning Research 27 (2026) 1-50 Submitted 1/25; Revised 2/26; Published 6/26

Differentially Private Synthetic Data Generation for Relational Databases

Kaveh Alim* MRZ@MIT.EDU
Massachusetts Institute of Technology

Hao Wang* HAO-WANG@REDHAT.COM
MIT-IBM Computing Research Lab & Red Hat AI

Ojas Gulati OJASG@MIT.EDU
Massachusetts Institute of Technology

Akash Srivastava AKASH.SRIVASTAVA@IBM.COM
MIT-IBM Computing Research Lab & IBM Core AI

Navid Azizan AZIZAN@MIT.EDU
Massachusetts Institute of Technology

Editor: Mahdi Soltanolkotabi

Abstract

Existing differentially private (DP) synthetic data generation mechanisms typically assume a single-source table. In practice, data is often distributed across multiple tables with relationships across tables. In this paper, we introduce the first-of-its-kind algorithm that can be combined with *any* existing DP mechanisms to generate synthetic relational databases. Our algorithm iteratively refines the relationship between individual synthetic tables to minimize their approximation errors in terms of low-order marginal distributions while maintaining referential integrity. This algorithm eliminates the need to flatten a relational database into a master table (saving space), operates efficiently (saving time), and scales effectively to high-dimensional data. We provide both DP and theoretical utility guarantees for our algorithm. Through numerical experiments on real-world datasets, we demonstrate the effectiveness of our method in preserving fidelity to the original data. An implementation of our method, alongside the code necessary to reproduce our empirical findings, is accessible at https://github.com/azizanlab/dp-relational.

Keywords: Differential privacy, synthetic data, optimization, sampling, relational database.

1. Introduction

Relational databases play a pivotal role in modern information systems and business operations due to their efficiency in managing structured data (Schleich et al., 2019). According to a Kaggle survey (Kaggle, 2017), 65.5% of users worked extensively with relational data. Additionally, the majority of leading database management systems (e.g., MySQL and Oracle) are built on relational database principles (Ranking, 2023). These systems organize data into multiple tables, each representing a specific entity, and the relationships between tables delineate the connections between these entities. However, the widespread use of re-

*. Equal contribution.

lational databases also carries a significant risk of privacy leakage. For example, if a single table suffers from a privacy breach, all other tables containing sensitive information can be exposed, as they are interconnected and share relationships. Moreover, deleting data in a relational database can be complex, and incomplete deletion can leave traces of data, which can potentially be accessed and reconstructed by attackers.

Today, differential privacy (DP) stands as the de facto standard for privacy protection (Abowd, 2018; The White House, 2023). There is a growing body of research focused on applying DP to generate private synthetic data (see, e.g., Ridgeway et al., 2021; McKenna et al., 2021; Liu et al., 2021b; SmartNoise, 2023). This effort enables data curators to share (synthetic) data while ensuring the privacy of individuals' personal information within the original dataset. In return, they can harness advanced machine learning (ML) techniques employed by end-users trained on the synthetic data. Numerous studies have provided evidence that state-of-the-art (SOTA) DP synthetic data effectively preserves both the statistical properties of the original data and the high performance of downstream predictive models trained on these synthetic datasets when deployed on the original data (Tao et al., 2021; Wang et al., 2023). However, all existing works assume a single-source database (with a few exceptions discussed in Related Work). This motivates the central question we aim to tackle in this paper:

Can we adapt existing DP synthetic data generation algorithms to relational databases while preserving their referential integrity?

The conventional approach—flattening relational databases into a single master table, generating a synthetic master table, and dividing it into separate databases—presents several challenges. To illustrate, consider education data as an example, with two tables (student and teacher information) linked by students' enrollment in a teacher's course. First, flattening may lead to data integrity issues by introducing numerous null values. This makes it difficult to distinguish between missing and intentionally null data. For instance, if a teacher is not currently teaching, their students' information in the master table will be represented by null values. Consequently, the master synthetic table will contain a significant number of null values even if the original tables do not have any null values. Second, flattening introduces concerns related to data scalability and redundancy. Each record in the master table may have an extremely large number of features, which poses challenges for SOTA DP synthetic data generation mechanisms, as their running time grows rapidly as the number of features increases (e.g., McKenna et al., 2022, 2021; Vietri et al., 2022; Aydore et al., 2021). Additionally, certain records from the original tables might be duplicated across multiple entries in the master table, resulting in data redundancy and an increased demand for storage space. Third, breaking down a synthetic master table may disrupt relationships, especially when there are records with shared feature values. For example, if the master table includes two rows with identical student demographic information, it is unclear whether they are the same student or different students with matching demographic details. Finally, since individual records may repeat multiple times in the master table, the sensitivity of statistical queries can significantly increase, potentially becoming unbounded. Consequently, applying existing DP mechanisms may result in a synthetic master table with limited utility.

The goal of this paper is to introduce the first-of-its-kind algorithm that can generate synthetic relational databases while preserving the privacy, statistical properties, and referential integrity of the original data. For this purpose, we first extend the definition of k-way marginal queries (Thaler et al., 2012; Ridgeway et al., 2021) to relational databases and articulate the requirements of referential integrity (Section 2). In the context of relational databases, k-way marginal queries measure the low-order marginal distributions of features (i.e., columns) across different tables. Preserving these distributions is essential for many downstream applications of synthetic data, such as data visualization, building data pre-processing pipelines, and training ML predictive models like logistic regression.

Our main contribution is an algorithm that effectively generates synthetic relational data by minimizing approximation errors in terms of k-way marginal queries (Algorithm 1). This algorithm works as follows. First, we apply an off-the-shelf DP mechanism to generate individual synthetic tables. Then we represent the relational synthetic database as a bipartite graph and establish relationships by learning its bi-adjacency matrix. We prove that k-way marginals can be reformulated as (fractional) linear functions of this bi-adjacency matrix (Lemma 3). Based on this observation, we propose a combinatorial optimization that minimizes the worst-case approximation error of marginal queries between the original and synthetic databases (Section 3). The combinatorial optimization is solved using an iterative algorithm. At each step, we identify a subset of k-way marginal queries with the highest approximation errors. Then we solve a relaxed optimization in continuous space to update the bi-adjacency matrix (Section 4). Following this, we introduce an unbiased sampling algorithm to convert the learned bi-adjacency matrix back into discrete space (Section 5). To generate large-scale synthetic data, we propose a random slicing heuristic that leverages the sparsity of the bi-adjacency matrix, allowing us to optimize bi-adjacency matrices with millions of parameters (Section 6).

Our approach offers several benefits. First, it is highly flexible, as it can be combined with *any* off-the-shelf DP synthetic data generation mechanisms. These DP mechanisms are used to generate individual synthetic tables, and our algorithm can then be applied to build relationships between these tables. Second, it avoids the need to flatten relational databases into a master table; instead, it only requires querying the relational databases to compute low-order marginal distributions, which can be done using SQL aggregate functions. Third, we establish convergence guarantees for our algorithm and prove that our algorithm has linear runtime complexity with respect to the size of the synthetic data (Theorem 7 and 8). These properties, combined with the random slicing heuristic, enable running our algorithm to generate large-scale synthetic data. Finally, we present utility theorems that upper bound the approximation error for estimating k-way marginals using synthetic data generated by our method (Theorem 9 and 11). The proofs leverage properties of negatively associated random variables (Joag-dev and Proschan, 1983) to establish Hoeffding's inequality for dependent variables, along with classical tools from convex geometry and concentration inequalities. We believe these new proof techniques for proving properties of synthetic data could be of independent interest to the DP synthetic data community.

We conduct numerical experiments to validate our method. We couple the iterative algorithm with SOTA DP synthetic data generation mechanisms to generate relational synthetic databases. We present a scalable PyTorch implementation of our algorithm suited for high-dimensional data. The results show that our algorithm effectively maintains key

statistical properties of the original relational database to a significant extent. Note that existing literature on synthetic relational data generation, even without DP guarantees, is limited (Patki et al., 2016; Mami et al., 2022; Yang et al., 2022; Xu et al., 2023; Cai et al., 2023; Francis, 2024; Pang et al., 2024). We hope our efforts (e.g., introducing new benchmark datasets and open-sourcing PyTorch code) can benefit the communities focused on DP synthetic data and inspire new research on this topic.

Related Work

Privacy-preserving synthetic tabular data generation is an active research topic (see e.g., Blum et al., 2013; Ullman and Vadhan, 2020; Boedihardjo et al., 2022; Chen et al., 2015; Ge et al., 2020; Rosenblatt et al., 2020; Wang et al., 2023; He et al., 2023; Donhauser et al., 2024; Greenewald et al., 2024; Bun et al., 2024). For example, a line of work consider learning probabilistic graphical models (Zhang et al., 2017; McKenna et al., 2019, 2021) or generative adversarial networks (Xie et al., 2018; Beaulieu-Jones et al., 2019; Jordon et al., 2019; Tantipongpipat et al., 2019; Neunhoeffer et al., 2020; Bie et al., 2023) with DP guarantees and generating synthetic data by sampling from these models. Another line of work propose to iteratively refine synthetic data to minimize its approximation error on a pre-selected set of workload queries (Hardt et al., 2012; Gaboardi et al., 2014; McKenna et al., 2022; Liu et al., 2021b,a; Vietri et al., 2020; Aydore et al., 2021; Vietri et al., 2022; Liu et al., 2023b; Mohapatra et al., 2023). Kamino is a constraint-aware DP synthesizer that enforces user-specified data constraints in a single-table setting (Ge et al., 2020), and Mohapatra et al. (2023) study DP data generation with missing values; both are single-table generators that could serve as components for our individual tables.

Most existing work on DP synthetic data generation only focuses on a single-source dataset without considering relational database. The only exceptions are Xu et al. (2023); Cai et al. (2023). Among them, Cai et al. (2023) uses graphical models for generating DP synthetic data, incorporating latent variables to capture the primal-foreign-key relationship[1]. However, their approach is restricted to one-to-many relationships (i.e., records in one table have exactly one relationship with the other table) and can only produce synthetic relational data of this type. As discussed in Appendix C, with only one-to-many relationships, the structure of relational data is simpler, making the synthetic data generation process more straightforward and computationally efficient. In contrast, our method can accommodate any type of relationship, including the more complex many-to-many cases (i.e., records in one table can be related to multiple records in another table, and vice versa). A more closely related work is Xu et al. (2023), which can produce multiple tables with many-to-many relationships by leveraging tools from random graph theory and representation learning. However, their method requires using DP-SGD to optimize their generative models for each table generation, whereas our method can seamlessly integrate with *any* existing DP mechanisms for synthetic table generation. This versatility is essential, particularly as marginal-based and workload-based mechanisms often produce higher-quality synthetic tabular data than DP-SGD-based mechanisms (Tao et al., 2021; McKenna et al.,

1. A primary key is a column in the parent table that uniquely identifies each row. A foreign key is a column in the child table that creates relationships between the two tables by referencing the primary key in the parent table.

2021; Wang et al., 2023). Finally, some research explores DP in relational database systems, primarily focusing on releasing statistical queries (see e.g., He, 2021; Johnson et al., 2018; Kotsogiannis et al., 2019). In contrast, our emphasis is on releasing a synthetic copy of the relational database, extending the scope of privacy-preserving data generation methods to a more practical and general setting.

Another line of work focused on the release of DP synthetic graphs (Liu et al., 2023a; Zhang et al., 2021; Eliáš et al., 2020; Upadhyay, 2013; Gupta et al., 2012; Blocki et al., 2012; Sala et al., 2011; Hay et al., 2009; Liu et al., 2024) with the goal of maintaining essential graph properties (e.g., connectivity, degree distribution, and cut approximation). While a relational database can often be represented as a multipartite graph (see Section 2.1), applying these existing approaches to generate synthetic relational databases encounters various challenges. First, most existing methods assume that the vertex set does not have any attribute information and is the same between real and synthetic graphs. They only focus on learning the edge set through DP mechanisms. In our case, however, the vertex sets (representing records) include attribute information (feature values of records) and differ between real and synthetic databases, requiring privacy preservation for both the vertex attributes and the edge set. The only exceptions are Chen et al. (2019); Jorgensen et al. (2016), which consider vertex attributes when generating DP synthetic graphs. However, Chen et al. (2019) only produces synthetic graphs with the same number of vertices as the original, while applying Jorgensen et al. (2016) to generate synthetic relational databases can lead to referential integrity violation (e.g., one-to-many relationships in the original data could become many-to-many in the synthetic data). Finally, existing approaches prioritize preserving graph properties, whereas our objective extends to preserve both graph properties (i.e., relationships between different tables) and statistical properties of relational databases (i.e., low-order marginal distributions).

Broadly, a body of work has investigated applications of relational databases without imposing privacy constraints (Patki et al., 2016; Mami et al., 2022; Yang et al., 2022; Francis, 2024; Pang et al., 2024). For instance, some works focus on training predictive models on relational data (Cvitkovic, 2020; Gan et al., 2024), while others aim to analyze and extract representations from these datasets (Cai et al., 2021; Fey et al., 2023). There is also research on generating synthetic relational databases, but these efforts are typically restricted to handling one-to-many relationships defined by primary-foreign keys (Patki et al., 2016; Mami et al., 2022; Pang et al., 2024). Additionally, some studies have introduced benchmarks for evaluating deep learning models on relational databases (Robinson et al., 2024). In contrast, our work focuses on synthetic relational data generation that supports both one-to-many and many-to-many relationships. Furthermore, we incorporate DP guarantees into the data generation process to ensure robust privacy protection for the original relational databases.

2. Preliminaries and Problem Formulation

We introduce notation, represent relational databases through bipartite graphs, review differential privacy and its properties, and provide an overview of our problem formulation.

2.1 Bipartite Graph and Relational Database

To simplify our presentation, we assume that the relational database $\mathcal{B}$ consists of only two tables $\mathcal{D}_1$ and $\mathcal{D}_2$ (e.g., student and teacher information), each having n_1 and n_2 rows. We represent the relational database as a bipartite graph $\mathcal{B} = (\mathcal{D}_1, \mathcal{D}_2, \boldsymbol{B})$ where each record corresponds to a node; the two tables define two distinct sets of nodes; and the relationships between the tables are depicted as edges connecting these nodes. We represent the edges using a bi-adjacency matrix, denoted as $\boldsymbol{B} \in \{0,1\}^{n_1 \times n_2}$, where $B_{i,j} = 1$ iff the i-th record from table $\mathcal{D}_1$ is connected to the j-th record from table $\mathcal{D}_2$ (e.g. if the i-th student is enrolled in the j-th teacher's course). For each record $\boldsymbol{x}$, we represent its degree as the total number of records from the other table connected to $\boldsymbol{x}$, denoted by $\deg(\boldsymbol{x})$. Finally, we assume the degree of each record is upper bounded by a constant $d_{\max}$.

2.2 Differential Privacy

We recall the definition of differential privacy (DP) (Dwork and Roth, 2014) and provide detailed background information in Appendix A.

Definition 1 *A randomized mechanism $\mathcal{M}$ that takes a relational database as input and returns an output from a set $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy, if for any adjacent relational databases $\mathcal{B}$ and $\mathcal{B}'$ and all possible subsets $\mathcal{O}$ of $\mathcal{R}$, we have*

$$\mathbb{P}(\mathcal{M}(\mathcal{B}) \in \mathcal{O}) \leq e^{\epsilon}\mathbb{P}(\mathcal{M}(\mathcal{B}') \in \mathcal{O}) + \delta. \tag{1}$$

We consider two relational databases $\mathcal{B}$ and $\mathcal{B}'$ adjacent if $\mathcal{B}'$ can be obtained from $\mathcal{B}$ by selecting a table, modifying the values of a single row within this table, and changing all relationships associated with this row.

DP plays a pivotal role in answering statistical queries, with two distinct categories to be considered for relational databases. The first class involves queries pertaining to a single table, say table $\mathcal{D}_1$. In the context of a student-teacher database, an example would be a query about whether a student is a freshman. If we denote the data domain of table $\mathcal{D}_1$ by $\mathcal{X}_1$, a statistical query can be represented by a function $q : \mathcal{X}_1 \to \{0,1\}$ and its average over a database is denoted as $q(\mathcal{B}) \triangleq \frac{1}{n_1}\sum_{\boldsymbol{x} \in \mathcal{D}_1} q(\boldsymbol{x})$. The second class includes cross-table queries, such as whether a teacher and a student belong to the same department. Analogously, these queries are represented by a function $q : \mathcal{X}_1 \times \mathcal{X}_2 \to \{0,1\}$. We denote the average of their values over a database by $q(\mathcal{B}) \triangleq \frac{1}{\mathbf{1}^T \cdot \mathrm{vec}(\boldsymbol{B})}\sum q(\boldsymbol{x}_i, \boldsymbol{x}_j)$ where the sum is over $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{D}_1 \times \mathcal{D}_2$ s.t. $B_{i,j} = 1$ and $\mathrm{vec}(\cdot)$ converts a matrix into a vector. When $(\mathcal{D}_1, \mathcal{D}_2)$ are clear from the context, we also express the query in terms of the bi-adjacency matrix $\boldsymbol{B}$.

2.3 Problem Formulation

Our goal is to generate a privacy-preserving synthetic database $\mathcal{B}_{\text{syn}} = (\mathcal{D}_1^{\text{syn}}, \mathcal{D}_2^{\text{syn}}, \boldsymbol{B}^{\text{syn}})$ that preserves both statistical properties and referential integrity of the original relational database. In terms of statistical properties, our aim is to ensure that the marginal distributions of subsets of features in the synthetic data closely align with those of the real data. To achieve this, we revisit the definition of k-way marginal query and extend it to suit relational databases. For any integer $d \geq 1$, we define $[d] \triangleq \{1, \cdots, d\}$.

Definition 2 *Suppose the domain of table $\mathcal{D}_i$ has d_i categorical features: $\mathcal{X}_i = \mathcal{X}_{i,1} \times \cdots \mathcal{X}_{i,d_i}$. We define a (single-table) k-way workload as a subset of k features: $\mathcal{S} \subseteq [d_1]$ (or $\mathcal{S} \subseteq [d_2]$) with $|\mathcal{S}| = k$. Additionally, given a reference value $\boldsymbol{y}$, we define a (single-table) k-way marginal query as*

$$q_{\mathcal{S},\boldsymbol{y}}(\boldsymbol{x}) \triangleq \prod_{j \in \mathcal{S}} \mathbb{I}(x_j = y_j) \quad \text{for } \boldsymbol{x} \in \mathcal{X}_1 \text{ (or } \mathcal{X}_2) \tag{2}$$

where $\mathbb{I}$ is an indicator function. Analogously, we define a (cross-table) k-way workload as $\mathcal{S}_1 \times \mathcal{S}_2 \subseteq [d_1] \times [d_2]$ with $|\mathcal{S}_1| + |\mathcal{S}_2| = k$, $0 < |\mathcal{S}_1| < k$; additionally, given a reference value $(\boldsymbol{y}_1, \boldsymbol{y}_2)$, we define a (cross-table) k-way marginal query as

$$q_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}(\boldsymbol{x}_1, \boldsymbol{x}_2) \triangleq \prod_{(j_1,j_2) \in \mathcal{S}} \mathbb{I}((x_{1,j_1}, x_{2,j_2}) = (y_{1,j_1}, y_{2,j_2})) \quad \text{for } (\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathcal{X}_1 \times \mathcal{X}_2.$$

We denote the set of all (cross-table) k-way marginal workloads by $\mathcal{W}_{cross,k}$ and their associated queries by $\mathcal{Q}_{cross,k}$.

Our main objective in generating synthetic relational databases is to preserve k-way marginal queries. These low-order statistics are useful for many downstream applications of synthetic data. For example, data visualization (e.g., histograms of each column and correlation matrices), data pre-processing pipelines (e.g., normalizing columns using their mean and standard deviation), and training simple ML models (e.g., linear models) all depend on low-order statistics. Therefore, preserving k-way marginals in synthetic data allows downstream users to maintain statistical insights extracted from synthetic data and train ML models that perform comparably (within a tolerance) to those trained on real data. Another example highlighting the importance of k-way marginals comes from the National Institute of Standards and Technology (NIST), which organized a competition in 2018 to emphasize the need for privacy preservation in synthetic data generation (McKenna et al., 2021; Ridgeway et al., 2021). In this competition, 3-way marginals were used as a utility metric for evaluating the quality of synthetic data. Here we extend the definition of k-way marginals to multiple synthetic tables with relationships.

In maintaining referential integrity, we use a linking table to establish relationships. It is designed to store the IDs of synthetic tables: the entry (i, j) is included in this table if $B_{i,j}^{\text{syn}} = 1$. Moreover, when the original database exhibits a one-to-many relationship (i.e., a record in the parent table can be related to one or more records in the child table, but a record in the child table can be related to *only one* record in the parent table) or one-to-one relationship[2], we expect the synthetic database to preserve this relationship. Finally, in the case of a one-to-many relationship in the original database, we ensure there are no orphaned rows in the synthetic database. Specifically, we impose the requirement that each record in the child table connects to a record in the parent table (although records in the parent table are permitted to have no associated child records).

