PG-HIVE: Schema Discovery for Property Graphs

Sophia Sideri
sophisid@csd.uoc.gr
LIPADE, Université Paris Cité & University of Crete
Paris, France

Ioannis Chiras
csd4071@csd.uoc.gr
CSD, University of Crete
Heraklion, Greece

Myron Giakoumakis
myron@ics.forth.gr
FORTH-ICS
Heraklion, Greece

Georgia Troullinou
georgia.troullinou@univ-grenoble-alpes.fr
CNRS, Univ. Grenoble Alpes
Grenoble, France

Elisjana Ymeralli
ymeralli@ics.forth.gr
FORTH-ICS
Heraklion, Greece

Vasilis Efthymiou
vefthym@hua.gr
Harokopio University of Athens, Greece

Dimitris Plexousakis
dp@ics.forth.gr
FORTH-ICS
Heraklion, Greece

Haridimos Kondylakis
kondylak@ics.forth.gr
FORTH-ICS
Heraklion, Greece

Abstract

Property graphs are increasingly used to model complex and highly interconnected data across a wide range of domains. However, their schema-free nature continues to hinder understanding, integration, and efficient management. In this demo, we present PG-HIVE, an interactive, end-to-end framework for automatic schema discovery in property graphs. PG-HIVE unveils latent node and edge types, infers property datatypes and cardinalities, and supports both user-driven and fully adaptive clustering through a hybrid, incremental pipeline that avoids costly recomputation as new data arrives. The demonstration showcases a lightweight, intuitive web interface connected to a graph storage backend, enabling users to explore discovered schemata in real time through rich visualizations and customizable discovery settings. Beyond usability, PG-HIVE delivers state-of-the-art performance, improving schema-discovery accuracy by up to 65% for nodes and 40% for edges and achieving up to 1.95× faster execution compared to existing approaches. Designed for rapid deployment and experimentation, PG-HIVE empowers practitioners to perform schema discovery interactively, at scale, and on the fly, unlocking a deeper understanding of property graph structure.

Keywords

Schema Discovery, Property Graphs, Graph Databases

1 Introduction

Property graphs (PGs) have become a dominant model for representing complex, interconnected data in domains such as social networks, biomedical knowledge graphs, cybersecurity, and digital forensics. Their flexibility—allowing nodes and edges to carry arbitrary properties and multiple labels—makes them attractive for practitioners who must ingest diverse or rapidly evolving data. Yet, this schema-free nature comes at a cost. As PGs grow, users and systems are oblivious to the structure of the graph: What types of nodes and edges exist? Which properties characterize each type? How do relationships connect these types, and with what cardinalities or constraints? Without this knowledge, developers struggle to integrate datasets, analysts lack a clear view of the graph, and even basic operations like querying, visualization, and validation become more difficult or error-prone [5, 6, 8].

Schema discovery, the task of automatically inferring node types, edge types, properties, constraints, and cardinalities, is therefore essential for effective PG management. However, existing approaches remain limited. They often (i) assume complete or consistent label annotations, (ii) rely solely on label-based grouping, (iii) cannot handle heterogeneous or noisy data, or (iv) only extract node types without modeling relationships or constraints. Systems such as SchemI [7] and GMMSchema [3] and DiscoPG [2] perform well when data is fully annotated and clean, but degrade substantially under missing properties, incomplete labeling, or structural inconsistencies—conditions that are typical in real-world PGs.

To address these challenges, we present PG-HIVE, a hybrid, incremental, and adaptive system for schema discovery on property graphs. PG-HIVE employs a two-stage inference strategy that combines semantic embeddings of labels with structural similarity derived from properties, clustering nodes and edges using Locality-Sensitive Hashing (LSH)—either via Euclidean LSH or MinHash. This hybrid representation allows PG-HIVE to infer schema elements even when labels are partially missing or entirely absent. A second merging phase refines the discovered clusters into coherent types based on labels and –if unlabeled– high-similarity structural patterns. PG-HIVE then infers mandatory vs. optional properties, property datatypes, and relationship cardinalities, and exports the resulting schema in widely compatible formats, including PG-Schema and XSD. PG-HIVE also has an incremental module, which allows incremental updates in the schema as new of data arrives without recomputing the dataset from scratch.

