Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Grafové databáze

Laura Beláková

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Laura Beláková

Informatika

Název práce

Grafové databáze

Název anglicky

Graph databases

Cíle práce

Cílem práce je navrhnout, vytvořit a otestovat malou databázovou aplikaci pro část studijní evidence (předměty, studenti, semestrální projekty, ...) v prostředí grafové databáze a také relační databáze.

Metodika

První část práce popíše použité teoretické nástroje. Druhá část práce bude vlastní projekt v prostředí Neo4j a MySQL. Pro dokumentaci bude použit standard UML. Obě aplikace (grafová i relační) budou obsahovat stejná data a stejné dotazy. V závěru práce budou zhodnoceny rozdíly, výhody a nevýhody obou použitých technologií.

Doporučený rozsah práce

40-60 stran

Klíčová slova

grafové databáze; Neo4j; relační databáze; MySQL

Doporučené zdroje informací

Meier, A. a Kaufmann, M., 2019. SQL & NoSQL databases: models, languages, consistency options and architectures for Big data management. Wiesbaden: Springer. ISBN 978-3-658-24548-1.

Merunka, V., 2002. Objektový přístup v databázových systémech. Praha: Credit. ISBN 80-213-0882-6.

Robinson, I., Webber, J. a Eifrem, E., 2015. Graph databases. Second edition. Sebastopol: O'Reilly Media. ISBN 978-1-491-93089-2.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

doc. Ing. Vojtěch Merunka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 31. 10. 2022

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 24. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 19. 02. 2023

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Grafové databáze" jsem vypracovala samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autorka uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušila autorská práva třetích osob.

V Praze dne 11.03.2023 _____

Poděkování

Ráda bych touto cestou poděkovala svému vedoucímu bakalářské práce doc. Ing. Vojtěchovi Merunkovi, Ph.D. za cenné rady, věcné připomínky a čas, který mi v průběhu zpracování práce věnoval.

Grafové databáze

Abstrakt

Bakalárska práca sa zaoberá návrhom malej databázovej aplikácie vytvorenej pre časť študijnej evidencie a jej implementáciou za účelom porovnania dvoch databázových prostredí, grafovej a relačnej databázy.

V teoretickej časti sú popísané techniky a nástroje databázovej technológie, ktoré sú použité v praktickej časti. Ponúka stručné predstavenie oboch databáz a ich porovnanie z viacerých hľadísk s cieľom ozrejmiť rozdiely medzi nimi. Porovnanie bolo zamerané na výkonnosť, objem potrebnej programátorskej práce a s tým súvisiacu náročnosť vývoja, a dĺžku zápisu dotazu. Predstavený bol jazyk Cypher Query Language.

Praktická časť bola zameraná na návrh samotnej aplikácie a dokumentáciu databázových modelov pomocou štandardu UML. V aplikácií boli evidované údaje o študentoch, ich semestrálnych projektoch, učiteľoch a vyučovaných predmetoch. V praktickej časti bola využívaná relačná databáza MySQL a grafová databáza Neo4j. Grafová aj relačná databáza obsahovali rovnaké dáta a vykonávané boli dotazy, ktoré vracali rovnaký výsledok. Zhodnotené boli rozdiely, výhody a nevýhody oboch použitých technológií. Porovnanie bolo doplnené o tabuľky, obrázky a časti kódu.

Kľúčové slová: grafové databáze; Neo4j; relační databáze; MySQL

Graph databases

Abstract

The bachelor's thesis examines and compares the implementation of graphing database technology against relational database technology for use within a small database application that processes student registration.

The theoretical portion of the thesis describes database technology tools and techniques which are later used in the practical portion. It provides a brief introduction to both database types and offers a comparison from several points of view in order to illustrate their differences. This comparison focuses on performance, the amount of programming work required, and the related difficulty and length of each query. Additionally, it introduces Cypher Query Language, which is used by Neo4j graph database.

The practical portion of the thesis focuses on the design of the application itself and the documentation of the database models using the UML standard. The application contains data about students, student's semester projects, teachers, and subjects the teachers taught. MySQL relational database and Neo4j graph database are applied in the practical portion of the thesis.

