**PERFORMANCE ANALYSIS OFACULTY AND STUDENTS USING Neo 4j**

S.Kiruthika, T.Dhivya & S.S.Kiruthika

Assistant Professor,

Department of CSE,

Knowledge Institute of Technology, Salem

**ABSTRACT**

Graph Database Management System (GDBMS) is a database which stores data in the form of graph structures. Neo4j is a world’s leading graph database. The Idea proposed in this system is to perform an analysis of faculty and students by certain categories. The aim of this project is to improve the organizational development and to analyze the students and faculty performance in their curriculum. The analyses are done by the following category, the faculty were requested to provide their co-curricular activities and they should also upload the marks and co-curricular activities of their respective students. Every faculty and student’s curriculum activities were monitored by the Head of the Department or by the authorized faculty. The Head of Department has the privilege to identify and monitor the individual and also overall performance of the students and faculties. The process helps them to analyze the faculty and students activities what they are done in their curriculum. The Neo4j server stores the data in the form of graph structures as well as in a tabular form. This project is very useful for every institution to analyze the performance of their faculty and students. Every student and teacher details are easily maintained by their institutions and can be retrieved easily.

**KEY**: Graph Database, Neo4j, NOSQL, CQL.

**1. INTRODUCTION**

**GRAPH DATABASE**

A Graph database is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and

store data. A key concept of the system is the graph (edge or relationship), which directly relates data items in the store. The connections permit information in the store to be connected

together specifically, and as a rule recovered with one operation. Directed Acyclic Graphs (DAG) is a common data structure to store data information and relationships.

Relationships give coordinated, named semantically pertinent associations between two nodes elements. A relationship always has a direction, a type, a start node, and an end node. As connections are stored proficiently, two nodes can share any number or relations type without giving up execution. There is one center steady lead in a graph database: "No broken connections". Since a relationship dependably has a beginning and end nodes, you can't erase a node without erasing its related connections. You can also always assume that an existing relationship will never point to a non- existing endpoint. Contrasted and relational databases, graph databases are frequently speeder for cooperative data maps and sets all the more directly to object oriented application structure. They depend less on a rigid schema, they are more suitable to manage ad hoc and changing data with evolving schemas. Graph databases are an intense tool for graph like queries. There are different types of graph database system have been introduced over a period of time. Neo4j is a highly usable, local graph database worked to impact information as well as its connections.

**Neo4j**

**­­** Neo4j is an open-source NoSQL graph database implemented in java and Scala. Neo4j is developed by Neo Technology, Inc., based in the San Francisco Bay Area, United States, North America, and also in Malmo, Sweden, Europe. With development starting in 2003, it has been publicly available since 2007. Neo4j is used today by a huge number of organizations and in almost all industries. Neo4j implements the Property Graph Model proficiently down to the storage level. Rather than diagram preparing or in-memory libraries, Neo4j gives full database attributes including ACID transaction consistence, cluster support, and runtime failover, making it appropriate to use graph data in production scenarios. Emerging of connections at creation time, resulting in no penalties for complex runtime queries. Enduring time traversals for connections in the graph both in depth and in breadth due to efficient representation of nodes and relationships. All connections in Neo4j are similarly vital and quick, making it conceivable to emerge and utilize relationships later on to “shortcut” and speed up the domain data when new needs arise. Smaller storage and memory getting for graphs, bringing about effective scale-up and billions of nodes in one database on direct equipment. Neo4j is a welcoming UI, Easy data modeling, Readable queries, active community, High performance, optional schema.

**RELATIONAL vs GRAPH DATABASE**

Conventional graph databases, the powerhouse of programming applications since the 1980s, function admirably when your information is fits and determined well into tables and wherever queries are not vey join-concentrated. Relational Database-Management System frameworks (RDBMS) is model information as an arrangement of tables, completing complex joins and self-joins when the dataset turns out to be more inter-related. Such inquiries are in technically complex to build and costly to run. In addition, making them work progressively while end clients hold up is difficult, with execution wavering as the aggregate dataset size increments. Consequently the ascent of another kind of database, enhanced for connected data: the graph database. Graph databases are convincing on the grounds that they empower organizations to understand the masses of connected data that exist today.

**NoSQL**

NoSQL is a way to deal with databases that speaks to a move far from traditional relational database management systems (RDBMS). To characterize NoSQL, it is useful to begin by depicting SQL, which is a query language used by RDBMS. Relational databases rely on tables, column, rows, or schemas to organize and retrieve data. In contrast, NoSQL databases do not rely on these structures and use more flexible data models. NoSQL can mean “not SQL” or “not only SQL”. As RDBMS have progressively neglected to meet the execution, versatility, and adaptability needs that next-generation, data-intensive applications require, NoSQL databases have been adopted by mainstream enterprises. NoSQL is especially helpful for storing unstructured data, which is becoming significantly more quickly than structured data and does not fit the relational schemas of RDBMS.

