**Title**: University Research Graph Database for Efficient Multi-Perspective Data Analysis using Neo4j

1. Research issue:

Our project centers on tackling the core challenge of optimizing the management and utilization of intricate relationships within data, particularly within the domain of large and complex structural datasets. The aim is to overcome existing limitations and enhance the efficiency of handling complex, multi-perspective, and abstract data analyses that traditional relational databases struggle with. Notably, relational databases, while optimized for transaction processing, exhibit reduced speed and effectiveness in the realm of in-depth research data analysis, where intricate relationships and abundant connections between various data points are commonplace.

**Our Contributions:**

Our team played a crucial role in tackling the previously mentioned research challenge by leading the creation of a specialized graph database tailored for handling university research data. We primarily contributed by performing a literature survey on overcoming the limitations of relations database, assessing the performance of the graph database in contrast to the relational database over different aspects like querying, traversal and joins.

In our project, the collaborative synergy within the team played a crucial role in uncovering significant discoveries. Each team member contributed substantially to the project's advancement. Aishwarya Vinod Menon and Mehul Jain spearheaded the collection, refinement of the dataset, and formulated the queries to be addressed by both models. They also conducted comparative analyses based on execution times between the two types. Vidya Sreekumar and Dibyanshi Singh were responsible for crafting the graph data model and its implementation in Neo4j using Cypher queries. Swasha Kumar and Trishala Reddy led the design of the relational schema and executed the relational database. Collectively, the entire group engaged in dynamic discussions, contributing to the conceptualization and workflow implementation. Moreover, our extensive literature review encompassing both models was a collaborative effort that greatly influenced the comprehensive final report, consolidating our collective findings.

1. Survey related work:
2. Technical Section:

Webscraping, Database formulation, MySQL, Neo4J,

Programming Language:

Python: Python is a versatile programming language, which has been employed for tasks ranging from web scraping to calculating the execution time for database operation.

Sql: SQL is a standardized language for managing and manipulating relational databases.

Cypher: Cypher is a query language specifically designed for interacting with Neo4j, a popular graph database. It is used to concisely represent complicated graph patterns and facilitate the rapid retrieval, manipulation, and analysis of graph data by users.

**Web Scraping :**

Employing Python's Beautiful Soup library, we executed a meticulous web scraping procedure on the Google Scholar website to extract details like authors’ names, article titles, and associated article IDs. This process entailed meticulous navigation through the website's HTML structure, diligently targeting and isolating specific HTML elements housing the required information. Leveraging the powerful parsing capabilities of Beautiful Soup, we efficiently retrieved and compiled the desired content. Upon completion of data extraction, we organized the gathered information into a structured format, employing a DataFrame in Python for streamlined management and manipulation. Subsequently, to ensure the preservation and accessibility of this acquired data, we adeptly exported the DataFrame contents into a CSV file format. This CSV file served as an efficient repository, facilitating easy integration into diverse analytical tools or databases for subsequent in-depth analysis, insights derivation, or seamless utilization in research endeavors or applications.

We meticulously parsed and separated the author names from the journal names, for further processing. To ensure data integrity, we systematically eliminated noisy or irrelevant data that might have obscured the dataset's clarity or precision.

**Database Formulation:**

We structured the refined data into distinct datasets, segregating information into different tables and handled null values. to create distinct identifiers for researchers, affiliations, DB Indexer, co-authors, co-leads, and publishers contained within the dataset. Subsequently, we manually inserted these identifiers into the respective database tables. This method allowed us to establish a structured and well-linked database, enabling efficient data retrieval and analysis across various facets of the project.

| No | Tables | Description |
| --- | --- | --- |
| 1. | Article | Articles are research projects that have been published. |
| 2. | Research Project | Includes all research projects which are undertaken by a researcher(s) |
| 3. | Researcher | Researcher refers to a person who authors or co-authors an article or leads or co-leads a research Project |
| 4. | Affiliation | Affiliation refers to the university to which the researchers are affiliated with |
| 5. | Publisher | Publisher refers to the publishing organisations that publish various articles |
| 6. | DB Indexer | DB indexer refers to the database indexing organisations that provide indexing services |

**Importing CSV files into MYSQL :**

In the process of incorporating CSv files to MySQL, we created tables that mirrored the structure of the CSV files which included the following steps of specifying column names, data types and constraints.

Subsequently, we imported the CSv file into the MySQl database ensuring that the insertion of data was seamless.

FOllowing the data import, comprehensive validation checks were meticulously conducted to guarantee the accuracy and integrity of the data within MySQL database.

**Loading CSV into Neo4j**:

With precision, the CSV file was loaded into Neo4j utilizing Cypher's "Load CSV" command. Subsequently, a comprehensive validation process was conducted to verify the alignment of each column with its respective entity within the graph. This ensured the establishment of a cohesive relationship between the various data sources incorporated into the database.

**Formulating Queries:**

Based on the structure of the database models of MySQL and Neo4j and the relationships between the entities we formulated queries and subsequently tested these queries to ensure that these queries give the desired results.

**Connecting MySQL and Neo4j with Python:**

We establish a connection between Python and a MySQL database by utilizing the mysql library. This library acts as a bridge, enabling seamless communication between Python and the MySQL database, allowing for the execution of SQL queries directly from Python scripts.

We utilize the Neo4j library to establish a connection between Python and Neo4j graph database. Neo4j library has comprehensive functionalities like managing nodes, relationships, and the graph structure.

To assess the time taken for execution of these queries, we employ the time module. Before executing the SQL query, we mark the start time, and upon its completion, we measure the elapsed time. This meticulous time-tracking methodology provides insights into the query's performance, aiding in the evaluation and optimization of SQL operations within the database.

**Leveraging Neo4j**

Neo4j excels at managing complex relationships and connected data. University research data often involve intricate relationships between researchers, departments, projects, publications, etc. Neo4j's graph model can represent these relationships natively and traverse them efficiently. In a relational database, these relationships would typically require multiple tables with foreign keys and join operations, which can become complex and less performant as the size and depth of relationships grow.

