Študija uporabe parametriziranih predlog poizvedb v preiskovalni platformi

Štefan Kohek[1], **Aleksander Pur**[2], **Lea Roj**[1],
Matej Brumen[1], **David Jesenko**[1], **Niko Lukač**[1]

[1] *Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Koroška cesta 46, 2000 Maribor, Slovenija*
[2] *Ministrstvo za notranje zadeve, Štefanova ulica 2, 1000 Ljubljana, Slovenija*
E-pošta: stefan.kohek@um.si, aleksander.pur@policija.si, lea.roj1@student.um.si,
{matej.brumen, david.jesenko, niko.lukac}@um.si

A study on the use of parameterized query templates in the investigation platform

Digital forensics benefit from large amount of heterogeneous data, as they can be used to improve the investigation of crime cases. Data inspection is usually performed through query languages. However, query languages are usually too complex for the typical usecase. Therefore, we have developed a support for parameterised query templates that make it both easier and faster for the user to query data and thus investigate crimes more efficiently. In the context of this study, we compared the querying time by directly using the Cypher query language or through the use of parameterised query templates on several examples. We found that query templates significantly improved investigation workflow.

1 Uvod

Dandanes nas digitalizacija spremlja na vsakem koraku, zato količina elektronskih podatkov, ki jih ustvarjamo, narašča eksponentno. Ti podatki vsebujejo informacije, ki so ključne za preprečevanje, odkrivanje in preiskovanje kaznivih ravnanj. Vendar odkrivanje teh informacij ni enostavno zaradi velikih količin podatkov v raznovrstnih formatih in oblikah, v katerih je potrebno poiskati ustrezne vzorce [1, 2]. Pri tem je potrebno upoštevati, da so zbrani podatki lahko tudi pomanjkljivi in napačni.

Za preprečevanje, odkrivanje in preiskovanje kaznivih ravnanj potrebujemo ustrezna programska orodja za iskanje koristnih informacij v velikih količinah heterogenih podatkov. Namen tovrstnih analiz podatkov je predvsem iskanje izstopajočih pojavov in sumljivih povezav. Zato se namesto orodij za poslovno poročanje, kot je Microsoft PowerBI, pogosteje uporabljajo orodja, kot so i2 Analyst notebook, Sentinel, Pajek, QlikView, Geotime [3]. Glavna slabost obstoječih orodij pa je v tem, da so pogosto prilagojena za določen tip podatkov, omejena glede zmožnosti preiskovanja, uporabniško neprijazna, ne omogočajo skupinskega dela ali niso dovolj prilagodljiva.

Za namen uporabniško prijaznega in hkrati naprednega preiskovanja velike količine heterogenih podatkovnih virov smo razvili preiskovalno platformo STALITA [3], katere temelj je napredna analitika podatkov umeščenih v grafih. V platformi STALITA se uporablja NoSQL grafna podatkovna baza Neo4J, kar prinaša dinamične podatkovne modele in učinkovito iskanje koristnih informacij. Dinamično prilagajanje podatkovnih modelov vhodnim podatkom je zelo pomembno, saj skoraj vsak preiskovalni primer vključuje drugačne podatke. Podatkovni modeli na podlagi teorije grafov pa za razliko od entitetno-relacijskih modelov omogočajo učinkovitejše iskanje izstopajočih pojavov, sumljivih povezav in zapletenih vzorcev ter ne zgolj preproste statistike in trendov. Pri tem pa je ključno poznavanje poizvedovalnega jezika Cypher [4], ki je del podatkovne baze Neo4J. Neposredna uporaba tega jezika omogoča izjemno izrazno moč in fleksibilnost, vendar zahteva dobro poznavanje jezika, česar od preiskovalcev z različnimi domenskimi znanji ne moremo zahtevati.

