Learning from Multi-Table Relational Data with the Relational Graph Perceiver

Divyansha Lachi*
University of Pennsylvania
Philadelphia, PA, USA
div11@upenn.edu

Mahmoud Mohammadi
SAP
Seattle, WA, USA
mahmoud.mohammadi@sap.com

Joe Meyer
SAP
Palo Alto, CA, USA
joseph.meyer@sap.com

Vinam Arora
University of Pennsylvania
Philadelphia, PA, USA
vinam@upenn.edu

Shivashriganesh P. Mahato
University of Pennsylvania
Philadelphia, PA, USA
smahato@upenn.edu

Tom Palczewski
SAP
Palo Alto, CA, USA
tom.palczewski@sap.com

Eva L. Dyer
University of Pennsylvania
Philadelphia, PA, USA
eva.dyer@upenn.edu

Abstract

Relational data in domains such as healthcare, finance, and e-commerce capture complex, time-evolving interactions among diverse entities. Models operating on such data must integrate long-range spatial and temporal dependencies while supporting multiple predictive tasks. However, existing relational graph models largely focus on spatial structure, treating time as a constraint rather than a modeling signal, and are typically designed for single-task prediction. We introduce the **Relational Graph Perceiver (RGP)**, a transformer architecture with a Perceiver-style latent bottleneck that integrates signals from diverse node and edge types into a shared latent space for global relational reasoning. RGP also includes a flexible cross-attention decoder for joint multi-task learning across disjoint label spaces and a temporal subgraph sampler that enhances context by retrieving time-relevant nodes beyond local neighborhoods. Experiments on RelBench, SALT, and CTU show that RGP delivers state-of-the-art performance, offering a general and scalable solution for relational deep learning.

1 Introduction

Relational data is central to many real-world systems in domains such as healthcare, finance, and e-commerce, capturing interactions between entities (e.g., patients and providers, customers and products) that evolve over time and span multiple modalities [3]. These datasets are typically stored in multi-table relational databases, presenting modeling challenges that involve both long-range structural dependencies and temporal dynamics. **Relational Deep Learning (RDL)** provides a principled framework for learning from such data by converting relational databases into heterogeneous temporal graphs [9], where nodes correspond to entities and edges represent typed relationships. While Graph Neural Networks (GNNs) have been widely used in RDL, they primarily capture local

*Work done during internship at SAP.

AI for Tabular Data Workshop
39th Conference on Neural Information Processing Systems (EurIPS 2025).

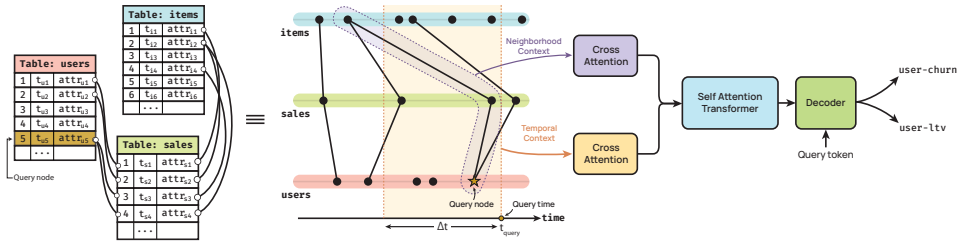

Figure 1: **Overview of the RGP architecture.** We convert relational databases into heterogeneous temporal graphs, where nodes represent entities (e.g., users) and edges capture their interactions. For a given query node, RGP encodes structural and temporal context through parallel cross-attention modules, producing latent tokens processed by self-attention blocks. A lightweight decoder then maps these latents to predictions across tasks such as churn or lifetime value (LTV).

structure and suffer from oversquashing and limited expressiveness [27, 1, 24]. **Graph Transformers (GTs)** offer a promising alternative by using attention mechanisms for global aggregation [4, 28], but most are designed for static, homogeneous graphs and struggle to capture the structural and temporal heterogeneity of relational data [7]. In addition, existing relational graph models, whether GNN [25] or GT [7] based, mainly focus on spatial structure and typically only use time to limit the sampled neighborhood instead of actively guiding context selection.

Our Approach. We propose the **Relational Graph Perceiver (RGP)**, a transformer architecture for heterogeneous temporal graphs. RGP adopts the Perceiver framework [14] to encode relational graphs through a fixed-size latent bottleneck, enabling efficient global reasoning across entity types and time. To go beyond local neighborhoods, RGP includes a *temporal subgraph sampler* that retrieves structure-agnostic, time-relevant nodes that complement conventional neighborhood sampling. In addition, RGP features a *multi-task decoder* that enables joint learning across tasks with disjoint label spaces using similarity-based loss over text-encoded labels. We evaluate RGP on **RelBench** [25], **CTU** [22], and **SALT** [17], spanning binary, multi-class, and ranking tasks. RGP achieves strong results across all settings while enabling scalable, parameter-efficient multi-task learning.