Due to its graph structure and index-free adjacency, Neo4j allows for efficient retrieval of connected data. This means that queries to find connections between researchers, projects, and publications can be executed quickly, regardless of the size of the data set. University research data is often dynamic, with new projects, researchers, and collaborations emerging regularly. Neo4j's schema-optional approach allows for easy adaptation and evolution of the database schema without extensive migration processes.SQL most of the times has a fixed schema that requires migrations to handle structural changes, which can be cumbersome and time-consuming. Neo4j uses Cypher, an expressive and powerful query language designed for graph traversals, making it easier to write and understand queries involving deep relationships.It also maintains consistent performance for relationship-driven queries due to index-free adjacency, meaning each node knows about its neighbors without the need for index lookups.Whereas, Performance can degrade with complex joins, especially as the dataset grows larger, leading to longer query times and potential bottlenecks. Neo4j has a strong community focused on graph databases, with plenty of libraries, tools, and extensions specifically designed for graph-related problems.On the other hand, SQL has a vast and mature ecosystem, with a wealth of tools and community knowledge that can be leveraged for a variety of applications, not just those that are relationship-centric. In summary, for a university research graph database project that requires handling complex relationships with high performance and flexible schema adaptations, Neo4j provides distinct advantages over traditional SQL databases. Its graph-centric approach and efficient handling of connected data make it a better fit for this specific project's requirements.

**Relational and graph:**

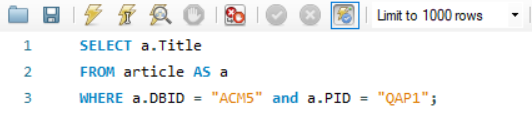
Relational and graph databases represent divergent models in the database landscape, each tailored to address nuanced data management needs. SQL queries, exemplifying relational databases, adhere to the structured, tabular format inherent in this model. This paradigm excels in scenarios where data relationships can be effectively represented through well-defined tables, and the emphasis lies on querying and manipulating data with established, predetermined schemas. Conversely, the Neo4j query embodies the principles of a graph database, which prioritizes the representation of complex relationships within the data. In the Neo4j graph model, data entities are nodes, and relationships between them are fundamental and intrinsic to the data representation, enabling efficient traversal and retrieval of interconnected data. This query's focus on articles and research projects underscores the graph database's strength in handling intricate, highly connected datasets. Graph databases shine in scenarios where relationships are dynamic, varied, and central to data analysis. The choice between relational and graph databases hinges on the inherent nature of the data and the specific analytical requirements, with relational databases excelling in structured data scenarios and graph databases providing a robust solution for managing intricate and dynamic relationships within the data model.

In the evaluation section:

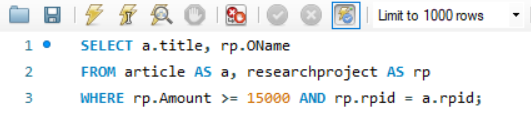
Python, linking with python, alphabetical queries,, ex of sql and cypher query along with their sample outputs.

MySQLqueries

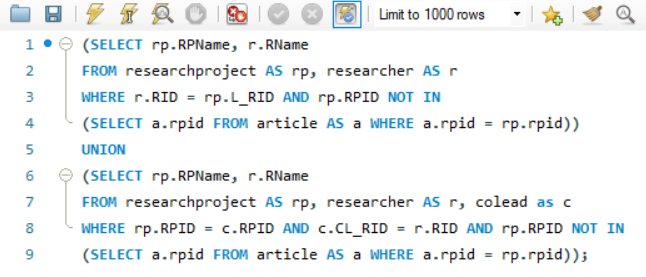
1. Retrieve article title from article table which have DBID=”ACH5” & PID=”QAP1”



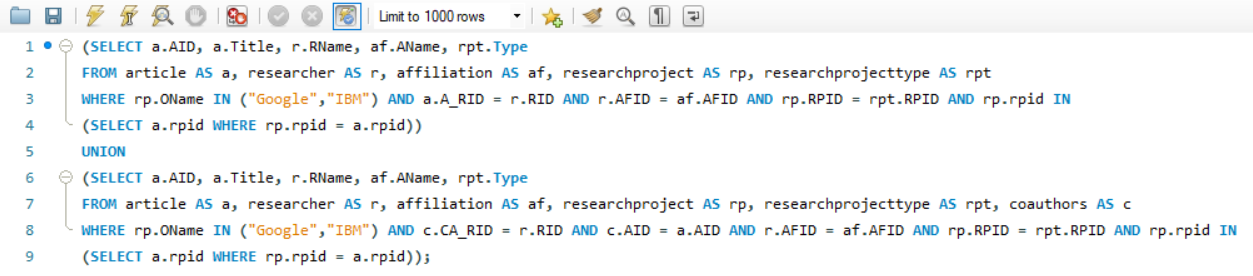
1. Retrieve all the article names that has a funding of >15000 along with their organisation name