Through extensive experiments on eight real and synthetic datasets, including heterogeneous, multi-labeled, and noisy graphs, PG-HIVE has been shown to consistently outperform state-of-the-art solutions [10]. It improves schema discovery accuracy by up to 65% for node types and 40% for edge types, and runs up to 1.95× faster than prior work for full schema inference. Crucially, PG-HIVE maintains accuracy even under extreme conditions, such as 0% label availability and up to 40% missing properties, where competing systems fail.

In this demonstration, we showcase the full interactive experience of PG-HIVE. Users can load a property graph from a backend store (e.g., Neo4j), explore clustering options (manual or adaptive), inspect intermediate and final schema elements, visualize discovered node and edge types, inspect property constraints, and compare schema extraction results under different parameter settings or noise levels, and baseline approaches. The demo highlights PG-HIVE's ability to (i) discover types in challenging scenarios, (ii) incrementally update schemas, and (iii) give users full control and transparency over the inference process

EDBT '26, Tampere (Finland)
© 2026 Copyright held by the owner/author(s). Published on OpenProceedings.org under ISBN 978-3-98318-104-9, series ISSN 2367-2005. Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0.

10.48786/edbt.2026.70

through a lightweight, intuitive web interface. More specifically, the contributions of this demonstration are the following:

- An interactive framework for PG schema discovery. Users can explore type inference, structural patterns, and schema characteristics in real time.
- A hybrid, LSH-based inference engine that remains effective even with incomplete labels, noisy data, or heterogeneous structures.
- Full schema extraction, including node/edge types, property data types, mandatory/optional attributes, and relationship cardinalities.
- Incremental schema discovery, enabling fast updates without recomputation as the graph grows.
- A hands-on walkthrough, allowing participants to tune clustering settings, observe trade-offs, and experiment with both static and incremental discovery modes.

Overall, this demonstration positions PG-HIVE as the first system to offer hybrid and incremental schema discovery for property graphs—bridging the gap between flexible PG data models and the growing demand for data management. The methodology of PG-HIVE is described in detail in the accompanying research paper [10]. The implementation of PG-HIVE[1] and its demo[2], along with all experimental configurations[3], are publicly available on GitHub.

2 System Overview

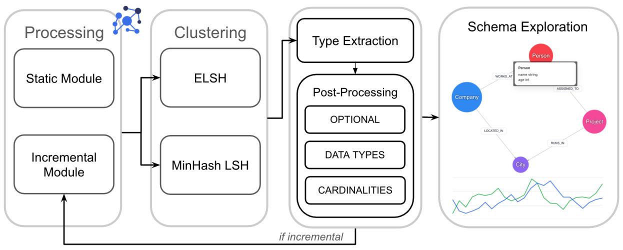


Figure 1: PG-HIVE high-level architecture.

PG-HIVE follows a modular architecture that connects a property graph backend with a hybrid discovery pipeline and an interactive web interface. Data flows from a graph store—typically Neo4j—into a processing engine creating vector representations, performing adaptive clustering, and inferring schema elements. The system supports both static execution, where the entire graph is processed at once, and incremental execution, where incoming batches update the schema without recomputing previous results. Once the schema is produced, it becomes accessible through a lightweight, browser-based interface that exposes all intermediate steps and final outputs. Figure 1 illustrates the overall architecture of the schema discovery and exploration framework. In the sequel we present in detail each one of the steps involved:

1. Preprocessing. The process begins with a uniform extraction of nodes, edges, and properties from the underlying graph. PG-HIVE issues a single compact query to retrieve the graph's structural information, ensuring consistency regardless of dataset size or shape. Each graph element is then transformed into a hybrid feature vector. Node and edge labels are encoded through a Word2Vec model trained on the dataset. Structural components,

[1]PG-HIVE: https://github.com/sophisid/PG-HIVE
[2]PG-HIVE Demo: https://github.com/sophisid/PG-HIVE_demo
[3]PG-Schema-Bench: https://github.com/sophisid/PG-SB

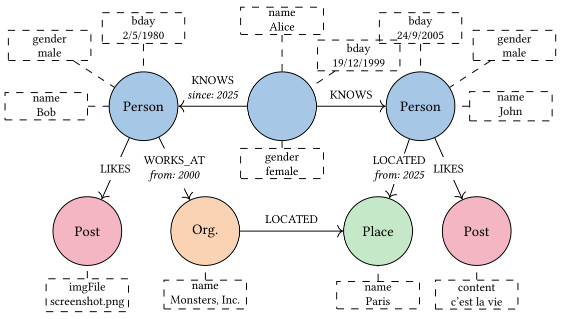


Figure 2: Example Property Graph.

i.e., the presence or absence of properties, are captured by binary vectors. This representation enables PG-HIVE to distinguish semantically different types even when they share similar structural signatures, while also allowing unlabeled or multi-labeled instances to be processed uniformly.

