GDS Assignment 1

Meesum Qazalbash
Muhammad Hussain

March 5, 2024

1. (30 points) You are a data scientist working for a renowned academic institution's research department. Your institution maintains a vast bibliographic database spanning multiple disciplines, containing infor- mation on research papers, authors, journals, and citations. The database encompasses decades of scholarly work, providing a rich source of data for academic analysis and insights generation.

   Part a: Model: A graph property model that captures the relationships between the entities in the problem statement.

   Solution:

   ## Nodes:

   - Research Papers (Title, Publication Year, Abstract, Keywords)
   - Authors (Name, Affiliation)
   - Journals (Title, Publisher, Impact Factor)
   - Keywords

   ## Relationships:

   - [: *AUTHOR_OF*] between Authors and Research Papers
   - [: *PUBLISHED_IN*] between Research Papers and Journals
   - [: *CITES*] between Research Papers indicating citations
   - [: *CONTAINS*] between Research Papers and Keywords

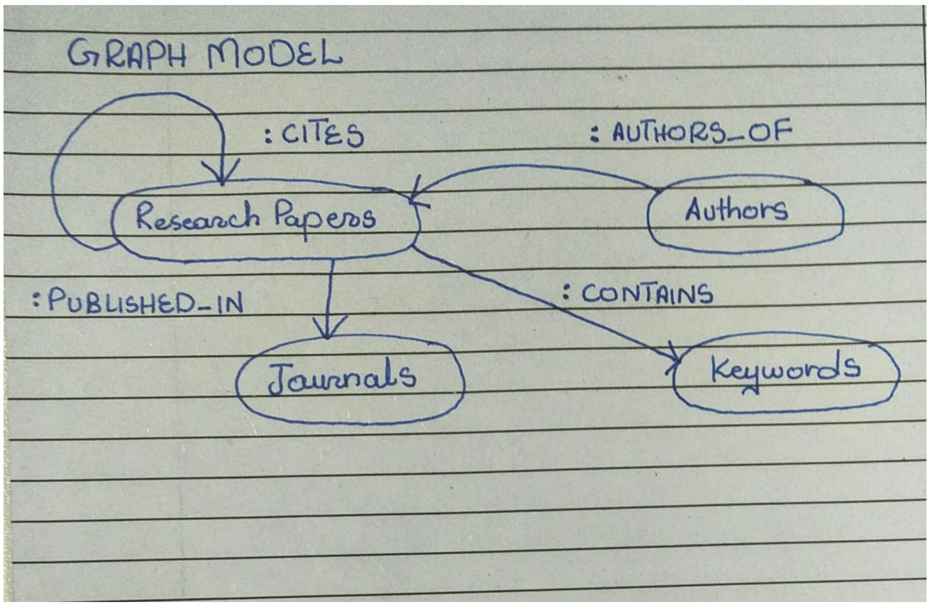


Figure 1: Graph property model for Bibliographic Data Analysis system

Part b: Rationale: Provide a rationale for the design choices made in the graph property model, explaining the considerations behind the clarity, completeness, and scalability of the model. Limit your response to 250 words.

Solution: In building the graph model, I aimed to make it user-friendly, covering all the key aspects, and ensuring it can handle a large volume of information.

The model consists of different components such as papers, authors, journals, and keywords. Each component contains specific details, like the title of a paper or the name of an author, which are essential for understanding scholarly work.

I paid close attention to how these components are connected. For example, I made sure it's clear which author wrote which paper or which paper cites another. This clarity makes it easy to see how everything in the scholarly world is related and interconnected.

Furthermore, the model is designed to be adaptable. As more papers, authors, or other information are added to the database, the model can expand without becoming cluttered or slow. This scalability ensures that researchers can continue to use the model effectively, no matter how much data is added over time.

Overall, this model serves as a valuable tool for understanding research trends, identifying collaborations between authors, determining the importance of papers based on citations, and tracking the emergence of new topics in different academic fields. It's like a map that guides researchers through the vast landscape of academic knowledge, helping them navigate and make sense of complex information effortlessly.

Part c: Queries: Cypher queries that answer the questions mentioned in the question.