1. Retrieve all the research projects name and their authors name and co-leads name which have yet not been published.

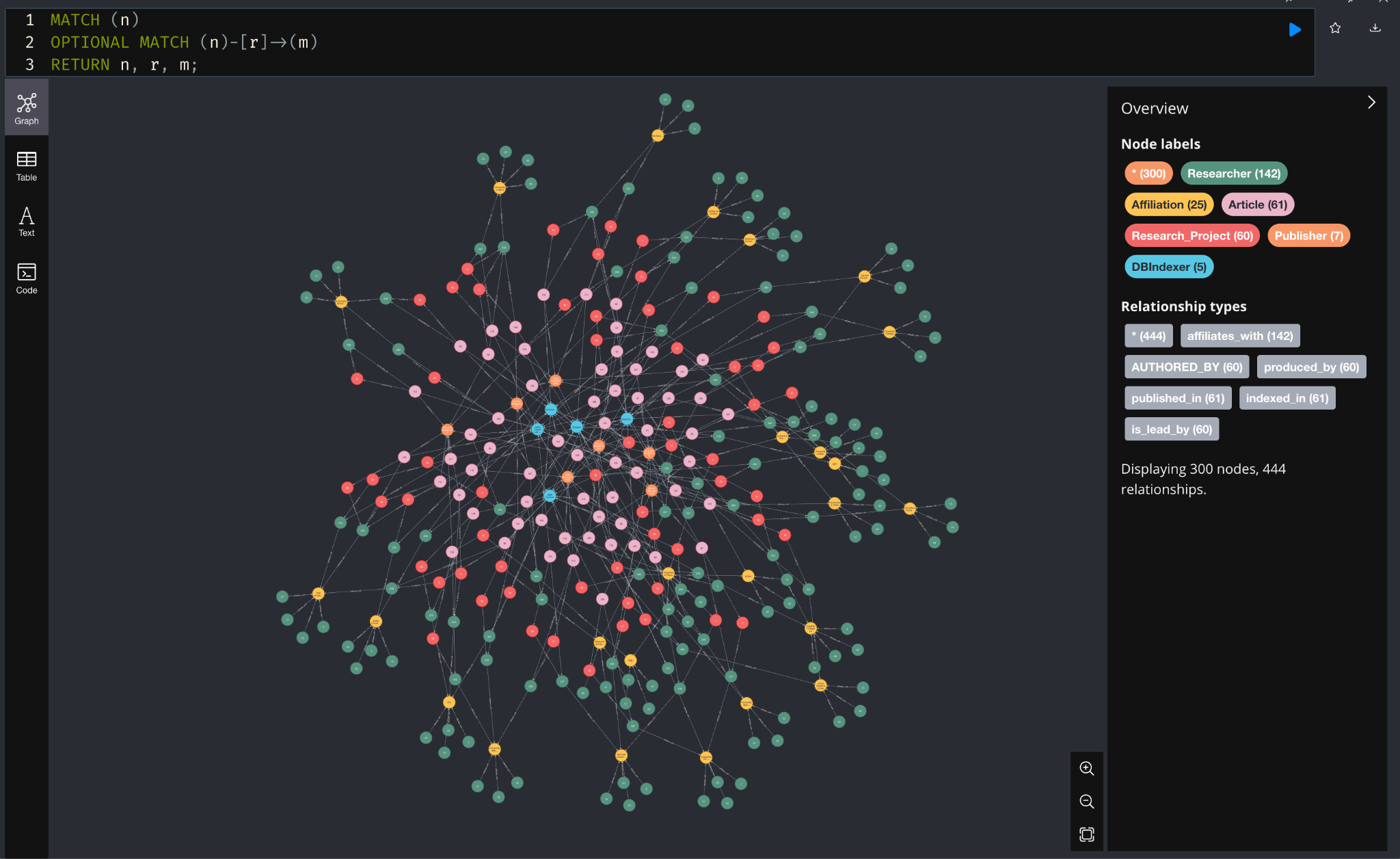


1. Retrieve all the articles name and id, their