**2. EXISTING SYSTEM**

In existing system, RDBMS comprises of rows and columns which is basically used for the storage of data. SQL is used to query the RDBMS which generates the specific results according to the needs of the user. Apart from querying the data, we can also update and modify the records in the database. But unfortunately it is not so easy to perform the above mentioned operations. This is because RDBMS is basically a concept of fixed schema, making changes in one relation may result in multiple changes. Moreover, sometimes for retrieving the results, it is necessary to join multiple relations after satisfying all the constraints that the relations hold. It is difficult to join multiple relations when the size of the data grows. The solution for the above problems is to move on to graph database.

**Drawbacks**

* Privacy Issues
* Less security
* Inaccurate Information
* Increase time consumption

**3. PROPOSED SYSTEM**

In our project we have developed innovative approach consisting of graph database which will help user to track, retrieve, and understand the relationship of two or more attributes. It is something similar like a social platform that will help to track students and faculty performance in every co-curricular and extra-curricular activity. In proposed system, everything is done with the help of the neo4j server. By using this server, we can store the data in the form of graph structure. Project portal integrated with students and faculty performance management system is used to enter all data in the same software. Their performance can be analyzed and easily categorized. This is very useful for every Institution to analyze the performance of their faculty and students. Finally, we can generate the report in the form of table structure.

**Login Interface**

Login Interface maintains the details of student and faculty in the project portal. The login form allows authenticated user to enter into this application by giving input such as username and password. There are separate user name and password for faculty and admin. If the admin login to this portal, it displays student and faculty performance. If any faculty login to this portal, there are two buttons available that is view and edit. View button is only for seeing the performance of students. Edit button is used to enroll the details of students and faculty. Once the user logged out, the data will be stored and updated into the server.

**Build Faculty Profile**

Faculty Profile maintains the faculty personal details and co-curricular details in the project portal. The faculty is requested to provide their personal details and co-curricular details. The details are monitored by the Head of the department or Organization.

**Build Student Profile**

Student profile maintains the student personal details and co-curricular details in the project portal. The respective faculty advisor of the particular student, are requested to provide their student personal detail and co-curricular detail. The details are monitored by the Head of the department and faculty.

**Faculty Report**

Faculty Report is used to analyze the faculty’s co-curricular activities. This module displays the details in the form of table structure. By using this, organization can easily identify which faculty is performing best in their curriculum activities.

**Student Report**

Student Report is used to analyze the student co-curricular activities and curriculum activities. This module displays the details in the form of table structure. By using this, organization can easily identify which student is performing best in their curriculum activities.

**Overall Report Generation**

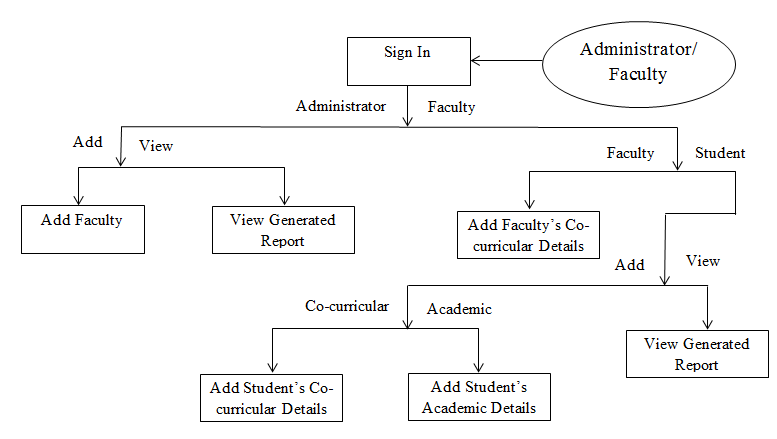
Report generation module takes the faculty and student information as data and the data is analyzed based on the criteria provided. This report can be viewed in the form of table and also in the form of pie chart.

**4. SYSTEM ARCHITECTURE**

**Process Flow of the Project**

* Initially every faculty will be provided with a user ID.
* All the faculties were requested to provide their co-curricular activities.
* Faculties should also upload the marks of their respective students.
* All the details which were stored will be classified based on certain category.
* The categorized data will be monitored frequently and updated to the respective authority.
* This engine will display the result in the form of visualization as well as in table format.

**Flow Diagram**

****

**5. CONCLUSION**

In overall, institute as a whole can perform better by enhancing the performance of its faculty and students. By using neo4j server, the institute administration will be able to make groups of faculty members and student with different parameter for future use. This helps in decision making by comparing various parameters of students and faculty. The administrator can analyze the performance of faculty and students. Here the categorized data will be monitored frequently and updated to the respective authority. Thus helps in decision making and in participating workshops, seminars or conference etc.

**6. REFERENCES**

[1]Batra S , Tyagi .May 2012. Comparative Analysis of Relational And Graph Databases. International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-2,(509-512).

[2]Femy M, Varghese S M. April 2016. Outcome Analysis Using Neo4j Graph Database. International Journal on Cybernetics & Informatics (IJCI) Vol. 5, No. 2,(229-236).

[3] Bunakov V. October 2015. Use cases for triple stores and graph databases in scalable data infrastructures. Proceedings of the XVII International Conference «Data Analytics and Management in Data Intensive Domains» (DAMDID/RCDL’2015), Obninsk, Russia,(37-40).

[4] Neo4j graph database. <http://neo4j.com/>

[5] Neo4j Blog, Available http://blog.neo4j.org/2009/04/current-database-debate-and-graph.ht ml.