Our contributions are as follows:

- We present the **Relational Graph Perceiver (RGP)**, the first Perceiver-based graph transformer architecture tailored for heterogeneous temporal graphs. RGP enables efficient global reasoning across relational data.
- We introduce a novel **temporal subgraph sampler** that selects nodes based on timestamp proximity, allowing the model to incorporate nonlocal temporal context beyond structural neighborhoods.
- We develop a **flexible multi-task decoder** that enables joint training across tasks with disjoint label spaces using task-conditioned queries and similarity-based loss over text-encoded labels.

2 Method

We introduce the **Relational Graph Perceiver (RGP)**, a general-purpose transformer architecture for heterogeneous temporal graphs. RGP consists of three components (Figure 1): (i) a temporal context sampler retrieving time-relevant nodes beyond the local neighborhood, (ii) a Perceiver-style encoder compressing variable-size relational subgraphs into fixed-length latent representations, and (iii) a flexible multi-task decoder supporting diverse prediction tasks without task-specific heads.

Tokenizing Heterogeneous Temporal Graphs We first convert relational databases into heterogeneous temporal graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \phi, \psi, \tau)$, where nodes $\mathcal{V}$ represent entities and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denote typed relations annotated with timestamps (Figure 1). Here, $\phi : \mathcal{V} \to \mathcal{T}_V$ maps each node to its entity type, $\psi : \mathcal{E} \to \mathcal{T}_E$ assigns relation types to edges, and $\tau : \mathcal{E} \cup \mathcal{V} \to \mathbb{R}$ provides temporal information for both nodes and edges. Each node v_i is represented by a token embedding $\mathbf{x}_i = \texttt{MultiModalEncoder}(\mathbf{u}_i) + \mathrm{PE}(v_i)$, where $\mathbf{u}_i$ denotes raw attributes and $\mathrm{PE}(v_i)$ is a composite positional encoding that integrates structural and temporal cues such as hop distance and centrality [7]: $\mathrm{PE}(v_i) = W_{\mathrm{PE}}[\,\mathbf{e}_{\text{type}} \| \mathbf{e}_{\text{cent}} \| \mathbf{e}_{\text{hop}} \| \mathbf{e}_{\text{time}}\,]$, with W_{PE} as a learned projection matrix. For more details about tokenization refer to Appendix A.2.

Temporal Sampler Standard neighborhood sampling methods restrict temporal windows around a target node to prevent information leakage [7, 25], treating time mainly as a boundary constraint on

graph-based sampling. In contrast, we introduce a *time-context sampler* that uses temporal proximity itself as a modeling signal, independent of graph connectivity. This sampler retrieves nodes and edges that are close in time to the query entity, allowing the model to incorporate nonlocal but temporally co-occurring events—such as concurrent user actions that enrich predictive context. Formally, given a graph $G = (V, E)$, node timestamps $T_v : V \to \mathbb{R}$, and a seed time t_{seed}, we assign timestamps to edges based on the node timestamps and select a subgraph using either a fixed time window Δt or the k closest edges in time. The full algorithm is provided in Appendix A.3.2. Nodes selected by the temporal sampler are processed in parallel with structurally sampled neighbors.

Compression via Cross Attention Standard self-attention has quadratic complexity $\mathcal{O}(N_g^2)$, making it infeasible for large graphs. RGP addresses this by adopting a Perceiver-style encoder [13, 19] that replaces dense pairwise attention with a small set of learnable latent tokens acting as an information bottleneck. These latents attend to input nodes through cross-attention, enabling efficient global aggregation.

Given the tokenized and sampled subgraph for each query entity, RGP applies the Perceiver encoder to aggregate information across all nodes efficiently. Let $\mathbf{X}_g = [\mathbf{x}_1, \ldots, \mathbf{x}_{N_g}]$ denote the sequence of node embeddings within the subgraph. We maintain $K \ll N_g$ learnable latent tokens $\mathbf{Z}_0 = [\mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,K}]$ that attend to the node embeddings through cross-attention:

$$\mathbf{Z}_g^{(1)} = \mathbf{Z}^{(0)} + \text{softmax}\Big(\mathbf{Q}\mathbf{K}_g^\top / \sqrt{d_k}\Big)\,\mathbf{V}_g, \tag{1}$$

where $\mathbf{Q} = \mathbf{W}_q\mathbf{Z}_0$, $\mathbf{K}_g = \mathbf{W}_k\mathbf{X}_g$, and $\mathbf{V}_g = \mathbf{W}_v\mathbf{X}_g$. Two parallel cross-attention branches independently process nodes from structural and temporal samplers, and their latent outputs are summed to form a unified latent embedding. The latents are then processed by L Transformer blocks operating in latent space, producing the final representation $\mathbf{Z}_g^{\text{out}}$. This design scales as $\mathcal{O}(KN_g + LK^2) \ll \mathcal{O}(N_g^2)$, offering large computational and memory savings compared to full Graph Transformers while retaining global context integration.

