Web Data Management Description

Group 12
Yuchen Huang 4511980, Lina He
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.

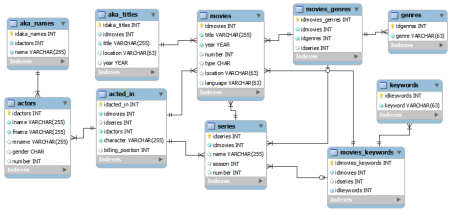

Fig. 1. McProf:Callgraph

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 named as BSON. Hence, data need to be exported from PostgresSQL and transformed into documents in the format of BSON. We collect the data by using the SQL from PostgreSQL and output them into two json files, one contains all the information for movies and the other contains the information for the actors. The two json files can be imported to MongoDB easily by using mongoimport command. The query in MongoDB is quite easy, simply use the built-in find() function.

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 Fig 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 need to be mentioned is that high speedup can be observed by using Index.

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.

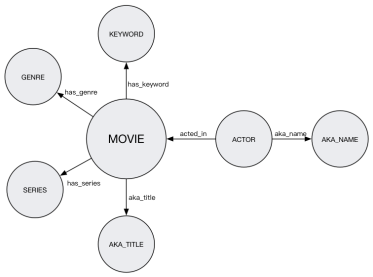

Fig. 2. McProf:Callgraph

V. Construction

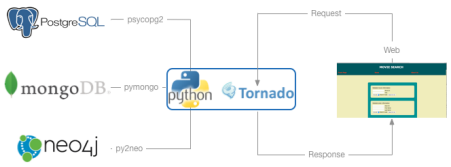

Fig. 3. Connection

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

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

- Return response and print it on web interface

VI. CONCLUSION

Comparison of three databases can be seen 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 movie research web service separately, we observe that Neo4j shows the best performance, for first three service interface, it takes around 0.1s or 0.2s. It is interesting that Index achieve speedup significantly (We actually implement Index after presentation, detail modification can be seen in the changelog in Neo4j directory). PostgreSQL also shows a similarly good performance. MongoDB shows less good performance may caused by two JSON-like files which need more time for searching.