Both the graph and the relational databases contained the same data and executed the same queries. Differences, advantages, and disadvantages of each technology used were then evaluated. Tables, pictures, and code sections are provided to aid with the comparison.

Keywords: graph databases; Neo4j; relational databases; MySQL

Obsah

1 Úvod

Táto bakalárska práca sa zaoberá návrhom malej databázovej aplikácie vytvorenej pre časť študijnej evidencie a jej implementáciou za účelom porovnania dvoch databázových prostredí, grafovej a relačnej databázy. Grafové databázy ponúkajú možnosť prehľadného zobrazenia situácií, v ktorých vzniká veľké množstvo väzieb medzi dátami.

Počas môjho štúdia ma zaujal predmet, na ktorom sme sa učili o relačnom spracovaní dát s využitím dotazovacieho jazyka SQL. Pri výbere témy bakalárskej práce ma lákala predstava osobnej výzvy v porozumení aj inému nerelačnému databázovému prostrediu, preto som zvolila práve grafové databázy. Zároveň som si však chcela prehĺbiť znalosti z oblasti relačných databáz o tvorbu zložitejších dotazov a prácu s väčším počtom tabuliek.

Grafové databázy neustále naberajú na popularite, o čom svedčí aj práve prebiehajúca štandardizácia nového dotazovacieho jazyka Graph Query Language (GQL). Návrh na jeho vytvorenie bol schválený členmi ISO/IEC JTC 1 s cieľom spájať to najlepšie z jazykov PGQL, G-CORE a Cypher. (One Property Graph Query Language, 2022)

2. Cieľ práce a metodika

2.1 Cieľ práce

Cieľom práce je navrhnúť, vytvoriť a otestovať malú databázovú aplikáciu pre časť študijnej evidencie v prostredí grafovej a relačnej databáze. V aplikácií budú evidované údaje o študentoch, ich semestrálnych projektoch, učiteľoch a vyučovaných predmetoch.

2.2 Metodika

Prvá časť popíše použité teoretické nástroje. Ponúkne stručné predstavenie relačných a grafových databáz. Bližšie priblíži jazyk Cypher Query Language. Druhá časť bude vlastný projekt v prostredí Neo4j a MySQL. Využívanými jazykmi budú SQL a Cypher Query Language. Doplnená bude o dátové modely oboch databázových systémov. Pre dokumentáciu bude použitý štandard UML. Obe aplikácie (grafová aj relačná) budú obsahovať rovnaké dáta a vykonávané budú dotazy vracajúce rovnaký výsledok. Bude obsahovať tabuľky, obrázky a časti kódu, ktoré budú slúžiť pre lepšiu názornosť pri záverečnom hodnotení. Na záver budú zhodnotené rozdiely, výhody a nevýhody oboch použitých technológií z viacerých hľadísk.

3. Teoretické východiská

Teoretická časť ponúka krátky úvod do databázových systémov, popisuje vývoj databáz z historického hľadiska, objasňuje rozdiely medzi grafovou a relačnou databázou. Rovnako približuje prácu s grafovou databázou v prostredí Neo4j.

