Grafdatabaser och verktyget Neo4j: en utvärdering

Mattias Levlin
Kandidatavhandling i datavetenskap
Handledare: Annamari Soini
Fakulteten för natuvetenskaper och teknik
Åbo Akademi
2017

Referat

Databaser är ett koncept som hör till de mest centrala inom datavetenskapen. Den vanligaste typen av databaser är relationsdatabaser, som är lämpliga att använda i de flesta typiska situationer där en databas behövs. I vissa fall kan dock grafdatabaser vara mer ändamålsenliga. En grafdatabas är ett lämpligt verktyg då man vill lagra och representera data i grafformat, istället för i vanligt relationsdatabasformat, och inom grafdatabaser ligger fokuset på relationerna noderna emellan, istället för dataattributen, som är vanligt i typiska relationsdatabastabeller. Denna avhandling behandlar grafdatabasens skillnader från den mer vanliga relationsdatabasen, grafdatabasens typiska struktur och dess användningsområden. En av de mest använda grafdatabaserna, Neo4j gås igenom och ett praktiskt verktyg utvecklat med hjälp av Neo4j beskrivs; histoGraph, som används för utforskande visualisering och utvärdering av historiska relationer i sociala nätverk.

Nyckelord: grafdatabas, databas, relationer, visualisering, Neo4j, histoGraph

Innehållsförteckning

1 Inledning: Grafdatabasens utveckling och dess kännetecken

En viktig inledande distinktion är den mellan en *databas* och en *databasmodell*. En databas är en specifik applikation, som kan användas genom att till exempel installeras på en serverdator. En *databasmodell* är det teoretiska schemat som databasen är designad enligt. Till exempel är databasen MySQL designad enligt relationsmodellen och databasen Neo4j är designad enligt grafdatabasmodellen. En annan distinktion gällande diskussioner om databaser är den mellan *schema* och *data*. *Schema* refererar till de logiska specifikationerna som bestämmer hur databasens struktur ser ut. *Data* refererar till själva instansmaterialet som finns inuti databasstrukturen.

Relationsdatabaser hör till de mest använda databaserna idag och är tillräckliga verktyg att använda som bas i de flesta generella databasbaserade applikationer. De fyra mest populära databaserna 2017 (Oracle, MySQL, Microsoft SQL Server och PostgreSQL) är alla baserade på relationsmodellen [1] [2]. Enligt Angles har dock vissa begränsningar hos den traditionella relationsdatabasmodellen lett till utvecklingen av nya databasteknologier som gemensamt kallas NoSQL-databaser [3]. Dessa nya databaser kan kategoriseras in i fyra kategorier: bredkolumn-lager, som följer Google:s BigTable-modell (till exempel databasen Cassandra); dokumentlager, som är avsedda för semistrukturerade data (till exempel MongoDB); nyckelvärdelager, som implementerar en nyckelvärdeskarta för indexering och hämtning (till exempel BerkeleyDB) samt grafdatabaser, som är avsedda för lagring av grafdata (till exempel Neo4j). I denna avhandling behandlas den sista av de ovannämnda NoSQL-kategorierna och kategorins främsta exempel: grafdatabaserna och Neo4j. Denna kategori av databaser kommer att jämföras med de traditionella relationsdatabaserna.

1.1 Grafdatabasens historia och utveckling

Grafdatabasmodellens rötter går att spåra till den så kallade nätverksmodellen, en databasmodell som var populär under 1960-talet men ej längre används i någon betydande utsträckning. Nätverksmodellens arkitektur specificerades av organisationen CODASYL (*Conference on Data Systems Languages*) 1971 och hade flera egenskaper som också den nuvarande grafdatabasmodellen har: den representerade data som en graf med noder och bågar, hade sammankopplade komponenter som huvudegenskap och saknade en tydlig hierarki [4].

Under 1980-talet uppkom behovet av att kunna modellera komplexa nätverk på en mer avancerad abstraktionsnivå än den som nätverksmodellen erbjöd. Därtill observerade forskare svagheter hos nätverksmodellen: det teoretiska schemat och implementationen flöt ofta ihop och det var svårt att utföra graftraverseringsoperationer. Andra existerande databasmodeller, såsom relations- och hierariska modeller, var inte heller ändamålsenliga för nätverksmodellering; de kritiserades för sina rigida krav i tillåtna datastrukturer, begränsade förfrågningsspråk och svårigheten i att överblicka data. Utgående från detta tog forskare fram idén om att utveckla en ny, mera raffinerad databasmodell - grafdatabasmodellen - delvis baserad på nätverksmodellen [5].

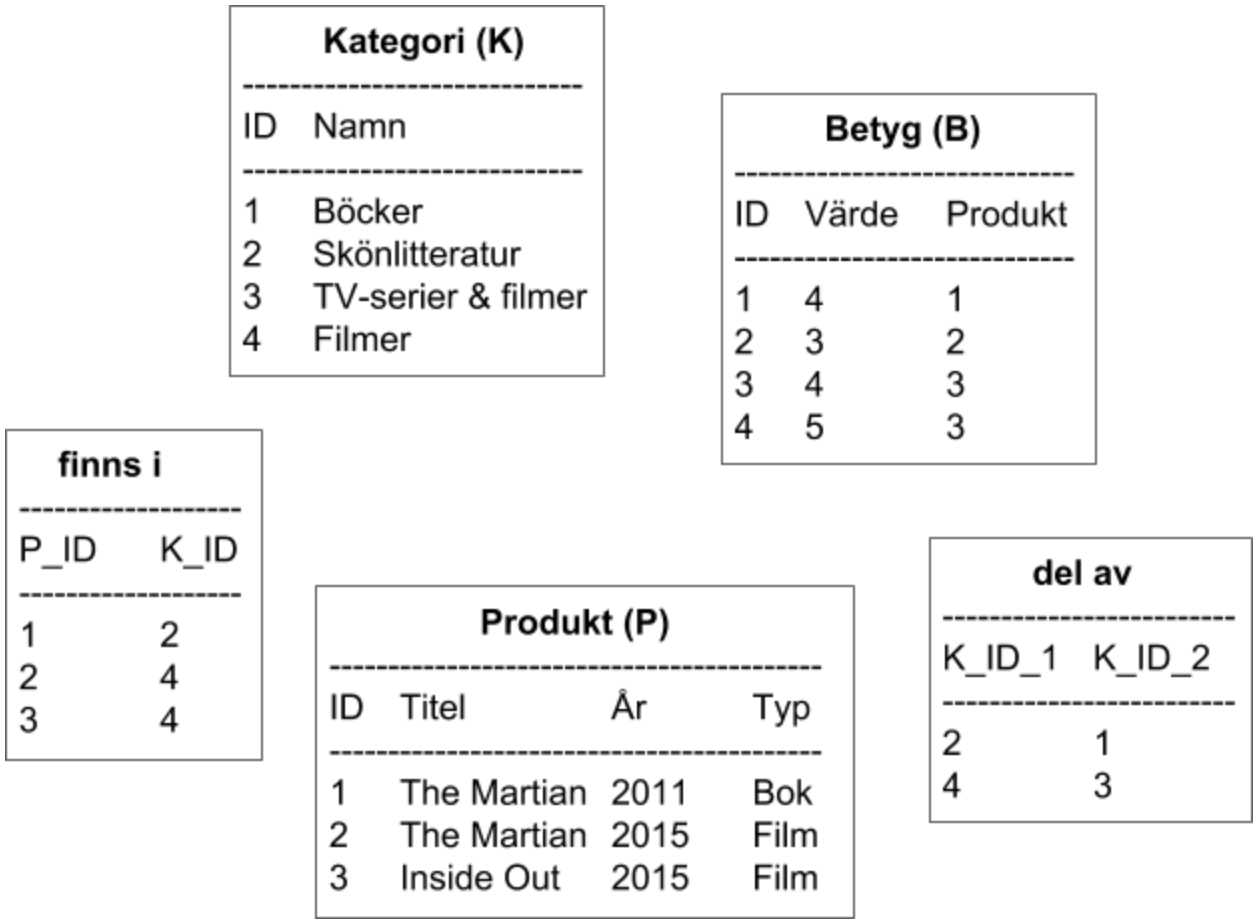
Den första tidsperioden då ämnet grafdatabaser var ett populärt ämne för forskare inom datavetenskap var under det tidiga 1990-talet. Flera olika grafdatabasmodeller var då under utveckling, men på grund av olika orsaker sjönk intresset kraftigt under senare delen av 1990-talet, till den grad att “ämnet nästan försvann” (egen översättning) enligt Angles och Gutierrez. Detta berodde till stor del på Internets genomslag: en första konsekvens av dess genomslag var att forskarna inom databasområdet blev mera intresserade av semistrukurerad data, som under 1990-talet inte ännu hade några kopplingar till grafdatabaser. Den andra konsekvensen var

