ANALYSİS OF MODERN DATABASES

Latafat Kardashova, Sona Karimova
Azerbaijan State Oil and Industry University

Abstract

In the current digital landscape, handling the vast and varied data requires customized processing and storage methods. Within a system, data within the same domain often uses different models, underscoring the importance of multi-model databases that can accommodate diverse data structures. These databases typically differentiate between primary and secondary models, necessitating clear data schema mappings. While the traditional relational data model persists as a well-studied and widely used approach, graph data models are gaining traction in areas like social networks, recommendation systems, and transportation networks. This article offers insights into the pros and cons of graph databases, providing a clearer understanding of their role in modern data management.

Keywords: Graph databases, Database architecture, Graph-based data management, Multi-model DBMS, Relational databases

Introduction

In the realm of next-generation databases, graph databases stand out as specialized tools adept at handling complex relationships and fine-tuning unstructured data. Their prowess extends to a variety of areas, including web analytics, social network analysis, and managing various types of data. Unlike their counterparts, graph databases boast a unique design that eschews traditional table structures, instead offering a column-less, point-to-point architecture that increases both speed and flexibility. However, their skills come at a cost. Effective use of graph databases requires a learning curve and depth of specialized knowledge. Additionally, while they shine in scenarios involving complex data relationships, they may be less optimal for use cases characterized by simple, structured data arrangements. In comparison, other types of next-generation databases also bring their own advantages to the table. For example, document-oriented databases offer extensibility and flexibility in schema design that respond well to unstructured or semi-structured data. Key-value stores excel at high-performance tasks such as caching and session management, although they may lack the depth of query capabilities found in graph databases. Optimized for write-heavy workloads, the column family stores robustness and fault tolerance at scale. Time series databases, on the other hand, are specifically tailored for time-stamped data, offering efficient storage and specialized query capabilities for time-based analysis. In fact, although graph databases are unique in their ability to handle complex relationships and unstructured data, each type of next-generation database brings its own strengths and considerations to the table. The choice ultimately depends on the specific needs of the application, weighing factors such as data structure, scalability, performance requirements, and query patterns. Therefore, I will give detailed information about graph databases in my article. [6]

Graph: An abstract data structure consisting of vertices and edges connecting those vertices. Vertices represent objects or entities, while edges represent connections or relationships between them. Graphs are widely used for modeling and analyzing various systems and networks.

Vertex: An element in a graph representing an object or entity. Each vertex can have a unique identifier and can be connected to other vertices through edges.

Edge: A link or relationship between two vertices in a graph. It can represent a directed or undirected connection between vertices and may have attributes like weight or label to provide additional information.

