다형 IoT 데이터 관리를 위한 관계형 데이터베이스와 그래프 데이터베이스 비교

뉘엔반퀴엣, 김경백
전자컴퓨터공학부, 전남대학교
e-mail : quyetict@utehy.edu.vn, kyungbaekkim@jnu.ac.kr

Comparison of Relational Databases and Graph Databases for Heterogeneous IoT Data Management

Van-Quyet Nguyen, Kyungbaek Kim
Dept. Electronics and Computer Engineering, Chonnam National University

Abstract

Nowadays, the Internet of Things (IoT) platforms have been widely deployed in many domains, such as smart buildings or smart agriculture. The major challenges of these platforms are storing and analyzing data from a huge number of 'things' which are highly connected to each other. This requires IoT platforms to have flexible data models and efficient data access mechanisms to guarantee the performance of the system. Traditional IoT platforms often use relational databases which are mature technologies for structured data. However, using a relational database is insufficient for handling heterogeneous IoT data (e.g., semi-structured, unstructured) due to complex relationships which require multiple nested queries and complex joins on multiple tables. Recently, graph database which is a kind of NoSQL databases, have emerged from the modern IoT platforms. This paper presents a comprehensive comparison of relational databases and graph databases for heterogeneous IoT data management. Through the comparison in various aspects and the experimental results, we find that graph databases are applicable for storing and analyzing the IoT connected data.

1. Introduction

Recent years, IoT products and services are emerging in various IoT application areas [1][2]. They not only make our life convenient, comfortable, and safe, but also enhance productivity and create new added-values for industrial business. For example, in a smart building, a smart evacuation system exploring data from smart devices (e.g., smart indicator, smart cameras, and smart sensors) can provide the efficient routes to evacuees in an emergency situation [3]. The devices are going to be connected in highly connected fashion [4][5]. Choosing a proper database that supports easy to manage and explore the connection between 'things' is particularly evident for heterogeneous IoT data management.

Traditional IoT platforms often use relational databases (e.g., MySQL, MSSQL, MariaDB) which are well-documented and mature technologies. However, using a relational database is insufficient for managing heterogeneous IoT data (e.g., structured, semi-structured, and unstructured) due to complex relationships which require nested queries and complex joins on multiple tables. Therefore, it is hard to guarantee the performance of querying massive and highly connected data in IoT applications.

Recent years, non-relation (NoSQL) databases have emerged as a popular alternative to relational databases, which allow representing unstructured and semi-structured data in a schema-free way. There are varied types of NoSQL databases including key-value, column-family, document, and graph databases. Among them, graph database is one of the most popular databases used by enterprises.

In this paper, we present a comprehensive comparison of relational databases and graph databases for heterogeneous IoT data management. We first present the characteristics of IoT data and the challenges of IoT data management. We then compare relational databases and graph databases in various aspects including data model, query performance, transaction support, and scalability. Finally, we evaluate their performance in querying data of real dataset 'Gnutella' that is an Internet peer-to-peer network. Through the comparison, we find that graph databases are applicable for storing and analyzing the IoT connected data.

2. IoT Data Characteristics and Challenges in Data Management

This section presents the major characteristics of IoT data as well as challenges of IoT data management.

2.1 Heterogeneity

With different 'things' in IoT depending on types of devices and services, IoT platforms generate different kinds of data including structured, semi-structured, and unstructured data. For example, in an IoT platform for smart building management, the data generated from many kinds of devices (e.g., temperature sensors, smoke sensors, cameras, etc) are unstructured; meanwhile, information about people or rooms in the building can be managed in structured data using data tables or in semi-structured data using XML files. Thus, how to manage heterogeneous IoT data so that they can be easily explored in IoT applications is considered as a challenge.

2.2 Highly Connected Data

In an Internet of Thing (IoT) environment, entities with different attributes and capacities are going to be connected in a highly connected fashion. Specifically, not only the mechanical and electronic devices but also other entities such as people, locations and applications are connected to each other. Understanding and managing these connections between things are challenges in IoT data management, which plays an important role for businesses.

2.3 Dynamic Changes

Most IoT applications work with dynamic and speedily changing data due to new entities are coming online and/or the connection between these entities can change regularly. This requires a data model to easily represent the entities, and support adding, deleting and updating relations between entities without impacting application availability.

2.4 Massive Real-time Data

Massive IoT data is generated every minute through multiple kinds of devices and services such as sensors and social networking. For instance, a huge number of images is generated in real-time through cameras in a smart building, which is unsuitable to reside in a traditional relational database (e.g., MySQL, SQL Server). Therefore, modeling such kind of data in an easy way and real-time processing are considered as challenges.