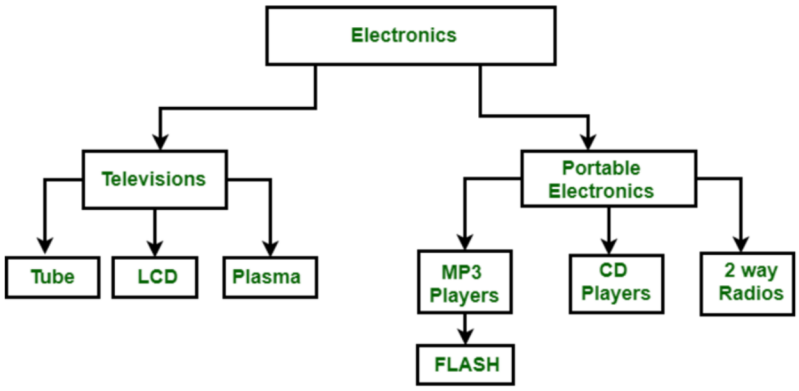
3.1 Úvod do databázových systémov

Poznanie histórie databázových systémov vedie k lepšiemu pochopeniu ich fungovania a základných vlastností. Potreba zaznamenávať a usporadúvať dáta podľa rôznych kritérií tu bola už od nepamäti. Predchodcom dnešných databáz boli papierové kartotéky. Medzi vôbec prvé strojové spracovanie dát patrí sčítanie ľudu z roku 1890 v USA. Vtedy využívaným pamäťovým médiom pre zaznamenanie dát bol dierny štítok a následné spracovanie vykonávali elektromechanické stroje. (Wikipédia, 2022a)

Nástup počítačov podporil potrebu efektívnejšieho spracovania dát. Následný vývoj môžeme rozdeliť podľa dátového modelu do troch etáp – navigačná, relačná a post-relačná.

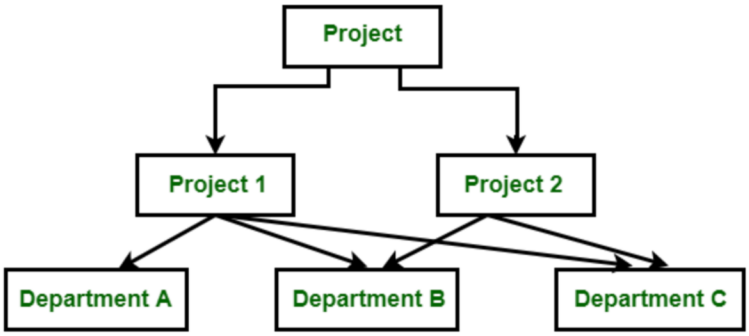
3.1.1 Navigačné modely

Do navigačných modelov radíme hierarchický model a sieťový model. V hierarchickom dátovom modeli sú dáta usporiadané v podobe stromovej štruktúry s jedným koreňom. Tento koncept bol predstavený v 60. rokoch 20. storočia a najznámejším hierarchickým databázovým systémom bol IMS od spoločnosti IBM. Implementoval vzťahy 1:1 a 1:N, za jeho pomoci však bolo problémové modelovať vzťahy kardinality M:N. Nasledujúci obrázok znázorňuje jeho štruktúru. (GeeksforGeeks, 2022a)



Obrázok 1 - Hierarchický model podľa (GeeksforGeeks, 2022a)

Pomocou sieťového modelu sa na rozdiel od toho hierarchického dali modelovať aj vzťahy kardinality M:N. Ďalším rozdielom bolo usporiadanie dát, v tomto prípade grafovou štruktúrou pomocou orientovaného grafu. Táto štruktúra je zobrazená na obrázku č.2. (GeeksforGeeks, 2022a)



Obrázok 2 - Sieťový model podľa (GeeksforGeeks, 2022a)

3.1.2 **Relačné databázové modely**

Za nástupcu navigačných modelov sú považované relačné databázové modely, ktoré na usporadúvanie dát využívajú jediný konštrukt – relácie, čiže tabuľky. S novým návrhom dátového modelu prišiel v 70. rokoch 20. storočia Edgar F. Codd. Tento spôsob organizácie záznamov sa od svojich predchodcov líšil formou reprezentácie dát a prepájaním vzťahov medzi nimi pomocou cudzích kľúčov. Koncept sa veľmi rýchlo ujal a z trhu vytlačil dovtedy využívané systémy. (GeeksforGeeks, 2022a)