Example 1. During preprocessing, these representations allow PG-HIVE to encode the structural and semantic identity of each graph element. For instance, in the graph of Figure 2, the nodes `Bob` and `John` both carry the label `Person` and share the properties `name`, `gender`, and `bday`; they therefore receive nearly identical hybrid vectors composed of the same binary property pattern and highly similar Word2Vec embeddings. By contrast, the unlabeled node `Alice` is assigned a zero-vector for the missing label information but retains the same property pattern as the other two persons, enabling PG-HIVE to later recover her latent type purely from structural resemblance. A similar effect appears for edges such as `WORKS_AT`, where the embedding encodes both the label of the relation and the label embeddings of the source and target nodes, together with a binary signature for properties like `from`. This preprocessing ensures that the subsequent clustering step operates on meaningful geometric representations of the data.

2. Adaptive Clustering. Using hybrid representations, PG-HIVE applies LSH to group structurally and semantically similar elements. The system supports both Euclidean LSH and MinHash, making it effective for different graph characteristics. Rather than relying on manual parameter tuning, PG-HIVE automatically derives clustering parameters by sampling the input graph. From this sample, it estimates the typical distance between elements by the sparsity of property distributions and the diversity of label sets. These guide the selection of bucket widths and the number of hash tables, ensuring that clusters remain coherent even under noisy or heterogeneous conditions. The output of this stage is a set of candidate clusters that represent preliminary node and edge types.

Example 2. The benefit of the adaptive parameter selection becomes evident when clustering structurally coherent patterns. For example, all `LIKES` edges in Figure 2 consistently connect a `Person` to a `Post` and lack additional properties, which causes their vectors to collide frequently across the LSH tables and thus form a tight, homogeneous cluster. Nodes `Bob` and `John` likewise cluster together because their representations are nearly identical. Even the unlabeled node `Alice` gravitates toward the same cluster due to her matching structural signature, despite lacking a label. Conversely, the two `Post` nodes initially fall into distinct buckets because one contains `imgFile` and the other `content`, demonstrating how the adaptive LSH configuration remains sensitive to fine-grained structural variation when necessary.

3. Schema Inference. The schema inference module refines the candidate clusters into formal schema types. Clusters associated with identical label sets are merged immediately, while unlabeled clusters are compared with labeled ones using Jaccard similarity over their property sets. When similarity exceeds a conservative threshold, the clusters merge; otherwise, unlabeled clusters remain and form ABSTRACT types according to PG-Schema [1, 4]. After the type structure stabilizes, PG-HIVE infers schema constraints such as mandatory and optional properties based on their frequency, datatypes through lightweight heuristics over observed values, and relationship cardinalities by examining in and out degree distributions. The final schema is serialized into PG-Schema (both STRICT and LOOSE variants) and XSD for usability. In incremental mode, newly produced clusters are merged into the existing schema, ensuring a monotonic evolution without reprocessing previous data.

Example 3. During schema inference, the clusters are refined into coherent types. For example, the unlabeled cluster containing `Alice` is merged into the `Person` type because its property set exhibits nearly perfect Jaccard similarity with the labeled `Person` cluster containing `Bob` and `John`. Likewise, the structurally distinct `Post` clusters are unified because they share the same label and represent semantically identical entities, leading to a `Post` type that includes both `imgFile` and `content` as possible properties. Once the types have been determined, the system infers property constraints such as mandatoriness: all `Person` instances include `name`, `gender`, and `bday`, while `Post` exhibits optional attributes due to its structural variability. Datatype inference characterizes properties such as `bday` as `DATE` or `name` as `STRING`, while cardinality analysis correctly identifies relations like `WORKS_AT` as N:1 and `KNOWS` as M:N, thus completing the schema.