3. Comparison of Relational Databases and Graph Databases

3.1 Data Model

Relational databases use fixed-schema by predefined tables with rows and columns for data storage. This makes them ill-suited for storing unstructed or semi-structured IoT data. Meanwhile, graph databases use schema-free by using graph models, in which, nodes are used to represent entities and edges represent the relationship between entities. This makes them easy to describe heterogeneous data and highly connected data.

Relational databases describe relationship between things by using standard relations: one-to-one, one-to-many, and many-to-many. A lot of mapped tables are generated with primary keys and foreign keys to ensure the consistency of the data. This causes difficulties to deal with the dynamic changing characteristics of IoT data. With graph databases, adding and removing entities and their relations are straightforward operations.

3.2 Query Performance

Relational databases use Structured Query Language (SQL) statements for accessing the data. SQL language is well-defined and common in both academic and business. However, relational databases are not designed to handle the massive data and highly connected data which are characteristics of IoT data in modern applications. Hence, the query performance could be low due to the possible multiple nested queries or a large number of joins from multiple tables. In contrast, graph databases are purposely-built to store and handle highly connected data; therefore, they can obtain high performance in querying IoT data. The key point that helps graph databases to achieve high performance is applying graph traversal techniques such as BFS and DFS. Meanwhile, relational databases use scanning and hash matching techniques which are costly with large tables or lots of joins. Thus, the query performance of relational databases decreases when increasing the number of records in tables and the number of relationships among tables (related tables), while with graph databases, its performance only decreases in relation to the number of connections between the records (things).

3.3 Transaction Support

An important function of relational databases which make them as the preferred choice in industries and applications is ACID (Atomic, Consistent, Isolated, Durable) transaction support. The ACID properties provide a mechanism to ensure that data of a transaction does not corrupt as a result of failure with any reason (e.g., transaction of transferring money

between accounts in the bank). While most of NoSQL databases use BASE (Basic Availability, Soft-state, Eventual consistency) consistency model to support transactions in their databases, current graph databases (e.g., Neo4j, OrientDB) retain ACID properties which are required by modern IoT applications where data reliability and consistency are essential.

3.4 Scalability

To deal with massive data, the scalability is critical important in IoT data management. Relational databases use vertically-scalable which means that improving the performance of handling massive data is performed by upgrading the storage and compute capacity (e.g., using SSD, increasing number of CPU cores, etc) of existing hardware in the system. Vertical scaling is typically expensive, and the fault-tolerant of the system is not guaranteed as a result of a single database server failure. On the other hand, graph databases use horizontally-scalable which means that when IoT data is rapidly growing up, we add more resources (e.g., server machines) to the system for scaling storage and improving query performance.

4. Evaluation

To compare the query performance between relational databases and graph databases, we do an experiment on a real-life dataset called Gnutella that is an Internet peer-to-peer network [6]. This dataset has 62,586 nodes representing hosts in the Gnutella network topology and 147,892 edges representing connections between nodes, which correspond the number of records relational database. For the relational database, we use MySQL to design a table named "Link" with two columns *FromNodeId* and *ToNodeId* to store Gnutella dataset. For the graph database, Neo4J is used to store the dataset, in which the relationship between two nodes is defined with the name *"CONNECT"*. To evaluate the impact of the size of dataset to the query performance, we create a varied size of Gnutella by cutting down data from the original one as shown in Table 1.

Table 1. Summary of dataset

Dataset	Nodes	Edges (Records)
G1	10,000	12,875
G2	20,000	29,253
G3	30,000	49,371
G4	40,000	74,481
G5	50,000	104,288
G6	60,000	138,142

We evaluate query performance for solving two particular types of problem. The first one is "*finding out all connections from a given host to its neighbours*". The second one is "*finding out all connections from a given host in four or fewer hops*", which is a common problem in the IoT connected networks where hosts having a large number of connections to others could be used for some purposes such as spreading messages. We can see that the first problem can be done with the simple queries as shown in Figure 1. Meanwhile, the second one requires the complex queries, in which, relational databases need to do multiple nested SQL queries that is illustrated in Figure 2(a), and graph databases use a bounded recursive query as shown in Figure 2(b). Here, we test with the given host identity being equal to '60'.

