Analysis of Film Data Based on Neo4j

Huiling Lu, Zhiguo Hong, Minyong Shi
School of Computer, Faculty of Science and Engineering,
Communication University of China,
Beijing, 100024, China
E-mail: 18763823073@163.com

***Abstract*—Film data analysis is important for website to find relationship between film data. Other than normal relational database, a novel and popular graph database, Neo4j uses graph related concepts to describe data models, the data can easily be nodes, edges and their associated attributes. By focusing on film data, Neo4j-based analysis is conducted in this paper. Firstly, Neo4j and Cypher Query Language are introduced. Then Neo4j is applied to analyze the associations among key objects in film data which are directors, actors etc. Neo4j database is good at dealing with complex and multi-connection data, using Neo4j database to store and manage film data makes it convenient for film data analysis.**

Keywords—Neo4j; film data; Cypher

I. Introduction

With the development of the Internet, film data growth rapidly, the relationship between film data become more and more complicated. The relationships between movies, actors and writers are important information for both film producers and audiences. The film data website not only need to store movie videos, they also need to store information about directors, writers, actors etc.

If such data is stored in a relational database, the connections among different tables can be stated via foreign key. However, when there are a lot of relationships, the traditional relational database not only has a large amount of data redundancy, but also become difficult to update dynamically. In addition, it’s hard to query complex relationships between two entities. For example, when you want to find out whether the main producers of the two films have a co-production of other films, you need to find other movies produced by the main producers of the two films, and then analyze whether there is an intersection between them.

Therefore, non-relational database is a wise choice for investigating and processing film data. Neo4j, which is an excellent graph database tool, stores data in the form of graph, which can represent objects with nodes, edges and properties. Consequently, it is suitable to store complex and dynamic relationships among objects of film data.

II. Graph database—NEO4J

To solve the problem of data storage in computer filed, we need to use different storage technologies. In many storage technologies, relational databases have long been dominant. With the application of Wed 2.0, the rise of social networks, the internal data dependence and complexity increase gradually, more and more problems arise in relational databases. Then, graph database appeared. In recent years there have been a number of high-performance graphics database for the product environment, such as Neo4j, Infinite, Graph, DEX, InfoGrid, HyperGraphDB, Trinity and so on[1]. Among them, Neo4j is the mainstream of a Java based open source software currently, its kernel is a very fast graphics engine, with the recovery, the two phase of the submission, support for XA transactions and other database product features.

Neo4j is a network-oriented database—that is, an embedded, disk-based, fully transactional Java persistence engine that stores data structured in networks rather than in tables. What makes Neo4j interesting is the use of the so called "network oriented database". In this model, domain data is expressed in a "node space" - a network of nodes, relationships and properties (key value pairs), compared to the relational model's tables, rows & columns. Relationships are first class objects and may also be annotated with properties, revealing the context in which nodes interact[2]. The network model is well suited to problem domains that are naturally hierarchically organized.

III. Create NEO4J DATABASE

The basic data model in Neo4j consists of nodes, relations and attributes. Nodes are similar to object instances, different nodes are connected by various relationships, a writer is a node, "name"and "age" are attributes of writer. Association is similar to the edge in a directed graph, the edge consists of three elements: the start node, the end node and the type. The orientation of the edge further clarifies the semantic relationship between nodes. The attributes of nodes and relationships can be defined by key-value. Actors, directors, writers and films are different entities when the film data is stored, the Neo4j database not only needs to store entities, but also needs to store the relationships among entities.

Creating a Neo4j database is pretty easy, just add nodes, relationships and their attributes into database. Example 1 shows how to create a Neo4j database storing film data.

Example 1:

```
Create
(TomHanks:Stars{name:'Tom Hanks',born:1956}),
(ForrestGump:Movie{title:'Forrest Gump', released:1994}),
(TomHanks)-[:ACTED_IN]->(ForrestGump)
```

The research work is supported by the Excellent Young Teachers Training Project (the second level, Project number: YXJS201508), Engineering Project of Communication University of China (Project number: 3132016XNG1608).

978-1-5090-5507-4/17/$31.00 ©2017 IEEE
ICIS 2017, May 24-26, 2017, Wuhan, China

In Neo4j, labels can be used to identify a collection of nodes, it is allowed to set indexes , define constraints, and query on the node collection[3]. In this example, "Create(TomHanks:Stars{name:"Tom Hanks",born:1956})" create a star named Tom Hanks, born in 1956, the "Stars" is the label represents that the node is a star, the same as label "Movie" and so on ; "Create (TomHanks)-[:ACTED_IN]->(ForrestGump)" represents a relationship named "ACTED_IN" between the star Tom Hanks and the movie <Forrest Gump>, means that Tome Hanks played the movie <Forrest Gump>.