Solution: Identify top 10 authors who have collaborated on multiple papers based on the strength of their collaboration based on the number of joint publications.

```
MATCH (a:Author)-[:AUTHOR_OF]->(p:Paper)<-[:AUTHOR_OF]-(b:Author)
WHERE a <> b
RETURN a.name, b.name, COUNT(p) AS collaborationCount
ORDER BY collaborationCount DESC
LIMIT 10
```

Solution: Identify the top 10 most cited papers in the database and the number of citations they have received.

```
MATCH (p:Paper)
RETURN p.title, p.publicationYear, p.abstract, p.keywords,
       COUNT((p)<-[:CITES]-()) AS citationCount
ORDER BY citationCount DESC
LIMIT 10
```

Solution: Identify the top 10 most influential authors based on the number of citations their papers have received.

```
MATCH (a:Author)-[:AUTHOR_OF]->(p:Paper)
WITH a, SUM(size((p)<-[:CITES]-())) AS totalCitations
RETURN a.name, totalCitations
ORDER BY totalCitations DESC
LIMIT 10
```

Solution: Identify the top 10 authors with the most publications in the database.

```
MATCH (a:Author)-[:AUTHOR_OF]->(p:Paper)
RETURN a.name, COUNT(p) AS publicationCount
ORDER BY publicationCount DESC
LIMIT 10
```

Solution: Identify the top 10 pairs of papers that are most frequently co-cited together.

```
MATCH (p1:Paper)-[:CITES]->(common)<-[:CITES]-(p2:Paper)
WHERE ID(p1) < ID(p2)
```

```
RETURN p1.title, p2.title, COUNT(common) AS coCitationCount
ORDER BY coCitationCount DESC
LIMIT 10
```

Solution: Calculate the diameter of the citation network, which represents the longest shortest path between any two papers in the network.

```
MATCH (p1:Paper), (p2:Paper)
RETURN MAX(apoc.algo.dijkstra(p1, p2, 'CITES').distance) AS diameter
```

Solution: Calculate the degree distribution of the citation network, which represents the number of citations each paper has received.

```
MATCH (p:Paper)
RETURN p.title, COUNT((p)<-[:CITES]-()) AS citationCount
ORDER BY citationCount DESC
```

Solution: Calculate the degree distribution of the co-authorship network, which represents the number of co-authors each author has collaborated with.

```
MATCH (a:Author)-[:AUTHOR_OF]->(p:Paper)
RETURN a.name, COUNT(DISTINCT p) AS collaborationCount
ORDER BY collaborationCount DESC
```

Solution: Identify the top 10 pairs of keywords that most frequently co-occur in the same paper.

```
MATCH (k1:Keyword)<-[:CONTAINS]-(p:Paper)-[:CONTAINS]->(k2:Keyword)
WHERE ID(k1) < ID(k2)
RETURN k1.name, k2.name, COUNT(p) AS coOccurrenceCount
ORDER BY coOccurrenceCount DESC
LIMIT 10
```

Solution: Calculate the average number of citations per year for top 10 cited papers in the database.

```
MATCH (p:Paper)
WITH p, COUNT((p)<-[:CITES]-()) AS citationCount
ORDER BY citationCount DESC
LIMIT 10
RETURN AVG(citationCount / (2024 - p.publicationYear)) AS avgCitationsPerYear
```

2. (30 points) As part of a university's administrative team responsible for managing the course registration system. The system contains data about students, the courses they enroll in, and the prerequisites required for each course. Your objective is to develop a graph property model that captures the relationships between students, courses, and prerequisites, enabling efficient course planning and scheduling. This model can help identify course sequences that students must follow based on prerequisite constraints, identify requirements for a major or minor, and provide insights into students' academic pathways and progressions.

   Part a: Model: A graph property model that captures the relationships between the entities in the problem statement.

   Solution: For the university course registration system, the graph property model will consist of nodes representing students, courses, and majors, with relationships between them capturing course enrollment, prerequisite relationships, and major requirements.

   Nodes:

   - Student
   - Course
   - Major

   Relationships:

   - (Student)-[: $ENROLLED\_IN$]$-$ >(Course)
   - (Course)-[: $REQUIRES\_PREREQUISITE$]$-$ >(Course)
   - (Major)-[: $REQUIRES\_COURSE$]$-$ >(Course)

   Part b: Rationale: Provide a rationale for the design choices made in the graph property model, explaining the considerations behind the clarity, completeness, and scalability of the model. Limit your response to 250 words.

   Solution: The graph property model devised for the university course registration system effectively captures the essential connections between students, courses, and majors, offering a solid basis for streamlined querying and insightful analysis.