2. We assume that information about the types of relationships in the original database (e.g., one-to-many or many-to-many) is publicly available.

2.4 Cross-Table Queries

Our key observation is that cross-table marginal queries can be written as a (fractional) linear function of the bi-adjacency matrix. This formulation enables learning the bi-adjacency matrix for a synthetic relational database through an optimization problem. We present this observation formally in the following lemma.

Lemma 3 *For a relational database* $(\mathcal{D}_1, \mathcal{D}_2, \boldsymbol{B})$ *and any* k*-way cross-table query* $q_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}$*, let* $\mathbf{1}_{(\mathcal{S}_1,\boldsymbol{y}_1)} \in \{0,1\}^{n_1}$ *denote an indicator vector whose* i*-th element equals* 1 *iff the* i*-th record in* $\mathcal{D}_1$ *satisfies* $x_{i,j} = y_{1,j}$ *for all* $j \in \mathcal{S}_1$ *and* $\mathbf{1}_{(\mathcal{S}_2,\boldsymbol{y}_2)}$ *is defined in a similar manner. Let* $\boldsymbol{q}_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)} = \mathrm{vec}(\mathbf{1}_{(\mathcal{S}_1,\boldsymbol{y}_1)} \cdot \mathbf{1}^T_{(\mathcal{S}_2,\boldsymbol{y}_2)})$*. Then we have:*

$$q_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}(\boldsymbol{B}) = \frac{\boldsymbol{q}^T_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)} \cdot \mathrm{vec}(\boldsymbol{B})}{\mathbf{1}^T \cdot \mathrm{vec}(\boldsymbol{B})}. \tag{3}$$

Proof See Appendix D.1. ■

The above lemma states that a cross-table query q can be computed as a result of the inner product between $\boldsymbol{q}^T_{(\mathcal{S}_1,\mathcal{S}_2)}$ that is only dependent on tables $\mathcal{D}_1$ and $\mathcal{D}_2$, and $\mathrm{vec}(\boldsymbol{B})$, which shows what edges are active in the bi-adjacency matrix.

3. Main Results

We present our main algorithm (Algorithm 1) for generating privacy-preserving synthetic relational databases. This algorithm can be combined with any existing DP synthetic data generation mechanisms, adapting them to produce relational databases by establishing relationships among individual synthetic tables. Specifically, it learns a bi-adjacency matrix by minimizing approximation errors in terms of cross-table k-way marginal queries compared to the original data. Given the inherent large size of the query class, our algorithm employs an iterative approach to identify the queries with the highest approximation errors and refine the bi-adjacency matrix to reduce these errors. Notably, our algorithm is designed for efficiency and scalability, making it well-suited for high-dimensional data. It also ensures referential integrity in the generated synthetic relational data. We end this section by providing an overview of the rest of the paper, which elaborates on the technical details for our algorithm.

Based on Lemma 3, we write the cross-table query and its corresponding vector $\boldsymbol{q}_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}$ interchangeably. We learn the bi-adjacency matrix for the synthetic database by solving the following optimization:

$$\min_{\substack{\boldsymbol{b}^{\mathrm{syn}} \in \{0,1\}^{n_1^{\mathrm{syn}} \cdot n_2^{\mathrm{syn}}} \\ \mathbf{1}^T \boldsymbol{b}^{\mathrm{syn}} = m^{\mathrm{syn}}}} \max_{\boldsymbol{q} \in \mathcal{Q}_{\mathrm{cross},k}} \left| \frac{1}{m^{\mathrm{syn}}} \boldsymbol{q}^T \boldsymbol{b}^{\mathrm{syn}} - a_q \right|, \tag{4}$$

where $\boldsymbol{b}^{\mathrm{syn}} = \mathrm{vec}(\boldsymbol{B}^{\mathrm{syn}})$ and a_q is the query value computed from the original real data. Since the minimization over $\boldsymbol{b}^{\mathrm{syn}}$ in (4) is a combinatorial optimization, which is inherently challenging to solve, we solve a relaxed problem by allowing $\boldsymbol{b}^{\mathrm{syn}} \in [0,1]^{n_1^{\mathrm{syn}} \cdot n_2^{\mathrm{syn}}}$ and use an unbiased sampling algorithm to convert the obtained values back into integer values.

We present an iterative algorithm for solving the above optimization with DP guarantees. At each iteration, we apply the exponential mechanism to identify cross-table k-way workloads whose corresponding queries have the highest approximation errors between synthetic and real databases. Then we add isotropic Gaussian noise to their query values and update the (vectorized) bi-adjacency matrix $\boldsymbol{b}^{\text{syn}}$ to reduce these approximation errors. Specifically, at iteration t, we stack all the queries reported so far into a matrix $\hat{\boldsymbol{Q}}$ and let their noisy answers be a vector $\hat{\boldsymbol{a}}$. To update $\boldsymbol{b}^{\text{syn}}$, we apply projected gradient descent to solve the following quadratic program (Section 4):

$$\min_{\substack{\boldsymbol{b}\in[0,1]^{n_1^{\text{syn}}\cdot n_2^{\text{syn}}} \\ \mathbf{1}^T\boldsymbol{b}=m^{\text{syn}}}} \left\| \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b} - \hat{\boldsymbol{a}} \right\|_2^2. \tag{5}$$

Suppose $\boldsymbol{b}^{\text{syn}}$ is the output from the preceding procedure, which we reshape into a matrix $\boldsymbol{B}^{\text{syn}}$. Next, we introduce an unbiased sampling algorithm to convert each element of $\boldsymbol{B}^{\text{syn}}$ into $\{0,1\}$ for defining the relationship between synthetic tables (Section 5). To ensure the scalability of our algorithm, we propose a random slicing method that allows us to generate large-scale relational databases (Section 6). Finally, we analyze the utility of our algorithm in preserving k-way marginal queries (Section 7) and validate our algorithm through numerical experiments (Section 8).

We outline the above procedure in Algorithm 1 and provide additional details, including our privacy budget tracking and hyper-parameters, in Section B. Throughout the paper, we assume that the original relational database demonstrates a many-to-many relationship. In Appendix C, we apply our algorithm to accommodate one-to-many relationships, which, as shown, are significantly easier to manage. We remark that iterative algorithms are a standard technique for query releasing in the DP literature (see Gupta et al., 2012; Hardt et al., 2012; Liu et al., 2021b; Aydore et al., 2021; McKenna et al., 2022, for examples in synthetic data/graph generation) and we extend its applications to establish relationships between synthetic tables. We end this section by establishing a DP guarantee and analyzing the overhead computational cost for our algorithm.

Theorem 4 *Assume that each record in the original relational database has at most $d_{\max}$ relationships with records in the other table. We apply any DP mechanism with privacy budgets (ϵ_1, δ_1) and (ϵ_2, δ_2) to generate individual synthetic tables, respectively. Then, we use Algorithm 1 with privacy budget $(\epsilon_{rel}, \delta_{rel})$ to build the relationships between them. The resulting synthetic relational database satisfies $(\epsilon_1 + \epsilon_2 + \epsilon_{rel}, \delta_1 + \delta_2 + \delta_{rel})$-DP.*

Proof For a more rigorous statement of the theorem and proof, see Appendix D.2. ■

Theorem 5 *Algorithm 1 taking $\mathcal{B} = (\mathcal{D}_1, \mathcal{D}_2, \boldsymbol{B})$ and $(\mathcal{D}_1^{syn}, \mathcal{D}_2^{syn})$ as inputs and outputting $\mathcal{B}^{syn} = (\mathcal{D}_1^{syn}, \mathcal{D}_2^{syn}, \boldsymbol{B}^{syn})$ has linear computational complexity with respect to the number of all possible edges in the synthetic relational database $N = n_1^{syn} \cdot n_2^{syn}$. That is, the overhead complexity of Algorithm 1 is $O(T \cdot N)$, where T is the number of iterations.*

Proof See Appendix D.3. ■

Algorithm 1 Adapting DP mechanisms to generate relational synthetic data.

Input: private relational database $\mathcal{B} = (\mathcal{D}_1, \mathcal{D}_2, \boldsymbol{B})$; privacy budget $(\epsilon_{\text{rel}}, \delta_{\text{rel}})$; individual synthetic tables $\mathcal{D}_1^{\text{syn}}$, $\mathcal{D}_2^{\text{syn}}$; workloads per iteration K; number of iterations T; number of relationship in synthetic tables m^{syn}

for $t = 1, \cdots, T$ **do**

- Select K informative workloads ▷ Algorithm 5
- Obtain noisy answers from the real database ▷ Algorithm 6
- Solve the relaxed optimization ▷ Algorithm 2
- Sample a discrete relationship matrix ▷ Algorithm 3

end for

Output: $\mathcal{B}^{\text{syn}} = (\mathcal{D}_1^{\text{syn}}, \mathcal{D}_2^{\text{syn}}, \boldsymbol{B}^{\text{syn}})$

4. Solving the Relaxed Optimization

In the previous section, we introduced our main algorithm for establishing relationships between individual synthetic tables. This algorithm updates the bi-adjacency matrix by solving a relaxed optimization and then mapping the solution back to integer values. In this section, we focus on solving the relaxed optimization:

$$\min_{\boldsymbol{b}\in\Omega} \; f(\boldsymbol{b}) \quad \text{where} \quad f(\boldsymbol{b}) \triangleq \left\| \frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}}\boldsymbol{b} - \hat{\boldsymbol{a}} \right\|_2^2 \quad \text{and} \quad \Omega \triangleq \{\boldsymbol{b} \mid 0 \le \boldsymbol{b} \le 1, \mathbf{1}^T\boldsymbol{b} = m^{\text{syn}}\} \subseteq \mathbb{R}^N. \tag{6}$$

Here we denote the dimension of the (vectorized) bi-adjacency matrix as $N \triangleq n_1^{\text{syn}} \cdot n_2^{\text{syn}}$.

The main challenge lies in the high dimensionality of the bi-adjacency matrix (i.e., N is extremely large). This makes off-the-shelf general convex quadratic program solvers either computationally expensive or unable to return solutions that satisfy the constraint set Ω in our problem. For example, running the interior point algorithm for a single iteration can take $O(N^5)$ arithmetic operations (Ye and Tse, 1989), while ADMM-based solvers may fail to fully satisfy all constraints before convergence, meaning the solution might not lie within Ω (Schubiger et al., 2020). Ensuring all constraints are satisfied is critical since the sampling algorithm later seen in the paper requires the bi-adjacency matrix returned from the relaxed optimization meeting all constraints to guarantee that each pair of records from different synthetic tables has at most one relationship, and the total number of relationships in the synthetic database is exactly m^{syn}.

To address these challenges, we use the projected gradient descent algorithm to solve (6). This choice is motivated by our findings that both the gradient of the objective function and the projection onto the constraint set Ω can be computed in a closed form. These analytical expressions enable each iteration to run in $O(N)$ operations, significantly reducing computational costs. Moreover, the closed-form expression for the projection onto Ω ensures that the constraints are satisfied with negligible error. Finally, we establish a convergence guarantee, showing that our algorithm converges to the optimal value at a rate of $O\left(\frac{1}{T}\right)$, where T is the number of iterations. This result also informs our choice of the learning rate. In the following, we provide details about our algorithm.

At iteration t, the projected gradient descent updates $\boldsymbol{b}$ following the updating rule:

$$\boldsymbol{b}_{t+1} = \Pi_\Omega \left(\boldsymbol{b}_t - \eta \nabla f(\boldsymbol{b}_t) \right), \tag{7}$$

where Π_Ω denotes the Euclidean projection onto the set Ω and η is the step size. The gradient of the objective quadratic function has a closed-form expression:

$$\nabla f(\boldsymbol{b}) = \frac{2}{m^{\text{syn}}}\hat{\boldsymbol{Q}}^T\left(\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b} - \hat{\boldsymbol{a}}\right). \tag{8}$$

Next, we explore how to project $\boldsymbol{b}$ (after taking the gradient descent step) onto the constraint set Ω. This projection can be formulated as a quadratic program with a single sum constraint, as well as upper and lower bound constraints. Using a classical result from Pardalos and Kovoor (1990), we derive a closed-form expression for the projection vector in the following theorem.

Theorem 6 *For a given $\boldsymbol{b} \in \mathbb{R}^N$, we define a collection of functions, indexed by $i \in [N]$, as follows:*

$$h_i(y) \triangleq \begin{cases} 0 & \text{if } y < -b_i, \\ y + b_i & \text{if } -b_i \le y \le 1 - b_i, \\ 1 & \text{if } y > 1 - b_i. \end{cases}$$

Then the Euclidean projection of $\boldsymbol{b}$ onto the set $\Omega \triangleq \{\boldsymbol{b} \mid 0 \le \boldsymbol{b} \le 1, \mathbf{1}^T\boldsymbol{b} = m^{syn}\}$ has a closed-form expression:

$$[\Pi_\Omega(\boldsymbol{b})]_i = h_i(y^*),$$

where $[\Pi_\Omega(\boldsymbol{b})]_i$ is the i-th element of $\Pi_\Omega(\boldsymbol{b})$ and y^* is any real value satisfying $\sum_{i=1}^N h_i(y^*) = m^{syn}$.

Proof See Appendix D.4. ■

The closed-form expression for the Euclidean projection $\Pi_\Omega(\boldsymbol{b})$ relies on a *single* auxiliary variable y. Given that $\sum_{i=1}^N h_i(y) = m^{\text{syn}}$ is a piecewise linear, non-decreasing function, we can apply binary search over the interval $[-\max_{i\in[N]}\{b_i\}, 1 - \min_{i\in[N]}\{b_i\}]$ to find a y that satisfies the required equality.

Theorem 6 provides several advantages for computing the projection onto Ω. First, it offers an efficient algorithm with $O(N)$ runtime complexity, as it requires solving for only a single variable y through the equality constraint. Moreover, the binary search approach provides full control over the solution's precision, ensuring a projection onto the constraint set with negligible error. This precision is necessary in our application since the sampling algorithm in the next section relies on the box constraint to ensure each pair of records from different synthetic tables has at most one relationship, and the sum constraint to guarantee that the total number of relationships in the synthetic database is exactly m^{syn}.

We can now compute the gradient of the objective function in closed form (8) and project any point onto the constraint set, also in closed form. These two results enable an efficient implementation of using the projected gradient descent to solve the relaxed optimization problem in (6) (see Algorithm 2 for a detailed implementation). Below, we establish a convergence guarantee for this algorithm and analyze its runtime complexity. The results show that our algorithm produces a weighted bi-adjacency matrix that satisfies the constraints, while achieving an objective value within $O\left(\frac{1}{T}\right)$ of the optimal solution after T iterations of projected gradient descent.

Theorem 7 *Let $\boldsymbol{b}^* \in \operatorname{argmin}_{\boldsymbol{b}\in\Omega} f(\boldsymbol{b})$ be an optimal solution of the relaxed optimization problem in (6). We denote the largest singular value of $\hat{\boldsymbol{Q}}$ by $\sigma_{max}(\hat{\boldsymbol{Q}})$. Let b_T denote the T-th iterate of projected gradient descent in (7) with a step size $\eta = \frac{1}{2}\cdot\left(\frac{m^{syn}}{\sigma_{max}(\hat{\boldsymbol{Q}})}\right)^2$. Then we have*

$$f(\boldsymbol{b}_T) - f(\boldsymbol{b}^*) \leq \frac{C}{T},$$

where $C \triangleq 3\|\boldsymbol{b}_0 - \boldsymbol{b}^*\|_2^2/\eta + f(\boldsymbol{b}_0) - f(\boldsymbol{b}^*)$ is a constant that only depends on the choice of the initialization. Additionally, the runtime complexity of the projected gradient descent as a function of N is $O(N)$.

Proof See Appendix D.5. ■

This convergence result suggests choosing the step size η as $\frac{1}{2}\cdot\left(\frac{m^{\text{syn}}}{\sigma_{\max}(\hat{\boldsymbol{Q}})}\right)^2$ to achieve a convergence rate of $O\left(\frac{1}{T}\right)$. This step size depends on the largest singular value of $\hat{\boldsymbol{Q}}$, which can be efficiently computed using the power iteration method. Since the power method only involves matrix-vector multiplications in each iteration, implementing it in PyTorch allows for GPU acceleration, significantly reducing computation time.

5. Unbiased Sampling

In the previous section, we applied the projected gradient descent algorithm to solve the relaxed optimization problem (5). This algorithm produced a weighted bi-adjacency matrix $\widetilde{\boldsymbol{B}}^{\text{syn}} \in [0,1]^{n_1^{\text{syn}}} \times [0,1]^{n_2^{\text{syn}}}$. In this section, we convert $\widetilde{\boldsymbol{B}}^{\text{syn}}$ back into the discrete space $\{0,1\}^{n_1^{\text{syn}}} \times \{0,1\}^{n_2^{\text{syn}}}$. Our goal is to sample an unbiased[3] instance $\boldsymbol{B}^{\text{syn}}$ from $\widetilde{\boldsymbol{B}}^{\text{syn}}$ such that the synthetic relational database has exactly m^{syn} relationships (i.e., $\mathbf{1}^T \operatorname{vec}(\boldsymbol{B}^{\text{syn}}) = m^{\text{syn}}$). We will discuss why the traditional sample-and-rejection method fails to provide such an unbiased sample. Then we will introduce a novel recursive sampling algorithm, prove that it yields an unbiased sample, and analyze its runtime complexity. To make the following proposed algorithm and properties, we have denoting $\widetilde{\boldsymbol{b}} \triangleq \operatorname{vec}(\boldsymbol{B}^{\text{syn}})$, and omitted the superscript syn from m^{syn} and used m instead, however, when applying results to the problem at hand we switch back to the notation used throughout the paper.

We formalize our objective as follows: given a vector $\widetilde{\boldsymbol{b}} \in [0,1]^N$ that satisfies $\mathbf{1}^T\widetilde{\boldsymbol{b}} = m$, we aim to sample an unbiased instance $\boldsymbol{b} \in \{0,1\}^N$ such that $\mathbb{P}(b_i = 1) = \widetilde{b}_i$ for all $i \in [N]$ and $\mathbf{1}^T\boldsymbol{b} = m$. If we did not have the hard constraint $\mathbf{1}^T\boldsymbol{b} = m$, unbiased sampling could be achieved by independently sampling each element of $\boldsymbol{b}$ according to the corresponding probability in $\widetilde{\boldsymbol{b}}$ (i.e., $b_i \sim \mathsf{Bernoulli}(\widetilde{b}_i)$ for $i \in [N]$). However, this strategy clearly does not guarantee that $\mathbf{1}^T\boldsymbol{b} = m$. This can be seen by considering a special example where $\widetilde{\boldsymbol{b}} = [\widetilde{b}, \cdots, \widetilde{b}]$. In this case, $\mathbb{P}(\mathbf{1}^T\boldsymbol{b} = m) = \binom{N}{m}\widetilde{b}^m(1-\widetilde{b})^{N-m} \neq 1$.

A possible solution to fix this issue is to use a sample-and-rejection method. In this approach, we repeatedly sample indices from $[N]$ according to the probability distribution $\widetilde{\boldsymbol{b}}/m$. If the current index has already been selected in a previous iteration, we reject it and continue sampling until m unique indices have been chosen. These selected indices

3. Unbiased means that $\mathbb{P}(B_{i,j}^{\text{syn}} = 1) = \widetilde{B}_{i,j}^{\text{syn}}$ for all $i \in [n_1^{\text{syn}}]$ and $j \in [n_2^{\text{syn}}]$.

Algorithm 2 Solving the relaxed optimization via projected gradient descent.

Input: query matrix $\hat{\boldsymbol{Q}}$; noisy answer vector $\hat{\boldsymbol{a}}$; number of relationships in synthetic tables m^{syn}; number of iterations for running projected gradient descent T; number of iterations for running power iteration T_{power}

function PROJECTION($\boldsymbol{b}$, m^{syn})
  Find y^* by solving $\sum_{i=1}^{N} \max\{\min\{y^*, 1-b_i\}, -b_i\} = m^{\text{syn}} - \mathbf{1}^T\boldsymbol{b}$ ▷ Apply binary search
  for $i = 1, \cdots, N$ **do**
    $b_i^{\text{proj}} = \max\{\min\{y^*, 1-b_i\}, -b_i\} + b_i$
  end for
  return $[b_1^{\text{proj}}, \cdots, b_N^{\text{proj}}]$
end function

// Determine the step size:
Initialize $\boldsymbol{v} \in \mathbb{R}^N$ randomly
for $t = 1, \cdots, T_{\text{power}}$ **do**
  $\boldsymbol{w} = \hat{\boldsymbol{Q}}^T\hat{\boldsymbol{Q}}\boldsymbol{v}$
  $\boldsymbol{v} = \boldsymbol{w}/\|\boldsymbol{w}\|_2$
end for
$\sigma_{\max} = \|\hat{\boldsymbol{Q}}\boldsymbol{v}\|_2$ ▷ Compute the largest singular value
$\eta = \frac{1}{2}\left(\frac{m^{\text{syn}}}{\sigma_{\max}}\right)^2$ ▷ Compute step size according to Theorem 7

// Run projected gradient descent:
Initialize $\boldsymbol{b}_0 \in \mathbb{R}^N$ randomly
for $t = 1, \cdots, T$ **do**
  $\nabla f(\boldsymbol{b}_{t-1}) = \frac{2}{m^{\text{syn}}}\hat{\boldsymbol{Q}}^T\left(\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b}_{t-1} - \hat{\boldsymbol{a}}\right)$
  $\boldsymbol{b}_t = \boldsymbol{b}_{t-1} - \eta\nabla f(\boldsymbol{b}_{t-1})$
  $\boldsymbol{b}_t = \text{PROJECTION}(\boldsymbol{b}_t, m^{\text{syn}})$
end for
$\widetilde{\boldsymbol{B}}^{\text{syn}} = \boldsymbol{b}_T.\text{reshape}(n_1^{\text{syn}}, n_2^{\text{syn}})$

Output: weighted bi-adjacency matrix $\widetilde{\boldsymbol{B}}^{\text{syn}}$

will determine the positions in $\boldsymbol{b}$ where the value is 1. Alas, while this approach ensures $\mathbf{1}^T\boldsymbol{b} = m$, it fails to provide an unbiased sample. We demonstrate it through the following example.