uppkomsten av XML (*Extensible Markup Language*), vars funktioner i många fall ersatte grafdatabasernas funktionalitet och därtill kunde hantera semistrukturerad data. En tredje konsekvens var att databasforskarna började intressera sig mera för specifika databastillämpningsområden som websidor, dokument och spatiell data, istället för grafdatabaser [5].

Under 2000-talet har intresset för grafdatabaser ökat igen. Nya, så kallade multimodellsystem har uppkommit. Dessa hybridsystem implementerar utöver grafdatabasmodellen även andra databasmodeller, såsom dokumentlager och nyckelvärdeslager. Framstående exempel är OrientDB och ArangoDB. Klassiska grafdatabaser, som baserar sig på endast grafdatabasmodellen, är dock fortfarande populära. Av de tre mest använda klassiska grafdatabaserna idag, uppkom alla under under 2000-talet eller senare; Neo4j släpptes 2007, Titan 2012 och Apache Giraph 2010. Neo4j är den mest populära grafdatabasen med god marginal; den estimerades 2017 vara upp till sex gånger mera populär än den näst mest populära databasen med grafdatafunktioner, OrientDB [1] [2]. Utgående från detta faktum behandlas Neo4j som ett typexempel på grafdatabaser i kommande kapitel.

1.2 Grundläggande skillnader mellan grafdatabaser och relationsdatabaser

Relationsdatabaser representerar och sparar data med hjälp av tabeller. Grafdatabaser använder sig av grafstrukturer med noder och bågar för att representera och spara data. Denna fundamentala arkitekturskillnad är en av de mest betydande olikheterna databastyperna emellan [6]. Se illustration i figurerna 1 och 2.



Kategori (K)

ID	Namn
1	Böcker
2	Skönlitteratur
3	TV-serier & filmer
4	Filmer

Betyg (B)

ID	Värde	Produkt
1	4	1
2	3	2
3	4	3
4	5	3

finns i

P_ID	K_ID
1	2
2	4
3	4

Produkt (P)

ID	Titel	År	Typ
1	The Martian	2011	Bok
2	The Martian	2015	Film
3	Inside Out	2015	Film

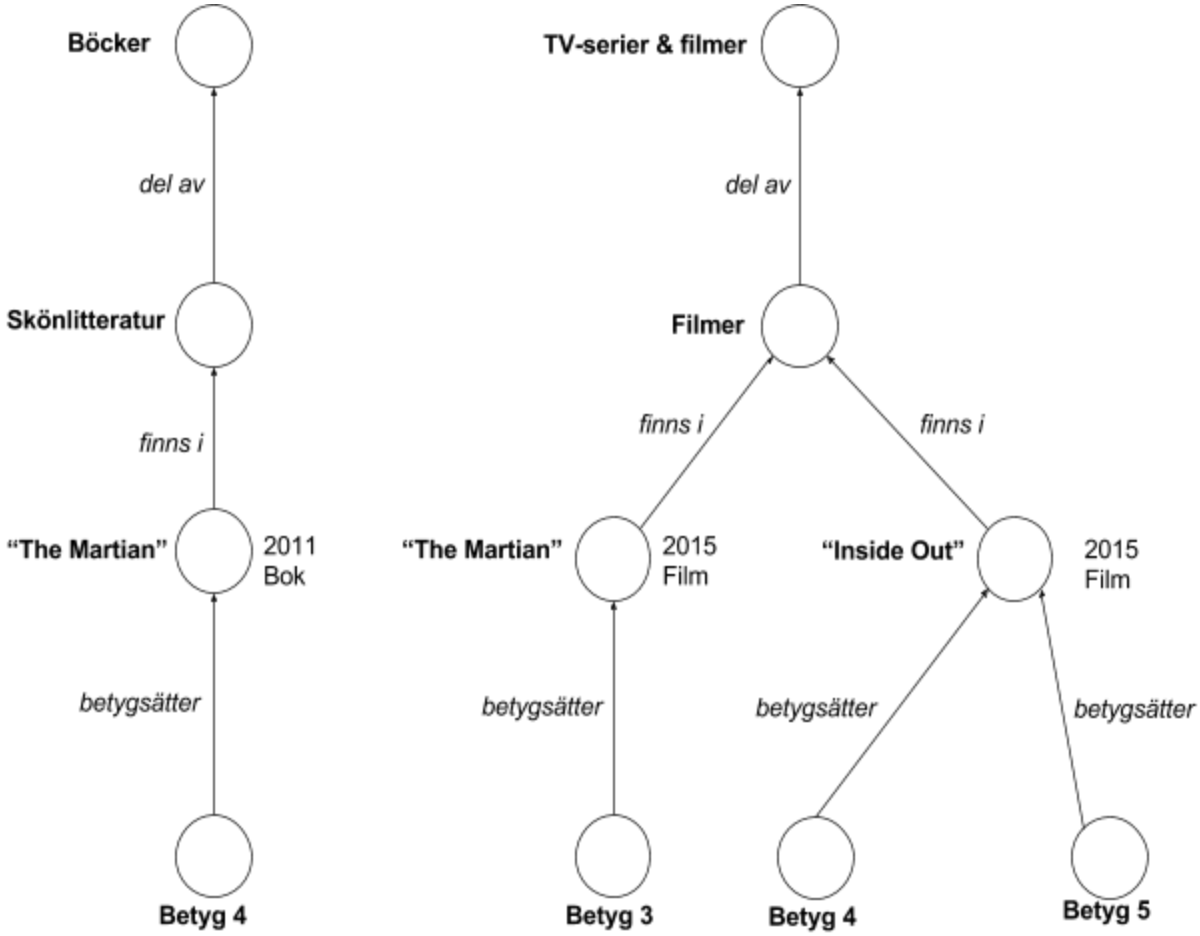
del av

K_ID_1	K_ID_2
2	1
4	3

Figur 1: exempeldata representerat med objekt och attribut i tabellformat, som i en typisk relationsdatabas.