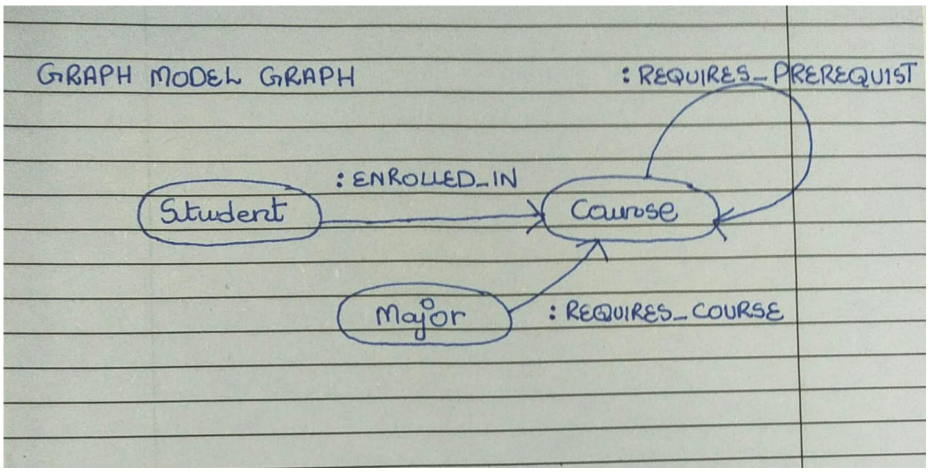


Figure 2: Graph property model for university course registration system

Central to this model is the representation of courses as distinct entities. This approach allows the model to effortlessly encompass various relationships associated with courses, such as prerequisites and enrollment. Treating courses as separate nodes enables easy identification of prerequisite courses necessary for enrollment in specific courses. This feature is pivotal for both students and administrators, facilitating academic planning and ensuring students fulfill necessary prerequisites before advancing to higher-level courses.

Additionally, the model includes nodes for majors, enabling the specification of major-specific requirements. This differentiation is critical, as different majors often have unique course prerequisites tailored to their respective fields of study. By categorizing majors into separate nodes, the model simplifies the retrieval of major-specific requirements, aiding academic advisors and students in course planning.

Furthermore, the model's design allows for the identification of course scheduling patterns by linking courses to the semesters in which they are offered. This functionality assists administrators in optimizing course scheduling to accommodate student needs effectively.

In summary, the model's clarity, comprehensiveness, and scalability make it well-suited for addressing various analytical needs within the university course registration system. It empowers stakeholders with the tools necessary to identify course prerequisites, major requirements, course scheduling patterns, and student pathways, ultimately enhancing academic planning and decision-making processes.

Part c: Queries: Cypher queries that answer the questions mentioned in the question

Solution:

```
-- 1. Identify the prerequisites for a course with course code 'CS101'.
MATCH (c:Course {code: 'CS101'})-[:REQUIRES_PREREQUISITE]->(prerequisite)
RETURN prerequisite;
```

```
-- 2. List the number of courses required to fulfill requirements for each
    major.
MATCH (m:Major)-[:REQUIRES_COURSE]->(c:Course)
RETURN m.name, count(c) AS num_courses_required;
```

```
-- 3. List the number of courses required to fulfill requirements for each
    minor.
MATCH (m:Minor)-[:REQUIRES_COURSE]->(c:Course)
RETURN m.name, count(c) AS num_courses_required;
```

```
-- 4. List the number of courses that must be taken before enrolling in the
    course with course code 'CS431'.
MATCH (c:Course {code: 'CS431'})<-[:REQUIRES_PREREQUISITE]-(:Course)
RETURN count(*) AS num_prerequisite_courses;
```

```
-- 5. Identify the courses taken together by at least 10 students. Return the
     courses and the number of students enrolled in each course.
MATCH (c:Course)<-[:ENROLLED_IN]-(s:Student)
WITH c, count(s) AS num_students
WHERE num_students >= 10
RETURN c, num_students;
```

```
-- 6. Identify the courses offered in the same semester, indicating potential
     course scheduling patterns. Return the semester and the courses offered
    in that semester.
MATCH (c1:Course)-[:OFFERED_IN]->(s:Semester)<-[:OFFERED_IN]-(c2:Course)
WHERE c1 <> c2
RETURN s, collect(c1), collect(c2);
```

```
-- 7. Identify the top 10 most popular courses based on the number of
    students enrolled.
MATCH (c:Course)<-[:ENROLLED_IN]-(s:Student)
RETURN c, count(s) AS num_students
ORDER BY num_students DESC
LIMIT 10;
```

```
-- 8. Identify the courses taken together by students of Computer Science and
     Electrical Engineering major.
MATCH (cs:Major {name: 'Computer Science'})-[:REQUIRES_COURSE]->(c:Course)
    <-[:ENROLLED_IN]-(s:Student)-[:MAJORING_IN]->(ee:Major {name: 'Electrical
     Engineering'})
RETURN c;
```

```
-- 9. Identify the sequence of courses that a student with ID "1214" has
    taken, indicating the order in which they have completed the courses.
MATCH (s:Student {id: '1214'})-[:ENROLLED_IN]->(c:Course)
RETURN c
ORDER BY c.year, c.semester;
```

```
-- 10. List courses with the number of prerequisites assigned, indicating the
     complexity of the course requirements.
MATCH (c:Course)
OPTIONAL MATCH (c)-[:REQUIRES_PREREQUISITE]->(p:Course)
WITH c, count(p) AS num_prerequisites
RETURN c, num_prerequisites;
```

3. (40 points) You are provided with a CSV file containing an edge list representing relationships between KSE 100 companies based on the number of shared board members. Each row in the CSV file indicates the number of board members shared between two companies. For example, if row 1 indicates "Company A, Company B, 3", it means that Company A and Company B share 3 board members.