3.1.3 **Post-relačné modely**

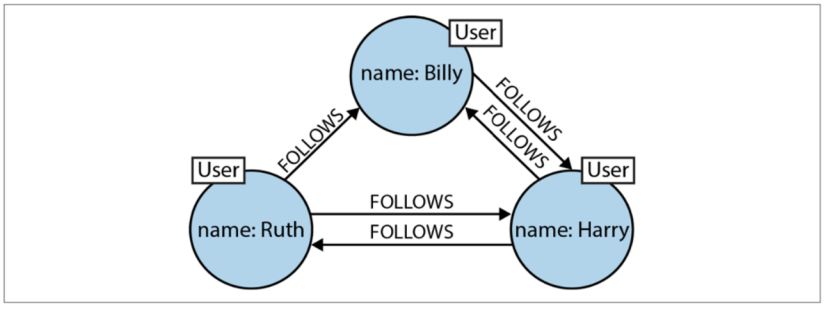
Medzi post-relačné modely zaraďujeme napríklad grafové databázy. Grafové databázy rovnako ako sieťový model využívajú grafovú štruktúru pri zobrazení údajov a ich vzájomných vzťahov. Mohli by sme teda v tomto prípade hovoriť o takzvanom znovuzrodení sieťového grafového modelu.

3.2. **Graf**

Graf môžeme definovať ako množinu uzlov a hrán. Uzly predstavujú jednotlivé entity a hrany väzby, a vzťahy medzi nimi. Pomocou grafov vieme reprezentovať veľkú škálu situácií a vďaka nim sa v týchto situáciách dokážeme lepšie orientovať.

V bežnom živote grafy veľakrát využívame bez toho, aby sme si to vôbec uvedomovali. Dobrým príkladom pre čitateľa môže byť zasadací poriadok na svadbe. Typicky si pri plánovaní na papier nakreslíme jednotlivé osoby, ktoré úplne prirodzene reprezentujeme ich menom v kruhu. Následne si zobrazujeme jednotlivé vzťahy medzi týmito svadobčanmi pomocou pomenovaných hrán.

Ďalším príkladom môže byť graf, ktorý zobrazuje používateľov jednej sociálnej siete a ich vzájomné prepojenia, uvedený v knihe Graph Databases. (Eifrem, Robinson a Weber, 2015)



Obrázok 3 - Graf s užívateľmi sociálnej siete podľa (Eifrem, Robinson a Weber, 2015)

Na tomto jednoduchom grafe každý uzol s označením (label) „User“ predstavuje jedného užívateľa sociálnej siete. Označenie sa používa pre upresnenie role danej entity v danej komunikačnej sieti. Tieto uzly sú poprepájané pomocou pomenovaných

orientovaných hrán, ktoré ďalej rozvíjajú vzťahy medzi jednotlivými entitami. Napríklad Ruth sleduje Billyho, avšak tento vzťah nie je zatiaľ opätovaný, pretože Billy nesleduje Ruth. Vzťahy sú pomenované, majú smer (Ruth sleduje Billyho a nie opačne) a vždy začínajú, a končia v nejakom uzle.

Na tomto príklade je demonštrovaná iba veľmi malá časť vzťahov danej sociálnej siete, avšak ak by sme chceli namodelovať reálne zobrazenie, graf by bol založený na presne rovnakom princípe.

Veľkou výhodou tohto zobrazenia je, že ľuďom príde častokrát intuitívne, prehľadné a ľahké na čítanie. Aj práve preto sa v dnešnej dobe dostávajú grafové databázy do popredia. (Eifrem, Robinson a Weber, 2015)

3.3. Porovnanie relačných a grafových databáz

V nasledujúcich riadkoch sa budem podrobnejšie venovať porovnaniu relačných a grafových databáz s cieľom vysvetliť rozdiely medzi nimi. Rovnako v tejto časti viac priblížim jazyk využívaný databázou Neo4j, Cypher Query Language.

3.3.1. Relačné databázy

Základným zobrazením dát je v relačných databázach relácia, čiže zjednodušene povedané dvojrozmerná tabuľka. Je tvorená riadkami, ktoré zobrazujú záznamy o určitej entite a stĺpcami, ktoré sú tiež nazývané atribútmi tabuľky. Tabuľka môže pre lepšie upresnenie a odlíšenie záznamov obsahovať primárne kľúče. Väzby medzi reláciami sú vyjadrené za využitia cudzích kľúčov pomocou odkazovania na atribút s primárnym kľúčom. Štandardizovaným štruktúrovaným dotazovacím jazykom relačných databáz je SQL. (Vostrovský, 2014). V praktickej časti tejto práce bude pre spracovávanie dát využívaná databáza MySQL.