Enligt Angles och Gutierrez är skillnaderna mellan grafdatabaser och relationsdatabaser utöver den ovan nämnda fortfarande mångfaldiga. En skillnad har att göra med typen av data som de två olika databasmodellerna är lämpliga för. Relationsdatabasen är väl anpassad för enklare registerdata, såsom flygplansbokningar eller inventarier, där datastrukturen är känd på förhand. Grafdatabaser är bättre anpassade för data där schemat behöver ändras då ny information kommer in i bilden, och där datastrukturen är mera diffus och föränderlig. Schemat för relationsdatabaser är fixerat och bestämt, och är således inte ändamålsenligt för att utvidga eller förändra. Vissa operationer som i en

relationsdatabas innebär en omstrukturering av tabellscheman, kan i grafdatabaser skötas genom ett tillägg av en enda nod [7].



Figur 2: samma data som i figur 1 representerat i grafformat med noder, bågar och attribut, som i en typisk grafdatabas.

Informationens fokus finns på olika ställen; i en relationsdatabas ligger informationens fokus på attributen i tabellerna, medan i en grafdatabas ligger informationens fokus på relationerna, alltså bågarna, som finns mellan noderna [5]. Relationsdatabasmodeller är optimerade för aggregerad data, medan grafdatabaser är optimerade för sammanhängande data [8].

Vissa operationer är grafdatabasen speciellt lämpad för, som att traversera längs med en graf. Detta är enkelt med grafdatabasen Neo4j; processen utförs genom att med hjälp av inbyggda metoder navigera från en nod till en annan, en process med linjär tidsutveckling. Motsvarande uppgift i relationsdatabasen MySQL blir mera komplicerat. En JOIN-operation (förening av tabeller) krävs för att utföra en traversering, och dessa JOIN-operationer har en exponentiell tidsutveckling baserat på antalet relationer som traverseras [9]. Resultatet blir att Neo4j i traverseringsförfrågningar kan utföra uppgiften till och med tio gånger snabbare än MySQL [10].

Relationsdatabaser är ett mer allmänt accepterat fenomen, och kan klassas som ett väletablerat verktyg. Tack vare detta finns en bred uppsättning av stödverktyg, guider och hjälpmedel. Företag som Oracle och Microsoft erbjuder bred support för sina kommersiella relationsdatabaser, och har investerat både ekonomiskt och imagomässigt i sina databassystem. Grafdatabaser är ett nyare fenomen med mindre marknadsandel, och erbjuder mindre support och stöd. Kommersiella grafdatabaser är generellt småskaliga och inriktade på en viss nisch, jämfört med massiva relationsdatabaser som Oracle, MySQL och Microsoft SQL Server, som har ett bredare användningsområde.

2 Grafdatabasens generella arkitektur

I detta kapitel beskrivs de generella egenskaper som kännetecknar grafdatabaser och skiljer dem åt från relationsdatabaserna. Vidare tas undergrupperingar av grafdatabaser fram, baserat på vilka typer av grafstrukturer de är gjorda för att hantera.

2.1 Typiska egenskaper

Enligt Angles och Gutierrez går det att beskriva tre grundläggande kännetecken för en grafdatabasmodell. Det första kännetecknet är själva grafen; datan och schemat i fråga representeras inom grafdatabasmodeller som en riktad graf, där noder har riktade bågar emellan varandra för att kännetecknа relationer. Det andra kännetecknet gäller manipuleringen av datan; datamanipulering görs genom graftransformationer eller andra relaterade operationer som genomförs på grafens vägar och mönster. Det tredje kännetecknet är integritetsbegränsningar, som används för att förstärka konsistensen på datan. Exempel på integritetsbegränsningar är identitetsintegritet (unika namn på element) och funktionella beroenden [5].

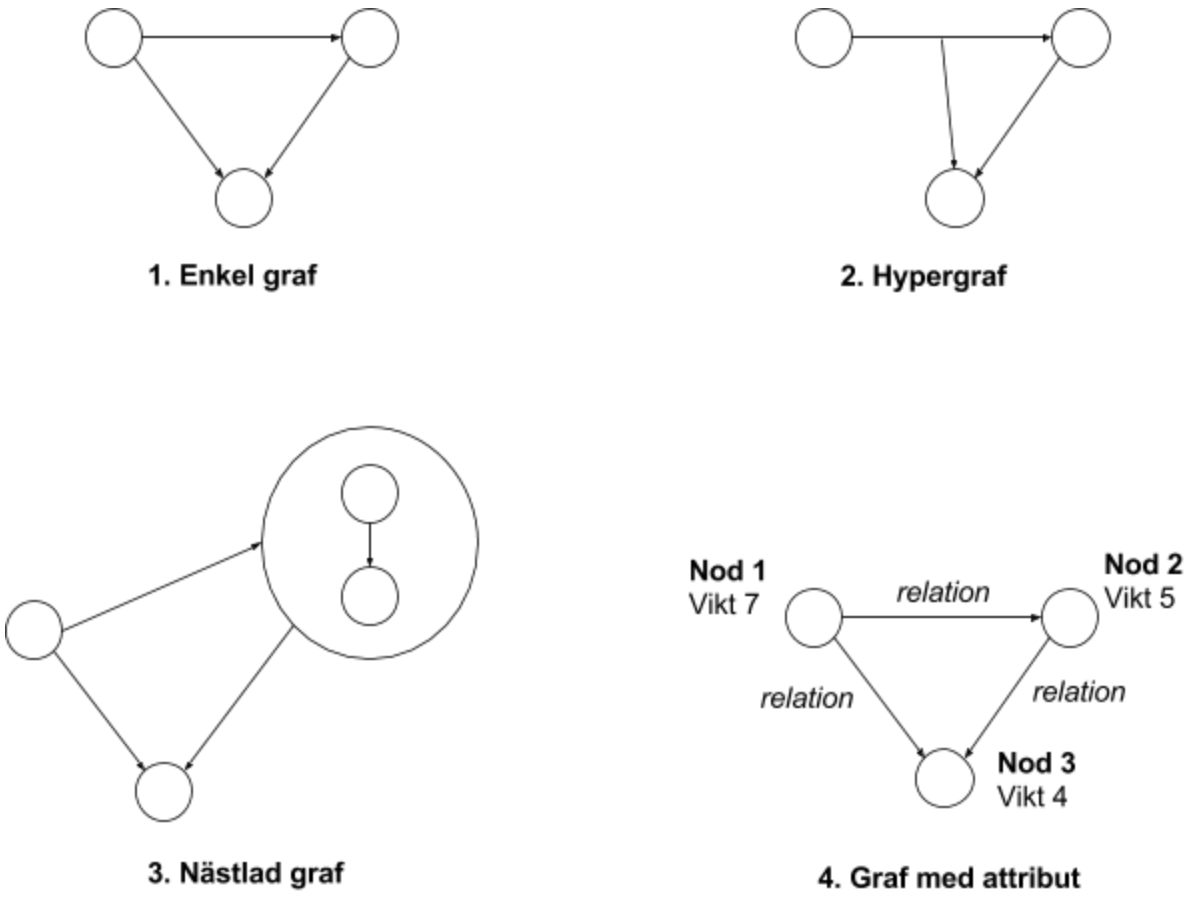
Bågarna mellan noderna i en grafdatabasmodell kan vara riktade eller oriktade, beroende på om relationen är symmetrisk eller inte. Om alla bågar är riktade kallas grafen för en riktad graf. Icke-riktade grafer kan klassas som ett särskild typ av riktade grafer, eftersom att varje båge kan ses som en riktad båge, i båda riktningarna [11].