Figure 3 illustrated the query performance comparison between relational databases and graph databases with a simple query. We observed that the simple graph query is slightly faster than the simple SQL query. The reason is the relational databases perform scanning all records in the 'Links' table for matching with the

```
SELECT * FROM links WHERE FromNodeId='60'
```

(a) Simple SQL Query

```
MATCH path = (a:Host{HostId:'60'})-[CONNECT]->(b)
RETURN path;
```

(b) Simple Graph Query

Figure 1. Simple queries in relational databases and graph databases

```
SELECT * FROM Links WHERE FromNodeId IN
(SELECT ToNodeId FROM Links WHERE FromNodeId IN
    (SELECT ToNodeId FROM Links WHERE FromNodeId IN
        (SELECT ToNodeId FROM Links WHERE FromNodeId = '60')))
UNION
SELECT * FROM Links WHERE FromNodeId IN
    (SELECT ToNodeId FROM Links WHERE FromNodeId IN
        (SELECT ToNodeId FROM Links WHERE FromNodeId = '60'))
UNION
SELECT * FROM Links WHERE FromNodeId IN
        (SELECT ToNodeId FROM Links WHERE FromNodeId = '60')
UNION
    SELECT * FROM Links WHERE FromNodeId = '60'
```

(a) Complex SQL Query

```
MATCH path = (a:Host{HostId:'60'})-[*1..4]->(b)
RETURN path;
```

(b) Complex Graph Query

Figure 2. Complex queries in relational databases and graph databases

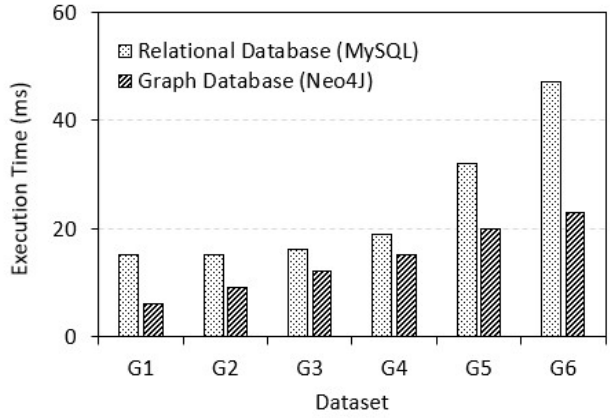

Figure 3. Query performance comparison between relational databases and graph databases with a simple query

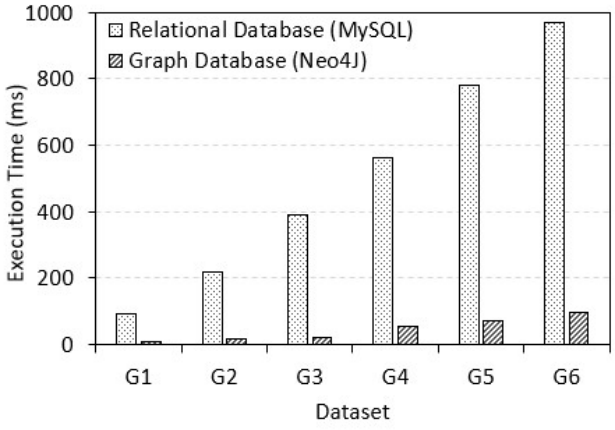

Figure 4. Query performance comparison between relational databases and graph databases with a complex query

given condition; meanwhile, the graph databases perform finding out a node with the given condition and search from that node with one hop in the graph. However, the query performance of graph databases is around 100 times faster than the one of relational databases in the case of complex query, as shown in Figure 4. It is not difficult to find that with the SQL complex query in Figure 2(a), the 'Links' table need to be scanned by 10 times. That is the reason for the long time response of the relational databases in querying IoT connected data.

5. Conclusion

This paper presented a comprehensive comparison of relational databases and graph databases for heterogeneous IoT data management. We compared them through various characteristics including data model, query performance, transaction support, and scalability. We also evaluated their query performance on real IoT dataset, Gnutella. Through the comparison in various aspects and the experimental results, we found that graph databases are applicable for storing and analyzing the IoT connected data.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2017R1A2B4012559). This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2016-0-00314) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

[1] Lee, I. and Lee, K., 2015. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business Horizons, 58(4), pp.431-440.

[2] Van-Quyet, Nguyen, et al. "Design of a Platform for Collecting and Analyzing Agricultural Big Data." JDCS vol. 18, no.1, pp. 149-158, 2017.

[3] Lin, C.Y., Chu, E., Ku, L.W. and Liu, J., 2014. Active disaster response system for a smart building. Sensors, 14(9), pp.17451-17470.

[4] Arora, Vaibhav, Faisal Nawab, Divyakant Agrawal, and Amr El Abbadi. "Multi-representation based data processing architecture for IoT applications." In Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on, pp. 2234-2239. IEEE, 2017.

[5] Van-Quyet Nguyen, Huu-Duy Nguyen, Giang-Truong Nguyen, Kyungbaek Kim, "A Graph Model of Heterogeneous IoT Data Representation: A Case Study from Smart Campus Management", In Proceedings of KISM Fall Conference 2018.

[6] M. Ripeanu and I. Foster and A. Iamnitchi. "Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design". IEEE Internet Computing Journal, 2002.