Master in Data Science

Universitat Politècnica de Catalunya

Property Graphs

Contributors

Laura González Villar
Jonàs Salat Torres

April 9, 2025

Contents

1 Part A: Modeling, Loading, Evolving

1.1 A.1 Modeling

The following graph representation (Figure 1) consists of nodes (entities) and edges (relationships) that model the structure of research publications in terms of conferences, journals, authors, and citations.

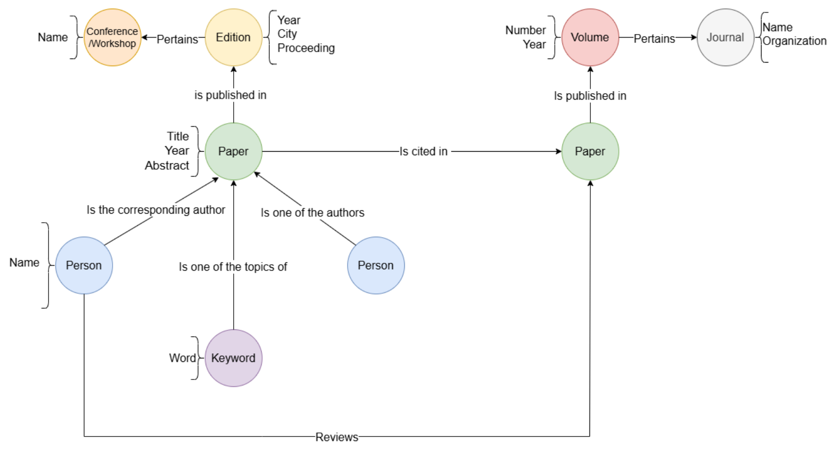

Figure 1: Graph schema representing the structure of research publications.

This design ensures scalability, efficiency, and expressiveness while maintaining data integrity and flexibility.

The Person node only includes the name attribute, and the roles (Author, Reviewer, Corresponding Author) are reflected in the relationships. This approach avoids triplicating people, as a single individual can act as a reviewer, corresponding author, or author, ensuring better data integrity and reducing redundancy. Proceedings are stored as an attribute of Events, without redundant relationships and improving query performance.

Finally, the h-index and impact factor are dynamically computed, ensuring data consistency while avoiding storage duplication. In conclusion, this design effectively represents complex relationships while keeping the graph scalable and adaptable for future expansions, maintaining an optimal balance between performance, integrity, and flexibility.

1.2 A.2 Instantiating/Loading

To instantiate and load the data into Neo4j, CSV files were generated for all nodes and relationships. The dataset combines real data retrieved via the `apicall.py` script using the Semantic Scholar API (querying over*graph databases* and *big data*) with synthetically generated data.

From Semantic Scholar API, the following data was extracted:

- **Papers**: Including ID, title, year, abstract, and DOI.
- **Persons**: Authors with their names and IDs.
- **Keywords**: Extracted from the fieldsOfStudy field.
- **Venues**: Classified as *Conference/Workshop* or *Journal* based on venue names.

The real relationships included:

- **Wrote** and **Corresponding_Author**: Connecting authors to papers (assuming the first author is the corresponding one).
- **Cites**: Built from reference lists using paper IDs.
- **Has_Keyword**: Links papers to their topics.

Synthetic data was used to complete the model:

- **Edition** and **Volume** nodes were created using paper year, venue name, and random cities.
- **Reviews**: Reviewers were randomly assigned, excluding authors of the same paper.
- **Pertains**: Links editions to conferences/workshops and volumes to journals.
- **Is_Published_In**: Associates each paper with its corresponding edition or volume.

1.3 A.3 Evolving the graph

The updated graph model can be observed in Figure 2. It introduces new elements and enhancements to support the evolution of the review process and the representation of author affiliations.