3.3.2 Grafové databázy

Ako už bolo spomínané vyššie, grafové databázy zobrazujú dáta schematicky v grafoch na rozdiel od tabuliek.

Najpoužívanejšou grafovou databázou je open-source Neo4j. Rovnako ako MySQL, je kompatibilná s veľkým množstvom programovacích jazykov. Oficiálne podporuje ovládače pre .Net, Java, JavaScript, Go a Python. (Neo4j: Graph Data Platform, 2022a)

Okrem Neo4j, ktorá vznikla v roku 2007, sa o desať rokov neskôr na trhu objavila aj open-source grafová databáza Memgraph, ktorá je s Neo4j kompatibilná. Takisto podporuje prácu s rôznymi programovacími jazykmi a v komunitnej edícií je zadarmo. (Memgraph, 2023a)

V porovnaní s relačnými databázami, ktorých štandardizovaným dotazovacím jazykom je SQL, sa v prípade grafových na obdobnom jazyku ešte len pracuje. Momentálne má používateľ na výber napríklad z Cypher Query Language, Gremlin, PGQL a iných. Od konca roku 2019 prebieha štandardizácia nového dotazovacieho jazyka Graph Query Language (GQL), ktorý má spájať to najlepšie z PGQL, G-CORE a Cypher. Návrh na jeho vytvorenie bol schválený členmi ISO/IEC JTC 1. Tento jazyk by mohol v grafových databázach v budúcnosti hrať rovnakú kľúčovú rolu ako SQL v relačnom prostredí. (One Property Graph Query Language, 2022)

3.3.3. **Cypher Query Language**

Cypher Query Language bol navrhnutý špeciálne pre Neo4j so zámerom byť prehľadným a ľahko čitateľným jazykom pre dátových analytikov, dátových inžinierov, vývojárov alebo pre bussiness-orientovaných ľudí. (Sasaki, 2018)

Veľa kľúčových slov ako napríklad WHERE, ORDER BY alebo AND bolo inšpirovaných jazykom SQL. Špeciálne kľúčové slová MATCH a RETURN boli prevzaté z jazyka SPARQL. (Neo Graph Data Platform, 2022b). Príkaz MATCH slúži k získaniu dát z databáze podľa určitej vlastnosti alebo vzoru. RETURN upresňuje uzol, vzťah alebo vlastnosť, ktorú chceme vrátiť. Cypher používateľovi umožňuje požiadať databázu, aby našla hodnoty, ktoré zodpovedajú špecifickému zadanému vzoru, a teda ho „match-ujú“. (Eifrem, Robinson a Weber, 2015)

MATCH/RETURN a SELECT

Ak by sme napríklad chceli zobraziť všetky záznamy o filme, ktorý má názov „Duna“, a teda spĺňa túto vlastnosť, v Neo4j by sme príkaz napísali nasledovne:

```
MATCH (D:Filmy { name: "Duna" } )
RETURN D
```

Môžeme si všimnúť, že príkaz napísaný v Cypher vráti uzol „Filmy“, kde vlastnosť „name“ je „Duna“. „D“ slúži ako premenná pre uchovávanie údajov. Hlavnými kľúčovými slovami sú teda MATCH a RETURN.

Oproti tomu v SQL je hlavným kľúčovým slovom pre výber a zobrazenie SELECT a príkaz by vyzeral takto:

```
SELECT * FROM Filmy WHERE name = 'Duna';
```

V Neo4j sú uzly v grafoch reprezentované pomocou symbolu „ ( ) “, vzťahy „ -> “a typ vzťahu medzi uzlami ako „ [ ] “.

Pre detailnejší popis tohto jazyka odporúčam navštíviť oficiálnu dokumentáciu na adrese https://neo4j.com/docs/.

Viac rozdielov a podobností v zápise príkazov pomocou SQL a Cypher Query Language bude demonštrovaných v praktickej časti tejto práce na konkrétnych príkladoch.