Directed graph: A graph where each edge has a direction, connecting a start vertex to an end vertex. The direction is indicated by an arrow.
Undirected graph: A graph where edges have no direction, simply connecting two vertices without specifying start and end points.
Degree of a vertex: The number of edges connected to a vertex. In directed graphs, it can be divided into in-degree (incoming edges) and out-degree (outgoing edges).
Path: A sequence of vertices connected by edges. It can be straight (all edges in the same direction) or cyclic (starts and ends at the same vertex).
Cycle: A path that begins and ends at the same vertex. It can be simple (no repeating vertices) or non-simple (repeating vertices).
Connected graph: A graph where there is a path between any two vertices, meaning each vertex can be reached from any other vertex.
Tree: A connected acyclic graph with exactly one path between any two vertices. It has a root vertex from which all other vertices branch out.[12]
Benefits of Using Graph Databases
Graph databases provide numerous advantages over other database types:
Flexibility in Data Modeling: The graph data model enables easy representation of intricate connections and relationships between data. This flexibility allows for convenient handling of diverse data types within a graph database.
Efficiency in Connection Management: Graph databases excel in executing queries related to searching and analyzing relationships between data. Leveraging specialized algorithms and data structures, they swiftly identify paths and connections between graph vertices.
Flexibility in Queries: Graph databases boast a robust query language, simplifying and enhancing the flexibility of data retrieval. This language empowers users to define intricate conditions and filters for pinpointing desired data.
Analysis of Connections and Relationships: Graph databases provide potent tools for analyzing connections and relationships within data. Leveraging graph algorithms and operations, users can unearth paths, cycles, subgraphs, and other structures, revealing hidden connections and patterns in the data.
Overall, graph databases offer flexibility, efficiency, and robust analytical capabilities for managing data with complex connections and relationships. Their versatility finds application across various domains, including social networks, recommender systems, bioinformatics, and more.
Challenges When Using Graph Databases. When designing and utilizing graph databases, several important problems and challenges may arise, which merit careful consideration:
Scalability:Graph databases often confront scalability issues, particularly when dealing with massive graphs containing millions or even billions of nodes and links. Processing such extensive graphs can be intricate and computationally demanding. Hence, selecting a graph database that efficiently scales and manages large data volumes is crucial.
Query Complexity: Another challenge involves the complexity of queries against graph databases. These queries can be intricate, necessitating specialized query languages like Graph Query Language (Cypher) or SPARQL-based query languages. Understanding and crafting such queries might prove challenging for developers, especially those unfamiliar with graph databases.
Dynamic Graph Structure: The structure of a graph can evolve over time, posing challenges for graph databases. Adding, deleting, or altering nodes and relationships may mandate updating the entire database, consuming significant time and resources. Thus, opting for a graph database that facilitates efficient graph structure updates is essential.
Complex Analysis: Analyzing graph data can be complex, requiring specialized algorithms and techniques. Certain operations, such as finding the shortest path or identifying communities in a graph,

can be computationally intensive, demanding substantial computational resources. Hence, selecting a graph database that offers efficient algorithms and operations for analyzing graph data is paramount.
In essence, while utilizing graph databases may present challenges, judicious selection of databases and adept use of graph algorithms and operations can mitigate these issues, rendering graph databases potent tools for storing and analyzing data relationships.
Examples of Using Graph Databases
Social Media: Graph databases play a vital role in social networks by storing and analyzing connections between users. For instance, on Facebook, a graph database manages user profiles, friend connections, likes, comments, and interactions. This facilitates complex queries like finding friends of friends, identifying influential users, and analyzing group connections. [4]
Recommender Systems: Graph databases are integral to recommender systems, offering personalized recommendations. In platforms like Netflix, a graph database holds data on movies, actors, directors, and user preferences. This enables finding similar movies, exploring relationships between actors and directors, and offering recommendations based on user preferences and similar interests Bioinformatics: In bioinformatics, graph databases store and analyze genetic data. For instance, a graph database might represent an organism's genome, with vertices representing genes and edges depicting connections between genes. This aids in identifying genes linked to specific diseases or traits and analyzing evolutionary relationships between organisms.Transport Networks: Graph databases model and analyze transportation networks such as roads, railroads, and airlines. For instance, a graph database contains data on cities, roads, and transportation schedules, enabling route optimization, traffic scheduling, and network capacity analysis. [9] Financial Systems: Graph databases are utilized in financial systems to analyze relationships between customers, transactions, and other financial entities. For instance, a graph database manages customer profiles, accounts, transactions, and their interconnections. This facilitates identifying fraudulent activities, analyzing financial flows, and building risk models. Graph Database Models
Graph databases organize data in a graph format comprising vertices and edges, with vertices representing entities and edges denoting connections between these entities. Various graph database models govern data organization and storage.
Ownership Model: In the ownership model, each vertex possesses a property defining its type. Edges signify relationships between vertices and can also possess properties. This model accommodates complex data structures like hierarchies, networks, and graphs.
Tag Model: The tag model assigns labels to each vertex and edge, specifying their types. Labels aid in classifying and filtering data. For instance, in a social network, vertices might carry labels such as "user," "group," or "event," while edges could be labeled "friendship," "subscription," or "participation."
Property Model: In the property model, vertices and edges can possess properties, represented as key-value pairs. These properties store additional information about entities and relationships. For example, in a social network graph, user nodes might feature properties like "name," "age," and "location."
Graph Database Model with Labels and Properties: Some graph databases merge label and property models, allowing for a flexible classification and storage of data. This enables more adaptable organization and structuring of information within the graph. Each graph database model offers its own set of advantages and drawbacks. The selection of a model hinges on the specific requirements and characteristics of the project at hand. Algorithms and Operations in Graph Databases
Graph databases offer a plethora of algorithms and operations for managing graph data effectively. Here are some key ones:

Path Finding: A fundamental operation involves finding paths between vertices. This facilitates tasks like identifying the shortest path between two points or discovering all paths meeting specific criteria.
Graph Traversal: Traversal entails visiting each vertex of the graph. It aids in tasks such as exploring connected vertices or executing operations on each vertex within the graph.
Connectivity Analysis: This analysis assesses the connectivity of a graph, determining its degree of connectedness and identifying isolated vertices.
Centrality Calculation: Centrality measures the importance of a vertex in the graph. Algorithms compute various centrality measures like degree, closeness, and betweenness centrality.
Clustering: Clustering involves grouping graph vertices based on similarity, facilitating tasks like community detection in social networks or structural analysis of graphs.
Recommendations: Recommendation operations leverage graph analysis to offer personalized suggestions to users. This could include friend recommendations in social networks or product recommendations based on past purchases.
These represent just a few of the algorithms and operations available in graph databases. Each database may offer its own set of algorithms and operations, in addition to supporting standard ones. Comparison of Graph Databases with Other Database Types
Graph databases stand out from relational databases and document-oriented databases in their efficient handling of data relationships. Here are some key distinctions and advantages of graph databases:
Data Model: Relational databases use tables, with rows representing entities and columns representing attributes. Document-oriented databases store data as documents, which can be hierarchical. Graph databases represent data as nodes and edges, with nodes symbolizing entities and edges denoting relationships between them.
Processing Links: Graph databases excel in efficiently managing data relationships. Unlike relational databases requiring complex JOIN operations, graph databases employ simple graph traversal methods like breadth-first or depth-first traversal for connection searches.
Model Flexibility: Graph databases offer greater data model flexibility compared to relational databases. While relational databases demand predefined and rigid schemas, graph databases allow dynamic schema modifications, enabling easy addition of new node and edge types without altering the schema.
Query Complexity: In relational databases, query complexity escalates with increasing data relationships. Conversely, in graph databases, query complexity remains tied to the number of nodes and edges, not the links. This renders graph databases more efficient for executing intricate queries involving relational analysis. Performance: Graph databases exhibit superior performance in relationship analysis operations like shortest path searches or finding related nodes. They excel in tasks such as social media recommendations and analysis. However, for non-relationship analysis operations, relational databases may offer better performance. [12] In summary, graph databases offer a robust solution for storing and processing data relationships. They prove particularly valuable when relationship analysis is pivotal or when dealing with data featuring complex relationships.

Conclusion

Graph databases are a potent tool for storing and analyzing data based on graph theory. They enable effective modeling and analysis of intricate relationships between objects, spanning domains like social networks, transport networks, and family trees. A key advantage of graph databases lies in their adeptness at handling queries related to path finding and connection analysis among graph nodes. This makes them invaluable for tasks requiring optimal route discovery, social connection analysis, recommendation systems, and more. Nevertheless, employing graph databases may pose challenges such as data modeling complexity, query optimization, and managing large datasets. However, advancements in technology and

tools for graph database management are steadily addressing these hurdles. In essence, graph databases serve as a vital tool for dissecting and processing complex data relationships. Their utilization can yield more efficient and precise outcomes across diverse fields.

