Database Study Points Assignment #2

Graph Databases

Frederik Bilgrav Andersen

Page Rank algorithm

The PageRank algorithm is an algorithm that measures the importance of each node based on its relationships. There's multiple scenarios where such importance can be measured, mostly it is between weighted and unweighted relations. If the relations are unweighted the importance of each node is based on the amount of relations to the node, meaning that the node with the highest relations attached is the most important. If the relations are weighted, for example with a numeric value, then the importance is calculated based on the total sum of the weight from each relation to the node.

The formula of page rank algorithm is as follows:

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)})$$

In our database on video games between 1980-2023 we used the Page Rank algorithm to check the most popular genres in the timeframe. Here we simply looked at the Game and Genre nodes and the relationship between them which was unweighted. Here we simply found the genre nodes with the highest amount of relations to game nodes:

```
CALL gds.pageRank.stream('PageRankProjection')
YIELD nodeId, score
WITH gds.util.asNode(nodeId) as node, score
WHERE node:Genre
RETURN node.name as genre, score
ORDER BY score DESC
LIMIT 10;
```

A

Advantages:

1. Performance: When comparing graph databases to relational databases, graph databases usually gain a significant performance gain when reading data that require deep traversal / multiple joins.
2. Visualization: Graph databases have, in my opinion, a more clear way to visualize data and the corresponding relations between them.

Disadvantages:

1. Complexity: Compared to my own knowledge of relational databases, graph databases seem more complex and difficult to implement.

Best & worst scenarios

The best scenario of graph databases is when dealing with highly relational data while the worst scenarios are data sets with nodes / entities with a limited amount of relations.

B

The Cypher queries I've implemented are mainly the one to create the database itself from the CSV file (1) as well as the query to get the most popular game genres (2).

As for the first query, to create the database in a relational database using SQL we would first have to define a schema for all the data before we insert the data. In Cypher and Neo4J we simply loaded the file and built the nodes and the database would be created from that query.

As the second query to get the most popular genre there would probably be foreign key on the Game table referencing the Genre table. Here you would be able to perform a query to see the amount of games per genre by getting the count of each game with corresponding foreign keys.

C

The data is stored locally on discs, this data is organized into nodes and relationships that correspond to a graph structure. Neo4J uses a query language called Cypher to execute queries.

D

1. Causal Clustering - create multiple clustered databases, core servers and read replicas using the leader / follower principle.
2. Database sharding.
3. Database partitioning.