2.2 Undergrupperingar av grafdatabaser

De mest kompletta grafdatabaserna som är i användning idag och som alla har ett eget API (*application programming interface*), databasspråk, databasmotor, lagringsmotor, transaktionsmotor samt hanterings- och driftsfunktioner, är AllegroGraph, DEX, HypergraphDB, InfiniteGraph, Neo4j och Sones [3]. Av dessa är Neo4j den mest använda.

Grafdatabaser går att dela in i undergrupperingar baserat på olika kriterier. En indelning baserar sig på vilka typer av grafdatastrukturer som grafdatabaserna är designade att hantera. Fyra stycken grafdatastrukturer kan beskrivas: enkla grafer,

som endast innehåller noder och bågar; hypergrafer, som därtill tillåter en båge att vara relaterad till ett godtyckligt antal noder; nästlade grafer, som möjliggör noder att bli egna grafer (kallade hypernoder); samt grafer med attribut, där varje nod och båge kan ha egna attribut som beskriver deras innehåll och egenskaper. För en visuell beskrivning av dessa strukturer, se figur 3.



Figur 3. Exempel på fyra grundläggande grafdatastrukturer enligt Angles [3].

AllegroGraph implementerar en simpel graf som datastruktur och HyperGraphDB implementerar hypergrafen som en datastruktur. En hypergraf tillåter en viss båge att sammankoppla mera än två noder. Enligt Angles "möjliggör denna modell en naturlig representation av relationer av högre ordning och är speciellt användbar för att modellera områden som kunskapsrepresentation, artificiell intelligens och

bioinformatik”. Ingen av de exempelgrafdatabaser som nämndes ovan implementerar nästlade grafer som datastruktur. DEX, InfiniteGraph och Neo4j implementerar grafer med attribut som datastruktur, och därtill implementerar Sones både hypergrafer och grafer med attribut [3].

Utöver dessa fyra grafstrukturer kan även multigrafer beskrivas. En multigraf är en graftyp som kan ha fler än en båge mellan två specifika noder. Detta är användbart i modellering av till exempel telefonnätverk, då flera kopplingar mellan de två noderna kan representera flera samtal [11]. Det är också användbart i en modellering där olika relationstyper finns, som till exempel i en modellering av en filmindustri där en person både har regisserat och medverkat som skådespelare i en viss film. Neo4j innehåller multigraffunktionalitet [9].

3 Grafdatabasens användningsområden

I detta kapitel beskrivs specifika situationer där grafdatabasen blir den mest relevanta databasmodellen att använda. Vidare jämförs grafdatabasen Neo4j:s egenskaper som snabbhet, skalbarhet och säkerhet med relationsdatabasen MySQL:s motsvarande egenskaper.

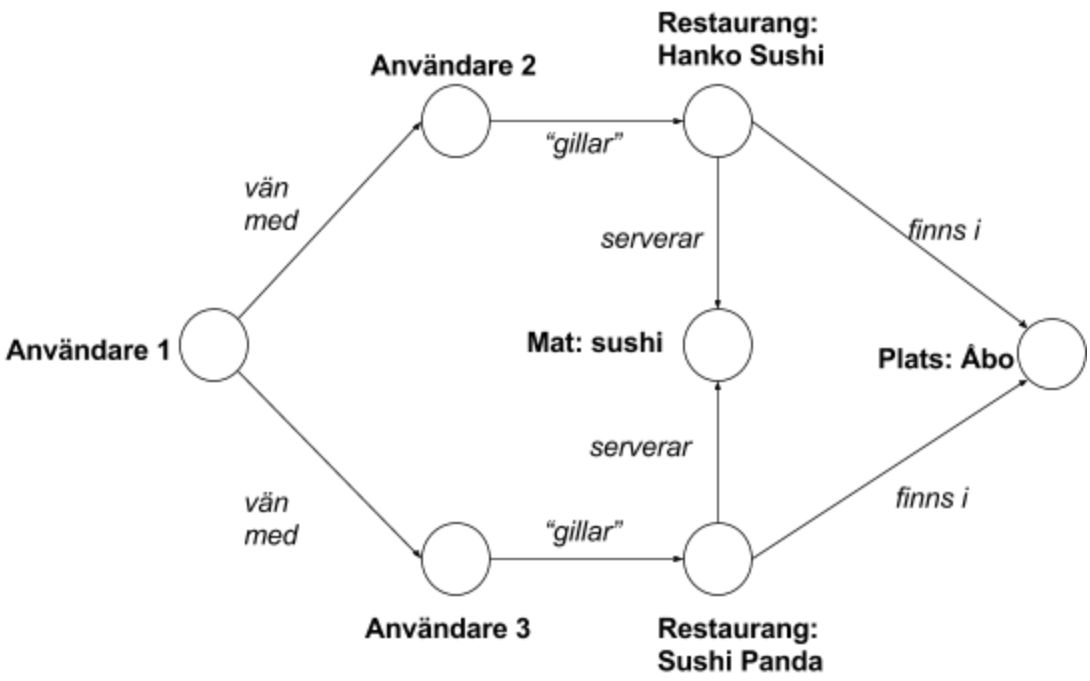
3.1 Typiska användningssituationer

Typiska användningssituationer för grafdatabaser innehåller ofta informationsvisualisering som ett centralt krav och behandlar modellering av nätverksstrukturer. Ett av de mest passande områdena för grafdatabasanvänding är modellering av sociala nätverk; bland fyra studier av grafdatabaser har alla fyra nämnt sociala nätverk som ett primärt användningsområde [5] [7] [8] [11]. Sociala nätverk kan modelleras för att utföra social nätverksanalys (SNA), som blivit ett nyckelområde för modern sociologi, mycket tack vare de stora och relativt

lättillgängliga datamängder som finns tillgänliga för analys via de olika sociala nätverken på Internet [12].

Dominguez-Sal beskriver sociala nätverk som en mycket relevant område för användning och testning av grafdatabaser: “sociala nätverksanalyser är väldigt signifikanta eftersom de använder sig av nästan alla av grafdatabasens operationstyper, är lätta att förstå av slutanvändaren och innehåller många naturliga förfrågningar med svar som är lätta att förstå konceptuellt” (egen översättning) [11]. I sociala nätverk kan noderna kan stå för antingen en person eller en grupp och bågarna kan till exempel stå för olika typer av personliga relationer, såsom vän, familjemedlem, föreningsmedlem, arbets- eller forskarkollega. En fallstudie av ett akademiskt verktyg för modellering och undersökning av historiska, sociala nätverk mellan europeiska politiker under 1900-talet gås igenom i kapitel 5 av denna kandidatavhandling.

