SYNFD: FUNCTIONAL DEPENDENCY REPAIR FOR TABULAR SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular data synthesis has made substantial progress in improving distributional fidelity. However, for relational tables, which are standard in relational databases, statistically realistic samples are not necessarily structurally valid. For such tables, functional dependencies (FDs) specify deterministic relationships among attributes, which cannot be easily formulated or guaranteed by tabular generative objectives. Motivated by this observation, we propose SYNFD, a framework that repairs synthesized tabular data for FD consistency. SYNFD exploits mapping lookups for seen determinants (left-hand-side value of an FD), runs conditional generation for unseen determinants, and follows topological dependency propagation to process chained FDs. Experiments across multiple tabular datasets show that SYNFD substantially improves FD consistency, achieving an FD-consistent rate of 98.4% on average and a training-mapping agreement rate of 99.8%, while maintaining strong distributional fidelity and downstream utility.

1 INTRODUCTION

Recent tabular generative models have made substantial progress in producing synthetic data with high distributional fidelity, using GAN-, VAE-, and diffusion-based methods (Xu et al., 2019; Stoian et al., 2024; Mueller et al., 2025; Shankar et al., 2026). Most of these methods focus on reproducing marginal distributions, cross-column correlations, and downstream utility, while structural constraints have largely been overlooked.

Table 1: Average FD-consistent rate ($\mathbf{R}_{\mathrm{FD}}$) and training-mapping agreement rate ($\mathbf{R}_{\mathrm{map}}$) across five datasets (%). Higher is better.

Method	$\mathbf{R}_{\mathrm{FD}} \uparrow$	$\mathbf{R}_{\mathrm{map}} \uparrow$
CTGAN (Xu et al., 2019)	39.35	77.33
TVAE (Xu et al., 2019)	<u>68.10</u>	<u>90.51</u>
GaussianCopula (Patki et al., 2016)	49.18	76.71
TabDDPM (Kotelnikov et al., 2023)	63.56	79.87
TabSyn (Zhang et al., 2024)	51.58	83.28
TabDiff (Shi et al., 2025)	54.21	87.39
GATD (Zagardo, 2026)	47.87	80.42
SYNFD (ours)	**98.40**	**99.77**

Structural constraints are particularly important for a special yet commonly used type of tables, i.e., relational tables. Such tables organize records as tuples under a fixed schema, where the attributes describe the fields of each record, and (structural) dependency constraints encode relationships that must hold consistently across the relation and are important for maintaining data integrity and quality (Fan et al., 2008).

A common form of such relationships is functional dependency (FD). FDs are typically written in the form of $X \rightarrow Y$, where X denotes a set of *determinant attributes* and Y denotes a set of *dependent values*. It means that any two tuples that share the same values in attributes X must also share the same values in attributes Y. For example, if a product identifier determines its brand, every tuple with the same identifier should share the same brand. FDs are fundamental constraints in relational data that encode deterministic relationships between attributes (Ge et al., 2021). Violating them can produce tuples that are individually plausible but mutually inconsistent (Umesh et al., 2025).

Strong distributional fidelity does not necessarily imply strong FD consistency. Modern tabular generators model the data distribution probabilistically and do not explicitly enforce the value mapping required by an FD. As a result, the same determinant configuration may be associated with different

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

dependent values across generated tuples. As shown in Table 1, this behavior occurs across several representative tabular generators, which report low FD-consistent rates for their synthesized data. This observation is consistent with recent work emphasizing that structural fidelity should be considered separately from conventional measures of synthetic data quality (Jiang et al., 2026b; Umesh et al., 2025; Liu et al., 2023; Vero et al., 2024).

Recent studies have begun to explicitly consider FDs during tabular synthesis. ITS-GAN (Chen et al., 2019) incorporates FD constraints into a GAN-based framework for augmenting a released sub-table under limited real data. More recently, HFGF (Umesh et al., 2026) generates attributes without dependencies first and then reconstructs dependent attributes according to FD mappings established before synthesis. These approaches demonstrate that explicitly accounting for FDs can improve the consistency of synthetic data. However, reconstruction from an existing mapping relies on the generated determinant configuration being covered by the existing FD mapping, which may not always be available as new and previously unseen determinant configuration could be generated by a tabular data synthesis model.

To fill the gap, we take a generate-then-repair approach. A complete synthetic relation is first generated, after which the specified FDs are repaired. This approach achieves FD preservation without requiring the underlying tabular generator to be redesigned. It also introduces new challenges when multiple FDs interact. Consider two FDs $A \rightarrow B$ and $(B, D) \rightarrow C$. If repairing a tuple for the first FD changes its value in attribute B, the values of the determinant (B, D) used by the second FD also changes. The updated configuration may not have been observed in the training relation. Therefore, post-generation FD repair requires more than applying a fixed determinant-to-dependent lookup: it must use known dependent values when available and resolve determinant configurations for which no observed dependent value exists.

We propose SYNFD, a directed acyclic graph (DAG)-guided framework for FD preservation. We consider a given acyclic FD set and organize them into an attribute-dependency DAG. A base generator first produces a complete synthetic relation. The FDs are then processed in topological order so that updates to attributes in FDs processed first are reflected before later FDs are evaluated. When a determinant configuration is covered by the known FD mapping, its dependent value is obtained through deterministic lookup. If an earlier FD repair produces a configuration for which no dependent value is available, a separately trained mask-conditioned generation model completes the dependent attribute using the current repaired row as context. New dependent values are added to the mapping and reused when the same determinant configuration appears again.

We further analyze why FD preservation is not guaranteed by probabilistic tabular generation. Our analysis characterizes exact FD recovery in terms of the information retained about the dependent attribute and shows that statistical distribution matching alone does not ensure recovery of the reference FD mapping. This provides a theoretical explanation for the empirical gap between distributional fidelity and FD preservation observed in existing generators.

Our main contributions are:

- We formalize functional dependency preservation under probabilistic tabular generation and show that learning a deterministic relationship between the determinant and dependent attributes is not sufficient to recover the reference FD mapping.
- We propose SYNFD, a DAG-guided generate-then-repair framework that processes interacting FDs in topological order and combines deterministic lookup with conditional completion for FD preservation in tabular data synthesis.
- We demonstrate across multiple tabular benchmarks that SYNFD substantially improves FD preservation while maintaining competitive distributional fidelity and utility.

2 PRELIMINARIES

Let $\mathcal{T} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ denote a relational table with d (numerical and categorical) attributes. Tabular synthesis learns a generative distribution $p_\theta(\mathbf{x})$ from $\mathcal{T}$ and generates synthetic tuples $\mathbf{x} \sim p_\theta(\mathbf{x})$.

We assume a given set of functional dependencies (FDs) $\mathcal{F} = \{r_1, \ldots, r_K\}$ imposed on $\mathcal{T}$ that need to be satisfied by the generated tuples. The dependencies may be provided by database schemas or domain knowledge, or discovered from $\mathcal{T}$; FD discovery itself is outside the scope of this work.

Let $[d] = \{1, \ldots, d\}$ denote the attribute index set. For an FD $r : \mathcal{S}_r \to j_r$, $\mathcal{S}_r \subseteq [d]$ denotes its *determinant attributes* and $j_r \in [d] \setminus \mathcal{S}_r$ its *dependent attribute*. Without loss of generality, we consider a single dependent attribute, since an FD with multiple dependent attributes can be decomposed into multiple FDs of the same determinant set. For a tuple $\mathbf{x}$, $\mathbf{x}_{\mathcal{S}_r}$ denotes its *determinant configuration*.

Definition 1 (Functional Dependency)**.** *A relation satisfies a functional dependency $r : \mathcal{S}_r \to j_r$ if, for any two tuples $\mathbf{x}$ and $\mathbf{x}'$ in the relation, $\mathbf{x}_{\mathcal{S}_r} = \mathbf{x}'_{\mathcal{S}_r}$ implies $x_{j_r} = x'_{j_r}$.*

Each unique determinant configuration is associated with a unique dependent value. We represent this dependency by a deterministic mapping (the "*reference mapping*"):

$$f_r : \mathcal{D}_r \to \mathcal{X}_{j_r}, \tag{1}$$

where $\mathcal{D}_r$ denotes the valid determinant domain and X_{j_r} denotes the domain of the dependent attribute j_r. A tuple consistent with the dependency therefore satisfies $x_{j_r} = f_r(\mathbf{x}_{\mathcal{S}_r})$. This formulation also allows composite determinants with $|\mathcal{S}_r| > 1$.

For the theoretical analysis, we restrict to generated determinant configurations within $\mathcal{D}_r$, i.e., $\Pr_\theta[\mathbf{x}_{\mathcal{S}_r} \in \mathcal{D}_r] = 1$. Generation outside this domain is treated separately as a support-mismatch problem. For $\mathbf{x} \sim p_\theta(\mathbf{x})$, we define the FD violation probability as $\epsilon_r(\theta) := \Pr_{\mathbf{x}\sim p_\theta}[x_{j_r} \neq f_r(\mathbf{x}_{\mathcal{S}_r})]$. Exact preservation requires $\epsilon_r(\theta) = 0$ for every specified FD.

Problem statement. Standard tabular synthesis learns a distribution $p_\theta(\mathbf{x})$ that captures the statistical properties of $\mathcal{T}$. Through the relation (i.e., a table) generated by $p_\theta(\mathbf{x})$, we aim to obtain a table of n tuples that all preserve given functional dependencies. We consider acyclic FDs, a well-studied class of dependency structures that admits a topological ordering for processing while still covering many practical relational structures, including composite determinants and dependencies that interact through shared attributes (Beeri et al., 1981; Yannakakis, 1981). For example, in Figure 1, $\texttt{A} \to \texttt{B}$ and $(\texttt{B}, \texttt{D}) \to \texttt{C}$ form an acyclic dependency chain, where the latter has a composite determinant and the two dependencies interact through the shared attribute B.

3 Theoretical Characterization of FD Preservation

We characterize FD preservation in terms of the conditional distribution learned by a tabular data generator. For clarity, consider a single FD $r : \mathcal{S}_r \to j_r$ with reference mapping f_r. For a determinant configuration $\mathbf{s} \in \mathcal{D}_r$, define $q_r(\mathbf{s}) := \Pr_\theta[x_{j_r} = f_r(\mathbf{s}) \mid \mathbf{x}_{\mathcal{S}_r} = \mathbf{s}]$. The quantity $q_r(\mathbf{s})$ measures the probability that the generator produces the reference-consistent dependent value for a given determinant configuration.

Exact FD preservation. For an FD $r : \mathcal{S}_r \to j_r$, the violation probability of a generatore p_θ satisfies $\epsilon_r(\theta) = 1 - \mathbb{E}_{\mathbf{s}\sim p_\theta(\mathbf{x}_{\mathcal{S}_r})}[q_r(\mathbf{s})]$. Straightforwardly, $\epsilon_r(\theta) = 0$ if and only if $q_r(\mathbf{s}) = 1$ for every determinant configuration $\mathbf{s}$ satisfying $p_\theta(\mathbf{x}_{\mathcal{S}_r} = \mathbf{s}) > 0$.

