Mustansiriyah Journal of Pure and Applied Sciences, Print ISSN: 2957-9910. Online ISSN: 2957-9929, Vol. 3, No. 2, 2025, Pages 153-168
DOI: 10.47831/mjpas.v3i2.325

MJPAS

MUSTANSIRIYAH JOURNAL OF PURE AND APPLIED SCIENCES

Journal homepage:
https://mjpas.uomustansiriyah.edu.iq/index.php/mjpas

RESEARCH ARTICLE – COMPUTER SCIENCE

Leveraging hybrid database models for enhanced gene-disease association analysis

Sama Salam Samaan [1*], Saja Dheyaa Khudhur [2], Omar Nowfal Mohammed Taher [3]

[1,2,3] Computer Engineering Department, University of Technology - Iraq
[*] Corresponding author E-mail: sama.s.samaan@uotechnology.edu.iq

Article Info.

Article history:

Received
17 September 2024

Accepted
10 October 2024

Publishing
30 March 2025

Abstract

Many diseases are driven by genetic variations. The Gene-Disease Association (GDA) dataset, structured as a network, evaluates the relationships between genes and diseases. Typically, the GDA dataset consists of semi-structured data, which does not conform to a tabular format. In this work, we propose a hybrid approach for processing, storing, and analyzing TBGA, a GDA dataset comprising over 200,000 JSON instances and 100,000 gene-disease pairs. We introduce two procedures to import the TBGA dataset into both a relational model and a graph model. SQL Server is employed for the relational model to support analytical and reporting tasks, while Neo4j is used for the graph model to enable visualization and the application of graph algorithms tailored for GDA analysis. Experimental results demonstrate the effectiveness of each model, with SQL Server excelling in analytical tasks and Neo4j in visualization and graph analysis..

The official journal published by the College of Education at Mustansiriyah University

Keywords: GDA, graph database, semi-structured data, TBGA.

1. Introduction

Genetic diseases occur when a mutation affects genes or when there is a wrong amount of genetic material. Studying genetic variation is essential in understanding and identifying the genetic factors involved in diseases. Although significant advancement has been achieved over the past two decades, deciphering the genetic foundations of rare diseases is still a challenging task, characterized by its intricacy. Genetic instability is the main cause of genetic heterogeneity. Many diseases are caused by gene abnormalities. Cancer, for example, shows highly heterogeneous genotypes, leading to difficulty identifying the biological markers [1]. Another example is Autism Spectrum Disorder (ASD) which involves more than 800 genes and numerous genetic syndromes [2] [3]. COVID-19 depicts a notable variability in clinical symptoms, ranging from asymptomatic disease to deadly pneumonia . Part of this heterogeneity can be explained by differences in the host genetic profile. These differences are determined by both common and irregular genetic variants in the host genome [4].

Gene-disease associations (GDA) quantify the relation between a pair of genes and disease. They are usually created as a network where we can probe the gene-disease mechanisms by considering multiple genes and disease factors. This task is to predict the association of any gene and disease from both a biochemical modeling and network edge classification perspective. Network-based computational methods are used for GDA prediction based on the assumption that the genes driving the same diseases are located close to each other in a molecular network [5].

GDA is usually considered semi-structured data, data that is not purely structured but also not completely unstructured [6]. It contains some level of organization but fits outside a rigid schema or data model and may include elements that are not easily categorized or classified. It is typically characterized by metadata or tags providing supplementary information about the data elements.

Examples of semi-structured data are CSV, XML, and JSON [7]. Semi-structured data is becoming increasingly common as organizations collect and process more data from various sources, including social media, IoT devices, and other unstructured sources. While semi-structured data can be more challenging to work with than strictly structured data, it offers greater flexibility and adaptability, making it a valuable tool for data analysis and management. NoSQL databases are considered a popular candidate for handling semi-structured data [8].

A graph database is a NoSQL database that stores data as a network graph [9]. It is different from the other NoSQL options in that it documents and prioritizes the relationships between data. Graph databases comprise nodes and edges, where nodes represent specific entities, while edges represent the connection between two nodes [10]. They are designed to be scalable and offer flexibility that's hard to find in other databases. While in a relational database, data is stored in tables, with each table defined by its columns and rows [11].

The main contribution of this work is to utilize both relational and graph databases in a hybrid approach to store and analyze GDA data. We exploit the strengths of both models to handle and process TBGA, a GDA extraction dataset generated from more than 700,000 publications that consist of over 200,000 instances and 100,000 gene-disease pairs [12]. We employ an SQL server for the relational model and Neo4j as a graph database model. The rest of this paper is organized as follows. Section 2 reviews the state-of-the-art in the use of relational and graph databases, particularly in the biomedical sector. Section 3 introduces two methodologies for processing, storing, and analyzing the TBGA dataset within relational and graph models, respectively. Section 4 details the experiments conducted using SQL Server and Neo4j databases. Section 5 discusses the results obtained from applying the two models. Finally, Section 6 concludes the paper.

2. Related work

Until now, relational databases have remained essential and dominant in various applications such as data mining, business intelligence, and data analysis. It has been the predominant data model for storing and accessing data since the 1970s [13]. A relational database is preferred among developers for its simplicity and flexibility, primarily due to its data abstraction feature, which shields users from the complexity of its internal data storage mechanisms. However, simplicity is now the mainstream in today's data-intensive world. H. R. Vyawahare et al. [14] recognize the limitations of traditional relational databases, especially in the context of Web 3.0 and big data, where the need to manage relationship-rich data is dominant. They highlight that graph databases, such as Neo4j, which support the property graph model, offer a flexible and efficient data handling structure. However, the paper needs to include a practical case study to demonstrate the real-world application and effectiveness of the proposed hybrid database model. In addition, the paper mentions the advantages of NoSQL databases in terms of scalability but needs to provide a detailed evaluation of the hybrid model's scalability.