Example 1 *Consider $N = 3$ and $m = 2$, with $\widetilde{\boldsymbol{b}} = [\widetilde{b}_1, \widetilde{b}_2, \widetilde{b}_3]$ such that $\widetilde{b}_1 + \widetilde{b}_2 + \widetilde{b}_3 = 2$. There are two scenarios in which the sample-and-rejection algorithm will output the index 1 among its two output indices: (i) the index 1 is selected in the first iteration, or (ii) the index 2 (or 3) is selected first, repeatedly selected and rejected, until finally, the index 1 is selected: $2\underbrace{2\ldots2}_{\text{reject}}1$ or $3\underbrace{3\ldots3}_{\text{reject}}1$. Hence, we have:*

$$\begin{aligned}\mathbb{P}(b_1 = 1) &= \frac{\widetilde{b}_1}{2} + \left(\frac{\widetilde{b}_2}{2} + \frac{\widetilde{b}_2^2}{4} + \cdots\right)\frac{\widetilde{b}_1}{2} + \left(\frac{\widetilde{b}_3}{2} + \frac{\widetilde{b}_3^2}{4} + \cdots\right)\frac{\widetilde{b}_1}{2} \\ &= \frac{\widetilde{b}_1}{2}\left(1 + \frac{\widetilde{b}_2}{2 - \widetilde{b}_2} + \frac{\widetilde{b}_3}{2 - \widetilde{b}_3}\right).\end{aligned}$$

In general, this probability is not equal to $\widetilde{b}_1$, leading to a biased sample when using the sample-and-rejection algorithm.

In Algorithm 3, we present the unbiased sampling algorithm (the function UBS) and demonstrate how it can be applied to the problem of generating the unbiased discrete bi-adjacency matrix as in Algorithm 1. The core of our algorithm is a function, denoted by UBS, which takes as input a vector $\boldsymbol{x} = [x_1, \cdots, x_N]$ where $0 \leq x_i \leq 1$ for all $i \in [N]$ and $\sum x_i = m$ for an integer m. It selects m distinct indices from the set $[N]$ in an unbiased manner:

$$\mathbb{P}(\text{index } k \text{ is selected}) = x_k \quad \forall k \in [N].$$

This function works as follows. We sequentially partition the indices of $\boldsymbol{x}$ into L groups such that the sum of x_i for each group does not exceed 1 and cannot be increased further. Then we select m groups and from each selected group, we choose exactly one index. The key idea of our algorithm is that instead of directly sampling m groups, we sample $L - m$ groups according to their complement probabilities, which can be done by recursively calling the UBS function itself. The greedy choice of groups guarantees that the number of groups decreases exponentially in each iteration of the recursion. As a result, the algorithm has linear runtime w.r.t. N (Theorem 8). Below, we provide more details about our algorithm (see Algorithm 3 for a pseudo-code).

Base cases. We begin with three base cases. If $m = 0$, the algorithm returns an empty set. If $m = 1$, then $\boldsymbol{x}$ already lies within a probability simplex, so we can directly sample an index according to the probability distribution $\boldsymbol{x}$ (i.e., sample from the categorical distribution parameterized by $\boldsymbol{x}$). If $m = N$, we know $x_i = 1$ for all $i \in [N]$, which implies that we can output the entire index set $[N]$. Otherwise, we recursively apply three steps: merge, complement, and sample.

Merging. We partition the indices of $\boldsymbol{x}$ into disjoint groups: $\mathcal{G}[0], \cdots, \mathcal{G}[L-1]$. These groups are formed greedily by iterating through the index set $[N]$ and sequentially adding indices to the current group. An index is added to the current group as long as the sum of x_i, for indices i within the group, does not exceed 1. We denote by $\mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j])$ the sum of elements in $\boldsymbol{x}$ whose index belongs to $\mathcal{G}[j]$:

$$\mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j]) \triangleq \sum_{i \in \mathcal{G}[j]} x_i. \tag{9}$$

Based on our construction, we have $\mathcal{G}[j] = \{z_j, z_j + 1, \cdots, z_{j+1} - 1\} \subseteq [N]$, $1 = z_0 < z_1 < \cdots < z_L = N + 1$, and $0 \leq \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j]) \leq 1$.

Complement. We aim to select m groups from $\mathcal{G}[0], \cdots, \mathcal{G}[L-1]$ such that the sampling process is unbiased:

$$\mathbb{P}(\text{group } \mathcal{G}[j] \text{ is selected}) = \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j]).$$

At first glance, one might attempt to solve this problem by applying $\mathsf{UBS}(\boldsymbol{p}, m)$, where $\boldsymbol{p} = [p_0, \ldots, p_{L-1}]$ with $p_j = \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j])$. However, this approach fails because $\boldsymbol{p}$ cannot

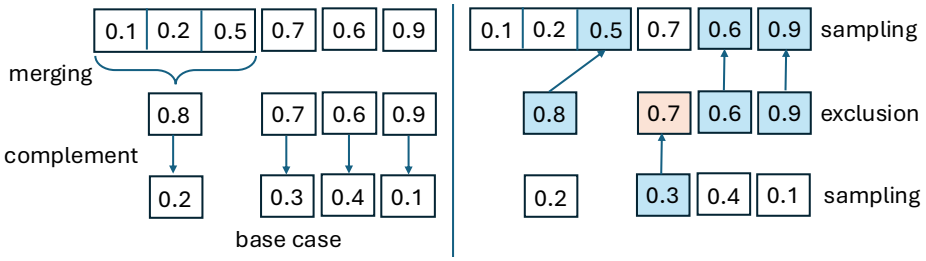

Figure 1: An illustration of the function UBS in Algorithm 3. Consider $\boldsymbol{x} = [0.1, 0.2, 0.5, 0.7, 0.6, 0.9]$ and $m = 3$. In the first merging step, we partition the indices of $\boldsymbol{x}$ into $L = 4$ groups: $\mathcal{G}[0] = \{1, 2, 3\}$, $\mathcal{G}[1] = \{4\}$, and so on. The corresponding group probabilities are $\mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[0]) = 0.8$, $\mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[1]) = 0.7$, etc. Next, we compute the complement of these probabilities and apply $\mathsf{UBS}([0.2, 0.3, 0.4, 0.1], 1)$ to select $L - m = 1$ group to exclude, which leads to the base case. We then sample from the probability vector $[0.2, 0.3, 0.4, 0.1]$; suppose we select group $\mathcal{G}[1]$, which is then excluded. After this, we sample exactly one index from $\mathcal{G}[0]$ according to the normalized probabilities $[0.1/0.8, 0.2/0.8, 0.5/0.8]$. Since $\mathcal{G}[2]$ and $\mathcal{G}[3]$ contain only one element each, those elements are included in the final output. The final output is the index set $\{3, 5, 6\}$.

be further merged, causing the recursive algorithm to enter an infinite loop. Our key idea is to apply $\mathsf{UBS}(\mathbf{1} - \boldsymbol{p}, L - m)$ to select $L - m$ groups to exclude. In other words, instead of directly selecting m groups from $\mathcal{G}[0], \ldots, \mathcal{G}[L-1]$, we reframe the problem: how can we select $L - m$ groups based on the complement probabilities? These groups will be ruled out, and the remaining m groups will be selected.

Sampling. Suppose $\mathcal{G}[j]$ is one of the m remaining non-excluded groups. From $\mathcal{G}[j]$, we sample exactly one index i with probability proportional to x_i:

$$\mathbb{P}(i \text{ is sampled} \mid \text{group } \mathcal{G}[j] \text{ is selected}) = \frac{x_i}{\mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j])} \quad \text{for } i \in \mathcal{G}[j].$$

We return the set of indices selected from all m non-excluded groups.

Theorem 8 *The function* UBS *in Algorithm 3 outputs an unbiased sample. Additionally, the runtime of Algorithm 3 is $O(N)$ where $N = n_1^{syn} \cdot n_2^{syn}$.*

Proof See Appendix D.6. ■

An unbiased sampling approach offers significant benefits over alternative methods. Suppose Algorithm 3 takes a weighted bi-adjacency matrix $\widetilde{\boldsymbol{B}}^{\text{syn}}$ as input and produces $\boldsymbol{B}^{\text{syn}}$ as output. The unbiased property ensures that $\mathbb{E}[\boldsymbol{B}^{\text{syn}}] = \widetilde{\boldsymbol{B}}^{\text{syn}}$. Given that k-way cross-table queries are linear functions of the bi-adjacency matrix, computing these queries from $\boldsymbol{B}^{\text{syn}}$ provides an unbiased estimate of the corresponding queries computed

Algorithm 3 A recursive algorithm for unbiased sampling.

Input: weighted bi-adjacency matrix $\widetilde{\boldsymbol{B}}^{\text{syn}} \in [0,1]^{n_1^{\text{syn}}} \times [0,1]^{n_2^{\text{syn}}}$, number of relationship in synthetic tables m^{syn}
function UBS($\boldsymbol{x}$, m)
  Assert $\mathbf{1}^T\boldsymbol{x} == m$
  $N = \text{length}(\boldsymbol{x})$
  if $m == 0$ **then**
    return $\{\}$
  else if $m == 1$ **then**
    Draw k from $\{1, \cdots, N\}$ according to the probability distribution $\boldsymbol{x}$
    return $\{k\}$
  else if $m == N$ **then**
    return $\{1, \cdots, N\}$
  end if
  // Merging procedure:
  $\mathcal{G} = [\{\}]$ ▷ List of groups, where groups are sets of indices
  $i = 0$
  while $i < N$ **do**
    if $\mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[-1]) + x_i > 1$ **then**
      Append an empty set to $\mathcal{G}$ ▷ Finish the current group
    else
      Add i to the set $\mathcal{G}[-1]$
    end if
    $i = i + 1$
  end while
  // Complement procedure and recursive step:
  $L = \text{length}(\mathcal{G})$
  $\boldsymbol{p}' = [1 - \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[i]) \text{ for } i = 0, \cdots, L-1]$
  $\mathcal{J} = \text{UBS}(\boldsymbol{p}', L - m)$
  // Sampling procedure:
  $\mathcal{O} = \{\}$
  for $i = 1, \cdots, L$ **do**
    if $i \notin \mathcal{J}$ **then**
      $\boldsymbol{q} = [x[j] \text{ for } j \in \mathcal{G}[i]]$ and $\boldsymbol{q} = \boldsymbol{q}/(\mathbf{1}^T\boldsymbol{q})$
      Draw s from $\mathcal{G}[i]$ according to the probability distribution $\boldsymbol{q}$
      Add s to $\mathcal{O}$
    end if
  end for
  return $\mathcal{O}$
end function
$\widetilde{\boldsymbol{b}}^{\text{syn}} = \widetilde{\boldsymbol{B}}^{\text{syn}}.\text{flatten}()$
$\mathcal{O} = \text{UBS}(\widetilde{\boldsymbol{b}}^{\text{syn}}, m^{\text{syn}})$
$\boldsymbol{b}^{\text{syn}} = [1 \text{ if } i \in \mathcal{O} \text{ else } 0 \text{ for } i = 1, \cdots, n_1^{\text{syn}} \cdot n_2^{\text{syn}}]$
$\boldsymbol{B}^{\text{syn}} = \boldsymbol{b}^{\text{syn}}.\text{reshape}(n_1^{\text{syn}}, n_2^{\text{syn}})$
Output: unweighted bi-adjacency matrix $\boldsymbol{B}^{\text{syn}}$

from $\widetilde{\boldsymbol{B}}^{\text{syn}}$. Specifically, Lemma 3 implies that, for any query vector $\boldsymbol{q}$, its value computed

from a relational database with bi-adjacency matrix $\boldsymbol{B}$ is $\frac{1}{m^{\text{syn}}}\boldsymbol{q}^T \operatorname{vec}(\boldsymbol{B})$. Thus, we have

$$\mathbb{E}\left[\frac{1}{m^{\text{syn}}}\boldsymbol{q}^T \operatorname{vec}(\boldsymbol{B}^{\text{syn}})\right] = \frac{1}{m^{\text{syn}}}\boldsymbol{q}^T \operatorname{vec}(\widetilde{\boldsymbol{B}}^{\text{syn}}).$$

In other words, as long as the relaxed optimization in Section 4 yields an accurate solution, the unbiased sample can maintain accuracy (on average) in estimating marginal queries.

Next, we provide a high-probability bound to control the deviation of $\frac{1}{m^{\text{syn}}}\boldsymbol{q}^T \operatorname{vec}(\boldsymbol{B}^{\text{syn}})$ from its expected value for specific instances generated by Algorithm 3. Typically, this can be achieved using Hoeffding's inequality (Hoeffding, 1963) or related concentration bounds. However, standard Hoeffding bounds assume independent random variables—a condition not met in our sampling problem due to the sum constraint $\mathbf{1}^T \operatorname{vec}(\boldsymbol{B}^{\text{syn}}) = m^{\text{syn}}$. To resolve this issue, we prove that the elements of $\boldsymbol{B}^{\text{syn}}$ are negatively associated random variables (Joag-dev and Proschan, 1983; Dubhashi and Ranjan, 1998), enabling us to apply Hoeffding's bound effectively. A key step of our proof is to show that the merging and complement steps in our algorithm preserve the negative association property. We formally state this result in the following theorem.

Theorem 9 *Suppose Algorithm 3 takes a weighted bi-adjacency matrix $\widetilde{\boldsymbol{B}}^{syn}$ as input and outputs $\boldsymbol{B}^{syn}$. Let $\mathcal{Q}$ represent a set of k-way cross-table queries of interest. Then, with probability at least $1-\beta$, any query vector $\boldsymbol{q} \in \mathcal{Q}$ satisfies*

$$\left|\frac{1}{m^{syn}}\boldsymbol{q}^T \operatorname{vec}(\boldsymbol{B}^{syn}) - \frac{1}{m^{syn}}\boldsymbol{q}^T \operatorname{vec}(\widetilde{\boldsymbol{B}}^{syn})\right| \leq \frac{1}{m^{syn}} \cdot \sqrt{\frac{N}{2} \cdot \log \frac{2|\mathcal{Q}|}{\beta}},$$

where $N = n_1^{syn} \cdot n_2^{syn}$.

Proof For a more rigorous statement of the theorem and proof, see Appendix D.7. ■

The results in the above theorem apply not only to k-way marginal queries but also to any statistical queries that can be expressed as a linear function of the bi-adjacency matrix. Additionally, we remark that Srinivasan (2001) also studied the problem of sampling N indicator random variables with a sum constrained to m^{syn} and proposed an unbiased sampling method. Although their algorithm and ours both have a worst-case time complexity of $O(N)$, empirical comparisons show that our algorithm runs significantly faster when the ratio m^{syn}/N is small. This improvement arises because, in our Algorithm 3, only the initial merging step requires $O(N)$ operations; thereafter, the probability vector's dimensionality reduces to $\leq 2m^{\text{syn}}+1$, yielding $O(m^{\text{syn}})$ operations in subsequent steps (see Appendix D.6 for details). In contrast, the runtime of the algorithm in Srinivasan (2001) does not decrease with m^{syn}. In practice, m^{syn}/N is often quite small (e.g., approximately 4.2×10^{-4} in the `MovieLens` dataset), making our algorithm significantly faster than that of Srinivasan (2001).

6. Extension for Improving Scalability

In the previous sections, we presented a projected gradient descent algorithm for learning a relaxed bi-adjacency matrix, along with an unbiased sampling algorithm to convert this

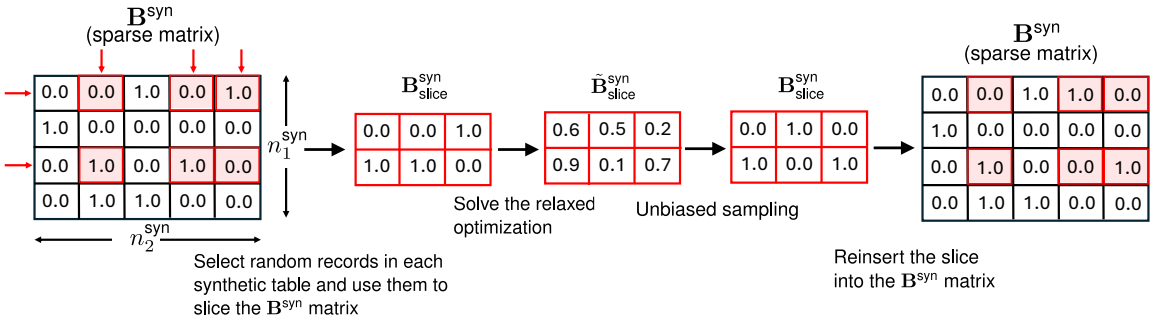

Figure 2: Illustration of how the random slicing heuristic enhances the scalability of the main algorithm. A full version of the main algorithm, incorporating the random slicing heuristic, is provided in Appendix B.

matrix back to discrete space. However, the runtime and memory requirements of both algorithms inherently depend on $n_1^{\text{syn}} \times n_2^{\text{syn}}$, which is the dimension of the bi-adjacency matrix $\boldsymbol{B}^{\text{syn}}$ for the synthetic database. In practice, this dimension can be extremely large. For example, in our experiments with the `MovieLens` dataset (Harper and Konstan, 2015), the user and movie tables contain $6,040$ and $3,900$ records, respectively. To generate a synthetic relational database with the same number of entries, $n_1^{\text{syn}} \times n_2^{\text{syn}}$ would be nearly 24 *million*. This high dimensionality makes it impractical to learn a bi-adjacency matrix that accurately captures the noisy observations derived from real data. To address this challenge, in this section, we introduce a heuristic—random slicing—that can facilitate the synthesis of large-scale relational databases (see Figure 2 for an illustration and Appendix B for its integration into our main algorithm).

The key idea of our heuristic is to explore the sparsity of the bi-adjacency matrix $\boldsymbol{B}^{\text{syn}}$. Specifically, we observe that the non-zero entries of $\boldsymbol{B}^{\text{syn}}$ typically scale as $O(n_1^{\text{syn}} + n_2^{\text{syn}})$. For example, in the `MovieLens` dataset, users tend to rate only a small subset of movies, and in the student-teacher scenario, each student generally enrolls in at most 10 courses, each taught by a different teacher. In such cases, the number of non-zero entries in $\boldsymbol{B}^{\text{syn}}$ (denoted m^{syn}) increases linearly with the size of the user (or student) synthetic table. Based on this, we store $\boldsymbol{B}^{\text{syn}}$ as a sparse matrix and update a subset of its elements at each iteration of Algorithm 1.

Our heuristic works as follows. At each iteration of Algorithm 1, we randomly select a fraction $\tau \in (0, 1)$ of records from each synthetic table and slice $\boldsymbol{B}^{\text{syn}}$ to include only the relationships among the selected records in each table. We denote the sliced bi-adjacency matrix as $\boldsymbol{B}^{\text{syn}}_{\text{slice}}$. We then solve a similar relaxed optimization using the projected gradient descent algorithm (Section 4) to optimize the relationships among these selected records, followed by applying the unbiased sampling algorithm (Section 5) to discretize the sliced bi-adjacency matrix. Finally, we re-insert these newly learned relationships into $\boldsymbol{B}^{\text{syn}}$. This approach reduces the matrix size for the projected gradient descent and the unbiased sampling algorithms from $n_1^{\text{syn}} n_2^{\text{syn}}$ to $\tau^2 n_1^{\text{syn}} n_2^{\text{syn}}$. Since these algorithms have linear time and space complexity w.r.t. the matrix size, their complexity is reduced to $O(\tau^2 n_1^{\text{syn}} n_2^{\text{syn}})$, leading to runtime and memory savings.