author’s and coauthor’s name,researcher’s affiliation name & research project type who have their funding from”Google,IBM”

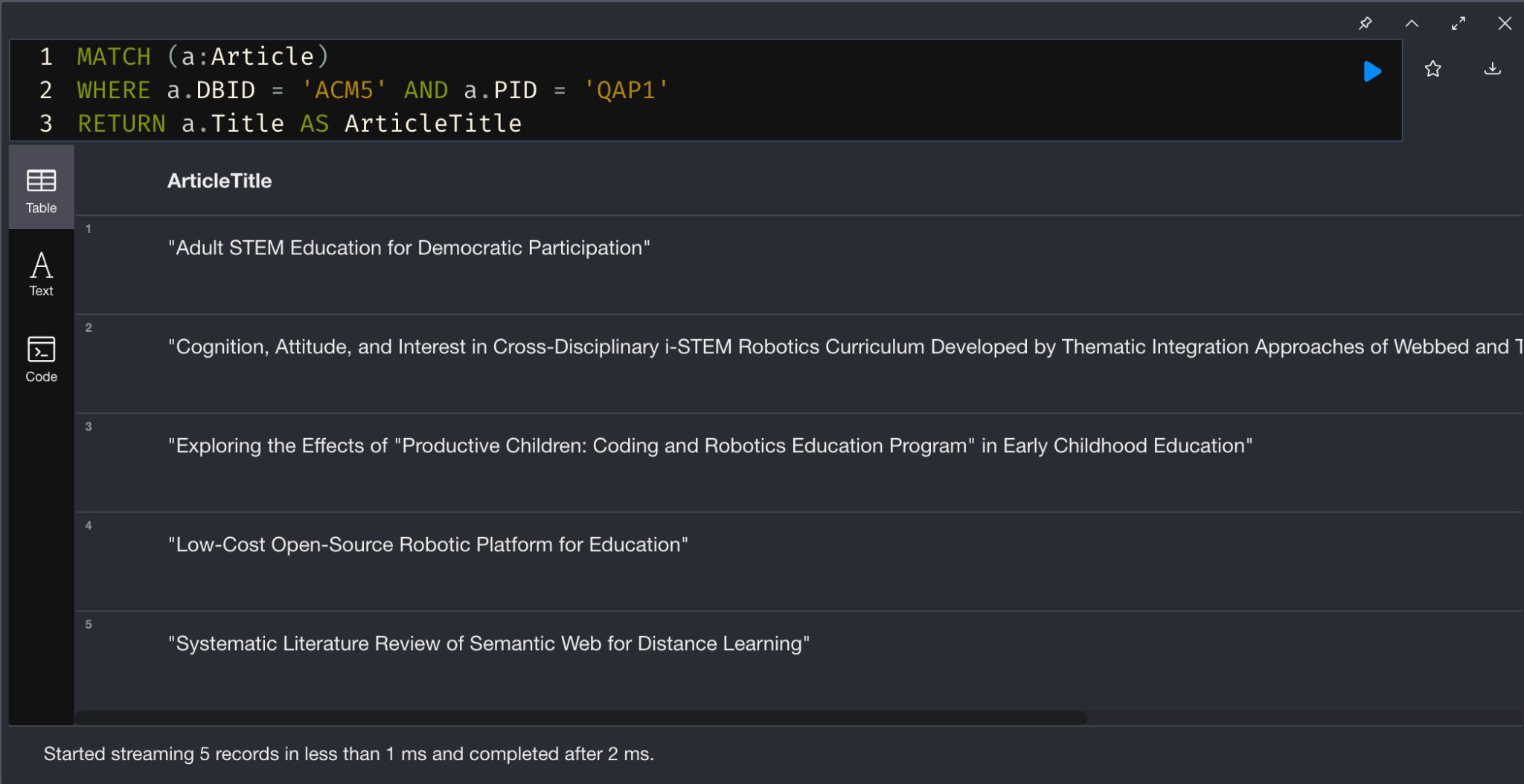
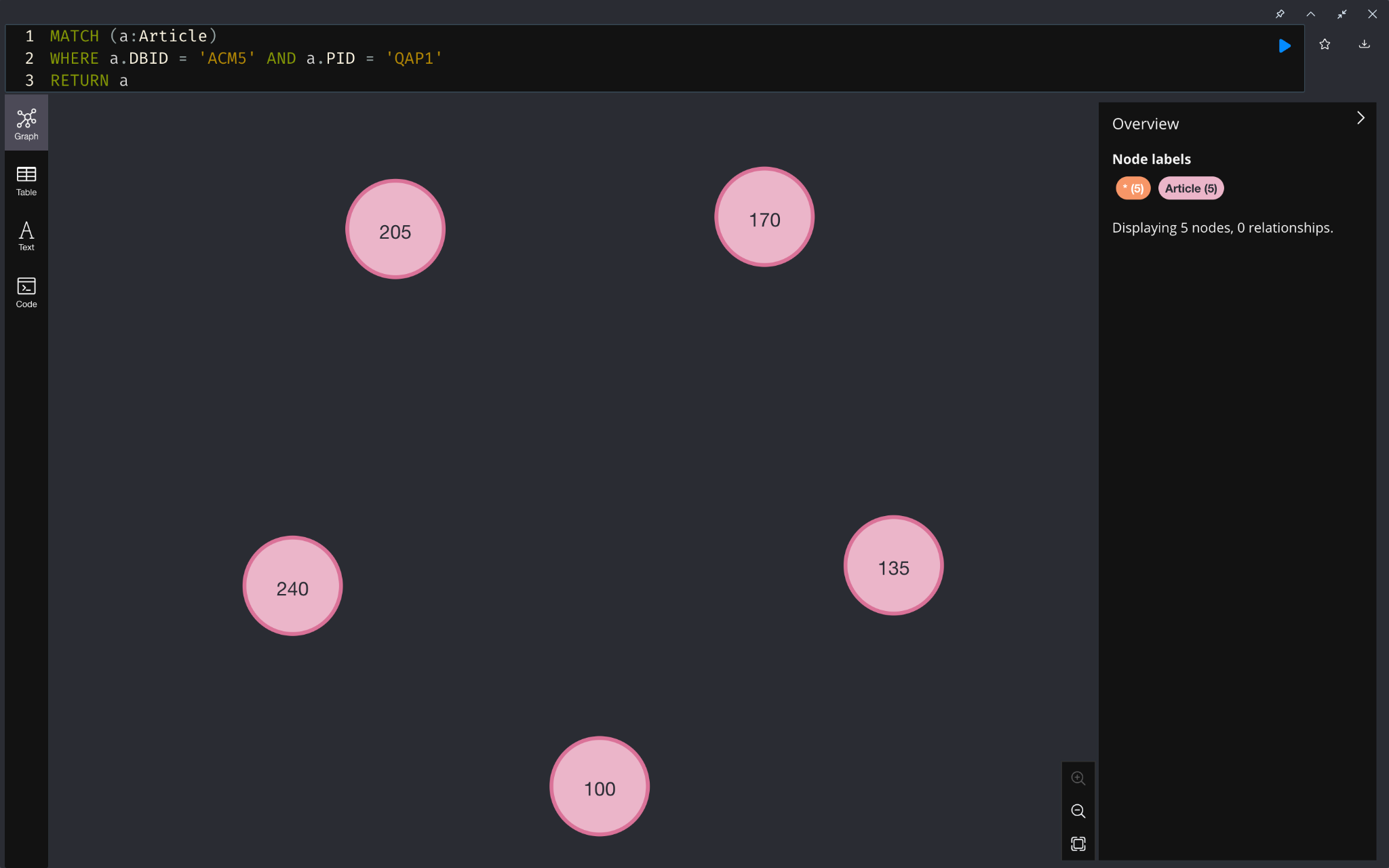


