Provide in writing, each student individually, answers to the following questions:

a. What are the advantages and disadvantages of using graph databases and which are the best and worse scenarios for it?

b. How would you code in SQL the Cypher statements you developed for your graph-algorithms-based query, if the same data was stored in a relational ### database?

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

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

Markus

A. What are the advantages and disadvantages of using graph databases and which are the best and worse scenarios for it?
Graph databases do have several advantages over a more traditional relational databases, which could include: Flexibility, because graph databases can store and manage complex data structures more efficiently than relational databases. Performance, in this case graph databases querying relationships between data points, making them ideal for use cases such as social networks and might be good for fraud detection aswell. Scalability, graph databases is able to scale horizontally easier than relational databases, making them good for applications that require high levels of performance and scalability.
However graph databases also have some disadvantages to consider: Which could be complexity, graph databases require a different approach to modelling data than relational databases, which can make them a bit more difficult to work with for us developers and maybe even data analysts who are not familiar with graph concepts. Limited querying aswell because graph databases excel at querying relationships between data points, they may be less suitable for other types of queries that are more easily handled by relational databases.

B. How would you code in SQL the Cypher statements you developed for your graph-algorithms-based query, if the same data was stored in a relational database?
Both SQL and Cypher are two different languages used for querying data, so it is not possible to directly translate Cypher statements into SQL. But if the same data was stored in a relational database, you would be able to use SQL to perform queries that could be able to achieve similar results as the Cypher statements. For example, if you had a graph database with nodes representing users and the edges representing

relationships between them, a Cypher statement like this: MATCH (user)-[:FRIEND]->(friend) WHERE user.name = ‘Markus’ RETURN friend.name Could be translated into SQL like this: SELECT friend.name FROM users AS user JOIN friendships AS f ON user.id = f.user_id JOIN users AS friend ON f.friend_id = friend.id WHERE user.name = 'Markus ; This query joins the users table with the friendships table to find all of Alice's friends.

C. How does the DBMS you work with organizes the data storage and the execution of the queries?
The specific DBMS I work with organizes data storage and query execution in a particular way, but in general, most DBMSs organize data storage using a combination of tables, indexes, and partitions. Tables store the actual data, indexes provide a way to quickly locate specific rows in a table, and partitions help to distribute data across multiple physical devices for improved performance and scalability. When executing queries, the DBMS uses a query optimizer to determine the most efficient way to access and retrieve data based on the query's structure and the available indexes.

D. Which methods for scaling and clustering of databases you are familiar with so far?
Some common methods for scaling and clustering databases include: Vertical scaling, to be adding more resources to a single database server to improve performance. Such as CPU. Horizontal scaling: Adding more database servers to deal with the workload and to improve performance and also the scalability. Replication: Creating multiple copies of the database on different servers to improve and secure that we have availability and fault tolerance.