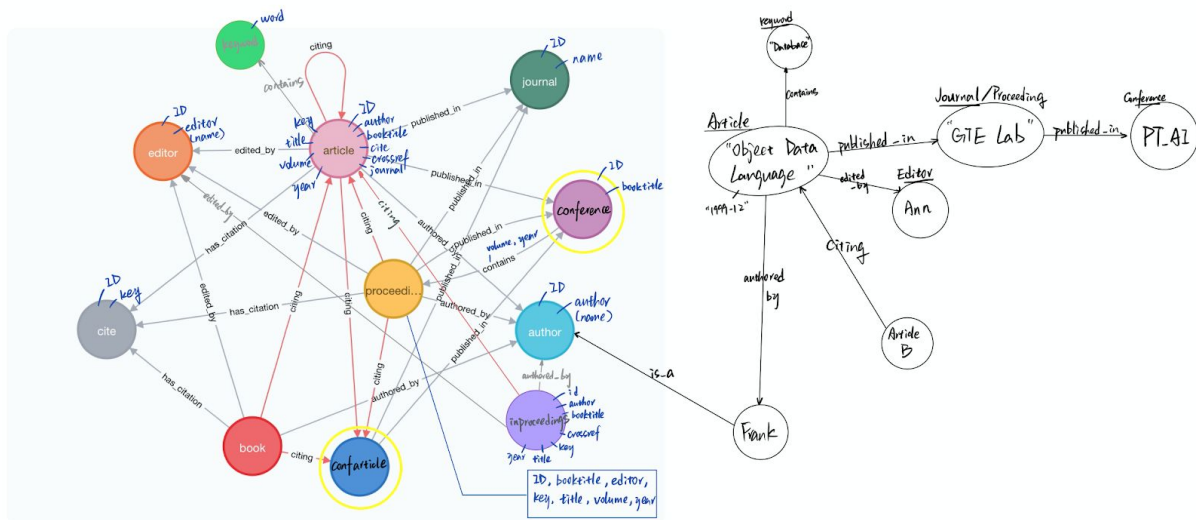


A.1 Modeling



Metadata

Data

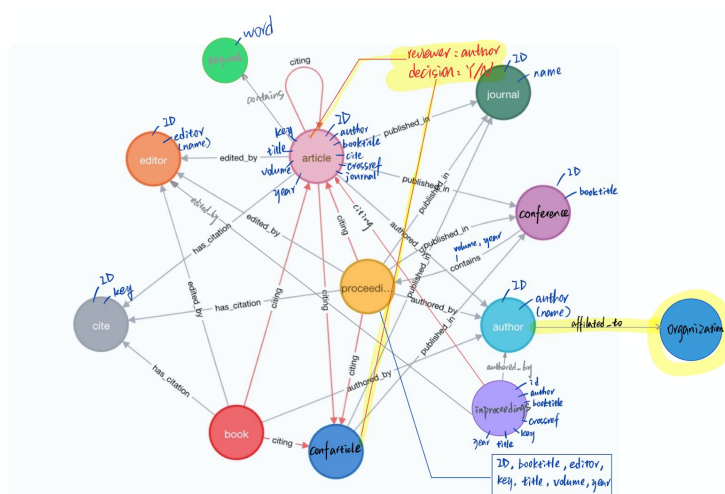
The base of the graph is followed by the default setting from `dblp`. Based on the query in Part B, general `Conference` nodes are derived from `Proceedings` where the edge `contains` has property of each `volume` and `year`. `confarticle` is derived from `inproceedings` with the same properties, but only cited works are extracted. Also, `citing` is created from `cite` to outperform the 'number of citation' aggregation.

A.2 Instantiating/Loading

The original database is parsed through the `dblp-to-csv` tool available in [GitHub](https://github.com/ThomHurks/dblp-to-csv). When passing the `--neo4j` option, the type annotations will be Neo4j compatible, and the tool generates a shell script called `neo4j_import.sh` that can be run to import the generated CSV files into a Neo4j graph database using the `neo4j-admin import` bulk importer tool.

Further changes are performed in Cypher and saved as `PartA.2_LiJin`.