Neo4j queries and result-table with graphs

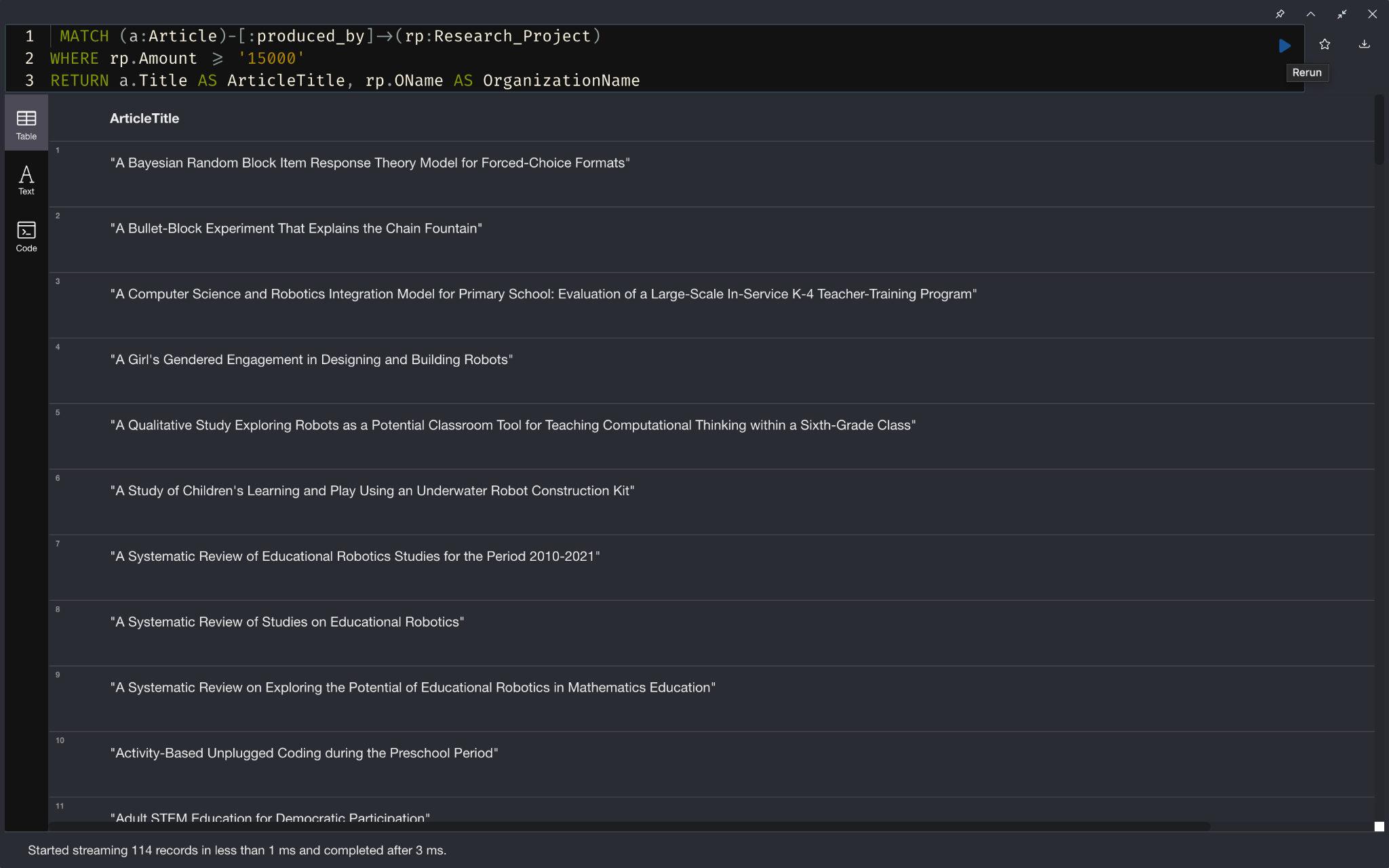
1.Complete view of University Research Graph Database