The characterization above shows that exact FD preservation requires more than learning statistical dependencies in the joint data distribution. For every determinant configuration that occurs with positive probability under the generated distribution, the dependent attribute must be equal to the unique reference value $f_r(\mathbf{s})$ with probability 1. Existing tabular generators (e.g., (Zhang et al., 2024; Shi et al., 2025)), however, are typically optimized to approximate the joint data distribution $p_\theta(\mathbf{x})$, without explicitly enforcing this reference-consistency condition for any given FD. Consequently, the induced distribution of the dependent attribute for a given determinant may still assign probability mass to values other than $f_r(\mathbf{s})$, yielding $q_r(\mathbf{s}) < 1$ even when the generated samples exhibit high distributional fidelity.

This raises the question of whether learning a deterministic relationship is sufficient for exact FD preservation. We next characterize this distinction using conditional entropy and show that determinism alone does not guarantee agreement with the reference mapping.

For a generated sample $\mathbf{x} \sim p_\theta$, we use $H_\theta(x_{j_r} \mid \mathbf{x}_{\mathcal{S}_r})$ to denote the conditional Shannon entropy under the distribution induced by $p_\theta(\mathbf{x})$.

Proposition 1 (Conditional determinism and FD preservation)**.** *Suppose the dependent attribute x_{j_r} takes values from a finite alphabet.*

1. *If a tabular data generator preserves the reference mapping of an FD r, i.e., $\epsilon_r(\theta) = 0$, then $H_\theta(x_{j_r} \mid \mathbf{x}_{\mathcal{S}_r}) = 0$.*
2. *The converse does not hold in general. In particular, $H_\theta(x_{j_r} \mid \mathbf{x}_{\mathcal{S}_r}) = 0$ only implies the existence of a deterministic mapping h_r such that $x_{j_r} = h_r(\mathbf{x}_{\mathcal{S}_r})$ almost surely.*
3. *Under conditional determinism, exact preservation additionally requires $h_r(\mathbf{s}) = f_r(\mathbf{s})$ for almost every generated determinant configuration $\mathbf{s}$.*

This proposition distinguishes learning a deterministic relationship from preserving an FD mapping. A tabular data generator may achieve zero conditional entropy while learning a mapping h_r that differs from f_r. Thus, conditional determinism alone is insufficient; the learned mapping must also agree with the dependency specified by an FD. Detailed proof of the proposition is in Appendix A.

Implications. The characterization above shows that FD preservation requires the generated dependent value to agree with $f_r(\mathbf{s})$, while Proposition 1 shows that this condition cannot be reduced to conditional determinism alone, since a deterministic mapping may still differ from f_r. Existing tabular generators do not explicitly enforce this agreement. We propose an algorithm named SYNFD to fill this gap. We take a post-generation data repair approach instead of enforcing the FDs for the generators, because repair can explicitly enforce both deterministic consistency and agreement with the reference mapping, whereas standard generators optimized for distributional learning do *not* naturally guarantee these exact conditions.

4 METHODOLOGY: DAG-GUIDED CONDITIONAL GENERATION

SYNFD takes a generate-then-repair approach, as illustrated in Figure 1. Given an input relational table $\mathcal{T}$, a set of FDs $\mathcal{F}$, and a target number n of tuples to be generated, a base tabular data generator $p_\theta(\mathbf{x})$ is trained on $\mathcal{T}$ and use to generate a synthesized relation $\mathcal{T}_{\text{syn}}^{(0)}$ of n tuples. Each generated tuple is check against every FD $r \in \mathcal{F}$. The tuples consistent with all FDs are added to the final set of output tuples $\mathcal{T}_{\text{syn}}$. The remaining tuples need to be "repaired" for FD consistency.

As there are a set of FDs to repair for, we construct a Directed Acyclic Graph (DAG) over the FDs with the table attributes as the graph nodes and each determinant attribute connected to its dependent attribute as an edge (Section 4.1). The topological order of the DAG establishes the repairing order. For each tuple $\mathbf{x}$ and each FD $r : \mathcal{S}_r \to j_r$ to be repaired for, if the determinant configuration $\mathbf{x}_{\mathcal{S}_r} = \mathbf{s}$ is already seen in $\mathcal{T}$ or a previously processed tuple, the dependent value x_{j_r} is directly fetched from the corresponding reference mapping $\mathcal{M}_r$ and replaces the generated value in $\mathbf{x}$. For efficiency, we maintain a reference mapping $\mathcal{M}_r$ for each FD r, which stores determinant configurations as keys and their corresponding dependent values as values, initialized from $\mathcal{T}$ and updated during repair.

For determinant configurations that are unseen in $\mathcal{T}$ and have not been changed in previous FD repair process, we retain the generated dependent value and add the mapping to $\mathcal{M}_r$. For determinant configurations that are unseen and are created during the repair process for an earlier FD, we mask the dependent value and use a separately trained mask-conditioned generation model p_ϕ to generate the masked values after all FDs at the current topological level are checked (Section 4.2). After all tuples are processed for all FDs, the repaired tuples are added to $\mathcal{T}_{\text{syn}}$, which then contains the target number n of tuples, as no tuples are dropped during repair.

In this work, we instantiate the base generator p_θ using TabDiff (Shi et al., 2025), while the FD repair procedure is designed independently of the specific parameterization of the base generator. Algorithm 2 in Appendix B summarizes the SYNFD FD repair framework.

4.1 TOPOLOGICAL FUNCTIONAL DEPENDENCY REPAIR

Functional Dependency Directed Acyclic Graph Given $\mathcal{F}$, we construct an attribute-level *FD dependency DAG* $\mathcal{G} = (V, E)$ with the set of nodes $V = [d]$. For each FD $r : \mathcal{S}_r \to j_r$ and each attribute $k \in \mathcal{S}_r$, we create an edge $k \to j_r$ and add it to the edge set E. Note that for composite determinants, the attributes in $\mathcal{S}_r$ remain joint during FD lookup. Graph $\mathcal{G}$ is used only to determine the repair order. The DAG induces a topological ordering over the attributes. For FD repair, each

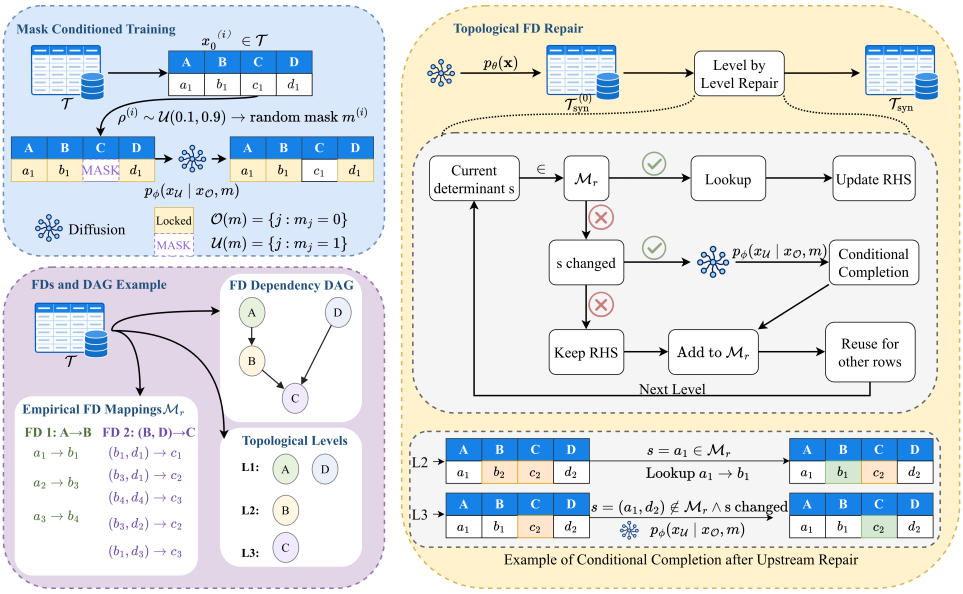


Figure 1: **Overview of the SynFD functional dependency repair framework.** The framework consists of two stages: (1) mask-conditioned diffusion training, which trains a diffusion model p_ϕ to support conditional completion for tuples with FD-inconsistent attributes; and (2) topological FD repair , which takes a relational table $\mathcal{T}$ and a set of FDs as input, ① generates a synthetic table $\mathcal{T}^{(0)_{\text{syn}}}$ using a base generator model p_θ trained on $\mathcal{T}$, and ② checks and repairs each generated tuple for FD consistency following a topological order of the FDs (example order at bottom-left of the figure): known determinant configurations are resolved through reference mapping ($\mathcal{M}_r$) lookups, newly generated configurations that have not be changed in the repair process are added to $\mathcal{M}_r$, and repair-induced new configurations are handled through conditional completion using p_ϕ. dependent attribute is processed only after all attributes in its determinant have been processed. This prevents an FD from being repaired using a determinant that could be changed later for another FD.

Reference Mapping Table Given the training relation $\mathcal{T}$, for each FD $r : \mathcal{S}_r \to j_r$, we initialize a reference mapping table $\mathcal{M}_r$ using all unique determinant configurations and their corresponding dependent values from $\mathcal{T}$. This mapping table is used to correct generated tuples when they are inconsistent with r. It will be expanded as new unique determinant configurations are processed.

Functional Dependency Repair Given a synthetic relation $\mathcal{T}_{\text{syn}}^{(0)}$ generated by a base generator p_θ trained from $\mathcal{T}$, we repair the FDs following the topological order of the dependency DAG level by level. All FDs whose dependent attributes belong to the same level are processed together. When multiple FDs determine the same dependent attribute, we process them in the given order and retain the first assigned value, so that the repair procedure remains deterministic.

Let $\mathbf{x}_i^{(0)}$ denote the initially generated row i in $\mathcal{T}_{\text{syn}}^{(0)}$, and let $\mathbf{x}_i$ denote its current state during repair in $\mathcal{T}_{\text{syn}}$. For an FD $r : \mathcal{S}_r \to j_r$ to be processed, let $\mathbf{s} := \mathbf{x}_{i,\mathcal{S}_r}$ denote the current determinant configuration of row i. The current determinant $\mathbf{s}$ may differ from its initial value $\mathbf{x}_{i,\mathcal{S}_r}^{(0)}$ if some determinant attributes have been modified by FDs processed at earlier topological levels.

If $\mathbf{s} \in \text{Dom}(\mathcal{M}_r)$, we directly set $x_{i,j_r} \leftarrow \mathcal{M}_r(\mathbf{s})$. Here, $\text{Dom}(\mathcal{M}_r)$ denotes the set of determinant configurations already in $\mathcal{M}_r$, and $\mathcal{M}_r(\mathbf{s})$ denotes the corresponding dependent value, which may come from the training relation or an assignment added earlier during repair.

