Graph databaser har sine styrker i applikationer som har et stort fokus på forholdene mellem datapunkter, derudover kan Graph database være meget fleksible da de ikke er bygget op omkring et schema eller regelsæt som definere det, dette medføre at din database kan håndtere en masse forskellige slags datatyper, hvilket gør det nemmere at videre udvikle den applikation som databasen ligger grundlag for.

Graph databaser har desuden også høj performance når man laver queries dette skylde at selve forholdene mellem noderne ligge som data i databasen, og det er derfor ikke behøvet at lave en beregning hver gang skal finde et data entry gennem en relation som man skal i f.eks. relationelle databaser.

Nogle af de ulemper som følger med brugen af Graph databaser. Et af disse punkter kan være at hvis man har valgt at bruge en Graph database i en transaktions baseret system, skal man vælge med omhug da det ikke er alle udbydere som tilbyder denne service. Derudover er der ikke en standardiseret måde at query graph database, hvilket kommer af at Graph database stadig er relativt nyt og af samme årsag kan det være svært for udviklere som benytter sig af Graph database at finder løsninger eller ekspertise i brugen af dem.

```
// pagerank eksempel I sql. //
MATCH (c:Character)
WITH {id: id(c), name: c.name} AS nodeMap
WITH collect(nodeMap) AS nodes
CALL gds.graph.create('GoT', 'Character', 'INTERACTS')
YIELD graphName
CALL gds.pageRank.stream(graphName, { maxIterations: 20, dampingFactor: 0.85 })
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS character, score
ORDER BY score DESC
LIMIT 5
```

I forhold til opbevaring af data, der gemmer neo4j hver node, relation og egenskab, som en separat entitet i stedet for i tabeller som f.eks. relationelle databaser gør, dette giver mulighed for meget
hurtigere at eventuelt finde information fra databasen.

Når det kommer til queries, bruger neo4j en query language som hedder Cypher, det minder på mange punkter om sql da det har draget inspiration derfra, men i stedet for at man søger efter data i tabeller, gør man ofte det at man matcher en nodes properties, f.eks. er persons navn, med en type relation
Et ekemsple kunne være MATCH(:Person{name:”Dan”})-[:LOVES]->(:Person{name:”Ann”})
I dette eksemple prøver vi at finde et match hvor en node som har en property name = Dan har en relation af type LOVES med en node med en property name = Ann.

Vi har primært haft undervisning i scalerbarhede af relationelle databaser, hvor vi har fokuseret på forskellige typer clustering som Windows Clustering technology og Availability groups. Og så har vi haft om forskellige typer af replication som Transactional replication, Snapshot replication og Merge Replication