7. Utility Guarantees

We present a utility theorem that provides an upper bound on the approximation error of cross-table queries using synthetic relational data generated by Algorithm 1. We begin by defining how to quantify approximation errors for a synthetic relational database $(\mathcal{D}_1^{\text{syn}}, \mathcal{D}_2^{\text{syn}}, \boldsymbol{B}^{\text{syn}})$. Suppose this synthetic database has m^{syn} relationships. We stack the query vectors of interest as a matrix $\hat{\boldsymbol{Q}}$. Lemma 3 shows that the query answers derived from this synthetic database can be expressed as $\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\,\text{vec}(\boldsymbol{B}^{\text{syn}})$. We now introduce the following definition to measure how accurately k-way cross-table queries computed from the synthetic database approximate those obtained from the original data. In this definition, we allow $\boldsymbol{B}^{\text{syn}}$ to be a weighted bi-adjacency matrix.

Definition 10 *Let $(\mathcal{D}_1^{syn}, \mathcal{D}_2^{syn}, \boldsymbol{B}^{syn})$ be a synthetic relational database such that $m^{syn} = \mathbf{1}^T \text{vec}(\boldsymbol{B}^{syn})$. We stack all the k-way cross-table query vectors as $\hat{\boldsymbol{Q}}$. Denote by $\boldsymbol{a}$ the true answers to these queries computed from the original database. We define the (per-workload) mean squared error as*

$$\mathsf{mse}(\mathcal{D}_1^{syn}, \mathcal{D}_2^{syn}, \boldsymbol{B}^{syn}) \triangleq \frac{1}{|\mathcal{W}_{cross,k}|} \cdot \left\| \frac{1}{m^{syn}} \hat{\boldsymbol{Q}}\,\text{vec}(\boldsymbol{B}^{syn}) - \boldsymbol{a} \right\|_2^2, \tag{10}$$

where $|\mathcal{W}_{cross,k}|$ denotes the total number of k-way cross-table marginal workloads.

We now present our utility theorem. It provides an upper bound for the mean squared error in approximating cross-table queries based on outputs from our algorithm. This bound depends on the error associated with the optimal bi-adjacency matrix for the given synthetic tables (i.e., the best achievable error without privacy constraints) and an additional term that approaches zero as the size of the original database increases. The proof techniques are based on classical tools from convex geometry and concentration inequalities.

Theorem 11 *Consider running a single iteration of Algorithm 1 (without unbiased sampling) to establish relationships between any individual synthetic tables $\mathcal{D}_1^{syn}$ and $\mathcal{D}_2^{syn}$. The output is a bi-adjacency matrix $\boldsymbol{B}^{syn}$. Let $\boldsymbol{B}^*$ denote the optimal weighted bi-adjacency matrix in minimizing the mean squared error (Definition 10) when no privacy constraints are enforced. With high probability, we have*

$$\mathsf{mse}(\mathcal{D}_1^{syn}, \mathcal{D}_2^{syn}, \boldsymbol{B}^{syn}) \leq 8 \cdot \mathsf{mse}(\mathcal{D}_1^{syn}, \mathcal{D}_2^{syn}, \boldsymbol{B}^{*}) + \widetilde{O}\left(\frac{d_{max}}{m\sqrt{\rho_{rel}}}\right),$$

where $\rho_{rel} = \left(\sqrt{\epsilon_{rel} + \log\frac{1}{\delta_{rel}}} - \sqrt{\log\frac{1}{\delta_{rel}}}\right)^2$, m is the number of relationships in the original database, and $\widetilde{O}(\cdot)$ ignores logarithmic factors.

Proof For a more rigorous statement of the theorem and proof, see Appendix D.8. ■

The theorem above establishes an upper bound on the (per-workload) squared error when estimating cross-table queries using synthetic data generated from Algorithm 1. This upper bound consists of the optimal error without privacy constraints plus an additional error term. The optimal error only hinges on the quality of the individual synthetic tables,

determined by the privacy mechanism used for generating these tables. In a hypothetical scenario where the synthetic tables are identical to the original tables, this term would be reduced to zero. On the other hand, the additional error term depends on three factors: m (the number of relationships in the original database), $d_{\max}$ (the maximum number of relationships a record in the original database can have), and ρ_{rel} (privacy parameter). With fixed $d_{\max}$ and ρ_{rel}, as the number of relationships increases, the additional error term can be reduced to zero. Finally, we note that existing utility theorems (e.g., Liu et al., 2023b; Aydore et al., 2021) often rely on the size of data domain $|\mathcal{X}_i|$—that is, the number of distinct values for each categorical feature. This quantity could be large, particularly if data are discretized into numerous bins. We eliminate this dependence in our utility theorem by leveraging the "marginal trick" (see Appendix D.4 in Liu et al., 2021b) in our algorithm.

Algorithm 11 is focused on $T = 1$, Extending the analysis to $T > 1$ would require tracking how adaptively chosen workloads, noisy measurements, and our constrained optimization interact across rounds under referential-integrity constraints, which introduces additional dependencies that are technically nontrivial to control.

8. Numerical Experiments

We demonstrate our algorithm through numerical experiments on real-world datasets.

Data. We consider the `MovieLens` dataset (Harper and Konstan, 2015) and the `IPUMS` dataset (Ruggles et al., 2021). The `MovieLens` dataset consists of three tables: users, movies, and ratings. The user table contains 6,040 entries, detailing demographic information like gender, age, and occupation. The movie table contains 3,883 entries, each tagged with associated genres. We convert the genres into 18 binary features—this is not one-hot encoding, as each movie can belong to multiple genres. The ratings table links users and movies in a many-to-many structure: users can rate multiple movies, and each movie can have multiple ratings. We pre-process the rating table to ensure that each movie is rated by no more than 10 users, and each user rates no more than 10 movies (i.e., $d_{\max} = 10$). After this pre-processing, the rating table contains 10,075 rows.

The `IPUMS` dataset contains anonymized U.S. census data. For our experiment, we focus on the user's table, which provides detailed individual-level data, including demographic, educational, and employment attributes containing 3,373,378 entries. To expedite the training, we only consider the first 5% (168,669 rows) of the records. To establish a many-to-many relationship within this dataset, we use parental links, as the dataset includes the person IDs of each individual's parents. An individual in the `IPUMS` dataset may have up to four recorded parents—two mothers and two fathers—leading us to set $d_{\max} = 4$. This design creates a self-referential structure within the users table, as the relational connections are entirely within the same dataset table. For the `IPUMS` dataset, we generate a synthetic users table of 10,000 records.

Setup. We use state-of-the-art DP mechanisms, `AIM` (McKenna et al., 2022) and `MST` (McKenna et al., 2021), to generate individual synthetic tables. Each synthetic table is created with a privacy budget of $\epsilon = 1$. We use the OpenDP library (SmartNoise, 2023) to implement these DP mechanisms. To build relationships within the synthetic tables,

we apply Algorithm 1 under varying privacy budgets. We evaluate its performance by measuring the average error in 3-way cross-table marginal queries (Definition 2) between synthetic and real data.

In our ablation study, we vary several parameters. By default, our settings are as follows: the algorithm iterates 15 times, using randomized slicing to create sub-tables of 1,000 synthetic records by 1,000 synthetic records. During each iteration, 3 slices are generated for IPUMS, while 8 are generated in MovieLens, with each slice containing at least 200 rows and 200 columns (20% of the slice) that include relational data to ensure a sufficient number of relationships in each subset. Within each iteration, the exponential mechanism, with a weight of $\alpha = 0.2$, selects 3 workloads for evaluation. From the queries selected by the exponential mechanism, we choose 8 of them with the highest error for updating each slice using projected gradient descent.

Computation Setting. All experiments were conducted on a single NVIDIA V100 GPU with 16 GB of memory. Our implementation is in PyTorch and utilizes GPU acceleration for key steps (e.g., solving the relaxed optimization), which significantly speeds up computation.

Results. We demonstrate the performance of our algorithm across various datasets and DP mechanisms in the figures below. Additionally, we conduct an ablation study to assess the impact of different hyper-parameter selections on our results.

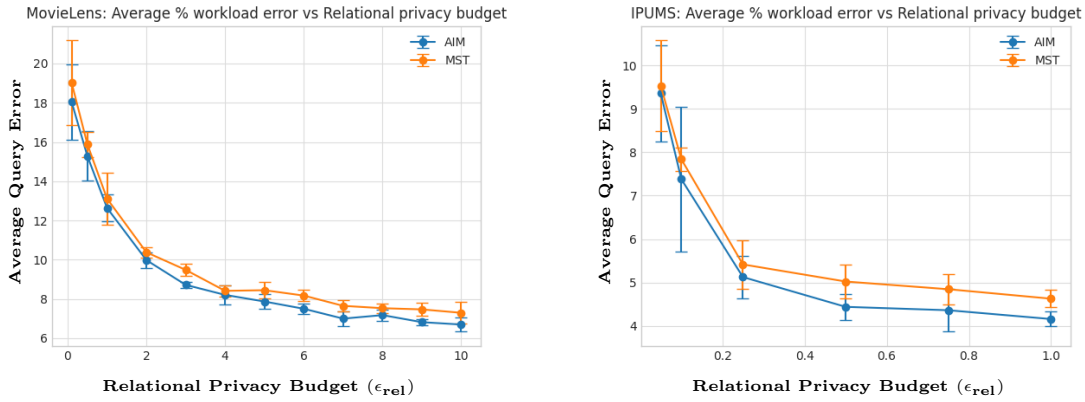

Figure 3: We use `AIM` and `MST` to produce individual synthetic tables and then apply our algorithm with privacy budget ϵ_{rel} to establish their relationships. We illustrate the impact of varying ϵ_{rel} on the average 3-way marginal query error for the `MovieLens` dataset (Left) and the `IPUMS` dataset (Right). As expected, the average error decreases as ϵ_{rel} increases, due to reduced noise.

- **Privacy budget.** Recall that ϵ_{rel} is the privacy budget allocated to Algorithm 1 for establishing relationships among synthetic tables. In Figure 3, we illustrate the effect of varying ϵ_{rel} on the average 3-way marginal query error. As ϵ_{rel} increases, the average error decreases due to reduced noise in both the selection (exponential mechanism) and observation (Gaussian mechanism) phases. This trend is clearly observed in results for

both `AIM` and `MST`. Furthermore, higher ϵ_{rel} values lead to a reduction not only in the average error but also in its standard deviation, resulting from the lower noise levels. Finally, with a small privacy budget (e.g., $\epsilon = 1$ for generating each synthetic table and $\epsilon_{\text{rel}} = 2.0$ for establishing relationships), our algorithm is capable of producing a synthetic relational database with high fidelity.

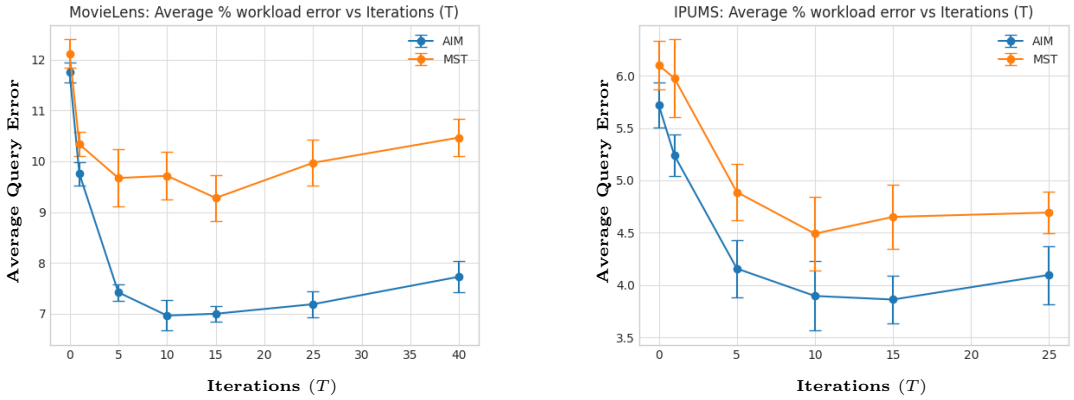

Figure 4: We analyze the impact of varying the number of iterations T in Algorithm 1 on the quality of synthetic data. The figures show a U-shaped curve between average error and T. This trend highlights a trade-off: increasing T allows for more workload queries being selected but their answers, computed from real data, become increasingly noisy.

- **Number of iterations.** We analyze how varying the number of iterations T in Algorithm 1 influences synthetic data quality, specifically in terms of the average 3-way marginal query error. Figure 4 shows a U-shaped curve between average error and T. This effect arises because the privacy budget ϵ_{rel} is evenly distributed across T iterations used by the Gaussian and exponential mechanisms. Hence, increasing T allows more workloads to be selected and optimized for synthetic data with the cost of adding higher noise to the workload query answers computed from the real data. Conversely, with very few iterations, the algorithm has limited chances to explore sufficient workload queries, leading to higher average error. This trade-off underpins the U-shaped curve, identifying an optimal range between $T = 10$ and $T = 15$ where the error is minimized.

- **Privacy budget allocation.** In Algorithm 4, a detailed version of our algorithm, we introduce a hyperparameter, $\alpha \in [0, 1]$, to divide the privacy budget between the Gaussian and exponential mechanisms. When $\alpha = 1.0$, the entire privacy budget is allocated to the exponential mechanism, enabling it to select the most relevant workloads. However, with no privacy budget assigned to the Gaussian mechanism, it outputs random values for the selected workload queries. Conversely, setting $\alpha = 0.0$ dedicates the entire privacy budget to the Gaussian mechanism, producing low-noise query values derived from real

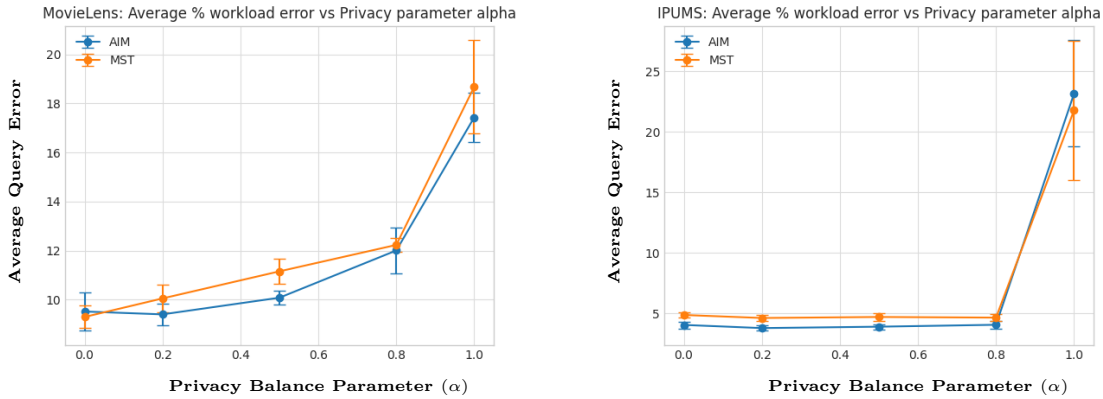

Figure 5: We show the effect of the hyperparameter $\alpha \in [0, 1]$ on the quality of synthetic data produced by our algorithm. This parameter allocates the privacy budget between the Gaussian and exponential mechanisms.

data for randomly selected workloads. Figure 5 illustrates the impact of varying α on the quality of the synthetic data. Based on this result, we select $\alpha = 0.2$ for the rest of our ablation study.

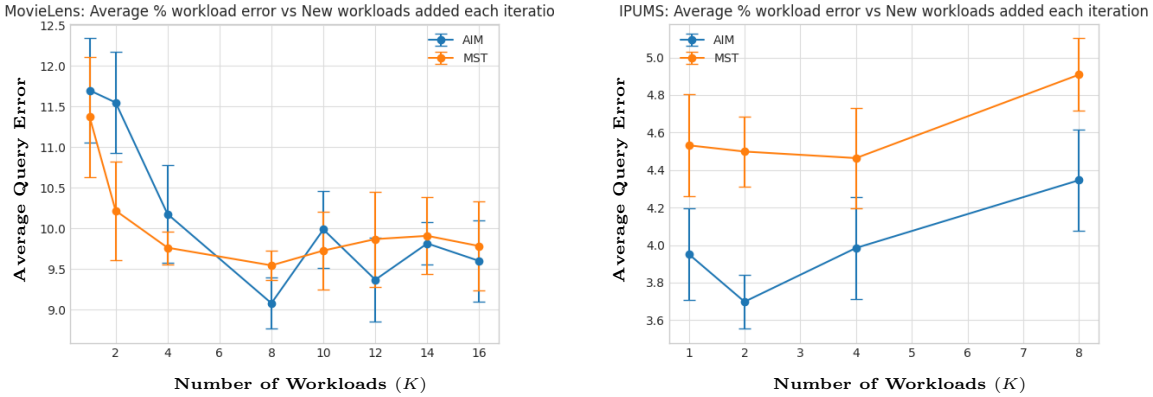

Figure 6: We demonstrate the effect of the number of workloads K selected by the exponential mechanism on the quality of synthetic data produced by our algorithm.

- **Number of workloads selected by the exponential mechanism.** The exponential mechanism in Algorithm 1 selects K workloads per iteration. With the privacy budget distributed evenly across these selected workloads, a larger K increases the noise added to their query answers computed from real data, while a smaller K may yield an insufficiently diverse set of observed workloads. Figure 6 illustrates the impact of different values of K on the average 3-way marginal error. We recommend using $K = 4$ by default, which

we found to work well across all settings, and only increasing K if one suspects that the workload space is very heterogeneous.

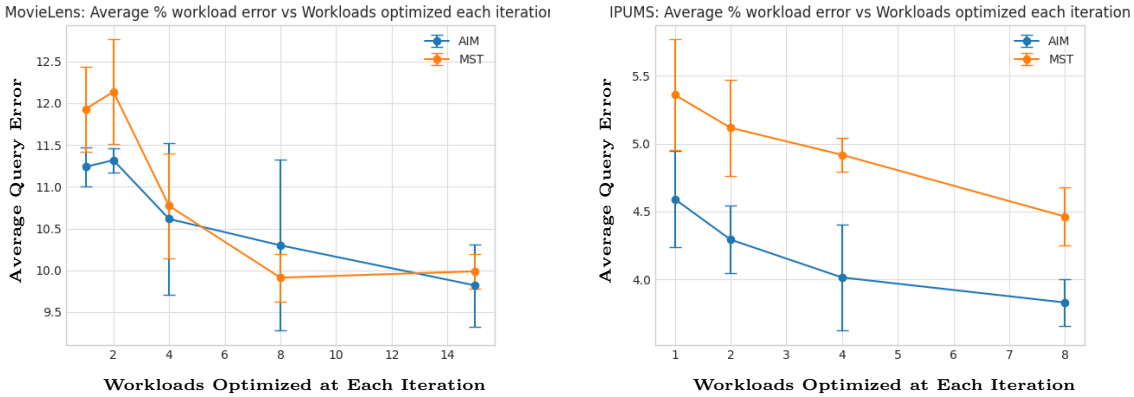

Figure 7: We demonstrate the effect of the number of workloads optimized by the relaxed optimization. As more workloads are included, the average 3-way marginal query error reduces.

- **Number of workloads optimized by the relaxed optimization.** To enhance the speed of our algorithm, we select a fixed number of workloads with the highest error each time we solve the relaxed optimization, choosing from all the workloads identified by the exponential mechanism so far. This selection strategy prioritizes computational efficiency, so we anticipate that including more workloads per step should reduce the average 3-way marginal query error. Figure 7 validates this hypothesis, showing a decline in error as the number of workloads grows.

9. Conclusion, Future Work, and Impact Statement

In this paper, we investigated synthetic relational database generation with DP guarantees. We proposed an algorithm that can be combined with any preexisting single-table generation mechanisms to maintain cross-table statistical properties and referential integrity. This algorithm creates relationships between individual synthetic tables by learning a bi-adjacency matrix, which is iteratively updated through solving a quadratic program and sampling an unbiased instance to minimize approximation errors of k-way marginals. We derived rigorous utility guarantees for our algorithm and provided a PyTorch implementation suited for high-dimensional and high-volume data. Furthermore, we conducted comprehensive experiments and introduced new benchmark datasets and evaluation metrics to assess the performance and scalability of our algorithm. We hope our effort can inspire new research and push the frontiers of DP synthetic data towards more practical scenarios.

There are several intriguing directions that deserve further exploration. First, we assumed any record in the relational database must be kept private. It would be interesting to explore scenarios in which only specific tables contain personal private information. For

example, in the context of education data, it may be pertinent to examine situations where only teacher and student information requires privacy-preserving, while course and department information, being often publicly available, does not. Second, our algorithm generates individual synthetic tables and uses an iterative algorithm to establish their relationship; one extension would be integrating synthetic table generation into the iterative algorithm, learning both single-table and cross-table queries simultaneously. However, achieving this may require white-box access to the generative models. Finally, while we generate synthetic relational databases with the goal of preserving low-order marginal distributions, there are other criteria worthy of exploration, such as logical consistency, temporal dynamics, and user-defined constraints. It would be valuable to understand whether these criteria can be incorporated into synthetic relational data generation while maintaining DP guarantees.