Flera av de större IT-aktörerna inom området sociala nätverk, såsom Facebook, Twitter och Google använder sig av grafteknologier och relaterade databassystem, för att överblicka sina användare, produkter och data. Dessa företag har utvecklat egna skräddarsydda system för ändamålet och använder sig i regel inte av Neo4j eller andra kommersiella grafdatabaser. Detta på grund av dessa företags massiva storlek och deras omfattande mängder av tillgänliga data. Facebook, den mest använda sociala nätverksplattformen på Internet, har utvecklat ett egen verktyg, “Facebook Social Graph”, för att visualisera sociala nätverk. Social Graph-verktyget är inte en grafdatabas i sig, utan är mer jämförbar med en sökmotors datastruktur. Verktyget använder sig ändå arkitekturmässigt av många liknande koncept som grafdatabaserna gör [13]. Ett exempel på hur detta system fungerar är illustrerat i figur 4. En sådan här system är möjlig att implementera i en grafdatabas, till exempel Neo4j, och kan då fungera till exempel som ett rekommendationssystem.



Figur 4: Ett socialt nätverk där användare, restauranger, en mattyp och en plats finns avbildade. Med hjälp av nätverket kan "Användare 1" bli rekommenderad restauranger som hen troligen kommer att vara intresserad av.

Rekommendationssystem är ett passande område för grafdatabaser och omnämndes i två av studierna. Rekommendationssystem är tillämpade i applikationer som Netflix och Spotify för att ta fram nya förslag på film och musik åt användare. Grafdatabasens styrka inom detta område är att både liknande produkter och användare kan kopplas ihop med olika bågstyrkor för att skapa nätverk som kan understöda rekommendationsprocessen [8] [11]. Biologiska nätverk nämndes i tre av studierna. Modellering av gendata och proteininteraktioner är speciella områden inom biologiska nätverk där grafdatabaser fungerar som ett gångbart alternativ. Även kemiska processer passar bra för modellering med grafdatabaser [14].

Angles och Gutierrez identifierar ytterligare två områden där grafdatabasen blir ett motiverat verktyg. Det första av dessa är generella informationsnätverk där informationsflöden är en viktig faktor, som modellering av citationer i akademiska

publikationer eller peer-to-peer-nätverk. Det andra exemplet är teknologiska nätverk såsom tåg-, flyg, vatten- och telefonnätverk. Inom dessa teknologiska nätverk är geografiska och spatiella faktorer viktiga, element som är svåra att representera med andra databasmodeller men möjliga att representera med grafdatabaser [5]. Olika web-relaterade användningsområden, såsom den semantiska webben och länkstrukturer på websidor är också omnämnda [7] [8].

3.2 Specifika, tekniska användningskriterium

I vissa situationer kan både graf- och relationsdatabaser se ut som gångbara alternativ för användning. I sådana fall kan det bli aktuellt att utvärdera mera tekniska aspekter hos de olika databastyperna. Enligt Vicknair et. al., som utvärderat grafdatabaser kontra relationsdatabaser genom att jämföra Neo4j och MySQL, har relationsdatabaser den fördelen gentemot grafdatabaser att ett programspråk, SQL, är generellt accepterat som en standard för majoriteten av relationsdatabaserna, medan de flesta grafdatabaser har egna språk. Detta gör att transitioner och sammankopplingar mellan olika grafdatabaser blir svårare än när det gäller relationsdatabaser [10]. Detta moment kan dock överkommas genom att använda endast en databas för ett visst utvecklingsprojekt.

Om säkerhet är en prioritet, till exempel gällande ett större system med användare och data som bör vara sekretessbelagda, är relationsdatabaser ett bättre alternativ. Grafdatabaser har i många fall inte haft som en standard att erbjuda specifika säkerhetssystem [7]. Neo4j har saknat stöd för flera användare än en per databas, något som MySQL länge erbjudit. Från och med Neo4j Enterprise Edition 3.1 har dock några hanteringsfunktioner för flera än en användare introducerats [15]. MySQL är ändå mer avancerat inom säkerhetsområdet, med en lång historia av omfattande multianvändarstöd och ACL-säkerhet (*Access Control List*), som är en kontrollista över de användartillstånd som ett visst objekt har [10].

Enligt Vicknairs studie presterar en relationsdatabas bättre resultat gällande numeriska förfrågningar än en relationsdatabas, speciellt om testdatan rör sig kring stora tal eller decimaltal. Gällande textförfrågningar så är situationen omvänd: grafdatabasen presterar bättre resultat än relationsdatabasen. Storleken på ett projekt är en annan viktig faktor. Neo4j är en liten och effektiv plattform som kan användas för både små, lokala projekt och större, server-baserade projekt. MySQL är avsedd för användning som en större serverdatabas och kan således vara ett lite väl omfattande verktyg att använda för mindre projekt, även om databasen är mycket väl optimiserad [10].

4 En genomgång av grafdatabasen Neo4j

I detta kapitel gås den specifika grafdatabasen Neo4j igenom. Dess bakgrund, arkitektur och användning beskrivs. I kapitel 2 beskrevs flera vanliga grafdatabaser; av dessa var Neo4j den mest använda grafdatabasen år 2017 enligt db-enginges.com. Neo4j är tillgänglig i en gratisversion med öppen källkod.

4.1 Neo4j: Bakgrund och design

Neo4j var den mest använda grafdatabasen i mars 2017 enligt Solid IT:s sida db-engines.com, som är en ledande spårningssida för databaser [1]. Databasen är utvecklad av företaget Neo Technology, Inc. och släpptes i sin första kommersiella version år 2007. Koden som ligger som bas till verktyget Neo4j är skrivet med programmeringsspråket Java. Neo4j innehåller en egen dedikerad minneshantering, stöder realtidsförfrågningar och skalbara minnesoperationer och uppfyller ACID-transaktionskraven (*Atomicity, Consistency, Isolation, Durability*). Enligt utvecklarna Neo Technology Inc. lämpar sig databasen både för mindre lokala databasprojekt och mera prestandakrävande projekt vars processering kan distribueras

över många datorer. Neo4j:s kommersiella version kallas “Enterprise Edition”, men en gratisversion med öppen källkod är också tillgänglig vid namn “Community Edition” [9]. I kapitel 4.3 beskrivs flera företags kommersiella databassystem där Neo4j är en grundläggande komponent.

I version 3.0 av Neo4j, som släpptes i november 2015, introducerades ett kommunikationsprotokoll för klient-server-kommunikation vid namn Bolt. Protokollet är egenutvecklat av Neo Technology, Inc. och möjliggör utveckling av och stöd för klient-server-applikationer designade med Neo4j. Bolt fungerar över TCP (*Transmission Control Protocol*) och kan implementeras med hjälp av programmeringsspråken Java, JavaScript eller Python [16]. Tidigare versioner av grafdatabasen Neo4j inte erbjudit något specifikt säkerhetssystem i sig, utan utgått ifrån att den virtuella miljön är säker [7]. I takt med att användarbasen har växt, så har också fler säkerhetsfunktioner lagts till i de nyare versionerna av Neo4j Enterprise Edition. Funktioner som användarprivilegier, föfrågningsuppföljning och loggning av användarhändelser är tillgängliga från och med Enterprise Edition 3.1 [15].

