Web Data Management Description

Group 12
Yuchen Huang 4511980, Lina He 4519051
Yujing Gong 4471458, Yuzhu Yan 4468023

I. Introduction

In this project, a movie information web service which connects multiple backends is developed. Three backends are implemented in this programming assignment, where PostgreSQL is initially used as the relational backend. Besides, the document-based database MongoDB and the graph-based database Neo4j are adopted as additional data storage. The web service provides five service interfaces in total. In addition to the different query languages used for the different databases, the main programming language used in this project is Python.

II. PostgreSQL

PostgreSQL is an object-relational database management system. It is ACID-compliant and transactional and it can handle complex queries using many indexing method that are not available in other databases. Figure 1 shows the database construction in PostgreSQL.

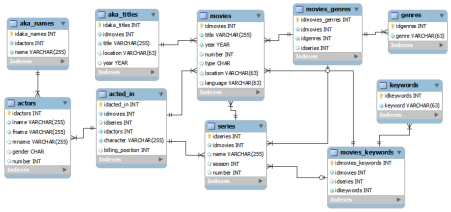

Fig. 1. Data model of PostgreSQL

III. MongoDB

MongoDB is a cross-platform document-oriented database. Unlike the traditional table-based relational database structure, MongoDB supports JSON-like documents with dynamic schemas which is called BSON. Hence, the data needs to be exported from PostgreSQL and transformed into documents in the format of BSON. The data is collected by using SQL in PostgreSQL and exported into two JSON files, one contains the information of the movies and the other contains the information of the actors. The two JSON files can be imported into MongoDB easily by using the mongoimport command. The query in MongoDB is quite easy, the built-in find() function can be used.

IV. Neo4j

Neo4j is a graph database. It is also an ACID-compliant transnational database using native graph storage and processing, which shows very good performance. The data, written in csv format, can be generated from the Postgres database, which is an easy but time-consuming process. Two kinds of csv files are needed, namely the node csv files and the relationship csv files. According to the analysis of the Postgres database, the nodes and relationships figure as shown in Figure 2 is generated, which indicates 7 node csv files and 6 relationships csv files. In addition, When using the neo4j-import command, importing node and relationship csv files into the Neo4j database could be quite easy. One more thing that needs to be mentioned is that speed-up of searching for data can be acquired by using indexes.

The query language used in Neo4j is called Cypher, which is a declarative and SQL-inspired language for describing patterns in graph database. It is quite easy to learn and use.

Above all, Neo4j shows its friendly and easy-to-use characteristics as a database.

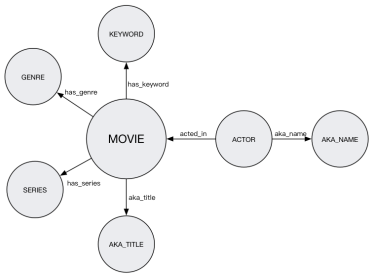

Fig. 2. Data model of Neo4j

V. Construction

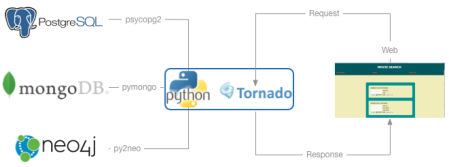

Fig. 3. Connection

Figure 3 shows the construction of the web service. The python web framework and asynchronous networking library are provided by Tornado. The web service works in the following work flow:

- Listen to the request from web interface.
- Request processing by Python and Tornado.
- Execute SQL(PostgresSQL)/ find() function (MongoDB)/ Cypher (Neo4j) at corresponding backend.

- Return response and print it on web interface

VI. CONCLUSION

Comparison of the three databases is shown in the following table:

TABLE I
COMPARISON

	Postgres	MongoDB	Neo4j
Advantages	Immunity to over-deployment Better support than the proprietary vendors Significant saving on staffing costs Legendary reliability and stability Extensible Cross platform Designed for high volume environments	Document-oriented storage Full Index Support Replication& High Availability Auto-sharding Querying Fast In-Place Updates Map/Reduce GridFS	Perfect for running ad-hoc graph queries Allow deep traversals faster than Relational Fast of k-hop traversals Represent multiple dimensions Exportable tabular results of any query result Graph visualization of query results containing nodes and relationships Convenient exploration of Neo4j's REST API
Disadvantages	Poor at handling data relationship Less performance for simple read-heavy operations Lacks behind in terms of popularity	No Joins No ACID Transactions Memory Usage Concurrency Issues	API necessary Still new, lack native implementations for different platforms
Implementation	Data integrity Complex custom procedures Integration Complex Design	Cloud Commodity	Handle highly connected data Like Google, Facebook, Linkedin and PayPal

By implementing PostgreSQL, MongoDB and Neo4j for the movie research web service separately. It shows that Neo4j gives the best performance with respect to the first three service interfaces, which take around 0.1 or 0.2 seconds. It is interesting that using indexes in Neo4j achieves speedup significantly (We did this improvement after the presentation. Details can be seen in the changelog file in the Neo4j directory). Besides, PostgreSQL also shows a similarly good performance. And MongoDB shows worse performance compared to the other tow, which may be caused by the use of two data files instead of only one.