IMPROVING RELATIONAL FORECASTING WITH TARGET HISTORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Relational learning, namely machine learning for relational databases, has attracted increased interest in recent years. However, despite the inherently temporal nature of forecasting tasks in relational learning, modeling of the temporal component has received limited attention. In particular, existing models often do not employ an entity's past target values in a principled way, even though these values may provide a strong predictive signal. In this work, we investigate multiple model-agnostic strategies for incorporating an entity's target history into predictions. We show that a simple strategy consistently improves the performance of multiple representative models on RelArena-α forecasting tasks, with the best resulting model achieving new state-of-the-art results among open-source model submissions. Finally, we conduct a case study showing how the current typically used evaluation protocol hinders performance on certain datasets by forbidding the use of information between the test threshold and the sample timestamp. We hope our findings will draw the community's attention to the importance of the temporal component in relational forecasting tasks and contribute to the development of better benchmarking protocols.[1]

1 INTRODUCTION

Relational databases store information across linked tables, such as customers, products, and transactions. Making predictions from these data requires combining information across tables, traditionally through task-specific feature engineering. Relational learning aims to reduce this effort by developing methods that operate directly on relational databases (Fey et al., 2023; Robinson et al., 2024). Recent approaches range from supervised graph models (Hamilton et al., 2017; Chen et al., 2025; Dwivedi et al., 2025) to relational foundation models (Wang et al., 2025; Hudovernik et al., 2026) and methods that combine automated feature construction with tabular learners (Xu et al., 2026; Wang et al., 2026; Huang et al., 2026). Despite their different approaches, these methods face a common question: which information in the database is useful for the prediction task?

For forecasting, a natural source of information is the entity's own past outcomes. Unlike autocomplete tasks, which fill in missing attributes, forecasting tasks predict an outcome for an entity over a future time window. For example, a model may predict a customer's total spending over the next month. The customer's spending in previous months can reveal both their usual activity level and recent changes in behavior.

Importantly, an entity's *target history* need not be limited to the labels supplied in a benchmark's training table. In RelBench, targets are computed through task-specific SQL queries over database records (Robinson et al., 2024). Applying the same task definition at earlier timestamps can therefore recover past outcomes without collecting additional data or annotations. For monthly spending, this amounts to computing spending over completed historical months from existing transactions. Thus, the task definition provides a direct way to turn existing records into task-relevant temporal features.

Existing methods can already capture parts of this signal, but through mechanisms with different limitations. Generic relational aggregation can summarize past activity without retaining target values for individual historical windows. Reusing labels from supplied task tables gives access to past targets, but only for the entity–timestamp pairs included in those tables. LLM-based feature

[1] Our code is available at `https://anonymous.4open.science/r/d8f85568c61bd2c6/`.

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.

generation can potentially reconstruct historical targets, but does not guarantee their construction. These observations motivate studying explicit reconstruction of target history and how best to expose it to downstream models.

In this work, we investigate simple, model-agnostic representations of reconstructed target history: individual lags, summary statistics, and their combinations. We evaluate them on RelArena-α (Hayler et al., 2026) and select a single representation using validation performance, then apply it across tasks and model families. Adding these features improves aggregate performance for TabPFN-Rel, GraphSAGE, and LightGBM, without changing their hyperparameter search spaces or tuning budgets. In particular, TabPFN-Rel with target history achieves the highest test Elo among the open-source model submissions in our comparison.

Our study also highlights an evaluation choice that affects the usefulness of temporal information. Relational forecasting benchmarks, including RelArena-α (Hayler et al., 2026), commonly freeze the database at each evaluation split's threshold, even for predictions made at later timestamps. Our main experiments retain this restriction for comparability with the benchmark baselines. We examine this restriction in a separate case study on `rel-f1`, the only database in our benchmark with prediction timestamps after the evaluation split thresholds. In this case study, we allow target history to include records after the split threshold but no later than the prediction timestamp. All three models improve on all three tasks, motivating a closer look at this restriction in future evaluation protocols.

Our contributions can be summarized as follows:

- We study model-agnostic ways to incorporate target history reconstructed from database records, comparing raw lags, summary statistics, and their combinations.
- We evaluate target-history features across 21 forecasting tasks and three model families, showing that a simple shared representation improves aggregate performance and achieves the highest test Elo score among open-source model submissions in our comparison.
- We show through a case study that freezing the database at the split threshold can exclude useful historical information for later predictions, motivating a reconsideration of information cutoffs in forecasting evaluation.

2 Background

2.1 Relational learning

Relational learning (Fey et al., 2023), or machine learning for relational databases (RDBs), has attracted increased attention in recent years. It shifts the focus from feature engineering and single-table models to end-to-end methods that operate directly on relational databases. A central idea is to represent an RDB as a temporal, heterogeneous graph, where each table row forms a node and edges correspond to primary–foreign key relationships.

Models Most current models for relational learning can be roughly categorized into three groups: supervised graph-based models, relational foundation models (RFMs) and tabularization-based methods.

Supervised graph-based models draw on graph neural networks (Hamilton et al., 2017) and graph transformers (Min et al., 2022) and are trained from scratch for each task. Examples include GraphSAGE (Hamilton et al., 2017), RelGNN (Chen et al., 2025), RelGT (Dwivedi et al., 2025) and HGT (Hu et al., 2020).