This paper focuses on privacy-preserving synthetic data generation, building upon prior efforts in DP synthetic data. We extend their application to a more practical scenario where data is stored in a relational database. This line of research may significantly impact several critical domains, such as finance, healthcare, and government, where safeguarding data privacy is paramount. The deployment of DP synthetic data can lead to more inclusive research practices, allowing organizations to share data without compromising the privacy of individuals' personal information. This, in turn, facilitates collaboration and propels advancements in research and business applications reliant on data-driven insights.

It is crucial to acknowledge potential challenges and ethical considerations associated with the deployment of synthetic data. Synthetic data proves to be a suitable substitute for tasks that do not necessitate absolute precision, such as data visualization, software testing, and initial model development. Nevertheless, there is often a trade-off between preserving the essential statistical properties of the original data and meeting privacy requirements. For applications with individual-level consequences, any methods or insights derived from synthetic data need to be carefully evaluated to avoid unintended consequences and biases in downstream applications. In summary, while our research advances the field of DP synthetic data generation, we recognize the importance of ethical considerations and the potential societal impact. By mitigating privacy concerns in data sharing, this work aspires to play a pivotal role in fostering the responsible and reliable development of advanced machine learning technologies.

Appendix A. Background on Differential Privacy (DP)

We recall the concept of (zero) concentrated differential privacy (Bun and Steinke, 2016; Steinke, 2022). It will be used in the proof of Theorem 4 for establishing DP guarantees for our algorithm.

Definition 12 *(Zero concentrated differential privacy (zCDP)). A randomized mechanism $\mathcal{M}$ satisfies ρ-zero Concentrated Differential Privacy (zCDP) if for all pairs of neighboring relational databases $\mathcal{B}, \mathcal{B}'$, and for all $\alpha \in (1, \infty)$:*

$$\mathbb{D}_\alpha \left(\mathcal{M}(\mathcal{B}), \mathcal{M}\left(\mathcal{B}\right) \right) \leq \rho\alpha,$$

where $\mathbb{D}_\alpha \left(\mathcal{M}(\mathcal{B}), \mathcal{M}\left(\mathcal{B}'\right) \right)$ denotes the α-Renyi divergence between the distributions of $\mathcal{M}(\mathcal{B})$ and $\mathcal{M}\left(\mathcal{B}'\right)$.

zCDP has nice composition and post-processing properties.

Lemma 13 *(Adaptive composition). Let $\mathcal{M}_1, \cdots, \mathcal{M}_T$ be $\rho_1, \cdots, \rho_T$-zCDP, respectively. Suppose $\mathcal{M}$ is an adaptive composition of $\mathcal{M}_1, \cdots, \mathcal{M}_T$. Then $\mathcal{M}$ satisfies $(\sum \rho_i)$-zCDP.*

Lemma 14 *(Post-processing). Let $\mathcal{M}$ be a ρ-zCDP mechanism. Suppose f is any (possibly randomized) function. Then $\mathcal{M}'(\mathcal{B}) = f(\mathcal{M}(\mathcal{B}))$ satisfies ρ-zCDP.*

Lemma 15 *(Relation between zCDP and DP). If $\mathcal{M}$ satisfies ρ-zCDP, then it also satisfies $(\rho + 2\sqrt{\rho \log(1/\delta)}, \delta)$-DP for any $\delta > 0$.*

Next, we recall two basic privacy mechanisms: Gaussian and exponential mechanisms. To start with, we define the L_p sensitivity of a function f as

$$\Delta_p(f) \triangleq \sup_{\mathcal{B} \sim \mathcal{B}'} \|f(\mathcal{B}) - f(\mathcal{B}')\|_p, \tag{11}$$

where $\mathcal{B} \sim \mathcal{B}'$ are two neighboring databases.

Lemma 16 *(Gaussian Mechanism). Given a function f, the Gaussian mechanism is defined as $\mathcal{M}(\mathcal{B}) = f(\mathcal{B}) + \sigma\Delta_2(f)N$ where N is a random vector drawn from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Then $\mathcal{M}$ satisfies $\frac{1}{2\sigma^2}$-zCDP.*

Finally, we recall the exponential mechanism. For a score function and a set $\mathcal{O}$, we define its sensitivity by

$$\Delta(\mathsf{score}) \triangleq \sup_{\mathcal{B} \sim \mathcal{B}'} \max_{o \in \mathcal{O}} |\mathsf{score}(\mathcal{B}, o) - \mathsf{score}(\mathcal{B}', o)|. \tag{12}$$

Lemma 17 *(Exponential Mechanism). Given a score function and the set $\mathcal{O}$ of all possible outputs, the exponential mechanism $\mathcal{M}$ is a randomized mechanism defined by*

$$\mathbb{P}(\mathcal{M}(\mathcal{B}) = o) \propto \exp\left(\frac{\epsilon}{2\Delta(\mathsf{score})} \mathsf{score}(\mathcal{B}, o) \right), \quad \forall o \in \mathcal{O}. \tag{13}$$

Then $\mathcal{M}$ satisfies $\frac{\epsilon^2}{8}$-zCDP.

Appendix B. More Details about Algorithm 1

In this section, we outline the implementation details of our main algorithm (Algorithm 1). The full version is presented in Algorithm 4 and illustrated in Figure 2. Recall that our algorithm selects worst-case workloads using the exponential mechanism and computes noisy observations of these workloads from real data using the Gaussian mechanism. Details on these mechanisms can be found in Algorithms 5 and 6, along with their respective privacy budget allocations. Additionally, we incorporate the random slicing mechanism (Section 6) to enhance scalability for generating large-scale synthetic relational databases.

To implement the random slicing mechanism, we begin by storing the bi-adjacency matrix as a sparse matrix $\boldsymbol{B}^{\text{syn}} \in \{0,1\}^{n_1^{\text{syn}}} \times \{0,1\}^{n_2^{\text{syn}}}$. We introduce a parameter $\tau < 1$, which specifies the fraction of relations to be learned in each iteration. Next, we define the slicing operation, which will be used to extract elements from $\boldsymbol{B}^{\text{syn}}$ before running the projected gradient descent and unbiased sampling algorithms.

Definition 18 *Given a matrix $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$ and vectors $\boldsymbol{v}_1 \in \{0,1\}^{n_1}$, $\boldsymbol{v}_2 \in \{0,1\}^{n_2}$, we define the slicing operation as $\left[\boldsymbol{v}^1\right] \boldsymbol{M} \left[\boldsymbol{v}^2\right]$. This operation yields a matrix with $\mathbf{1}^T \boldsymbol{v}_1^{slice}$ rows and $\mathbf{1}^T \boldsymbol{v}_2^{slice}$ columns. The (i,j) entry of this matrix corresponds to the (v_i^1, v_j^2) entry of $\boldsymbol{M}$, where v_i^1 is defined as the i-th nonzero entry of $\boldsymbol{v}^1$ (similarly for v_j^2). If only the columns are sliced, we write $\boldsymbol{M}\left[\boldsymbol{v}^2\right]$.*

Algorithm 4 Full Algorithm 1 (detailed version with scalability enhancements).

Input: private relational database $\mathcal{B} = (\mathcal{D}_1, \mathcal{D}_2, \boldsymbol{B})$; privacy budget $(\epsilon_{\text{rel}}, \delta_{\text{rel}})$; individual synthetic tables $\mathcal{D}_1^{\text{syn}}$, $\mathcal{D}_2^{\text{syn}}$; workloads per iteration K; number of iterations T; number of relationship in synthetic tables m^{syn}; slicing fraction τ; hyper-parameter balancing Gaussian and exponential mechanisms α

Initialize: $\hat{\boldsymbol{Q}} = \emptyset, \hat{\boldsymbol{a}} = \emptyset$, n_1^{syn} by n_2^{syn} sparse matrix $\boldsymbol{B}^{\text{syn}}$ with entries $\in \{0,1\}$

Convert $(\epsilon_{\text{rel}}, \delta_{\text{rel}})$-DP into ρ_{rel}-zCDP and compute a privacy parameter $\epsilon_0 = \sqrt{\frac{2\rho_{\text{rel}}}{KT}}$

for $t = 1, \cdots, T$ **do**

Select K workloads via the exponential mechanism ▷ Algorithm 5

Append all queries associated with the newly selected workloads into $\hat{\boldsymbol{Q}}$

Add noise to the answers of the new queries and append them into $\hat{\boldsymbol{a}}$ ▷ Algorithm 6

Randomly choose $\boldsymbol{v}_1^{\text{slice}} \in \{0,1\}^{n_1^{\text{syn}}}$, $\boldsymbol{v}_2^{\text{slice}} \in \{0,1\}^{n_2^{\text{syn}}}$ s.t. $\mathsf{sum}(\boldsymbol{v}_i^{\text{slice}}) = \lfloor \tau n_i^{\text{syn}} \rfloor$ for $i \in \{1,2\}$

Set $\boldsymbol{B}_{\text{slice}}^{\text{syn}} = [\boldsymbol{v}_1^{\text{slice}}] \boldsymbol{B}^{\text{syn}} [\boldsymbol{v}_2^{\text{slice}}]$ and $\boldsymbol{Q}_{\text{slice}} = \boldsymbol{Q}[\text{vec}(\boldsymbol{v}_1^{\text{slice}} \cdot (\boldsymbol{v}_2^{\text{slice}}.\text{transpose}))]$ ▷ Definition 18

Compute $m_{\text{slice}} = \mathsf{sum}(\boldsymbol{B}_{\text{slice}}^{\text{syn}})$

Solve the relaxed optimization: ▷ Algorithm 2

$$\widetilde{\boldsymbol{B}}_{\text{slice}}^{\text{syn}} = \underset{\substack{B_{i,j} \in [0,1] \\ \mathsf{sum}(\boldsymbol{B}_{\text{slice}}^{\text{syn}}) = m_{\text{slice}}}}{\text{argmin}} \left\| \frac{1}{m_{\text{slice}}} \boldsymbol{Q}_{\text{slice}} \mathsf{flatten}(\boldsymbol{B}_{\text{slice}}^{\text{syn}}) - \hat{\boldsymbol{a}} \right\|_2^2$$

$\boldsymbol{B}_{\text{slice}}^{\text{syn}} = \text{UnbiasedSampling}(\mathsf{flatten}(\widetilde{\boldsymbol{B}}_{\text{slice}}^{\text{syn}}), m_{\text{slice}})$ ▷ Algorithm 3

$[\boldsymbol{v}_1^{\text{slice}}] \boldsymbol{B}^{\text{syn}} [\boldsymbol{v}_2^{\text{slice}}] = \boldsymbol{B}_{\text{slice}}^{\text{syn}}$ ▷ Re-insert $\boldsymbol{B}_{\text{slice}}^{\text{syn}}$ into $\boldsymbol{B}^{\text{syn}}$

end for

Output: $\mathcal{B}^{\text{syn}} = (\mathcal{D}_1^{\text{syn}}, \mathcal{D}_2^{\text{syn}}, \boldsymbol{B}^{\text{syn}})$

Algorithm 5 Exponential mechanism for workload selection.

Input: original private relational database $\mathcal{B} = (\mathcal{D}_1, \mathcal{D}_2, \boldsymbol{B})$; synthetic relational database $\mathcal{B}^{\text{syn}} = (\mathcal{D}_1^{\text{syn}}, \mathcal{D}_2^{\text{syn}}, \boldsymbol{B}^{\text{syn}})$; workloads per iteration K; all k-way cross-table workloads $\{(\mathcal{S}_1^1, \mathcal{S}_2^1), \cdots, (\mathcal{S}_1^W, \mathcal{S}_2^W)\}$; workloads that have already been selected $\mathcal{T}_{\text{selected}}$; hyper-parameter balancing Gaussian and exponential mechanisms α; privacy parameter ϵ_0
Initialize: $\mathcal{O} = \emptyset$ and $\mathcal{W} = \mathsf{Subsample}\left(\{(\mathcal{S}_1^1, \mathcal{S}_2^1), \cdots, (\mathcal{S}_1^W, \mathcal{S}_2^W)\} \backslash \mathcal{T}_{\text{selected}}\right)$ ▷ Save runtime
for $w \in \mathcal{W}$ **do**
  $\mathsf{score}(w) = \frac{1}{2}\|P_w(\mathcal{B}) - P_w(\mathcal{B}^{\text{syn}})\|_1$ ▷ total variance distance between marginal distributions of $\mathcal{B}$ and $\mathcal{B}^{\text{syn}}$ on workload w
end for
for $t = 1, \cdots, K$ **do**
  $\boldsymbol{p} = [\mathsf{score}(w) \text{ for } w \in \mathcal{W}]$ ▷ Scores of workloads that have not been selected
  $\boldsymbol{p} = \mathsf{softmax}\left(\sqrt{\alpha}\epsilon_0 \frac{m}{d_{\max}} \boldsymbol{p}\right)$ ▷ $m = \mathbf{1}^T \cdot \text{vec}(\boldsymbol{B})$ and $d_{\max}$ is defined in Section 2.1
  Sample a workload w according to the probability vector $\boldsymbol{p}$
  $\mathcal{O} = \mathcal{O} \cup w$ and $\mathcal{W} = \mathcal{W} \backslash \{w\}$
end for
$\mathcal{T}_{\text{selected}} = \mathcal{T}_{\text{selected}} \cup \mathcal{O}$
Output: $\mathcal{O}$

Algorithm 6 Gaussian mechanism for noisy answer computation.

Input: (empirical) marginal distributions of $\mathcal{B}$ on K newly selected workloads $\{\boldsymbol{p}_1, \cdots, \boldsymbol{p}_K\}$; hyper-parameter balancing Gaussian and exponential mechanisms α; privacy parameter ϵ_0
for $t = 1, \cdots, K$ **do**
  Assert $\text{sum}(\boldsymbol{p}_t) == 1$ and $\boldsymbol{p}_t \geq 0$
  $\hat{\boldsymbol{a}}_t = \boldsymbol{p}_t + \left(\frac{\sqrt{2}d_{\max}}{m\sqrt{(1-\alpha)\epsilon_0}}\right)\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ ▷ Add Gaussian noise to each probability vector
end for
Output: $\{\hat{\boldsymbol{a}}_1, \cdots, \hat{\boldsymbol{a}}_K\}$

Appendix C. One-to-Many Relationship

In the main body of the paper, we focused on many-to-many relationship. In some cases, the original database may exhibit a one-to-many relationship. For instance, in the context of education data (e.g., student and school data), each school can have multiple students, but each student belongs to only one school. When generating a synthetic database, it is crucial to preserve this relationship. In this section, we apply our algorithm to this scenario. As shown, by incorporating this additional assumption—that records in one table have exactly one relationship with the other table—our algorithm becomes significantly simpler and operates much more efficiently.

Consider that each record in the first table of the original database has only one corresponding relationship (e.g., the student table from the example above). We denote the rows of the bi-adjacency matrix $\boldsymbol{B}^{\text{syn}}$ as $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_{n_1^{\text{syn}}}$. We introduce the constraint $\mathbf{1}^T\boldsymbol{b}_i = 1$ for all $i \in [n_1^{\text{syn}}]$. After applying the unbiased sampling algorithm, this constraint will translate into the condition that each record in the first synthetic table has exactly one relationship with a record in the second table. Finally, we denote the concatenation of $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_{n_1^{\text{syn}}}$ by

$\boldsymbol{b}$ (i.e., $\boldsymbol{b}$ is the vectorization of $\boldsymbol{B}^{\text{syn}}$). We solve the following relaxed optimization:

$$\min_{\boldsymbol{b}\in\Omega} f(\boldsymbol{b}) \quad \text{where} \quad f(\boldsymbol{b}) \triangleq \left\| \frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}}\boldsymbol{b} - \hat{\boldsymbol{a}} \right\|_2^2 \quad \text{and} \quad \Omega \triangleq \{\boldsymbol{b} \mid 0 \le \boldsymbol{b} \le 1 \text{ and } \mathbf{1}^T \boldsymbol{b}_i = 1 \ \forall i \in [n_1^{\text{syn}}]\}. \tag{14}$$

Note that the constraint set Ω can be reformulated for each individual vector $\boldsymbol{b}_i$:

$$\Omega_i \triangleq \{\boldsymbol{b}_i \mid 0 \le \boldsymbol{b}_i \le 1, \mathbf{1}^T \boldsymbol{b}_i = 1\}.$$

Therefore, we can still apply the projected gradient descent algorithm to solve the relaxed optimization problem in (14). In each iteration, after computing the gradient (8) based on the last iterate, we update the solution in the direction opposite to the gradient. Instead of projecting the entire vector $\boldsymbol{b}$ onto Ω, we project each sub-vector $\boldsymbol{b}_i$ onto its respective constraint set Ω_i. This can be achieved by applying the same closed-form expression from Theorem 6 to each $\boldsymbol{b}_i$. The projection step maintains the same time complexity but offers the added advantage of allowing parallel projection of the sub-vectors onto their respective constraint sets. Lastly, since the objective function remains unchanged, the convergence guarantee and learning rate are preserved.

Our projected gradient descent algorithm ensures that each row of $\widetilde{\boldsymbol{B}}^{\text{syn}}$ is a valid probability vector. Hence, we can directly sample from the categorical distribution parameterized by each row. We introduce Algorithm 7 that samples from $\widetilde{\boldsymbol{B}}^{\text{syn}}$ while ensuring the one-to-many relationship. This approach provides an unbiased sample and has the added benefit of enabling parallel sampling for each record in the first table.

Algorithm 7 Categorical rounding for one-to-many relationship.

Input: relaxed bi-adjacency matrix $\widetilde{\boldsymbol{B}}^{\text{syn}} \in [0,1]^{n_1^{\text{syn}}} \times [0,1]^{n_2^{\text{syn}}}$
Assert sum$(\widetilde{\boldsymbol{B}}_{i,:}^{\text{syn}}) == 1$ for $i \in \{1, \cdots, n_1^{\text{syn}}\}$
$\boldsymbol{B}^{\text{syn}} = \mathsf{zeros}(n_1^{\text{syn}}, n_2^{\text{syn}})$
for $i = 1, \cdots, n_1^{\text{syn}}$ **do**
  Draw k from $\{1, \cdots, n_2^{\text{syn}}\}$ according to the probability distribution $\widetilde{\boldsymbol{B}}_{i,:}^{\text{syn}}$
  $B_{i,k}^{\text{syn}} = 1$
end for
Output: unweighted bi-adjacency matrix $\boldsymbol{B}^{\text{syn}}$

In short, generating synthetic data becomes significantly easier when the original relational database only contains one-to-many relationships. In our case, both the projected gradient descent algorithm and the unbiased sampling algorithm are simplified, enabling more efficient execution through parallel computation.

Appendix D. Omitted Proofs

We provide omitted proofs in this section.

D.1 Proof of Lemma 3

Proof Recall from Section 2.2 that for any k-way cross-table query $\boldsymbol{q}_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}$, its average over a relational database $(\mathcal{D}_1, \mathcal{D}_2, \boldsymbol{B})$ can be computed as

$$\boldsymbol{q}_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}(\mathcal{B}) = \frac{1}{\mathbf{1}^T \cdot \mathrm{vec}(\boldsymbol{B})} \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \boldsymbol{q}_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}(\boldsymbol{x}_i, \boldsymbol{x}_j) B_{i,j} \tag{15}$$

where $\boldsymbol{x}_i \in \mathcal{D}_1$ and $\boldsymbol{x}_j \in \mathcal{D}_2$. We define $\boldsymbol{Q}_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)} \triangleq \mathbf{1}_{(\mathcal{S}_1,\boldsymbol{y}_1)} \cdot \mathbf{1}^T_{(\mathcal{S}_2,\boldsymbol{y}_2)} \in \mathbb{R}^{n_1 \times n_2}$ with its i-th row and j-th column denoted by $\left[\boldsymbol{Q}_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}\right]_{i,j}$. By the definition of $\mathbf{1}_{(\mathcal{S}_1,\boldsymbol{y}_1)}$ and $\mathbf{1}_{(\mathcal{S}_2,\boldsymbol{y}_2)}$, we have $\left[\boldsymbol{Q}_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}\right]_{i,j} = \boldsymbol{q}_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Therefore,

$$\begin{aligned}
\mathrm{vec}\left(\boldsymbol{Q}_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}\right)^T \cdot \mathrm{vec}(\boldsymbol{B}) &= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} [\boldsymbol{Q}_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}]_{i,j} B_{i,j} \\
&= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \boldsymbol{q}_{(\mathcal{S}_1,\mathcal{S}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)}(\boldsymbol{x}_i, \boldsymbol{x}_j) B_{i,j}.
\end{aligned}$$

Combined this with (15) yields the desired result. ■

D.2 Proof of Theorem 4

We recall Theorem 4 below and provide additional details along with the proof.