Multi-Task Decoder After encoding, the latent representation is mapped to task-specific predictions through a multi-task decoder that combines cross-attention with similarity-based loss. Task descriptions and labels are encoded using a frozen text encoder. Given a task embedding $\mathbf{q}_{\text{task}}$ and the tokenized target node $\mathbf{x}_i$, we form a task-aware query $\mathbf{q}_i = \mathbf{x}_i + \mathbf{q}_{\text{task}}$ which attends to the encoder output $\mathbf{Z}_g^{\text{out}} \in \mathbb{R}^{K\times d}$ via cross-attention, $\mathbf{z}_i = \text{CrossAttn}(\mathbf{q}_i, \mathbf{Z}_g^{\text{out}})$. Label embeddings $\mathbf{E}_{\text{label}}$ are used to compute logits $\mathbf{s}_i = \mathbf{E}_{\text{label}}\mathbf{z}_i$, followed by a softmax and cross-entropy loss. This unified decoding scheme enables scalable multi-task learning across disjoint label spaces without task-specific output heads.

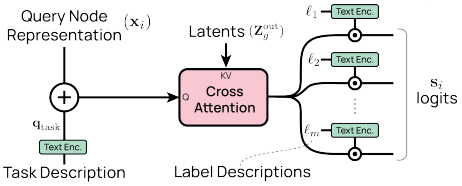

Figure 2: **Overview of Flexible multi-task decoder**: The decoder receives a query node representation (combined with a task description embedding) and attends to the latents from the perceiver encoder via cross attention.

3 Results

We evaluate the Relational Graph Perceiver (RGP) on three heterogeneous temporal graph benchmarks—RelBench[25], CTU[22], and SALT[17]. Our experiments focus on the node (or entity) classification task, where the goal is to predict categorical attributes of entities in relational graphs. We implement RGP within the RDL pipeline [25], replacing the original GNN component while retaining the existing task logic and infrastructure. Following prior work, we tune only key architectural parameters such as the number of layers and latent tokens, while keeping other settings fixed across datasets. Please refer to Appendix A.4.1 for more details. Due to the diversity of these benchmarks, we report results separately for each and compare RGP against the publicly available baselines in each setting. Across all benchmarks, we also include comparisons with RDL [8], a widely adopted pipeline that combines RelGNN [2] with GraphSAGE aggregation, serving as a strong reference point for relational deep learning tasks.

Results on Benchmarks On **RelBench** [25], a recent benchmark for relational deep learning comprising seven datasets from domains such as e-commerce, social networks, and sports, RGP achieves state-of-the-art performance (Table 1), outperforming RelGT on 10 out of 12 tasks. Notably,

Table 1: **Results on Relbench:** We report Area Under the ROC Curve (AUC) as the evaluation metric. Best values are in **bold**, and relative gains show the percentage improvement of RGP over RelGT.

Dataset	Task	RDL	HGT	HGT+PE	RelGT	**RGP (ours)**	% Rel Gain vs. RelGT
rel-f1	driver-dnf	0.7262	0.7142	0.7109	0.7587	**0.7844**	+3.39
	driver-top3	0.7554	0.6389	0.8340	0.8352	**0.8789**	+5.22
rel-avito	user-clicks	0.6590	0.6584	0.6387	0.6830	**0.6943**	+1.66
	user-visits	0.6620	0.6426	0.6507	**0.6678**	0.6662	-0.24
rel-event	user-repeat	0.7689	0.6717	0.6590	0.7609	**0.7894**	+3.75
	user-ignore	0.8162	0.8348	0.8161	0.8157	**0.8439**	+3.46
rel-trial	study-outcome	0.6860	0.5679	0.5691	0.6861	**0.7027**	+2.42
rel-amazon	user-churn	0.7042	0.6608	0.6589	0.7039	**0.7089**	+0.71
	item-churn	**0.8281**	0.7824	0.7840	0.8255	0.8262	+0.08
rel-stack	user-engagement	0.9021	0.8898	0.8818	**0.9053**	0.9045	-0.09
	user-badge	**0.8966**	0.8652	0.8636	0.8624	0.8868	+2.83
rel-hm	user-churn	0.6988	0.6773	0.6491	0.6927	**0.7025**	+1.41
						Average Gain vs. RelGT (%)	**2.20**

