What If: Causal Analysis with Graph Databases

Amedeo Pachera
Lyon1 University, CNRS Liris
Lyon, France
amedeo.pachera@univ-lyon1.fr

Mattia Palmiotto
Lyon1 University, CNRS Liris
Lyon, France
mattia.palmiotto@univ-lyon1.fr

Angela Bonifati
Lyon1 University, CNRS Liris & IUF
Lyon, France
angela.bonifati@univ-lyon1.fr

Andrea Mauri
Lyon1 University, CNRS Liris
Lyon, France
andrea.mauri@univ-lyon1.fr

ABSTRACT

Graphs are powerful abstractions for modeling relationships and enabling data science tasks. In causal inference, Directed Acyclic Graphs (DAGs) serve as a key formalism, but they are typically handcrafted by experts and rarely treated as first-class data artifacts in graph data management systems. This paper presents a novel vision to align causal analysis with property graphs—the foundation of modern graph databases—by rethinking graph models to incorporate hypernodes, structural equations, and causality-aware query semantics. By unifying graph databases with causal reasoning, our approach enables the declarative expression of DAG manipulation operations along with interventions and counterfactuals, combining expressiveness with computational efficiency. We validate this vision through a proof-of-concept implementation supporting scalable causal queries over DAGs, ultimately aiming to make graph databases causally aware and support data-driven, personalized decision-making across several scientific domains.

PVLDB Reference Format:
Amedeo Pachera, Mattia Palmiotto, Angela Bonifati, and Andrea Mauri. What If: Causal Analysis with Graph Databases. PVLDB, 18(11): 4009 - 4016, 2025.
doi:10.14778/3749646.3749671

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/pake97/What-If-Causal-Analysis-with-Graph-Databases.

1 INTRODUCTION

Causal analysis [44] focuses on identifying, representing, and quantifying cause–effect relationships. Its application is crucial in various domains [33], as it goes beyond correlation and association to extract causal relationships between variables. Causal directed acyclic graphs [22] are a cornerstone of causal analysis, providing a rigorous framework for representing and analyzing causal relationships. By modeling variables as nodes and causal connections as directed edges, causal DAGs facilitate the visualization of complex systems, the identification of confounders, mediators, and colliders, and the extraction of causal paths.

However, conducting causal analysis remains a significant challenge, as it involves a large amount of manual labor [12, 21, 29]. This process is not only time-consuming and costly but also prone to errors, limiting the applicability of causal analysis, particularly in complex domains with evolving or heterogeneous data.

We argue that causal analysis, to some extent, shares commonalities with data management. Defining a causal DAG is akin to graph data modeling, as both involve identifying relevant entities and relationships. Additionally, causal analysis can be viewed as performing queries on graph observational data and causal DAGs.

The data management community already explored how to integrate data management methodologies and causal analysis. Existing research has focused on relational databases and has explored how to facilitate causal analysis by treating it as relational queries [48, 49]. Further extensions have been considered with counterfactual queries [25, 50], using provenance to determine the effect of interventions and counterfactuals in SQL queries. Other works utilize causality to explain the causes of query results [34, 54]. Additionally, some studies have established a connection between these explanations and database repairs [14–16], reducing the problem of causal query explanation to a repair problem. Another work [55] has considered the problem of data completeness of causal DAGs while considering information coming from different tuples in a relational table. To the best of our knowledge, the idea of leveraging a graph database to carry out causal inference over interlinked data has not been explored before.

Graph data models and graph query languages are becoming increasingly widespread across various domains [47]. They serve as the underlying data models for numerous open-source and commercial database tools (e.g., Neo4j [3], Amazon Neptune [1], Oracle PGX [4], SAP Hana Graph [6], RedisGraph [5], Sparksee [7], Kuzu [2], etc.) and have recently been standardized with the introduction of graph query languages such as GQL and SQL/PGQ.

For this reason, we propose leveraging property graphs (PGs) [8, 17] as a foundational model for causal analysis. The expressive power and flexibility of PG allow the encoding of multi-valued nodes and edges, along with edge and node properties represented as key-value pairs, enabling a natural mapping of causal variables and relationships to nodes and edges. This representation can be further utilized to extract causal paths and align disparate data sources with greater consistency and efficiency. Another advantage

Proceedings of the VLDB Endowment, Vol. 18, No. 11 ISSN 2150-8097.
doi:10.14778/3749646.3749671

of property graphs is their compatibility with concrete graph query languages, such as openCypher [38], GQL, or SQL/PGQ [23]. These languages enable graph pattern matching operations, retrieving and outputting results in the form of tuples. Additionally, recent advancements in high-level abstractions, such as path-based algebraic foundations for graph query languages, enhance the compositionality of graph queries, allowing property graphs to be returned as results instead of plain relational tuples [11].

In our vision, we leverage the underlying property graph data models and declarative formalisms to allow the precise extraction of all causal paths, identification of bias-inducing relationships, and inference on the extracted graph. By using graph queries, we can automate the discovery of confounder, mediator, and collider paths, as well as explore the causal structure across different datasets.

Furthermore, using PGs allows us to embed the causal DAG directly within the data, whereas, in a relational model, it must reside outside the data model itself. This enables us to leverage the expressive power of graph data models and query languages to manipulate the observational data and the causal DAG seamlessly.

However, the property graph model and query languages are currently not expressive enough for causal analysis, particularly when entire paths and subgraphs need to be inspected as a whole. For example, as we show in Section 4, simple analyses, such as identifying mediators or confounders, can be performed using existing Cypher/GQL constructs. However, more complex causal tasks that require navigating between the causal DAG and the observational data necessitate additional constructs that are not present in current languages—for instance, incorporating path algebra [11].

