Phantom Edges: Expanding Functionality in Graph Databases

Adam Pazdor
umpazdor@umanitoba.ca
University of Manitoba
Winnipeg, Manitoba, Canada

Dr. Carson Leung
Carson.Leung@umanitoba.ca
University of Manitoba
Winnipeg, Manitoba, Canada

Abstract

Graphs have been used in the field of data science for many years, both as a visual model for problems naturally suited to networks (pathfinding, shortest path, the Travelling Salesman problem), and as a way to model databases whose interactions and content lend themselves better to graphs than to the more traditional relational model. Graphs have seen applications in almost every discipline, from biology to computer networks to social networks. However, for as versatile as graphs are, there are still many areas available for expansion.

The standard graph model consists strictly of two components: individual nodes and edges connecting those nodes. This is a flexible framework, but not all data translates neatly into those distinct pieces, and not all data suited to being an edge has only two endpoints. Moreover, existing solutions to these sorts of problems (such as hypergraphs) are not in widespread usage in either academia or industry, limiting their usefulness. In my thesis work, I propose a new graph design model that can sit on top of existing graph architecture and still provide a solution to difficulties in relational-database-to-graph translation.

CCS Concepts

• **Theory of computation** → *Data modeling*; • **Mathematics of computing** → *Graph theory*; • **Information systems** → **Graph-based database models**.

Keywords

graphs, data modeling, databases, data visualization

ACM Reference Format:
Adam Pazdor and Dr. Carson Leung. 2025. Phantom Edges: Expanding Functionality in Graph Databases. In *Proceedings of Knowledge Discovery and Data Mining (KDD '25).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

As technology has improved and, in particular, as visual software has become more powerful and more readily accessible, the use of graphs in data science has only increased. As a flexible visual medium, a graph is more easily absorbed and understood (especially to a lay person) than a table. In this current age with focus on topics like explainable AI (XAI) and the public's awareness of the increasing prominence of black-box technologies in their daily lives, it has become more important than ever that data scientists can not only effectively analyze data, but present their results in a way that others can quickly and easily understand.

Unfortunately, while graphs may be easier for humans to process, they are not perfect. For one, they are relatively new in the data science space as compared to the relational model. The vast majority of data in the world is (and probably will continue to be for the foreseeable future) stored in tables. Conversion methods to and from graphs exist [2, 4], but they are not yet widely used. Another weakness of the traditional graph model is that it is bound by its definitions, particularly of nodes and edges. There is some flexibility here (e.g., multigraphs, directed graphs), but the core of a graph always comes down to separating its components into nodes and edges; objects in the space and the relationships between them. This works in many cases, but there are situations in which it falls short.

For example, consider an insurance claim in a company's database. Is it a node or an edge? It has some node-like properties (many potential attributes, many points of connection, it is a noun and represents a physical or digital object), but it also has edge-like properties, the most obvious of which is that a claim is the result of an action being taken and connects the claimant to the company (or the insurance provider). Making a claim a type of node causes it to lose out on some of its connective power, but making it a type of edge makes it more ephemeral than it ought to be.

The best approximation with current research is likely to make it a hyperedge in a hypergraph [1], which would allow it to connect to multiple nodes while retaining its useful edge properties, but hypergraphs are not yet widely used in academia or industry. Just as the vast majority of data are in table format, the vast majority of graphing software and programming libraries use the standard model. As such, there is a gap to be filled; an intermediate step that provides or simulates advanced functionality as seen in hypergraphs, but that can be built within or on top of existing architecture.

2 Phantom Graphs

We introduce the concept of a Phantom Graph as a means of solving problems where an entity's status as node or edge is ambiguous, such as the insurance example described above. A Phantom Graph is a specialized type of graph that contains nodes and edges with additional properties, referred to as Hub Nodes, Bypass Edges, and Phantom Edges.