This paper takes the movie <The Green Mill> as the starting point, create a small database with 19 nodes and 19 relationships as shown in Figure 1. In the figure, the green node means it is a film node, yellow means that the label for the actor's node, pink label means that it is a writer, blue label means that it is a director, different colors used to represent different labels.

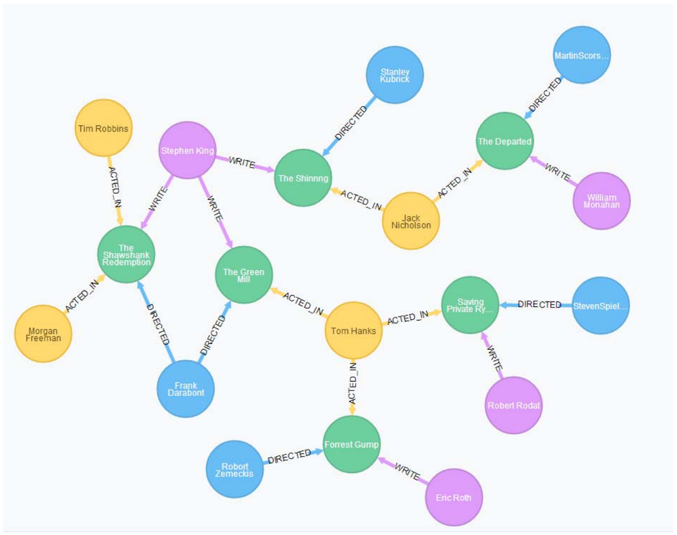


Figure 1. Film data graph

IV. FILM DATA ANALYSIS

After creating database of film data, it is time to start the a film data analysis, and the analysis will focus on the following aspects: Firstly, the attributes of single node will be analyzed; The second point is the relationship between two or more nodes; Thirdly, how to find the hidden relationships among nodes will be shown.

To analyze film data, the main method is to traverse and query the database. Neo4j focuses on solving the problem of performance degradation in traditional RDBMS queries with a large number of links. The database will traverse the nodes and edges at the same speed, there is no relation between the ergodic speed and the data quantity of the graph[4].

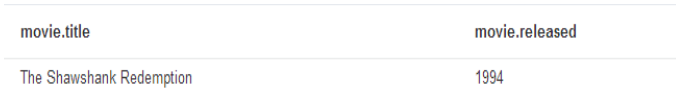
Cypher was originally created by Neo Technology for its graph database Neo4j. Neo4j supports very complex graph traversal operations, Cypher query language makes it quite easy to query data from database. Cypher contains a variety of clauses. Among the most common are: MATCH and WHERE. These functions are slightly different than in SQL. MATCH is used for describing the structure of the pattern searched for, primarily based on relationships. WHERE is used to add additional constraints to patterns. Cypher additionally contains clauses for writing, updating, and deleting data. CREATE and DELETE are used to create and delete nodes and relationships[5]. Next the film data analysis will be executed through database query.

A. *Query of attributes of single node*

When we only know the name of a film, and want to know other attributes of the film, the query of node's attributes is necessary. Here is an example using Cypher to query attributes of a node:

```
match movie
where movie.title="The Shawshank Redemption"
return movie.title,movie.released
```
```
return 1 line, times 32 ms
```

As shown in Figure 2, it is simple to query attributes of a node after knowing one of its attributes.



movie.title	movie.released
The Shawshank Redemption	1994

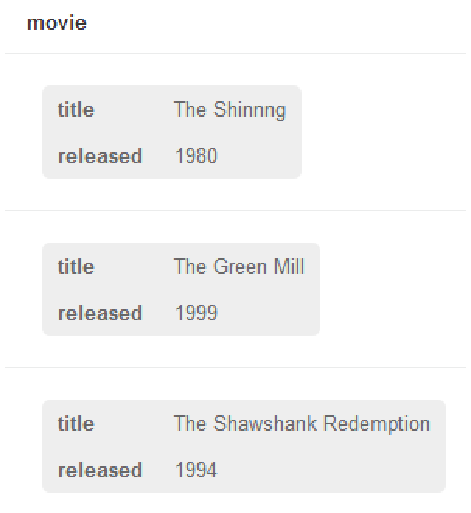
Figure 2. Query result

B. *Query the relationship between two or more nodes*

Graph database is good at dealing with a large number of complex, interconnected data, when we want to know the other node that has relationship with the node we know, it is nature to query the relationship between the two nodes. For example, find the list of films written by "Stephen King", this query is very useful, because in our daily life, we may look for movies writen by the writer whose style is attractive and unique.

```
match (writer:Writer)
where writer.name="Stephen King"
match writer-[:WRITE]->movie
return movie
```
```
return 3 lines, times 3 ms
```

After executing this query, the results are shown in Figure 3. By querying, it is easily to find movies writen by "Stephen King".



movie

title	The Shinnng
released	1980

title	The Green Mill
released	1999

title	The Shawshank Redemption
released	1994

Figure 3. Query results

C. *Find the hidden relationships among nodes*

Due to the increasing number of film data, users often fell confused and difficult to choose what film to watch. One efficient solution is to recommend films that users may be interested in[6]. which can actively perceive the needs of various audience. In order to recommend film to users more accurately, we can analyze relationships between movies that the user has seen.