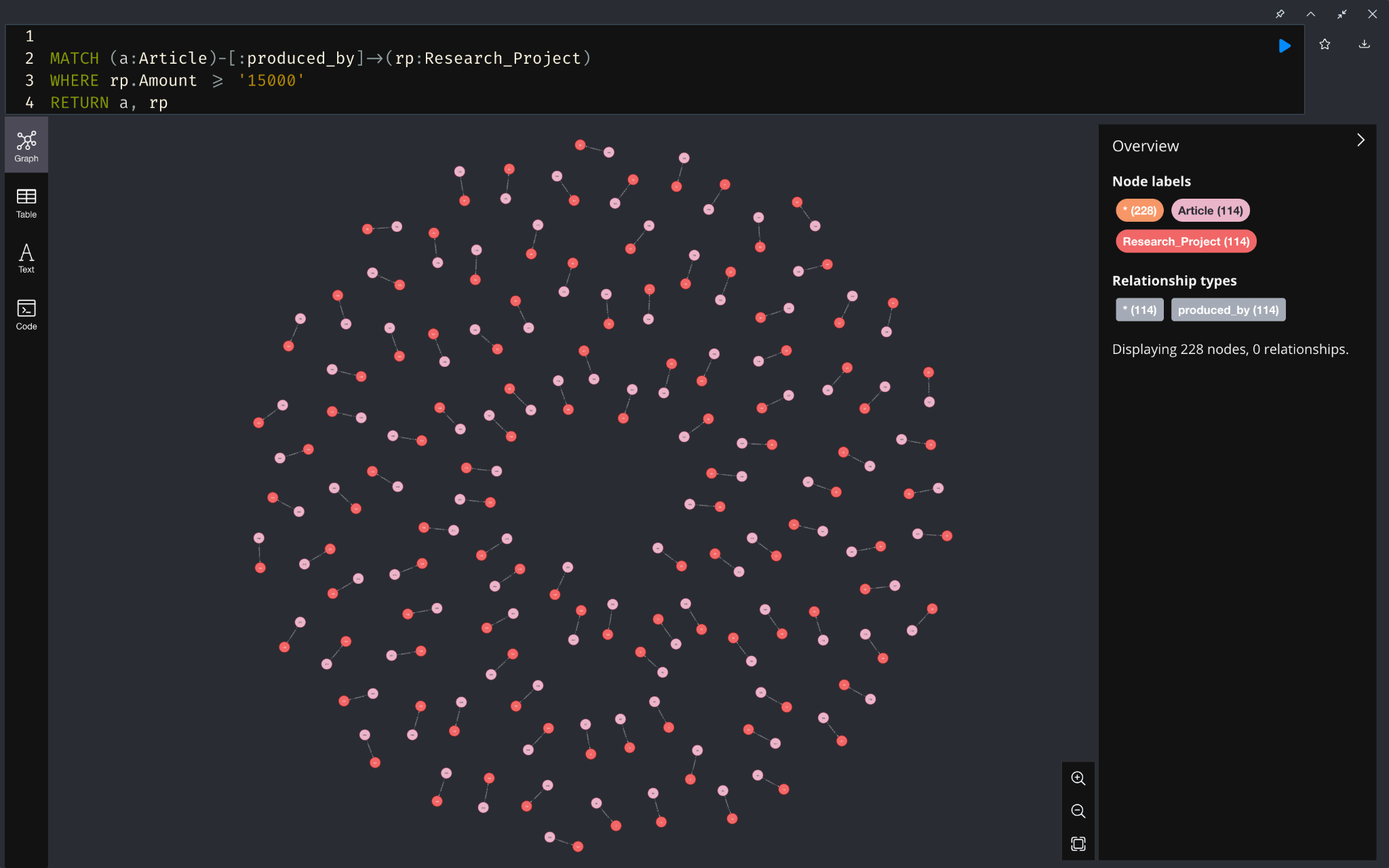


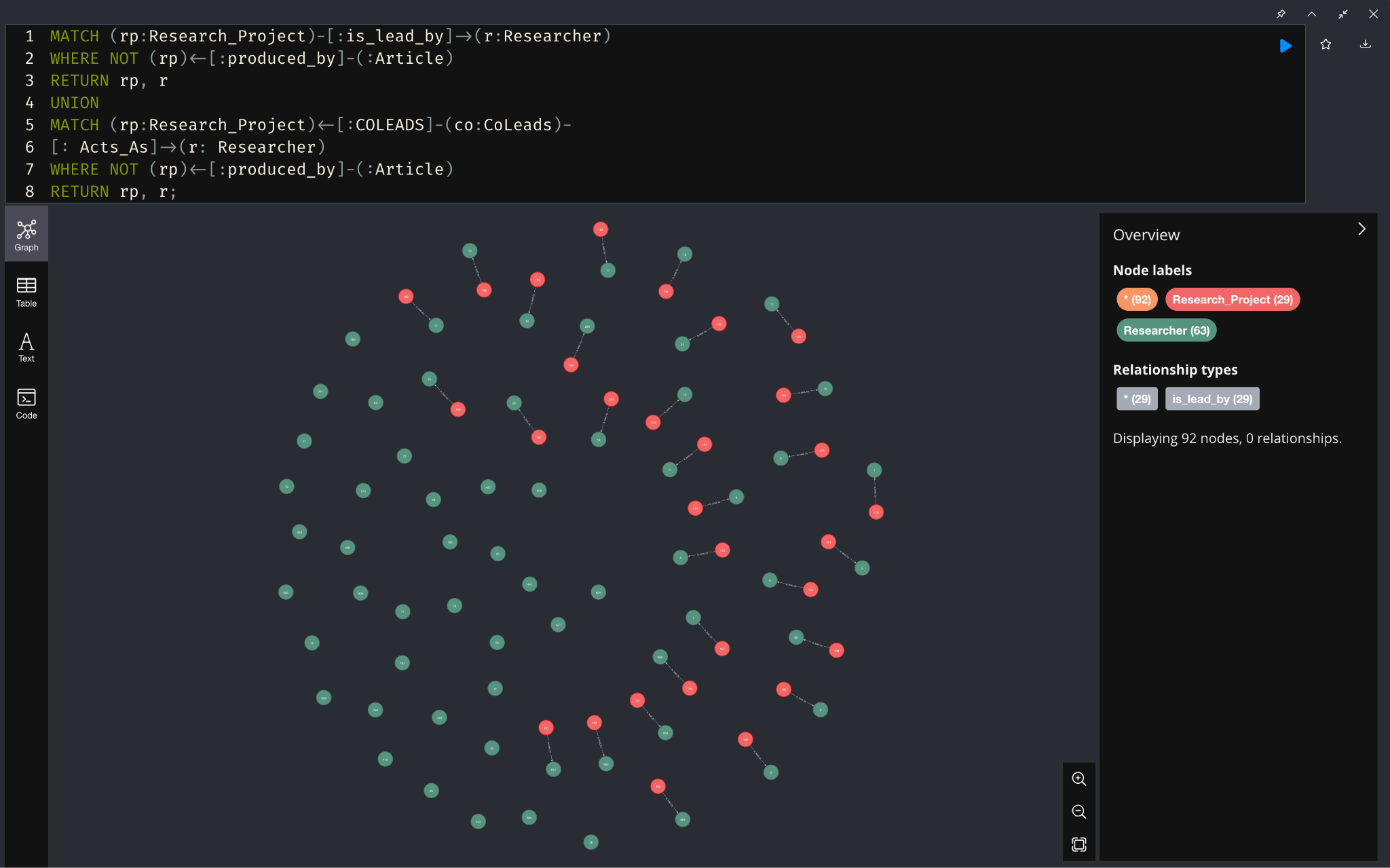
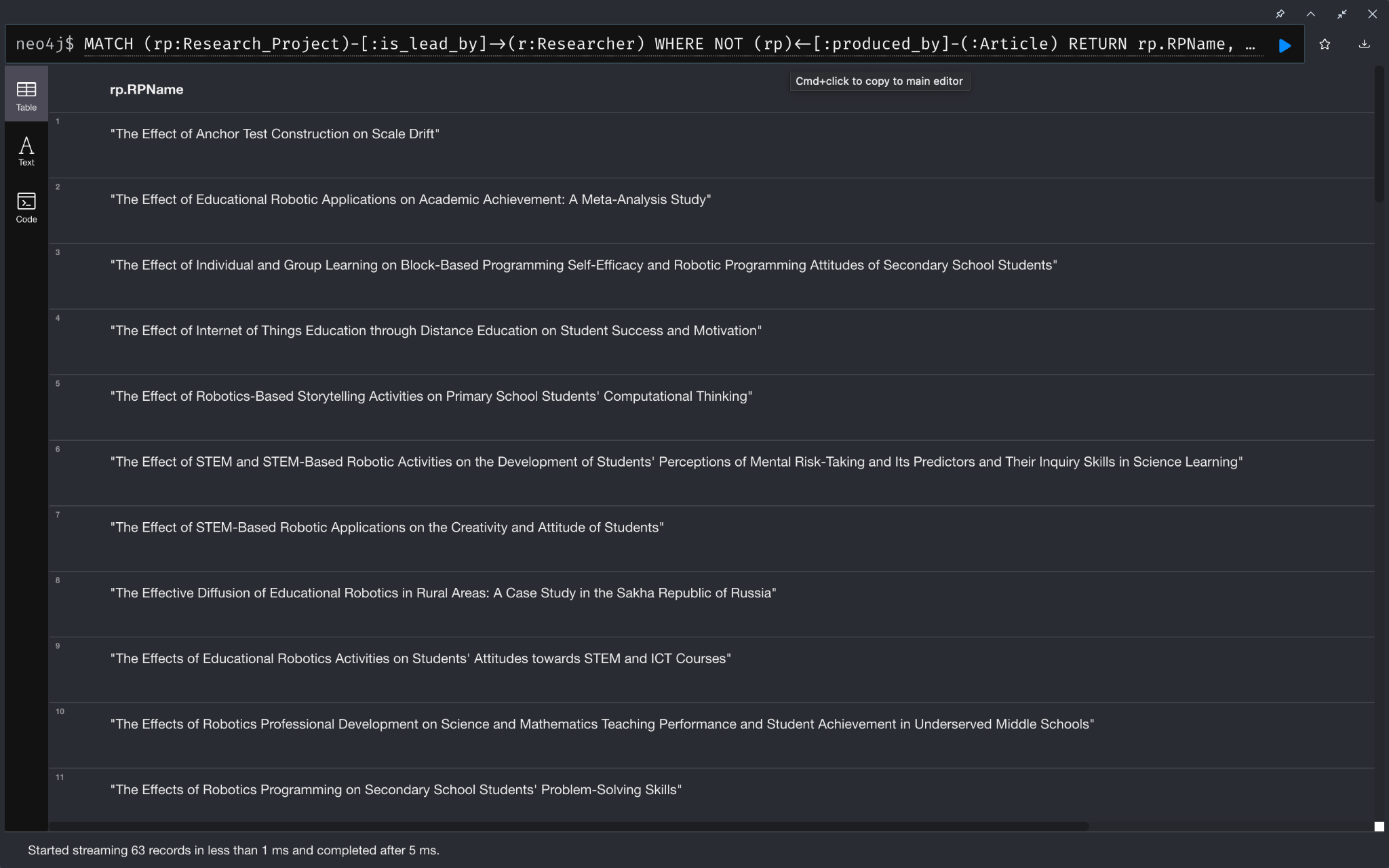
2.Retrieve article title from article table which have DBID=”ACH5” & PID=”QAP1”