Theorem 19 *Assume that each record in the original relational database has at most $d_{\max}$ relationships with records in the other table. We apply any DP mechanism with privacy budgets (ϵ_1, δ_1) and (ϵ_2, δ_2) to generate individual synthetic tables, respectively. Then we apply Algorithm 1 with privacy budget $(\epsilon_{rel}, \delta_{rel})$ to build their relationship. The synthetic relational database satisfies $(\epsilon_1 + \epsilon_2 + \epsilon_{rel}, \delta_1 + \delta_2 + \delta_{rel})$-DP, where $\epsilon_{rel}(\delta_{rel}) \leq \rho_{rel} + 2\sqrt{\rho_{rel}\log(1/\delta_{rel})}$. This condition can be achieved by choosing:*

$$\rho_{rel} = \left(\sqrt{\log(1/\delta_{rel}) + \epsilon_{rel}} - \sqrt{\log(1/\delta_{rel})}\right)^2$$

Proof At each iteration, Algorithm 1 makes K call to the exponential mechanism and K call to the Gaussian mechanism. The exponential mechanism uses the score function:

$$\mathsf{score}(w) = \frac{1}{2}\|P_w(\mathcal{B}) - P_w(\mathcal{B}^{\mathrm{syn}})\|_1. \tag{16}$$

Recall that we consider two relational databases $\mathcal{B}$ and $\mathcal{B}'$ adjacent if $\mathcal{B}'$ can be obtained from $\mathcal{B}$ by selecting a table, modifying the values of a single row within this table, and changing all relationship associated with this row. We focus on the setting of bounded DP so both $\mathcal{B}$ and $\mathcal{B}'$ have m relationships. Additionally, we assume each record can have at most $d_{\max}$ relationships with records in the other table. Hence, for any workload w, the

total variance distance between the empirical distributions of $\mathcal{B}$ and $\mathcal{B}'$ on w has an upper bound:

$$\frac{1}{2}\|P_w(\mathcal{B}) - P_w(\mathcal{B}')\|_1 \leq \frac{d_{\max}}{m}. \tag{17}$$

By the triangle inequality, we can upper bounded the sensitivity of score by $\frac{d_{\max}}{m}$. Similarly, for each workload w, we can upper bound the L_2 sensitivity of each empirical distribution $P_w(\mathcal{B})$ by $\frac{\sqrt{2}d_{\max}}{m}$. By Lemmas 16 and 17, at each iteration, the exponential and Gaussian mechanisms satisfy $\frac{K}{2}\alpha \cdot \epsilon_0^2$-zCDP and $\frac{K}{2}(1-\alpha) \cdot \epsilon_0^2$-zCDP, respectively. The composition theorem in Lemma 13 ensures that Algorithm 1 satisfies $\frac{KT(\alpha+(1-\alpha))}{2}\epsilon_0^2$-zCDP. Recall that we chose $\epsilon_0 = \sqrt{\frac{2\rho_{\text{rel}}}{KT}}$. Hence, Algorithm 1 satisfies ρ_{rel}-zCDP. As a result, Lemma 15 yields that it also satisfies $(\rho_{\text{rel}} + 2\sqrt{\rho_{\text{rel}}\log(1/\delta)}, \delta)$-DP. In particular, if we choose $\rho_{\text{rel}} = \left(\sqrt{\log(1/\delta_{\text{rel}}) + \epsilon_{\text{rel}}} - \sqrt{\log(1/\delta_{\text{rel}})}\right)^2$, then Algorithm 1 will satisfy $(\epsilon_{\text{rel}}, \delta_{\text{rel}})$-DP. ■

D.3 Proof of Theorem 5

Proof Algorithm 1 has different steps that we analyze here. First, the exponential mechanism, fixing the family of workloads we are considering to k-way marginals for a fixed k, we have $C_{\mathcal{W}} = \binom{d_1+d_2}{k} - \binom{d_1}{k} - \binom{d_2}{k}$ cross-table workloads, which is constant in N and T, we calculate their corresponding vector once, and have to calculate the inner product of that vector with the relational database adjaceny matrix at iteration $T = t$, requiring linear time in N per workload, for constant number of workloads. After that selection requires sampling K workloads from $C_{\mathcal{W}}$ that does not depend on N. When applying the Gaussian mechanism, we are adding noise independently to each of K new selected workloads, which has constant complexity in terms of N. Further Theorem 7 and Theorem 8 ensure that optimization and discretization steps detailed in Algorithm 1 have linear complexity in $N = n_1^{\text{syn}} \cdot n_2^{\text{syn}}$. Therefore, each iteration in Algorithm 1 has time complexity of $O(N)$, resulting in an overall $O(T \cdot N)$ complexity. ■

D.4 Proof of Theorem 6

Proof The Euclidean projection of a vector $\boldsymbol{b}$ onto the set Ω can be written as solving the following optimization:

$$\min_{\boldsymbol{b}_p} \ \|\boldsymbol{b}_p - \boldsymbol{b}\|_2^2 \quad \text{s.t. } 0 \leq \boldsymbol{b}_p \leq 1, \mathbf{1}^T\boldsymbol{b}_p = m^{\text{syn}}.$$

We denote $\boldsymbol{\delta} \triangleq \boldsymbol{b}_p - \boldsymbol{b}$. Then we can rewrite the above optimization as

$$\begin{aligned} \min_{\boldsymbol{\delta}} \ & \|\boldsymbol{\delta}\|_2^2 \\ \text{s.t. } & -\boldsymbol{b} \leq \boldsymbol{\delta} \leq \mathbf{1} - \boldsymbol{b} \\ & \mathbf{1}^T\boldsymbol{\delta} = m^{\text{syn}} - \mathbf{1}^T\boldsymbol{b}. \end{aligned}$$

Theorem 2.1 from Pardalos and Kovoor (1990) implies that $\boldsymbol{\delta}^*$ is the global optimal solution if and only if there exists a $y \in \mathbb{R}$ such that $\forall i \in [N]$

$$\delta_i^*(y) = \begin{cases} -b_i & \text{if } y < -b_i, \\ y & \text{if } -b_i \le y \le 1 - b_i, \\ 1 - b_i & \text{if } y > 1 - b_i. \end{cases} \tag{18}$$

and $\sum_{i=1}^N \delta_i^*(y) = m^{\text{syn}} - \mathbf{1}^T \boldsymbol{b}$. Note that each $\delta_i^*(y)$ is a piecewise linear, monotone non-decreasing function of y. As a result, the function $\sum_{i=1}^N \delta_i^*(y)$ is also a piecewise linear, monotone non-decreasing function. This property allows us to use binary search to efficiently find a y within the interval $[-\max_{i\in[N]}\{b_i\}, 1 - \min_{i\in[N]}\{b_i\}]$ that satisfies $\sum_{i=1}^N \delta_i^*(y) = m^{\text{syn}} - \mathbf{1}^T \boldsymbol{b}$. Once we find such a value of y, we can substitute it into (18) and use $\boldsymbol{b}_p = \boldsymbol{\delta} + \boldsymbol{b}$ to determine the projection vector $\boldsymbol{b}_p$. ■

D.5 Proof of Theorem 7

Proof of convergence rate. We begin by revisiting a classical convergence result for solving convex optimization problems using the projected gradient descent algorithm (Theorem 3.7 in Bubeck et al., 2015).

Lemma 20 *For a convex function f and a convex set Ω, consider the problem of solving $\min_{\boldsymbol{b}\in\Omega} f(\boldsymbol{b})$ using projected gradient descent, with the updating rule given by $\boldsymbol{b}_{t+1} = \Pi_\Omega\left(\boldsymbol{b}_t - \eta \nabla f(\boldsymbol{b}_t)\right)$. Assume that $f(\cdot)$ is L-smooth: $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \le L \cdot \|\boldsymbol{x} - \boldsymbol{y}\|$. If we set the step size $\eta = \frac{1}{L}$, the following bound holds:*

$$f(\boldsymbol{b}_T) - f(\boldsymbol{b}^*) \le \frac{3L\|\boldsymbol{b}_0 - \boldsymbol{b}^*\|_2^2 + f(\boldsymbol{b}_0) - f(\boldsymbol{b}^*)}{T},$$

where $\boldsymbol{b}^* \in \operatorname{argmin}_{\boldsymbol{b}\in\Omega} f(\boldsymbol{b})$ is an optimal solution to the optimization problem.

Now we prove the convergence rate in Theorem 7 using the above lemma.

Proof We first show that our objective function $f(\boldsymbol{b}) = \|\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b} - \hat{\boldsymbol{a}}\|_2^2$ is a L-smooth function with $L = 2 \cdot \left(\frac{\sigma_{\max}(\hat{\boldsymbol{Q}})}{m^{\text{syn}}}\right)^2$ where $\sigma_{\max}(\hat{\boldsymbol{Q}})$ is the largest singular value of $\hat{\boldsymbol{Q}}$. For any $\boldsymbol{x}, \boldsymbol{y}$,

$$\begin{aligned}
\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 &= \left\| \frac{2}{m^{\text{syn}}}\hat{\boldsymbol{Q}}^T\left(\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{x} - \hat{\boldsymbol{a}}\right) - \frac{2}{m^{\text{syn}}}\hat{\boldsymbol{Q}}^T\left(\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{y} - \hat{\boldsymbol{a}}\right)\right\|_2 \\
&= \frac{2}{(m^{\text{syn}})^2} \cdot \left\|\hat{\boldsymbol{Q}}^T\hat{\boldsymbol{Q}}(\boldsymbol{x} - \boldsymbol{y})\right\|_2 \\
&\le \frac{2}{(m^{\text{syn}})^2} \cdot \left\|\hat{\boldsymbol{Q}}^T\hat{\boldsymbol{Q}}\right\|_2 \cdot \|\boldsymbol{x} - \boldsymbol{y}\|_2 \\
&= 2\left(\frac{\sigma_{\max}(\hat{\boldsymbol{Q}})}{m^{\text{syn}}}\right)^2 \cdot \|\boldsymbol{x} - \boldsymbol{y}\|_2.
\end{aligned}$$

Now by applying Lemma 20, after T iteration of projected gradient descent, we obtain the desired inequality:

$$f(\boldsymbol{b}_T) - f(\boldsymbol{b}^*) \leq \frac{3L\|\boldsymbol{b}_0 - \boldsymbol{b}^*\|_2^2 + f(\boldsymbol{b}_0) - f(\boldsymbol{b}^*)}{T}.$$

■

Runtime analysis. Let the query matrix $\hat{\boldsymbol{Q}} \in \mathbb{R}^{d\times N}$, where d denotes the number of cross-table k-way marginal queries selected by the exponential mechanism, and N represents the dimension of the bi-adjacency matrix for the synthetic relational database. Our algorithm begins by running the power iteration method to compute $\sigma_{\max}(\hat{\boldsymbol{Q}})$, which is used to determine the step size. In each iteration, the power iteration performs the operation $\hat{\boldsymbol{Q}}^T(\hat{\boldsymbol{Q}}\boldsymbol{v})$, followed by normalizing the resulting vector. The matrix-vector multiplications require $O(d \cdot N)$ operations, while normalization takes $O(N)$ operations. Thus, the overall computational complexity of this step is $O(T_{\text{power}} \cdot d \cdot N)$, where T_{power} denotes the number of power iterations.

We run the projected gradient descent algorithm for T iterations. In each iteration, the gradient of the objective function is computed as $\nabla f(\boldsymbol{b}_t) = \frac{2}{m^{\text{syn}}}\hat{\boldsymbol{Q}}^T(\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b}_t - \hat{\boldsymbol{a}})$. The matrix-vector multiplication involved in this computation has a complexity of $O(d \cdot N)$. Afterward, we project the solution onto the constraint set Ω. Using the closed-form expression from Theorem 6, the projection of $\boldsymbol{b}_t - \eta\nabla f(\boldsymbol{b}_t)$ onto Ω has $O(N)$ complexity. Therefore, the total computational complexity is $O(T \cdot d \cdot N)$. We omit multiplicative constants T_{power}, T, and d in our final expression as they are relatively way much smaller than N.

D.6 Proof of Theorem 8

Proof of unbiased sampling. We prove it via mathematical induction. For the two base cases $m = 0$ and $m = 1$, $\mathsf{UBS}(\boldsymbol{x}, m)$ always returns an unbiased sample. Now assume that for $m - 1 \geq 1$, $\mathsf{UBS}(\boldsymbol{x}, m-1)$ returns an unbiased sample for any vector $\boldsymbol{x}$. We will prove that the same holds for $\mathsf{UBS}(\boldsymbol{x}, m)$.

In the first step of our algorithm (merging), the indices of $\boldsymbol{x}$ are partitioned into groups $\mathcal{G}[0], \cdots, \mathcal{G}[L-1]$. We prove that $2 \leq m \leq L \leq 2m - 1$. From (9), we have

$$m = \mathbf{1}^T\boldsymbol{x} = \sum_{j=0}^{L-1} \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j]) \leq L,$$

where the last step is because $\mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j]) \leq 1$. Therefore, we have $2 \leq m \leq L$. Note that

$$\mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j]) + \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j+1]) > 1 \quad \text{for } j = 0, \cdots, L-2 \tag{19}$$

since otherwise $\mathcal{G}[j]$ and $\mathcal{G}[j+1]$ would have been merged in the previous step.

Case 1. If $L \geq 2$ is even, then

$$m = \sum_{j=0}^{L-1} \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j]) = \sum_{j=0}^{L/2-1} (\mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[2j]) + \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[2j+1])) > L/2.$$

Thus, $2m > L$, and since L is an integer, we have $2m - 1 \geq L$.
Case 2. If $L \geq 2$ is odd,

$$m = \sum_{j=0}^{L-1} \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j]) = \sum_{j=0}^{(L-1)/2-1} (\mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[2j]) + \mathcal{G}[2j+1]) + \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[L-1])$$
$$> (L-1)/2.$$

This implies $2m + 1 > L$, and since L is odd, we conclude that $2m - 1 \geq L$.
According to Algorithm 3, $\mathsf{UBS}(\boldsymbol{x}, m)$ calls $\mathsf{UBS}(\boldsymbol{p}', L - m)$ to sample $L - m$ groups to exclude where $p'_j = 1 - \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j])$. Since we just proved that $0 \leq L - m \leq m - 1$, by our assumption $\mathsf{UBS}(\boldsymbol{p}', L - m)$ returns an unbiased sample:

$$\mathbb{P}(\text{group } \mathcal{G}[j] \text{ is selected}) = 1 - \mathbb{P}(\text{group } \mathcal{G}[j] \text{ is excluded}) = 1 - p'_j = \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j]).$$

Finally, by our sampling strategy, for the index $i \in \mathcal{G}[j]$

$$\mathbb{P}(i \text{ is sampled}) = \mathbb{P}(\text{group } \mathcal{G}[j] \text{ is selected}) * \mathbb{P}(i \text{ is sampled} \mid \text{group } \mathcal{G}[j] \text{ is selected})$$
$$= \mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j]) * \frac{x_i}{\mathsf{sum}(\boldsymbol{x} \mid \mathcal{G}[j])} = x_i.$$

This condition holds for any index i. Thus, $\mathsf{UBS}(\boldsymbol{x}, m)$ returns an unbiased sample, completing the inductive step. As a corollary, our algorithm will terminate since each recursive call to UBS reduces the value of its second argument by at least one.

Runtime analysis. We analyze the runtime of Algorithm 3. Since the vector $\boldsymbol{b}^{\text{syn}}$ has dimension N, the first time UBS merges the indices of $\boldsymbol{b}^{\text{syn}}$ into groups, the runtime is $O(N)$. This merging process results in L groups, and we have already proven that $0 \leq L - m \leq m - 1$. Therefore, when the recursive algorithm UBS is called for the second time, its input vector $\boldsymbol{x}$ has a dimension smaller than m. Below, we analyze its runtime. The key idea in our analysis is that the number of groups after merging decreases exponentially by a factor of at least $2/3$ in each iteration.

For any n, let $T(n)$ denote the runtime of UBS on a vector $\boldsymbol{x}$ of size n, which is the output of the recursive algorithm in the previous step. We omit the dependence on the second argument of UBS, as it is always smaller than n.

Since $\boldsymbol{x}$ is the output of the previous recursion, it represents a vector of complement probabilities with dimension n. As a result, $\forall 1 \leq j \leq L - 1 : x_j + x_{j+1} < 1$; otherwise, the j-th and $(j+1)$-th groups would have already been merged in the previous iteration (similar to the reasoning in (19)). In the current iteration, when we partition the indices of $\boldsymbol{x}$ into disjoint groups, each group, except the last one, will contain at least two indices. Thus, the number of groups in the current iteration, n', satisfies $n' \leq 1 + \frac{n-1}{2} \leq \frac{2}{3}n$ for $n \geq 3$. As a result, when the current iteration calls the function UBS itself, the length of its input vector will be at most $\frac{2}{3}n$.

On the other hand, the merging step can be completed in a single scan of the vector $\boldsymbol{x}$, which has dimension n, resulting in a runtime of $O(n)$. Hence,

$$T(n) = T\left(\frac{2}{3}n\right) + O(n).$$

By the master theorem (Cormen et al., 2022), the overall runtime is $T(n) = O(n)$ for any n. Putting everything together, the runtime of Algorithm 3 is $O(N) + T(m) = O(N)$, since $T(m) = O(m)$ and $m \leq N$.

D.7 Proof of Theorem 9

Before presenting the proof of Theorem 9, we first recall the definition of negatively associated (NA) random variables, explaining why this property is crucial for deriving Hoeffding's inequality in the context of dependent random variables. The key step in proving Theorem 9 is to establish that the outputs from Algorithm 3 are NA random variables. Since Algorithm 3 is a recursive algorithm that alternates between complementing and merging, we next introduce two useful lemmas.

Lemma 25 shows that if a set of binary random variables $(X_1, \cdots, X_n)$ is NA, then their complements $(1 - X_1, \cdots, 1 - X_n)$ also satisfy the NA property. This result ensures that the complement step in Algorithm 3 preserves the NA property. Lemma 26 shows that if the random variables satisfy the NA property after a merging step in Algorithm 3, then they maintain the NA property prior to this merging. Together, these lemmas reduce the problem to proving that the random variables in the base case, $m = 1$, are NA.

Definition 21 *Random variables $X_1, X_2, \cdots, X_N$ are said to satisfy* negatively associated *(NA) if for any $\mathcal{A}_1, \mathcal{A}_2$ which are disjoint subsets of $[N]$, we have*

$$\operatorname{Cov}\left(f_1(X_i, i \in \mathcal{A}_1), f_2(X_j, j \in \mathcal{A}_2)\right) \leq 0, \tag{20}$$

whenever f_1 and f_2 are both non-increasing or both non-decreasing. This inequality is also equivalent to

$$\mathbb{E}[f_1(X_i, i \in \mathcal{A}_1) \cdot f_2(X_j, j \in \mathcal{A}_2)] \leq \mathbb{E}[f_1(X_i, i \in \mathcal{A}_1)] \cdot \mathbb{E}[f_2(X_j, j \in \mathcal{A}_2)]. \tag{21}$$

Lemma 22 *(Dubhashi and Ranjan, 1998) Let $X_1, X_2, \ldots, X_N$ be NA binary random variables. The Hoeffding's inequality is applicable to their sum:*

$$\mathbb{P}\left(\left|\sum_i X_i - \mathbb{E}\left[\sum_i X_i\right]\right| \geq t\right) \leq 2\exp\left(\frac{-2t^2}{N}\right).$$

Lemma 23 *(Joag-dev and Proschan, 1983) If $X_1, X_2, \ldots, X_N$ are NA random variables, for any subset of indices $\mathcal{I} \subseteq [N]$, the set of random variables $\{X_i\}_{i \in I}$ also satisfies NA.*

Combining Lemma 22 and 23 leads to the following result.

Lemma 24 *Let $X_1, X_2, \ldots, X_N$ be NA binary random variables. For any subset of indices $\mathcal{I} \subseteq [N]$, the sum of random variables in $\mathcal{I}$ satisfies Hoeffding's inequality:*

$$\mathbb{P}\left(\left|\sum_{i \in \mathcal{I}} X_i - \mathbb{E}\left[\sum_{i \in \mathcal{I}} X_i\right]\right| \geq t\right) \leq 2\exp\left(\frac{-2t^2}{N}\right).$$

Proof Lemma 23 implies that $\{X_i\}_{i\in\mathcal{I}}$ are NA. Therefore, by applying Lemma 22 on them, we have

$$\mathbb{P}\left(\left|\sum_{i\in\mathcal{I}} X_i - \mathbb{E}\left[\sum_{i\in\mathcal{I}} X_i\right]\right| \geq t\right) \leq 2\exp\left(\frac{-2t^2}{|\mathcal{I}|}\right).$$

Since $\mathcal{I} \subseteq [N]$, we have $|\mathcal{I}| \leq N$ and, hence,

$$\mathbb{P}\left(\left|\sum_{i\in\mathcal{I}} X_i - \mathbb{E}\left[\sum_{i\in\mathcal{I}} X_i\right]\right| \geq t\right) \leq 2\exp\left(\frac{-2t^2}{|\mathcal{I}|}\right) \leq 2\exp\left(\frac{-2t^2}{N}\right),$$

which completes the proof. ■

The next result shows that the complement step in Algorithm 3 preserves the NA property.