Recently, graphs have gained significance in big data applications like social network analysis, spatiotemporal analysis, navigation, and consumer analytics. They excel in capturing intricate relationships and data dependencies. For instance, in social networks, vertices represent users, pictures, and events, while edges depict relationships between them. Y. Cheng et al. [15] provide a detailed comparison between Relational Database Management Systems (RDBMSs) and Graph Database Management Systems (GDBMSs). They develop a unified benchmark to evaluate both systems using the same datasets and metrics, addressing differences in data models and query languages. The study reveals that RDBMSs outperform GDBMSs in workloads involving group-by, sort, and aggregation operations, while GDBMSs excel in multi-table joins, pattern matching, and path identification. The paper highlights the lack of a unified graph model and query language in GDBMSs as a significant limitation, alongside the considerable programming and maintenance overhead required for their use.

Although the study is comprehensive, it does not include a real-world case study, which could have provided additional insights into the practical applications of both systems.

SQL databases have traditionally managed and queried data across various domains, including biomedical fields [16]. However, the rise of NoSQL databases presents an alternative solution to handle big data challenges posed by large volume, velocity, variety, and veracity of data. Among these, the Neo4j graph database is increasingly adopted for its efficiency, flexibility, and scalability in querying and integrating data based on a graph data model, particularly suited for relationship-oriented or knowledge-based applications. I. K. Subagja et al. [16] explore using Neo4j to integrate various immunological datasets. The authors developed a system that supports complex query capabilities using the Cypher query language and provides a user-friendly web-based interface for researchers. This interface allows users to easily navigate and visualize the integrated data without learning Cypher. The primary advantage highlighted is the efficiency and flexibility of graph databases in managing highly relational biological data compared to traditional SQL databases. However, the paper needs extensive case studies demonstrating real-world applications, limiting the empirical validation of the system's effectiveness. The paper needs to include detailed performance benchmarks comparing LinkedImm with traditional relational or other graph databases, and it addresses the scalability of handling large datasets. Additionally, it must profoundly explore data consistency and integration challenges or include user feedback and usability studies to assess the web-based interface's practical usability.

Rapid advancements in experimental and analytical techniques provide diverse information about biological components. Although individual details are essential, the fundamental aim of biology is to understand the complex interactions among heterogeneous data contributing to cellular functions. Identifying these intricate relationships is challenging due to their complexity. S. A. C. Bukhari et al. [17] clarified the feasibility of using graph databases for representing and storing complex biological data and relationships. They collected diverse biological data, such as gene-disease associations, protein-protein interaction, etc., and compared the performance of MySQL and Neo4j as a relational database and graph database, respectively. Neo4j outperformed MySQL in all perspectives, particularly query response time and search speed.

J. Schäfer et al. [18] developed a web application that utilizes Neo4j as a graph database, which was gained by transforming an SQL database. The developed application provided analysis and visualization of patients. In addition, they used NeoDash, a tool for creating a dashboard in Neo4j, to visualize and query a specific patient or group of patients, display information such as gender distribution, and find particular patterns shared among patients. Some limitations are observed, such as directly exporting report charts and tables from the dashboard. In addition, the paper didn't exploit graph algorithms to observe patterns, such as patients with identical or comparable medical histories. These observations are instrumental in pinpointing a medical treatment roadmap for each patient. Although previous works provide valuable insights into the use of relational and NoSQL databases, they do not address the analysis and knowledge extraction from large GDA datasets like the TBGA. Furthermore, they lack real-world case studies that demonstrate practical applications of both models. Additionally, previous studies have not employed graph algorithms to discover patterns such as gene or disease similarity, which is crucial for suggesting personalized treatment roadmaps for patients Table 1 shows a related work summary showcasing the employed database, the key findings, and the work limitations.

Table 1. Related Work Summary.

Ref	Authors	Publication Year	Database Type	Key Findings	Limitation
[14]	H. R. Vyawahare, P. P. Karde, V. M. Thakare	2019	Hybrid Database	Highlights the advantages of graph databases like Neo4j over traditional relational databases in the context of Web 3.0 and big data.	Needs a practical case study to demonstrate the real-world application and effectiveness of the proposed hybrid database model. Requires a detailed evaluation of the hybrid model's scalability.
[15]	Y. Cheng, P. Ding, T. Wang, W. Lu, X. Du	2019	Relational and Graph Database	Reveals that RDBMSs outperform GDBMSs in group-by, sort, and aggregation operations, while GDBMSs excel in multi-table joins, pattern matching, and path identification.	Lacks a unified graph model and query language in GDBMSs, presenting a significant limitation. Requires considerable programming and maintenance overhead for GDBMSs. Does not include a real-world case study, which could have provided additional insights into the practical applications of both systems.
[16]	I. K. Subagja et al.	2019	NoSQL Database	Explores the use of Neo4j for integrating immunological datasets, highlighting efficiency and flexibility in managing highly relational biological data.	Needs extensive case studies to demonstrate real-world applications, limiting the empirical validation of the system's effectiveness. Lacks detailed performance benchmarks comparing LinkedImm with traditional relational or other graph databases. Needs to profoundly explore data consistency and integration challenges. Does not include user feedback and usability studies to assess the web-based interface's practical usability.
[17]	S. A. C. Bukhari, S. Pawar, J. Mandell, S. H. Kleinstein, K.-H. Cheung	2021	Graph Database	Demonstrates the feasibility of using graph databases for representing and storing complex biological data, with Neo4j outperforming MySQL in query response time and search speed.	Needs detailed performance benchmarks comparing the system with traditional relational or other graph databases. Requires exploration of data consistency and integration challenges. Lacks user feedback and usability studies to assess the web-based interface's practical usability.
[18]	J. Schäfer, M. Tang, D. Luu, A. K. Bergmann, L. Wiese	2022	Graph Database	Developed a web application using Neo4j for analysis and visualization of patients, utilizing NeoDash for dashboard creation.	Lacks the ability to directly export report charts and tables from the dashboard. Does not exploit graph algorithms to observe patterns such as patients with identical or comparable medical histories, which are instrumental in pinpointing a medical treatment

				Highlights limitations in report export and lack of graph algorithms for pattern observation.	roadmap for each patient.