Neo4j innehåller extensionsmöjligheter och kan utvidgas med hjälp av procedurer. Detta innebär att funktioner som inte finns inbyggda i Neo4j relativt lätt kan läggas till enligt behov. Eftersom Neo4j är utvecklat med hjälp av Java, så kodningsspråket för dessa extensionsprocedurer även Java. För att tillsätta en procedur som kodats i Java, så ska den kompileras och sparas i filformatet .jar (*Java Archive*) och läggas till i Neo4j:s plug-in-mapp. Efter en omstart av Neo4j kan denna procedur anropas med hjälp av Neo4j:s egna programspråk Cypher (genom den inbyggda `CALL`-funktionen), och den nya procedurens funktionalitet blir då tillagd i databasprojektet. Procedurer är användbara för att till exempel definiera manual indexering, utrusta Neo4j med tillgång till tredjepartssystem eller för att utföra avancerade schemaoperationer [9]

4.2 Neo4j: Praktisk användning och programspråket Cypher

Neo4j använder sig av ett egenutvecklat förfrågningsspråk, Cypher, för att behandla förfrågningar [17]. Cypher är baserat på förfrågningsspråket SQL (*Structured Query Language*), som används i de flesta relationsdatabaser. Noder i grafemodellen skapas genom `CREATE`-kommandot, vilket kan jämföras med `INSERT_INTO`-kommandot i SQL som används för att skapa objekt i tabellerna. Nodernas attribut specificeras inom klammerparenteser. Cypher-syntaxen för att skapa en nod är följande:

```
CREATE (nodnamn:nodtyp {nodattribut_1:'ett_attribut',
nodattribut_2:'ett_annat_attribut', …})
```

Bågar skapas också med hjälp av `CREATE`-kommandot, och bågarnas eventuella attribut specificeras även inom klammerparenteser. De noder som bågen sammankopplar måste definieras och själva bågen skapas med hjälp av en pil och klamrar. Cypher-syntaxen för att skapa en båge är följande:

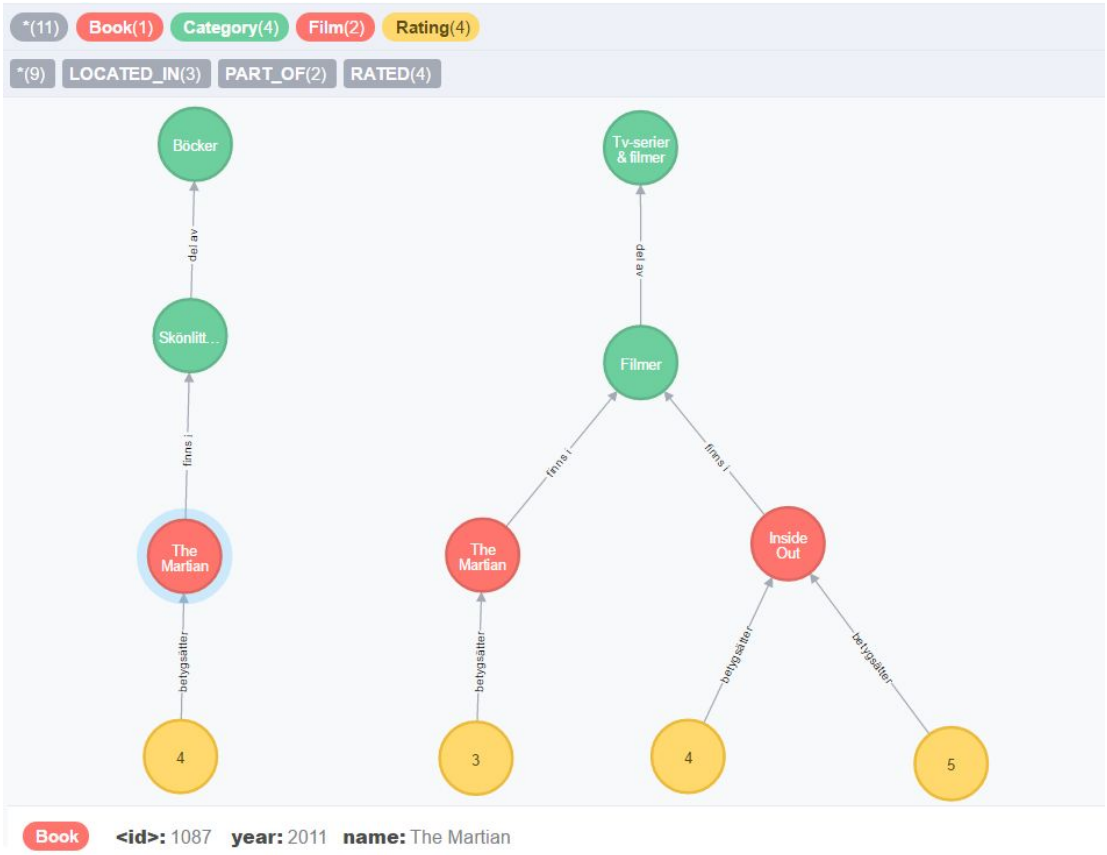
```
CREATE (nodnamn_1)-[:relationstyp
{relationsattribut:['relaterad_till']}]->(nodnamn_2)
```

För att hämta data ur databasen används `MATCH`-kommandot, som kan jämföras med SQL-kommandot `SELECT`. Genom att endast använda `MATCH`-kommandot kan alla noder som i databasen hämtas. `WHERE`-kommandot kan användas i kombination med `MATCH`-kommandot, på samma sätt som `SELECT-WHERE`-kombinationen i SQL, för att hämta en eller ett visst antal noder baserat på något kriterium. Cypher-syntaxen för att hämta en viss nod, vars ID är 0, är följande:

```
MATCH (n)
WHERE id(n)= 0
RETURN n
```

Neo4j har två skilda huvuddelar: kodgränssnittet och grafgränssnittet. I kodgränssnittet kan noder, bågar och attribut skapas och sparas med hjälp av kodspråket Cypher. Där kan också förfrågningar skapas för att hämta de önskade data som finns i databasen. I grafgränssnittet visualiseras de data som hämtas och där kan också de olika elementens utseende manipuleras visuellt, till exempel genom att byta färg på alla noder som hör till en viss grupp [9]. Se figur 5 för en enkel visualisering i Neo4j (Cypher-koden finns i appendix-delen).

En slutsats som kan dras från användningen av Neo4j är att mängder av noder och bågar visualiseras effektivt, medan nod- och bågattribut endast modelleras med textdata. Detta medför att till exempel kvantitativa nodattribut inte blir ideala för modellering; att kunna ändra storlek på noderna utgående från attributdata är inte en funktion som finns inbyggd i Neo4j. Kvantiteter modelleras inbyggt bättre med hjälp av nod- och bågantal. Om funktionen att ändra nodstorlek behövs i ett projekt måste den konstrueras med ett utomstående verktyg, vilket kan ses som en svaghet hos Neo4j. Ytterligare positiva aspekter hos Neo4j kan dock hittas: ett användarvänligt gränssnitt, ett välutvecklat förfrågningsspråk baserat på SQL, Cypher, och en allt mer heltäckande support.



figur 5: En modellering av samma testdata som i figurerna 1 och 2, implementerad i Neo4j.