Lemma 25 *If $X_1, X_2, \cdots, X_N$ are NA random variables, then $1-X_1, 1-X_2, \cdots, 1-X_N$ also satisfy NA.*

Proof For any non-decreasing functions f_1 and f_2 and two disjoint subsets $\mathcal{A}_1, \mathcal{A}_2 \subseteq [N]$, we can define g_1 and g_2 as follows:

$$g_i(X_i, i \in \mathcal{A}_i) \triangleq f_i(1 - X_i, i \in \mathcal{A}_i) \quad \text{for } i \in \{1, 2\}.$$

By definition, g_1 and g_2 are non-increasing functions. Since $X_1, \cdots, X_N$ are NA, we have

$$\mathrm{Cov}\left(f_1(1 - X_i, i \in \mathcal{A}_1), f_2(1 - X_j, j \in \mathcal{A}_2)\right) = \mathrm{Cov}\left(g_1(X_i, i \in \mathcal{A}_1), g_2(X_j, j \in \mathcal{A}_2)\right) \leq 0.$$

The proof for non-increasing f_1 and f_2 can be derived in a similar manner. ■

Lemma 26 *Let $X_1, X_2, \cdots, X_N$ be binary NA random variables, and let Z be a random variable, independent with $X_1, \cdots, X_n$, taking values in $[n]$ with probability distribution:*

$$\mathbb{P}(Z = i) = \frac{p_i}{\sum_{j\in[n]} p_j} \quad \forall i \in [n].$$

We define $Y_1, \cdots, Y_n$ as binary random variables where $Y_i = 1$ if and only if $Z = i$ and $X_N = 1$. Then $X_1, X_2, \cdots, X_{N-1}, Y_1, \cdots, Y_n$ are also NA.

Proof Let f_1 and f_2 be non-decreasing functions. Let $\mathcal{A}_1, \mathcal{A}_2$ be disjoint subsets of $[N-1]$ and $\mathcal{B}_1, \mathcal{B}_2$ be disjoint subsets of $[n]$. We aim to prove that

$$\mathrm{Cov}\left(f_1(X_{i_\mathcal{A}}, Y_{i_\mathcal{B}}, i_\mathcal{A} \in \mathcal{A}_1, i_\mathcal{B} \in \mathcal{B}_1), f_2(X_{j_\mathcal{A}}, Y_{j_\mathcal{B}}, j \in \mathcal{A}_2, j_\mathcal{B} \in \mathcal{B}_2)\right) \leq 0.$$

Using the law of total covariance and conditioning on Z, we have

$$\mathrm{Cov}\left(f_1(\cdot), f_2(\cdot)\right) = \mathbb{E}[\mathrm{Cov}\left(f_1(\cdot), f_2(\cdot) \mid Z\right)] + \mathrm{Cov}\left(\mathbb{E}[f_1(\cdot) \mid Z], \mathbb{E}[f_2(\cdot) \mid Z]\right). \tag{22}$$

Here we simplify notation by denoting $f_1(\cdot) \triangleq f_1(X_{i_\mathcal{A}}, Y_{i_\mathcal{B}}, i_\mathcal{A} \in \mathcal{A}_1, i_\mathcal{B} \in \mathcal{B}_1)$, with $f_2(\cdot)$ defined analogously. Note that $Y_1, \cdots, Y_n$ can be fully determined by X_N and Z since we can write:

$$Y_Z = X_N \quad \text{and} \quad Y_i = 0 \text{ for } i \neq Z.$$

Without loss of generality, assume that $z \in \mathcal{B}_1$. We define $\mathcal{A}_1' = \mathcal{A}_1 \cup \{N\}$. Conditioning on $Z = z$, we define

$$f_1^z(X_i, i \in \mathcal{A}_1') = f_1(X_{i_\mathcal{A}}, Y_{i_\mathcal{B}}, i_\mathcal{A} \in \mathcal{A}_1, i_\mathcal{B} \in \mathcal{B}_1).$$
$$f_2^z(X_i, i \in \mathcal{A}_2) = f_2(X_{j_\mathcal{A}}, Y_{j_\mathcal{B}}, j_\mathcal{A} \in \mathcal{A}_2, j_\mathcal{B} \in \mathcal{B}_2).$$

These two functions are still non-decreasing since they can be obtained by fixing $n-1$ variables (i.e., Y_i for $i \in [n] \backslash \{z\}$) to zero and changing the variable name from Y_z to X_N. Since $X_1, \cdots, X_N$ are NA random variables, we have

$$\operatorname{Cov}\left(f_1^z(X_i, i \in \mathcal{A}_1'), f_2^z(X_i, i \in \mathcal{A}_2)\right) \leq 0, \quad \forall z \in [n].$$

Averaging this inequality over the distribution of Z leads to

$$\mathbb{E}[\operatorname{Cov}\left(f_1(\cdot), f_2(\cdot) \mid Z\right)] \leq 0.$$

Next, we prove that the second term on the right-hand-side of (22) is also non-positive. We define

$$g_i(Z) \triangleq \mathbb{E}[f_i(\cdot) \mid Z] - \mathbb{E}[f_i(\cdot) \mid \forall j \in \mathcal{B}_i : Y_j = 0].$$

Since covariance is linear shift-invariant, we have

$$\begin{aligned} \operatorname{Cov}\left(\mathbb{E}[f_1(\cdot) \mid Z], \mathbb{E}[f_2(\cdot) \mid Z]\right) &= \operatorname{Cov}\left(g_1(Z), g_2(Z)\right) \\ &= \mathbb{E}\left[g_1(Z) g_2(Z)\right] - \mathbb{E}\left[g_1(Z)\right] \mathbb{E}\left[g_2(Z)\right]. \end{aligned} \tag{23}$$

If $z \notin \mathcal{B}_1$, then $Y_j = 0$ for $\forall j \in \mathcal{B}_i$. In this case, $g_1(z) = 0$ by its definition. Similarly, if $z \notin \mathcal{B}_2$, then $g_2(z) = 0$. Since $\mathcal{B}_1$ and $\mathcal{B}_2$ are disjoint, at least one of the events $Z \notin \mathcal{B}_1$ or $Z \notin \mathcal{B}_2$ occurs. Therefore,

$$g_1(z) g_2(z) = 0, \quad \forall z \in [n].$$

Taking expectation over Z, we have

$$\mathbb{E}\left[g_1(Z) g_2(Z)\right] = 0. \tag{24}$$

Since f_i is a non-decreasing function, $g_i(z) \geq 0$ for $\forall z \in [n]$. Therefore, $\mathbb{E}\left[g_i(Z)\right] \geq 0$, yielding

$$\mathbb{E}\left[g_1(Z)\right] \mathbb{E}\left[g_2(Z)\right] \geq 0. \tag{25}$$

Substituting (24) and (25) into (23) leads to the desired result. ■

We extend Lemma 26 to analyze the merging step in Algorithm 3. During this step, we partition $[N]$ into L groups, $\mathcal{G}[1], \cdots, \mathcal{G}[L]$, where each group satisfies the conditions of Lemma 26. We sample an index within each group with probability proportional to its desired probability, normalized by the total probability of the subset. Lemma 26 ensures that if the set of the random variables corresponding to the merged sets is NA, then unmerging one group and replacing its random variable with those of its elements still preserve the NA property. In Lemma 27, we present this result formally.

Lemma 27 *Suppose $\mathcal{G}[1], \cdots, \mathcal{G}[L]$ are disjoint subsets that partition the set $[N]$. Let $X_1, X_2, \cdots, X_L$ be binary NA random variables. Additionally, let $Z_1, \cdots, Z_L$ be random variables, mutually independent and independent with $X_1, X_2, \cdots, X_L$. Each Z_i takes values in $\mathcal{G}[i]$ with probability:*

$$\mathbb{P}[Z_i = j] = \frac{p_{i,j}}{\sum_{j' \in G[i]} p_{i,j'}} \quad \forall j \in G[i].$$

We define $\{Y_{i,j}\}_{i \in [L], j \in G[i]}$ as binary random variables where $Y_{i,j} = 1$ if and only if $Z_i = j$ and $X_i = 1$. Then $\{Y_{i,j}\}_{i \in [L], j \in G[i]}$ are also NA.

Proof We denote the number of subsets containing more than one element by $k \triangleq |\{i \mid |G[i]| \geq 2\}|$. We prove this lemma by induction on k. In the base case, where all groups have exactly one element (i.e., $k = 0$), we have $\mathbb{P}[Z_i = j] = 1$ for $j \in G[i]$. Then, by definition, $Y_{i,j} = X_i$ for $j \in G[i]$. Consequently, $\{Y_{i,j}\}_{i \in [L], j \in G[i]}$ is equivalent to $\{X_i\}_{i \in [L]}$ and is NA.

Assume that the lemma holds for k. Next, we prove that it holds for the case where $k + 1$ subsets have more than one element. Without loss of generality, suppose that $G[L] = \{L_1, \cdots, L_s\}$ with $s \geq 2$. By Lemma 26, replacing X_L with $\{Y_{L,j}\}_{j \in G[L]}$ in the set $\{X_1, X_2, \cdots, X_L\}$ maintains the NA property. This new set of random variables corresponds to the following disjoint subsets that partition $[N]$:

$$\mathcal{G}[1], \cdots, \mathcal{G}[L-1], \{L_1\}, \cdots, \{L_s\}.$$

Among them, there are only k subsets having more than one element. The induction hypothesis implies that unpacking the remaining subsets with more than one element will also preserve the NA property. ■

Now we provide a proof of Theorem 9. The key idea is to prove that the random variables in Algorithm 3 satisfy the NA property throughout the recursive sampling process. In the proof, we denote $\boldsymbol{b} \triangleq \mathrm{vec}(\boldsymbol{B}^{\mathrm{syn}})$ and $\widetilde{\boldsymbol{b}} \triangleq \mathrm{vec}(\widetilde{\boldsymbol{B}}^{\mathrm{syn}})$.

Proof We first prove that the function $\mathsf{UBS}(\boldsymbol{x}, m)$ in Algorithm 3 outputs NA random variables. We use mathematical induction on m to prove this. For the base case $m = 1$, there is only one group, so sampling reduces to a categorical selection where one indicator variable equals 1, and the rest are 0. These random indicator variables satisfy NA (see Section 3 (a) in Joag-dev and Proschan, 1983). Now assume that the NA property holds for m, and consider the case of $m + 1$. According to Algorithm 3, we first apply a merging step followed by an inversion step. In the proof of Theorem 8, we established that after these steps, m decreases, allowing us to apply the induction hypothesis. By Lemma 27, if

the random variables are NA after the merging step, then the NA property holds for the variables prior to the merging step. Similarly, Lemma 25 ensures that the NA property is preserved through the inversion step. Therefore, by successively applying the merging and inversion steps, we conclude that the original set of random variables retains the NA property for $m+1$. Hence, $\mathsf{UBS}(\boldsymbol{x}, m)$ outputs NA random variables, and the vectorized bi-adjacency matrix $\boldsymbol{b}$, produced by Algorithm 3, satisfies the NA property.

For any marginal query vector $\boldsymbol{q} \in \mathcal{Q}$, by its definition in Lemma 3, there exists a subset $\mathcal{I}_q \in [N]$ such that $\boldsymbol{q}^T\boldsymbol{b} = \sum_{i\in\mathcal{I}_q} b_i$. Additionally, recall that $\boldsymbol{b}$ is an unbiased estimator of $\widetilde{\boldsymbol{b}}$ (Theorem 8). Hence, Lemma 24 implies that

$$\mathbb{P}\left(\left|\boldsymbol{q}^T\boldsymbol{b} - \boldsymbol{q}^T\widetilde{\boldsymbol{b}}\right| \geq t\right) = \mathbb{P}\left(\left|\sum_{i\in\mathcal{I}_q} b_i - \mathbb{E}\left[\sum_{i\in\mathcal{I}_q} b_i\right]\right| \geq t\right) \leq 2\exp\left(\frac{-2t^2}{N}\right).$$

This bound holds for any single query vector $\boldsymbol{q} \in \mathcal{Q}$. Next, we apply the union bound to extend this result for all $\boldsymbol{q} \in \mathcal{Q}$:

$$\begin{aligned}
\mathbb{P}\left(\exists \boldsymbol{q} \in \mathcal{Q} : \left|\boldsymbol{q}^T\boldsymbol{b} - \boldsymbol{q}^T\widetilde{\boldsymbol{b}}\right| \geq t\right) &\leq \sum_{\boldsymbol{q}\in\mathcal{Q}} \mathbb{P}\left(\left|\boldsymbol{q}^T\boldsymbol{b} - \boldsymbol{q}^T\widetilde{\boldsymbol{b}}\right| \geq t\right) \\
&\leq \sum_{\boldsymbol{q}\in\mathcal{Q}} 2\exp\left(\frac{-2t^2}{N}\right) \\
&= 2|\mathcal{Q}|\exp\left(\frac{-2t^2}{N}\right).
\end{aligned}$$

Therefore, we have

$$\mathbb{P}\left(\forall \boldsymbol{q} \in \mathcal{Q} : \left|\frac{1}{m^{\text{syn}}}\boldsymbol{q}^T\boldsymbol{b} - \frac{1}{m^{\text{syn}}}\boldsymbol{q}^T\widetilde{\boldsymbol{b}}\right| \leq \frac{t}{m^{\text{syn}}}\right) \geq 1 - 2|\mathcal{Q}|\exp\left(\frac{-2t^2}{N}\right).$$

In other words, with at least probability $1-\beta$, for $\forall \boldsymbol{q} \in \mathcal{Q}$, we have

$$\left|\frac{1}{m^{\text{syn}}}\boldsymbol{q}^T\boldsymbol{b} - \frac{1}{m^{\text{syn}}}\boldsymbol{q}^T\widetilde{\boldsymbol{b}}\right| \leq \frac{1}{m^{\text{syn}}} \cdot \sqrt{\frac{N}{2} \cdot \log\frac{2|\mathcal{Q}|}{\beta}},$$

which completes the proof. ■

D.8 Proof of Theorem 11

We provide a more rigorous statement of Theorem 11 along with its proof.

Theorem 28 *Let $\mathcal{D}_1^{syn}$ and $\mathcal{D}_2^{syn}$ be any individual synthetic tables. We stack all the k-way cross-table query vectors as $\hat{\boldsymbol{Q}}$. Denote by $\boldsymbol{a}$ the true answers to these queries computed from the original database. Suppose we use a privacy budget $(\epsilon_{rel}, \delta_{rel})$ to run a single iteration of Algorithm 1 (without unbiased sampling), which outputs $\boldsymbol{B}^{syn}$. Let $\boldsymbol{B}^*$ denote*

the optimal weighted bi-adjacency matrix in minimizing the mean squared error when no privacy constraints are enforced. With probability at least $1-\beta$*, we have*

$$\left\|\frac{1}{m^{syn}}\hat{\boldsymbol{Q}}\,\mathrm{vec}(\boldsymbol{B}^{syn})-\boldsymbol{a}\right\|_2^2 \leq 8\left\|\frac{1}{m^{syn}}\hat{\boldsymbol{Q}}\,\mathrm{vec}(\boldsymbol{B}^{*})-\boldsymbol{a}\right\|_2^2 + 8\sqrt{2}|\mathcal{W}_{cross,k}|\frac{d_{max}}{m\sqrt{\rho_{rel}}}\left(\log\frac{2n_1^{syn}n_2^{syn}}{\beta}\right)^{\frac{1}{2}},$$

where $\rho_{rel}=\left(\sqrt{\epsilon_{rel}+\log\frac{1}{\delta_{rel}}}-\sqrt{\log\frac{1}{\delta_{rel}}}\right)^2$ *and* m *is the number of relationships in the original database.*

We introduce some useful lemmas that will be used in the proof of Theorem 28.

Lemma 29 *(Contractive property of convex sets projection Schneider (2013)) Let* $\mathcal{Y}$ *be a non-empty closed convex subset of* $\mathbb{R}^d$*, then for any* $\boldsymbol{x}_1,\boldsymbol{x}_2$ *if we denote their projections by* $\boldsymbol{y}_1,\boldsymbol{y}_2$ *respectively, i.e.* $\boldsymbol{y}_i=\arg\min_{\boldsymbol{y}\in\mathcal{Y}}\|\boldsymbol{x}_i-\boldsymbol{y}\|_2^2$*, then we have*

$$\|\boldsymbol{y}_1-\boldsymbol{y}_2\|_2^2\leq\|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2^2.$$

Proof Let $\boldsymbol{v}\triangleq\boldsymbol{y}_2-\boldsymbol{y}_2\neq o$. The function f defined by $f(t)\triangleq|\boldsymbol{x}_1-(\boldsymbol{y}_1+t\boldsymbol{v})|^2$ for $t\in[0,1]$ has a minimum at $t=0$, hence $f'(0)\geq 0$. This gives $\langle\boldsymbol{x}_1-\boldsymbol{y}_1,\boldsymbol{v}\rangle\leq 0$. Similarly we obtain $\langle\boldsymbol{x}_2-\boldsymbol{y}_2,\boldsymbol{v}\rangle\geq 0$. Thus, the segment $[\boldsymbol{x}_1,\boldsymbol{x}_2]$ meets the two hyperplanes that are orthogonal to $\boldsymbol{v}$ and that go through $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, respectively. ■

Lemma 30 *Let* $\mathcal{Y}$ *be a non-empty closed set. For any* $\boldsymbol{x}$*, we denote its projection onto* $\mathcal{Y}$ *by* $\boldsymbol{y}^*$*, i.e.* $\boldsymbol{y}^*=\arg\min_{\boldsymbol{y}\in\mathcal{Y}}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2$*. Then for any* $\boldsymbol{y}'\in\mathcal{Y}$ *we have*

$$\|\boldsymbol{y}^*-\boldsymbol{y}'\|_2^2\leq 2\langle\boldsymbol{y}^*-\boldsymbol{y}',\boldsymbol{x}-\boldsymbol{y}'\rangle.$$

Proof By the optimality of $\boldsymbol{y}^*$ and $\boldsymbol{y}'\in\mathcal{Y}$, we have

$$\|\boldsymbol{x}-\boldsymbol{y}^*\|_2^2\leq\|\boldsymbol{x}-\boldsymbol{y}'\|_2^2.$$

This implies that

$$\|\boldsymbol{x}-\boldsymbol{y}'\|_2^2+2\langle\boldsymbol{x}-\boldsymbol{y}',\boldsymbol{y}'-\boldsymbol{y}^*\rangle+\|\boldsymbol{y}'-\boldsymbol{y}^*\|_2^2\leq\|\boldsymbol{x}-\boldsymbol{y}'\|_2^2,$$

which yields the desired inequality. ■

The following lemma is a standard result from concentration inequalities (Boucheron et al., 2003).

Lemma 31 *Let* $X_1,\ldots,X_n$ *be* $\mathcal{N}(0,\sigma^2)$ *normal random variables with zero mean. Then* $\mathbb{P}\left\{\max_{1\leq i\leq n}X_i\geq\sqrt{2\sigma^2(\log n+t)}\right\}\leq\exp(-t)$.