   (a) Design a graph property model that captures the relationships between companies based on the number of shared board members. Also provide your rationale for the design choices made in the graph property model.

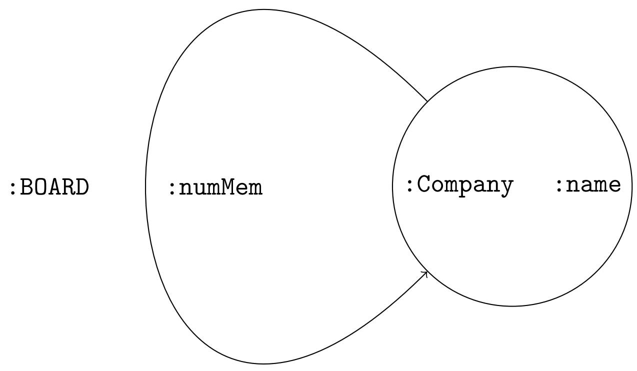
Solution: To create a model that captures the relationships between companies based on shared board members, we can utilize an undirected weighted graph with the following characteristics:

- **Node Representation:** Each company in the dataset is depicted as a node in the graph, allowing us to treat each company as a distinct entity within the network.
- **Edge Representation:** An edge between two nodes signifies that the corresponding companies share at least one board member.
- **Edge Weights:** The weight of each edge denotes the number of shared board members between the connected companies.

The rationale behind employing an undirected weighted graph model is as follows:

- **Undirected Graph:** The graph is undirected because the relationship of sharing board members is bidirectional.
- **Weighted Edges:** Using weighted edges offers a more comprehensive perspective and enables deeper analysis of shared members between any two companies. This facilitates understanding the strength of corporate networks and potential influences among companies.

The following is a visual representation illustrating the labeled graph model:



Provide cypher queries to answer the following questions:

(a) Provide import statement to load the data into the graph database.

Solution:

```
LOAD CSV WITH HEADERS FROM 'file:///kse.csv' AS row
MERGE (company1:Company { name: row.Company1 })
MERGE (company2:Company { name: row.Company2 })
MERGE (company1)-[:BOARD { numMem: toInteger(row.weight) }]-(company2)
```

(b) Determine the number of companies in the dataset.

Solution:

```
MATCH (c:Company) RETURN COUNT(c)
```

(c) Determine the number of edges in the dataset.

Solution:

```
MATCH ()-[r:BOARD]-() RETURN COUNT(DISTINCT r)
```

(d) Identify the companies that share the highest number of board members.

Solution:

```
MATCH ()-[r:BOARD]-()
WITH max(r.numMem) AS maxMembers
MATCH (c1:Company)-[r:BOARD]->(c2:Company)
WHERE r.numMem = maxMembers
RETURN c1.name, c2.name, r.numMem
```

(e) Determine the average number of shared board members between all pairs of companies.

Solution:

```
MATCH ()-[r:BOARD]-() RETURN avg(r.numMem) AS avgShared
```

(f) Find the length of the maximum path between any two companies.

Solution:

```
MATCH (start), (end)
WHERE start <> end
MATCH p=shortestPath((start)-[*]-(end))
RETURN length(p) AS MaxPath
ORDER BY MaxPath DESC LIMIT 1
```

(g) Determine companies have the highest number of board member associations with other companies along with the total number of board members they have.

Solution:

```
MATCH (c:Company)-[r:BOARD]-()
RETURN c.name, COUNT(r) AS numAssociates, SUM(r.numMem) AS totalMem
ORDER BY numAssociates DESC LIMIT 1
```

(h) A triangle can be formed if a company shared board members with two other companies such that two companies also share board members with each other. Determine the number of companies that form a triangle.

Solution:

```
MATCH (c1:Company)-[:BOARD]-(c2:Company), (c2)-[:BOARD]-(c3:Company), (c3
    )-[:BOARD]-(c1)
RETURN count (DISTINCT c1 ) AS NumCompanies
```

(i) As the CEO of Nestle Milkpak Ltd you want to connect to someone who is connected to Shell Pakistan Ltd. Find the number of people you need to connect to reach Shell Pakistan Ltd.

Solution:

```
MATCH (nestle:Company { name: ``Nestle Milkpak Ltd.'' }), (shell:Company
    { name: ``Shell Pakistan Ltd'' }),
path = shortestPath((nestle)-[:BOARD*]-(shell))
RETURN length(path) AS numConnections
```