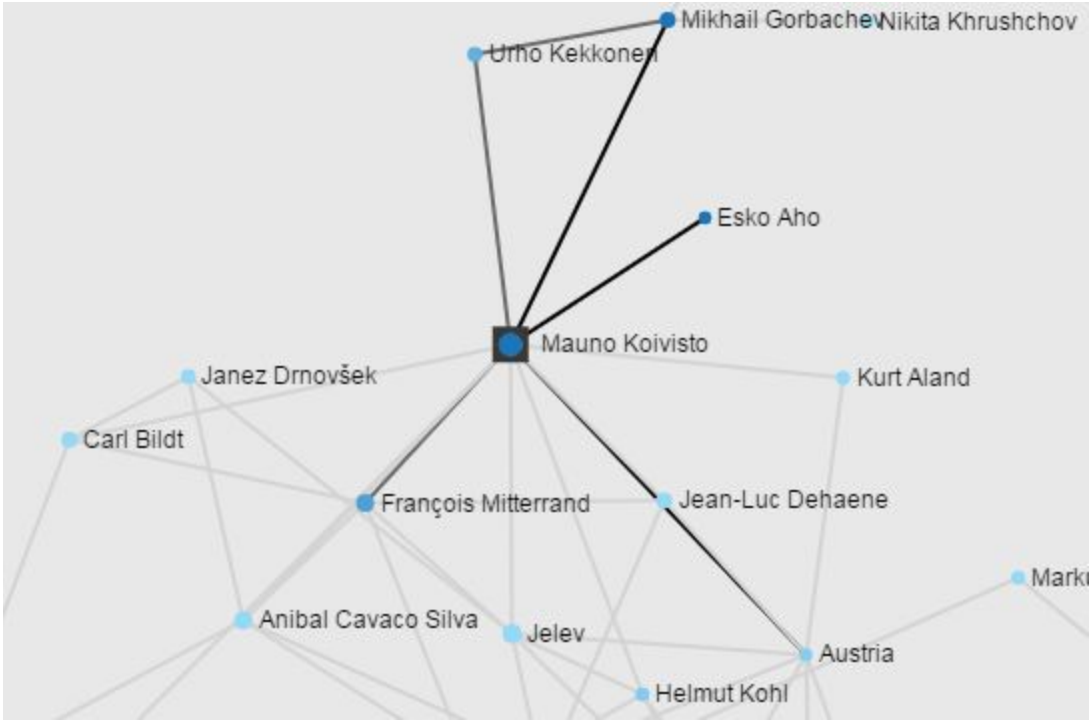
4.3 Neo4j:s kommersiella användningsområden

Neo4j Enterprise Edition används kommersiellt av företag inom finanssektorn, hälsovård, IT, media, detaljhandeln, telekommunikationer och transportsektorn. Konkreta exempel är detaljhandelskedjan Walmart, som använder en skräddarsydd plattform byggd med Neo4j för att generera varurekommendationer i realtid åt sina kunder samt internetföretaget Ebay, som använder sig av ett multiplattformsystem där Neo4j är en av komponenterna, för att hantera e-handelsledning och planering av leveranser [18].

5 histoGraph: CVCE:s Neo4j-baserade projekt

Som beskrivet i kapitel 4.3, så används Neo4j i många kommersiella sammanhang. Grafdatabasen är också användbar i forskningsammanhang, vilket tas fram i detta kapitel genom en undersökning av projektet histoGraph, ett interaktivt verktyg för utforskande visualisering och undersökning av historiska sociala nätverk. histoGraph har konstruerats med Neo4j som bas. histoGraph består av tudelat användargränssnitt; i den ena delen kan nyhetsartiklar, bilder och media överblickas och i den andra delen illustreras relationerna mellan personer och teman i grafformat. I detta kapitel diskuteras främst den andra delen.

5.1 Beskrivning och syfte



Figur 6: Skärmdump från grafverktyget i plattformen histoGraph, utvecklat av CVCE. Bilden är genererad med en öppen testversion av plattformen, tillgänglig via adressen histograph.cvce.eu.

histoGraph är ett interaktivt verktyg för utforskande visualisering och undersökning av historiska sociala nätverk och specifikt behandlar plattformen europeisk integration under 1900-talet [19]. histoGraph-plattformean har utvecklats av forskare inom MDI (människa-datorinteraktion) samt programmerare och historiker vid institutionen Centre virtuel de la connaissance sur l'Europe (CVCE) i Sanem, Luxemburg. Den primära användarmålgruppen för histoGraph-verktyget är historiker och forskare inom historia, som vill analysera relationer mellan politiker, teman och personer. CVCE har använt sig av verktyget Neo4j för att utveckla och bygga upp histoGraph. [20]. histoGraph skapar ett nätverk av personer baserat på i hur många bilder de förekommer tillsammans i källmaterialet. Källmaterialet är europeiska nyhetsartiklar och reportage under 1900-talet.

5.2 Struktur och användning

En vikitig distinktion gällande histoGraph-projektet är det mellan databasens struktur, som bestäms av Neo4j, och visualiseringen av informationen, som innehåller CVCE:s egenutvecklade processer och metoder. Skälen till att projektet använder sig av en grafdatabas (Neo4j) går att utläsa ur designkraven; ett socialt nätverk av personer är det som ska modelleras, där relationerna är den viktigaste komponenten. Som beskrivet i kapitel 3.1 är sociala nätverk något av de områden som grafdatabaser passar bäst för.

Grunden till histoGraph är Neo4j, och relationerna mellan alla entiteter i histoGraph (människor och ämnen) sparas back-end-mässigt i en Neo4j-databas. Enligt Wieneke: "Detta tillvägagångssätt skulle innebära förfrågningar som är beräkningsmässigt omfattande i en relationsdatabas, men som är enkla att utföra med en grafdatabas, som till exempel beräkningar av sökvägar för entiteter (noder) som inte är direkt sammankopplade" (egen översättning) [21]. Metoder för att visualisera nätverket har utvecklats utgående från databasinnehållet. När en användare väljer att ta fram en viss

person, så utför kommandot `MATCH-WHERE` i Neo4j-databasen för att hämta personen och alla hens relationer.

Den grafiska delen av histoGraph (figur 6) kan jämföras med Neo4j:s grafgränssnitt (figur 5). Utvecklarna från CVCE har lagt till många egenskaper utöver Neo4j:s inbyggda grafgränssnitt. Användarvänligheten är relativt hög och manipulation av grafens data är lätt. Källmaterialets kriterium kan ändras, vilka år som källmaterialet tas från kan begränsas. Om två personer förekommer i många bilder tillsammans så räknas deras relation vara starkare, och de är således närmare relaterade till varandra i nätverket. Relationernas styrka representeras med hjälp av tjockleken på bågen; en tjockare båge betyder en starkare relation där personerna förekommer i samma bild flera gånger. Då en viss båge väljs av användaren, så hämtas också de foton och dokument där personerna som de två noderna representerar förekommer [19].

Vid visualiseringen av objekt kan kluster av relaterade objekt urskiljas. I de data som implementerats i histoGraph kan till exempel personers nationalitet vara en avgörande faktor som sammanbinder och avgränsar ett visst kluster. Användarna kan på ett intuitivt sätt se sambanden mellan olika personer och länder [19].

6 Resultat och avslutande diskussion (1-2 sidor)

I detta avslutande kapitel diskuteras de sammanfattade resultaten av grafdatabserna kontra relationsdatabaser, grafdatabasens generella struktur och användningsområden, Neo4j:s utvärdering och fallstudien gällande histoGraph-plattformen. Slutsatser som kan dras utgående från dessa utvärderingar som helhet tas även fram.