Relational foundation models are pretrained to solve unseen tasks with minimal finetuning or no weight updates, often using in-context learning (ICL). Examples include KumoRFM (Fey et al., 2025; Hudovernik et al., 2026), Griffin (Wang et al., 2025), Relational Transformer (Ranjan et al., 2026) and PluRel (Kothapalli et al., 2026).

Tabularization-based methods flatten an RDB into a single table and apply a tabular learner to it. Some methods, including RDBLearn (Xu et al., 2026), RDB-PFN (Wang et al., 2026) and TabPFN-Rel (Hayler et al., 2026), focus on the tabular learner, using tabular foundation models such as TabPFN (Hollmann et al., 2023; 2025; Grinsztajn et al., 2026) for prediction and relatively

straightforward techniques such as deep feature synthesis (Kanter & Veeramachaneni, 2015) for tabularization. In contrast, RelAgent (Huang et al., 2026) and ReFuGe (Kim et al., 2026) propose more complex tabularization approaches, such as using LLM agents for feature generation.

Benchmarks Several benchmarks have been proposed for relational learning, including RelBench (Robinson et al., 2024), RelBench v2 (Gu et al., 2026) and 4DBInfer (Wang et al., 2024). Tasks in these benchmarks can be grouped into three types (Gu et al., 2026). First, autocomplete tasks fill in missing values in an RDB, such as a user's country when it is known for some users but not others. Second, forecasting tasks predict future outcomes for an entity at a given timestamp, such as a user's total spending over the next month. Third, recommendation tasks predict a ranked list of the top-K target entities for each source entity and seed timestamp, such as the items a user will purchase over the next month.

More recently, RelArena-α (Hayler et al., 2026) was introduced with aim to make evaluation fairer and more robust. It uses 21 forecasting tasks from RelBench v1 (Robinson et al., 2024) and provides a unified evaluation protocol covering hyperparameter tuning and metric aggregation, along with a representative set of reproducible, tuned baselines.

2.2 Relational forecasting

In entity-level relational forecasting, each example is an entity–timestamp pair (e, t), and the target describes an outcome for entity e over a future window $(t, t + \Delta]$, where Δ is the prediction horizon. For example, a task may predict a customer's total spending over the next month. Unlike autocomplete, which fills a missing attribute in a database snapshot, such as a customer's country, forecasting explicitly indexes the outcome by time. The same customer can therefore have different target values at different timestamps, giving rise to a *target history* that may be useful for predicting future outcomes.

In RelBench, forecasting targets are computed by task-specific SQL queries over the relational database (Robinson et al., 2024; Hayler et al., 2026). Applying the same task definition at earlier timestamps allows us to reconstruct historical targets from existing records, without collecting additional data or annotations. For the spending example, a customer's spending over completed past months can be computed directly from transaction records already stored in the database. Thus, target history need not be limited to the labels already included in the supplied training table: it can be computed for earlier timestamps using the task definition and the available database records.

The availability of this history depends on both the database and the task. A historical target can be used only once its outcome window has ended and the records needed to compute it are available under the evaluation protocol. Some values may also be unavailable because an entity did not satisfy the task's eligibility conditions at an earlier timestamp. Target history is therefore not necessarily a complete time series for every entity.

2.3 Handling of target history in existing methods

Existing methods can obtain information about past target values through three broad mechanisms: generic relational aggregation, reuse of supplied task labels, and task-dependent feature generation with LLM agents. These mechanisms are not mutually exclusive: a single method may combine several of them.

Relational aggregation As discussed above, historical targets can be computed by applying task-specific SQL queries to existing database records. Generic aggregations of the same records may therefore capture information similar to historical targets or their summaries, without explicitly applying the task definition. Deep feature synthesis (DFS) (Kanter & Veeramachaneni, 2015), used by RDBLearn (Xu et al., 2026) and TabPFN-Rel (Hayler et al., 2026), provides one such mechanism by aggregating related records along primary–foreign key relationships. For example, when the target is monthly spending, aggregating past transactions can yield total spending over the observed history. Combined with the duration of that history, this can provide an estimate of average monthly spending. Such features can capture part of the signal in target history without using supplied task labels, but they need not reproduce the task's window boundaries, eligibility conditions, or aggregation semantics. Moreover, a summary over the entire observed history does not preserve variation across

individual historical windows. GraphSAGE (Hamilton et al., 2017) provides another form of relational aggregation, learning to aggregate related records through message passing. However, it samples a fixed number of neighbors per node, so for entities with many related records it does not guarantee access to the full history needed to reconstruct past targets.

Reuse of labels from task tables Past target values can also be obtained from the supplied task tables: the training table and, where permitted by the evaluation protocol, the validation table. Labels from earlier timestamps can therefore be reused as inputs for later predictions. For example, RDBLearn supports adding a training-label table to the database, allowing DFS to aggregate past target values in addition to the original database records (Xu et al., 2026). Both RDBLearn and TabPFN-Rel use this mechanism by default in their RelArena-α implementations (Hayler et al., 2026). TabPFN-Rel also offers lag features containing an entity's most recent available labels and their ages, but these are disabled by default. The RelBench GNN example optionally includes task tables in the graph, making past labels available through message passing (Robinson et al., 2024), though RelArena-α's GraphSAGE implementation does not support this option. However, all of these mechanisms rely on the entity–timestamp pairs present in the supplied task tables: they do not themselves evaluate the task definition at additional historical timestamps.

