NATIONAL INSTITUTE OF TECHNOLOGY RAIPUR

ADVANTAGES AND SECURITY IN GRAPH DATABASE MANAGEMENT SYSTEM

Shreedutt Dixit 19111056
6th Semester, DBMS
Biomedical Engineering Department
Email: Shreedutt77@gmail.com

Guided by :
Saurabh Gupta Sir

Abstract

The rapid development of the Internet has drawn a lot of attention to network security situation awareness. Network security situation awareness data in large-scale complex networks has the characteristics of being large-scale, multi-source, and heterogeneous. Much study has been done recently on network security situation awareness. The majority of existing solutions, on the other hand, store different types of data in distinct ways, making data query and analysis inefficient.

We propose a graph database-based hierarchical multi-domain network security situation awareness data storing solution to overcome this challenge. We propose a hierarchical multi-domain network security situation awareness model to partition the network into multiple domains, allowing for more effective collection and disposal of awareness data.

Meanwhile, we create network security situation awareness data storage rules and procedures based on graph databases to unify our storage mode. Finally, rigorous tests on real datasets demonstrate that our suggested strategy is more efficient than current storage models.

TABLE OF CONTENT

1 Introduction to Graph Database Management System

A graph database is a single-purpose, specialised platform for designing and controlling graphs. Graphs are comprised of nodes, edges, and properties, which are all used to symbolise and hold data in a way that relational databases can't.
Another commonly used term is graph analytics, which refers to the process of analysing data in a graph format with data points acting as nodes and relationships acting as edges.
A database that can support graph formats is required for graph analytics; this can be a dedicated graph database or a converged database that supports multiple data models, including graph.
Property graphs and **RDF** graphs are two popular graph database models.
The property graph is more concerned with analytics and querying, whereas the RDF graph is more concerned with data integration. Both types of graphs are made up of a set of points (vertices) and the connections that connect them (edges).
Graph databases address large problems that many of us face on a daily basis. Modern data problems frequently involve many-to-many relationships with heterogeneous data, which necessitates the following:

- Navigate deep hierarchies
- Discover hidden connections between distant items
- Find out how items are related to one another.

Whether it's a social network, a payment network, or a road network, everything is an interconnected graph of relationships. And when we want to ask questions about the real world, many of them are about relationships rather than individual data elements.

1.1 Property graphs

Property graphs are used to model data relationships and enable query and data analytics based on these relationships. A property graph is made up of vertices that contain detailed information about a subject and edges that represent the relationship between the vertices. The vertices and edges can have attributes, known as properties, that are associated with them.
Property graphs are used in a wide variety of industries and sectors due to their versatility, including finance, manufacturing, public safety, retail, and many others.

1.2 RDF graphs

RDF graphs (RDF stands for Resource Description Framework) adhere to a set of W3C (Worldwide Web Consortium) standards designed to represent statements and are best suited to representing complex metadata and master data. They are frequently used in the context of linked data, data integration, and knowledge graphs. They can represent complex domain concepts or provide rich semantics and inference on data.
A statement is represented in the RDF model by three elements: two vertices connected by an edge that reflect the subject, predicate, and object of a sentence—this is known as an RDF triple. A unique URI, or Unique Resource Identifier, identifies each vertex and edge. The RDF model enables information exchange by allowing data to be published in a standard format with well-defined semantics. RDF graphs have been widely adopted by government statistics agencies, pharmaceutical companies, and healthcare organisations.

2 NEO4J GRAPH DATABASE

Neo4j is an open source, high-performance graph database that describes its data model using graph-related concepts. Node, relationship, property, and label are the four core data elements of Neo4j. Neo4j is a high-availability, fault-tolerant, and scalable cluster that can store hundreds of trillions of items and can be used in the enterprise. Neo4j can be used for large-scale data storage, query, backup, and redundancy, and it also possesses the qualities of atomicity, consistency, isolation, and durability (ACID).

At the same time, it supports Cypher, a declarative graph database query language that is both expressive and efficient. Furthermore, Cypher is so scalable that users can easily design their own query methods.

Organizations nowadays capture massive amounts of data for future study. In recent years, a slew of non-relational databases (NoSQL) have sprung out in response to the massive amounts of data and Web2.0 requirements. However, numerous security features found in relational databases (such as access control) have been left in non-relational management systems to be built by the application, thereby exposing the application to security risks.

This study provides a security model for a NoSQL graph-oriented database management system that is based on the utilisation of metadata. The purpose is to aid the development of applications that employ graph-oriented databases in maintaining data integrity and protecting it from unauthorised access. As a proof of concept, the model was created and implemented for the Neo4j database in a case study.

The results revealed that access restrictions were correctly imposed, preventing unwanted access. A schema for Neo4j was provided, once it does not have a native one.

3 Graph database-based data storage rules

The node, label, relationship, and property are the four core data structures of Neo4j. Entity information is often stored in the node. Each node can have numerous labels for indexing and some model constraints. The relationship is what ties the nodes together, and there might be several relationships between two nodes in different ways. Both nodes and relationships can have one or more key-value pairs as properties. As a result, we suggest the following modelling principles, which are based on the hierarchical multi-domain NSSA model and the Neo4j data structure:

3.1 Node

A node is a creature or item that interacts with the outside world. All objects that are associated to other nodes are viewed as a single node for ease of processing and data analysis, and the node is given the entity or object's name.

3.2 Label

Because each node can have one or more labels, we utilise the name of the domain or domains as the node label when a node belongs to a certain domain or domains. We can also add the node's category labels for query management at the same time, and label names are all capital letters.

3.3 Property

The information of a node that does not interact with other nodes is called property. The name of the node and the metadata of some required item or object are both considered properties. One or more key-value pairs are used to represent a property.

3.4 Relationship

Communication, subordination, and connection are examples of relationships. The nodes can have one or more separate directed relationships, and neither the start nor the end node can be null. Although each connection can have one or more attributes, it can only have one type, and the relationship type's name must be in capital letters.

4 Conclusion

5 References

1.