Janus Rasmussen – cph-jr270@cphbusiness.dk

Degree centrality:

The Degree Centrality algorithm can be used to find popular nodes within a graph.

Degree centrality measures the number of incoming or outgoing (or both) relationships from a node, depending on the orientation of a relationship projection.

Degree centrality tells you which node has the most connections. In the example below it checks which game from the database has the most connections with other nodes.

```
1 //Calculate which node has the highest degree
2 MATCH (g:Game)
3 OPTIONAL MATCH (g)-[]→()
4 WITH g, count(*) as degree
5 RETURN g.title AS gameName, degree
6 ORDER BY degree DESC
```

matches all nodes with a "Game" label and then counts the number of relationships connected to each node. The nodes and their corresponding degrees are then sorted in descending order based on the degree, and the result is returned as a table with the game name and its degree.

	gameName	degree
1	"Fire Emblem: Thracia 776"	100
2	"Uncharted: The Lost Legacy"	98
3	"Shantae and the Seven Sirens"	92
4	"Fire Emblem: Genealogy of the Holy War"	68
5	"Kingdoms of Amalur: Reckoning"	64
6	"Marvel Snap"	62

Advantages of using graph databases:

- It possesses the ability to manage complex relationships between data points.
- It helps connected data processing and queries to run more quickly.

Disadvantages of using graph databases:

- graph databases may have limitations when it comes to handling large volumes of data or supporting high write volumes. graph databases often require more processing power to traverse relationships between nodes, which can slow down query performance as the size of the graph grows.

Best scenarios for using graph databases.

- It can help verifying identities and detecting fraud.

How does the DBMS you work with organizes the data storage and the execution of the queries?

- Neo4j organizes data in a graph structure and stores it in files on disk. Its query engine parses and optimizes queries, taking advantage of graph structure and indexes for efficient retrieval.

Which methods for scaling and clustering of databases you are familiar with so far?

Replication

- creating multiple copies of a database and distributing them across multiple servers to improve performance.

Load balancing

- distributing incoming requests across multiple servers to improve performance and avoid overloading any single server.

Sharding

- partitioning a database into smaller, more manageable pieces called shards, and distributing those shards across multiple servers.