LLM-based feature generation RelAgent (Huang et al., 2026) and ReFuGe (Kim et al., 2026) use LLM agents to generate task-dependent relational features. Unlike generic DFS, their generated queries can incorporate task-specific conditions and could potentially reconstruct historical targets directly from database records. RelAgent also allows feature construction to access supplied training labels, overlapping with the preceding category. Such task-dependent feature generation can potentially recover historical targets or related summaries, although it does not explicitly guarantee their construction.

3 Incorporating target history

We incorporate target history as additional features for each entity–timestamp pair. The construction uses the task definition to recover past outcomes, then represents them through individual lags, summary statistics, or both. These features can complement different relational models without changing how they represent the database.

Constructing historical targets For each prediction (e, t), we reconstruct past targets over consecutive windows of length Δ, working backwards from a history cutoff s. During training, we use the prediction timestamp as the cutoff, so $s = t$. For validation and testing, the RelArena-α protocol freezes the database at the validation and test thresholds, respectively (Hayler et al., 2026). To align with this protocol, we clip the prediction timestamp to the corresponding threshold τ, giving $s = \min(t, \tau)$.

Let $y(e, t)$ denote the target for entity e over the task window $(t, t + \Delta]$. The kth lag for prediction (e, t) is then

$$h_k(e, t) = y(e, s - k\Delta), \qquad s = \min(t, \tau), \qquad k = 1, 2, \ldots. \tag{1}$$

Thus, the first lag covers $(s - \Delta, s]$, the second covers $(s - 2\Delta, s - \Delta]$, and all lag windows end at or before s. We compute the lags by evaluating the task's SQL query at these historical timestamps and matching the resulting targets to entity e. This uses the task's target definition and eligibility conditions directly, without restricting the history to labels in the supplied training table. If the query returns no target for an entity at a historical timestamp, we leave that lag missing.

Without clipping, recent lag windows for predictions after τ would extend beyond the frozen database's information cutoff, leaving these targets unavailable. Anchoring the windows at τ instead retains the most recent available history while respecting the protocol, at the cost of using older outcomes. In our benchmark, this only affects `rel-f1`, which has multiple validation and test timestamps. All other datasets have a single timestamp per evaluation split, equal to the corresponding threshold. We revisit this restriction in Section 6, where we evaluate forecasting on `rel-f1` without clipping.

History representations We consider two complementary representations: raw lags, which retain temporal order, and aggregates, which summarize the distribution of past outcomes. Let L be the

number of raw lags and A the number of lags used for aggregation. The combined feature vector is

$$z_{L,A}(e,t) = \big[h_1(e,t), \ldots, h_L(e,t), \mathrm{Agg}(h_1(e,t), \ldots, h_A(e,t))\big]. \tag{2}$$

Here, Agg returns eight statistics: the mean, sample standard deviation, minimum, 25th percentile, median, 75th percentile, maximum, and number of observed values. The statistics ignore missing lags, and undefined statistics remain missing. The two lengths are independent: A may exceed L to combine a short ordered history with summaries over a longer period. Setting $A = 0$ omits the aggregates, while setting $L = 0$ omits the raw lags. The representation therefore contains L features without aggregation and $L + 8$ features with aggregation.

Model integrations The history vector $z_{L,A}(e,t)$ is model-agnostic and can be incorporated through simple additions to a model's existing inputs. For LightGBM, we append the history vector to the entity-table features. For TabPFN-Rel, we add the history features to the prediction table before running DFS. These columns are retained alongside the generated relational features in the input to TabPFN. For GraphSAGE, history must be associated with an entity–timestamp pair rather than the entity alone, since the same entity can have different histories at different timestamps. We therefore incorporate it into the seed-node embedding for each prediction rather than storing it as a static node attribute. We normalize the history features and apply a learned linear projection, then add the projected vector to the seed-node embedding before message passing.

4 Experimental setup

Benchmark We evaluate on the 21 entity-level forecasting tasks from seven RelBench v1 databases included in RelArena-α (Robinson et al., 2024; Hayler et al., 2026). We use its temporal splits and freeze the database at the validation threshold during model selection and at the test threshold during final evaluation. Target history computation respects the same information cutoffs. A separate case study on `rel-f1` relaxes this restriction for target history, see Section 6 for details.

Models We compare TabPFN-Rel (OSS), GraphSAGE, and LightGBM with and without target-history features. TabPFN-Rel and GraphSAGE provide strong baselines for two different approaches to relational learning: tabularization with a foundation model and supervised graph learning. In the RelArena-α release, TabPFN-Rel (OSS) is the highest-ranked open-source model submission, while GraphSAGE is the highest-ranked model that does not use a foundation model (Hayler et al., 2026). LightGBM lets us test how competitive a simple tabular model can become by combining entity-table features with target history.