In summary, the research vision that we investigate with this line of work centers around the following fundamental question: *How can we empower graph databases, property graph models and queries with causal analysis capabilities?*

To address the above question, in this paper we make the following contributions:

- We propose and formalize a causal property graph model augmented with hypernodes representing possibly related causes and with meta-properties encoding conditional and interventional probabilities along with structural equations.
- We study the extensions needed in current graph query languages to support *path-based semantics* [11] and *do-calculus* constructs. The former are meant to extract from property graph observational data different variants of causal paths, namely confounding paths, bias paths, mediator paths and collider paths.
- In a proof-of-concept implementation, we show the performances of causal path queries on causal DAGs of various sizes, showing that queries are executed efficiently on the causal DAGs and guarantee constant time access to the property graph data.

The above vision will **make graph databases aware of causal knowledge** and pave the way to data-driven personalized decision-making in several scientific fields. By embedding causal constructs into the graph data model and query language, graph databases gain expressive power to support interventions, counterfactuals, and structural reasoning that are fundamental for causal analysis. In addition, graph databases can benefit from explainable query results, by supporting queries that return not only their results but also the corresponding causes for these results.

This extension will make graph databases more interoperable with machine learning models, as they could be employed in various stages of the ML pipeline, querying causal effects post hoc, or integrating causal graphs into model interpretation workflows.

The paper is organized as follows: Section 2 covers causal model preliminaries. Section 3 introduces our property graph extension, including DAG extraction and causal views. Section 4 details causal analysis in GQL/Cypher with necessary language extensions. Section 6 presents a proof-of-concept implementation. Section 7 concludes with limitations and future directions.

2 CAUSAL MODELS

One of the most established theory of causality is represented by structural causal models (SCMs). SCMs consist of a causal graph and structural equations. Formally, a causal graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a directed graph where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set. Each node in $\mathcal{V}$ represents a random variable while an edge $x \longrightarrow y$ represents a causal effect between two variables $x, y \in \mathcal{V}$. Given an edge $x \longrightarrow y$ we call x the *exposure* and y the outcome. The node set $\mathcal{V}$ contains also all the observed and unobserved variables. In particular, we consider causal directed acyclic graphs [22].

Figure 2(b) shows an example of a DAG, including the key elements of causal analysis: i) `(INCOME LEVEL)-->(STRESS)--> (SMOKING)` is a causal path representing an indirect effect of `INCOME LEVEL` on `SMOKING` through `STRESS` as **mediator** variable. ii) The path `(SMOKING) <--(AGE)-->(COPD)` is a confounding path with `AGE` as a **confounder** (e.g. having effect on both `SMOKING` and `COPD`). iii) The path `(STRESS)-->(COPD) <--(SMOKING)` is a collider path with `COPD` as **collider**.

The conditional independence is guaranteed in causal graphs through *d-separation* [43]. Two variables are d-separated if there is a *blocked* node in every path between them where blocking means conditioning a variable on a suitable set of other variables. In the chain `(INCOME LEVEL)-->(STRESS) -->(SMOKING)`, blocking the variable `STRESS(P(SMOKING|STRESS))` d-separates the variables `INCOME LEVEL` and `SMOKING`. Intuitively, once the stress is known, the income level has no more influence on the smoking habit. The same applies to the confounding path `(SMOKING)<--(AGE)-->(COPD)`: conditioning on `AGE` allows us to study the causal relationships between the `COPD` and the smoking habit excluding the confounding bias. On the contrary, if two variables are connected through a collider path, they are already d-separated (and thus causally independent). Conditioning the collider though, opens the path between the two variables. Conditioning on the node `COPD` in the path `(STRESS)-->(COPD)<--(SMOKING)` allows us to study the association between the stress and the smoking habit of an individual induced by the selection bias.

The second component of a Structural Causal Model consists of structural equations. Structural equations are non-parametric equations that quantify the causal effects between variables. For instance, the structural equations for the SCM in Figure 1(a) are:

$$\begin{aligned}
&Age = f_{Age}(\epsilon_{age})\\
&Income\ Level = f_{Income\ Level}(Age, \epsilon_{Income\ Level})\\
&Smoking = f_{Smoking}(Age, Income\ Level, \epsilon_{Smoking})
\end{aligned}$$

where ϵ_x denotes the noise of the observed variable x. The functions $f_x()$ quantify the causal relationships between the LHS variable and the RHS ones. Structural equations provide a quantitative way to represent *intervention* on a variable in a DAG. The *do-calculus* [44] utilizes the $do(x^{'})$ operator, which denotes the intervention of setting the value of a variable x to $x^{'}$. In general, intervening on a variable x by setting its value to $x^{'}$ affects the probability distribution of all variables y for which x is a direct or an indirect cause. The distribution $P(y|do(x^{'}))$ is called *interventional distribution* [43].

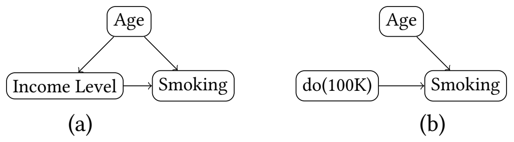

Figure 1: SCM without (a) and with (b) intervention.

Figure 1(b) shows an example of intervention, where $do(100K)$ sets the value for the variable *Income Level* to *100K*. The structural equation for *Income Level* becomes $Income\ Level = 100K$ and the interventional distribution of *Smoking* is $P(Smoking|do(100K))$. It is important to note that the interventional distribution $P(y|do(x))$ and the conditional probability distribution $P(y|x)$ are not the same. This difference introduces a *confounding bias*. In practice, verifying the absence of the bias translates into verifying that $P(y|do(x)) = P(y|x)$. In Figure 1(a), with *Age* being a confounder, the probability distribution $P(Smoking|Income\ Level)$ results from combining the causal effect $P(Smoking|do(Income\ Level))$ and the statistical association of the confounding path. To obtain an unbiased estimate, one needs to remove the confounding bias. This can be done via *causal identification*, that is, blocking the confounding path. This operation requires to estimate the causal influence of the cause variable x on y within subpopulations in which the confounding variables do not vary, ensuring that they do not distort the causal relationship (i.e. adjustment).