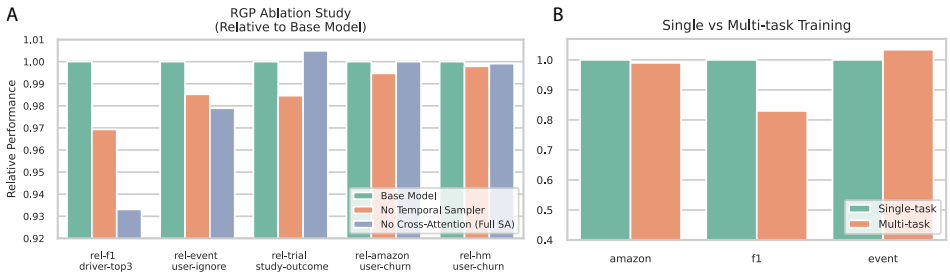

Figure 3: **(A)** Ablation results showing performance impact of removing key RGP components. Values are reported relative to the full model. **(B)** Relative performance of multi-task vs. single-task training: the Y-axis shows the average multi-task performance normalized to the average single-task performance across all tasks within each dataset.

RGP yields an average relative improvement of **2.2%** over RelGT across all tasks. Gains are particularly pronounced on smaller datasets, where self-attention-based models like RelGT are more prone to overfitting. For example, on the driver-top3 task from the F1 dataset, RGP outperforms RelGT by **3.39%**, and on the driver-dnf task, by **5.22%**. Next, we evaluated on **CTU** [22], a collection of heterogeneous graphs from domains such as insurance, law, and retail with multi-class classification tasks. We report F1 scores following standard practice from ReDeLEx [23]. As shown in Table 5, RGP consistently outperforms the DBFormer baseline across all datasets, achieving up to a **8.19%** gain on *dallas*. However, we note that tree-based methods such as LightGBM sometimes outperform RDL-based approaches on two out of three CTU tasks. Nevertheless, RGP consistently achieves best performance within RDL based methods. Finally, on **SALT**[17], a real-world dataset derived from an enterprise resource planning (ERP) system, RGP outperforms both baselines on three of four tasks (Table 4). On *sales-payterms*, it performs slightly below HGT (0.58 vs. 0.60 MRR) while still surpassing RDL. Across all benchmarks, RGP outperforms strong baselines across binary, multi-class, and ranking tasks, demonstrating its versatility as a general relational graph model.

Ablation Study To better understand the key components of our model, we ablate (1) the temporal context sampler and (2) the Perceiver bottleneck. Relative performance to the full model is shown in Figure 3A. Removing the **temporal sampler**, which retrieves nodes based on temporal proximity, leads to consistent performance drops—most notably a $\sim$3% decline on *rel-f1*, which includes many cold-start driver nodes lacking structural history. Replacing the **Perceiver bottleneck** with full self-attention yields slight improvements on some tasks but introduces significant computational overhead and overfitting on smaller datasets (e.g., a $\sim$6.6% drop on *driver-top3*). Overall, the Perceiver's cross-attention design provides a more efficient and regularized alternative, maintaining strong performance while being 2–6% more compute-efficient than full self-attention (see Appendix A.4.2).

Multi Task Results Finally, we evaluate the **multi-task decoder**, which enables scalable learning across diverse label spaces. Unlike prior task-specific approaches, RGP performs task-conditioned decoding via cross-attention and similarity-based supervision over text-encoded labels, allowing a

single model to handle multiple tasks jointly. As shown in Figure 3B, multi-task training matches or surpasses single-task performance while reducing training cost. It improves results on the *event* dataset but slightly lowers on *f1* due to task imbalance (1.5k vs. 10k samples). Overall, RGP enables efficient and generalizable multi-task learning without retraining or architectural changes.

4 Conclusion

We introduced RGP, a Transformer-based architecture for heterogeneous temporal graphs that combines a Perceiver-style encoder for scalable global reasoning, a temporal subgraph sampler for capturing temporally relevant context, and a flexible multi-task decoder for joint prediction across diverse label spaces. Across multiple benchmarks, RGP achieves state-of-the-art performance while reducing computational cost. Ablations confirm the value of temporal context and the Perceiver bottleneck, and the unified decoder enables efficient multi-task learning—positioning RGP as a strong candidate for large-scale foundation models in relational domains.

Acknowledgements and Funding Disclosure

Thanks to Keertika Saroj for insightful discussions and feedback on this manuscript. This project was also supported by NSF CAREER Award RI:2146072, NSF award CIF:RI:2212182 as well as generous gifts from the CIFAR Azrieli Global Scholars Program (D.L, V.A and E.L.D).

References

[1] M. Black, Z. Wan, A. Nayyeri, and Y. Wang. Understanding oversquashing in gnns through the lens of effective resistance. In *International Conference on Machine Learning*, pages 2528–2547. PMLR, 2023.