If $\mathbf{s} \notin \text{Dom}(\mathcal{M}_r)$, we compare it with its initial value $\mathbf{x}_{i,\mathcal{S}_r}^{(0)}$. When $\mathbf{s} = \mathbf{x}_{i,\mathcal{S}_r}^{(0)}$, the determinant has not been changed by an earlier repair. Since its dependent value was generated together with this determinant, we retain x_{i,j_r} and add the resulting assignment to the mapping table $\mathcal{M}_r$. When $\mathbf{s} \neq \mathbf{x}_{i,\mathcal{S}_r}^{(0)}$, the determinant has changed during repair, and the original dependent value was gener-

ated under a different determinant configuration. Although retaining this value does not necessarily violate the FD, it may no longer be compatible with the current repaired context and may distort the joint distribution. We therefore mark the dependent attribute as unresolved and defer its completion until all FDs at the current topological level have been processed. The effect of context-aware completion is further evaluated in Appendix D.1.

After all FDs at the current topological level have been processed, the unresolved attributes are masked and completed in a batch by our separately trained mask-conditioned generator model p_ϕ (detailed next), conditioned on the remaining values of the current repaired rows.

4.2 Mask-Conditioned Tabular Generator

We train a separate *mask-conditioned tabular generator* (MCGen, a diffusion model) p_ϕ for local completion when deterministic dependent value lookup is unavailable. We instantiate p_ϕ using the same backbone and mixed-type diffusion process as TabDiff (Shi et al., 2025). For a training tuple $x_0 = (x_{0,1}, \ldots, x_{0,d}) \in \mathcal{T}$, let $m \in \{0,1\}^d$ denote a binary mask, with $\mathcal{O}(m) = \{j : m_j = 0\}$ and $\mathcal{U}(m) = \{j : m_j = 1\}$. The model learns $p_\phi(x_{0,\mathcal{U}} \mid x_{0,\mathcal{O}}, m)$.

We sample a masking ratio $\rho \sim \mathrm{Unif}(\rho_{\min}, \rho_{\max})$, with $\rho_{\min} = 0.1$ and $\rho_{\max} = 0.9$, and randomly mask the corresponding fraction of attributes. We also sample a diffusion time $t \sim \mathrm{Unif}(0,1)$. We optimize ϕ using the underlying diffusion objective, evaluated only on the masked attributes:

$$\mathcal{L}_{\mathrm{mask}}(\phi) = \mathbb{E}_{x_0,t,\rho,m}\left[|\mathcal{U}(m)|^{-1} \sum_{j \in \mathcal{U}(m)} \ell^{(j)}_{\mathrm{diff},\phi}(x_{0,j}, t \mid x_{0,\mathcal{O}}, m)\right], \tag{2}$$

where $\ell^{(j)}_{\mathrm{diff},\phi}$ denotes the corresponding numerical or categorical diffusion loss defined in TabDiff , which we use to form MCGen. During repair, the unresolved dependent attributes at the current topological level are masked, while the remaining attributes of the current repaired rows provide the conditioning context.

The role of p_ϕ is not to enforce FD consistency by itself, but to provide context-conditioned values for repair-induced unseen determinant configurations, whose assignments are then stored in $\mathcal{M}_r$ and reused for consistency. It is only triggered when the current determinant configuration is unseen in $\mathcal{M}_r$ and has been changed by an earlier repair. Since no assignment exists for this configuration, the generated value defines a new mapping in $\mathcal{M}_r$ and is reused later on, thereby preserving consistency with respect to FD r. We use p_ϕ only for local completion rather than full-table generation, so that the joint structure learned by the base generator p_θ is preserved whenever possible and only attributes requiring repair are modified. The detailed training algorithm can be found in Appendix B.

Discussion We note that our repair procedure does not guarantee full FD consistency. When multiple FDs determine the same dependent attribute and impose conflicting assignments, we only retain the value assigned by the first FD examined, which may leave the later FDs unsatisfied. To obtain exactly n fully FD-consistent tuples, the remaining inconsistent tuples can be filtered and additional tuples are generated until n FD-consistent tuples are obtained.

5 Experiments

5.1 Experimental Setup

Datasets We conduct experiments on four real-world tabular datasets with FDs, including Airbnb, Beijing, Hospital, and Migraine, together with a relational benchmark derived from TPC-H. Detailed dataset statistics are provided in Appendix C.

Baselines We compare SynFD with seven popular tabular generative models from four representative families: 1) GAN-based: CTGAN (Xu et al., 2019); 2) VAE-based: TVAE (Xu et al., 2019); 3) statistical: GaussianCopula (Patki et al., 2016); and 4) diffusion-based: TabDDPM (Kotelnikov et al., 2023), TabSyn (Zhang et al., 2024), TabDiff (Shi et al., 2025) and GATD (Zagardo, 2026) . Existing FD- or constraint-aware synthesis methods are not directly applicable to our experimental setting; detailed discussion is provided in Section 6.

Table 2: FD consistency and training-mapping agreement (%, ↑) of different models. Results are reported as mean ± standard deviation over three runs.

Method	Airbnb	Beijing	Hospital	Migraine	TPC-H	Avg.
FD-consistent rate R_{FD}						
CTGAN	5.20±0.33	75.82±0.06	42.70±0.52	72.43±1.99	0.60±0.05	39.35
TVAE	46.92±1.16	90.95±0.12	**100.00±0.00**	**99.89±0.19**	2.75±0.06	68.10
GaussianCopula	4.76±0.00	76.25±0.00	78.83±0.00	86.05±0.00	0.03±0.00	49.18
TabDDPM	**98.63±0.02**	74.80±0.23	45.20±0.44	99.11±0.19	0.03±0.00	63.56
TabSyn	74.16±0.60	80.29±0.11	3.46±0.06	98.34±1.45	1.67±0.09	51.58
TabDiff	51.41±0.92	80.50±0.08	44.37±0.51	90.59±2.00	4.18±0.06	54.21
GATD	0.41±0.07	96.39±0.15	44.35±0.59	98.12±0.38	0.10±0.02	47.87
SynFD	94.71±0.00	**100.00±0.00**	99.95±0.00	97.34±0.00	**100.00±0.00**	**98.40**
Training-mapping agreement rate R_{map}						
CTGAN	69.72±0.36	75.82±0.06	96.33±0.04	98.45±0.33	46.33±0.03	77.33
TVAE	93.64±0.01	90.95±0.12	**100.00±0.00**	**100.00±0.00**	67.96±0.03	90.51
GaussianCopula	73.06±0.00	76.25±0.00	99.73±0.00	99.27±0.00	35.24±0.00	76.71
TabDDPM	**99.89±0.02**	74.80±0.23	97.08±0.03	99.99±0.01	27.59±0.05	79.87
TabSyn	95.94±0.09	80.29±0.11	88.77±0.09	99.96±0.03	51.43±0.04	83.28
TabDiff	94.51±0.16	80.50±0.08	97.06±0.02	99.43±0.02	65.43±0.02	87.39
GATD	69.91±0.05	96.39±0.15	97.61±0.02	99.88±0.01	38.29±0.09	80.42
SynFD	99.11±0.00	**100.00±0.00**	**100.00±0.00**	99.73±0.00	**100.00±0.00**	**99.77**

FD Discovery and Filtering Protocol We discover and preprocess the FDs for each dataset and evaluate FD preservation using the FD-consistent rate R_{FD} (percentage of generated tuples that are fully FD consistent) and training-mapping agreement rate R_{map} (percentage of generated tuples that are FD-consistent when their determinant configurations are seen in the training table; note that generated tuples that are FD-consistent do not necessarily appear in the training table); the detailed FD discovery, filtering, and evaluation protocols are provided in Appendix C.4.

Evaluation Metrics We evaluate table generation quality from four complementary perspectives. **1) Structural fidelity:** We report both R_{FD} and R_{map}, measuring overall FD consistency and agreement with the training FD mappings, respectively. **2) Distributional fidelity:** Following prior studies Zhang et al. (2024); Shi et al. (2025), we evaluate Shape, Trend, α-Precision, and β-Recall to measure how well the synthesized data preserve the real-data distribution and cross-attribute statistics. **3) Downstream utility:** We evaluate machine learning effectiveness using predictive models trained on synthesized data and tested on held-out real data. **4) Repair efficiency:** We report sampling efficiency using the sampling inflation and keep ratio, which quantify the additional sampling effort required to obtain FD-consistent synthetic relations. Detailed definitions of all metrics are provided in Appendix C.

For each method, we independently generate three synthetic relations and report the average performance results. Additional implementation and evaluation details are provided in Appendix C.

5.2 Results

Preserving Functional Dependency Table 2 evaluates FD preservation using both R_{FD} and R_{map}. Although some baselines achieve high FD preservation on individual datasets, their performance varies substantially across datasets because their training objectives focus on fitting the data distribution rather than explicitly preserving FDs. The generally higher R_{map} than R_{FD} further suggests that generative models can often reproduce mappings observed in the training data, but do not maintain them as stable deterministic relationships during generation. In contrast, SynFD achieves high FD-consistent rates (at or close to 100%) across all datasets tested, demonstrating more robust FD preservation. The remaining violations mainly arise when multiple FDs determine the same dependent attribute but require different values for the same tuple. In such cases, our fixed processing order retains the value assigned by the first processed FD, leaving the later FDs unsatisfied.

We further evaluate FD preservation under numerical and mixed-type dependencies on TPC-H, with the detailed setup and results reported in Appendix D.2.

Table 3: Distributional fidelity and downstream utility. Larger values are better for all metrics.

Method	Shape ↑	Trend ↑	α-Precision ↑	β-Recall ↑	F1 ↑	R^2 ↑
CTGAN + Filter	0.7500±0.0038	0.4600±0.0104	0.7117±0.0107	0.3150±0.0169	0.3524±0.0333	-0.4158±0.1787
TVAE + Filter	0.7170±0.0030	0.5501±0.0063	0.4528±0.0080	0.2176±0.0046	**0.7337±0.1140**	-0.5075±0.0694
GaussianCopula + Filter	0.7812±0.0000	0.5128±0.0000	0.4940±0.0000	0.3055±0.0000	0.2732±0.0370	-0.1566±0.2899
TabDDPM + Filter	0.6746±0.0039	0.4721±0.0046	0.3009±0.0026	0.2150±0.0282	0.3017±0.0287	-2.2317±0.2530
TabSyn + Filter	0.8795±0.0024	0.6072±0.0048	0.5810±0.0107	0.3470±0.0080	0.4450±0.0279	0.2900±0.0666
TabDiff + Filter	0.9222±0.0047	0.6923±0.0079	0.7005±0.0121	0.3977±0.0183	0.3682±0.0224	0.3173±0.0203
GATD + Filter	0.3390±0.0124	0.3064±0.0133	0.5072±0.0517	**0.6177±0.0392**	0.5158±0.1499	**0.3764±0.0040**
SYNFD	**0.9555±0.0000**	**0.7987±0.0000**	**0.8748±0.0000**	0.3903±0.0000	0.4198±0.0165	0.3181±0.0819

Table 4: Ablation of conditional completion and HFGF-style reconstruction, averaged across five datasets. Larger values are better for all metrics.