3. Materials and Methodology

This section demonstrates the TBGA dataset employed in this work, detailing its construction, format, and size. Additionally, we propose two procedures for importing TBGA, the first procedure imports it into a relational model using SQL Server and the second procedure imports it into a graph model using the Neo4j database.

3.1. Dataset Description

The TBGA dataset is designed for Biomedical Relation Extraction (BioRE) [19], focusing on GDA. DisGeNET database is exploited to build the TBGA dataset, which is a GDA dataset created in a semi-automagical manner [20]. The DisGeNET is one of the largest collections of human genes and diseases [21]. The TBGA is extracted from more than 700,000 publications. It comprises 218,973 instances and more than 100,000 gene-disease pair [12]. Each instance consists of three parts, the sentence from which the GDA is extracted, the corresponding GDA, and the information about the gene-disease pair. Fig. **1** shows a sample record from the TBGA dataset showing the sentence from which the GDA was extracted, the relationship name associated with the given GDA, and the JSON objects representing the gene and disease within the record.

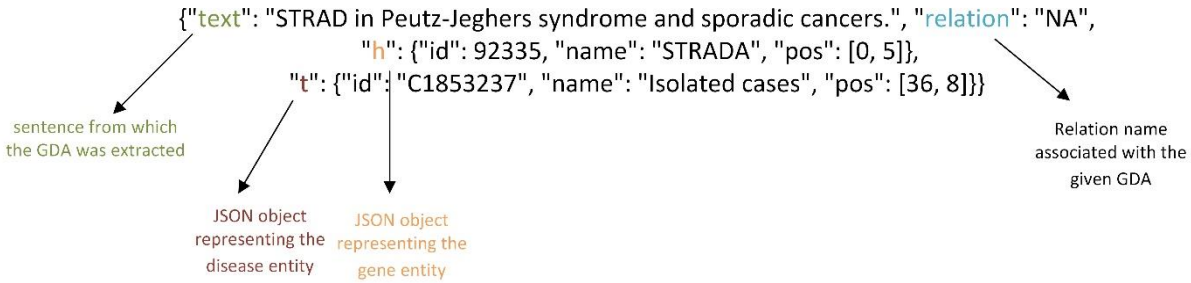

Fig. 1. A sample record from the TBGA dataset.

3.2. Importing TBGA into a Relational Model

To effectively import the TBGA dataset into a relational model, we develop a Python application to import the data from the TBGA JSON file into the SQL Server database. The procedure consists of the following steps:

1. Database Connection: A connection to the SQL Server using Windows Authentication is established.
2. Tables Creation: Three tables are created: `Genes`, `Diseases`, and `Associations`. The `Genes` and `Diseases` tables store unique identifiers and names, while the `Associations` table captures the relationships between genes and diseases, including relevant metadata.
3. Loading TBGA Dataset: The JSON data is loaded line by line from the specified file.
4. Parsing JSON Record: The JSON file is read line by line to handle multiple JSON objects. Each JSON record in the TBGA dataset consists of four parts: `text`, `relation`, `h`, and `t`. The `id` and `name` of the gene are extracted from the `h` part, and the `id` and `name` of the disease are extracted from the `t` part.

5. Data Ingestion: The extracted data is inserted into their respective tables, `Genes`, and `Diseases`, ensuring no duplicates. The association data is inserted into the `Associations` table, linking genes and diseases through foreign keys.

Following the steps above, three tables named `Genes`, `Diseases`, and `Associations` are created. The `Genes` table has 11,784 rows, the `Diseases` table has 9,199 rows, and the `Associations` table has 218,973 rows. Fig. **2** shows the relational schema of the TBGA database.

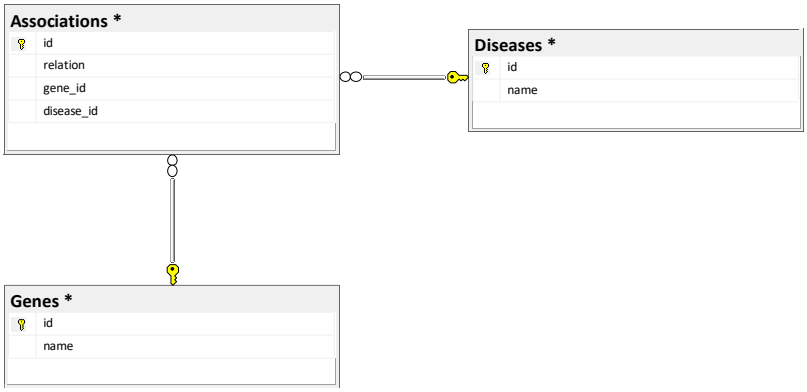

Fig. 2. The relational schema of the TBGA database.

3.3. Importing the TBGA dataset into a Graph Model

This section outlines the step-by-step procedure for constructing the property graph and importing the TBGA dataset into the Neo4j graph database.

1. Nodes Creation: A schema for two types of nodes, `Genes` and `Diseases`, is defined.
2. Loading TBGA Dataset: The TBGA dataset file, containing data in JSON format, is loaded.
3. Parsing JSON Record: The necessary information is extracted for each record in the dataset. The gene ID and `Gene name` are extracted from the h part for the `Gene` node, and the disease ID and `Disease Name` are extracted from the `t` part for the `Disease` node.
4. Graph Population: The extracted information is inserted into their respective nodes, `Genes` and `Diseases`.
5. Relationships Creation: For each record, a `Cause` relationship is created from the `Genes` node to the `Diseases` node based on the extracted information. This involves matching the two nodes and creating a relationship between them.