We also include a persistence baseline that uses the most recent historical target, $h_1(e,t)$, directly as its prediction. If that target is missing, it falls back to an entity-specific constant computed from the labels available for fitting, or a global constant if the entity has no such labels. These constants are means for binary classification and medians for regression.

Evaluation We use ROC-AUC for binary classification and MAE for regression. For both history selection and the main comparisons, we average each method's metric across 5 seeds separately for each task, then compute Elo ratings from the resulting task-level comparisons using RelArena-α's aggregation procedure. Our 5-seed baseline reruns replace the corresponding released TabPFN-Rel, GraphSAGE, and LightGBM results in the comparisons. For the remaining reference methods, we retain the released RelArena-α results.

Tuning For each task, we retain the backbone's RelArena-α search space and tuning budget, both with and without target history. TabPFN-Rel evaluates three DFS depths, 2, 3, and 4. GraphSAGE evaluates its default configuration and four randomly sampled configurations, while LightGBM evaluates its default and thirty randomly sampled configurations. We select backbone hyperparameters using validation performance and refit the selected configuration on the combined training and validation tables before test evaluation. To account for run-to-run variation, we repeat the full tuning-and-refit pipeline with 5 random seeds for each backbone and history configuration, including the baselines without added history.

Table 1: Target-history representation selection using TabPFN-Rel on the validation splits of all 21 tasks. L is the number of raw lags and A is the number of lags summarized by the aggregate features. Each row compares one candidate with the unmodified baseline in a separate pool containing the same reference methods, so baseline Elo varies across rows.

	Baseline Elo	Candidate Elo	ΔElo
$L = 1, A = 0$	1,870	1,870	0
$L = 2, A = 0$	1,860	1,892	32
$L = 4, A = 0$	1,856	1,898	42
$L = 8, A = 0$	1,847	1,903	56
$L = 16, A = 0$	1,853	1,928	75
$L = 32, A = 0$	1,861	1,893	32
$L = 64, A = 0$	1,857	1,899	42
$L = 1, A = 64$	1,852	1,885	32
$L = 2, A = 64$	1,852	1,922	70
$L = 4, A = 64$	1,861	1,893	32
$L = 8, A = 64$	1,852	1,908	56
$L = 16, A = 64$	1,857	1,903	47
$L = 32, A = 64$	1,861	1,884	23
$L = 64, A = 64$	1,857	1,880	23
$L = 0, A = 64$	1,870	1,870	0

History representation selection We first use TabPFN-Rel to select a single history representation on the validation splits across all 21 tasks. We compare raw lags with $L \in \{1, 2, 4, 8, 16, 32, 64\}$ and $A = 0$, aggregates alone with $(L, A) = (0, 64)$, and combined representations with the same seven values of L and $A = 64$. For each candidate, we compute its validation Elo improvement over TabPFN-Rel without the added history features. Each comparison pool contains the same RelArena reference methods with available validation scores, the unmodified TabPFN-Rel baseline, and only that candidate. We select the representation with the largest Elo improvement and keep it fixed across tasks and all three backbones in the main experiments. This lets us test whether a representation selected with TabPFN-Rel also benefits other backbones, without tuning the history configuration separately for each task or model.

5 Experimental results

5.1 Selecting a history representation

We first compare target-history representations using TabPFN-Rel, selecting a single configuration by its validation Elo improvement over the baseline across all 21 tasks. Table 1 reports the results, from which we draw the following observations.

Observation 1. *Adding more history does not always improve performance.* With raw lags alone, the validation Elo gain increases from 0 points for one lag to 75 points for 16 lags, then falls to 32 and 42 points for 32 and 64 lags, respectively. When aggregates over 64 lags are included, the gain also varies non-monotonically with the number of raw lags, peaking at 70 points for two raw lags. Adding aggregates improves some raw-lag configurations but worsens others. Thus, neither extending the history nor adding summary statistics consistently improves performance.

Observation 2. *A single lag and aggregates provide no Elo gain individually, but help in combination.* Neither the most recent lag alone nor aggregates over 64 lags improve validation Elo over the baseline, whereas combining them with $(L, A) = (1, 64)$ yields a gain of 32 points. This suggests that recent outcomes and longer-term summaries can provide complementary signals.

We select $(L, A) = (16, 0)$, which achieves the largest validation Elo improvement among the evaluated configurations. This representation consists of just 16 raw-lag features, without summary statistics. We keep it fixed across all tasks and backbones in the main experiments, rather than selecting a separate history representation for each model or task.