3.3.4 **PGQL**

Ďalším z jazykov, ktorý je možné využívať v grafových databázach je Property Graph Query Language. Tento dotazovací jazyk je postavený na SQL, no zároveň v sebe spája prvky pripomínajúce syntax Cypher Query Language. Dotazy spolu s SQL konštruktmi ako napríklad SELECT, FROM alebo WHERE obsahujú aj kľúčové slovo MATCH, ktoré je typické pre syntax Cypher. (Property Graph Query Language, 2022)

Zobrazovanie a zaznamenávanie dát pomocou relácií neustále dominuje na profesionálnom trhu, no postupne sa čoraz viac do povedomia dostáva reprezentácia pomocou grafov. Využitie grafových databáz získava na popularite najmä tam, kde vzniká veľké množstvo väzieb medzi dátami. (Eifrem, Robinson a Weber, 2015)

3.4 Sila grafových databáz

3.4.1 Výkonnosť

Na rozdiel od relačnej databázy, kde klesá výkonnosť a predlžuje sa čas odpovede na dotaz s rastúcim počtom množiny údajov, výkonnosť a časová odpoveď zostáva v prípade grafovej databázy relatívne konštantná aj so zvyšujúcim sa počtom uzlov a väzieb. Dôvodom je odlišná časť databáze, ktorá je pri dotaze prehľadávaná. Pri relačných databázach sa prehľadáva celá množina dát, oproti tomu v grafových je to len časť množiny, ktorá je prepojená so súvisiacimi uzlami alebo hranami. Časová odpoveď na dotaz je tak priamo úmerná len takej veľkosti grafu, ktorú je nutné prejsť na jeho úspešné vykonanie, a nie celej veľkosti grafu. (Eifrem, Robinson a Weber, 2015)

Jonas Partner a Aleksa Vukotic v knihe Neo4j In Action (Partner a Vukotic, 2014) realizovali zaujímavý pokus. Na príklade sociálnej siete demonštrovali rýchlosť odpovede na dotaz za použitia grafovej a relačnej databázy. Najskôr začali s dotazom vyhľadať všetkých priateľov priateľov používateľa. Čiže napríklad, ak som používateľom sociálnej siete ja, v druhom stupni dotazu hľadám mojich priateľov a všetkých ich priateľov. S ďalšími opakovaniami vždy zvyšovali úroveň náročnosti dotazu o stupeň, čiže pri treťom stupni sa dotaz zmenil na nájdenie všetkých priateľov priateľov priateľov používateľa a tak ďalej. Obdobne ako bolo vysvetlené vyššie, ak som používateľom ja, v treťom stupni dotazu hľadám mojich priateľov, všetkých priateľov mojich priateľov a všetkých priateľov ich priateľov. Aleksa a Jonas vytvorili obe databázy o celkovom počte 1 000 000 užívateľov. (Partner a Vukotic, 2014)

Na nasledujúcom obrázku je zobrazená hĺbka dotazovaných vzťahov (stupne) a čas, ktorý trvalo vykonať dotaz v MySQL a Neo4j. Rýchlosť odpovede na dotaz bola uvedená v sekundách a každý používateľ mal približne 50 priateľov. (Neo4j: Graph Data Platform, 2022c)

Depth	Execution Time – MySQL	Execution Time –Neo4j
2	0.016	0.010
3	30.267	0.168
4	1,543.505	1.359
5	Not Finished in 1 Hour	2.132

Obrázok 4 - Porovnanie rýchlosti odpovede na dotaz v MySQL a Neo4j podľa (Neo4j: Graph Data Platform, 2022c)