4. Schema Exploration. The resulting schema is made available through a dedicated web interface (refer to Figure 3). Users can inspect the discovered node and edge types, explore their associated labels and structural patterns, examine inferred properties and cardinalities, and visualize how clusters and types evolve when new batches of data are introduced. The interface allows switching between manual and adaptive clustering configurations and reveals intermediate representations, enabling users to understand why certain elements were grouped together or kept apart. Because the interface is lightweight and browser-based, schema exploration remains responsive even for large graphs, making PG-HIVE suitable both for research environments and real-world deployment.

Example 4. The resulting schema can be explored through the system's PG-Schema interface, where each inferred type is presented in both its LOOSE and STRICT forms. For example, the `Person` type inferred from `Alice`, `Bob`, and `John` is displayed with its mandatory attributes `name`, `gender`, and `bday`, along with their inferred datatypes. Users may edit these declarations or express new ones, such as defining additional constraints or adding optional properties, and PG-HIVE immediately reflects these modifications in a graphical visualization of the schema. Similarly, the `WORKS_AT` and `LIKES` edge types appear with their respective endpoints and inferred cardinalities, allowing users to validate or refine their structural assumptions interactively.

3 Demonstration

The goal of the demonstration is to guide participants through the complete workflow of schema discovery on property graphs using PG-HIVE. Throughout the session, users will observe how PG-HIVE handles fully labeled, partially labeled, noisy, and evolving datasets, and will be able to interact with each stage of the system.

Datasets. We demonstrate PG-HIVE using eight datasets [9] that vary in size, structure, and complexity, including both real-world and synthetic data: POLE, MB6, HET.IO, FIB25, ICIJ, LDBC, CORD19, and IYP [4]. These datasets have diverse schema characteristics, ranging from well-structured graphs to highly incomplete or noisy ones, providing a comprehensive test for schema discovery. Additionally, we test multiple noise (10%, 20%, 30%, and 40%) and label availability variants (full label availability, 50% label availability, and no labels available), showing how they impact the quality of the schema discovery. The demonstration will proceed in five concrete steps:

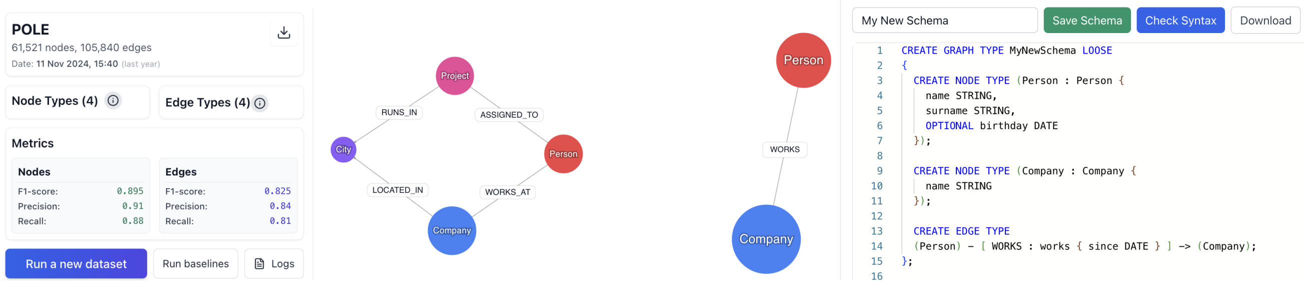
1. Connecting to the Property Graph Store. The demonstration begins with the PG-HIVE interface open, where the user first selects the desired database endpoint and provides the corresponding username and password. After authentication, the system presents a list of available datasets, previously described in the demo context, and the user chooses one to analyze. For each dataset, PG-HIVE immediately displays key statistics such as the total number of nodes and edges, giving an overview of the graph's scale before schema discovery begins. In the same initial configuration stage, the user can also select the clustering backend, choosing either ELSH or MinHash-LSH, and specify whether clustering should run in adaptive mode—allowing PG-HIVE to automatically derive optimal parameters—or in manual mode. When opting for manual configuration, users may set the number of LSH hash tables, the bucket length, and the Jaccard similarity threshold that governs cluster merging. Once these parameters are defined, the system is ready to proceed to the vectorization and discovery phases.

