Results

For the NoSQL database, I choose Neo4j to design my database and finish the graph to show the relationships between each elements. The database is about the movie of China, and the elements are **genre, movie and actor. Besides, the relationship between the movie and the actor and between movie and genre will also be shown from the graph**. So there will be 5 csv files for the database.

I import these files to the Neo4j and code them to the program. Here is the code:

//import genre

LOAD CSV WITH HEADERS FROM "file:///genre.csv" AS line

MERGE (p:Genre{gid:toInteger(line.gid),name:line.gname})

//import actor

LOAD CSV WITH HEADERS FROM 'file:///person.csv' AS line

MERGE (p:Person { pid:toInteger(line.pid),birth:line.birth,

death:line.death,name:line.name,

biography:line.biography,

birthplace:line.birthplace})

//import movie

LOAD CSV WITH HEADERS FROM "file:///movie.csv" AS line

MERGE (p:Movie{mid:toInteger(line.mid),title:line.title,introduction:line.introduction,

rating:toFloat(line.rating),releasedate:line.releasedate})

// import the relationship between the actor and the movie

LOAD CSV WITH HEADERS FROM "file:///person\_to\_movie.csv" AS line

match (from:Person{pid:toInteger(line.pid)}),(to:Movie{mid:toInteger(line.mid)})

merge (from)-[r:actedin{pid:toInteger(line.pid),mid:toInteger(line.mid)}]->(to)

//import the relationship between the genre and the movie

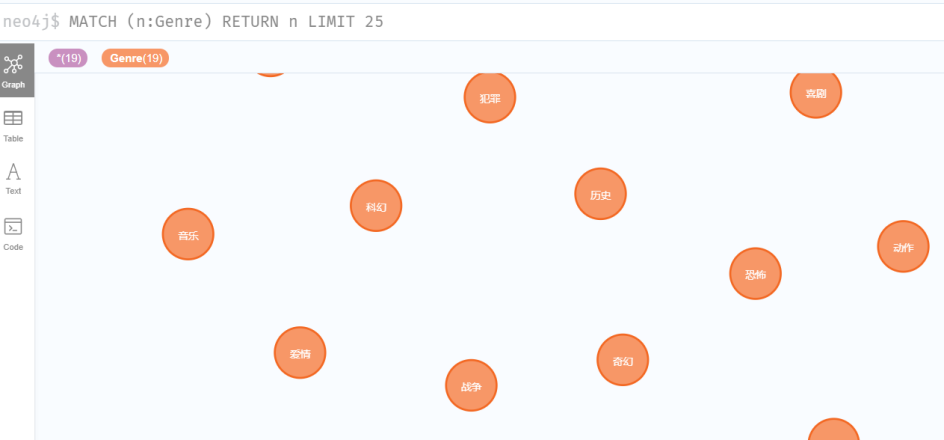
LOAD CSV WITH HEADERS FROM "file:///movie\_to\_genre.csv" AS line

match (from:Movie{mid:toInteger(line.mid)}),(to:Genre{gid:toInteger(line.gid)})