Following the steps above, a graph of 20,983 nodes (11,784 Gene nodes and 9,199 Disease nodes) and 106,032 relationships is constructed. This graph represents the complex interplay between genes and diseases, providing a robust foundation for further analysis and exploration in biomedical research. Fig. **3** shows the graph schema of the TBGA database.

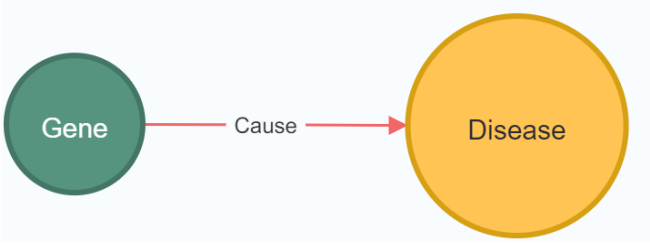

Fig. 3. The graph schema of the TBGA database.

The difference in the number of rows imported into the `Associations` table (218,973 rows) and the number of relationships created in the graph database (106,032 relationships) can be attributed to several factors:

- Duplicate Relationships: The same gene-disease association might be listed multiple times with different metadata in the relational database. However, these would be represented in the graph database as a single relationship between two nodes.
- Data Normalization: The relational model might store more detailed records that result in multiple rows, while the graph model could condense these into fewer relationships.

The block diagram presented in Fig. **4** effectively illustrates importing the TBGA dataset data into relational and graph databases utilizing SQL Server and Neo4j, respectively. This hybrid approach highlights the strengths of both relational and graph databases in managing and analyzing complex GDA data.

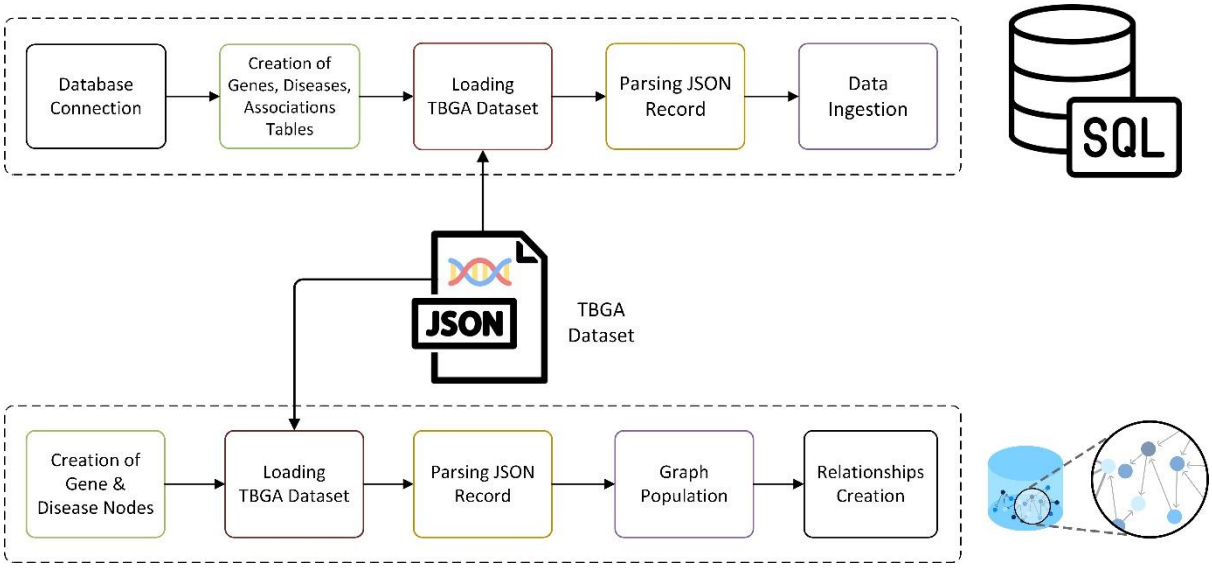

Fig. 4. A block diagram illustrating the steps of importing the TBGA dataset into SQL Server and Neo4j databases.

4. Experiments

To demonstrate the effectiveness of the proposed hybrid approach, we conducted a series of experiments employing SQL server and Neo4j, which represent relational and graph databases, respectively.

4.1. Experiment Setup

This work uses Neo4j version 4.4.35 as a graph database due to its scalability, high performance, and native graph storage and processing capabilities [22]. Cypher is used to implement the importing procedure since it is the most widely adopted query language for property graph databases, such as Neo4j. SQL Server Management Studio (SSMS) 20.2 is used for relational database model. All experiments were executed on a laptop with an Intel Core i7 CPU, four cores, and 16 GB of RAM.

4.2. Experimenting with SQL Server

The first part of the experimentation entails executing a series of SQL queries on the imported TBGA dataset to demonstrate relational databases' superiority in data analysis and reporting. The experiments exhibit how efficiently SQL Server can handle complex queries, perform aggregations, and generate insightful reports from the created database.

a. Listing all genes causing a specific disease

The query clarified below is used to retrieve the genes associated with a specific disease. It includes three joins, an aggregate function (COUNT), and a GROUP BY clause. The query displays the count of distinct genes and a comma-separated list of these gene names.

SQL Query 1 Retrieve the genes associated with a specific disease.

```
SELECT
    d.name AS disease_name,
    COUNT(DISTINCT g.id) AS gene_count,
    STUFF((
        SELECT DISTINCT ', ' + g2.name
        FROM associations a2
        INNER JOIN genes g2 ON a2.gene_id = g2.id
        WHERE a2.disease_id = d.id
        FOR XML PATH(''), TYPE).value('.', 'NVARCHAR(MAX)'), 1, 2, '') AS
gene_names
FROM diseases d
INNER JOIN associations a ON d.id = a.disease_id
INNER JOIN genes g ON a.gene_id = g.id
WHERE d.name = Disease Name
GROUP BY d.name, d.id;
```

b. Counting the Number of Diseases Associated with a specific Gene

To investigate the number of diseases associated with a specific gene, we executed the SQL query clarified as follows.