Hub Nodes are the solution for the claim problem mentioned above. They are a node, but their nature makes them a point of connection for other nodes. They can be treated as a normal node for all purposes, only revealing their additional properties when they are used as an endpoint for a Bypass Edge. Bypass Edges come

KDD '25, Toronto, ON

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

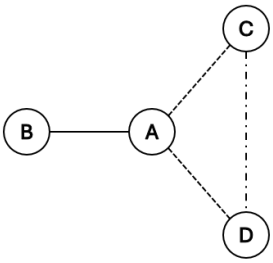

Figure 1: A Phantom Graph, containing a Hub Node (A), Bypass Edges (AC, AD), and a Phantom Edge (CD)

in pairs (or more); they signify when one of the edge's endpoints is being used more like a collection of edge properties than an endpoint. A Bypass Edge must be connected to a Hub Node at one end, and when two or more Bypass Edges are connected to the same Hub Node, a Phantom Edge is automatically created between the other endpoints.

The last part is the key step; Phantom Edges do not exist in the database initially, they are added dynamically when the graph is created as a consequence of creating Hub Nodes and Bypass Edges. Phantom Edges are also by default not displayed if the graph is visualized so as to reduce clutter (and because humans can infer the connections that Phantom Edges exist to inform algorithms about).

This is illustrated in Figure 1. Node A is created as a Hub Node, and Edges AC and AD are inputted as Bypass Edges. The graph then automatically creates the Phantom Edge CD, allowing algorithms which run on the graph to know that C and D are directly related.

3 Preliminary Results

3.1 Implementation

An alpha version of the Phantom Graph library has been implemented in Python 3.9, as an extension of the widely-used NetworkX graph library [3]. As an extension, a Phantom Graph possesses all the same functionality as an ordinary graph, including the ability to run the built-in algorithms coded into NetworkX.

Moreover, the library architecture is lightweight and generic enough that it should be reproducible as an extension to other graph libraries without needing to rebuild from scratch.

3.2 Comparison

Preliminary tests were run on an insurance fraud dataset available on Kaggle (https://www.kaggle.com/datasets/mastmustu/insurance-claims-fraud-data?resource=download).

As can be seen in Table 1, the addition of the Phantom Edges does not materially increase the time required to generate the graph.

Table 2 shows the time taken to run standard graph algorithms (built into NetworkX) on both a Normal and a Phantom Graph. The results are encouraging, as they demonstrate that using a Phantom Graph doesn't materially increase the time taken for an algorithm to run when the algorithm doesn't need its functionality, and it provides a substantial improvement to algorithms like finding the maximum clique.

Comparison	Standard	Phantom
Nodes	31,800	31,800
Edges	36,755	46,755
Generation	0.4734 s	0.4876 s

Table 1: Building a normal graph and a Phantom Graph on Insurance Fraud Data

Algorithm	Standard	Phantom
Connectivity	0.0323s	0.0193s
Max Cliques	13.9325s	7.7924s
Clustering	0.0045s	0.0026s
K-Components	0.1982s	0.1578s

Table 2: Running algorithms on a Normal Graph and a Phantom Graph on Insurance Fraud Data

All tests were run on a 13th Gen Intel(R) Core i7-13700HX Windows laptop with 24 GB RAM.

4 Future Work

Many specific cases and variations of graphs exist, and Phantom Graphs must be adapted to fit. Questions to explore in this area include:

(1) How does being a Phantom Graph impact runtimes of graph algorithms such as clique detection?
(2) How would a Phantom Graph function for a graph stored as an adjacency matrix?
(3) Can multiple "sets" of Bypass Edges be connected to a single Hub Node?
(4) Does the behaviour of Phantom Graphs change for Directed Graphs or Multigraphs?

Acknowledgments

I would like to acknowledge Dr. Carson Leung, my Master's and PhD supervisor, for his continued mentorship and support.

This work is partially supported by NSERC (Canada) and the University of Manitoba.

References

[1] Ronald Fagin. 1983. Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM* 30, 3 (jul 1983), 514–550. doi:10.1145/2402.322390
[2] Fayed FM Ghaleb, Azza A Taha, Maryam Hazman, Mahmoud Abd ElLatif, and Mona Abbass. 2020. RDF-BF-Hypergraph Representation for Relational Database. *International Journal of Mathematics & Computer Science* 15, 1 (2020).
[3] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). Pasadena, CA USA, 11 – 15.
[4] Yelda Unal and Halit Oguztuzun. 2018. Migration of data from relational database to graph database. In *ACM International Conference Proceeding Series (ICIST '18)*. ACM, 1–5.