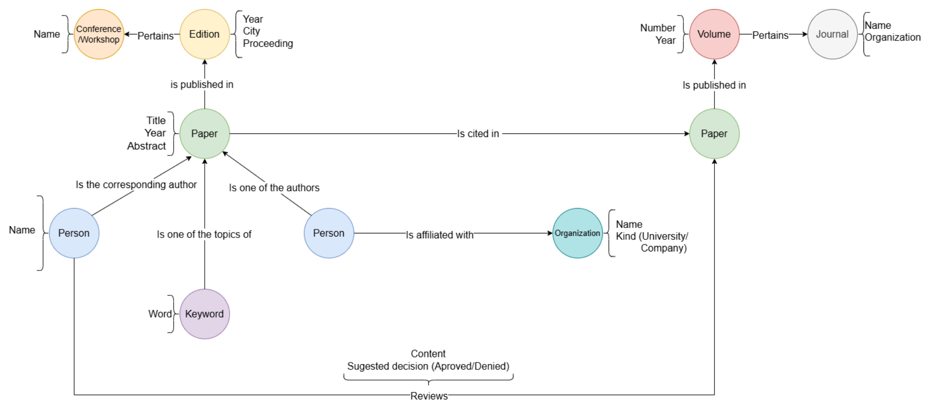

Figure 2: Updated graph schema.

The Reviews relationship has been extended with two new attributes: content (the review itself) and decision (either approved or denied). These changes provide a more faithful representation of peer review processes, directly embedding key review data within the graph.

Regarding the review policy, it was decided not to explicitly model a Review Policy node in the graph. Since the available data comes from Semantic Scholar API, the involved papers are already accepted publications, and modeling variable review policies per venue lacks practical value in this context. Instead, the review dynamics were simulated programmatically using logic embedded in the extendreviews.py script. In particular, a rule ensuring that a paper is only accepted if at least half of its reviews are positive was implemented, respecting the assumption that acceptance requires majority approval.

This design decision streamlines the graph structure by avoiding unnecessary nodes while still reflecting review logic through data generation scripts. At the same time, the model remains scalable and expressive, accurately reflecting core concepts such as author affiliation and review outcomes while staying compatible with real-world data limitations. This evolved design improves the graph in terms of maintainability, reusability, non-redundancy, and performance. By extending the Reviews relationships instead of creating an additional node, the graph structure remains compact and easier to maintain. The Organization node and its relationships are generic enough to be reused in future extensions, such as modeling research collaborations or institutional metrics. From a performance standpoint, storing review content and decision directly within the relationship enables efficient filtering and querying, which directly benefits the performance of queries. Moreover, affiliation data can be easily leveraged to analyze author communities and collaboration networks. Overall, this evolution maintains a clean and coherent graph structure while significantly increasing its expressiveness and adaptability to complex research data scenarios.

2 Part B: Querying

This section focuses on querying the Neo4j graph developed in the previous stages, with the goal of extracting meaningful insights from the structured research publication data. A set of Cypher queries has been designed to explore different aspects of the graph, including citation patterns, author contributions, and venue impact. Selected queries and representative results are presented and briefly analyzed throughout this section to illustrate the expressiveness and analytical power of the property graph model.

First query – Top 3 most cited papers of each conference/workshop

```
MATCH (p:Paper)-[:PUBLISHED_IN]->(:Edition)-[:PERTAINS]->(c:ConferenceWorkshop)
OPTIONAL MATCH (:Paper)-[:CITES]->(p)
WITH c.name AS conference, p.title AS title, p.paper_id AS id, count(*) AS
↪  citation_count
ORDER BY conference, citation_count DESC
WITH conference, collect({title: title, id: id, citations: citation_count}) AS papers
RETURN conference, papers[0..3] AS top_3_papers;
```

Table 1: Venues with their top 3 cited papers and their citation count

Conference/Workshop	**Paper Title**	**Citations**
ACM Computing Surveys	Foundations of Modern Query Languages for Graph Databases	10
	Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries	1

Conference/Workshop	Paper Title	Citations
	An Overview of End-to-End Entity Resolution for Big Data	1