Variant	**Shape ↑**	**Trend ↑**	**α-Prec. ↑**	**β-Rec. ↑**	**F1 ↑**	**R^2 ↑**
w/o Conditional	0.954±0.000	0.793±0.000	0.868±0.000	**0.394±0.000**	0.412±0.007	0.277±0.064
HFGF-style reconstruction	**0.959±0.0000**	0.759±0.0000	0.825±0.0000	0.376±0.0000	**0.489±0.0127**	0.309±0.0103
SYNFD	0.956±0.000	**0.799±0.000**	**0.875±0.000**	0.390±0.000	0.420±0.017	**0.318±0.082**

Preserving Distributional Fidelity and Downstream Utility Enforcing an FD is relatively straightforward if structural validity is the only objective. The more important question is whether structural consistency can be improved without substantially degrading the statistical properties and practical utility of the synthetic data. We report both distributional fidelity and downstream predictive performance results after FD enforcement in Table 3, with additional discussion in Appendix D.2. Here, "+Filter" denotes adding a filtering step to the generated relation by removing tuples that are FD-inconsistent (Appendix C.4).

Overall, SYNFD achieves the highest average Shape, Trend, and α-Precision scores, indicating strong preservation of the overall data distribution after FD enforcement. Although GATD achieves higher β-Recall and R^2, its lower Shape, Trend, and α-Precision indicate weaker overall distributional fidelity. This suggests that filtering can preserve broad coverage or task-relevant patterns even when the retained samples do not closely match the full data distribution. Compared with TabDiff+Filter, SYNFD substantially improves Shape, Trend, α-Precision, and F1, while maintaining comparable R^2 and β-Recall. This suggests that FD repair improves structural and distributional fidelity without noticeably sacrificing coverage or downstream utility.

We further compare SYNFD with TabDiff+Filter under matched FD-valid sample sizes, with the detailed setup and results reported in Appendix D.2.

Ablation and Design Analysis Table 4 compares SYNFD with two alternative repair designs, with detailed per-dataset results provided in Appendix D. Replacing conditional completion with randomly assigning unresolved dependent values ("w/o Conditional") generally degrades Shape, Trend, α-Precision, F1, and R^2, showing the benefit of conditioning on the repaired row context for unseen determinant configurations. We also construct an HFGF-style adaptation (Umesh et al., 2026), which uses the same mixed-type base generator as SYNFD but reconstructs categorical FD-dependent attributes from pre-computed mappings, with unseen determinant configurations handled by empirical random sampling. While this adaptation achieves comparable Shape and competitive F1, it generally yields lower Trend, α-Precision, and β-Recall than SYNFD, suggesting that separately reconstructing dependent attributes may disrupt cross-attribute relationships captured by joint generation. Overall, SYNFD better preserves the learned joint distribution by retaining the generated tuple whenever possible and applying context-aware local repair only when necessary.

To further understand how SYNFD performs FD repair, we report the proportions of FD-processing instances handled by reference mapping, native unseen assignments, and mask-conditioned completion on Airbnb. The detailed statistics are provided in Appendix D.1.

Sampling Efficiency for FD-Consistent Generation We further evaluate efficiency by measuring the *sampling inflation rate*, which quantifies how many synthetic tuples must be generated relative to the target relation size n to obtain n FD-consistent tuples. We additionally report the *FD keep ratio*,

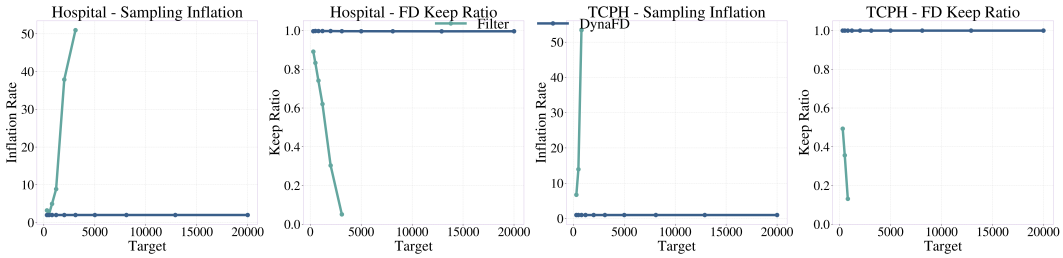


Figure 2: Sampling efficiency of rejection-based filtering and SYNFD on Hospital and TPC-H.

which measures the fraction of generated tuples that are retained after FD filtering. We compare SYNFD with a simple rejection-based filtering (i.e., reject and resample until n FD-consistent tuples are obtained) baseline as detailed in Appendix C.3.

As shown in Figure 2, rejection-based filtering becomes increasingly costly as the target relation grows. With larger synthetic relations, determinant configurations are more frequently repeated, increasing the opportunity to expose inconsistent dependent values produced by the stochastic generator. Consequently, the fraction of tuples retained after FD filtering decreases, and progressively larger generation batches are required before enough FD-valid samples can be obtained. In contrast, SYNFD resolves such inconsistencies by repairing affected attributes in place rather than discarding entire tuples. As a result, it maintains a constantly low sampling overhead as n increases.

6 Related Work

Tabular data synthesis has been extensively studied. Early methods typically rely on statistical dependency modeling (Patki et al., 2016), while subsequent work has explored GAN- and VAE-based approaches (Park et al., 2018; Xu et al., 2019; Zhao et al., 2021; 2023). Another line of work formulates tabular synthesis as autoregressive sequence generation (Borisov et al., 2023; Solatorio & Dupriez, 2023; Ding et al., 2025; Neto, 2026). More recently, diffusion-based generative models have become a prominent direction (Kim et al., 2023; Kotelnikov et al., 2023; Lee et al., 2023; Zhang et al., 2024; Shi et al., 2025; Lin et al., 2025; Jiang et al., 2026a), TabDDPM (Kotelnikov et al., 2023). Our work additionally requires the generated relation to preserve functional dependencies.

Several studies have considered constraint- or dependency-aware tabular synthesis. ITS-GAN (Chen et al., 2019) studies a bounded-data augmentation setting that assumes access to a released subtable together with statistics of the full table. Kamino (Ge et al., 2021) incorporates user-specified integrity constraints into differentially private synthesis. CuTS (Vero et al., 2024) supports row-wise logical constraints over discretized features, but does not directly support cross-row functional dependencies requiring identical determinant configurations to share the same dependent value. HFGF (Umesh et al., 2026) explicitly preserves functional dependencies by first generating root attributes and then reconstructing dependent attributes from pre-computed mappings. However, its mapping-based reconstruction cannot directly handle unseen determinant configurations. We therefore use an HFGF-style adaptation in the ablation study rather than directly including HFGF as a main baseline.

7 Conclusion

We studied FD preservation in tabular data synthesis and showed that strong distributional fidelity does not necessarily imply strong FD preservation. To address this gap, we proposed SYNFD, a generate-then-repair framework that separates distribution learning from FD repair. We start with a base generator to produce a complete synthetic relation, after which the given FDs are repaired following a topological order. Dependent values of determinant compositions that are seen in training data are simply retrieved from the training data, while those for unseen determinants are completed using a mask-conditioned model. Across real-world datasets and a popular database benchmark, TPC-H, SYNFD substantially improves FD preservation, achieving an average FD-consistent rate of 98.4% and a training-mapping agreement rate of 99.8%, while maintaining competitive distributional fidelity and downstream utility.

AI USE STATEMENT

We used generative AI tools to assist in the writing of proofs and to formulate mathematical claims. Additionally, we used generative AI tools to create or edit software code. We have reviewed all AI-assisted work. AI-assisted mathematical claims and proofs were checked by the authors to ensure consistency with the theoretical framework and assumptions developed in this work. AI-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Ahmed M. Alaa, Boris van Breugel, Evgeny Saveliev, and Mihaela van der Schaar. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. In *ICML*, pp. 290–306, 2022.

Catriel Beeri, Ronald Fagin, David Maier, Alberto Mendelzon, Jeffrey Ullman, and Mihalis Yannakakis. Properties of acyclic database schemes. In *ACM Symposium on Theory of Computing*, pp. 355–362, 1981.

Vadim Borisov, Kathrin Sessler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. In *ICLR*, 2023.

Haipeng Chen, Sushil Jajodia, Jing Liu, Noseong Park, Vadim Sokolov, and V. S. Subrahmanian. FakeTables: Using GANs to generate functional dependency preserving tables with bounded real data. In *IJCAI*, pp. 2074–2080, 2019.

Jiayuan Ding, Jianhui Lin, Shiyu Jiang, Yixin Wang, Ziyang Miao, Zhaoyu Fang, Jiliang Tang, Min Li, and Xiaojie Qiu. Tabula: A tabular self-supervised foundation model for single-cell transcriptomics. In *NeurIPS*, 2025.

Wenfei Fan, Floris Geerts, and Xibei Jia. A revival of integrity constraints for data cleaning. *PVLDB*, 1(2):1522–1523, 2008.

Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F. Ilyas. Kamino: constraint-aware differentially private data synthesis. *Proc. VLDB Endow.*, 14(10):1886–1899, 2021.

Xiangjian Jiang, Mingxuan Liu, Nikola Simidjievski, Tassilo Klein, and Mateja Jamnik. Tabular foundation model for generative modelling, 2026a.

Xiangjian Jiang, Nikola Simidjievski, and Mateja Jamnik. Tabstruct: Measuring structural fidelity of tabular data. In *ICLR*, 2026b.

Jayoung Kim, Chaejeong Lee, and Noseong Park. STaSy: Score-based tabular data synthesis. In *ICLR*, 2023.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. TabDDPM: Modelling tabular data with diffusion models. In *ICML*, pp. 17564–17579, 2023.

Chaejeong Lee, Jayoung Kim, and Noseong Park. CoDi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis. In *ICML*, pp. 18940–18956, 2023.

Xiaofeng Lin, Chenheng Xu, Matthew Yang, and Guang Cheng. CTSyn: A foundation model for cross tabular data generation. In *ICLR*, 2025.

Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. GOGGLE: Generative modelling for tabular data by learning relational structure. In *ICLR*, 2023.

Markus Mueller, Kathrin Gruber, and Dennis Fok. Continuous diffusion for mixed-type tabular data. In *ICLR*, 2025.

Elias Chaibub Neto. Using maximal information auxiliary variables to improve synthetic data generation based on tabPFN foundation models. In *ICLR*, 2026.

Thorsten Papenbrock and Felix Naumann. A hybrid approach to functional dependency discovery. In *SIGMOD*, pp. 821–833, 2016.

Marcel Parciak, Sebastiaan Weytjens, Niel Hens, Frank Neven, Liesbet M. Peeters, and Stijn Vansummeren. Measuring approximate functional dependencies: a comparative study. *The VLDB Journal*, 34(4), 2025.

Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, 2018.

Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *DSAA*, pp. 399–410, 2016.

Aditya Shankar, Yuandou Wang, Rihan Hai, and Lydia Y. Chen. Harpoon: Generalised manifold guidance for conditional tabular diffusion. In *ICLR*, 2026.

Juntong Shi, Minkai Xu, Harper Hua, Hengrui Zhang, Stefano Ermon, and Jure Leskovec. Tabdiff: a mixed-type diffusion model for tabular data generation. In *ICLR*, 2025.

Aivin V. Solatorio and Olivier Dupriez. Realtabformer: Generating realistic relational and tabular data using transformers. *arXiv preprint arXiv:2302.02041*, 2023.

Shaoxu Song and Lei Chen. Differential dependencies: Reasoning and discovery. *ACM Trans. Database Syst.*, 36(3), 2011.