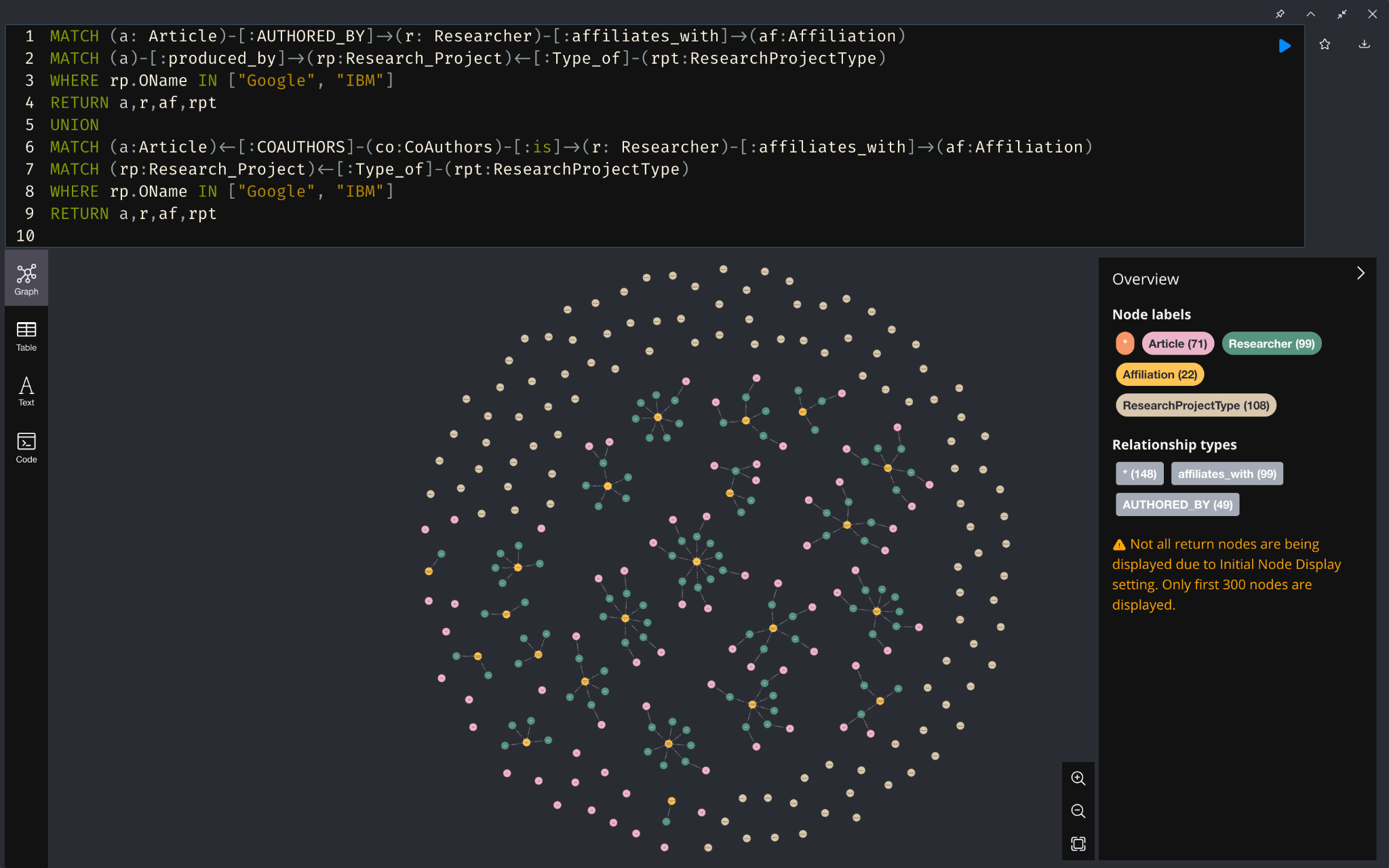
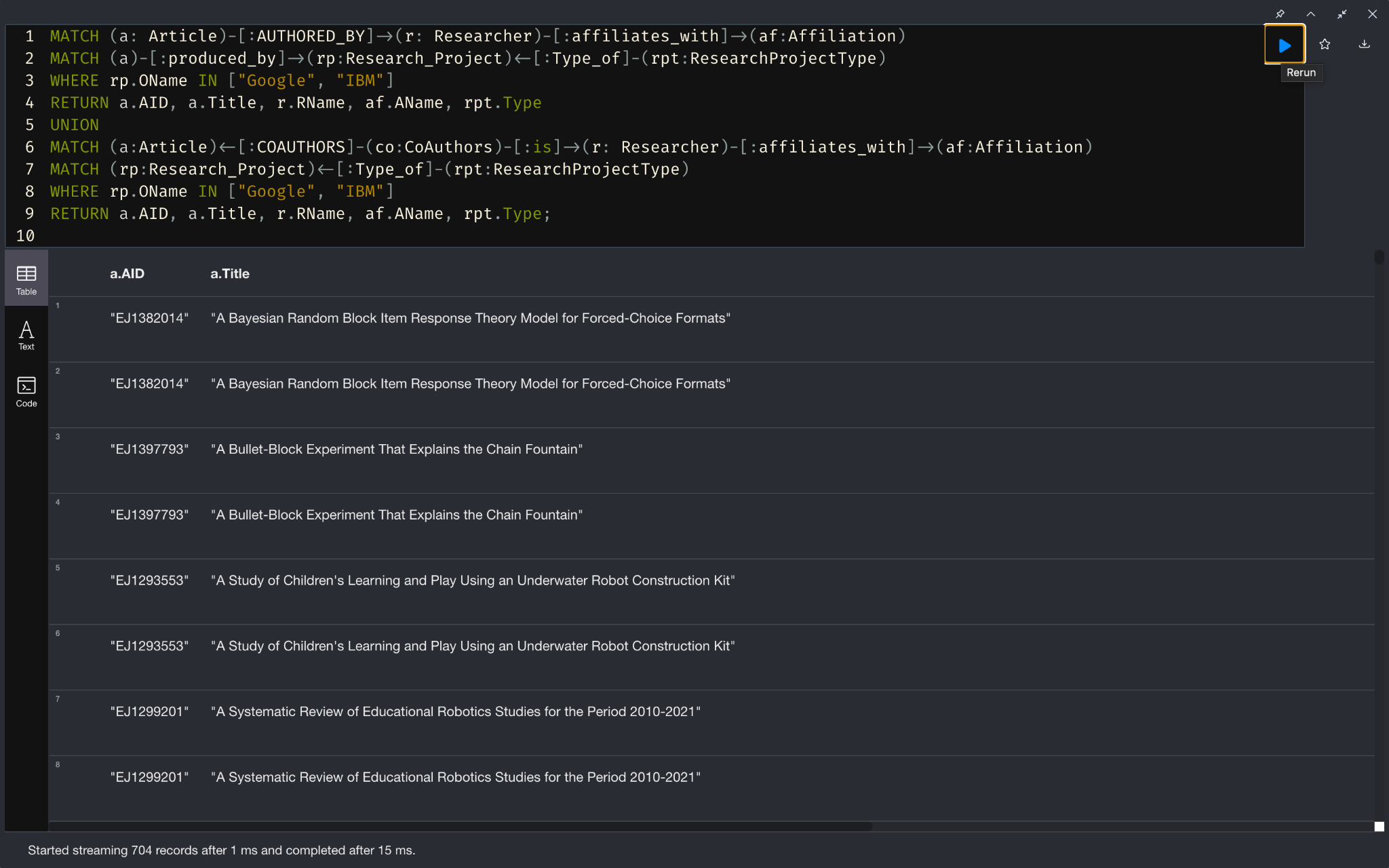
3. Retrieve all the article names that has a funding of >15000 along with their organisation name.