Here is an example, according to the user has seen the film <The Shawshank Redemption> and <Forrest Gump>, recommend other films that related to the two films to the user. By analysis we find that Tom Hanks acted in <Forrest Gump> and <The Green Mill>, the director Frank Darabont not only directed <THE Green Mill> but also directed <The Shawshank Redemption>, so we find the potential relationship between the movie <Forrest Gump> , <The Shawshank Redemption> and <The Green Mill>, so we can recommend the movie <The Green Mill> to the user. The following is the Cypher query on this question.

```
match movie1
where movie1.title="The Shawshank Redemption"
match movie2
where movie2.title="Forrest Gump"
match
searchPath=movie1<-[*]-(people1)-[*]->movie-[*]-(people
2)-[*]->movie2
return  distinct movie.title,searchPath
```

```
return 1 line, times 20 ms
```

After executing this query, the results are shown in Figure 4. As we can see, the Cypher query is concise and clear. As comparison, when we use relational database to execute this query: Firstly, we need to query the creaters of the two movies; Secondly, we need to query other all movies created by the creaters; Thirdly, we also need to find the movie co-created by creaters of the two movies. Not only the query is complex, we also need to create many tables in relational database. Neo4j database query can quickly search through all kinds of relations between nodes, find the hidden relationship between the producers of the two films.

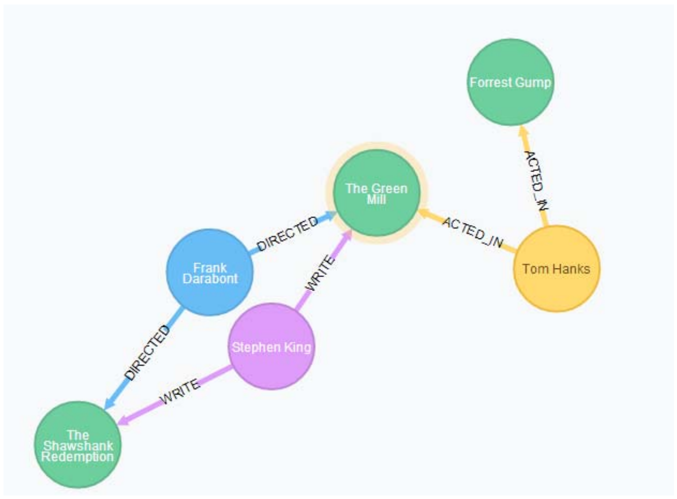


Figure 4. Query result

V. CONCLUSION

Neo4j is suitable for database creation, query, update, etc, particularly suitable for handling a large number of complex, dynamic, interactive, low structured data. It is an effective method to solve the problem of mass data storage in the fields of social network and information visualization. Nowadays film data is growing exponentially, the relationship between data become complicated more and more. Most of time we need to focus not only on the data itself, the hidden relationship between perceived data is also needed to focus on. Traditional relational database can not meet this demand, Neo4j database is good at dealing with complex and multi-connection data, using Neo4j database to store and manage film data makes it convenient for film website to meet user's requirements and interest.

REFERENCES

[1] S. Sencer and K. Eren. "Graph Database for Agent Based Emergency Response Model," Proceedings of the 2014 international conference on advances in big data analytics: July 21-24, 2014.

[2] K. Nagi. "A New Representation of WordNet using Graph Databases" The fifth international conference on advances in databases, knowledge, and data applications. 2013:1-8

[3] J. Webber. "A Programmatic Introduction to Neo4j," 2012 ACM conference on systems, programming, and applications: software for humanity, October 19-26, 2012, Tucson, Arizona.

[4] Y. L. Wang. "A comparative study of graphic database NEO4J and relational database," Modern Electronic Technology, vol. 35, pp. 77-7, 2012.

[5] L. Wang. "Construction of Document Resources Association Network Based on graph database technology," Digital Library Forum, pp. 59-65, 2014.

[6] Y. L. Wang. "Research on embedded application of graphic database," Neo4j modern electronic technology, vol. 35, pp. 22, 2012.