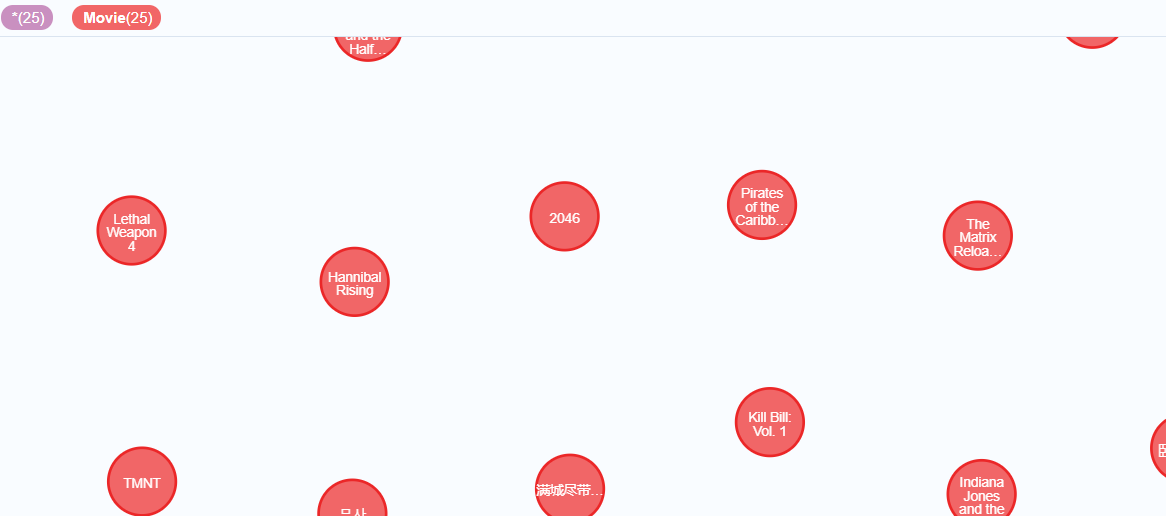
merge (from)-[r:is{mid:toInteger(line.mid),gid:toInteger(line.gid)}]->(to)

The results:

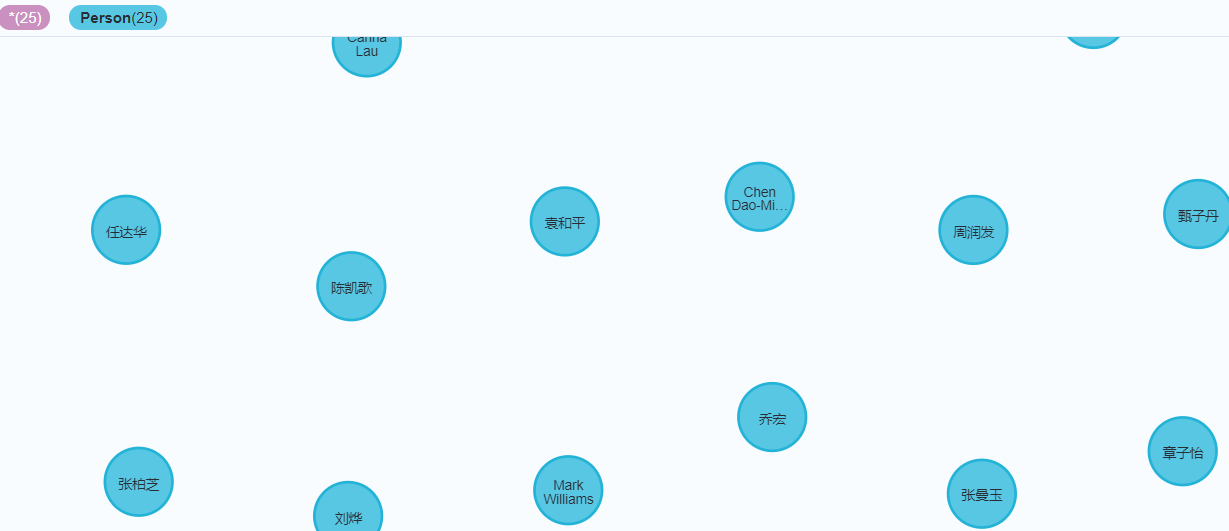
Genre:



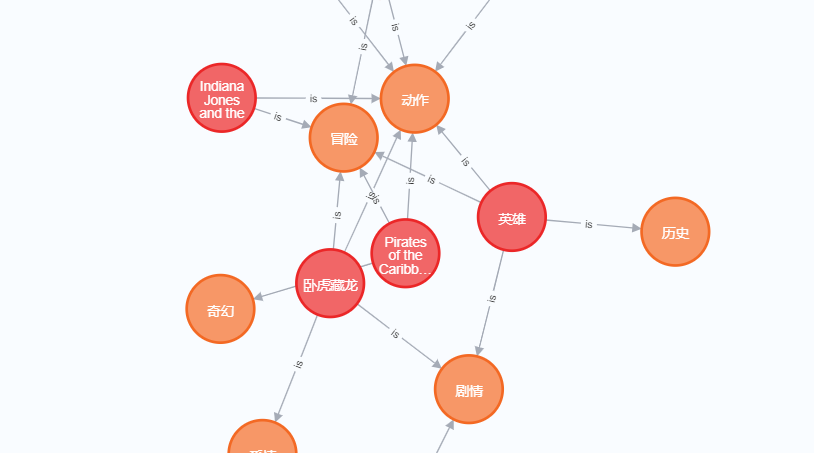
Movie:



Actors:



Relationship(movie and genre):



Relationship(actor and movie):

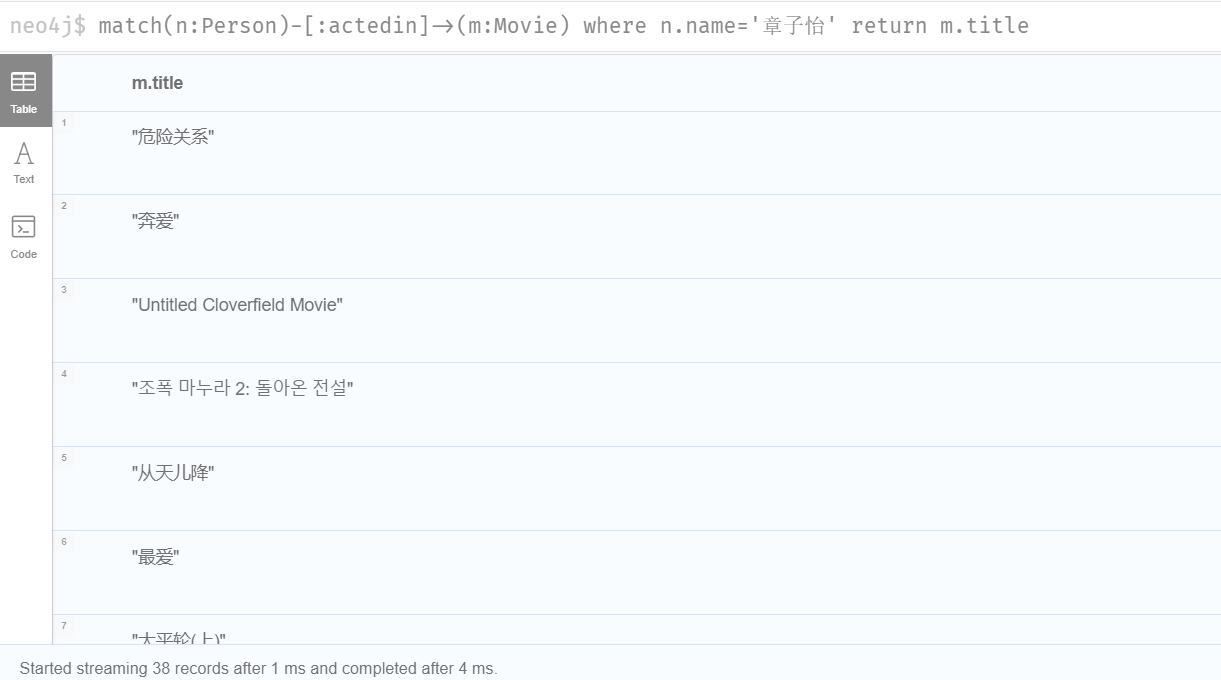


In this database, for example, if we want to access the information of the movies of 章子怡(zhang ziyi). We can use the code like this:

Code:

match(n:Person)-[:actedin]->(m:Movie) where n.name='章子怡' return m.title

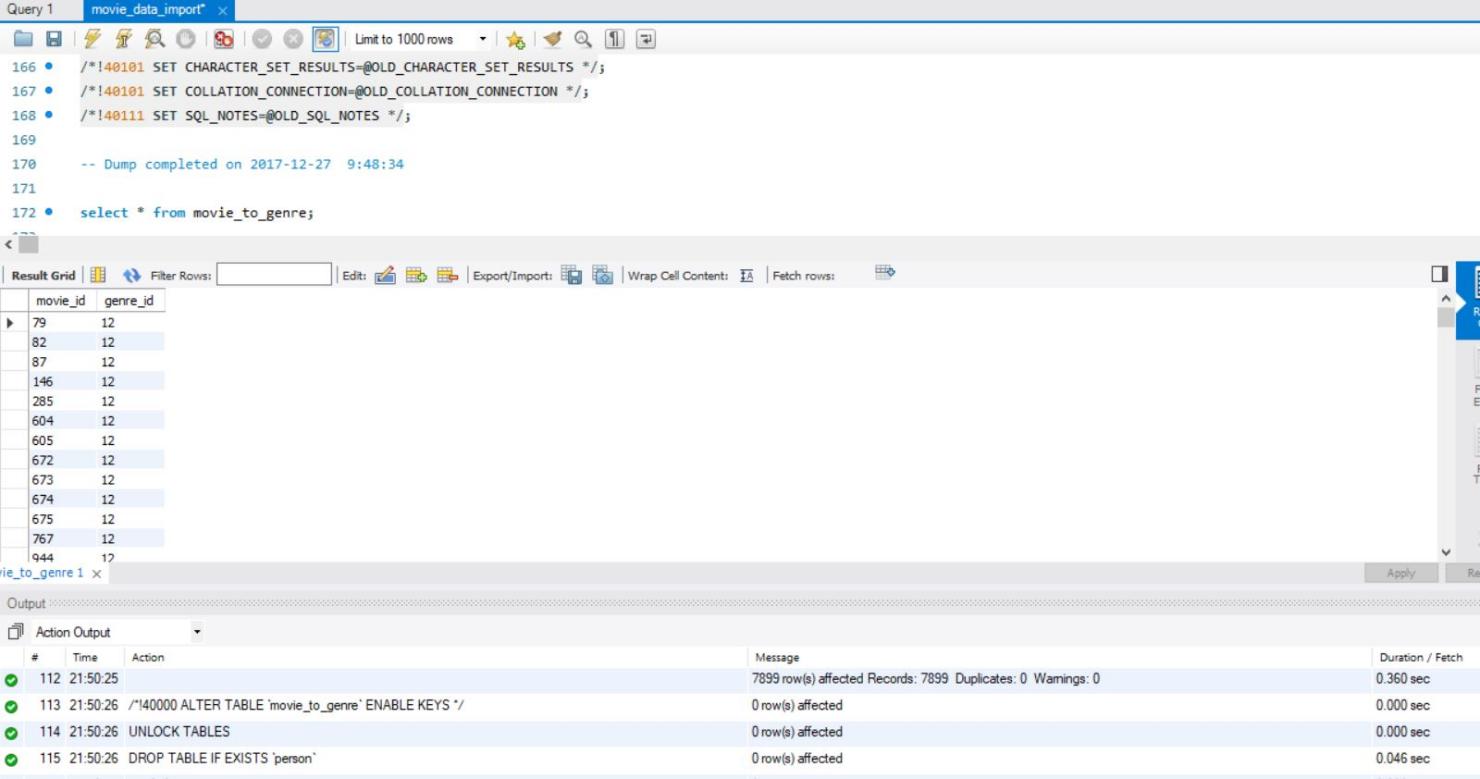
Result:



As we can see from the figure, the movies can be shown in 4ms from 38 records. The efficiency is high.

For the RDB, I import the database to MySQL.

By using the code: select \* from movie\_to\_genre, the search results can be seen.



To compare the NoSQL and SQL, we can see from the Neo4j and MySQL, the performance of NoSQL is better than the SQL. Besides, from the figure of Neo4j, the relationship between the elements can be relatively easily seen, and it is more flexible for NoSQL comparing to SQL. However, the advantage of SQL is that because it is relation oriented, the consistency of SQL is much better, while for NoSQL, it can not fully support ACID transactions, which could cause inconsistency for the system.