A.3 Evolving the graph



We create the `reviewers` property for each article/inproceeding based on the community of the corresponding journal/conference (derived from PartB.3). Random 3 reviewers are assigned to each paper

(different from the author). `decision` on the review is also randomly assigned to each reviewer based on a `rand()` function in Cyther.

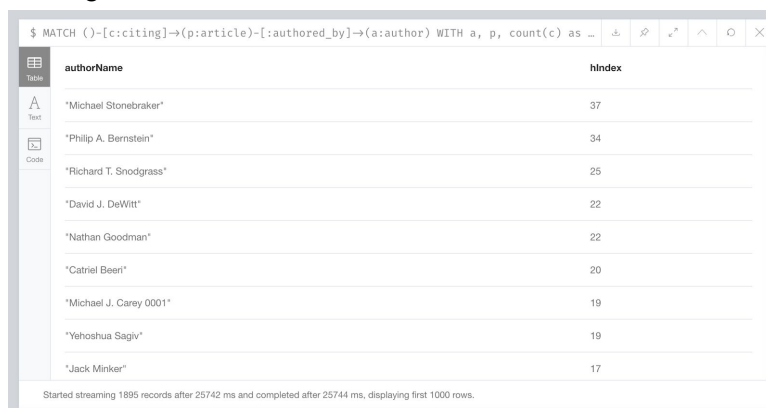
For the `affiliation`, given the missing raw data, it is randomly assigned for each author a specific university.

B Querying

1. H-index

```
MATCH ()-[c:citing]->(p:article)-[:authored_by]->(a:author)
WITH a, p, count(c) as numCit
ORDER BY numCit
WITH a, collect(numCit) AS NumCit
WITH a, NumCit, range(0, size(NumCit)) AS is
UNWIND is AS i
WITH a, NumCit[i] as numCit, i
WHERE (i+1) < numCit
RETURN distinct a.author as authorName, max(i) as hIndex
ORDER BY hIndex DESC
```

Testing result:



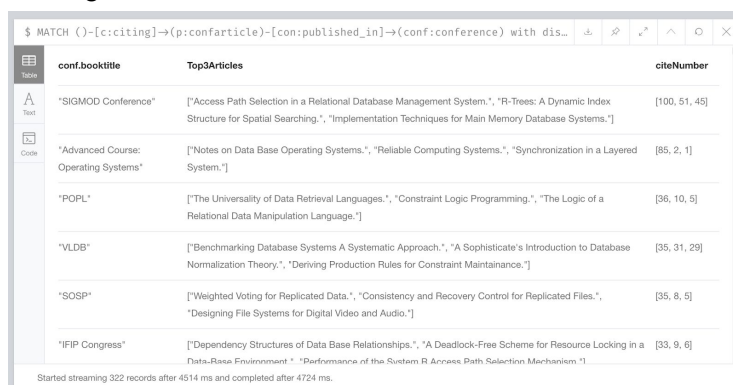
authorName	hindex
"Michael Stonebraker"	37
"Philip A. Bernstein"	34
"Richard T. Snodgrass"	25
"David J. DeWitt"	22
"Nathan Goodman"	22
"Catriel Beeri"	20
"Michael J. Carey 0001"	19
"Yehoshua Sagiv"	19
"Jack Minker"	17

Started streaming 1895 records after 25742 ms and completed after 25744 ms, displaying first 1000 rows.

2. Top 3 for each conference

```
MATCH
()-[c:citing]->(p:confarticle)-[con:published_in]->(conf:conference)
with distinct p, count(c) as citnum, conf ORDER BY citnum DESC
with conf, collect(p.title) as array_1, collect(citnum) as array_2
return distinct conf.booktitle, array_1[..3], array_2[..3]
```

Testing result:



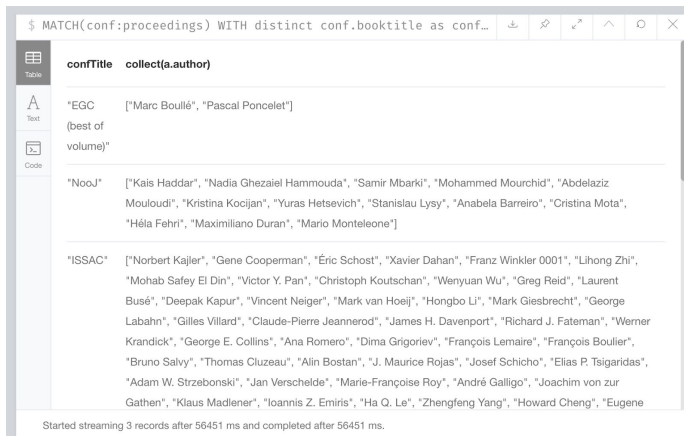
conf.booktitle	Top3Articles	citeNumber
"SIGMOD Conference"	["Access Path Selection in a Relational Database Management System.", "R-Trees: A Dynamic Index Structure for Spatial Searching.", "Implementation Techniques for Main Memory Database Systems."]	[100, 51, 45]
"Advanced Course: Operating Systems"	["Notes on Data Base Operating Systems.", "Reliable Computing Systems.", "Synchronization in a Layered System."]	[85, 2, 1]
"POPL"	["The Universality of Data Retrieval Languages.", "Constraint Logic Programming.", "The Logic of a Relational Data Manipulation Language."]	[36, 10, 5]
"VLDB"	["Benchmarking Database Systems A Systematic Approach.", "A Sophisticate's Introduction to Database Normalization Theory.", "Deriving Production Rules for Constraint Maintenance."]	[35, 31, 29]
"SOSP"	["Weighted Voting for Replicated Data.", "Consistency and Recovery Control for Replicated Files.", "Designing File Systems for Digital Video and Audio."]	[35, 8, 5]
"FIP Congress"	["Dependency Structures of Data Base Relationships.", "A Deadlock-Free Scheme for Resource Locking in a Data-Base Environment.", "Performance of the System R Access Path Selection Mechanism"]	[33, 9, 6]

Started streaming 322 records after 4514 ms and completed after 4724 ms.

3. Community for each Conference

```
MATCH(conf:proceedings)
WITH distinct conf.booktitle as confTitle #LIMIT 25
MATCH
(inp:inproceedings{booktitle:confTitle})-[:authored_by]->(a:author)
WITH distinct a, collect(distinct inp.key) as inps, confTitle
WHERE size(inps)>=4
RETURN confTitle, collect(a.author)
```

Testing result:



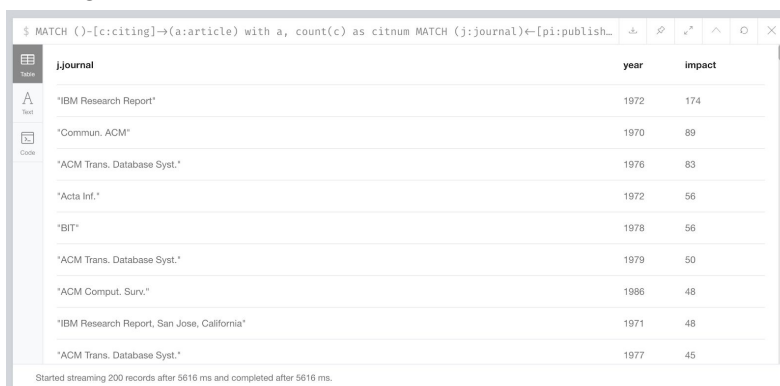
The screenshot shows a Cypher query result table with two columns: 'confTitle' and 'collect(a.author)'. The table contains three rows of data. The first row is for 'EGC (best of volume)' with authors '["Marc Boulié", "Pascal Poncelet"]'. The second row is for 'NooJ' with a long list of authors including '["Kais Haddar", "Nadia Ghezaiel Hammouda", "Samir Mbarki", "Mohammed Mourchid", "Abdelaziz Mouloudi", "Kristina Kocijan", "Yuras Hetsevich", "Stanislau Lysy", "Anabela Barreiro", "Cristina Mota", "Héla Fehri", "Maximiliano Duran", "Mario Monteleone"]'. The third row is for 'ISSAC' with another long list of authors including '["Norbert Kallier", "Gene Cooperman", "Éric Schost", "Xavier Dahan", "Franz Winkler 0001", "Lihong Zhi", "Mohab Safey El Din", "Victor Y. Pan", "Christoph Koutschan", "Wenyuan Wu", "Greg Reid", "Laurent Busé", "Deepak Kapur", "Vincent Neiger", "Mark van Hoeij", "Hongbo Li", "Mark Giesbrecht", "George Labahn", "Gilles Villard", "Claude-Pierre Jeannerod", "James H. Davenport", "Richard J. Fateman", "Werner Randick", "George E. Collins", "Ana Romero", "Dima Grigoriev", "François Lemaire", "François Boulier", "Bruno Salvy", "Thomas Cluzeau", "Alin Bostan", "J. Maurice Rojas", "Josef Schicho", "Elias P. Tsigaridas", "Adam W. Strzebonski", "Jan Verschelde", "Marie-Françoise Roy", "André Galligo", "Joachim von zur Gathen", "Klaus Madlener", "Ioannis Z. Emiris", "Ha Q. Le", "Zhengfeng Yang", "Howard Cheng", "Eugene"]'. At the bottom, it says 'Started streaming 3 records after 56451 ms and completed after 56451 ms.'