Mihaela C Stoian, Salijona Dyrmishi, Maxime Cordy, Thomas Lukasiewicz, and Eleonora Giunchiglia. How realistic is your synthetic data? constraining deep generative models for tabular data. In *ICLR*, 2024.

Chaithra Umesh, Kristian Schultz, Manjunath Mahendra, Saptarshi Bej, and Olaf Wolkenhauer. Preserving logical and functional dependencies in synthetic tabular data. *Pattern Recognition*, 163:111459, 2025.

Chaithra Umesh, Kristian Seegel-Schultz, Manjunath Mahendra, Saptarshi Bej, and Olaf Wolkenhauer. Dependency-aware synthetic tabular data generation. *Pattern Recognition*, 179:113819, 2026.

Mark Vero, Mislav Balunovic, and Martin Vechev. CuTS: Customizable tabular synthetic data generation. In *ICML*, 2024.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. In *NeurIPS*, pp. 7335–7345, 2019.

Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, 1981.

David Turtora Zagardo. Geometry-aware tabular diffusion. In *ICML*, 2026.

Hengrui Zhang, Jiani Zhang, Balasubramaniam Srinivasan, Zhengyuan Shen, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *ICLR*, 2024.

Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y. Chen. CTAB-GAN: Effective table data synthesizing. In *ACML*, pp. 97–112, 2021.

Zilong Zhao, Aditya Kunar, Robert Birke, Hiek Van der Scheer, and Lydia Y Chen. CTAB-GAN+: Enhancing tabular data synthesis. *Frontiers in Big Data*, 6:1296508, 2023.

A Proofs and Additional Theoretical Details

We provide additional details for the characterization of exact FD preservation and the proof of Proposition 1. Recall that, for an FD $r : \mathcal{S}_r \to j_r$, the corresponding dependency is represented by a deterministic reference mapping $f_r : \mathcal{D}_r \to \mathcal{X}_{j_r}$. Throughout the theoretical analysis, we assume that generated determinant configurations lie within the valid domain, i.e., $\Pr_\theta[\mathbf{x}_{\mathcal{S}_r} \in \mathcal{D}_r] = 1$.

For a generated tuple $\mathbf{x} \sim p_\theta(\mathbf{x})$, the FD violation probability is $\epsilon_r(\theta) := \Pr_\theta[x_{j_r} \neq f_r(\mathbf{x}_{\mathcal{S}_r})]$. For each determinant configuration $\mathbf{s} \in \mathcal{D}_r$, we define $q_r(\mathbf{s}) := \Pr_\theta[x_{j_r} = f_r(\mathbf{s}) \mid \mathbf{x}_{\mathcal{S}_r} = \mathbf{s}]$. When the determinant contains continuous attributes, the conditional probability is understood through a regular conditional distribution.

A.1 Exact FD Preservation

By conditioning on the generated determinant configuration and applying the law of total expectation, we obtain

$$\begin{aligned}\epsilon_r(\theta) &= \Pr_\theta\left[x_{j_r} \neq f_r(\mathbf{x}_{\mathcal{S}_r})\right] \\ &= \mathbb{E}_{\mathbf{s}\sim p_\theta(\mathbf{x}_{\mathcal{S}_r})}\left[\Pr_\theta\left[x_{j_r} \neq f_r(\mathbf{s}) \mid \mathbf{x}_{\mathcal{S}_r} = \mathbf{s}\right]\right].\end{aligned}$$

By the definition of $q_r(\mathbf{s})$,

$$\Pr_\theta\left[x_{j_r} \neq f_r(\mathbf{s}) \mid \mathbf{x}_{\mathcal{S}_r} = \mathbf{s}\right] = 1 - q_r(\mathbf{s}),$$

and therefore

$$\epsilon_r(\theta) = 1 - \mathbb{E}_{\mathbf{s}\sim p_\theta(\mathbf{x}_{\mathcal{S}_r})}\left[q_r(\mathbf{s})\right].$$

Since $1 - q_r(\mathbf{s}) \geq 0$, we have $\epsilon_r(\theta) = 0$ if and only if $q_r(\mathbf{s}) = 1$ for every determinant configuration $\mathbf{s}$ satisfying $p_\theta(\mathbf{x}_{\mathcal{S}_r} = \mathbf{s}) > 0$. Thus, exact FD preservation requires the generated dependent value to agree with $f_r(\mathbf{s})$ with probability one for every determinant configuration that occurs with positive probability under the generated distribution.

A.2 Proof of Proposition 1

We prove the three statements in Proposition 1 in order.

First, suppose that the tabular data generator exactly preserves the FD, so that $\epsilon_r(\theta) = 0$. From the characterization above, $q_r(\mathbf{s}) = 1$ for every determinant configuration $\mathbf{s}$ satisfying $p_\theta(\mathbf{x}_{\mathcal{S}_r} = \mathbf{s}) > 0$. Hence, conditioned on $\mathbf{x}_{\mathcal{S}_r} = \mathbf{s}$, the dependent attribute equals $f_r(\mathbf{s})$ with probability one. Since x_{j_r} takes values in a finite alphabet, its conditional distribution is concentrated on a single outcome, and therefore

$$H_\theta\left(x_{j_r} \mid \mathbf{x}_{\mathcal{S}_r}\right) = 0.$$

Second, the converse does not hold in general. If

$$H_\theta\left(x_{j_r} \mid \mathbf{x}_{\mathcal{S}_r}\right) = 0,$$

since x_{j_r} is finite-valued, for every determinant configuration $\mathbf{s}$ satisfying $p_\theta(\mathbf{x}_{\mathcal{S}_r} = \mathbf{s}) > 0$, the conditional distribution of x_{j_r} given $\mathbf{x}_{\mathcal{S}_r} = \mathbf{s}$ is concentrated on a single outcome. Therefore, there exists a deterministic mapping h_r such that

$$x_{j_r} = h_r(\mathbf{x}_{\mathcal{S}_r})$$

with probability one under p_θ.

However, this condition does not require h_r to coincide with f_r. If there exists a determinant configuration $\mathbf{s}$ such that $p_\theta(\mathbf{x}_{\mathcal{S}_r} = \mathbf{s}) > 0$ and $h_r(\mathbf{s}) \neq f_r(\mathbf{s})$, then the generated relationship remains deterministic and therefore has zero conditional entropy, while tuples with determinant configuration $\mathbf{s}$ violate the specified FD mapping with positive probability.

Finally, under conditional determinism, exact preservation therefore requires $h_r(\mathbf{s}) = f_r(\mathbf{s})$ for every determinant configuration $\mathbf{s}$ satisfying $p_\theta(\mathbf{x}_{\mathcal{S}_r} = \mathbf{s}) > 0$. Hence, zero conditional entropy guarantees only the existence of a deterministic generated mapping, whereas exact FD preservation additionally requires that this mapping agree with f_r. □

Algorithm 1 MCGEN Training

Require: Training relation $\mathcal{T}$, mask-conditioned diffusion model p_ϕ, masking range $[\rho_{\min}, \rho_{\max}]$
Ensure: Trained conditional model p_ϕ

1: **for** each training sample $x_0 \sim \mathcal{T}$ **do**
2: Sample masking ratio $\rho \sim \mathrm{Unif}(\rho_{\min}, \rho_{\max})$
3: Sample binary mask $m \in \{0,1\}^d$ with masking ratio ρ
4: Define $\mathcal{O} = \{j : m_j = 0\}$ and $\mathcal{U} = \{j : m_j = 1\}$
5: Sample diffusion time $t \sim \mathrm{Unif}(0,1)$
6: Compute the mask-conditioned diffusion loss

$$\mathcal{L}_{\text{mask}}(\phi) \leftarrow \frac{1}{|\mathcal{U}|} \sum_{j \in \mathcal{U}} \ell^{(j)}_{\text{diff},\phi}\left(x_{0,j}, t \mid x_{0,\mathcal{O}}, m\right)$$

7: Update ϕ using $\nabla_\phi \mathcal{L}_{\text{mask}}(\phi)$
8: **end for**
9: **return** p_ϕ

Table 5: Statistics of the datasets. # Rows denotes the number of tuples, # Num and # Cat denote the numbers of numerical and categorical attributes, respectively, and # FDs denotes the number of functional dependencies used in our experiments, and Task denotes the downstream prediction task.

Dataset	# Rows	# Num	# Cat	# FDs	Task
Airbnb	3,048	4	14	28	Regression
Beijing	43,824	7	5	1	Regression
Hospital	5,419	17	19	154	Classification
Migraine	377	1	21	138	Classification
TPC-H	30,201	6	25	17	Classification

B ALGORITHMS

This section presents the detailed algorithms for training the mask-conditioned generator (Algorithm 1) and performing DAG-guided FD repair (Algorithm 2).

C DETAILS OF EXPERIMENTAL SETUPS

C.1 DATASETS

We evaluate SYNFD on five tabular datasets spanning different application domains: Airbnb, Beijing, Hospital, Migraine, and TPC-H. The first four are real-world tabular datasets containing mixed numerical and categorical attributes, while TPC-H is derived from the TPC-H database benchmark and contains schema-level structural dependencies. Table 5 summarizes the dataset statistics.

Below, we briefly describe each dataset.

- **Airbnb.** The Airbnb dataset is obtained from the *Airbnb New User Bookings* Kaggle competition.[1] It contains user demographic and account-related attributes for predicting the destination of a new user's first booking. Following the experimental setting in previous work (Umesh et al., 2025), we use a subset containing 3,048 rows.
- **Beijing.** The Beijing dataset is obtained from the UCI Machine Learning Repository.[2] It contains hourly air-pollution and meteorological measurements in Beijing, including $PM_{2.5}$ concentrations and weather-related attributes.