Uporabniško prijazni način sestavljanja poizvedb je v preteklosti že bil predmet več raziskav, kot so vizualni jeziki za poizvedovanje po podatkovnih bazah [5, 6] in samodejna tvorba poizvedb na podlago uporabniško podanih primerov rezultatov poizvedb [7]. Toda prevelika splošnost in velik nabor možnih funkcionalnosti so lahko za uporabnika neugodne. Zato se v zadnjem času kot alternativen pristop za poizvedovanje po podatkovnih bazah razvija uporaba naravnega jezika [8], kar vključuje poizvedovanje v naravnem jeziku v obliki pogovora [9, 10]. Toda bistvena omejitev je v tem, da uporabnik pogosto ne ve, kateri podatki so na voljo in tudi katere poizvedbe lahko izvede. Prav tako je zgolj na podlagi pogovora kompleksnejše poizvedbe težje ponoviti.

Z ozirom na te omejitve smo za preiskovalno platformo STALITA razvili informacijsko podporo za parametrizirane predloge poizvedb, ki jih napredni uporabnik – poznavalec poizvedovalnega jezika Cypher, vnaprej pripravi. Manj napredni uporabniki pa te predloge uporabljajo, kar omogoča učinkovitejše iskanje koristnih informacij brez zahteve po poznavanju jezika Cypher. Za večjo fleksibilnost se pri predlogah poizvedb lahko dodajo parametri brez potrebe po dodatnem pisanju programske kode. Da lahko potrdimo korist uporabe predlog poizvedb, smo izvedli študijo primera uporabe predlog poizvedb. Pri tem smo preverili, ali uporaba predlog poizvedb omogoča hitrejše poizvedovanje proti ročnemu pisanju in spreminjanju poizvedb.

2 Metodologija

Preiskovalna platforma STALITA [3] je sestavljena iz zalednega dela, na katerem se hranijo podatki, izvajajo poizvedbe in upravljajo dostopi do podatkov, ter iz čelnega

```
1 | MATCH (n1:Racun {stevilka: 'A'})-
  | [t]->(n2:Racun {stevilka: 'B'})
2 | RETURN n1, t, n2
```

Slika 1: Primer poizvedbe za iskanje transakcij med transakcijskima računoma *A* in *B* v poizvedovalnem jeziku Cypher.

```
1 | MATCH (n1:Racun {stevilka: $trrA})
  | -[t]-(n2:Racun {stevilka: $trrB})
2 | RETURN n1, t, n2
```

Slika 2: Poizvedba v poizvedovalnem jeziku Cypher v predlogi poizvedbe za iskanje transakcij med transakcijskima računoma, podanima s parametroma *trrA* in *trrB*.

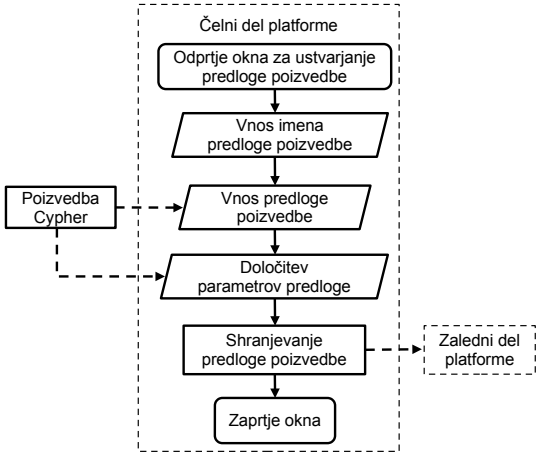
dela v obliki spletne aplikacije, ki preko grafičnega uporabniškega vmesnika omogoča dostop do zalednega dela, omogoča zagon poizvedb in prikaz rezultatov poizvedb z uporabno naprednih vizualizacij podatkov.

Spletna aplikacija platforme deluje tako, da uporabnik v spletnem brskalniku na začetku izbere preiskovalni primer in nato izvede poizvedbo, s katero analizira vnaprej naložene podatke preiskovalnega primera. Kot rezultat poizvedbe se praviloma posredujejo podatki v grafni obliki, kar pomeni, da so vsi heterogeni podatki predstavljeni z vozlišči in povezavami med njimi. Tako lahko hranimo na primer vozlišča: banka, transakcijski račun in oseba, pri čemer povezave določajo pripadnost transakcijskega računa banki in osebi ter tudi transakcije med dvema transakcijskima računoma. Dobljene podatke lahko uporabnik pregleduje v grafični [11] ali v tabelarični obliki.