2. Clustering. Once the initial configuration is completed, PG-HIVE begins the clustering phase by first loading the selected dataset from the backend store and constructing hybrid vector representations for all nodes and edges. In parallel, the system generates semantic label embeddings and binary property indicators. When vectorization is complete, PG-HIVE applies its LSH-based clustering engine—using either ELSH or MinHash-LSH depending on the user's choice—to group structurally and semantically similar graph elements. The interface then presents the resulting clusters together with summary statistics such as the number of node and edge types discovered. From these clusters, PG-HIVE extracts refined node and edge types by merging labeled clusters with identical label sets and integrating unlabeled clusters when they exhibit sufficiently high structural similarity, while unmatched ones form abstract types according to the PG-Schema framework. The corresponding evaluation metrics, including precision, recall, and F1-score when ground truth is available.

3. Cardinalities and Constraints. The system enriches each type by inferring its mandatory and optional properties, computing datatypes from observed values, and deriving cardinalities for each relationship type using degree distributions. These results are visualized as an interactive schema graph, allowing users to explore the discovered structure by inspecting each type's loose and strict PG-Schema representations as well as its associated properties and constraints. Through this unified clustering and inference workflow, PG-HIVE offers a complete and interpretable view of the schema underlying the input property graph.

4. Comparison with Baselines. Next, we present comparisons with existing schema discovery systems such as GMMSchema and SchemI (Figure 4). Participants can view, for the same dataset,

[4]PG-Schema-Bench: https://github.com/sophisid/PG-SB



POLE
61,521 nodes, 105,840 edges
Date: 11 Nov 2024, 15:40 (last year)

Node Types (4) | Edge Types (4)

Metrics

Nodes		Edges	
F1-score:	0.895	F1-score:	0.825
Precision:	0.91	Precision:	0.84
Recall:	0.88	Recall:	0.81

```
CREATE GRAPH TYPE MyNewSchema LOOSE
{
  CREATE NODE TYPE (Person : Person {
    name STRING,
    surname STRING,
    OPTIONAL birthday DATE
  });

  CREATE NODE TYPE (Company : Company {
    name STRING
  });

  CREATE EDGE TYPE
  (Person) – [ WORKS : works { since DATE } ] -> (Company);
};
```

Figure 3: The GUI of PG-HIVE, showing results (left) and exploring PG-Schema (right).

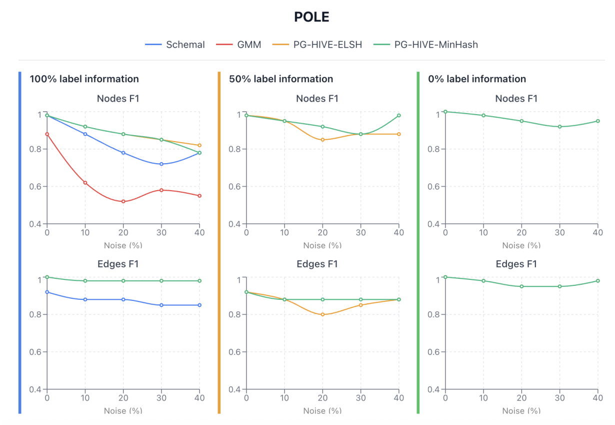


Figure 4: Dashboard of systematic evaluation of PG-HIVE against the baselines.

the clusters and inferred types produced by each system, along with their precision, recall, and F1-scores based on the ground-truth schema. The overall comparison is presented in a dashboard, highlighting cases where baselines struggle—particularly under missing labels or injected noise—and contrasts them with PG-HIVE's stable and accurate results. Execution-time summaries are also displayed, illustrating that PG-HIVE achieves competitive or superior performance even while discovering both node and edge types and inferring richer schema characteristics. This comparative view allows participants to directly observe how PG-HIVE improves accuracy by up to 65% for node types, up to 40% for edge types, and achieves up to 1.95× faster performance compared to previous approaches.