3 GRAPH MODEL

Our vision leverages the expressiveness and flexibility of property graphs to streamline the extraction of SCMs, facilitate their maintenance, and enhance their interpretability. By embedding causal semantics directly into the graph structure, property graphs enable causal analysis through queries using (or extending) existing graph data management methods, bridging the gap between raw data and actionable causal insights. Ideally, this approach makes the causal analysis process less tedious, as many of the tasks now requiring ad-hoc scripting, can be executed in a declarative fashion. Property graphs are multi-label directed graphs characterized by node and edge properties (i.e. metadata stored as key-value pairs) [17]. Unlike conventional DAGs, property graphs naturally capture the rich context of causal systems, accommodating multi-faceted dependencies, metadata about variables and relationships, and evolving structures over time. These attributes make PGs uniquely suited for extracting and maintaining causal DAGs from complex and evolving data.

Figure 2(a) shows an example of a property graph. It models a person named Ali who smokes and drinks some alcohol and has two diseases (`COPD` and `Stress`), and a person named Kate suffering from the same `COPD` condition, but having also a job as an engineer.

To perform causal analysis, we need to extract causal variables from the PG and the relationships between them. Moreover, in order to apply interventions and perform causal analysis, we need to maintain a mapping between the PG instances and the causal variables in the causal DAG. It is seldom the case that one variable in a causal DAG is mapped to a single vertex in a property graph. In several cases, this mapping is not only one-to-many but also compositional. For this reason, we introduce a novel extension of the PG data model to cover hypergraphs. In particular, a hypervertex is a subgraph containing nodes and edges in the original property graph but can be considered a new node and can be linked to other existing vertices and new hypervertices. Increasingly expressive property graph models have already been discussed in the past [17, 46], motivating their need in several application domains.

Causal Directed Acyclic Hypergraph (cDAH): Given a property graph $G = (V, E)$, a cDAH is a structure $(H, F, X, S, P, I, \gamma, \lambda, \eta, \mu, \nu)$, where H is a finite set of hypervertices, F is a set of edges disjoint from E, X is a set of causal variables, S is a set of structural equations, and P and I are sets of conditional and interventional probability distributions, respectively. The function $\gamma : H \longrightarrow \mathscr{P}(V) \times \mathscr{P}(E)$ assigns to each hypervertex a subset of nodes and edges from G. An injective function $\lambda : H \longrightarrow X$ maps each hypervertex to a causal variable, and $\eta : S \longrightarrow H$ assigns structural equations to hypervertices. The bijections $\mu : F \longrightarrow P$ and $\nu : F \longrightarrow I$ assign conditional and interventional distributions to edges in F.

Representing causal DAGs as property graphs offers several key advantages. First, it enables the causal DAG and the associated observational data to coexist within a unified data artifact, allowing them to be queried and analyzed jointly. Second, this representation makes it possible to leverage well-established techniques from graph data management—such as PG-Schema [9], PG-keys [10], and graph views [18, 27]—as well as powerful declarative query languages like Cypher, GQL, or SQL/PGQ for expressive and efficient analysis. We argue that causal analysis is fundamentally navigational, as causal relationships are naturally encoded as edges in a graph. The property graph model is therefore particularly well-suited since graph query languages are designed precisely to support such navigational reasoning. From the above definition, it is clear that hypernodes need to be extracted from the underlying property graphs. A solution would be to adopt tuple generating dependencies [27] or graph transformations [18]. An example of graph transformation merging *Smoking* and *COPD* in a causal DAG is the following Cypher query, where a new relationship is generated when a condition is met (the matched pattern):

```
MATCH (p:Person)-[:HAS_HABIT]->(h:Habit),
(p)-[:HAS_CONDITION]->(c:Condition)
WHERE c.name = "COPD" AND h.type="Cigarettes" WITH h,c
MERGE (h)-[:BELONGS]->(x:SMOKING)-->(y:COPD)<-[:BELONGS]-(c)
```

However, this query produces a node (variable) for each matched path. In Figure 2(b), for example, a variable (`SMOKING`) is extracted twice from the property graph instance. To deal with duplicates, we should merge the generated nodes that have the same label. This is possible in concrete graph query languages but requires multiple complex queries. The first step of our vision consists of extending the GQL syntax with a new operator `EXTRACT`, that allows us to easily express the causal variable extraction by abstracting out the details of hypernodes and graph transformations:

```
MATCH (p:Person)-[:HAS_HABIT]->(h:Habit),
(p)-[:HAS_CONDITION]->(c:Condition)
WHERE c.name = "COPD" AND h.type="Cigarettes"
EXTRACT (x:SMOKING)-->(y:COPD)
```

