LAB Assignment

Property Graph

Tejaswini Dhupad

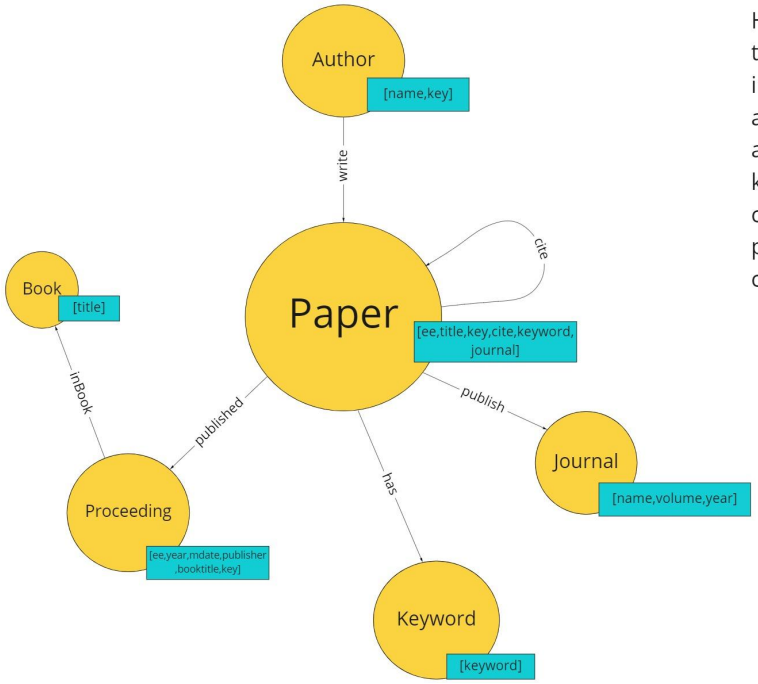
Chun Han Li

GitHub : https://github.com/tejaswinidhupad/Property-Graphs-Modeling-Research-Paper

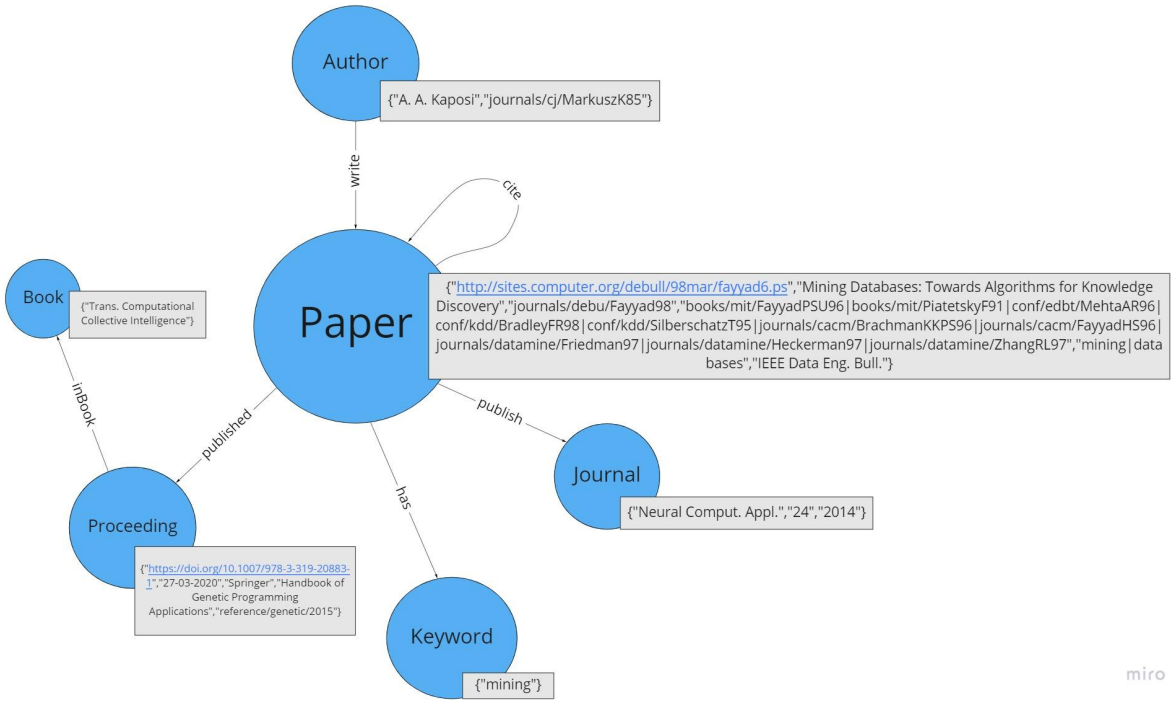
---

A. Modeling, Loading, Evolving

A1. Modeling



Here, the Paper is published in the journal or it can be published in the book. An author can write a paper as well as can have co-authors. The paper has keywords. The paper can also be cited by other papers. The proceeding papers can collectively be part of the book.