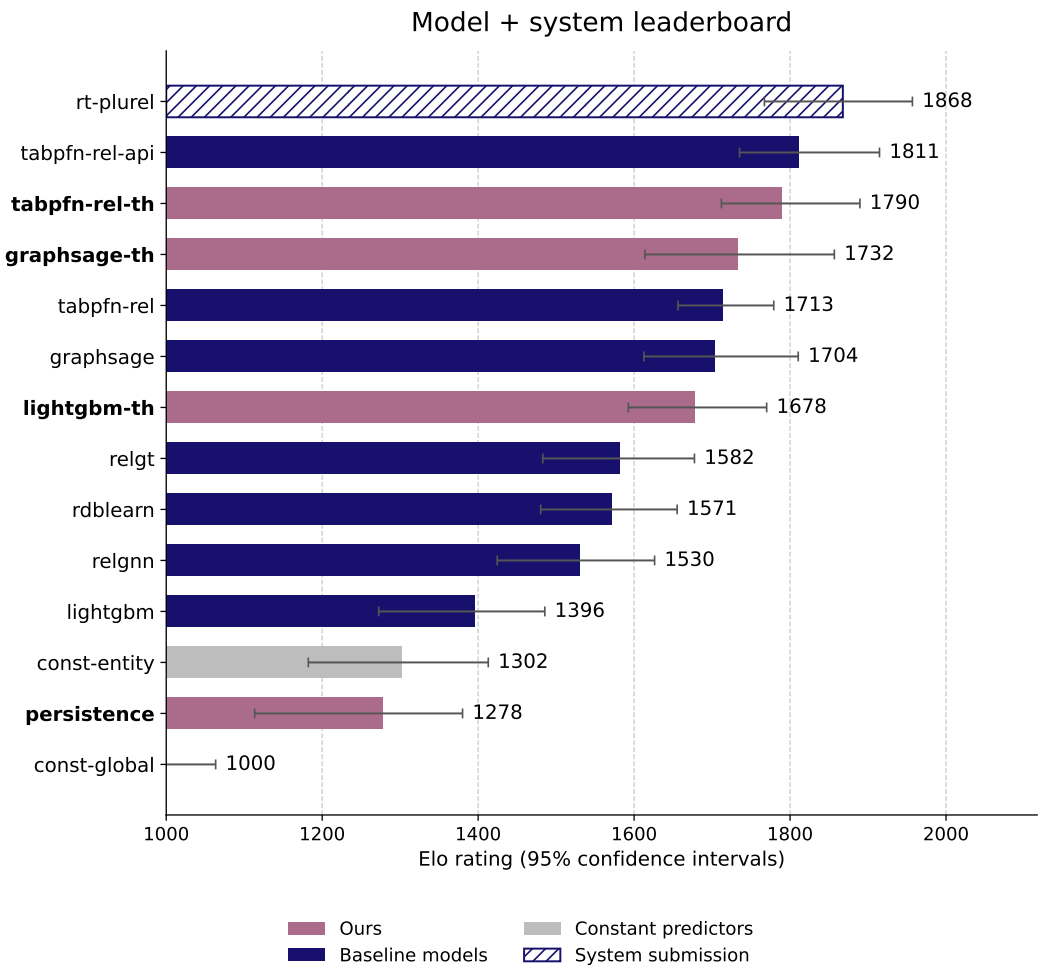


Figure 1: Test Elo ratings on the 21 RelArena-α forecasting tasks. The suffix `-th` (target history) denotes models augmented with 16 raw target lags. For our runs, task metrics are averaged over five seeds before Elo aggregation. Other reference methods use the released RelArena-α results. Error bars show 95% confidence intervals.

5.2 Main experiments

Figure 1 presents the aggregated results of our main experiments. Full per-task results are provided in Table 2. We augment LightGBM, GraphSAGE, and TabPFN-Rel with the same 16 raw lags selected in the preceding subsection. Each augmented model retains the backbone search space and tuning budget of its original version, with no further search over history representations. We draw the following observations from these comparisons.

Observation 3. *TabPFN-Rel with target history ranks first among open-source model submissions.* TabPFN-Rel with target history achieves approximately 1,790 Elo, the highest score among the open-source model submissions in our comparison. Thus, adding just 16 target-history features improves an already strong baseline.

Observation 4. *Target history improves all three evaluated methods.* Adding target history increases Elo by approximately 77 points for TabPFN-Rel (from 1,713 to 1,790), 28 points for GraphSAGE (from 1,704 to 1,732), and 282 points for LightGBM (from 1,396 to 1,678). The gains for GraphSAGE and LightGBM show that the history representation selected using TabPFN-Rel transfers to other models without model-specific history selection. Target history also changes the ranking across model families: GraphSAGE with target history ranks above unmodified TabPFN-Rel by Elo score.

Table 2: Per-task test results on the 21 RelArena-α forecasting tasks. We report ROC-AUC for binary classification and MAE for regression. The suffix `-th` (target history) denotes models augmented with 16 raw target lags.