Poizvedovalni jezik Cypher je temeljno orodje za preiskovanje v preiskovalni platformi. Uporabnik izvaja poizvedbe tako, da v pojavnem oknu spletne aplikacije napiše poizvedbo v jeziku Cypher in jo nato pošlje v zaledni del, kjer se poizvedba v podatkovni bazi Neo4J izvede in nato se rezultat pošlje uporabniku za nadaljnjo analizo. V primeru, da želimo pridobiti vse transakcije med transakcijskima računoma *A* in *B*, izvedemo poizvedbo, prikazano na sliki 1. Rezultat poizvedbe je seznam povezav med vozliščema *n1* in *n2*, pri čemer vsaka povezava predstavlja eno transakcijo.

Od začetnega uporabnika ne moremo pričakovati dobrega poznavanja poizvedovalnega jezika Cypher. Prav tako lahko tudi napredni uporabnik naredi napake pri sestavljanju poizvedb. Zato smo razvili podporo za parametrizirane predloge vnaprej pripravljenih poizvedb, ki jih lahko uporabnik preko grafičnega uporabniškega vmesnika poišče, ustrezno parametrizira in izvede.

Informacijska podpora za predloge poizvedb deluje tako, da napredni uporabnik vnaprej zapiše poizvedbo Cypher in ji določi kategorijo in ime. Poizvedbo na sliki 1 bi lahko kategorizirali pod "Bančni podatki" in poimenovali kot "Seznam transakcij med trr. A in B". Nato se v poizvedbi Cypher določi vnosna polja oziroma parametre, ki jih uporabnik lahko spreminja. Primer takšne poizvedbe je na sliki 2 pri čemer *trrA* in *trrB* predstavljata parametre predloge poizvedbe. Nato se v ločenem oknu določi lastnosti parametrov, v našem primeru to storimo



Slika 3: Delovanje platforme STALITA pri izdelavi predloge poizvedbe, pri čemer polna črta predstavlja tok obdelave, črtkana črta pa predstavlja tok podatkov.

```
1 | MATCH (a:Racun)-[t]-(b:Racun)
2 | RETURN DISTINCT a.stevilka
```

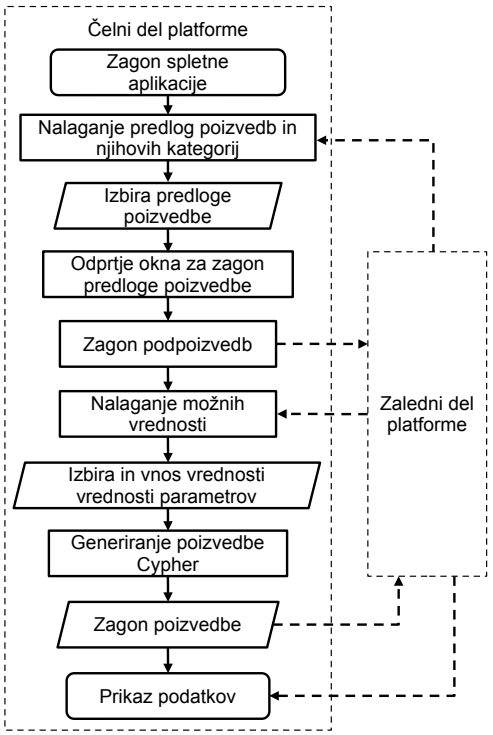
Slika 4: Primer poizvedbe v poizvedovalnem jeziku Cypher, ki vrne seznam vseh številk bančnih računov.

za *trrA* in *trrB*. Za vsak parameter določimo ime, opis in tip vrednosti. Pri tem so dovoljeni tipi vrednosti niz, število ali datum. Glede na to, da je za določene parametre predviden omejen nabor vrednosti, smo zasnovali tudi podporo za podpoizvedbe Cypher, ki za podan parameter napolnijo seznam možnih vrednosti. Delovanje platforme pri izdelavi predloge poizvedbe je prikazano na sliki 3.