A2. Instantiating/Loading

The data is used from the official website of DBLP. The dataset was in the XML format, hence was needed to convert to CSV format for loading into Neo4j. The CSV files were generated by running the below command :

```
XMLToCSV.py xml_filename dtd_filename outputfile
```

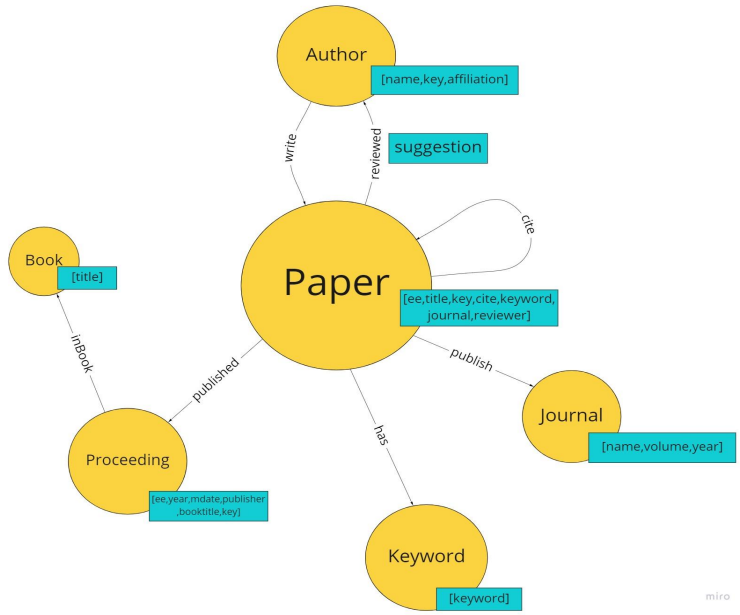
After that, we got several CSV files. The data in the CSV is explained as below :

- *article* – An article from a journal or magazine.
- *inproceedings* – A paper in a conference or workshop proceedings.
- *proceedings* – The proceedings volume of a conference or workshop.
- *book* – An authored monograph or an edited collection of articles.
- *incollection* – A part or chapter in a monograph.
- *phdthesis* – A PhD thesis.
- *mastersthesis* – A Master's thesis. There are only very few Master's theses in dblp.
- *www* – A web page. There are only very few web pages in *dblp*

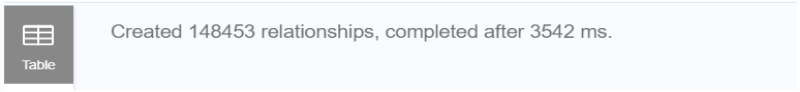
Following the model we designed in A1 task, we load the data into Neo4j, including nodes, edges, and set keys as indices as mentioned in *PartA2_LiDhupad.py*.

A3. Evolving the graph

In this case, a new relation “reviewed” with property “suggestion” from node Author to node Paper and “affiliation” to the node Author were added. And this was achieved by adding just three Cypher queries mentioned in *PartA3_LiDhupad.py* file. This shows that graph databases are highly flexible.



```
// Here is an example of one of the Cypher query to adjust the design changes
LOAD CSV WITH HEADERS FROM 'file:///papers.csv'
AS tr
MATCH (paper:Paper{key:tr.key})
UNWIND SPLIT(tr.reviewer,"|") as reviewer
MATCH (author:Author {name:reviewer})
MERGE (paper)-[:reviewed]->(author)
```



After adding these properties and relation, authors can give positive or negative comments to the paper using property “suggestion”, see more detail in PartA3_LiDhupad.py

B. Querying

1. Find the top 3 most cited papers of each conference.

```
MATCH (citingPaper:Paper)-[citation:Cite]→(citedPaper:Paper)-[:published]-(proceeding:Proceeding)
WITH COUNT(citation) AS citedCount, citedPaper, proceeding ORDER BY citedCount DESC
WITH COLLECT(citedPaper.title) AS mostCitedPaper, proceeding
RETURN proceeding.title as conference, mostCitedPaper[0..3] as top_3_most_cited_papers
```

"conference"	"top_3_most_cited_papers"
"Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, USA, May 30 - June 1."	["Access Path Selection in a Relational Database Management System.","FQL - A Functional Query Language.","Data Abstractions, Views and Updates in RIGEL."]
"Operating Systems, An Advanced Course"	["Notes on Data Base Operating Systems.","The Object Model: A Conceptual Tool for Structuring Software.","Reliable Computing Systems."]
"SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, USA, June 18-21, 1984"	["R-Trees: A Dynamic Index Structure for Spatial Searching.","Implementation Techniques for Main Memory Database Systems.","Making Smalltalk a Database System."]