	lightgbm	lightgbm-th	graphsage	graphsage-th	tabpfn-rel	tabpfn-rel-th	persistence
rel-amazon/item-churn ↑	0.6557 ± 0.0090	0.7834 ± 0.0014	0.8296 ± 0.0012	0.8301 ± 0.0006	0.8277 ± 0.0002	0.8278 ± 0.0001	0.6643 ± 0.0000
rel-amazon/user-churn ↑	0.5169 ± 0.0004	0.6641 ± 0.0001	0.7078 ± 0.0020	0.7083 ± 0.0012	0.7025 ± 0.0002	0.7023 ± 0.0002	0.6207 ± 0.0000
rel-avito/user-clicks ↑	0.5642 ± 0.0000	0.6171 ± 0.0076	0.6422 ± 0.0360	0.6444 ± 0.0343	0.6414 ± 0.0438	0.6191 ± 0.0580	0.4544 ± 0.0000
rel-avito/user-visits ↑	0.5294 ± 0.0020	0.6550 ± 0.0008	0.6614 ± 0.0068	0.6632 ± 0.0020	0.6694 ± 0.0029	0.6695 ± 0.0024	0.6112 ± 0.0000
rel-event/user-ignore ↑	0.7762 ± 0.0119	0.8628 ± 0.0057	0.7766 ± 0.0260	0.7905 ± 0.0191	0.6961 ± 0.0378	0.7315 ± 0.0272	0.5862 ± 0.0000
rel-event/user-repeat ↑	0.7580 ± 0.0126	0.7940 ± 0.0050	0.7659 ± 0.0201	0.7386 ± 0.0294	0.7617 ± 0.0109	0.7612 ± 0.0110	0.7614 ± 0.0000
rel-f1/driver-dnf ↑	0.7213 ± 0.0172	0.6991 ± 0.0287	0.6879 ± 0.0311	0.6974 ± 0.0035	0.7044 ± 0.0045	0.7211 ± 0.0070	0.5410 ± 0.0000
rel-f1/driver-top3 ↑	0.7333 ± 0.0081	0.7989 ± 0.0095	0.7611 ± 0.0097	0.8215 ± 0.0128	0.7906 ± 0.0038	0.8432 ± 0.0089	0.7159 ± 0.0000
rel-hm/user-churn ↑	0.5871 ± 0.0019	0.6869 ± 0.0004	0.7072 ± 0.0051	0.7084 ± 0.0030	0.7059 ± 0.0002	0.7051 ± 0.0006	0.6373 ± 0.0000
rel-stack/user-badge ↑	0.6844 ± 0.0063	0.8187 ± 0.0005	0.8891 ± 0.0008	0.8893 ± 0.0010	0.8587 ± 0.0153	0.8618 ± 0.0076	0.6112 ± 0.0000
rel-stack/user-engagement ↑	0.8047 ± 0.0080	0.8942 ± 0.0008	0.9066 ± 0.0010	0.9074 ± 0.0006	0.9059 ± 0.0004	0.9059 ± 0.0002	0.7108 ± 0.0000
rel-trial/study-outcome ↑	0.7112 ± 0.0073	0.7132 ± 0.0088	0.6855 ± 0.0135	0.6853 ± 0.0066	0.7184 ± 0.0067	0.7254 ± 0.0033	0.5000 ± 0.0000
rel-amazon/item-ltv ↓	56.5190 ± 0.5085	44.1802 ± 0.1016	48.8015 ± 0.7188	47.4271 ± 0.2614	47.3728 ± 0.3387	46.6394 ± 0.1946	49.6855 ± 0.0000
rel-amazon/user-ltv ↓	16.7654 ± 0.0185	14.1968 ± 0.0050	14.2703 ± 0.0448	14.1898 ± 0.0492	14.3877 ± 0.0198	14.2903 ± 0.0247	19.6350 ± 0.0000
rel-avito/ad-ctr ↓	0.0413 ± 0.0001	0.0389 ± 0.0001	0.0409 ± 0.0033	0.0412 ± 0.0020	0.0315 ± 0.0001	0.0314 ± 0.0000	0.0401 ± 0.0000
rel-event/user-attendance ↓	0.2625 ± 0.0006	0.2553 ± 0.0012	0.2427 ± 0.0019	0.2479 ± 0.0046	0.2405 ± 0.0011	0.2382 ± 0.0020	0.2717 ± 0.0000
rel-f1/driver-position ↓	4.0324 ± 0.0500	3.8472 ± 0.0838	4.1579 ± 0.1838	4.0959 ± 0.1842	3.7862 ± 0.0437	3.7732 ± 0.0172	4.1130 ± 0.0000
rel-hm/item-sales ↓	0.0750 ± 0.0004	0.0352 ± 0.0003	0.0548 ± 0.0005	0.0511 ± 0.0004	0.0608 ± 0.0005	0.0589 ± 0.0008	0.0423 ± 0.0000
rel-stack/post-votes ↓	0.0663 ± 0.0003	0.0629 ± 0.0001	0.0652 ± 0.0002	0.0653 ± 0.0005	0.0678 ± 0.0001	0.0677 ± 0.0001	0.0970 ± 0.0000
rel-trial/site-success ↓	0.4371 ± 0.0056	0.4394 ± 0.0047	0.3508 ± 0.0051	0.3574 ± 0.0099	0.4009 ± 0.0083	0.3908 ± 0.0086	0.4550 ± 0.0000
rel-trial/study-adverse ↓	44.8425 ± 2.2274	43.8266 ± 0.7738	44.0050 ± 0.1824	44.0661 ± 0.3237	42.4270 ± 0.0384	42.4207 ± 0.0512	57.5332 ± 0.0000

Observation 5. *LightGBM benefits most from target history.* LightGBM obtains the largest Elo gain among the three models, despite using the history representation selected for TabPFN-Rel. With only entity-table features and 16 target lags, it ranks above RelGT (1,582), RDBLearn (1,571), and RelGNN (1,530). Thus, task-specific target history can make a simple tabular learner competitive with methods that use broader relational inputs. These results support using target history as a strong source of predictive information.