[1] https://www.kaggle.com/competitions/airbnb-recruiting-new-user-bookings/overview

[2] https://archive.ics.uci.edu/

Algorithm 2 SYNFD FD Repair

Require: Training relation $\mathcal{T}$, initial synthetic relation $\mathcal{T}_{\text{syn}}^{(0)}$, acyclic FD set $\mathcal{F}$, empirical mappings $\{\mathcal{M}_r^{\text{train}}\}_{r\in\mathcal{F}}$, mask-conditioned model p_ϕ
Ensure: Repaired synthetic relation $\mathcal{T}_{\text{syn}}$

$\mathcal{T}_{\text{syn}} \leftarrow \mathcal{T}_{\text{syn}}^{(0)}$
Construct the FD dependency DAG and its topological levels $\mathcal{L}_1, \ldots, \mathcal{L}_H$
Fix a deterministic processing order over FDs within each level
for each FD $r \in \mathcal{F}$ **do**
    Compute $\mathcal{M}_r$ from $\mathcal{T}$
end for
for $\ell = 1, \ldots, H$ **do**
    $\mathcal{P} \leftarrow \emptyset$ ▷ repair-induced unresolved entries
    $\mathcal{C} \leftarrow \emptyset$ ▷ claimed dependent cells
    for each FD $r : \mathcal{S}_r \to j_r$ with $j_r \in \mathcal{L}_\ell$, in the given order **do**
        for each row $\mathbf{x}_i \in \mathcal{T}_{\text{syn}}$ **do**
            if $(i, j_r) \notin \mathcal{C}$ **then**
                $\mathbf{s} \leftarrow \mathbf{x}_{i,\mathcal{S}_r}$
                if $\mathbf{s} \in \text{Dom}(\mathcal{M}_r)$ **then**
                    $x_{i,j_r} \leftarrow \mathcal{M}_r(\mathbf{s})$
                else if $\mathbf{s} = \mathbf{x}_{i,\mathcal{S}_r}^{(0)}$ **then**
                    $\mathcal{M}_r(\mathbf{s}) \leftarrow x_{i,j_r}$ ▷ native unseen
                else
                    Add $(i, r, \mathbf{s})$ to $\mathcal{P}$ ▷ repair-induced unseen
                end if
                $\mathcal{C} \leftarrow \mathcal{C} \cup \{(i, j_r)\}$ ▷ first applicable FD takes precedence
            end if
        end for
    end for
    if $\mathcal{P} \neq \emptyset$ **then**
        Mask all dependent attributes indicated by $\mathcal{P}$ in $\mathcal{T}_{\text{syn}}$ and complete them jointly using p_ϕ
        Group $\mathcal{P}$ by $(r, \mathbf{s})$
        for each group associated with $(r, \mathbf{s})$ **do**
            Select one completed value v
            $\mathcal{M}_r(\mathbf{s}) \leftarrow v$
            Assign v to the dependent attribute of all rows in the group
        end for
    end if
end for
return $\mathcal{T}_{\text{syn}}$

- **Hospital.** The Hospital dataset is obtained from the Centers for Medicare & Medicaid Services (CMS) Provider Data repository.[3] The dataset contains hospital-related information, including attributes describing hospital identity, location, and overall hospital rating.
- **Migraine.** The Migraine dataset is obtained from the *Migraine Symptom Dataset for Classification* on Kaggle.[4] It contains symptom-related and diagnostic attributes associated with migraine classification.
- **TPC-H.** TPC-H is a standard decision-support database benchmark containing explicitly defined relational semantics and schema-level dependencies. [5] We construct a single-table benchmark from the relevant TPC-H attributes and use its known semantics to specify functional dependencies (detailed next).

[3] https://data.cms.gov/provider-data/dataset/xubh-q36u#overview
[4] https://www.kaggle.com/
[5] https://www.tpc.org/tpch/

Algorithm 3 Construction of the TPC-H Single-Table Benchmark

Require: Original TPC-H relations $\mathcal{R}$ and relational schema $\mathcal{S}$
Ensure: Flattened relation $\mathcal{T}$ and FD set $\mathcal{F}$
1: Generate the original TPC-H relations
2: Select the lowest-level transactional relation as the central table
3: $\mathcal{T} \leftarrow$ central table
4: **for** each related entity reachable through a primary–foreign key relationship **do**
5: Join the entity attributes into $\mathcal{T}$ according to $\mathcal{S}$
6: **end for**
7: Initialize $\mathcal{F} \leftarrow \emptyset$
8: **for** each primary–foreign key relationship preserved in $\mathcal{T}$ **do**
9: Identify the entity attributes determined by the corresponding identifier
10: Add the induced categorical dependencies to $\mathcal{F}$
11: **end for**
12: **return** $\mathcal{T}, \mathcal{F}$

Construction of TPC-H TPC-H is originally a multi-table relational benchmark, whereas our experiments focus on single-table synthesis. We therefore flatten the original TPC-H relations into a single wide table by following the primary-foreign key relationships defined in the benchmark schema. This transformation preserves the deterministic entity relationships in the original database while converting TPC-H into our single-table setting. We construct the initial FD set directly from the predefined primary-foreign key relationships: attributes belonging to the same entity are functionally determined by the corresponding entity identifier after flattening. This provides a natural schema-derived FD set without relying on dependency discovery from a particular sampled training instance. The construction procedure is summarized in Algorithm 3.

To further evaluate whether our framework extends beyond categorical FDs, we additionally select three semantically meaningful dependencies involving continuous attributes. The first captures that an order uniquely determines its total price, `l_orderkey` $\rightarrow$ `o_totalprice`. The second captures that a part uniquely determines its size, `l_partkey` $\rightarrow$ `p_size`. The third represents a mixed-type dependency, (`l_partkey`, `l_quantity`) $\rightarrow$ `l_extendedprice`, where the extended price is determined jointly by the part identity and ordered quantity. These additional dependencies are used specifically to evaluate FD preservation when continuous attributes appear in either the determinant or the dependent side.

C.2 Baselines

We brief the baseline tabular generation methods considered in our experiments.

- **CTGAN and TVAE** (Xu et al., 2019) are two representative deep generative models for mixed-type tabular data. CTGAN is based on Generative Adversarial Networks (GANs), whereas TVAE adopts a Variational AutoEncoder (VAE). Both models introduce mode-specific normalization to better model complex numerical distributions. CTGAN additionally employs conditional training and sampling to handle imbalanced categorical attributes and improve the modeling of heterogeneous tabular distributions.
- **GaussianCopula** (Patki et al., 2016) is a classical statistical approach for synthetic tabular data generation. It estimates the marginal distribution of each attribute separately and models dependencies among attributes through a Gaussian copula. Synthetic tuples are generated by first sampling from the learned multivariate copula and then transforming each dimension back to its corresponding marginal distribution. GaussianCopula provides a simple non-neural baseline for modeling tabular dependencies.
- **TabDDPM** (Kotelnikov et al., 2023) is a diffusion-based tabular generator designed for mixed numerical and categorical attributes. Numerical attributes are modeled using Gaussian diffusion, while categorical attributes are modeled using multinomial diffusion. The two types of attributes are jointly processed by a shared denoising network, allowing the model to capture dependencies across heterogeneous columns.

- **TabSyn** (Zhang et al., 2024) performs synthesis through latent diffusion. It first trains a VAE to transform mixed-type tabular tuples into a continuous latent representation. A diffusion model is then trained in this latent space, and sampled latent vectors are decoded back into tabular tuples through the VAE decoder. By conducting diffusion in a compact continuous representation, TabSyn improves both sampling quality and computational efficiency compared with directly applying diffusion in the original mixed-type data space.
- **TabDiff** (Shi et al., 2025) is a diffusion-based generative model that directly models mixed-type tabular data. It jointly learns the distributions of numerical and categorical attributes within a unified diffusion framework, allowing information to be exchanged across columns during denoising.
- **GATD** (Zagardo, 2026) is a geometry-aware diffusion model for mixed-type tabular data generation. It explicitly models inter-column relationships using pairwise geometric features, including angles and lengths computed from normalized column values. These features are incorporated as model inputs and auxiliary prediction targets during diffusion training, providing explicit supervision for learning dependencies between columns.

For all baseline generators, we use the default hyperparameter settings provided by their original implementations. We additionally evaluate an FD-filtered variant, denoted by "+ Filter", where the baseline model first generates synthetic tuples using its original sampling procedure, after which tuples violating any specified FD are removed. When a fixed number of FD-consistent tuples is required, sampling is repeated until the number of retained FD-consistent tuples matches the target size. This provides a common post-processing baseline for comparing distributional fidelity and downstream utility under the same structural validity requirement.

C.3 METRICS

We evaluate synthetic data from multiple perspectives:

Shape and Trend We use the Shape and Trend scores implemented in SDMetrics[6] to evaluate marginal and pairwise distributional similarity, respectively. Shape measures how closely each synthetic column matches its real-data counterpart. For numerical attributes, similarity is computed from the Kolmogorov-Smirnov statistic between the real and synthetic empirical distributions, while categorical attributes are compared using total variation distance. The resulting column-wise similarities are averaged across attributes.

Trend evaluates whether pairwise relationships between attributes are preserved. For numerical attribute pairs, it compares the corresponding Pearson correlations in the real and synthetic data. For categorical pairs, it compares their contingency tables using a total-variation-based similarity. Higher Shape and Trend scores therefore indicate better preservation of the marginal distributions and pairwise dependencies, respectively.

α-Precision and β-Recall We further evaluate sample-level fidelity and coverage using α-Precision and β-Recall (Liu et al., 2023; Alaa et al., 2022). α-Precision measures whether synthetic samples lie within regions supported by the real-data distribution, and thus reflects the fidelity of generated samples. In contrast, β-Recall measures how well the synthetic distribution covers the support of the real data. Higher values indicate better fidelity and coverage, respectively.

Downstream Utility We evaluate whether the statistical relationships learned from the synthetic data remain useful for downstream prediction. For each dataset, a predictive model is trained using only the synthetic training relation and evaluated on the held-out real test set. We report F1 score for classification tasks and R^2 for regression tasks. The same preprocessing and downstream evaluation protocol are used for all generative methods, so differences in performance reflect the utility of the corresponding synthetic data.

FD-consistent Rate To quantify structural consistency, we measure the fraction of generated tuples that can be retained after enforcing all specified FDs. Let $\mathcal{T}_{\text{syn}}$ denote a generated relation.

[6] https://docs.sdv.dev/sdmetrics

For each FD $r : \mathcal{S}_r \to j_r$, we group the current relation by the determinant configuration $\mathbf{x}_{\mathcal{S}_r}$. For a determinant configuration $\mathbf{s}$, we identify the most frequently generated dependent value as $\hat{f}_r(\mathbf{s}) = \arg\max_y N_r(\mathbf{s}, y)$, where $N_r(\mathbf{s}, y)$ denotes the number of tuples in the current relation with $\mathbf{x}_{\mathcal{S}_r} = \mathbf{s}$ and $x_{j_r} = y$. For ties, we select the first encountered dependent value. We then retain only tuples satisfying $x_{j_r} = \hat{f}_r(\mathbf{x}_{\mathcal{S}_r})$.

The procedure is applied sequentially to all specified FDs, with each subsequent dependency evaluated on the relation retained by the preceding steps. All methods use the same fixed FD order given by the input dependency list. Let $\mathcal{T}_{\mathrm{FD}}$ denote the final filtered relation. We define the FD-consistent rate as $R_{\mathrm{FD}} = \frac{|\mathcal{T}_{\mathrm{FD}}|}{|\mathcal{T}_{\mathrm{syn}}|} \times 100\%$. A higher retention rate indicates that a larger fraction of the generated relation is already compatible with a single-valued assignment for all specified dependencies and therefore requires less tuple removal to obtain an FD-consistent relation.

Training-mapping Agreement Rate While R_{FD} measures internal consistency within the generated relation, it does not determine whether the generated mapping agrees with the FD mapping observed in the training data. We therefore additionally report the training-mapping agreement rate R_{map}. For each FD r, let $\mathcal{D}_r^{\mathrm{train}} = \mathrm{Dom}(\mathcal{M}_r^{\mathrm{train}})$ denote the set of determinant configurations covered by the training mapping. For generated tuples with $\mathbf{x}_{\mathcal{S}_r} \in \mathcal{D}_r^{\mathrm{train}}$, we compute $R_{\mathrm{map}}^{(r)} = N_{\mathrm{agree}}^{(r)} / N_{\mathrm{observed}}^{(r)} \times 100\%$, where $N_{\mathrm{observed}}^{(r)}$ denotes the number of generated tuples whose determinant configurations belong to $\mathcal{D}_r^{\mathrm{train}}$, and $N_{\mathrm{agree}}^{(r)}$ denotes the number of these tuples satisfying $x_{j_r} = \mathcal{M}_r^{\mathrm{train}}(\mathbf{x}_{\mathcal{S}_r})$. We then report $R_{\mathrm{map}} = \frac{1}{K} \sum_{r=1}^{K} R_{\mathrm{map}}^{(r)}$. A higher R_{map} indicates stronger agreement between the generated dependent values and those observed in the training relation.