SQL Query 2 Retrieve the number of diseases associated with a specific gene.

```
SELECT g.name AS gene_name, COUNT(a.disease_id) AS disease_count
FROM associations a
INNER JOIN genes g ON a.gene_id = g.id
Where g.name= Gene Name
Group by g.name
```

c. Identifying the Genes Most Frequently Linked with Diseases

This query is used to identify each gene and the number of diseases associated with it, ordered in descending order.

SQL Query 3 Retrieve each gene and the count of diseases associated with that gene.

```
SELECT Genes.name AS gene_name, COUNT(DISTINCT Associations.disease_id) AS
disease_count
FROM associations
INNER JOIN Genes ON Associations.gene_id = Genes.id
GROUP BY Genes.name
ORDER BY disease_count DESC
```

d. Identifying the Diseases Most Frequently Linked with Genes

The SQL query shown below is used to identify each disease and the number of genes associated with it, ordered in descending order.

SQL Query 4 Retrieve each disease and the count of genes associated with that disease.

```
SELECT Diseases.name AS disease_name, COUNT(distinct Associations.gene_id) AS
gene_count
FROM Associations
INNER JOIN Genes ON Associations.gene_id = Genes.id
INNER JOIN diseases ON Associations.disease_id = Diseases.id
GROUP BY Diseases.name
ORDER BY gene_count DESC
```

4.3. Experimenting with Neo4j

The second part of the experiment involves implementing a series of Cypher queries on the imported TBGA dataset. It aims to reveal graph databases' superiority in data visualization and handling complex relationships between genes and diseases to uncover meaningful patterns and insights. The experiments illustrate executing multiple graph algorithms tailored for GDA.

a. Visualization

One key benefit of using a graph database is the ability to visualize complex relationships. In this part of the experimentation, we focus on visualizing the genes associated with a specific disease. A cypher query is executed to find and visualize all the genes that cause a specific disease, as shown in the Neo4j Query 1 listed below.

Neo4j Query 1 Finding and visualizing all the genes causing a specific disease.

```
MATCH (d:Disease {name: Disease Name})<-[:Cause]-(g:Gene)
RETURN d, g
```

b. Similarity Analysis

Beyond visualization, we also explore the similarity between a pair of diseases based on the genes that cause each of them. Using the Node Similarity algorithm, we can measure the similarity between diseases to identify potential common treatments and to understand the shared underlying mechanisms of both diseases, since diseases with similar genetic profiles may respond similarly to certain treatments [23].

c. Gene Ranking

The PageRank algorithm measures the most influential genes within the network. Initially developed for ranking web pages, the PageRank algorithm is used here to determine the importance of genes based on their connections within a network of GDA. It assigns a score to each gene based on its connections. This approach helps identify which genes play pivotal roles within the database, prioritizing research and understanding the network's dynamics.

5. Results

5.1. SQL Server Experiment Results

a. Listing all genes causing the "Brain Tumor" disease

For "Brain Tumor, Primary", executing the SQL Query 1 identifies 18 genes associated with this disease as shown in

Fig. 5. The query ensures no duplicate genes are counted or listed, offering an accurate and comprehensive output. The query's performance can vary based on the database size and indexing.

disease_name	gene_count	gene_names
Brain Tumor, Primary	18	ADAM23, AMT, CDKN2A, CST6, IDO1, LDHA, LRP1B, MAP2K7, MAPK8, MMRN1, MPP5, MYC, NQO1, PMS2, PTN, RASSF1, TGM2, XRCC1

Fig. 5. List of genes associated with the Brain Tumor disease.

b. Counting the Number of Diseases Associated with a specific Gene

In this example, executing the SQL Query 2 results in retrieving the count of diseases linked to the "PMS2" gene, revealing that it is associated with 106 diseases, highlighting the significant role of the "PMS2" gene in various diseases.

gene_name	disease_count
PMS2	106

Fig. 6. The "PMS2" gene and the count of diseases associated with it.

c. Identifying the Genes Most Frequently Linked with Diseases

Executing the SQL Query 3 results in identifying each gene and the count of diseases associated with it, ordered in descending order, as shown in Fig. 7.

	gene_name	disease_count
1	CYREN	415
2	MTSS1	365
3	CD4	360
4	TNF	320
5	IFNA1	316
6	SH3PXD2A	304
7	TP53	287
8	IL6	261
9	NFKB1	237
10	CDKN2A	232

Fig. 7. Part of a table showing the top ten genes and the count of diseases associated with them.

d. Identifying the Diseases Most Frequently Linked with Genes

Executing the SQL Query 4 results in identifying each disease and the count of genes associated with it, ordered in descending order, as shown in

	disease_name	gene_count
1	Neoplastic Cell Transformation	1539
2	Chromosomal translocation	1254
3	Inflammation	1197
4	Isolated cases	1187
5	Congenital Abnormality	1154
6	Hypoxia	977
7	Tissue Adhesions	911
8	Mild Adverse Event	875
9	Neoplasms	806
10	Malignant Neoplasms	758

Fig. 8. Part of a table showing the top ten diseases and the count of genes associated with them.

5.2. Experimenting with Neo4j

a. Visualization

An example of executing the Neo4j cypher query 1 is to retrieve all the genes associated with the "Brain Tumor" disease through the "Cause" relationship. As seen in Fig. **9**, the central node represents the "Brain Tumor" disease, and the surrounding nodes represent the genes associated with it. The arrows denote the "Cause" relationship, indicating that each gene has been linked to the disease. Another example is shown in Fig. **10**, which shows the genes associated with the "Tumor Vasculature" disease. By visualizing the data in this manner, researchers can quickly identify critical genes involved in a specific disease, facilitating further analysis and potentially leading to discoveries in disease mechanisms and therapeutic targets.