6.1 Resultat och tolkning

Resultaten av undersökningen av de situationer där grafdatabaser är ett bra alternativ avslöjar att nätverksmodellering i de flesta former är ett område där grafdatabaser fungerar som bäst.

Beskrivningen, undersökningen och användningen av Neo4j visar att verktyget är användbart, både kommersiellt och privat och för större och mindre projekt. Flera kommersiella, praktiska exempel på de teoretiskt beskrivna användningsområdena för grafdatabser i kapitel 3 (sociala nätverk, rekommendationssystem och generella informationsnätverk) hittades i kapitel 4.3. Ett annat modellerat historiskat socialt nätverk, histoGraph, hittades och beskrevsi kapitel 5. histoGraph-plattformen är ett typexempel på en situation där en skräddarsydd lösning med en grafdatabas som bas är ett mycket gångbart alternativ.

6.2 Avslutande diskussion

Relationsdatabaser kommer troligen också i framtiden vara den mest använda databastypen, men inom flera databasområden kan grafdatabaser vara ett passande alternativ, speciellt gällande nätverksmodelleringar. Grafdatabasens nätverksmodellering möjliggör analys av relationer och informationsflöden. Grafdatabaser kan användas både inom nom både kommersiella applikationer och akademiska projekt. Neo4j är det mest framstående exemplet på en grafdatabas, tack vare dess relativt stora användarbas, ökande support och möjlighet för utvidgning.

7 Källor

[1] Solid IT. *DB-Engines Complete Ranking.* Tillgänglig från adressen http://db-engines.com/en/ranking, 4 april 2017.

[2] Solid IT. *Method of calculating the scores of the DB-Engines Ranking*. Tillgänglig från adressen http://db-engines.com/en/ranking_definition, 4 april 2017. *"Rankingen baserar sig på ett följande parametrar: antal resultatsidor i sökmotorer; Google Trends-index; antal omnämnanden i tekniska diskussioner, i arbetsannonser och på LinkedIn-profilsidor; samt relevans i sociala nätverk. [...] Den matematiska distansen bevaras i rankingsberäkningen. Detta betyder att om system A har ett dubbelt större rankingsvärde än system B medför det att system A är två gånger mera populärt än system B."* (egen översättning).

[3] R. Angles. "A comparison of current graph database models." *Data Engineering Workshops (ICDEW)*, 2012 IEEE 28th International Conference on. IEEE, 2012.

[4] A. Silberschatz. (28 January 2010). *Database System Concepts, Sixth Edition* (PDF). McGraw-Hill. p. D-29. ISBN 0-07-352332-1.

[5] R. Angles och C. Gutierrez, "Survey of graph database models." *ACM Computing Surveys (CSUR)* Volume 40, Issue 1 (2008).

[6] M. Rudolf *et al.*, "The Graph Story of the SAP HANA Database." *BTW*. Vol. 13. 2013.

[7] S. Batra och C. Tyagi, "Comparative analysis of relational and graph databases." *International Journal of Soft Computing and Engineering (IJSCE)* 2.2 (2012): pp 509-512.

[8] J.J. Miller, "Graph database applications and concepts with Neo4j." Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA. Vol. 2324. 2013.

[9] Neo Technology, Inc. "The Neo4j Developer Manual version 3.1". Tillgänglig från adressen http://neo4j.com/docs/developer-manual/current/, 22 mars 2017.

[10] C. Vicknair *et al.*, (April 2010) "A comparison of a graph database and a relational database: a data provenance perspective" *ACM SE* '10, Article No. 42.

[11] D. Dominguez-Sal *et al.*, "A discussion on the design of graph database benchmarks." *Technology Conference on Performance Evaluation and Benchmarking*. Springer Berlin Heidelberg, 2010.

[12] R. Angles *et al.*, (2013, June). "Benchmarking database systems for social network applications" in *First International Workshop on Graph Data Management Experiences and Systems* (p. 15). ACM.

[13] E. Eifrem och P. Rathle. (17 januari 2013) Why the most important part of Facebook Graph Search is 'Graph'. Tillgänglig via adressen https://neo4j.com/blog/why-the-most-important-part-of-facebook-graph-search-is-graph/, 4 april 2017.

[14] M. Graves *et al*., "Graph database systems for genomics." *IEEE Engineering in Medicine and Biology Magazine* 14.6 (1995): 737-745.

[15] I. Borojevic. “Security and Access Control in Neo4j Enterprise Edition”. *Neo Technology, Inc* (2017).

[16] A. Kollegger, “The Neo4j 3.0 Milestone 1 Release”. Tillgänglig från adressen https://neo4j.com/blog/neo4j-3-0-milestone-1-release/, 22 mars 2017.

[17] F. Holzschuher och R. Peinl, "Performance of graph query languages: comparison of cypher, gremlin and native access in Neo4j." *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. ACM, 2013.

[18] K. Nixon, “Sustainable Competitive Advantage: Creating Business Value through Data Relationships,” Neo Technology Inc., San Mateo, CA (februari 2015).

[19] J. Novak, *et al.*, "HistoGraph--A visualization tool for collaborative analysis of networks from historical social multimedia collections." Information Visualisation (IV), 2014 18th International Conference on. IEEE, 2014.

[20] L. Wieneke *et al.*, (2014) “histoGraph - graph-based exploration and crowd-based indexation”. Tillgänglig från adressen http://www.cvce.eu/digital-innovation/projects/netviz/histograph, 21 mars 2017.

[21] L. Wieneke *et al.*, (2016). Introducing HistoGraph 2: Exploration of Cultural Heritage Documents Based on Co-Occurrence Graphs. In Digital Humanities 2016: Conference Abstracts. Jagiellonian University & Pedagogical University, Kraków, pp. 400-402.

8 Appendix: Cypher-kod för figur 5

Här beskrivs Cypher-koden som användes för att skapa grafen i figur 5.

```
CREATE (TMB:Book {name:'The Martian', year:2011})
CREATE (TMF:Film {name:'The Martian', year:2015})
CREATE (IOF:Film {name:'Inside Out', year:2015})

CREATE (R01:Rating {rating:4})
CREATE (R02:Rating {rating:3})
CREATE (R03:Rating {rating:4})
CREATE (R04:Rating {rating:5})

CREATE (BOOKS:Category {name:'Böcker'})
CREATE (NONFICTION:Category {name:'Skönlitteratur'})
CREATE (TV_SERIES_AND_FILMS:Category {name:'Tv-serier & filmer'})
CREATE (FILMS:Category {name:'Filmer'})

CREATE
       (R01)-[:RATED {name:['betygsätter']}]->(TMB),
       (R02)-[:RATED {name:['betygsätter']}]->(TMF),
       (R03)-[:RATED {name:['betygsätter']}]->(IOF),
       (R04)-[:RATED {name:['betygsätter']}]->(IOF),

       (TMB)-[:LOCATED_IN {name:['finns i']}]->(NONFICTION),
       (TMF)-[:LOCATED_IN {name:['finns i']}]->(FILMS),
       (IOF)-[:LOCATED_IN {name:['finns i']}]->(FILMS),

       (NONFICTION)-[:PART_OF {name:['del av']}]->(BOOKS),
       (FILMS)-[:PART_OF {name:['del av']}]->(TV_SERIES_AND_FILMS)
```

Noderna och bågarna visualiseras genom följande Cypher-kod:

```
START n=node(*) MATCH (n)-[r]->(m) RETURN n,r,m;
```