Second query – Identify authors communities

```
MATCH (a:Person)-[:WROTE]->(p:Paper)-[:PUBLISHED_IN]->(e:Edition)-[:PERTAINS]->
(c:ConferenceWorkshop)
WITH c.name AS conference, a.name AS author, count(DISTINCT e.edition_id) AS editions
WHERE editions >= 4
RETURN conference, author, editions
ORDER BY conference, editions DESC;
```

Table 2: Authors considered part of a community

Conference/Workshop	Author	Editions
Journal of Big Data	T. Khoshgoftaar	4

Third query – Impact factors

```
// Step 1: Get all papers and their journals
MATCH (j:Journal)<-[:PERTAINS]-(v:Volume)<-[:PUBLISHED_IN]-(p:Paper)
WITH j, p
// Step 2: Count how many times each paper is cited
OPTIONAL MATCH (p)<-[:CITES]-(:Paper)
WITH j.journal_id AS journal, count(*) AS citation_count, count(DISTINCT p) AS
↪  paper_count
RETURN journal,
       sum(citation_count) * 1.0 / paper_count AS impact_factor,    //Use 1.0 to get
       ↪  decimal result
       sum(citation_count) AS total_citations,
       paper_count AS total_papers
ORDER BY impact_factor DESC;
```

Table 3: Impact factor

Journal	Impact Factor	Total Citations	Total Papers
IEEE transactions on intelligent transportation systems (Print)	1.5	3	2

Forth query – H-index

```
// Step 1: Get each author's papers and their citation counts
MATCH (a:Person)-[:WROTE]->(p:Paper)
OPTIONAL MATCH (:Paper)-[:CITES]->(p)
WITH a.name AS author, p, count(*) AS citation_count

// Step 2: Group citation counts into a list per author
WITH author, collect(citation_count) AS citations

// Step 3: Unwind and sort manually (descending)
UNWIND citations AS c
WITH author, c
ORDER BY author, c DESC

// Step 4: Rank citations per author
WITH author, collect(c) AS sorted_citations
UNWIND range(0, size(sorted_citations)-1) AS i
WITH author, i + 1 AS rank, sorted_citations[i] AS citation
WHERE citation >= rank
WITH author, max(rank) AS h_index
RETURN author, h_index, sorted_citations
ORDER BY h_index DESC;
```

Table 4: H-index

Author	**H-index**
A. Mazein	2

3 Part C: Recommender

This section aims to build a reviewer recommender for conferences and journals by analyzing the publication graph and identifying authors with strong expertise in the field of databases. The process is divided into four stages, each implemented as a single Cypher query.

Step 1 – Identify research community papers

```
MATCH (p:Paper)
WHERE any(word IN ["data management", "indexing", "data modeling", "big data",
                   "data processing", "data storage", "data querying"]
           WHERE toLower(p.title) CONTAINS word)
SET p:DatabaseCommunityPaper;
```

This query tags papers whose titles contain key database-related terms with the label :DatabaseCommunityPaper. This allows the system to classify which papers belong to the database research community. Using a label here facilitates fast access to these nodes in later steps and avoids repetitive keyword filtering.

Step 2 – Detect relevant venues (conferences, workshops, and journals)

```
// For Conference/Workshop
MATCH (v:ConferenceWorkshop)<-[:PUBLISHED_IN]-(p:Paper)
WITH v, collect(p) AS papers
UNWIND papers AS p
WITH v, count(DISTINCT p) AS total,
     count(DISTINCT CASE WHEN p:DatabaseCommunityPaper THEN p END) AS db_related
WHERE total > 0 AND db_related * 1.0 / total >= 0.9
SET v:DatabaseVenue;

// For Journals
MATCH (j:Journal)<-[:PERTAINS]-(v:Volume)<-[:PUBLISHED_IN]-(p:Paper)
WITH j, collect(p) AS papers
UNWIND papers AS p
WITH j, count(DISTINCT p) AS total,
     count(DISTINCT CASE WHEN p:DatabaseCommunityPaper THEN p END) AS db_related
WHERE total > 0 AND db_related * 1.0 / total >= 0.9
SET j:DatabaseVenue;
```

This step identifies venues whose content is highly focused on the database domain. If 90% or more of the papers in a conference edition or journal volume are part of the database community, the venue is labeled as :DatabaseVenue. This classification helps isolate venues that are domain-specific, improving the precision of future recommendations.