[2] T. Chen, C. Kanatsoulis, and J. Leskovec. Relgnn: Composite message passing for relational deep learning. *arXiv preprint arXiv:2502.06784*, 2025.

[3] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[4] V. P. Dwivedi and X. Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[5] V. P. Dwivedi and M. M. Bronstein. A generalization of transformer networks to graphs. *AAAI DLG Workshop*, 2021.

[6] V. P. Dwivedi et al. Graph neural networks with learnable structural and positional representations. *ICLR 2022*, 2022.

[7] V. P. Dwivedi, S. Jaladi, Y. Shen, F. López, C. I. Kanatsoulis, R. Puri, M. Fey, and J. Leskovec. Relational graph transformer. *arXiv preprint arXiv:2505.10960*, 2025.

[8] M. Fey, W. Hu, K. Huang, J. E. Lenssen, R. Ranjan, J. Robinson, R. Ying, J. You, and J. Leskovec. Relational deep learning: Graph representation learning on relational databases. *arXiv preprint arXiv:2312.04615*, 2023.

[9] M. Fey, W. Hu, K. Huang, J. E. Lenssen, R. Ranjan, J. Robinson, R. Ying, J. You, and J. Leskovec. Position: Relational deep learning-graph representation learning on relational databases. In *Forty-first International Conference on Machine Learning*, 2024.

[10] W. L. Hamilton, Z. Ying, and J. Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, pages 1024–1034, 2017.

[11] W. Hu, Y. Yuan, Z. Zhang, A. Nitta, K. Cao, V. Kocijan, J. Sunil, J. Leskovec, and M. Fey. Pytorch frame: A modular framework for multi-modal tabular learning. *arXiv preprint arXiv:2404.00776*, 2024.

[12] Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pages 2704–2710, 2020.

[13] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, A. Brock, E. Shelhamer, O. Hénaff, M. M. Botvinick, A. Zisserman, O. Vinyals, and J. Carreira. Perceiver io: A general architecture for structured inputs & outputs. In *ICLR 2022 (Spotlight)*, 2022. Also available as arXiv:2107.14795.

[14] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021.

[15] B. Jiang and D. Ma. Defining least community as a homogeneous group in complex networks. *Physica A: Statistical Mechanics and its Applications*, 428:154–160, 2015.

[16] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[17] T. Klein, C. Biehl, M. Costa, A. Sres, J. Kolk, and J. Hoffart. Salt: Sales autocompletion linked business tables dataset. *arXiv preprint arXiv:2501.03413*, 2025.

[18] D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

[19] D. Lachi, M. Azabou, V. Arora, and E. Dyer. Graphfm: A scalable framework for multi-graph pretraining. *arXiv preprint arXiv:2407.11907*, 2024.

[20] Q. Mao, Z. Liu, C. Liu, and J. Sun. Hinormer: Representation learning on heterogeneous information networks with graph transformer. In *Proceedings of the ACM web conference 2023*, pages 599–610, 2023.

[21] G. Mialon, D. Chen, M. Selosse, and J. Mairal. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021.

[22] J. Motl and O. Schulte. The ctu prague relational learning repository, 2024.

[23] J. Peleška and G. Šír. Redelex: A framework for relational deep learning exploration. *arXiv preprint arXiv:2506.22199*, 2025.

[24] S. Qureshi et al. Limits of depth: Over-smoothing and over-squashing in gnns. *Big Data Mining and Analytics*, 7(1):205–216, 2023.

[25] J. Robinson, R. Ranjan, W. Hu, K. Huang, J. Han, A. Dobles, M. Fey, J. E. Lenssen, Y. Yuan, Z. Zhang, X. He, and J. Leskovec. Relbench: A benchmark for deep learning on relational databases. In *NeurIPS 2024 Datasets and Benchmarks Track*, 2024. Poster; also available as arXiv:2407.20060.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[27] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[28] R. Ying, Z. Liu, J. You, Y. Zhou, C. Li, M. Jiang, and J. Leskovec. Do transformers really perform badly for graph representation? In *ICLR 2021*, 2021.

[29] Y. Yuan, Z. Zhang, X. He, A. Nitta, W. Hu, D. Wang, M. Shah, S. Huang, B. Stojanovič, A. Krumholz, et al. Contextgnn: Beyond two-tower recommendation systems. *arXiv preprint arXiv:2411.19513*, 2024.

A Appendix

A.1 Related Work

Representing relational datasets as heterogeneous temporal graphs has enabled the use of graph learning methods for tasks like node classification and link prediction. The RelBench benchmark[25] formalizes this paradigm and provides a strong baseline using Heterogeneous GraphSAGE [10] with temporal neighbor sampling, surpassing classical tabular methods like LightGBM [16]. Several architectures have been proposed to better exploit relational structure: RelGNN [2] introduces composite message passing to preserve information across bridge and hub nodes, while ContextGNN [29] combines pair-wise and two-tower encoders for recommendation scenarios.