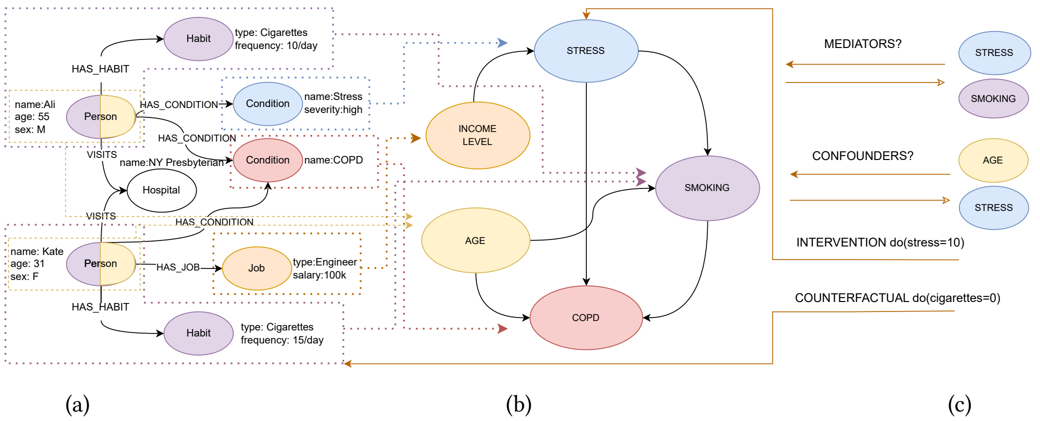

Figure 2: Example of a Causal Directed Acyclic Hypergraph model including the observational data (a), their mapping to the causal variables in the causal DAG (b), and the results of different causal queries (c).

After extracting the variables from the property graph instance, the causal DAG is not yet complete as it needs to be enriched with the probability distributions to the edges as properties.

Figure 2(b) shows an example of causal DAG extracted from a property graph (the colors indicate the mapping between the observational data in the property graph and the causal variables). We can see how different paths in the property graph are mapped to the causal DAG, specifically the query listed above created the path `(SMOKING)-->(COPD)` from the path including `Person` and `Habit`.

4 CAUSAL PG QUERIES

Causal investigation consists of a three-level hierarchy of analysis. First, association, which serves as the starting point, identifies relationships between variables. While this alone cannot answer causal queries, it might guide hypothesis generation. It consists of posing questions like *"What if I see..?"* or *"Is the value of X relevant for estimating the value of Y?"*. Second, intervention, which is used for interventional queries, addresses questions like *"What happens if we do X? Why?"*. Finally, counterfactual, which is essential for providing insights into hypothetical alternate realities, handles questions like *"What if I had done...? Why?"*. These investigations need to be paired with the structural analysis of causal models, which plays a crucial role in the broader process of causal discovery and causal inference. During the causal discovery phase, identifying confounder paths (and blocking them) prevents a biased interventional analysis because confounders can create spurious associations or hide true causal effects. The same occurs with colliders path because they can create spurious associations if incorrectly conditioned on.

In this section, we propose our vision about **conducting causal analysis using a declarative approach relying on graph queries and their path semantics**, highlighting the advantages of relying on property graph models. In particular, we identified a list of queries that correspond to a specific type of analysis and we show how to write them in a standard graph query language. In our examples, we use **bold** for the part of the queries that refer to hypernodes (the causal variables in Figure 2(b)) and the green color to indicate our proposed extensions to the query language. All the following graph queries are expressed in GQL [31], but they can be similarly encoded in Cypher or SQL/PGQ [30].

4.1 Causal Path Queries

Causal mediation analysis examines how an exposure influences an outcome through intermediary variables, known as mediators. This approach decomposes the total effect of an exposure into a direct effect, the impact of the exposure on the outcome not mediated by the intermediary, and an indirect effect, the portion of the effect that operates through the mediator [42]. Correct mediation analysis has to be conducted after checking for existing confounding paths between the investigated variables.

Identifying the mediator nodes can be done using graph queries supported by the current standards. For instance, identifying the mediators between two variables X and Y checking the absence of confounding paths, can be achieved via this simple query:

```
MATCH (a:X)-->(m)-->(b:Y)
WHERE NOT EXISTS { MATCH  (a)<--(c)-->(b) }
AND NOT EXISTS { MATCH (a)<--(c)-->(m) }
AND NOT EXISTS { MATCH (m)<--(c)-->(b) }
RETURN a as Exposure, m as Mediator, b as Outcome
```

Current graph query languages (GQL and SQL/PGQ 1.0) cannot retrieve all relevant paths needed for statistical adjustment. To address this, path operators must be added, as in [11]. This becomes especially important for multiple mediators, e.g., matching `(a:X)-->(m)-->(b:Y)` where $*$ denotes the Kleene star for paths with zero or more intermediates [36].

The advantages of relying on PGs come from the possibility of accessing directly the observational data for further analysis since the DAG and the instances belong to the same data model. For example, to know which graph instance corresponds to the mediator between the variable `INCOME LEVEL` and `COPD` in the case of `Kate` in the graph in Figure 2, we can use the following query:

```
MATCH (p)--(j:Job)-[:BELONGS]->(a:INCOME LEVEL)-->(m)-->(c:COPD)
WHERE p.name="Kate" WITH m
MATCH ALL instance=(m)<-[:BELONGS]-(n1)
-[]-{*}(n2)-[:BELONGS]->(m)
RETURN instance
```

Where MATCH ALL is the path operator that lists all the path that matches the given pattern. This query first reaches the causal variable INCOME LEVEL from the node Person. Then it identifies the mediator via pattern matching. Finally, it leverages the path algebra to return the subgraph that corresponds to the mediator (e.g. the path expressing that Kate is smoking 15 cigarettes a day).

The same type of query applies to retrieve collider paths. Colliders block the path between two variables. By conditioning on the collider, we can open the path and study the association between the variables induced by the selection bias. Retrieving the colliders can be easily done by matching the path (a:X)-->(c)<--(b:Y) where c is the collider. Regular path queries [17] allow us to list the descendants of the collider, since conditioning on one of them also opens the path [43]. Moreover, conditioning on a variable means identifying it with a value. Similarly to what we discussed with confounder path queries, we can leverage path algebra to directly access the instances of a collider path and select accordingly the right value to identify the collider random variable.