Proof Let $u \triangleq \sqrt{2\sigma^2(\log n + t)}$. We have

$$\begin{aligned}\mathbb{P}\left\{\max_{1\le i\le n} X_i \ge u\right\} &= \mathbb{P}\left\{\exists i, X_i \ge u\right\} \\ &\le \sum_{i=1}^{n} \mathbb{P}\left\{X_i \ge u\right\} \\ &\le n\exp\left(-\frac{u^2}{2\sigma^2}\right) = \exp(-t),\end{aligned}$$

which yields the desired inequality. ■

Next, we provide the proof of Theorem 28

Proof We define $N = n_1^{\text{syn}} \cdot n_2^{\text{syn}}$. Additionally, we define the projection set with respect to $\hat{\boldsymbol{Q}}$ as

$$\mathcal{K} = \left\{ \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b} \;\middle|\; \boldsymbol{b} \in \mathbb{R}^N, \mathbf{1}^T\boldsymbol{b} = m^{\text{syn}}, 0 \le \boldsymbol{b} \le 1 \right\}. \tag{26}$$

Recall that $\boldsymbol{B}^*$ is the optimal bi-adjacency matrix achievable for the synthetic individual tables without privacy constraint and $\boldsymbol{B}^{\text{syn}}$ is the output of Algorithm 1. We denote

$$\boldsymbol{b}^* \triangleq \text{vec}(\boldsymbol{B}^*), \quad \hat{\boldsymbol{b}} \triangleq \text{vec}(\boldsymbol{B}^{\text{syn}}).$$

By definition,

$$\begin{aligned}\boldsymbol{b}^* &= \operatorname*{argmin}_{\hat{\boldsymbol{Q}}\boldsymbol{b}/m^{\text{syn}} \in \mathcal{K}} \left\| \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b} - \boldsymbol{a} \right\|_2^2, \\ \hat{\boldsymbol{b}} &= \operatorname*{argmin}_{\hat{\boldsymbol{Q}}\boldsymbol{b}/m^{\text{syn}} \in \mathcal{K}} \left\| \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b} - (\boldsymbol{a} + \boldsymbol{w}) \right\|_2^2,\end{aligned}$$

where $\boldsymbol{w}$ is a Gaussian vector with zero-mean added to preserve DP guarantees, which we will specify later. Additionally, we introduce

$$\widetilde{\boldsymbol{b}} = \operatorname*{argmin}_{\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b} \in \mathcal{K}} \left\| \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b} - (\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b}^* + \boldsymbol{w}) \right\|_2^2. \tag{27}$$

By the triangle inequality, we have:

$$\left\| \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\hat{\boldsymbol{b}} - \boldsymbol{a} \right\|_2 \le \left\| \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\hat{\boldsymbol{b}} - \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\widetilde{\boldsymbol{b}} \right\|_2 + \left\| \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\widetilde{\boldsymbol{b}} - \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b}^* \right\|_2 + \left\| \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b}^* - \boldsymbol{a} \right\|_2.$$

Since $\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\hat{\boldsymbol{b}}$ and $\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\widetilde{\boldsymbol{b}}$ are the projections of $(\boldsymbol{a}+\boldsymbol{w})$ and $(\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b}^*+\boldsymbol{w})$ onto the convex set $\mathcal{K}$, respectively, Lemma 29 implies that

$$\left\| \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\hat{\boldsymbol{b}} - \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\widetilde{\boldsymbol{b}} \right\|_2 \le \left\| (\boldsymbol{a} + \boldsymbol{w}) - (\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b}^* + \boldsymbol{w}) \right\|_2 = \left\| \boldsymbol{a} - \frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b}^* \right\|_2.$$

Hence, we have

$$\left\| \frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}} \hat{\boldsymbol{b}} - \boldsymbol{a} \right\|_2 \leq 2 \left\| \boldsymbol{a} - \frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}} \boldsymbol{b}^* \right\|_2 + \left\| \frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}} \widetilde{\boldsymbol{b}} - \frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}} \boldsymbol{b}^* \right\|_2 . \tag{28}$$

By the definition of $\widetilde{\boldsymbol{b}}$ in (27) and Lemma 30, we have

$$\begin{aligned} \left\| \frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}} \widetilde{\boldsymbol{b}} - \frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}} \boldsymbol{b}^* \right\|_2^2 &\leq \frac{2}{m^{\text{syn}}} \left\langle \hat{\boldsymbol{Q}} \widetilde{\boldsymbol{b}} - \hat{\boldsymbol{Q}} \boldsymbol{b}^*, (\frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}} \boldsymbol{b}^* + \boldsymbol{w}) - \frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}} \boldsymbol{b}^* \right\rangle \\ &= \frac{2}{m^{\text{syn}}} \langle \hat{\boldsymbol{Q}} (\widetilde{\boldsymbol{b}} - \boldsymbol{b}^*), \boldsymbol{w} \rangle. \end{aligned}$$

Since $\frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}} \boldsymbol{b}^*, \frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}} \widetilde{\boldsymbol{b}} \in \mathcal{K}$, we have $\|\boldsymbol{b}^*\|_1 = \|\widetilde{\boldsymbol{b}}\|_1 = m^{\text{syn}}$. We denote the j-th column of $\hat{\boldsymbol{Q}}$ by $\boldsymbol{p}_j$ for $j = 1, \cdots, N$. Then

$$\begin{aligned} \left\| \frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}} \widetilde{\boldsymbol{b}} - \frac{1}{m^{\text{syn}}} \hat{\boldsymbol{Q}} \boldsymbol{b}^* \right\|_2^2 &\leq \frac{2}{m^{\text{syn}}} \left\langle \hat{\boldsymbol{Q}} (\widetilde{\boldsymbol{b}} - \boldsymbol{b}^*), \boldsymbol{w} \right\rangle \\ &= \frac{2}{m^{\text{syn}}} \sum_{j=1}^{N} (\widetilde{b}_j - b_j^*) \cdot \langle \boldsymbol{p}_j, \boldsymbol{w} \rangle \\ &\leq \frac{2}{m^{\text{syn}}} \sum_{j=1}^{N} |\widetilde{b}_j - b_j^*| \cdot \max_{j=1}^{N} |\langle \boldsymbol{p}_j, \boldsymbol{w} \rangle| \\ &\leq \frac{2}{m^{\text{syn}}} \cdot (\|\widetilde{\boldsymbol{b}}\|_1 + \|\boldsymbol{b}^*\|_1) \cdot \max_{j=1}^{N} |\langle \boldsymbol{p}_j, \boldsymbol{w} \rangle| \\ &= 4 \cdot \max_{j=1}^{N} |\langle \boldsymbol{p}_j, \boldsymbol{w} \rangle|. \end{aligned}$$

We introduce random variables:

$$X_j \triangleq \langle \boldsymbol{p}_j, \boldsymbol{w} \rangle, \ X_{N+j} \triangleq -\langle \boldsymbol{p}_j, \boldsymbol{w} \rangle, \quad \text{for } j = 1, \cdots, N.$$

Since each random variable X_j for $j \in [2N]$ is a linear combination of independent zero mean Gaussian random variables, X_j is also a zero mean Gaussian random variable. Next, we prove that $\boldsymbol{p}_j$ has $|\mathcal{W}_{\text{cross},k}|$ elements equal to 1 and the remaining elements being 0. Here $|\mathcal{W}_{\text{cross},k}|$ represents the number of (cross-table) k-way workloads. Note that there exist $r \in [n_1^{\text{syn}}], s \in [n_2^{\text{syn}}]$ such that

$$j = (r-1) n_2^{\text{syn}} + s.$$

Recall the definition of query vector in Lemma 3. The i-th element of $\boldsymbol{p}_j$ is an indicator of whether the value of the i-th cross-table k-way marginal query of $(\boldsymbol{x}_r^{\text{syn}}, \boldsymbol{x}_s^{\text{syn}})$ is 1. Here $\boldsymbol{x}_r^{\text{syn}}$ and $\boldsymbol{x}_s^{\text{syn}}$ are the r-th and s-th records of $\mathcal{D}_1^{\text{syn}}$ and $\mathcal{D}_2^{\text{syn}}$, respectively. There are $|\mathcal{W}_{\text{cross},k}|$ workloads in total. Within each workload, for queries associated with it, only one query can have a value of 1 for $(\boldsymbol{x}_r^{\text{syn}}, \boldsymbol{x}_s^{\text{syn}})$. Hence, there are $|\mathcal{W}_{\text{cross},k}|$ elements of $\boldsymbol{p}_j$ being 1. Then for $j = 1, \cdots, N$,

$$\begin{aligned} \mathsf{Var}\left(X_{N+j}\right) &= \mathsf{Var}\left(X_j\right) \\ &= \mathsf{Var}\left(\langle \boldsymbol{p}_j, \boldsymbol{w} \rangle\right) = |\mathcal{W}_{\text{cross},k}| \cdot \mathsf{Var}\left(w_i\right). \end{aligned}$$

Lemma 31 implies that with probability at least $1-\beta$, we have

$$\left\|\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\widetilde{\boldsymbol{b}}-\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b}^*\right\|_2^2 \le 4\cdot\max_{j=1}^{2N} X_j$$
$$\le 4\cdot\sqrt{2|\mathcal{W}_{\text{cross},k}|\cdot\mathsf{Var}\left(w_i\right)\left(\log(2N)+\log\frac{1}{\beta}\right)}.$$

Substituting the above inequality into (28) yields

$$\left\|\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\hat{\boldsymbol{b}}-\boldsymbol{a}\right\|_2 \le 2\left\|\boldsymbol{a}-\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b}^*\right\|_2+\left(32|\mathcal{W}_{\text{cross},k}|\cdot\mathsf{Var}\left(w_i\right)\left(\log(2N)+\log\frac{1}{\beta}\right)\right)^{\frac{1}{4}}.$$

Since we run Algorithm 1 (without unbiased sampling) for a single iteration across all k-way workloads, we have $T=1$, $K=|\mathcal{W}_{\text{cross},k}|$, and $\alpha=0$. Therefore,

$$\mathsf{Var}\left(w_i\right)=\left(\frac{\sqrt{2}d_{\max}}{m\epsilon_0}\right)^2=K\cdot\frac{d_{\max}^2}{m^2\rho_{\text{rel}}}=|\mathcal{W}_{\text{cross},k}|\cdot\frac{d_{\max}^2}{m^2\rho_{\text{rel}}}.$$

As a result,

$$\left\|\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\hat{\boldsymbol{b}}-\boldsymbol{a}\right\|_2^2 \le 8\left\|\frac{1}{m^{\text{syn}}}\hat{\boldsymbol{Q}}\boldsymbol{b}^*-\boldsymbol{a}\right\|_2^2+8\sqrt{2}|\mathcal{W}_{\text{cross},k}|\frac{d_{\max}}{m\sqrt{\rho_{\text{rel}}}}\left(\log\frac{2N}{\beta}\right)^{\frac{1}{2}}.$$

■

Acknowledgments

This work was supported in part by the MIT-IBM Computing Research Lab, the MIT-Amazon Science Hub, Jane Street, and Netflix.

References

John M Abowd. The US census bureau adopts differential privacy. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2867–2867, 2018.

Sergul Aydore, William Brown, Michael Kearns, Krishnaram Kenthapadi, Luca Melis, Aaron Roth, and Ankit A Siva. Differentially private query release through adaptive projection. In *International Conference on Machine Learning*, pages 457–467. PMLR, 2021.

Brett K Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P Bhavnani, James Brian Byrd, and Casey S Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*, 12(7):e005122, 2019.

Alex Bie, Gautam Kamath, and Guojun Zhang. Private GANs, revisited. *Transactions on Machine Learning Research*, 2023.

Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The Johnson-Lindenstrauss transform itself preserves differential privacy. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 410–419. IEEE, 2012.

Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)*, 60(2):1–25, 2013.

March Boedihardjo, Thomas Strohmer, and Roman Vershynin. Privacy of synthetic data: A statistical framework. *IEEE Transactions on Information Theory*, 69(1):520–527, 2022.

Stéphane Boucheron, Gábor Lugosi, and Olivier Bousquet. Concentration inequalities. In *Summer school on machine learning*, pages 208–240. Springer, 2003.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part I*, pages 635–658. Springer, 2016.

Mark Bun, Marco Gaboardi, Marcel Neunhoeffer, and Wanrong Zhang. Continual release of differentially private synthetic data from longitudinal data collections. *Proceedings of the ACM on Management of Data*, 2(2):1–26, 2024.

Kuntai Cai, Xiaokui Xiao, and Graham Cormode. Privlava: synthesizing relational data with foreign keys under differential privacy. *Proceedings of the ACM on Management of Data*, 1(2):1–25, 2023.

Shaofeng Cai, Kaiping Zheng, Gang Chen, HV Jagadish, Beng Chin Ooi, and Meihui Zhang. Arm-net: Adaptive relation modeling network for structured data. In *Proceedings of the 2021 International Conference on Management of Data*, pages 207–220, 2021.

Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. Differentially private high-dimensional data publication via sampling-based inference. In *ACM SIGKDD international conference on knowledge discovery and data mining*, pages 129–138, 2015.

Xihui Chen, Sjouke Mauw, and Yunior Ramírez-Cruz. Publishing community-preserving attributed social graphs with a differential privacy guarantee. *arXiv preprint arXiv:1909.00280*, 2019.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

Milan Cvitkovic. Supervised learning on relational databases with graph neural networks. *arXiv preprint arXiv:2002.02046*, 2020.

Konstantin Donhauser, Javier Abad, Neha Hulkund, and Fanny Yang. Privacy-preserving data release leveraging optimal transport and particle gradient descent. *arXiv preprint arXiv:2401.17823*, 2024.

Devdatt Dubhashi and Desh Ranjan. Balls and bins: a study in negative dependence. *Random Struct. Algorithms*, 13(2):99–124, September 1998. ISSN 1042-9832.

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

Marek Eliáš, Michael Kapralov, Janardhan Kulkarni, and Yin Tat Lee. Differentially private release of synthetic graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 560–578. SIAM, 2020.

Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. Relational deep learning: Graph representation learning on relational databases. *arXiv preprint arXiv:2312.04615*, 2023.

Paul Francis. A comparison of syndiffix multi-table versus single-table synthetic data. *arXiv preprint arXiv:2403.08463*, 2024.

Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. Dual query: Practical private query release for high dimensional data. In *International Conference on Machine Learning*, pages 1170–1178. PMLR, 2014.

Quan Gan, Minjie Wang, David Wipf, and Christos Faloutsos. Graph machine learning meets multi-table relational data. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6502–6512, 2024.

Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F Ilyas. Kamino: Constraint-aware differentially private data synthesis. *arXiv preprint arXiv:2012.15713*, 2020.

Kristjan Greenewald, Yuancheng Yu, Hao Wang, and Kai Xu. Privacy without noisy gradients: Slicing mechanism for generative model training. *Advances in Neural Information Processing Systems*, 2024.

Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In *Theory of Cryptography: 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings 9*, pages 339–356. Springer, 2012.

Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. *Advances in neural information processing systems*, 25, 2012.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*, pages 169–178. IEEE, 2009.

Xi He. Differential privacy for complex data: Answering queries across multiple data tables. https://www.nist.gov/blogs/cybersecurity-insights/differential-privacy-complex-data-answering-queries-across-multiple, 2021.

Yiyun He, Roman Vershynin, and Yizhe Zhu. Algorithmically effective differentially private synthetic data. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 3941–3968. PMLR, 2023.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. ISSN 01621459. URL http://www.jstor.org/stable/2282952.

Kumar Joag-dev and Frank Proschan. Negative association of random variables with applications. *Annals of Statistics*, 11:286–295, 1983. URL https://api.semanticscholar.org/CorpusID:119668291.

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

James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2019.

Zach Jorgensen, Ting Yu, and Graham Cormode. Publishing attributed social graphs with formal privacy guarantees. In *Proceedings of the 2016 international conference on management of data*, pages 107–122, 2016.

Kaggle. Kaggle 2017 survey results. https://www.kaggle.com/code/amberthomas/kaggle-2017-survey-results, 2017.

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

Jingcheng Liu, Jalaj Upadhyay, and Zongrui Zou. Optimal bounds on private graph approximation. *arXiv preprint arXiv:2309.17330*, 2023a.

Shang Liu, Hao Du, Yang Cao, Bo Yan, Jinfei Liu, and Masatoshi Yoshikawa. Pgb: Benchmarking differentially private synthetic graph generation algorithms. *arXiv preprint arXiv:2408.02928*, 2024.

Terrance Liu, Giuseppe Vietri, Thomas Steinke, Jonathan Ullman, and Steven Wu. Leveraging public data for practical private query release. In *International Conference on Machine Learning*, pages 6968–6977. PMLR, 2021a.

Terrance Liu, Giuseppe Vietri, and Steven Z Wu. Iterative methods for private synthetic data: Unifying framework and new methods. *Advances in Neural Information Processing Systems*, 34:690–702, 2021b.

Terrance Liu, Jingwu Tang, Giuseppe Vietri, and Steven Wu. Generating private synthetic data with genetic algorithms. In *International Conference on Machine Learning*, pages 22009–22027. PMLR, 2023b.

Ciro Antonio Mami, Andrea Coser, Eric Medvet, Alexander TP Boudewijn, Marco Volpe, Michael Whitworth, Borut Svara, Gabriele Sgroi, Daniele Panfilo, and Sebastiano Saccani. Generating realistic synthetic relational data through graph variational autoencoders. *arXiv preprint arXiv:2211.16889*, 2022.

Ryan McKenna, Daniel Sheldon, and Gerome Miklau. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*, pages 4435–4444. PMLR, 2019.

Ryan McKenna, Gerome Miklau, and Daniel Sheldon. Winning the NIST contest: A scalable and general approach to differentially private synthetic data. *arXiv preprint arXiv:2108.04978*, 2021.

Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. Aim: An adaptive and iterative mechanism for differentially private synthetic data. *arXiv preprint arXiv:2201.12677*, 2022.

Shubhankar Mohapatra, Jianqiao Zong, Florian Kerschbaum, and Xi He. Differentially private data generation with missing data. *arXiv preprint arXiv:2310.11548*, 2023.

Marcel Neunhoeffer, Zhiwei Steven Wu, and Cynthia Dwork. Private post-gan boosting. *arXiv preprint arXiv:2007.11934*, 2020.

Wei Pang, Masoumeh Shafieinejad, Lucy Liu, and Xi He. Clavaddpm: Multi-relational data synthesis with cluster-guided diffusion models. *arXiv preprint arXiv:2405.17724*, 2024.

Panos M Pardalos and Naina Kovoor. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, 46:321–328, 1990.

Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410. IEEE, 2016.

DB-Engines Ranking. DB-engines ranking - popularity ranking of database management systems. https://db-engines.com/en/ranking, 2023.

Diane Ridgeway, Mary F Theofanos, Terese W Manley, and Christine Task. Challenge design and lessons learned from the 2018 differential privacy challenges. *NIST Technical Note 2151*, 2021.

Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan E Lenssen, Yiwen Yuan, Zecheng Zhang, et al. Relbench: A benchmark for deep learning on relational databases. *arXiv preprint arXiv:2407.20060*, 2024.

Lucas Rosenblatt, Xiaoyan Liu, Samira Pouyanfar, Eduardo de Leon, Anuj Desai, and Joshua Allen. Differentially private synthetic data: Applied evaluations and enhancements. *arXiv preprint arXiv:2011.05537*, 2020.

Steven Ruggles, Sarah Flood, Ronald Goeken, Josiah Grover, Erin Meyer, Jose Pacas, and Matthew Sobek. IPUMS USA: Version 10.0 [dataset]. minneapolis, mn: Ipums, 2020. *URL https://doi. org/10.18128 D*, 10, 2021.

Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 81–98, 2011.

Maximilian Schleich, Dan Olteanu, Mahmoud Abo-Khamis, Hung Q Ngo, and XuanLong Nguyen. Learning models over relational data: A brief tutorial. In *Scalable Uncertainty Management: 13th International Conference, SUM 2019, Compiègne, France, December 16–18, 2019, Proceedings 13*, pages 423–432. Springer, 2019.

Rolf Schneider. *Convex Bodies: The Brunn–Minkowski Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2 edition, 2013.

Michel Schubiger, Goran Banjac, and John Lygeros. Gpu acceleration of admm for large-scale quadratic programming. *Journal of Parallel and Distributed Computing*, 144:55–67, 2020.

SmartNoise. Smartnoise sdk: Tools for differential privacy on tabular data. https://github.com/opendp/smartnoise-sdk, 2023.

A. Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 588–597, 2001. doi: 10.1109/SFCS.2001.959935.

Thomas Steinke. Composition of differential privacy & privacy amplification by subsampling. *arXiv preprint arXiv:2210.00597*, 2022.

Uthaipon Tantipongpipat, Chris Waites, Digvijay Boob, Amaresh Ankit Siva, and Rachel Cummings. Differentially private mixed-type data generation for unsupervised learning. *arXiv preprint arXiv:1912.03250*, 1:13, 2019.

Yuchao Tao, Ryan McKenna, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. Benchmarking differentially private synthetic data generation algorithms. *arXiv preprint arXiv:2112.09238*, 2021.

Justin Thaler, Jonathan Ullman, and Salil Vadhan. Faster algorithms for privately releasing marginals. In *International Colloquium on Automata, Languages, and Programming*, pages 810–821. Springer, 2012.

The White House. Executive order on the safe, secure, and trustworthy development and use of artificial intelligence, 2023.

Jonathan Ullman and Salil Vadhan. PCPs and the hardness of generating synthetic data. *Journal of Cryptology*, 33(4):2078–2112, 2020.

Jalaj Upadhyay. Random projections, graph sparsification, and differential privacy. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 276–295. Springer, 2013.

Giuseppe Vietri, Grace Tian, Mark Bun, Thomas Steinke, and Steven Wu. New oracle-efficient algorithms for private synthetic data release. In *International Conference on Machine Learning*, pages 9765–9774. PMLR, 2020.

Giuseppe Vietri, Cedric Archambeau, Sergul Aydore, William Brown, Michael Kearns, Aaron Roth, Ankit Siva, Shuai Tang, and Steven Z Wu. Private synthetic data for multitask learning and marginal queries. *Advances in Neural Information Processing Systems*, 35:18282–18295, 2022.

Hao Wang, Shivchander Sudalairaj, John Henning, Kristjan Greenewald, and Akash Srivastava. Post-processing private synthetic data for improving utility on selected measures. In *Conference on Neural Information Processing Systems*, 2023.

Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

Kai Xu, Georgi Ganev, Emile Joubert, Rees Davison, Olivier Van Acker, and Luke Robinson. Synthetic data generation of many-to-many datasets via random graph generation. In *International Conference on Learning Representations*, 2023.

Jingyi Yang, Peizhi Wu, Gao Cong, Tieying Zhang, and Xiao He. Sam: Database generation from query workloads with supervised autoregressive models. In *Proceedings of the 2022 International Conference on Management of Data*, pages 1542–1555, 2022.

Yinyu Ye and Edison Tse. An extension of karmarkar's projective algorithm for convex quadratic programming. *Mathematical programming*, 44:157–179, 1989.

Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. PrivBayes: Private data release via Bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42(4):1–41, 2017.

Sen Zhang, Weiwei Ni, and Nan Fu. Differentially private graph publishing with degree distribution preservation. *Computers & Security*, 106:102285, 2021.