Sampling Inflation Rate A simple alternative to repairing FD violations is rejection-based filtering. Given a target relation size N_{target}, we first generate a batch of synthetic tuples and apply FD filtering. If the filtered batch contains fewer than N_{target} valid tuples, we increase the requested batch size by default $1.3x$ and regenerate a new batch from scratch. This process is repeated until a single generation round retains at least N_{target} FD-valid tuples. We regenerate each batch from scratch rather than accumulating FD-valid tuples from previous rounds, so that each round evaluates whether the requested batch size is sufficient to produce the target relation in a single generation. For SynFD, the same inflation procedure is applied if fewer than N_{target} tuples remain FD-consistent after repair. Thus, its sampling inflation rate may also exceed 1 (e.g., 1.3), but mask-conditioned completion itself is not counted as generating an additional complete tuple.

We measure the resulting sampling cost using the *sampling inflation rate*, defined as $\rho_s = \sum_{r=1}^{R} N_{\mathrm{generated}}^{(r)} / N_{\mathrm{target}}$, where R is the first successful generation round. We additionally report the *final-round keep ratio*, $N_{\mathrm{valid}}^{(R)} / N_{\mathrm{generated}}^{(R)}$, which measures the fraction of tuples retained in the successful round. To avoid prohibitively expensive sampling, we terminate an experiment once the sampling inflation exceeds $50\times$ the target relation size.

C.4 Implementation Details

All experiments are implemented in PyTorch and conducted on a single NVIDIA A100 GPU with 80 GB GPU memory and 32 GB system memory. Unless otherwise specified, all reported results are averaged over three independent runs.

FD Discovery and Filtering Protocol There are limited datasets that come with known FDs. For the four real-world datasets, we discover FDs from the categorical attributes of the training relation using HyFD, an algorithm that computes FDs from relational tuples (Papenbrock & Naumann, 2016). After FD discovery with HyFD, we remove cyclic dependencies and retain an acyclic FD subset for DAG construction, prioritizing shorter dependencies when breaking cycles. We focus on categorical attributes because classical FD discovery relies on exact value equality, which can be overly restrictive for continuous attributes affected by measurement noise or small numerical variations (Song & Chen, 2011; Parciak et al., 2025). For TPC-H, we first flatten the original relational schema into a single table through its primary-foreign key relationships to match our single-table synthesis setting, and then derive FDs from these primary-foreign key relationships.

For each method, we generate a synthetic relation containing the same number of tuples as the corresponding training relation and apply a common FD filtering procedure that removes FD-inconsistent tuples until the retained relation satisfies the dependencies.

We report the **FD-consistent rate**, $R_{\mathrm{FD}} = N_{\mathrm{consistent}}/N_{\mathrm{generated}} \times 100\%$, where $N_{\mathrm{generated}}$ and $N_{\mathrm{consistent}}$ denote the numbers of generated and retained tuples, respectively. A higher retention rate indicates that a larger fraction of the original synthetic relation is consistent to the specified FD structure.

We additionally report the **training-mapping agreement rate**, R_{map}, which measures how well generated tuples follow the FD mappings observed in the training relation. We define $R_{\mathrm{map}} = \frac{1}{K}\sum_{r=1}^{K} \frac{N_{\mathrm{agree}}^{(r)}}{N_{\mathrm{observed}}^{(r)}} \times 100\%$, where K is the number of FDs, $N_{\mathrm{observed}}^{(r)}$ is the number of generated tuples whose determinant configurations are observed in the training relation, and $N_{\mathrm{agree}}^{(r)}$ is the number of these tuples whose dependent values match the corresponding training assignments. A higher R_{map} indicates stronger agreement with the FD mappings observed in the training data.

Base Generator We build SynFD on top of TabDiff (Shi et al., 2025). The original TabDiff model is used as the unconditional generator p_θ to produce the initial synthetic relation. We retain the original data preprocessing, denoising architecture, diffusion process, optimization settings, and sampling configuration of TabDiff.

MCGen The mask-conditioned model p_ϕ uses the same backbone architecture and diffusion configuration as the base TabDiff model, but is trained with the masking objective introduced in Section 4.2. For each training tuple, the masking ratio is sampled as $\rho \sim \mathcal{U}(0.1, 0.9)$, and the corresponding attributes are randomly selected as reconstruction targets. At inference time, p_ϕ is invoked only when the DAG repair encounters an unresolved determinant configuration; the dependent attribute is masked while the current repaired row is provided as conditioning context.

D Additional Experiments

D.1 Detailed Ablation and Design Analysis

For the conditional-completion ablation, we use the same base generator and FD repair procedure as SynFD, but replace mask-conditioned completion for repair-induced unseen determinant configurations with random assignment of the dependent value.

For the HFGF-style reconstruction, we use the same mixed-type base generator as SynFD, while following the HFGF-style procedure to reconstruct categorical FD-dependent attributes from precomputed mappings. For unseen determinant configurations not covered by the mappings, we randomly sample the dependent value from its empirical distribution.

Table 6 reports the full per-dataset results for the conditional-completion ablation and the HFGF-style reconstruction baseline. Overall, the detailed results are consistent with the main-text analysis: context-aware completion provides a more effective fallback for unseen determinant configurations than random assignment, while SynFD achieves stronger overall preservation of cross-attribute distributional structure than the reconstruction-based alternative. These results further support our generate-then-repair design, which retains jointly generated tuples whenever possible and modifies dependent values only when required for FD preservation.

Repair-Path Statistics To examine how the different repair mechanisms are used in practice, we report in Table 7 the proportions of FD-processing instances resolved through reference mapping, retained as native unseen assignments, or handled by mask-conditioned completion on Airbnb. We use Airbnb as a representative case because all three repair paths occur on this dataset. The results show that most instances can be resolved through reference mapping, while native unseen assignments and mask-conditioned completion handle the remaining unseen determinant configurations. In particular, the use of mask-conditioned completion shows that repair-induced unseen configurations occur in practice and require the additional completion mechanism.

Table 6: Detailed ablation results for conditional completion and HFGF-style reconstruction across five datasets. Results are reported as mean ± standard deviation over three runs. F1 is reported for classification datasets and R^2 for regression datasets. Higher is better for all metrics.

Dataset	Variant	Shape ↑	Trend ↑	α-Prec. ↑	β-Rec. ↑	F1 ↑	R^2 ↑
Airbnb	w/o Conditional	0.965±0.000	0.902±0.000	0.951±0.000	0.268±0.000	–	-0.090±0.127
	HFGF-style reconstruction	0.973±0.000	**0.926±0.000**	0.976±0.000	**0.309±0.000**	–	-0.031±0.020
	SynFD	**0.975±0.000**	0.924±0.000	**0.981±0.000**	0.306±0.000	–	**-0.003±0.143**
Beijing	w/o Conditional	0.984±0.000	0.966±0.000	0.988±0.000	0.627±0.000	–	0.643±0.008
	HFGF-style reconstruction	**0.986±0.000**	**0.970±0.000**	0.986±0.000	**0.630±0.000**	–	**0.649±0.010**
	SynFD	0.983±0.000	0.966±0.000	**0.989±0.000**	0.626±0.000	–	0.639±0.021
Hospital	w/o Conditional	0.891±0.000	**0.624±0.000**	0.494±0.000	**0.599±0.000**	0.241±0.024	–
	HFGF-style reconstruction	**0.900±0.000**	0.421±0.000	0.273±0.000	0.449±0.000	0.221±0.015	–
	SynFD	0.891±0.000	0.623±0.000	**0.495±0.000**	0.554±0.000	**0.256±0.003**	–
Migraine	w/o Conditional	0.979±0.000	**0.929±0.000**	**0.925±0.000**	0.208±0.000	0.335±0.002	–
	HFGF-style reconstruction	**0.980±0.000**	0.919±0.000	0.894±0.000	**0.224±0.000**	**0.577±0.029**	–
	SynFD	0.976±0.000	0.925±0.000	0.924±0.000	0.197±0.000	0.341±0.046	–
TPC-H	w/o Conditional	0.952±0.000	0.544±0.000	0.980±0.000	**0.270±0.000**	0.661±0.002	–
	HFGF-style reconstruction	**0.957±0.000**	**0.557±0.000**	**0.997±0.000**	0.268±0.000	**0.668±0.006**	–
	SynFD	0.952±0.000	0.555±0.000	0.984±0.000	0.269±0.000	0.662±0.000	–
Average	w/o Conditional	0.954±0.000	0.793±0.000	0.868±0.000	**0.394±0.000**	0.412±0.007	0.277±0.064
	HFGF-style reconstruction	**0.959±0.000**	0.759±0.000	0.825±0.000	0.376±0.000	**0.489±0.013**	0.309±0.010
	SynFD	0.956±0.000	**0.799±0.000**	**0.875±0.000**	0.390±0.000	0.420±0.017	**0.318±0.082**

Table 7: Repair-path statistics on Airbnb.

Dataset	Reference mapping	Native unseen	Conditional completion
Airbnb	99.42%	0.30%	0.30%

D.2 Detailed Distributional Fidelity and Downstream Utility

Tables 8–12 report the per-dataset results corresponding to the averaged results in Table 3. All baseline results are evaluated after FD-based filtering, while the reported SynFD results correspond to the repaired synthetic relation. Results are reported as mean ± standard deviation over three runs.

Overall, the detailed results are consistent with the trends observed in the main text. Across datasets, SynFD generally maintains strong distributional fidelity after FD repair and remains competitive on downstream predictive tasks. This further indicates that explicitly enforcing functional dependencies does not require broadly sacrificing the statistical quality or practical utility of the generated data.

Table 10: Detailed distributional fidelity and downstream utility on Hospital.

Method	Shape ↑	Trend ↑	α-Prec. ↑	β-Rec. ↑	F1 ↑
CTGAN + Filter	0.6307±0.0004	0.3484±0.0025	**0.8247±0.0211**	0.1108±0.0164	0.2260±0.0180
TVAE + Filter	0.4326±0.0005	0.1644±0.0019	0.4966±0.0000	0.0000±0.0000	**1.0000±0.0000**
GaussianCopula + Filter	0.8177±0.0000	0.3787±0.0000	0.3572±0.0000	0.0658±0.0000	0.1910±0.0370
TabDDPM + Filter	0.6790±0.0020	0.1550±0.0006	0.1592±0.0019	0.0000±0.0000	0.1554±0.0233
TabSyn + Filter	0.7212±0.0017	0.1468±0.0029	0.6280±0.0398	0.1091±0.0079	0.1779±0.0336
TabDiff + Filter	0.8417±0.0176	0.4503±0.0135	0.7290±0.0265	0.1383±0.0255	0.2662±0.0203
GATD + Filter	0.1001±0.0004	0.0230±0.0001	0.0406±0.0014	**0.7214±0.0014**	0.1516±0.0012
SynFD	**0.8910±0.0000**	**0.6234±0.0000**	0.4948±0.0000	0.5539±0.0000	0.2563±0.0034

Table 8: Detailed distributional fidelity and downstream utility on Airbnb.