4.2 Interventional Query

Interventional queries focus on predicting the effects of specific interventions or manipulations by involving do-calculus [41] to simulate interventions. An example consists of investigating what will be the risk of contracting the COPD from smoking 10 cigarettes a day. In our model, this can be answered by traversing the path between the variables (SMOKING and COPD in Figure 2(b)) to obtain the structural equations from the node properties:

```
MATCH SHORTEST 1 (a:SMOKING)-->{*}(m)-->(b:COPD)
RETURN a.s_eq as s_s, m.s_eq as s_m, b.s_eq as s_l
```

where SHORTEST 1 is the path algebra operator that matches the shortest path between two nodes [11]. Then, we need to apply the do-operator $do(cigarettes = 10)$ to the variable SMOKING and to evaluate the impact of this intervention on the variable COPD. To obtain the values for the other variables in the equations we can leverage the path operators to access the data instances in the property graph as described in Section 4.1. Linear equations can be solved in GQL since they do not require Turing completeness. However, GQL does not have built-in matrix operations, so solving systems with multiple variables is inefficient or impractical. Current solutions consist of relying on procedural languages (such as custom APOC procedure [35]). However, this obliges to go beyond the declarativeness of the query language. We propose to extend the GQL standard with an operator DO-CALCULUS([values],[equations]) that evaluates the structural equations for an identified value.

Table 1: Comparison of tools on causal inference tasks.

Task	PyWhy	CausalAI	Graph DBMS	WhatIf
Causal Discovery	✓	✓	✗	✓
Causal DAG Mining	✗	✗	✗	✓
Causal Path Finding	~	~	✓	✓
Causal DAG Manipulation	✗	✗	✗	✓
Causal DAG Maintenance	✗	✗	✗	✓
Causal DAG Transportability	✗	✗	✗	✓
Intervention	✓	✓	~	✓
Counterfactual	✓	✓	~	✓

Table 2: Runtimes (s) by query type and DAG size; bracketed values show runtimes without the causal model, and best runtimes are in bold.

#Nodes (#C.Var.)	Extract	Mediators	Confound.	Colliders	Interv.	Counterf.
5k (5)	0.780	**0.003** (0.648)	**0.004** (0.656)	**0.003** (2.218)	0.194 (0.113)	**0.016** (0.017)
25k (25)	1.236	**0.005** (3.660)	**0.004** (8.246)	**0.012** (24.003)	0.801 (**0.104**)	0.055 (**0.020**)
50k (50)	4.074	**0.009** (25.855)	**0.013** (14.095)	**0.004** (39.596)	10.01 (**2.954**)	0.926 (**0.289**)
50k (5)	1.032	**0.010** (2.917)	**0.008** (4.891)	**0.004** (11.358)	**0.109** (0.757)	**0.023** (0.048)
250k (25)	1.277	**0.009** (37.431)	**0.009** (82.466)	**0.012** (230.949)	**0.838** (1.161)	0.056 (**0.044**)
500k (50)	3.974	**0.011** (367.681)	**0.011** (771.086)	**0.00**4 (2274.338)	9.951 (**3.262**)	0.935 (**0.293**)
500k (5)	1.028	**0.008** (20.081)	**0.007** (33.072)	**0.004** (64.904)	**0.106** (4.233)	**0.020** (0.472)
2500k (25)	1.207	**0.006** (93.703)	**0.006** (537.99)	**0.003** (961.93)	0.834 (**0.314**)	0.053 (**0.042**)
5000k (50)	3.989	**0.007** (162.502)	**0.005** (1242.618)	**0.004** (1842.14)	9.831 (**0.653**)	0.892 (**0.104**)

4.3 Counterfactual Query

Counterfactual queries focus on exploring "what if" scenarios, both on individual-level, for example asking *"If patient A had not smoked, would they still have COPD?"*, or on a population-level, for example asking *"If smoking rates had been reduced by 20%, how much would lung-related mortality have decreased?"* [52].

Answering counterfactual queries implies forcing a variable to take a specific value regardless of its natural causes. Interventional queries involve directly manipulating a variable ($do(X = x)$) to observe its effect on an outcome, while counterfactual queries extend this by conditioning on observed data to hypothesize alternate scenarios. The process of solving counterfactual queries simulates the intervention within a causal model, integrated with additional steps to account for observed evidence. This translates into computing the noise for the specific instance using the observed evidence.

For example, if we want to find out what the COPD severity of *Ali* would have been if he had not smoked, we can proceed by computing the noise of the variable COPD from the observed data. This means that knowing that he smokes 10 cigarettes a day and the severity level of COPD is moderate, we can use the structural equations to derive the noise in his case. Then we apply the do-operator $do(cigarettes = 0)$ to obtain the result. With our model, this translates into:

```
MATCH (h:Habit)<-[HAS_HABIT]-(p:Person)-[HAS_CONDITION]->(c:Condition)
WHERE p.name="Ali" AND h.type="Cigarettes" AND c.name="COPD"
RETURN h.frequency as f, c.severity as s
WITH f,s
MATCH SHORTEST 1  path=(a:SMOKING)-->(b:COPD)
RETURN DO-CALCULUS([SMOKING=f,COPD=s],[b.s_eq]) as noise
WITH noise
MATCH SHORTEST 1  path=(a:SMOKING)-->(b:COPD)
RETURN DO-CALCULUS([SMOKING=0,ε_COPD=noise],[b.s_eq])
```

4.4 Causal DAG Maintenance

Maintaining the causal DAG according to the changes in the underlying data is a very tedious process that entails manually rerunning all causal analysis on the new and updated data, since existing methods disregard the case where the underlying data change [44, 51, 53]. This is important because it avoids recomputing the causal DAG from scratch each time the underlying data change. Our model allows us to use established data management techniques such as incremental view maintenance [18, 26–28, 56] and reactive graph data management [19, 20]. We envision utilizing triggers [20] to maintain the causal DAG by reacting to changes on edge patterns corresponding to causal variables, and running queries to compute the new probability and interventional distributions Another option is to adapt works on view maintenance [18, 27, 28, 56]. In this case, the causal DAG will be implemented as one or more graph views and the incremental maintenance can be done by either extending recent proposals on graph views and transformations [18, 28], or by adapting the well-known DRed algorithm [26] for the graph case.