5. PG-Schema Creator. Finally, users can further explore PG-HIVE's integrated PG-Schema learning environment, which offers a dedicated panel for understanding and authoring PG-Schema declarations. The system automatically translates each discovered node and edge type into both its LOOSE and STRICT PG-Schema variants, allowing users to examine how structural patterns, property constraints, and relationship cardinalities are formally encoded. In this view, users can experiment by editing or writing PG-Schema declarations themselves, guided by contextual hints and examples derived from the inferred schema. As declarations are written or modified, PG-HIVE immediately renders their graphical representation, showing the corresponding types, properties, and relationships in a visual schema graph. This bidirectional exploration—moving from the inferred schema to the formal PG-Schema syntax and back to an automatically updated visualization—enables users to gain hands-on familiarity with the language, understand its expressiveness and constraints, and verify the correctness of their own schema definitions through real-time feedback.

4 Conclusion

PG-HIVE introduces a robust and adaptive framework for schema discovery in property graphs, addressing key challenges posed by missing labels, heterogeneous structures, and evolving datasets. By combining hybrid vector representations with adaptive LSH clustering and a principled schema inference pipeline, PG-HIVE delivers accurate node and edge type discovery together with inferred datatypes, constraints, and cardinalities. Its incremental processing capabilities further enable schema maintenance as new data arrives, avoiding costly recomputation. Through an intuitive and transparent web interface, users can explore intermediate steps, adjust configurations, and compare results with existing approaches. Overall, the demonstration highlights PG-HIVE as a practical, scalable, and versatile system that advances schema discovery for real-world property graph applications.

Acknowledgments

Work supported by the TALE H.F.R.I. (16819), DataGEMS (101188416), and by DEDALUS CREs (TA 5180519) projects .

References

[1] Shqiponja Ahmetaj, Iovka Boneva, Jan Hidders, Katja Hose, Maxime Jakubowski, Jose Emilio Labra Gayo, Wim Martens, Fabio Mogavero, Filip Murlak, Cem Okulmus, et al. 2025. Common foundations for SHACL, ShEx, and PG-Schema. In *WWW*. 8–21.
[2] Angela Bonifati, Stefania-Gabriela Dumbrava, Emile Martinez, Fatemeh Ghasemi, Malo Jaffré, Pacome Luton, and Thomas Pickles. 2022. DiscoPG: Property Graph Schema Discovery and Exploration. *PVLDB* (2022).
[3] Angela Bonifati, Stefania Dumbrava, and Nicolas Mir. 2022. Hierarchical Clustering for Property Graph Schema Discovery. In *EDBT*.
[4] Kasidis Chantharojwong, Sourav S Bhowmick, and Byron Choi. 2025. LICS: Towards Theory-Informed Effective Visual Abstraction of Property Graph Schemas. *SIGMOD* 3, 3 (2025), 1–26.
[5] Haridimos Kondylakis, Stefania Dumbrava, Matteo Lissandrini, Nikolay Yakovets, Angela Bonifati, Vasilis Efthymiou, George Fletcher, Dimitris Plexousakis, Riccardo Tommasini, Georgia Troullinou, and Elisjana Ymeralli. 2025. Property Graph Standards: State of the Art & Open Challenges. *PVLDB* 18, 12 (2025).
[6] Haridimos Kondylakis, Vassilis Efthymiou, Georgia Troullinou, Elisjana Ymeralli, and Dimitris Plexousakis. 2024. Property Graphs at Scale: A Roadmap and Vision for the Future (Short Paper). In *CAiSE*. Springer, 180–185.
[7] Hanâ Lbath, Angela Bonifati, and Russ Harmer. 2021. Schema Inference for Property Graphs. In *EDBT*. 499–504. doi:10.5441/002/EDBT.2021.58
[8] Sophia Sideri, Vasilis Efthymiou, Dimitris Plexousakis, and Haridimos Kondylakis. 2025. PG2RDF: Schema-Guided Transformation of Property Graphs to RDF. In *IJCKG*.
[9] Sophia Sideri, Georgia Troullinou, Elisjana Ymeralli, Vasilis Efthymiou, Dimitris Plexousakis, and Haridimos Kondylakis. 2025. *PG-Schema Bench: A Benchmark for Schema Discovery in Property Graphs.* doi:10.5281/zenodo.17801336
[10] Sophia Sideri, Georgia Troullinou, Elisjana Ymeralli, Vasilis Efthymiou, Dimitris Plexousakis, and Haridimos Kondylakis. 2026. PG-HIVE: Hybrid Incremental Schema Discovery for Property Graphs. *EDBT* (2026).