Drugim uporabnikom, ki ne poznajo poizvedovalnega jezika Cypher, so predloge poizvedb dostopne preko grafičnega menija v spletni aplikaciji. V spustnem meniju se nahajajo kategorije, v katerih so imena predlog poizvedb. Ko uporabnik izbere predlogo poizvedbe, se odpre pojavno okno, kjer se prikaže zgolj seznam parametrov poizvedb z njihovimi opis. V našem primeru se prikažeta zgolj dve vnosni polji za transakcijska računa *A* in *B*. Glede na to, da smo za *trrA* in *trrB* določili podpoizvedbi Cypher za nalaganje vseh možnih številk transakcijskih računov, ki je prikazana na sliki 4, se ob odprtju okna izvedeta podpoizvedbi, ki za obe vnosni polji napolnita seznama možnih vednosti. Uporabnik nato vnese ustrezne vrednosti, pri čemer si v primeru podpoizvedb pomaga s spustnim menijem. Nato se v ozadju generira poizvedba Cypher, pri čemer se na mesta parametrov zapišejo dejanske vrednosti parametrov. Točen potek uporabe in zagona predloge poizvedb je prikazan na sliki 5.

Opisane predloge poizvedb imajo sledeče prednosti pred neposrednim pisanjem poizvedb:

- uporabniku ni potrebno poznati poizvedovalnega jezika Cypher;
- preko menija so pogosto uporabne poizvedbe dostopnejše, kar pohitri preiskovanje;
- na podlagi menija s predlogami poizvedb je možno hitro ugotoviti možne načine poizvedovanja;



Slika 5: Potek poizvedovanja s predlogami poizvedb, pri čemer polna črta predstavlja tok obdelave, črtkana črta pa predstavlja tok podatkov.

- z uporabo podpoizvedb in spustnega menija s seznamom dovoljenih vrednosti zmanjšamo verjetnost vnosa napak.

3 Rezultati in diskusija

Uporabnost predlog poizvedb smo preverili tako, da smo primerjali čas poizvedovanja petih uporabnikov z uporabo predlog poizvedb proti ročnemu vnašanju poizvedb Cypher. Poizvedovali smo nad podatkovno bazo bančnih transakcij, ki je hranila 1523 vozlišč in 8098 povezav, kar vključuje 197 transakcijskih računov in 356 nakazil. Merili smo sledeče poizvedbe:

- P1 - Izpis vsote vseh bančnih transakcij, kar smo izvedli s poizvedbo Cypher na sliki 6. P1 predstavlja torej vnaprej pripravljeno poizvedbo Cypher brez parametrov.
- P2 - Izpis bančnih transakcij, katerih znesek je višji od parametra *$znesek*, kar smo izvedli s poizvedbo na sliki 7. P2 predstavlja torej poizvedbo z enim numeričnim parametrom.
- P3 - Izpis transakcij tekočega računa *trrA*, kar smo izvedli s poizvedbo na sliki 8. P3 predstavlja poizvedbo z enim tekstovnim parametrom, pri čemer je nabor možnih vrednosti omejen in ga je možno prebrati iz podatkovne baze z uporabo podpoizvedbe na sliki 4.

Merjenje P1 smo pričeli ob prvem kliku na meni v spletni aplikaciji, kjer je vsak uporabnik ročno vnesel poizvedbo Cypher ali je izbral vnaprej pripravljeno predlogo

```
1 MATCH (a)-[t]->(b)
2 RETURN sum(t.znesek) AS Vsota
```

Slika 6: Poizvedba P1 v poizvedovalnem jeziku Cypher za izpis vsote bančnih transakcij.

```
1 MATCH (a:Racun)-[t]->(b:Racun)
2 WHERE t.znesek > $znesek
3 RETURN a, t, b
```

Slika 7: Poizvedba P2 v poizvedovalnem jeziku Cypher za izpis bančnih transakcij, katerih znesek je višji od parametra *znesek*.

poizvedbe. Pri prvem načinu je uporabnik ročno prepisal poizvedbo iz slike 6. Meritev smo zaključili ob kliku na zagon poizvedbe. Merjenje priprave poizvedbe smo pri vsakem uporabniku izvedli desetkrat. Na podoben način smo merili tudi poizvedbi P2 in P3.