5 RELATED APPROACHES

In this section, we discuss the features of our approach compared with causal analysis frameworks - such as PyWhy [39] and Salesforce CausalAI [13] - and graph DBMSs. We distinguish tasks related to causal path finding and causal DAG management tasks, as well as causal graph discovery, and interventional and counterfactual analysis as foundations of causal analysis [43].

Causal analysis frameworks are empowered with imperative programming interfaces, requiring familiarity with both causal analysis and the APIs of the respective libraries. Whereas they support causality-oriented operations, such as causal discovery leading to learn a causal DAG from the data, along with API methods for interventions and counterfactuals, they are not able to natively support graph-oriented operations, such as causal pathfinding, leading to retrieve all collider, mediator, and confounder paths between two variables (as shown by the first two columns of Table 1).

In contrast, our framework (last column) adopts a different perspective grounded in data management, allowing DAG manipulation operations based on declarative update languages, efficient causal path traversals through graph queries, along with DAG mining utilizing the topology and semantics of the underlying observational property graph data. A causality-driven approach in a graph database can also facilitate one of the hardest tasks in the theory of causality, namely DAG transportability (i.e., applying causal DAGs, from one context to another [45]). It elevates path-based management to a first-class operation where users can express complex causal queries, directly interacting with causal paths and their properties to enumerate, filter, transform, and compare causal paths using a declarative approach without external libraries, as causal structures are encoded in the graph itself.

The third column of Table 1 finally shows the contrast between what a current graph DBMS supports and what our approach supports. As also shown in the experiments, a graph DBMS does not handle interventions and counterfactuals natively but with the support of APOC procedures in an external library. We envision extensions of graph query languages with DSL operators.

Our framework is not meant to replace existing tools but to complement them. It can, for example, retrieve valid adjustment sets or causal paths to be used by PyWhy for estimation, with results then annotated in the graph for further querying and analysis.

6 PROTOTYPE AND EXPERIMENTAL STUDY

To assess the feasibility of our vision, we implemented a proof-of-concept causal graph database model and conducted preliminary experiments to evaluate the performance of our proposed queries. Since hypernodes aren't currently supported in any graph DBMS, in our prototype we emulate their behavior by creating new nodes and linking them to their respective instances. Using `DoWhy` [24], we generated synthetic datasets based on causal DAGs that accurately capture realistic dependencies between variables. We constructed three DAGs with 5, 25, and 50 causal variables, ensuring a balanced proportion of mediators, colliders, and confounders. Additionally, we varied the number of instances per causal variable (1000, 10k, and 100k), representing each instance as a node in the graph. We ran experiments on Neo4j 5.12 [37], using a MacBook M3 Pro Max (128GB RAM, 32GB allocated to Neo4j). Code is available at [40].

We tested different types of queries to evaluate the performance. The *Extract Queries* extract causal variables as hypernodes, as described in Section 3. The *Mediator, Confounder, and Collider Queries* retrieve all mediators, confounders, and colliders from the graph. The *Intervention Queries* were implemented by setting the property of each instance to a specified input value and using Cypher aggregators and mathematical operators to solve linear equations. The *Counterfactual Queries* followed the same implementation but were limited to a single instance at a time.

Table 2 reports the performances of various queries with our solution integrating property graphs and causal models in the same artifact with hypernodes and comparing it with the case when they are kept separate. We can observe that for the majority of the queries, the comparative solution is significantly less efficient than our integrated model. Specifically, for interventional and counterfactual queries our solution is slightly less efficient as the property graphs grow in size. This can be explained by the fact that, since the hypernodes are not supported by current graph DBMS, there is no efficient way to query both the DAG and the data at the same time while guaranteeing scalability. The results show that runtimes depend on the size of the causal DAGs while the size of observational property graph data has a negligible impact on the performances.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we discussed a new research direction leading to integrate causal analysis into property graphs. We also showed that causal analysis with property graphs is not only feasible but also scales well with the size of property graphs and causal variables.

Our vision also highlights a number of limitations. The first limitation is the **extraction of the causal DAGs**. Traditionally, they are manually labeled and constructed by domain experts or learned using causal discovery algorithms [51, 53]; however, we argue that by leveraging the property graph model, we can employ methods that use both topological structure of the graph and the semantics of data values.

Second, the **causal DAG maintenance and transportability** are important yet challenging tasks. For both, we envision extending works on reactive data management and view maintenance, and developing specific Global-as-View methods, mapping local causal graphs into a global representation [32].

Third, current graph DBMSs do not support causal queries or the data structure needed for the model. The integration of our model should be carefully designed to avoid negative impacts on query performance. Future work should focus on **extending Cypher** and potentially **designing a DSL specific for causal graph analytics**, allowing also non-expert users to express causal queries declaratively, including constructs for identifying causal paths, defining interventions, and performing counterfactual reasoning.

REFERENCES

[1] [n.d.]. Amazon Neptune. https://aws.amazon.com/neptune/ Accessed: 2024-12-04.

[2] [n.d.]. Kuzu Graph Database. https://kuzudb.com/ Accessed: 2024-12-04.

[3] [n.d.]. Neo4j Graph Database Platform. https://neo4j.com/ Accessed: 2024-12-04.