2. For each conference find its community: i.e., those authors that have published papers on that conference in, at least, 4 different editions.

```
MATCH (a:Author)-[:Write]→(pa:Paper)-[pi:published]→(p:Proceeding)
with a, p, count(pa.key) as no_edition
where p.key =~ 'conf.*'
and no_edition ⩾ 4
return a.name as author_name,p.key as conference, no_edition
```

"author_name"	"conference"	"no_edition"
"Clement T. Yu"	"conf/icde/95"	4
"Rajeev Rastogi"	"conf/icde/99"	4
"Peter P. Chen"	"conf/er/81"	4
"Hector Garcia-Molina"	"conf/sigmod/97"	6

3. Find the impact factors of the journals in your graph.

Impact Factor(IF) is calculated by the below formula :

$$\mathrm{IF}_y = \frac{\mathrm{Citations}_y}{\mathrm{Publications}_{y-1} + \mathrm{Publications}_{y-2}}.$$

Here, we have calculated the IF for the year = 2000.

```
MATCH (p1:Paper)-[:PublishedIn]→(j1:Journal)
        WHERE toInteger(j1.year)=(2000-1) OR toInteger(j1.year)=(2000-2)
        WITH j1.name as JournalName, size(COLLECT(p1)) AS nop, COLLECT(p1) AS c_journal
        MATCH(p1:Paper)-[c1:Cite]→(p2:Paper)
        WHERE p1 IN c_journal
        RETURN JournalName, (toFloat(COUNT(c1))/nop) AS ImpactFactor ORDER BY ImpactFactor DESC
```

"JournalName"	"ImpactFactor"
"ACM Comput. Surv."	76.66666666666667
"VLDB J."	26.88888888888889

4. Find the h-indexes of the authors in your graph.

If we have the function *f* ordered in decreasing order from the largest value to the lowest one, we can compute the *h-index* as follows:

$$h\text{-index } (f) = \max\{i \in \mathbb{N} : f(i) \geq i\}$$

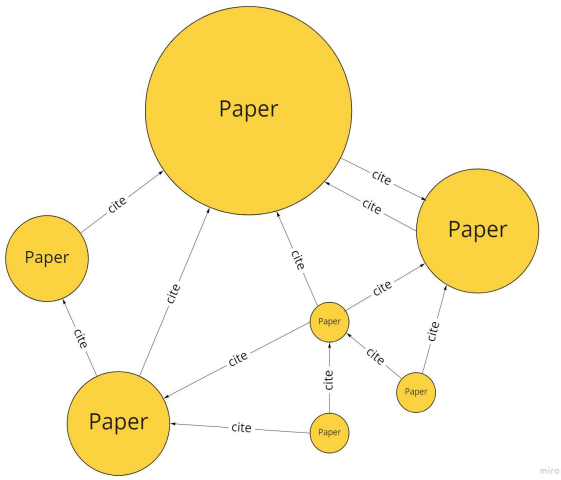
```
match (a:Author)-[w:Write]→(p:Paper)-[c:Cite]→(p1:Paper)
        with a,p1,collect([id(p),p.title]) as rows
        WITH a,p1,RANGE(1,SIZE(rows))AS enumerated_rows
        UNWIND enumerated_rows AS er
        with a,er AS rank,count(p1) as cited
        where rank ⩽ cited
        return a.name as authorname, rank as hindex
        order by hindex desc
```

"authorname"	"hindex"
"C. Mohan 0001"	10
"Christos Faloutsos"	10

The above-mentioned queries are in *PartB_LiDhupad.py* file.

C. Graph algorithms

1. **PageRank of Paper citation** : The *PageRank* [1]algorithm measures the importance of each node within the graph, based on the number of incoming relationships and the importance of the corresponding source nodes. The assumption is that a page is only as important as the pages that link to it.