Method	Shape ↑	Trend ↑	α-Prec. ↑	β-Rec. ↑	$\mathbf{R^2}$ ↑
CTGAN + Filter	0.7667±0.0062	0.5255±0.0214	0.9060±0.0246	0.1756±0.0405	-0.7033±0.3295
TVAE + Filter	0.8976±0.0058	0.8126±0.0078	0.7988±0.0184	0.2619±0.0085	-0.1842±0.1057
GaussianCopula + Filter	0.7912±0.0000	0.5198±0.0000	0.3730±0.0000	0.1471±0.0000	-0.6078±0.5359
TabDDPM + Filter	0.9701±0.0049	**0.9330±0.0093**	0.4618±0.0016	0.0000±0.0000	-0.1402±0.1090
TabSyn + Filter	0.9640±0.0021	0.9062±0.0011	0.4632±0.0025	0.0000±0.0000	-0.1197±0.1296
TabDiff + Filter	0.9601±0.0036	0.8975±0.0064	0.9336±0.0198	0.3611±0.0160	-0.0155±0.0203
GATD + Filter	0.3893±0.0534	0.2222±0.0565	0.5920±0.2212	**0.5330±0.1528**	0.0000±0.0000
SYNFD	**0.9748±0.0000**	0.9237±0.0000	**0.9814±0.0000**	0.3056±0.0000	**-0.0032±0.1433**

Table 9: Detailed distributional fidelity and downstream utility on Beijing.

Method	Shape ↑	Trend ↑	α-Prec. ↑	β-Rec. ↑	$\mathbf{R^2}$ ↑
CTGAN + Filter	0.8062±0.0010	0.6506±0.0032	0.9201±0.0036	0.3099±0.0023	-0.1282±0.0279
TVAE + Filter	0.6751±0.0007	0.8209±0.0048	0.6878±0.0015	0.0308±0.0011	-0.8307±0.0330
GaussianCopula + Filter	0.8748±0.0000	0.7419±0.0000	0.8837±0.0000	0.3654±0.0000	0.2946±0.0440
TabDDPM + Filter	0.3578±0.0009	0.3061±0.0010	0.0032±0.0000	0.0035±0.0006	-4.3231±0.3969
TabSyn + Filter	0.9789±0.0004	0.9444±0.0096	0.9483±0.0008	0.5343±0.0042	0.6997±0.0036
TabDiff + Filter	0.9830±0.0004	**0.9679±0.0037**	0.9450±0.0010	0.5099±0.0017	0.6501±0.0203
GATD + Filter	0.1582±0.0002	0.3191±0.0009	0.9872±0.0016	**0.6964±0.0040**	**0.7529±0.0080**
SYNFD	**0.9834±0.0000**	0.9662±0.0000	**0.9893±0.0000**	0.6256±0.0000	0.6395±0.0205

Table 11: Detailed distributional fidelity and downstream utility on Migraine.

Method	Shape ↑	Trend ↑	α-Prec. ↑	β-Rec. ↑	F1 ↑
CTGAN + Filter	0.8822±0.0017	0.6596±0.0207	0.9078±0.0043	0.0989±0.0086	0.2083±0.0615
TVAE + Filter	0.8713±0.0064	0.7574±0.0156	0.2739±0.0186	0.0310±0.0028	0.5597±0.3386
GaussianCopula + Filter	0.9759±0.0000	0.8956±0.0000	0.8560±0.0000	0.1647±0.0000	0.2280±0.0740
TabDDPM + Filter	0.9771±0.0039	0.9456±0.0106	0.8805±0.0094	0.2614±0.0315	0.5429±0.0282
TabSyn + Filter	0.9578±0.0039	0.8771±0.0097	0.8654±0.0100	0.2248±0.0212	0.5030±0.0454
TabDiff + Filter	0.9717±0.0013	0.9048±0.0152	0.8832±0.0112	0.1828±0.0320	0.1775±0.0420
GATD + Filter	**0.9820±0.0035**	**0.9639±0.0075**	0.9162±0.0341	**0.2658±0.0239**	**0.6191±0.0618**
SYNFD	0.9761±0.0000	0.9254±0.0000	**0.9244±0.0000**	0.1975±0.0000	0.3413±0.0461

Table 12: Detailed distributional fidelity and downstream utility on TPC-H.

Method	Shape ↑	Trend ↑	α-Prec. ↑	β-Rec. ↑	F1 ↑
CTGAN + Filter	0.6640±0.0095	0.1161±0.0043	0.0000±0.0000	**0.8798±0.0169**	0.6228±0.0204
TVAE + Filter	0.7083±0.0015	0.1951±0.0014	0.0072±0.0015	0.7642±0.0108	0.6414±0.0035
GaussianCopula + Filter	0.4462±0.0000	0.0281±0.0000	0.0000±0.0000	0.7843±0.0000	0.4008±0.0000
TabDDPM + Filter	0.3889±0.0077	0.0210±0.0015	0.0000±0.0000	0.8103±0.1090	0.2068±0.0344
TabSyn + Filter	0.7754±0.0039	0.1613±0.0008	0.0001±0.0001	0.8668±0.0065	0.6540±0.0046
TabDiff + Filter	0.8547±0.0007	0.2411±0.0007	0.0116±0.0021	0.7967±0.0163	0.6610±0.0050
GATD + Filter	0.0655±0.0042	0.0040±0.0016	0.0000±0.0000	0.8718±0.0138	**0.7767±0.3868**
SYNFD	**0.9524±0.0000**	**0.5547±0.0000**	**0.9841±0.0000**	0.2688±0.0000	0.6618±0.0000

FD Preservation with Numerical and Mixed-type Dependency We further evaluate SYNFD on TPC-H with an extended FD set that augments the categorical dependencies with three semantically meaningful numerical and mixed-type FDs, as detailed in Appendix C.1. All dependencies are enforced jointly. As shown in Table 13, existing generators retain only a small fraction of tuples under the extended FD set. With the additional constraints, the performance of several baselines decreases slightly. In contrast, SYNFD achieves a 100% FD-consistent rate, as the TPC-H dependency set does not introduce conflicting assignments during repair. This demonstrates that SYNFD is effective beyond purely categorical FDs and can jointly enforce mixed-type dependency structures.

Table 14: Distributional fidelity and downstream utility under matched FD-valid sample sizes. Both methods generate until the number of FD-valid synthetic tuples matches the training-set size. Results are reported as mean ± standard deviation over three runs. Higher is better for all metrics.

Dataset	Method	Shape ↑	Trend ↑	α-Prec. ↑	β-Rec. ↑	F1 ↑	R^2 ↑
Airbnb	TabDiff + Filter	0.966 ± 0.0033	0.910 ± 0.0089	0.941 ± 0.0208	0.307 ± 0.0032	–	-0.047 ± 0.0951
	SynFD	$\mathbf{0.972 \pm 0.0022}$	$\mathbf{0.920 \pm 0.0030}$	$\mathbf{0.986 \pm 0.0050}$	$\mathbf{0.307 \pm 0.0093}$	–	$\mathbf{0.016 \pm 0.0543}$
Beijing	TabDiff + Filter	$\mathbf{0.982 \pm 0.0005}$	$\mathbf{0.965 \pm 0.0029}$	0.942 ± 0.0019	0.558 ± 0.0037	–	$\mathbf{0.660 \pm 0.0192}$
	SynFD	0.982 ± 0.0003	0.961 ± 0.0042	$\mathbf{0.986 \pm 0.0007}$	$\mathbf{0.623 \pm 0.0028}$	–	0.627 ± 0.0067
Migraine	TabDiff + Filter	0.972 ± 0.0003	0.901 ± 0.0116	0.869 ± 0.0040	$\mathbf{0.178 \pm 0.0276}$	0.272 ± 0.0845	–
	SynFD	$\mathbf{0.979 \pm 0.0034}$	$\mathbf{0.927 \pm 0.0093}$	$\mathbf{0.913 \pm 0.0074}$	0.169 ± 0.0004	$\mathbf{0.304 \pm 0.1090}$	–

Table 15: Distributional fidelity and downstream utility under matched FD-valid sample sizes. Both methods generate until the number of FD-valid synthetic tuples matches the training-set size. Results are reported as mean ± standard deviation over three runs. Higher is better for all metrics.

Dataset	Method	Shape ↑	Trend ↑	α-Prec. ↑	β-Rec. ↑	F1 ↑	R^2 ↑
Airbnb	TabDiff + Filter	0.966 ± 0.0033	0.910 ± 0.0089	0.941 ± 0.0208	0.307 ± 0.0032	–	-0.047 ± 0.0951
	SynFD	$\mathbf{0.972 \pm 0.0022}$	$\mathbf{0.920 \pm 0.0030}$	$\mathbf{0.986 \pm 0.0050}$	$\mathbf{0.307 \pm 0.0093}$	–	$\mathbf{0.016 \pm 0.0543}$
Beijing	TabDiff + Filter	$\mathbf{0.982 \pm 0.0005}$	$\mathbf{0.965 \pm 0.0029}$	0.942 ± 0.0019	0.558 ± 0.0037	–	$\mathbf{0.660 \pm 0.0192}$
	SynFD	0.982 ± 0.0003	0.961 ± 0.0042	$\mathbf{0.986 \pm 0.0007}$	$\mathbf{0.623 \pm 0.0028}$	–	0.627 ± 0.0067
Migraine	TabDiff + Filter	0.972 ± 0.0003	0.901 ± 0.0116	0.869 ± 0.0040	$\mathbf{0.178 \pm 0.0276}$	0.272 ± 0.0845	–
	SynFD	$\mathbf{0.979 \pm 0.0034}$	$\mathbf{0.927 \pm 0.0093}$	$\mathbf{0.913 \pm 0.0074}$	0.169 ± 0.0004	$\mathbf{0.304 \pm 0.1090}$	–

Distributional Fidelity and Utility under Matched FD-consistent Sample Sizes Table 15 compares SynFD with TabDiff+Filter when both methods produce the same number of FD-consistent tuples as the training relation, thereby removing the effect of sample-size reduction caused by filtering. Overall, SynFD achieves comparable distributional fidelity and downstream utility to TabDiff+Filter under the matched sample size. This indicates that SynFD preserves the underlying data distribution while enforcing structural consistency. For Hospital and TPC-H, TabDiff+Filter cannot obtain the required number of FD-consistent tuples within the $50\times$ sampling budget, and is therefore excluded from this comparison.

Table 13: FD-consistent rates (%) on TPC-H under the extended FD set. Higher is better.

Method	R_{FD}
CTGAN	0.52
TVAE	2.70
GaussianCopula	0.03
TabDDPM	0.03
TabSyn	1.80
TabDiff	4.21
GATD	0.10
SynFD	**100.00**