Graph Transformers (GTs) adapt the self-attention paradigm [26] to graph-structured data, capturing long-range dependencies without iterative neighborhood aggregation [5]. Early GT variants focused on local attention with positional encodings like Laplacian eigenvectors [6], while later designs incorporated global attention mechanisms [28, 21, 18] and scaling strategies such as hierarchical pooling or sparse attention. Heterogeneous GTs such as HGT [12] and Hinormer [20] model multi-type graphs, but face quadratic cost and per-task training inefficiency. Most relevant to our work is the Relational Graph Transformer (RelGT) [7], which introduced a hybrid local/global attention, establishing strong performance on RelBench. Our approach differs by adopting a Perceiver-style latent bottleneck and temporal sampling, enabling greater scalability and multi-task capability.

While GTs highlight the importance of global context, their computational footprint motivates exploring efficiency-focused alternatives. The Perceiver architecture [14] introduces a fixed-size latent array that attends to high-dimensional inputs via cross-attention, followed by latent self-attention, decoupling input size from computational complexity. Perceiver IO [13] extends this framework to handle diverse output domains from a shared latent representation. In graph learning, early Perceiver-inspired adaptations have been proposed for homogeneous graphs or static settings [15], but these typically omit temporal sampling and are not optimized for multi-task inference. Our encoder adapts this latent bottleneck principle specifically to heterogeneous temporal graphs.

Existing RDL methods excel at modeling relational structure but face oversquashing and limited temporal reach. Graph Transformers capture long-range context but remain computationally heavy and often schema-restrictive. Perceiver-based models address scalability but have not been tailored to heterogeneous temporal graphs or multi-task learning in relational settings. This motivates architectures like ours that combine the latent efficiency of Perceivers with relational and temporal inductive biases, while enabling shared encoders across multiple tasks.

A.2 Tokenizing Heterogeneous Temporal Graphs

To process relational data with transformer-based models, we first convert relational databases into graph-structured inputs (Figure 1), enabling end-to-end learning without the need for manual feature engineering. Following prior work in relational deep learning [7, 25], we represent relational databases as *relational entity graphs* (REGs), modeled as heterogeneous temporal graphs.

Relational Graph: A relational database can be formally described as a tuple $(\mathcal{T}, \mathcal{R})$, where $\mathcal{T} = T_1, \ldots, T_n$ is a collection of entity tables, and $\mathcal{R} \subseteq \mathcal{T} \times \mathcal{T}$ is a set of inter-table relationships. Each relation $(T_{\text{fkey}}, T_{\text{pkey}}) \in \mathcal{R}$ denotes a foreign-key reference from one table to the primary key of another. Each table T_i contains a set of entities (rows), where each entity is typically defined by (1) a unique identifier, (2) foreign-key references, (3) entity-specific attributes (e.g., numeric, categorical), and (4) timestamp metadata.

We transform this database into a heterogeneous temporal graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \phi, \psi, \tau)$ where $\mathcal{V}$ is the set of nodes (entities), $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges representing primary-foreign key relationships, $\phi : \mathcal{V} \to \mathcal{T}_V$ maps each node to its source table (entity type), $\psi : \mathcal{E} \to \mathcal{T}_E$ assigns relation types to edges, and $\tau : \mathcal{E} \cup \mathcal{V} \to \mathbb{R}$ associates timestamps with both nodes and edges. This graph representation captures both the schema structure and temporal dynamics of the database.

Token Construction. Each node $v_i \in \mathcal{V}$ is mapped to a token embedding $\mathbf{x}_i \in \mathbb{R}^d$ by applying a multi-modal encoder to its raw attributes, followed by the addition of a positional encoding:

$$\mathbf{x}_i = \texttt{MultiModalEncoder}(\mathbf{u}_i) + \text{PE}(v_i),$$

where $\mathbf{u}_i$ denotes the raw attributes of the node (e.g., tabular, categorical, or multi-modal features), `MultiModalEncoder` is the modality-aware encoder taken from [11]. This encoder applies separate encoders for each modality (e.g., numerical, categorical, or text) and aggregates their outputs into a unified embedding using a ResNet (see Appendix A.3.1 for more details). $\mathrm{PE}(v_i)$ captures structural and temporal context such as node centrality, hop distances, or timestamp embeddings. Each input graph is mapped to a full input sequence formed by these node-level tokens.