confTitle	collect(a.author)
"EGC (best of volume)"	["Marc Boulié", "Pascal Poncelet"]
"NooJ"	["Kais Haddar", "Nadia Ghezaiel Hammouda", "Samir Mbarki", "Mohammed Mourchid", "Abdelaziz Mouloudi", "Kristina Kocijan", "Yuras Hetsevich", "Stanislau Lysy", "Anabela Barreiro", "Cristina Mota", "Héla Fehri", "Maximiliano Duran", "Mario Monteleone"]
"ISSAC"	["Norbert Kallier", "Gene Cooperman", "Éric Schost", "Xavier Dahan", "Franz Winkler 0001", "Lihong Zhi", "Mohab Safey El Din", "Victor Y. Pan", "Christoph Koutschan", "Wenyuan Wu", "Greg Reid", "Laurent Busé", "Deepak Kapur", "Vincent Neiger", "Mark van Hoeij", "Hongbo Li", "Mark Giesbrecht", "George Labahn", "Gilles Villard", "Claude-Pierre Jeannerod", "James H. Davenport", "Richard J. Fateman", "Werner Randick", "George E. Collins", "Ana Romero", "Dima Grigoriev", "François Lemaire", "François Boulier", "Bruno Salvy", "Thomas Cluzeau", "Alin Bostan", "J. Maurice Rojas", "Josef Schicho", "Elias P. Tsigaridas", "Adam W. Strzebonski", "Jan Verschelde", "Marie-Françoise Roy", "André Galligo", "Joachim von zur Gathen", "Klaus Madlener", "Ioannis Z. Emiris", "Ha Q. Le", "Zhengfeng Yang", "Howard Cheng", "Eugene"]

4. Impact factor of journals

```
MATCH ()-[c:citing]->(a:article)
with a, count(c) as citnum
MATCH (j:journal)<-[pi:published_in]-(a:article)
with j, a.year as year, count(pi) as pnum, sum(citnum) as cnum
with j, collect(year) as Year, collect(pnum) as Pnum, collect(cnum) as Cnum
with j, Year, Pnum, Cnum, range(0, size(Year)-1) AS i
UNWIND i AS i
return j.journal, Year[i] as year, Cnum[i]/Pnum[i] as impact
order by impact desc, j.journal, year
```

Testing result:



The screenshot shows a Cypher query result table with three columns: 'j.journal', 'year', and 'impact'. The table contains ten rows of data. The first row is for 'IBM Research Report' with year 1972 and impact 174. The second row is for 'Commun. ACM' with year 1970 and impact 89. The third row is for 'ACM Trans. Database Syst.' with year 1976 and impact 83. The fourth row is for 'Acta Inf.' with year 1972 and impact 56. The fifth row is for 'BIT' with year 1978 and impact 56. The sixth row is for 'ACM Trans. Database Syst.' with year 1979 and impact 50. The seventh row is for 'ACM Comput. Surv.' with year 1986 and impact 48. The eighth row is for 'IBM Research Report, San Jose, California' with year 1971 and impact 48. The ninth row is for 'ACM Trans. Database Syst.' with year 1977 and impact 45. At the bottom, it says 'Started streaming 200 records after 5616 ms and completed after 5616 ms.'

j.journal	year	impact
"IBM Research Report"	1972	174
"Commun. ACM"	1970	89
"ACM Trans. Database Syst."	1976	83
"Acta Inf."	1972	56
"BIT"	1978	56
"ACM Trans. Database Syst."	1979	50
"ACM Comput. Surv."	1986	48
"IBM Research Report, San Jose, California"	1971	48
"ACM Trans. Database Syst."	1977	45

C Graph algorithms

1. Case Description

In order to realize the function of the page rank algorithm, based on the lowest level sub-graph can not only know the details of algorithms more clearly but also reduce the running time of large amounts of the dataset. So, paying attention to specific node and relationship with less amount is a good approach thus “book” and “school” node as well as “submitted at” relationship using in our lab.

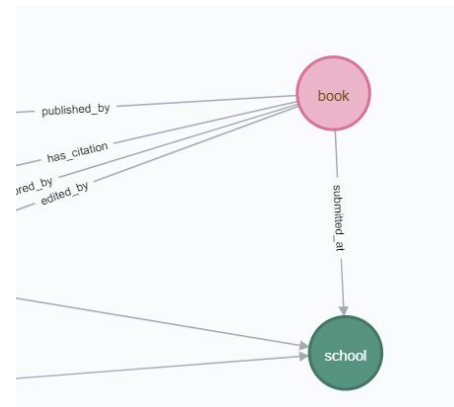


Figure C.1 Case schema for algorithm

2. Page Rank Algorithm

PageRank, page rank, also known as page rank or page rank, is a technique for ranking pages based on mutual hyperlinks between pages, named after the founder of Google, Larry Page. Google uses it to reflect the relevance and importance of web pages and is one of the effective indicators for evaluating web page optimization in search engine optimization operations. It determines the rank of a page through the vast network of hyperlink relationships. Google interprets the link from page A to page B as page A votes for page B. Google determines the new rank based on the rank of the voting source (or even the source of the source) and the voting target. Simply put, a high-level page can increase the level of other low-level pages.

Calling “page rank. stream” with Cypher project in two nodes, we get the result of the “book” node page rank score for its every “book id”.Figure C.2 Page Rank results1 illustrates that score-0.15000000000000002 for every “book id” means that every book has little and same incoming link in the relationship of “book” and “school” node due to the book id is an attribute with minimum level with little incoming relationships.

```
$ CALL algo.pageRank.stream( 'MATCH (b:book) WHERE exists( (b)-[:submitted_at]-() ) RETURN
```

node.book	score
3930	0.15000000000000002
4897	0.15000000000000002
12017	0.15000000000000002
21053	0.15000000000000002
21634	0.15000000000000002

Figure C.2 Page Rank results1

```
CALL algo.pageRank.stream(
  'MATCH (b:book) WHERE exists( (b)-[:submitted_at]-() ) RETURN id(b) as id',
  'MATCH (b:book)-[:submitted_at]-(s:school) RETURN id(b) as source, id(s) as target',
  {graph:'cypher'}
) YIELD nodeId,score with algo.asNode(nodeId) as node, score order by score desc
limit 10
RETURN node.book, score
```

Also, calling “alog.pageRank” function with the same nodes and relationship. iterations, loadMillis, computeMillis, writeMillis, dampingFactor, write and writeProperty shows the detail configuration and result of page rank algorithm.(Figure C.3 Page Rank results2)



nodes	iterations	loadMillis	computeMillis	writeMillis	dampingFactor	write	writeProperty
17825	1	56	1	30	0.85	true	'pagerank'