Observation 6. *Persistence ranks below the entity-specific constant baseline.* The persistence baseline achieves approximately 1,278 Elo, compared with 1,302 for the entity-specific constant baseline. This is consistent with the representation-selection results, where the most recent lag alone provides no Elo gain. Together, these findings suggest that recency signals alone are insufficient, but can become useful when combined with longer-term historical information.

6 Case study: effect of timestamp clipping

The RelArena-α protocol freezes the database at the validation or test threshold, excluding later records even when they precede an example's prediction timestamp. Our main experiments respect this restriction by clipping the target-history cutoff to the corresponding threshold. However, this restriction can be unnecessarily strict for forecasting systems that receive new records over time. At prediction time t, records from between the split threshold τ and t could already be available, provided they have arrived by the time the prediction is made. Freezing the database at τ prevents a model from using this information.

We examine the predictive value of these records on `rel-f1`, the only database in our benchmark with multiple timestamps in each evaluation split. We compare the same 16-lag history representation with and without timestamp clipping. With clipping, the history cutoff is $s = \min(t, \tau)$, so the history becomes older relative to the prediction timestamp as t advances beyond τ. Without clipping, we use $s = t$, allowing historical target windows to include records after the split threshold but no later than the prediction timestamp. We relax access only for constructing target-history features, not for computing the future target being predicted. All other database inputs remain frozen at the split threshold, including records used by DFS and graph-based models.

Observation 7. *Removing clipping improves all three models on all three tasks.* Table 3 shows consistent improvements over clipped history across all nine model–task combinations. LightGBM obtains the largest gains: ROC-AUC increases from 0.6991 to 0.8193 on `driver-dnf` and from 0.7989 to 0.9012 on `driver-top3`, while MAE decreases from 3.8472 to 2.9043 on `driver-position`. GraphSAGE and TabPFN-Rel also improve on every task. Thus, records between the split threshold and the prediction timestamp carry useful predictive information, even when accessed only through target history features.

These findings motivate revisiting the information cutoffs used in relational forecasting benchmarks. When records are available before a prediction is made, future protocols could allow all methods to

Table 3: Effect of timestamp clipping on the three `rel-f1` tasks. We report test ROC-AUC for `driver-dnf` and `driver-top3`, and test MAE for `driver-position`. The suffix `-th` denotes models augmented with 16 raw target lags.

	`driver-dnf` ↑	`driver-top3` ↑	`driver-position` ↓
`lightgbm`	0.7213 ± 0.0172	0.7333 ± 0.0081	4.0324 ± 0.0500
`lightgbm-th` w/ clipping	0.6991 ± 0.0287	0.7989 ± 0.0095	3.8472 ± 0.0838
`lightgbm-th` w/o clipping	0.8193 ± 0.0037	0.9012 ± 0.0108	2.9043 ± 0.0397
`graphsage`	0.6879 ± 0.0311	0.7611 ± 0.0097	4.1579 ± 0.1838
`graphsage-th` w/ clipping	0.6974 ± 0.0035	0.8215 ± 0.0128	4.0959 ± 0.1842
`graphsage-th` w/o clipping	0.7485 ± 0.0080	0.8776 ± 0.0137	3.2640 ± 0.1688
`tabpfn-rel`	0.7044 ± 0.0045	0.7906 ± 0.0038	3.7862 ± 0.0437
`tabpfn-rel-th` w/ clipping	0.7211 ± 0.0070	0.8432 ± 0.0089	3.7732 ± 0.0172
`tabpfn-rel-th` w/o clipping	0.7585 ± 0.0028	0.8893 ± 0.0014	3.3753 ± 0.0290

use them, rather than excluding them solely because they fall after the split threshold. This would better reflect deployment settings in which new records arrive over time.

7 Conclusion

Our results show that explicitly incorporating an entity's target history is a simple and effective way to improve relational forecasting. By applying the task definition to existing database records, we recover past outcomes beyond the labels supplied in the training table, without collecting additional data or annotations. A shared representation of 16 raw target lags improves aggregate performance across three model families on the 21 RelArena-α forecasting tasks, highlighting the value of task-relevant temporal features alongside existing relational representations. Our `rel-f1` case study further shows that allowing target history to use records available by the prediction timestamp, rather than freezing it at the split threshold, improves all three models on all three tasks and motivates revisiting information cutoffs in forecasting benchmarks.

8 Limitations and future work

Use of the task definition Our method reconstructs historical targets by evaluating the task's SQL query at earlier timestamps. These targets are computed only from database records that are available under the evaluation protocol, so the method does not use information beyond the protocol's cutoffs. However, it does use the task definition itself, while many existing methods only receive the supplied task tables. Current benchmark protocols, including RelArena-α, do not explicitly state whether methods may evaluate the task definition beyond these tables. If this is allowed, the same queries could also be used to construct additional training examples at new timestamps, which goes beyond adding input features. In this work, we use the reconstructed targets only as features and train on the supplied task tables. We believe that in practice the task definition is known to the practitioner, but clear rules on its use would make comparisons between methods more fair.