Positional Encodings. To represent the position of each node in a heterogeneous and temporal relational graph, we combine multiple structure and time-aware signals into a unified encoding. Specifically, for each node v_i, we compute:

- **Node type embedding** $\mathbf{e}_{\text{type}}(v_i)$: a learned embedding based on the node type $\phi(v_i)$.
- **Centrality embedding** $\mathbf{e}_{\text{cent}}(v_i)$: a linear projection of centrality scores (e.g., degree, PageRank).
- **Hop distance embedding** $\mathbf{e}_{\text{hop}}(v_i)$: a learned embedding of the hop distance from a designated entity node (e.g., the seed node or query node in the task).
- **Relative time encoding** $\mathbf{e}_{\text{time}}(v_i)$: a projection of $\tau(v_i) - \tau_{\text{seed}}$ to capture temporal alignment.

We concatenate these components and project them into the final positional encoding:

$$\mathrm{PE}(v_i) = W_{\text{PE}} \cdot [\mathbf{e}_{\text{type}}(v_i) \| \mathbf{e}_{\text{cent}}(v_i) \| \mathbf{e}_{\text{hop}}(v_i) \| \mathbf{e}_{\text{time}}(v_i)] \,,$$

where $W_{\text{PE}} \in \mathbb{R}^{d' \times d}$ is a learned projection matrix and $\|$ denotes concatenation.

Remarks. This multi-element positional encoding allows the model to incorporate fine-grained structural, temporal, and schema-level context across highly diverse relational graphs. While the individual components of this encoding are adapted from established techniques [7, 28], their integration provides a unified representation that is suitable for heterogeneous graph modeling.

Once tokenized, we construct an input subgraph around each target entity using a combination of structural sampling [25] and temporal context sampling. The resulting node sequence is then passed to our Perceiver-based encoder, which compresses it into a fixed-size latent representation via cross-attention.

A.3 Model Details

A.3.1 Multi Model Encoder

In heterogeneous temporal relational graphs derived from relational databases, each node may contain a rich set of multi-modal attributes, such as numerical values, categorical fields, text descriptions, timestamps, and image features. To obtain expressive node-level representations suitable for downstream tasks, we use a modality-aware feature encoder taken from prior work in relational deep learning [25, 11].

Given a node $v \in \mathcal{V}$, we denote its raw attributes as $\mathbf{x}_v = \{\mathbf{x}_v^{(m)}\}_{m \in \mathcal{M}_v}$, where $\mathcal{M}_v \subseteq \mathcal{M}$ is the subset of modalities present for node v. Each feature within each modality is independently encoded using a modality-specific function ϕ_m, yielding a set of intermediate embeddings:

$$\mathbf{h}_v^{(m)} = \phi_m \left(\mathbf{x}_v^{(m)} \right), \quad \phi_m : \mathcal{X}^{(m)} \to \mathbb{R}^{d_m}, \tag{2}$$

where d_m is the dimensionality assigned to modality m. Supported modalities include:

- **Numerical features:** encoded via linear layers or small MLPs.
- **Categorical features:** embedded using learned lookup tables.
- **Text features:** embedded via pretrained or fine-tuned language models (e.g., BERT).
- **Timestamps:** embedded as scalar values or periodic functions (e.g., sinusoidal encodings).

The resulting embeddings are concatenated:

$$\mathbf{h}_v^{\text{concat}} = \bigoplus_{m \in \mathcal{M}_v} \mathbf{h}_v^{(m)} \in \mathbb{R}^{\sum_{m \in \mathcal{M}_v} d_m}, \tag{3}$$

and passed through a table-specific (or node-type-specific) projection function—a ResNet in our case—to obtain the final node representation:

$$\mathbf{h}_v^{(0)} = f_{\tau(v)}\left(\mathbf{h}_v^{\text{concat}}\right), \quad f_{\tau(v)} : \mathbb{R}^{\sum d_m} \to \mathbb{R}^d, \tag{4}$$

where $\tau(v)$ denotes the node type (i.e., source table) and d is the unified hidden dimension used across the model.

A.3.2 Time-Context Sampling Algorithm

Edge Timestamp Assignment. Each edge is assigned a timestamp equal to the maximum timestamp of its two endpoint nodes:

$$T_e(u, v) = \max\left(T_v(u), T_v(v)\right).$$

Time-Context-Aware Sampling. Given a reference timestamp t_{seed}, we sample a subgraph by selecting edges based on one of two strategies:

- Time window: include all edges where $|T_e(u, v) - t_{\text{seed}}| \leq \Delta t$.
- Top-k: select the k edges closest in time to t_{seed}.

The resulting subgraph contains all nodes incident to the selected edges.

