Graph Representations for Relational Deep Learning

Tamara Cucumides[2], Pablo Barceló[1], and Floris Geerts[2]

[1] Universidad Católica de Chile & IMFD Chile CENIA Chile
pbarcelo@uc.cl
[2] University of Antwerp, Belgium
{tamara.cucumidesfaundez,floris.geerts}@uantwerp.be

Abstract. We explore different graph construction strategies for relational deep learning, revealing new ways to optimize representation efficiency and expressivity. Our work extends existing methods and highlights new research directions in relational learning.

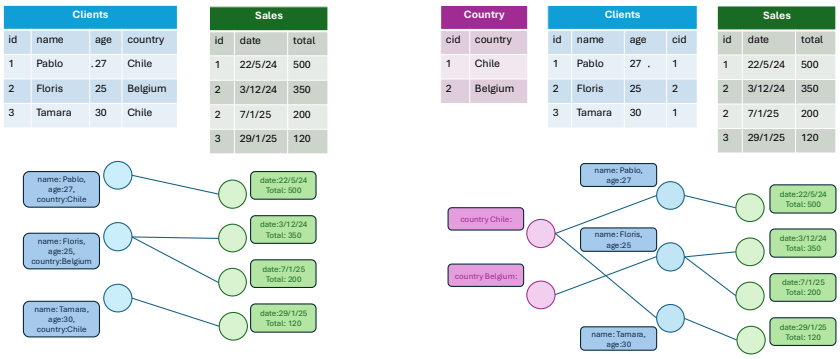
Introduction. Machine learning (ML) on structured data, particularly relational databases, has gained attention as an alternative to traditional feature engineering. Relational Deep Learning [1] proposes modeling relational databases as temporal, heterogeneous graphs, where rows correspond to nodes, and primary-foreign key constraints define edges. Message Passing Graph Neural Networks (GNNs) can then automatically learn representations over this structured data to later tackle different tasks, such as node classification, regression and link prediction.

However, a fundamental challenge in this approach is that the graph structure is highly dependent on database design choices. The same underlying data can lead to different graphs depending on normalization levels, key constraints, and schema design. This raises critical questions: (1) How do different graph construction strategies impact learned representations? (2) How can transformations balance information preservation and efficiency? (3) What are the trade-offs between fully preserving relational structure and simplifying it?

In this work, we explore different strategies for transforming relational databases into graphs and their potential impact on ML tasks.

Different graph constructions for relational databases The standard approach in Relational Deep Learning constructs graphs where: (i) Each tuple (row) in a table becomes a node, (ii) edges are defined by primary-foreign key relationships, and (iii) attributes are stored as node features. However, other graph constructions could be explored.

In particular, we propose to consider the scenario in which some attributes are moved to separate tables by introducing additional key dependencies (or *hyper-normalizing*). By introducing new key dependencies, this process creates additional nodes representing attribute values, connecting previously unrelated entities through shared attributes. (see Figure 1). In an extreme case, we can create a fine-grained graph representation by encoding each different data value as



(a) Graph construction for schema Clients(id, name, age, country), Sales(id, date, total)

(b) Graph construction for schema Clients(id, name, age, cid) Sales(id, date, total), Country(cid, country)

Fig. 1. Figures (a) and (b) illustrate two different graph constructions for the same data, based on different schema choices. In Figure (a), the Country attribute is stored as a node attribute, whereas in Figure (b), it is pushed into a separate table, introducing new nodes and connections in the graph.

nodes in the graph, which resembles the relational graph representation proposed in [2]. Other graph constructions strategies can be explored using compact data structures that reduce redundancy. For example, by using graph constructions inspired by factorized data representations [4], one can incorporate dependency structures between attribute values.

Beyond graphs, hypergraphs offer an alternative representation (see e.g.[5]). When nodes represent tuples, hyperedges can indicate shared attributes among them. Alternatively, if nodes represent data values, both tuples and key constraints can be modeled as hyperedges. For representation learning, GNNs designed for hypergraphs [3] provide a suitable framework.

Discussion and open questions When transforming relational databases into graphs for learning tasks, the choice of graph construction directly impacts:

- **Expressivity.** Which properties of the original relational database are preserved in the graph representation? How do these choices influence the types of tasks that can be effectively performed on the graph?
- **Complexity.** Constructing a graph from a relational database can be computationally expensive. Additionally, the resulting graph's structure (e.g., size, density) directly affects the efficiency and feasibility of learning tasks performed on it.

We aim to systematically explore how different graph representations of relational databases impact Relational Deep Learning, bridging machine learning and database research to establish principled graph construction methods for improved representation learning and predictive tasks.

References

1. Fey, M., Hu, W., Huang, K., Lenssen, J.E., Ranjan, R., Robinson, J., Ying, R., You, J., Leskovec, J.: Position: Relational deep learning - graph representation learning on relational databases. In: Forty-first International Conference on Machine Learning (2024), https://openreview.net/forum?id=BIMSHniyCP
2. Grohe, M.: word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. In: Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems. pp. 1–16 (2020)
3. Huang, X., Orth, M.R., Barceló, P., Bronstein, M.M., Ceylan, İ.İ.: Link prediction with relational hypergraphs. arXiv preprint arXiv:2402.04062 (2024), https://arxiv.org/abs/2402.04062
4. Olteanu, D., Schleich, M.: Factorized databases. SIGMOD Rec. **45**(2), 5–16 (Sep 2016). https://doi.org/10.1145/3003665.3003667, https://doi.org/10.1145/3003665.3003667
5. Zahradník, L., Neumann, J., Šír, G.: A deep learning blueprint for relational databases. In: NeurIPS 2023 Second Table Representation Learning Workshop (2023), https://openreview.net/pdf?id=b4GEmjsHAB