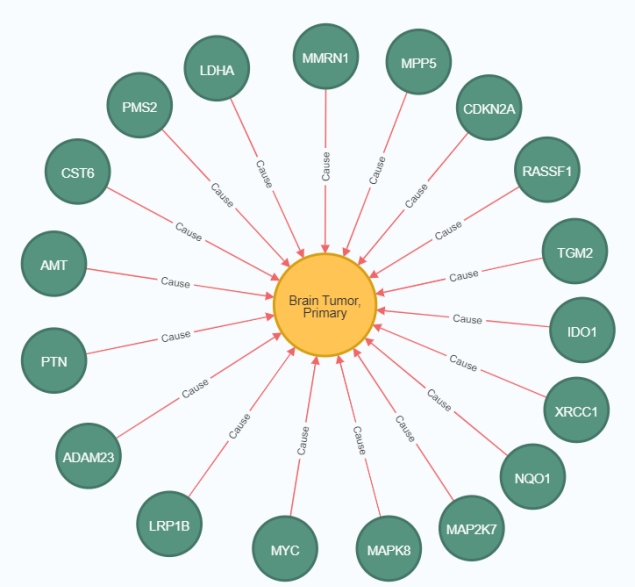

Fig. 9. Genes causing the "Brain Tumor, Primary" disease.

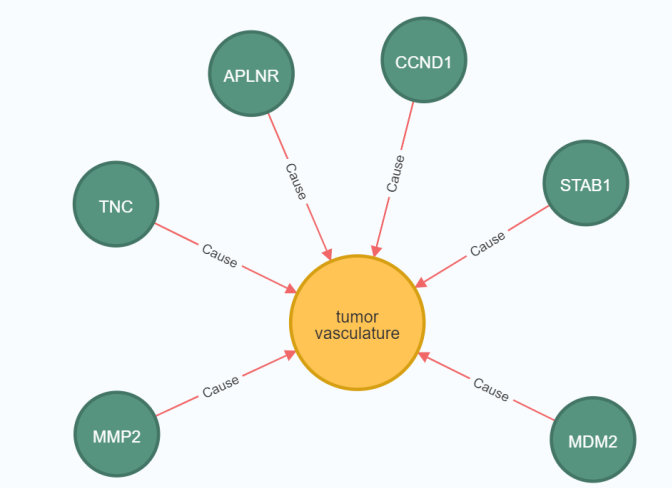

Fig. 10. Genes causing the "Tumor Vasculature" disease.

b. Similarity Analysis

For instance, the following visualizations show a couple of diseases, "Acrania" and "Diastematomyelia", each caused by three common genes, CSF2, PRSS8, and IFNG. As illustrated in Fig. **11**, both diseases share these three genes, indicating a potential similarity. The similarity score of both diseases is illustrated in

, which shows that the similarity score between "Acrania" and "Diastematomyelia" equals 1. In addition, "Acrania" is similar to three other diseases as listed in the same table. This similarity suggests that these diseases benefit from similar therapeutic approaches or share common biological pathways.

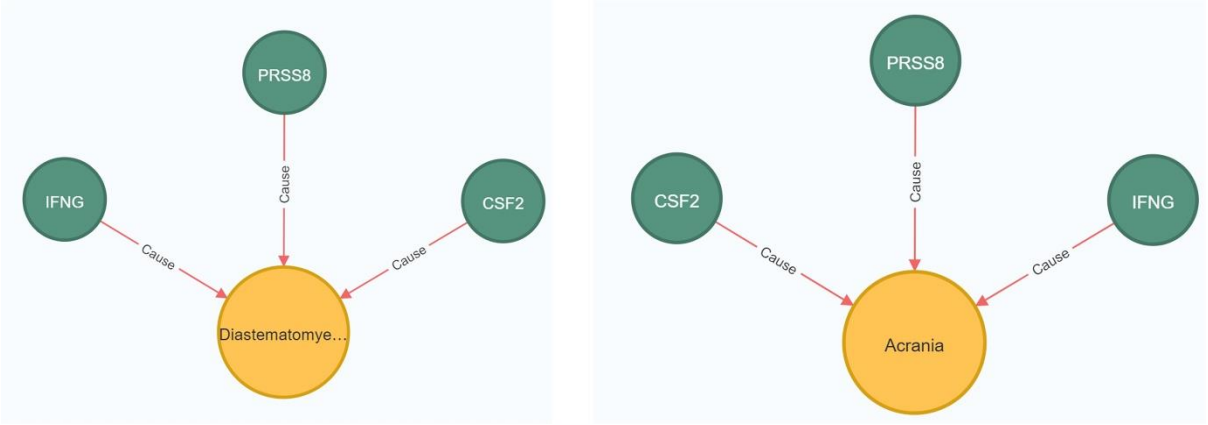

Fig. 11. Genes causing the "Diastematomyelia" and "Acrania" diseases

Table 2. Sample of similarity scores between a couple of diseases.

Disease name	Disease name	Similarity score
Acrania	Diastematomyelia	1
Acrania	Iniencephaly	1
Acrania	Neurenteric Cyst	1
Acrania	Spinal Cord Myelodysplasia	1
Mesothelioma, Cystic	Redness of eye	0.5

Additionally, Fig. 12 illustrates another example with a similarity score of 0.5. Here, "Mesothelioma, Cystic" is caused by the genes "CDKN2A" and "EXOSC1", while "Redness of the Eye" is caused by the "CDKN2A" gene only. The shared gene, "CDKN2A", indicates a partial similarity, i.e., a

similarity score equals 0.5, as illustrated in Table 2. These results demonstrate how genetic overlaps can be identified and analyzed efficiently using a graph database.