[4] [n.d.]. Oracle PGX: Parallel Graph AnalytiX. https://www.oracle.com/database/technologies/graph-analytics/ Accessed: 2024-12-04.

[5] [n.d.]. RedisGraph Module. https://redis.io/docs/stack/graph/ Accessed: 2024-12-04.

[6] [n.d.]. SAP HANA Graph. https://help.sap.com/viewer/product/SAP_HANA_GRAPH/ Accessed: 2024-12-04.

[7] [n.d.]. Sparksee Graph Database. https://sparsity-technologies.com/sparksee/ Accessed: 2024-12-04.

[8] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. 2017. Foundations of Modern Query Languages for Graph Databases. *ACM Comput. Surv.* 50, 5 (2017), 68:1–68:40. https://doi.org/10.1145/3104031

[9] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savkovic, Michael Schmidt, Juan Sequeda, Slawek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, and Dusan Zivkovic. 2023. PG-Schema: Schemas for Property Graphs. *Proc. ACM Manag. Data* 1, 2 (2023), 198:1–198:25. https://doi.org/10.1145/3589778

[10] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Keith W. Hare, Jan Hidders, Victor E. Lee, Bei Li, Leonid Libkin, Wim Martens, Filip Murlak, Josh Perryman, Ognjen Savkovic, Michael Schmidt, Juan F. Sequeda, Slawek Staworko, and Dominik Tomaszuk. 2021. PG-Keys: Keys for Property Graphs. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*. 2423–2436. https://doi.org/10.1145/3448016.3457561

[11] Renzo Angles, Angela Bonifati, Roberto García, and Domagoj Vrgoč. 2025. Path-based Algebraic Foundations of Graph Query Languages. In *Proceedings of the 28th International Conference on Extendng Database Technology (to appear)*.

[12] Anonymous. 2024. Probabilistic Graphical Models and Causal Inference. https://arxiv.org/abs/2405.08793 arXiv preprint arXiv:2405.08793.

[13] Devansh Arpit, Matthew Fernandez, Itai Feigenbaum, Weiran Yao, Chenghao Liu, Wenzhuo Yang, Paul Josel, Shelby Heinecke, Eric Hu, Huan Wang, Stephen Hoi, Caiming Xiong, Kun Zhang, and Juan Carlos Niebles. 2023. Salesforce CausalAI Library: A Fast and Scalable Framework for Causal Analysis of Time Series and Tabular Data. *arXiv preprint arXiv:2301.10859* (2023). arXiv:2301.10859 [cs.LG]

[14] Leopoldo Bertossi. 2021. Specifying and computing causes for query answers in databases via database repairs and repair-programs. *Knowl. Inf. Syst.* 63, 1 (Jan. 2021), 199–231. https://doi.org/10.1007/s10115-020-01516-6

[15] Leopoldo E. Bertossi and Babak Salimi. 2017. Causes for query answers from databases: Datalog abduction, view-updates, and integrity constraints. *Int. J. Approx. Reason.* 90 (2017), 226–252. https://doi.org/10.1016/J.IJAR.2017.07.010

[16] Leopoldo E. Bertossi and Babak Salimi. 2017. From Causes for Database Queries to Repairs and Model-Based Diagnosis and Back. *Theory Comput. Syst.* 61, 1 (2017), 191–232. https://doi.org/10.1007/S00224-016-9718-9

[17] Angela Bonifati, George H. L. Fletcher, Hannes Voigt, and Nikolay Yakovets. 2018. *Querying Graphs.* Morgan & Claypool Publishers. https://doi.org/10.2200/S00873ED1V01Y201808DTM051

[18] Angela Bonifati, Filip Murlak, and Yann Ramusat. 2024. Transforming Property Graphs. *Proc. VLDB Endow.* 17, 11 (2024), 2906–2918. https://www.vldb.org/pvldb/vol17/p2906-ramusat.pdf

[19] Stefano Ceri, Anna Bernasconi, and Alessia Gagliardi. 2024. Reactive Knowledge Management. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 5574–5582. https://doi.org/10.1109/ICDE60146.2024.00445

[20] Stefano Ceri, Anna Bernasconi, Alessia Gagliardi, Davide Martinenghi, Luigi Bellomarini, and Davide Magnanimi. 2024. PG-Triggers: Triggers for Property Graphs. In *Companion of the 2024 International Conference on Management of Data*. 373–385.

[21] G. Chi, M. E. Roberts, B. M. Stewart, et al. 2022. How to make causal inferences using texts. *Science Advances* 8, 44 (2022), eabg2652. https://doi.org/10.1126/sciadv.abg2652

[22] Rina Dechter and Judea Pearl. 1991. Directed Constraint Networks: A Relational Framework for Causal Modeling. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991*. 1164–1170.

[23] Alin Deutsch, Nadime Francis, Alastair Green, Keith Hare, Bei Li, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Wim Martens, Jan Michels, Filip Murlak, Stefan Plantikow, Petra Selmer, Oskar van Rest, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, and Fred Zemke. 2022. Graph Pattern Matching in GQL and SQL/PGQ. In *SIGMOD*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 2246–2258. https://doi.org/10.1145/3514221.3526057

[24] DoWhy. 2025. DoWhy v0.12 - Causal Inference Library. https://www.pywhy.org/dowhy/v0.12/ Accessed: February 27, 2025.