Glede na to, da smo pri P1 že podrobno merili ročno vnašanje poizvedbe iz slike, smo pri P2 uporabniku dovolili, da poizvedbo P2 prilepi iz odložišča. Uporabnik je moral pri vsakem zagonu namesto parametra *$znesek* ročno napisati številko zagona. Prav tako je moral pri uporabi predlog poizvedb kot edini parameter podati številko zagona.

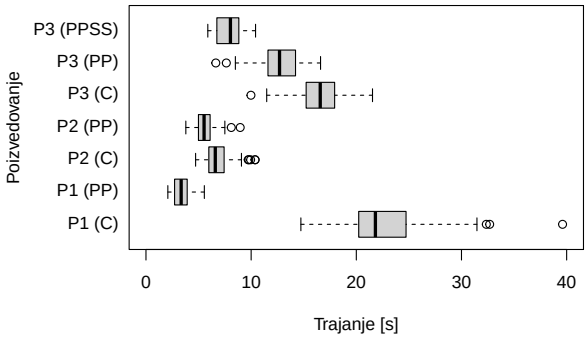
Tudi pri P3 smo merili čas vnosa poizvedbe iz odložišča in potem ročni vnos številke obstoječega transakcijskega račun iz seznama glede na zaporedno številko merjenja. Pred vsakim zagonom smo torej ročno spremenili poizvedbo Cypher tako, da smo vnesli vnaprej izbrano številko transakcijskega računa. Pri uporabi predlog poizvedb je uporabnik v vnosno polje na enak način ročno vnesel številko transakcijskega računa.

Način poizvedovanja pri P3 smo nadgradili z uporabo spustnega seznama vsem možnih številk transakcijskih računov. Pri tem je uporabnik z vnosom zgolj začetnih štirih znakov in nato z izbiro vnosa iz spustnega menija skrajšal čas vnašanja. Med vnosom se je namreč zožil nabor ponujenih številk transakcijskih računov na enega ali dva.

Rezultati meritev poizvedovanj v obliki grafikona kvantilov so prikazani na sliki 9. Za osamelce sklepamo, da predstavljajo napake pri vnašanju podatkov. Glede na razliko med P1 (C) in P1 (PP) sklepamo, da je ročni vnos poizvedb izrazito počasnejši, zato so predloge poizvedb nujne. Glede na razliko med P1 (C) in P2 (C) sklepamo, da uporaba odložišča pohitri poizvedovanje že pri tako kratkih poizvedbah, kot sta P1 in P2. Razlika med P2 (C) in P2 (PP) nakazuje, da je vnašanje številk v vnosno polje učinkovitejše od neposrednega spreminjanja poizvedbe Cypher. Glede na razliko med P2 (C) in P3 (C) ter med

```
1 MATCH (n1:Racun {stevilka:
  $trrA})-[t]->(n2:Racun)
2 RETURN n1, t, n2
```

Slika 8: Poizvedba P3 v poizvedovalnem jeziku Cypher za izpis vseh bančnih transakcij iz transakcijskega računa *trrA*, podanega kot parameter.



Slika 9: Grafikon kvantilov in osamelcev trajanja priprave poizvedb P1, P2 in P3 glede na ročni vnos (C), uporabo predloge poizvedb (PP) in uporabo predloge poizvedb s spustnim seznamom (PPSS).

P2 (PP) in P3 (PP) opazimo, da ročno vnašanje številk računov vzame bistveno več časa kot vnos manjših vrednosti. Meritve pri P3 (PPSS) dokazujejo, da uporaba spustnega seznama bistveno pohitri preiskovanje, zato ga je smiselno uporabiti pri vnosu daljših nizov. Uporaba predlog poizvedb pa ne prinaša samo prihranka v času priprave poizvedb, ampak omogoča preiskovalcem, da s pripravljenimi poizvedbami poiščejo koristne informacije v podatkih brez poznavanja jezika Cypher in teorije grafov.

Predloge poizvedb omogočajo bistveno hitrejše poizvedovanje tudi od nekaterih alternativnih načinov poizvedovanja. Tvorba novih poizvedb na podlagi primerov uporabe [7] lahko od uporabnika zahteva več minut za vnašanje primerov. Grafično poizvedovanje [6] bistveno olajša poizvedovanje, vendar od uporabnika lahko zahteva več 10 sekund.