Fig. 12. A visualization showing the genes causing the "Mesothelioma, Cystic" disease (left), and the "Redness of eye" disease (right).

d. Gene Ranking

The chart in Fig. **13** presents the results of executing the PageRank algorithm on the TBGA database, highlighting the top ten genes with the highest PageRank scores.

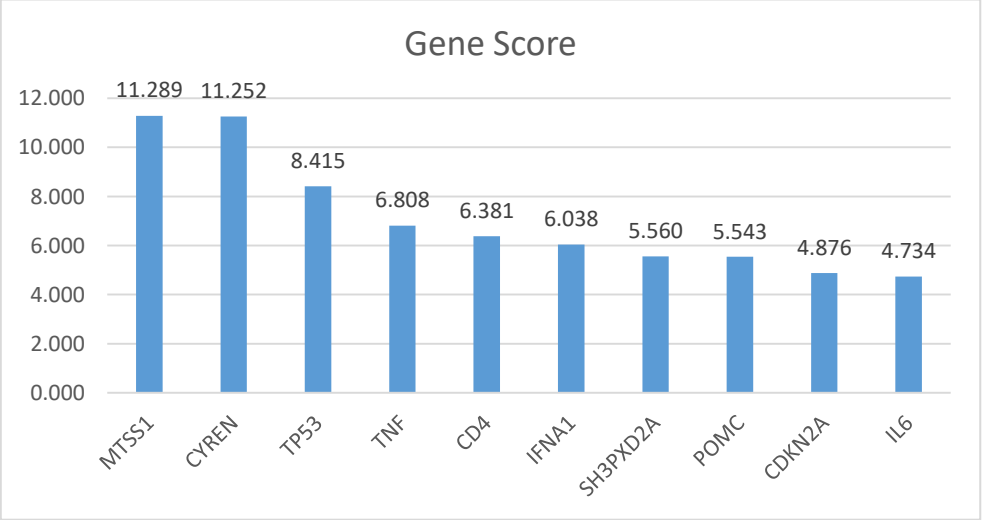

Fig. 13. The top ten genes with the highest PageRank scores.

As shown in Fig. **13**, the top two genes, MTSS1 (Score: 11.289) and CYREN (Score: 11.252) have the highest PageRank scores, suggesting they have significant influence and connections within the TBGA network. Their high scores indicate strong relationships with other essential genes and potentially crucial roles in multiple disease pathways. TP53 (Score: 8.415), TNF (Score: 6.808), and CD4 (Score: 6.381) genes also rank highly, reflecting their critical roles in gene-disease associations. TP53 is well-known for its involvement in cancer [24], while TNF and CD4 are pivotal in immune response regulation [25] [26]. Although IFNA1 (Score: 6.038), SH3PXD2A (Score: 5.560), POMC (Score: 5.543), CDKN2A (Score: 4.876), and IL6 (Score: 4.734) genes have lower scores compared to the top three, they still hold significant positions in the network. IL6, for instance, is a crucial mediator in inflammatory responses [27].

6. Conclusions

RDBMSs have been the standard for data management for decades, but the increasing complexity of modern applications highlights their limitations in handling intricate relationships. GDBMSs, with their graph structures, offer powerful tools for managing relational and graph data, blurring the lines between the two. The proposed hybrid approach leverages the strengths of both models, SQL Server's robust querying capabilities and Neo4j's advanced graph analytics. This hybrid approach, in which each model excels in specific areas, enhances the accuracy and scalability of automated GDA extraction and moves forward in biomedical research. This paper analyzed TBGA, a semi-structured GDA dataset, where each record is in JSON format. We proposed two importing procedures: the first involves importing the TBGA dataset into a relational model using SQL Server, and the second imports the same dataset into a graph model using Neo4j. A number of SQL queries is employed to demonstrate the strengths of the relational model, such as the use of aggregate functions and the GROUP BY clause, which are efficient in GDA analysis and reporting. Moreover, we utilized Neo4j to visualize all genes that cause a particular disease. In addition, multiple graph algorithms are implemented that are tailored for GDA analysis. For example, we employ the Node Similarity algorithm, which compares a set of nodes based on the nodes they are connected to. Two nodes are deemed to be similar if they share many of the same neighbours. In the GDA context, the Node Similarity algorithm is used to find the similarity score between two specific diseases based on the genes associated with them. In addition, we employed the PageRank algorithm to identify the most influential genes within the network. These analyses can highlight valuable connections and aid in developing more effective treatment protocols. By applying advanced graph techniques, we can gain deeper insights into the relationships between genes and diseases, ultimately contributing to improved biomedical research and healthcare outcomes.

References

[1] T. J. Ballinger, M. E. Sanders, and V. G. Abramson, “Current HER2 testing recommendations and clinical relevance as a predictor of response to targeted therapy,” *Clin. Breast Cancer*, vol. 15, no. 3, pp. 171–180, Jun. 2015.

[2] A. Genovese and M. G. Butler, “The autism spectrum: Behavioral, psychiatric and genetic associations,” *Genes (Basel)*, vol. 14, no. 3, Mar. 2023, doi: 10.3390/genes14030677. Available: http://dx.doi.org/10.3390/genes14030677

[3] D. D. Khudhur and S. D. Khudhur, “The classification of autism spectrum disorder by machine learning methods on multiple datasets for four age groups,” *Measur. Sens.*, vol. 27, no. 100774, p. 100774, Jun. 2023.

[4] C. I. van der Made, M. G. Netea, F. L. van der Veerdonk, and A. Hoischen, “Clinical implications of host genetic variation and susceptibility to severe or critical COVID-19,” *Genome Med.*, vol. 14, no. 1, p. 96, Aug. 2022.