Step 3 – Select top papers in the community

```
MATCH (p:Paper)-[:PUBLISHED_IN]->(v)
OPTIONAL MATCH (v)<-[:PERTAINS]-(vol:Volume)-[:PERTAINS]->(j:Journal)
WITH p, v, vol, j
WHERE v:DatabaseVenue OR j:DatabaseVenue
OPTIONAL MATCH (:Paper)-[:CITES]->(p)
WITH p, count(*) AS citations
ORDER BY citations DESC
LIMIT 100
SET p:TopDBPaper;
```

In this stage, the system selects the top 100 most cited papers published in venues marked as

:DatabaseVenue. These papers are assumed to represent the most influential contributions in the database community and are labeled :TopDBPaper for easy reference.

Step 4 – Identify expert reviewers and gurus

```
MATCH (a:Person)-[:WROTE]->(p:TopDBPaper)
WITH a, count(p) AS top_paper_count
SET a:PotentialReviewer
WITH a, top_paper_count
WHERE top_paper_count >= 2
SET a:Guru;
```

Authors who wrote at least one top paper are labeled :PotentialReviewer, and those who contributed to two or more are labeled :Guru. This distinction enables fine-grained reviewer selection.

All inferred knowledge is incorporated through semantic labels rather than creating new nodes or relationships. This design decision aligns with the principles established in Section A.1 and A.3 of maintaining a non-redundant, semantically rich, and performance-oriented graph model. Labels such as :DatabaseCommunityPaper, :DatabaseVenue, :TopDBPaper, :PotentialReviewer, and :Guru enrich existing nodes without altering the core structure of the graph. Thisdesign choice keeps the model clean and supports fast pattern matching. Overall, this strategy offers a scalable and maintainable way to assert inferred knowledge into the graph.

4 Part D: Graph algorithms

This section aims to apply graph algorithms to compute meaningful metrics. The selected algorithms are PageRank and Louvain, chosen not only for their computational relevance, but also because they are semantically well-aligned with the structure and goals of our dataset.

PageRank was selected to identify the most influential papers within the citation network. In the context of academic publishing, citations are a strong indicator of relevance and impact. PageRank leverages both the quantity and quality of incoming citations, making it a suitable metric to highlight key publications in the research community.

Louvain was chosen to detect communities of papers based on citation patterns. This algorithm is particularly useful to uncover thematic clusters, suggesting common topics, research areas, or disciplinary boundaries. Given that part C of the project involved identifying research communities, Louvain naturally complements that goal.

Prior to execution, in-memory graph projections were constructed to reflect the desired semantics.

For PageRank, a directed graph was used where edges represent citation relationships. Additionally, for Louvain, an undirected version of the citation graph was built, as community detection is generally symmetric with respect to connection strength rather than directionality.

This ensures that the algorithms are not only used correctly but also appropriately for our domain.

The output for PageRank consists of a ranked list of papers, each with a PageRank score. High scores indicate papers that are central to the graph in terms of incoming citations, especially from other well-cited papers. These can be interpreted as seminal or foundational works within the dataset, and can guide, for instance, reviewer selection or research trend analysis.

On the other hand, Louvain results show papers grouped into communities (louvainCommunity) based on the density of citation links. Each community can be viewed as a subfield or topic cluster, inferred purely from citation behavior. For example, papers in the same community likely address similar problems, use related methodologies, or belong to the same research tradition. These communities could also help classify events, venues, or even authors and support further tasks like community-aware recommendation.

The use of these two algorithms provides complementary insights. While PageRank highlights individual influence, Louvain reveals group structure. Together, they enrich the understanding of the graph beyond simple structural properties, aligning well with the domain of academic research networks.