[25] Sainyam Galhotra, Amir Gilad, Sudeepa Roy, and Babak Salimi. 2022. HypeR: Hypothetical Reasoning With What-If and How-To Queries Using a Probabilistic Causal Approach. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1598–1611. https://doi.org/10.1145/3514221.3526149

[26] Ashish Gupta, Inderpal Singh Mumick, and V. S. Subrahmanian. 1993. Maintaining views incrementally. *SIGMOD Rec.* 22, 2 (June 1993), 157–166. https://doi.org/10.1145/170036.170066

[27] Soonbo Han and Zachary G. Ives. 2024. Implementation Strategies for Views over Property Graphs. *Proc. ACM Manag. Data* 2, 3, Article 146 (May 2024), 26 pages. https://doi.org/10.1145/3654949

[28] Soonbo Han and Zachary G. Ives. 2025. Implementing Views for Property Graphs. *SIGMOD Record* 54, 1 (2025).

[29] Guido W. Imbens and Donald B. Rubin. 2015. *Causal Inference for Statistics, Social, and Biomedical Sciences.* Cambridge University Press.

[30] ISO. 2023. ISO/IEC 9075-16:2023 Information technology — Database languages SQL - Part 16: Property Graph Queries (SQL/PGQ). https://www.iso.org/standard/79473.html.

[31] ISO. 2024. ISO/IEC 39075:2024 Information technology — Database languages — GQL. https://www.iso.org/standard/76120.html.

[32] Maurizio Lenzerini. 2002. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 233–246.

[33] Roseanne McNamee. 2003. Confounding and confounders. *Occupational and environmental medicine* 60, 3 (2003), 227–234.

[34] Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu. 2010. The Complexity of Causality and Responsibility for Query Answers and non-Answers. *Proc. VLDB Endow.* 4, 1 (2010), 34–45. https://doi.org/10.14778/1880172.1880176

[35] Neo4j. 2025. APOC - Awesome Procedures On Cypher. https://neo4j.com/labs/apoc/. Last accessed: 2025-07-08.

[36] Neo4j. 2025. Cypher Manual: Reference Pattern Usage. https://neo4j.com/docs/cypher-manual/current/patterns/reference/. Last accessed: 2025-07-08.

[37] Neo4j. 2025. Neo4j Download. https://neo4j.com/download Accessed: February 27, 2025.

[38] openCypher Project. 2025. openCypher Implementation Proposal (CIP). https://github.com/opencypher/openCypher/tree/main/cip. Last accessed: 2025-07-10.

[39] PyWhy organization. 2025. PyWhy. https://www.pywhy.org. Accessed: 2025-05-03.

[40] Amedeo Pachera, Mattia Palmiotto, Angela Bonifati, and Andrea Mauri. 2025. *Code repository.* Retrieved March 1, 2025 from https://github.com/pake97/What-If-Causal-Analysis-with-Graph-Databases

[41] Judea Pearl. 1995. Causal diagrams for empirical research. *Biometrika* 82, 4 (1995), 669–688.

[42] Judea Pearl. 2001. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence* (Seattle, Washington) *(UAI'01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 411–420.

[43] Judea Pearl. 2009. Causal inference in statistics: An overview. (2009).

[44] J. Pearl. 2009. *Causality.* Cambridge University Press. https://books.google.fr/books?id=f4nuexsNVZIC

[45] Judea Pearl and Elias Bareinboim. 2011. Transportability of causal and statistical relations: A formal approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 25. 247–254.

[46] Sepehr Sadoughi, Nikolay Yakovets, and George Fletcher. 2025. Breaking Down the Data-metadata Barrier for Effective Property Graph Data Management. In *Proceedings 28th International Conference on Extending Database Technology, EDBT 2025, Barcelona, Spain, March 25-28, 2025*. 978–984. https://doi.org/10.48786/EDBT.2025.80

[47] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid G. Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. 2021. The future is big graphs: a community view on graph processing systems. *Commun. ACM* 64, 9 (2021), 62–71. https://doi.org/10.1145/3434642

[48] Babak Salimi, Corey Cole, Dan R. K. Ports, and Dan Suciu. 2017. ZaliQL: Causal Inference from Observational Data at Scale. *Proc. VLDB Endow.* 10, 12 (2017), 1957–1960. https://doi.org/10.14778/3137765.3137818

[49] Babak Salimi, Harsh Parikh, Moe Kayali, Lise Getoor, Sudeepa Roy, and Dan Suciu. 2020. Causal Relational Learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 241–256. https://doi.org/10.1145/3318464.3389759

[50] Fangzhu Shen, Kayvon Heravi, Oscar Gomez, Sainyam Galhotra, Amir Gilad, Sudeepa Roy, and Babak Salimi. 2023. Causal What-If and How-To Analysis Using HypeR. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 3663–3666. https://doi.org/10.1109/ICDE55515.2023.00293

[51] Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, Antti Kerminen, and Michael Jordan. 2006. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research* 7, 10 (2006).

[52] Ilya Shpitser and Judea Pearl. 2007. What counterfactuals can be tested. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence* (Vancouver, BC, Canada) *(UAI'07)*. AUAI Press, Arlington, Virginia, USA, 352–359.

[53] Peter Spirtes, Clark Glymour, and Richard Scheines. 2001. *Causation, prediction, and search.* MIT press.

[54] Brit Youngmann, Michael Cafarella, Amir Gilad, and Sudeepa Roy. 2024. Summarized Causal Explanations For Aggregate Views. *Proc. ACM Manag. Data* 2, 1, Article 71 (March 2024), 27 pages. https://doi.org/10.1145/3639328

[55] Brit Youngmann, Michael J. Cafarella, Babak Salimi, and Anna Zeng. 2023. Causal Data Integration. *Proc. VLDB Endow.* 16, 10 (2023), 2659–2665. https://doi.org/10.14778/3603581.3603602

[56] Y. Zhuge and H. Garcia-Molina. 1998. Graph structured views and their incremental maintenance. In *Proceedings 14th International Conference on Data Engineering*. 116–125. https://doi.org/10.1109/ICDE.1998.655767