INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2026-06-11

EXAMENSARBETE Improving the Compilation Time of Semantically Similar Cypher Queries
STUDENTER Jacob Johansson, Jonatan Svahn
HANDLEDARE Niklas Fors (LTH), Filip Hedén (Neo4j), Henrik Nyman (Neo4j)
EXAMINATOR Görel Hedin (LTH)

Snabbare hantering av semantiskt liknande grafdatabasfrågor

POPULÄRVETENSKAPLIG SAMMANFATTNING **Jacob Johansson, Jonatan Svahn**

Varje gång du söker i en databas översätts din fråga till en detaljerad plan för hur svaret ska hämtas. Den planen tar tid att skapa. Vi presenterar två optimeringar för grafdatabasen Neo4j som kan göra översättningen från fråga till plan snabbare.

Databaser är en självklar del av alla stora verksamheter. Grafdatabaser är en specifik typ av databas som har vuxit i popularitet under de senaste åren. Grafdatabaser sparar all information i ett nätverk av noder med relationer mellan dem. Den populäraste grafdatabasen är utan tvekan Neo4j. Neo4j använder ett programmeringsspråk som kallas *Cypher* för att kommunicera med databasen. Med hjälp av Cypher kan en användare hämta ut data genom att beskriva vad man letar efter i grafen i form av en Cypher-fråga. Innan frågan kan besvaras måste databasen översätta den till en *plan*, det vill säga räkna fram det mest effektiva sättet att hämta ut informationen man är ute efter.

Översättningen av en fråga till en plan kan däremot vara väldigt långsam. Stora organisationer kan skicka miljontals med frågor varje sekund till sina databaser, och om varje fråga måste översättas från grunden, kan det slösa enorma mängder resurser. Neo4j har redan vissa mekanismer för att undvika detta, genom att exempelvis återanvända planer från föregående frågor. Men för *semantiskt liknande* frågor kan prestandaproblem fortfarande uppstå. Semantiskt liknande frågor definierar vi som frågor som gör liknande eller samma sak, men som inte ser likadana ut.

I vårt examensarbete har vi implementerat och utvärderat två optimeringar, som bemöter just detta problem. Den första optimeringen kallar vi *variabelnormalisering*. Denna optimering ersätter alla namn på variabler i en fråga till neutrala, enhetliga namn. Två frågor som bara skiljer sig i variabelnamnen kommer efter denna optimering att se identiska ut, vilket tillåter en tidigare plan att återanvändas. Originalnamnen bevaras och återställs, så att ingenting förändras från användarens perspektiv. Den andra optimeringen kallar vi *fuserad operatorcachning*. Neo4j:s Enterpriseversion slår samman vissa delar av en plan till egna enhetliga komponenter, vilket ökar exekveringsprestandan. Vi presenterar en ny teknik för att återanvända komponenter mellan frågor med liknande struktur. För att öka återanvändningsgraden abstraherar vi även bort konstanta värden, vilket gör varje komponent mer generell.

Resultaten för variabelnormaliseringen visar på en markant förbättring i identiska frågor med varierande variabelnamn, upp till fyra gånger snabbare totalt. Resultaten för fuserad operatorcachning gav störst nytta i frågor med många upprepade delar, upp till 1,3 gånger snabbare översättning. Vi får också en markant minskning i minnesanvändning, upp till åtta gånger mindre. I allmänna situationer påverkas prestandan inte märkbart utav någon av optimeringarna. I vissa specifika situationer kan prestandan påverkas negativt av den andra optimeringen, då abstraherade värden som tidigare var konstanter, nu måste läsas in från datorns minne.