BIG DATA (CSP 554) - PROJECT DRAFT

Radhika Malhotra (A20491601)
Aastha Dhir (A20468022)
Aman Singh (A20491333)
Rohit Kumar (A20501314)

BOLLYWOOD MOVIES ANALYSIS Using NEO4J

Introduction

With their significant emphasis on "relationships", graph databases represent a paradigm change from relational databases. Graph databases save relationships for quick querying and data retrieval as opposed to database systems, which compute relationships at runtime. Graph databases are an important part of the NoSQL Database family. Graph databases mostly comprise nodes and edges where nodes signify entities and edges signify the relationships between the nodes. Neo4j is the leading database for linked data that was designed from the ground up to leverage data and data connections. Neo4j is fundamentally distinct from other data stores since it is a native graph database. Neo4j is built around a straightforward yet effective optimization. Direct pointers to all the nodes that it is connected to are contained in each data record or node. Relationships are what we term these direct pointers. The node itself contains all the data required to identify the subsequent node in the sequence. A connected graph serves as the native storage layer. Native implies exactly that. Neo4j doesn't have to calculate the relationships between your data at query time as a result of this approach. Traditionally, many SQL-based databases like MySQL, SQLite, and PostgreSQL are being used in various mobile and web applications and it is often seen that data engineers and scientists who try to adjust product requirements in relational database systems face problems in doing so. This is because most of the operations in relational databases are based on set operations and, as the size of data grows, the RDBMS operations get slower. Also, since most of the operations in RDBMS are based on set operations, an unexpected latency or memory usage occurs in query execution. It is difficult to represent relationships for connected data in other NoSQL databases. This is because they use disconnected aggregates for storing datasets. Hence, expensive Join Operations may be required which are often time-consuming. The database already has the connections saved. This leads to orders of magnitude faster inquiries on highly connected data. Direct pointers between records are not saved in other databases. They must compute links by looking through an index, a different data structure. This lookup method is quite expensive and becomes exponentially slower as the number of data and the complexity of the queries increase. It must be repeated to discover each link. As a result, they are fundamentally slower for relationship-intensive queries than Neo4j. So, we decided to move forth with this topic.

Abstract

Our team will be conducting an analysis of Bollywood movies and understand them in a better way using Neo4j. Neo4j will be used to analyze the associations among the key objects in the Bollywood movie data, including but not limited to directors, actors, writers, etc. The Neo4j database is a convenient tool to use because of its ability to deal with complex and multi-connection data. With the development of the Internet, data has grown immensely large to be handled by a single relational database with connections among different tables using foreign keys, and it's hard to query complex relationships among different entities. The volume and complexity of Bollywood movie data represent a big challenge in terms of data. Also, other NoSQL databases like MongoDB and Cassandra do not handle relationships very easily and features like foreign keys must be used explicitly to implement them. Hence, Neo4j is an appropriate choice for this project.

Technologies Implemented

Neo4J graph database, Cypher SQL, Python, and Machine Learning

Related Work and Background

According to previous works of literature, there has been a rise in the requirement for a flexible system of schema and hence, RDBMS is becoming a less preferred choice when current big data applications come into the picture. As mentioned earlier, joining many tables decreases Relational databases' efficiency. Graph databases such as Neo4j help in overcoming these problems. As per a recent study, Graph databases are replacing Relational Databases in several domains like semantic web networking, and recommendation engines to name a few. There are differences in terms of query structures and data models when we talk about Neo4j and RDBMS. Neo4j is an open-source program with a fast graphics engine at its core, recovery capabilities, two-phase submission, support for XA transactions, and other database product characteristics, which is presently the industry standard. It is an embedded, disk-based, fully transactional Java persistence engine that stores data structured in networks rather than tables. It is a network-oriented database. Neo4j's use of the so-called "network-oriented database" is what makes it intriguing. As opposed to the relational model's tables, rows, and columns, this paradigm expresses domain data as a "node space," which is a network of nodes, relationships, and attributes (key-value pairs). The context in which nodes interact is shown via the properties that can be annotated on relationships, which are first-class objects. Natural hierarchically arranged problem areas are well suited for the network model.

Data Collection

We will use the IMDB and Bollywood Movies datasets for our project. These datasets can be easily obtained from Kaggle. Both datasets have information about Bollywood movies from the 1990s to 2021. We are also using a Bollywood movie dataset compiled by P. Premkumar includes roughly 1700 Bollywood films from a two-year period for our reference.
Apart from using the Neo4j database for our project, we will also use Cypher SQL, a declarative, textual query language, like SQL used for graphs. It consists of expressions and keywords for execution purposes, some of which we are already aware of like WHERE, ORDER BY, SKIP, AND, etc. It's the query language for graphs. We can get data from the graph with Neo4j's Cypher. Because it is like SQL for graphs and was inspired by SQL, we can directly focus on collecting the desired data from the graph. It is by far the simplest graph language to learn due to its similarity to other languages and its intuitiveness.
Unlike SQL, Cypher is all about expressing graph patterns. There is a special clause MATCH for **matching** those patterns in your data. One would usually draw these patterns on a whiteboard, just converted into text using **ASCII-art symbols**.

Expected Result of Project Execution

We anticipate having a thorough understanding of NoSQL databases (Neo4j graph databases) and architectures, as well as Cypher SQL, by the time this project is finished. Additionally, we will try and integrate a machine learning model on the Neo4J graph database that will have been visualized.

References

- https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7960078
- https://www.computer.org/csdl/proceedings-article/icis/2017/07960078/12OmNzBOi3I
- https://neo4j.com/developer/get-started/
- https://www.computer.org/csdl/proceedings-article/icis/2017/07960078/12OmNzBOi3I
- https://www.semanticscholar.org/paper/Analysis-of-film-data-based-on-Neo4j-Lu-Hong/f307c76bc6399f91502f494b82eca1c5a498106f
- https://www.kaggle.com/datasets/mitesh58/bollywood-movie-dataset
- https://www.kaggle.com/general/231149
- https://www.kaggle.com/code/adrianmcmahon/imdb-indian-movies-eda/notebook

Project Plan and Milestones

Task	Expected Date of Completion	Status	Assigned To
Understanding architecture of Neo4J graph database	Oct 17	100%	Aman
Project Discussion 1	Oct 21	100%	Aman, Radhika, Aastha, Rohit
Project Topic Finalization; Finding BOLLYWOOD Dataset; Project Proposal Submission	Nov 3	100%	Aman, Radhika, Aastha, Rohit
Setting up Neo4j Environment	Nov 8	50%	Rohit
Understanding Cypher syntax and Logistics	Nov 12	75%	Radhika
Injecting the BOLLYWOOD dataset into Neo4J Environment; Query Processing in Neo4j	Nov 17	50%	Aastha
Project Literature Draft	Nov 21	100%	Radhika
Analysis and Machine Learning Implementation	Nov 28	10%	Aman
Implementing the System using Neo4j and Machine Learning	Dec 5	0%	Aman, Radhika, Aastha, Rohit
Testing and Validations	Dec 7	0%	Aastha
Future Scope	Dec 8	5%	Rohit
Final Project Report	Dec 8	0%	Aman, Radhika, Aastha, Rohit