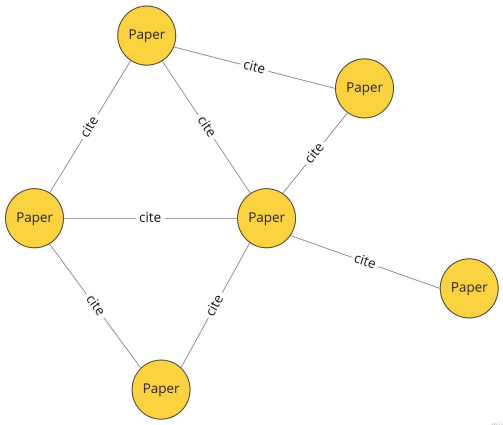

```
CALL gds.pageRank.stream('myGraph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).title AS paper,score
ORDER BY score DESC
```

"paper"	"score"
"A Relational Model of Data for Large Shared Data Banks."	27.847341458244156
"The Entity-Relationship Model - Toward a Unified View of Data."	15.388143200479087
"Further Normalization of the Data Base Relational Model."	11.95376485815876
"System R: Relational Approach to Database Management."	11.50421761507598
"The Notions of Consistency and Predicate Locks in a Database System."	10.361983311937948

2. **Triangle Count for Paper citations** : The *Triangle Count* [2]algorithm counts the number of triangles for each node in the graph. A triangle is a set of three nodes where each node has a relationship to the other two. In graph theory terminology, this is sometimes referred to as a 3-clique. The Triangle Count algorithm in the GDS library only finds triangles in undirected graphs.

   NOTE : We needed to change the graph database property from directed to undirected graph.

[1] *PageRank - Neo4j Graph Data Science*. (2022). Neo4j Graph Data Platform. https://neo4j.com/docs/graph-data-science/current/algorithms/page-rank/
[2] *Triangle Count - Neo4j Graph Data Science*. (2022). Neo4j Graph Data Platform. https://neo4j.com/docs/graph-data-science/current/algorithms/triangle-count/




```
1 CALL gds.triangleCount.stream('myGraph2')
2 YIELD nodeId, triangleCount
3 RETURN gds.util.asNode(nodeId).title AS title, triangleCount
4 ORDER BY triangleCount DESC
```

Table | Text | Code

"title"	"triangleCount"
"The Entity-Relationship Model - Toward a Unified View of Data."	2316
"A Relational Model of Data for Large Shared Data Banks."	2194
"Access Path Selection in a Relational Database Management System."	2149
"R-Trees: A Dynamic Index Structure for Spatial Searching."	1714
"Query Evaluation Techniques for Large Databases."	1602

The queries for the Graph algorithm are in *PartC_LiDhupad.py* file.

D. Recommender

In this recommender there are 3 major Cypher queries, please check the python file *PartD_LiDhupad.py* file. In the first query, we define “Research Community” based on the keywords mentioned in the PDF. However, in this case we use 30% of the papers published instead of 90% because we want to get enough results. Below is the result for the same.

```
community
['Application and Theory of Petri Nets', 'Multimedia Information Systems', 'AISCM', 'Conceptual Mode
ling', 'NLULP', 'LID', 'CL', 'CIKM', 'ICODW', 'VL', 'Rules in Database Systems', 'DAISD']
```

Secondly, now based on the output of the first, we search the top 100 most relevant papers using the PageRank algorithm. Below is the result for the same.

```
top_100
['Events in an Active Object-Oriented Database System.', 'Integrity Constraints Representation in Ob
ject-Oriented Databases.', 'The Design of an Expert System for Database Design.', 'Interfacing Prolo
g and Relational Data Base Management Systems.', 'Modeling and Enactment of Workflow Systems.', 'Bet
ter Termination Analysis for Active Databases.', 'Dimensions of Active Behaviour.', 'Informative and
```

Thirdly, now based on the output of second, we search to find the authors who have more than two papers in the top-100 identified papers and term them as “Gurus”. Below is the result for the same.

```
Answer
['Licia Sbattella', 'Arantza Illarramendi', 'Dimitris Tombros', 'Andreas Geppert', 'Elena Baralis',
'Stefano Paraboschi', 'Stefano Ceri', 'Arturo Jaime', 'Norman W. Paton', 'Oscar Díaz', 'Mokrane Bouz
eghoub', 'Letizia Tanca', 'Suzanne M. Embury', 'Peter M. D. Gray', 'Alvaro A. A. Fernandes', 'M. How
ard Williams', 'Jennifer Widom', 'Stella Gatziu', 'Klaus R. Dittrich', 'Piero Fraternali', 'Jack Cam
pin', 'Matilde Celma', 'Juan Carlos Casamayor', 'Hendrik Decker']
```