Figure C.3 Page Rank results2

```
CALL algo.pageRank(
  'MATCH (p:book) RETURN id(p) as id',
  'MATCH (p1:book)-[:submitted_at]->(p2:school) RETURN id(p1) as source, id(p2) as target',
  {graph:'cypher', iterations:1, write: true});
```

3. The Betweenness Centrality Algorithm

It is one of the measures for the centrality of the network graph based on the shortest path. For a fully connected network graph, where any two nodes have at least one shortest path, in a weightless network graph, the shortest path is the sum of the number of edges that the path contains, and in a weighted network graph, the shortest path is the weight of the path containing edges. Sum. The median centrality of each node is the number of times these shortest paths pass through the node. Betweenness Centrality has a wide range of applications in network theory: it represents the degree of interaction between a node and other nodes. For example, in a communication network, a node with higher median centrality has stronger control capabilities in the network, because more information will pass through the node when it is passed.

Needless to say, using the same nodes and relationships can identify the difference between centrality algorithm and page rank algorithm thus using school and nodes relationships. Unlike the previous one, based on the Cypher language as below, we check the Centrality of the school name with any kind of relationship in our schema. According to Figure 4.4 Betweenness Centrality result1, the Centrality 0 means there is no shortest path go via in these school name because they are low-level attribute and has no other links.

```
$ CALL algo.betweenness.stream( 'school', '', {direction:'out'}) YIELD nodeId, centrality MATCH (school:sc...
```

schoolname	centrality
"Aarhus University"	0.0
"University of Trier, Germany"	0.0
"University of California at Berkeley"	0.0
"Uwe Trier, FB 4, Informatik"	0.0
"U. C. Berkeley"	0.0
"Ludwig Maximilian University of Munich, Germany"	0.0

Figure 4.4 Betweenness Centrality result1

```
CALL algo.betweenness.stream( 'school','',{direction:'out'})
YIELD nodeId, centrality
MATCH (school:school) WHERE id(school) = nodeId
RETURN school.school AS schoolname,centrality
ORDER BY centrality DESC;
```

Figure 4.5 Betweenness Centrality result2 shows that the details results of algorithms.

```
$ CALL algo.betweenness('school','', {direction:'out',write:true, writeProperty:'centrality'}) YIELD nodes...
```

nodes	minCentrality	maxCentrality	sumCentrality	loadMillis	computeMillis	writeMillis
1858	-1.0	-1.0	-1.0	1354	3	44

Figure 4.5 Betweenness Centrality result2

```
CALL algo.betweenness('school','', {direction:'out',write:true,
writeProperty:'centrality'})
YIELD nodes, minCentrality, maxCentrality, sumCentrality, loadMillis,
computeMillis, writeMillis;
```

D Recommender

The `keywords` are derived from the paper title (tokenize & remove stop words) (details in `d_dataprocess`).

1 Define research community

```
MATCH (paper:paper {key:row.key})
MATCH (usedKeyword:keyword {keyword:keyword})
MERGE (paper)-[:HasKeyword]-(usedKeyword)
```

2 Conference/Journal with Community

```
MATCH (a:article)-[:published_in]->(conf)
WITH distinct conf, a, count(a) as noArticle
MATCH (key:keyword)-[hk:hasKeyword]->(a)
WITH distinct conf, key, count(hk) as noWithKeywords, noArticle
WHERE noWithKeywords/noArticle >= 0.9
CREATE (conf)-[:communityOf]->(key)
```

3 Top 100 papers of the conference/journal

```
CALL algo.pageRank.stream('Paper', 'Cite', {iterations:10,
dampingFactor:0.85})
YIELD nodeId, score
WITH algo.getNodeById(nodeId) AS paper, score
MATCH (book:Book)-[:PartOf]-(:Proceeding)-[:PublishedIn]-(paper)
ORDER BY score DESC
RETURN COLLECT(paper.title), score
Limit 100
```

4 Identify gurus

```
MATCH (a:article)-[:qualifiedReviewer]->(key:keyword)
WITH a, key
MATCH (a)-[ab:authored_by]->(b:author)
WITH a, b, count(ab) as weight
WHERE weight >= 2
RETURN a as Gurus
```