4.Retrieve all the research projects name and their authors name and co-leads name which have yet not been published. 

5.Retrieve all the articles name and id, their author’s and coauthor’s name,researcher’s affiliation name & research project type who have their funding from”Google,IBM”



Summary section, present your conclusions:

**Summarize everything so far**

In summary, our comprehensive analysis has revealed a clear performance distinction between graph and relational databases based on query complexity. Specifically, our findings affirm that graph databases excel in executing complex queries involving intricate joins, demonstrating notably faster execution times compared to their relational counterparts. Conversely, for relatively simpler queries (where a lot of joins between the tables is nowhere to be seen), the relational database exhibits superior performance**.**

**Execution time of queries-table.**

**Conclusion stating which one is better and when**

In summary, while relational databases are ubiquitous in academic institutions, they struggle with complex research data analysis involving diverse, interconnected information. Graph databases overcome these limitations through flexible, schema-free network models that efficiently capture the rich connectivity between academic entities and activities. This paper presents a novel graph database implementation using Neo4j tailored to university research data analysis in which the data is extensively correlated, therefore making graph databases as the ideal option. Comparative query evaluation confirms the graph approach enables more efficient, expressive and insightful multi-perspective analysis than rigid relational systems.The paper delivers an important proof point in applying graph technologies to unlock new value from academic knowledge networks.

A paragraph describing the contributions of each group member should be included in the report and duties in the project:

|  |  |
| --- | --- |
| **Methodology** | **Team members’ workload** |
| **Collecting data, identifying data entities and relationships in google scholar** | Mehul Jain, Swasha Kumar, Vidya Sreekumar |
| **Frame questions to be solved by graph data model** | Trishala Reddy, Dibyanshi Singh, Aishwarya Vinod Menon |
| **Design Graph Data Model** | All |
| **Create Graph database in Neo4j** | Trishala Reddy, Dibyanshi Singh, Aishwarya Vinod Menon |
| **Extract and load data from google scholar into graph database and run the queries** | Mehul Jain, Swasha Kumar, Vidya Sreekumar |
| **Creation of relational database** | Trishala Reddy, Dibyanshi Singh, Aishwarya Vinod Menon |
| **Analysis and comparison between graph and relational database** | All |

Provide a URL so we can obtain your source code, program, and data set in order to replicate your experiment's findings:

Zip file of all the programs along with the readme file along with all the csv data files

References:

1. <https://www.researchgate.net/publication/220996559>

Vicknair, Chad & Macias, Michael & Zhao, Zhendong & Nan, Xiaofei & Chen, Yixin & Wilkins, Dawn. (2010). A comparison of a graph database and a relational database: A data provenance perspective. The 48th ACM Southeast Conference. 10. 42. 10.1145/1900008.1900067.

1. <https://memgraph.com/blog/graph-database-vs-relational-database>

Memgraph, "Graph Database vs Relational Database", May 2023.[Online]. Available:https://memgraph.com/blog/graph-database-vs-relational-database

1. <https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf>

E.F.Codd (1970), "A Relational Model of Data for Large Shared Data Banks", Communications of the ACM, Volume 13, Number 6.

1. [https://www.academia.edu/5352898](https://www.academia.edu/5352898/NoSQL_Database_New_Era_of_Databases_for_Big_Data_Analytics_Classification_Characteristics_and_Comparison)

A B M Moniruzzaman, Syed Akhter Hossain (2013), “NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison”, International Journal of Database Theory and Application, Vol. 6, No. 4.

1. <https://www.geeksforgeeks.org/types-of-nosql-databases/>

Geeks for Geeks, "Types of NoSQL Databases", May 2023. [Online].Available:https://www.geeksforgeeks.org/types-of-nosql-databases/

1. <https://ieeexplore.ieee.org/document/6313676>

R. Angles, "A Comparison of Current Graph Database Models," 2012 IEEE 28th International Conference on Data Engineering Workshops, Arlington, VA, USA, 2012, pp. 171-177, doi: 10.1109/ICDEW.2012.31.

1. <https://www.researchgate.net/publication/323720876>

Patil N.S., Kiran P, Kavya N.P., Naresh Patel K. M. (2018), "A Survey on Graph Database Management Techniques for Huge Unstructured Data", International Journal of Electrical and Computer Engineering, Vol. 8, No. 2.

1. <https://www.sciencedirect.com/science/article/pii/0169023X9190033T>

Veda C. Storey, Relational database design based on the entity-relationship model,

Data & Knowledge Engineering, Volume 7, Issue 1,1991,Pages 47-83,

ISSN 0169-023X, https://doi.org/10.1016/0169-023X(91)90033-T.

(https://www.sciencedirect.com/science/article/pii/0169023X9190033T)

1. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7731929/>

Simpson CM, Gnad F. Applying graph database technology for analyzing perturbed co-expression networks in cancer. Database (Oxford). 2020 Dec 11;2020:baaa110. doi: 10.1093/database/baaa110. PMID: 33306799; PMCID: PMC7731929.

1. <https://www.researchgate.net/publication/332997927>

Sharma, Monika & Sharma, Vishal & Bundele, Mahesh. (2018). Performance Analysis of RDBMS and No SQL Databases: PostgreSQL, MongoDB and Neo4j. 1-5. 10.1109/ICRAIE.2018.8710439.