References

[1] "Next Generation Databases" by Guy Harrison
[2] "InavoluGRAPH DB WITH NEO4J" by Satya Srikanth Inavolu
[3] "Graph Database Modeling With Neo4j" by Ajit Singh
[4] https://bigdataschool.ru/blog/graph-database-under-the-hood.html
[5] https://appmaster.io/ru/blog/arkhitektura-programmnogo-obespecheniia-grafovykh-baz-dannykh
[6] https://www.oracle.com/autonomous-database/what-is-graph-database/
[7] https://www.decube.io/post/graph-database-concept
[8] "Graph Databases" - Lan Robinson, Jim Webber & Emil Eifrem
[9] "Graph Databases For Beginners" - Bryce Merkl Sasaki, Joy Chao & Rachel Howard
[10] "Graph Databases" -Adrian Silvescu, Doina Caragea, Anna Atramentov
[11] https://ericvanier.com/next-generation-database-architectures-for-scale-and-performance/
[12] https://ericvanier.com/next-generation-data-management-trends-and-opportunities-in-2023/
[13] https://wiki.merionet.ru/articles/chto-takoe-grafovaya-baza-dannyh
[14] https://aws.amazon.com/ru/compare/the-difference-between-graph-and-relational-database/

DİFFİE-HELMAN ALQORİTMİ VƏ ONUN TƏTBİQİ

Fidan Paşayeva
"R.İ.S.K. Elmi İstehsalat Şirkəti" Qapalı Səhmdar Cəmiyyəti

Xülasə

Əsas məqsəd, məlumatın təhlilində, özünü qorumaq və mübahisəsiz məlumatı mübadilə etmək üçün təhlükəsiz bir platform yaratmaqdır. Ənənəvi kriptoqrafiyanın əsas problemlərindən biri, məlumatı açıq şəkildə göndərən və qəbul edən tərəflərin müxtəlifliklərı ilə əlaqədar olan təhlükələrdir. Diffie-Hellman alqoritmi, bu problemləri həll etmək üçün inkişaf etdirilmiş bir protokoldur. Sözügedən alqoritmi, açıq şifrələmə əsasında əsaslanır və iki tərəf arasında gizli açarları mübadilə etmək üçün istifadə olunur. Bu alqoritmin əsas fikri, bir məlumatın qorunmasında məlumatın özünü deyil, özü üçün gizli olan açarların bir müddət sonra dəyişməsi ilə əlaqəlidir. Bu, qarşı tərəf məlumatı şifrələmək və açmaq üçün birbirilərindən asılı olan açarları yaratmaq imkanı verir. Diffie-Hellman alqoritmi, məlumatın müəyyən edilməsinin asanlaşdırılması ilə əlaqədar olan ənənəvi kriptoqrafiya texnikalarından fərqlənir. Bu alqoritmin əsas qüsurları arasında həmin məlumatın qarşı tərəf tərəfindən mənimsənilməsinin qeyri-mümkünlüyü və ələ keçirilmə riski sayılmaqdadır. Əksinə, həmin haqqında söz açılan alqoritmi, təhlükəsiz şifrələməni təmin etmək və gizliliyi qorumaq üçün innovativ bir yanaşma təklif edir. Bu konfrans, Diffie-Hellman alqoritminin əsas prinsiplərini, işləmə mexanizmini və təhlükəsizliyini izah edəcək. Həmçinin, alqoritmin mövcud tətbiq sahələri və müasir məqamları da müzakirə olunacaq. Bu konfrans, kriptoqrafiya sahəsində yenilikləri axtarış və təşviq etməyə həsr edilmişdir. Məqaləmin mövzusu "Diffie-Hellman Alqoritmi və Onun Tətbiqi". Bu alqoritmin təsviri və əsas prinsipləri təqdim olunacaq. İlk olaraq, Diffie-Hellman alqoritminin məqsədi və əhəmiyyəti açıqlanacaq. Sonra, əsas