4 Zaključek

V prispevku smo predstavili predloge poizvedb na primeru preiskovalne platforme. Predloge poizvedb lahko za večino scenarijev uporabe nadomestijo neposredni vnos poizvedb. Uporaba predlog poizvedb omogoča hitrejše preiskovanje, saj ni potrebnega ročnega vnašanja poizvedb. Poleg tega predlagan spustni seznam znotraj predlog poizvedb skrajša vnašanje parametrov. Najbolj pomemben prispevek teh predlog je, da lahko platformo STALITA uporabljajo tudi preiskovalci, ki niso vešči pisanja poizvedb v programskem jeziku Cypher.

V prihodnjem delu bi želeli najprej nadgraditi predloge poizvedb z novimi funkcionalnostmi. Najprej bi dodali podporo za veriženje predlog poizvedb, kar bi omogočalo uporabniško prijazno kompleksnejše poizvedovanje. Nato pa bi dodali podporo za preverjanje pravilnosti vnesenih parametrov poizvedb, npr. da so parametri poizvedb samo znotraj dovoljenih vrednosti. V prihodnosti želimo prav tako preveriti uporabnost predlog poizvedb na širšem krogu uporabnikov z različnim predznanjem na več scenarijih uporabe. Pri tem bi želeli vključiti tudi druge podatke poleg bančnih transakcij. Prav tako vidimo priložnost, da te predloge nadgradimo z modulom, ki bo uporabniku pomagal poiskati najbolj primerno poizvedbo glede na njegove potrebe.

5 Zahvala

Raziskovalno delo v danem prispevku je sofinancirala Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije, ter Ministrstvo za notranje zadeve, v sklopu ciljnega raziskovalnega programa V2-2260 - Integracija in analiza heterogenih podatkovnih tokov v preiskovalni platformi. Raziskovalni program št. P2-0041 je sofinancirala Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije.

Literatura

[1] N. M. Karie in H. S. Venter. Taxonomy of challenges for digital forensics. *Journal of forensic sciences*, 60(4):885–893, 2015.

[2] S. Satpathy in S. N. Mohanty. *Big data analytics and computing for digital forensic investigations*. CRC Press, 2020.

[3] D. Jesenko, Š. Kohek, B. Žalik, M. Brumen, D. Kavran, N. Lukač, A. Živec in A. Pur. Stalita: Innovative platform for bank transactions analysis. *Applied Sciences*, 12(23):12492, 2022.

[4] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer in A. Taylor. Cypher: An evolving query language for property graphs. Objavljeno v *Proceedings of the 2018 international conference on management of data*, strani 1433–1445, 2018.

[5] S. Polyviou, G. Samaras in P. Evripidou. A relationally complete visual query language for heterogeneous data sources and pervasive querying. Objavljeno v *21st International Conference on Data Engineering (ICDE'05)*, strani 471–482. IEEE, 2005.

[6] G. Draper in R. Riesenfeld. Who votes for what? a visual query language for opinion data. *IEEE transactions on visualization and computer graphics*, 14(6):1197–1204, 2008.

[7] S. Zhang in Y. Sun. Automatically synthesizing sql queries from input-output examples. Objavljeno v *28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, strani 224–234. IEEE, 2013.

[8] G. Katsogiannis-Meimarakis in G. Koutrika. A survey on deep learning approaches for text-to-sql. *The VLDB Journal*, strani 1–32, 2023.

[9] K. Dhamdhere, K. S. McCurley, R. Nahmias, M. Sundararajan in Q. Yan. Analyza: Exploring data with conversation. Objavljeno v *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, strani 493–504, 2017.

[10] R. Omar, O. Mangukiya, P. Kalnis in E. Mansour. Chatgpt versus traditional question answering for knowledge graphs: Current status and future directions towards knowledge graph chatbots. *arXiv preprint arXiv:2302.06466*, 2023.

[11] A. Granda, N. Lukač, A. Pur in Š. Kohek. Efficient machine learning based graph layout calculation for investigation platform. In F. Solina, editor, *Proceedings of the 8th Student Computing Research Symposium (SCORES'22)*, strani 5–8. UL Fakulteta za računalništvo in informatiko, Ljubljana, 2022.