Algorithm 1 Time-Context-Aware Edge Sampling

Input: Graph $G = (V, E)$, timestamp function $T : V \to \mathbb{R}$, reference time t_{seed}, sampling rule (Δt or k)
Output: Subgraph $G_{\text{sub}} = (V_{\text{sub}}, E_{\text{sub}})$

Initialize $E_{\text{scored}} \leftarrow \emptyset$
for each edge $(u, v) \in E$ **do**
  $t_e \leftarrow \max(T(u), T(v))$
  score $\leftarrow |t_e - t_{\text{seed}}|$
  Add $(u, v, t_e, \text{score})$ to E_{scored}
end for

if time window Δt specified **then**
  $E_{\text{selected}} \leftarrow \{(u, v) \mid (u, v, t_e, s) \in E_{\text{scored}},\ s \leq \Delta t\}$
else if edge count k specified **then**
  Sort E_{scored} ascending by score
  $E_{\text{selected}} \leftarrow$ first k edges
else
  $E_{\text{selected}} \leftarrow E$
end if

$V_{\text{sub}} \leftarrow$ nodes appearing in E_{selected}
return $G_{\text{sub}} = (V_{\text{sub}}, E_{\text{selected}})$

A.4 Experiment details

A.4.1 Hyperparameter

We use a controlled hyperparameter setup to ensure consistent evaluation across diverse datasets without exhaustive tuning. Following prior work, we fix most settings and tune only two architectural hyperparameters: the number of Perceiver layers $L \in \{2, 4, 6\}$ and the number of latent tokens $n \in \{8, 16, 32\}$ in the cross-attention bottleneck. All other hyperparameters are kept fixed across datasets to reflect realistic training scenarios where compute budgets limit per-task tuning.

Table 2 summarizes the fixed hyperparameter settings used in our experiments. All models are trained using the AdamW optimizer with a learning rate of 10^{-3} and a weight decay of 10^{-5}. We use a cosine learning rate scheduler with 10 warmup steps, and all experiments are run for 200 epochs.

Table 2: Summary of hyperparameter settings used in RGP experiments.

Component	Value / Setting
Optimizer and Training Setup	
Optimizer	AdamW
Learning rate	10^{-3}
Weight decay	10^{-5}
Scheduler	Cosine with 10 warmup steps
Epochs	200
Batch size	512
Model Architecture	
Latent token count n	{8, 16, 32} (tuned)
Transformer layers L	{2, 4, 6} (tuned)
Dropout	0.2
hidden dim	128
Temporal Sampler	
Edges per type	10
Temporal decay	0.1

A.4.2 Runtime Comparison: Cross-Attention vs Self-Attention

To quantify the efficiency benefits of using a cross-attention (CA) bottleneck over full self-attention (SA), we measure the total training time across three representative tasks. As shown in Table 3, the Perceiver-based encoder with CA consistently achieves lower runtime compared to its SA counterpart. While SA sometimes offers marginal accuracy improvements on larger datasets, the increase in computational cost is substantial.

Table 3: Average training time for Cross-Attention (CA) vs. Self-Attention (SA) models. Experiments were run on a single B200 GPU.

Dataset	Cross-Attention (CA)	Self-Attention (SA)
rel-f1	2.7 min	7.5 min
rel-amazon	3.5 hrs	18.5 hrs
rel-event	4.5 min	22 min

B Results on additional datasets

Tables 5 and 4 present detailed results for the CTU[22] and SALT[17] benchmarks. For CTU, we follow the evaluation protocol from ReDeLEx [23] and report macro-F1 scores across multiple heterogeneous graph datasets. RGP consistently outperforms the DBFormer baseline, achieving up to an 8.19% gain on dallas. For SALT, derived from an enterprise ERP system, we report MRR scores for four relational prediction tasks. RGP surpasses both baselines on three out of four tasks, with slightly lower performance on sales-payterms (0.58 vs. 0.60 for HGT). Overall, these extended results confirm RGP's strong and consistent performance across diverse relational domains.

Table 4: **Results on SALT:** We report MRR score as the evaluation metric. Best values are in **bold**. Relative gains are percentage improvement over HGT.

Task	RDL	HGT	**RGP (ours)**	% Rel Gain v.s HGT
item-incoterms	0.64	0.75	**0.81**	+8.00
sales-group	0.20	0.31	**0.34**	+9.68
sales-payterms	0.39	**0.60**	0.58	-3.33
sales-shipcond	0.59	0.76	**0.81**	+6.58

Table 5: **Results on CTU benchmarks:** We report F1 score as the evaluation metric. Best values are in **bold**. Relative gains are percentage improvement over DBFormer.

Dataset	Task	LightGBM	XGBoost	TabResNet	Linear	SAGE (RDL)	DBFormer	**RGP (ours)**	**Rel.** (%) Gain vs. DBF
accidents	temp.	0.170	0.336	0.187	0.583	0.566	0.727	**0.743**	+2.20
dallas	temp.	**0.584**	0.512	0.247	0.393	0.424	0.513	0.555	+8.19
legalacts	temp.	**0.851**	0.220	0.220	0.721	0.698	0.703	0.736	+4.69