Model-agnostic integration We focus on simple, model-agnostic ways to incorporate target history, namely appending lags or summary statistics to existing inputs. This makes the approach easy to apply to different model families, but it may not use target history to its full extent. In particular, deep learning models such as GraphSAGE could benefit from specialized components, for example sequence encoders over target history or architectures that jointly model target history and relational context. We also select a single history representation using TabPFN-Rel and keep it fixed across all tasks and models. Selecting the representation separately for each model or task, or using lag steps that differ from the prediction horizon, may give further gains.

Other task types Our method applies only to forecasting tasks, where the target is explicitly indexed by time. Autocomplete tasks do not have a natural notion of target history. Recommendation tasks do have a temporal structure, as past interactions between source and target entities can be seen as a form of target history, but using it requires a different representation than the one studied here. Extending our approach to recommendation tasks is a promising direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to aid method implementation and to aid and polish writing. We have not used generative AI tools for proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, cleaning and reformatting datasets, supporting qualitative and thematic data analysis and interpreting results. The following tasks with required disclosure are not applicable to our work: generating synthetic datasets, helping developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of AI.

REFERENCES

Tianlang Chen, Charilaos Kanatsoulis, and Jure Leskovec. RelGNN: Composite message passing for relational deep learning. In *International Conference on Machine Learning*, pp. 8296–8312, 2025.

Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico López, Charilaos I Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. Relational graph transformer. *arXiv preprint arXiv:2505.10960*, 2025.

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.

Matthias Fey, Vid Kocijan, Federico Lopez, Jan Eric Lenssen, and Jure Leskovec. KumoRFM: A foundation model for in-context learning on relational data, 2025.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026.

Justin Gu, Rishabh Ranjan, Charilaos Kanatsoulis, Haiming Tang, Martin Jurkovic, Valter Hudovernik, Mark Znidar, Pranshu Chaturvedi, Parth Shroff, Fengyu Li, et al. RelBench v2: A large-scale benchmark and repository for relational data. *arXiv preprint arXiv:2602.12606*, 2026.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

Adrian Hayler, Klemens Flöge, Alan Arazi, Rishabh Ranjan, Jure Leskovec, Felix Birkel, Brendan Roof, Anurag Garg, Kristina Collins, Lydia Sidhoum, et al. Advancing open and reproducible relational learning: RelArena-α, TabPFN-Rel and RPI. *arXiv preprint arXiv:2608.16319*, 2026.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pp. 2704–2710, 2020.

Xingyue Huang, Louis Tichelman, Jinwoo Kim, Krzysztof Olejniczak, and Ismail Ilkan Ceylan. RelAgent: Llm agents as data scientists for relational learning. *arXiv preprint arXiv:2605.07840*, 2026.

Valter Hudovernik, Federico López, Vid Kocijan, Akihiro Nitta, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. KumoRFM-2: Scaling foundation models for relational learning. *arXiv preprint arXiv:2604.12596*, 2026.

James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pp. 1–10. IEEE, 2015.

Kyungho Kim, Geon Lee, Juyeon Kim, Dongwon Choi, Shinhwan Kang, and Kijung Shin. ReFuGe: Feature generation for prediction tasks on relational databases with llm agents. In *Proceedings of the ACM Web Conference 2026*, pp. 8501–8504, 2026.

Vignesh Kothapalli, Rishabh Ranjan, Valter Hudovernik, Vijay Prakash Dwivedi, Johannes Hoffart, Carlos Guestrin, and Jure Leskovec. PluRel: Synthetic data unlocks scaling laws for relational foundation models. In *International Conference on Machine Learning*, 2026.

Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*, 2022.

Rishabh Ranjan, Valter Hudovernik, Mark Znidar, Charilaos Kanatsoulis, Roshan Upendra, Mahmoud Mohammadi, Joe Meyer, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. Relational transformer: Toward zero-shot foundation models for relational data. In *International Conference on Learning Representations*, volume 2026, pp. 69149–69174, 2026.

Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan Eric Lenssen, Yiwen Yuan, Zecheng Zhang, et al. RelBench: A benchmark for deep learning on relational databases. *Advances in Neural Information Processing Systems*, 37:21330–21341, 2024.

Minjie Wang, Quan Gan, David Wipf, Zhenkun Cai, Ning Li, Jianheng Tang, Yanlin Zhang, Zizhao Zhang, Zunyao Mao, Yakun Song, et al. 4DBInfer: A 4D benchmarking toolbox for graph-centric predictive modeling on relational DBs. *Advances in Neural Information Processing Systems*, 37, 2024.

Yanbo Wang, Xiyuan Wang, Quan Gan, Minjie Wang, Qibin Yang, David Wipf, and Muhan Zhang. Griffin: Towards a graph-centric relational database foundation model. In *International Conference on Machine Learning*, pp. 64604–64627, 2025.

Yanbo Wang, Jiaxuan You, Chuan Shi, and Muhan Zhang. Relational in-context learning via synthetic pre-training with structural prior. *arXiv preprint arXiv:2603.03805*, 2026.

Linjie Xu, Yanlin Zhang, Quan Gan, Minjie Wang, and David Wipf. No need to train your RDB foundation model. *arXiv preprint arXiv:2602.13697*, 2026.