[5] A.-L. Barabási, N. Gulbahce, and J. Loscalzo, “Network medicine: a network-based approach to human disease,” *Nat. Rev. Genet.*, vol. 12, no. 1, pp. 56–68, Jan. 2011.

[6] Z. J. M. Ameen and S. S. Samaan, “A WEB BASED APPLICATION FOR CLINICAL LABORATORY INFORMATION MANAGEMENT SYSTEM,” *J. eng. sustain. dev.*, vol. 24, no. 6, pp. 127–136, Nov. 2020.

[7] J. Yan, Y. Meng, L. Lu, and L. Li, “Industrial big data in an industry 4.0 environment: Challenges, schemes, and applications for predictive maintenance,” *IEEE Access*, vol. 5, pp. 23484–23491, 2017.

[8] S. Ramzan, I. S. Bajwa, R. Kazmi, and Amna, “Challenges in NoSQL-based distributed data storage: A Systematic Literature Review,” *Electronics (Basel)*, vol. 8, no. 5, p. 488, Apr. 2019.

[9] G. Gricourt, T. Duigou, S. Dérozier, and J.-L. Faulon, "neo4jsbml: import systems biology markup language data into the graph database Neo4j," *PeerJ*, vol. 12, p. e16726, Jan. 2024.

[10] "A survey of graph techniques applied in Software Defined Networking," *Iraqi Journal of Computer, Communication, Control and System Engineering*, pp. 115–124, Sep. 2023.

[11] P. Kotiranta, M. Junkkari, and J. Nummenmaa, "Performance of graph and relational databases in complex queries," *Appl. Sci. (Basel)*, vol. 12, no. 13, p. 6490, Jun. 2022.

[12] S. Marchesin and G. Silvello, "TBGA: a large-scale Gene-Disease Association dataset for Biomedical Relation Extraction," *BMC Bioinformatics*, vol. 23, no. 1, p. 111, Mar. 2022.

[13] J. F. Naughton, "DBMS: Lessons from the first 50 years, speculations for the next 50," in *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, IEEE, 2010. doi: 10.1109/icde.2010.5447648. Available: http://dx.doi.org/10.1109/icde.2010.5447648

[14] H. R. Vyawahare, P. P. Karde, and V. M. Thakare, "Hybrid database model for efficient performance," *Procedia Comput. Sci.*, vol. 152, pp. 172–178, 2019.

[15] Y. Cheng, P. Ding, T. Wang, W. Lu, and X. Du, "Which category is better: Benchmarking relational and graph database management systems," *Data Sci. Eng.*, vol. 4, no. 4, pp. 309–322, Dec. 2019.

[16] I. K. Subagja *et al.*, "Evaluation of big data analytics in medical science," *Int. J. Eng. Adv. Technol.*, vol. 8, no. 6s3, pp. 717–720, Nov. 2019.

[17] S. A. C. Bukhari, S. Pawar, J. Mandell, S. H. Kleinstein, and K.-H. Cheung, "LinkedImm: a linked data graph database for integrating immunological data," *BMC Bioinformatics*, vol. 22, no. Suppl 9, p. 105, Aug. 2021.

[18] J. Schäfer, M. Tang, D. Luu, A. K. Bergmann, and L. Wiese, "Graph4Med: a web application and a graph database for visualizing and analyzing medical databases," *BMC Bioinformatics*, vol. 23, no. 1, p. 537, Dec. 2022.

[19] P.-T. Lai, C.-H. Wei, L. Luo, Q. Chen, and Z. Lu, "BioREx: Improving biomedical relation extraction by leveraging heterogeneous datasets," *J. Biomed. Inform.*, vol. 146, no. 104487, p. 104487, Oct. 2023.

[20] Shao *et al.*, "LNPL-MIL: Learning from Noisy Pseudo Labels for Promoting Multiple Instance Learning in Whole Slide Image," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2023, pp. 21438–21438.

[21] J. Piñero *et al.*, "DisGeNET: a comprehensive platform integrating information on human disease-associated genes and variants," *Nucleic Acids Res.*, vol. 45, no. D1, pp. D833–D839, Jan. 2017.

[22] J. Chen, Q. Song, C. Zhao, and Z. Li, "Graph database and relational database performance comparison on a transportation network," in *Communications in Computer and Information Science*, in Communications in computer and information science. Singapore: Springer Singapore, 2020, pp. 407–418.

[23] P. Csermely, T. Korcsmáros, H. J. M. Kiss, G. London, and R. Nussinov, "Structure and dynamics of molecular networks: a novel paradigm of drug discovery: a comprehensive review," *Pharmacol. Ther.*, vol. 138, no. 3, pp. 333–408, Jun. 2013.

[24] J. Hu *et al.*, "Targeting mutant p53 for cancer therapy: direct and indirect strategies," *J. Hematol. Oncol.*, vol. 14, no. 1, p. 157, Sep. 2021.

[25] S. L. Montgomery and W. J. Bowers, "Tumor necrosis factor-alpha and the roles it plays in homeostatic and degenerative processes within the central nervous system," *J. Neuroimmune Pharmacol.*, vol. 7, no. 1, pp. 42–59, Mar. 2012.

[26] S. Yang, J. Wang, D. D. Brand, and S. G. Zheng, "Role of TNF–TNF receptor 2 signal in regulatory T cells and its therapeutic implications," *Front. Immunol.*, vol. 9, Apr. 2018, doi: 10.3389/fimmu.2018.00784. Available: http://dx.doi.org/10.3389/fimmu.2018.00784

[27] A. Shahini and A. Shahini, "Role of interleukin-6-mediated inflammation in the pathogenesis of inflammatory bowel disease: focus on the available therapeutic approaches and gut microbiome," *J. Cell Commun. Signal.*, vol. 17, no. 1, pp. 55–74, Mar. 2023.