Na základe výsledkov môžeme vidieť, že čas odpovede na druhý stupeň hĺbky dotazu, a teda nájdenie priateľov priateľov používateľa, bol v grafovej databáze o trochu rýchlejší ako v relačnej. Koncový používateľ by si tento rozdiel v milisekundách sotva všimol a preto môžeme konštatovať, že si obe databázy v druhom stupni viedli dostatočne dobre na to, aby mohli byť využité v online prostredí. Avšak, v treťom stupni sa rýchlosť za použitia MySQL výrazne znížila, a to až o 180-krát. Pri dĺžke trvania odpovede 30 sekúnd je prakticky nemožné využiť relačnú databázu v online systéme. Na rozdiel od MySQL, rýchlosť odpovede pri Neo4j zostáva relatívne rovnaká v porovnaní s druhým stupňom. Pri štvrtom stupni je Neo4j takmer 1 135 - krát rýchlejšia, MySQL potrebuje na odpoveď približne 26 minút. V piatom stupni sa v Neo4j čas odpovede zhorší na 2 sekundy, čo je stále dostatočne validný výsledok pre použitie v online systéme. Podľa tabuľky môžeme vidieť, že relačná databáza úlohu nezvládne dokončiť ani za hodinu. (Eifrem, Robinson a Weber, 2015)

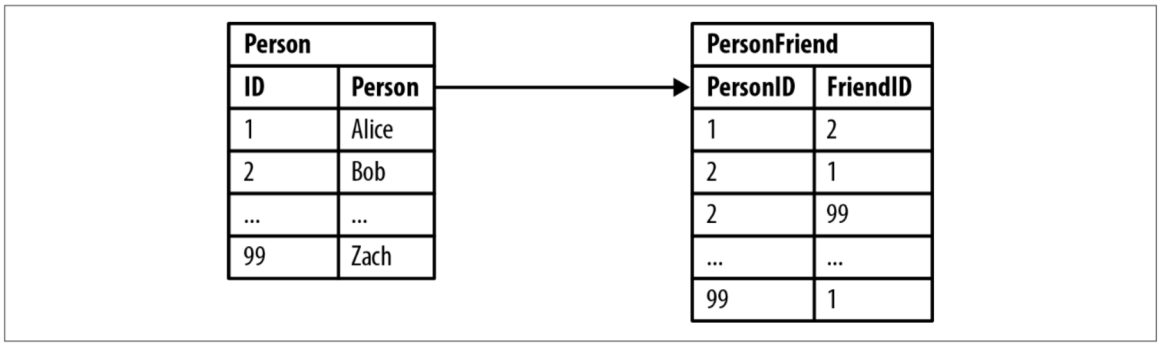
Práve na tomto príklade sa dá poukázať na neuveriteľný výkon grafových databáz v situáciách, v ktorých sa očakáva veľké množstvo väzieb medzi dátami.

3.4.2 **Náročnosť a dĺžka zápisu dotazu**

V predošlej časti sme sa na konkrétnom príklade venovali porovnaniu výkonov grafových a relačných databáz. Okrem výkonu je však na tom istom modeli možné demonštrovať aj náročnosť zápisu v oboch databázach.

V relačných databázach môže byť niekedy spájanie tabuliek pomocou cudzích kľúčov a join-ov pomerne náročný, neprehľadný a zdĺhavý proces.

Nasledujúci obrázok zobrazuje vzťah medzi dvomi tabuľkami, ktoré zhromažďujú záznamy o používateľovi (tabuľka „Person“) a priateľoch používateľa (tabuľka „PersonFriend“). (Eifrem, Robinson a Weber, 2015)



Obrázok 5 - Zobrazenie vzťahu medzi používateľom a jeho priateľmi v relačnom modeli podľa (Eifrem, Robinson a Weber, 2015)

Ak by sme zostali iba pri dotaze hĺbky prvého stupňa a teda len nájdení používateľových priateľov, v tomto prípade priateľov Boba, dotaz v Cypher Query Language by sme zapísali takto:

```
MATCH (b {name:"Bob"})-[:FRIEND]->(bobsFriends)
RETURN b, bobsFriends
```

V porovnaní s Cypher, dotaz v SQL by sme zapísali nasledovne:

```
SELECT p1.Person
FROM Person p1 JOIN PersonFriend
ON PersonFriend.FriendID = p1.ID
JOIN Person p2
ON PersonFriend.PersonID = p2.ID
WHERE p2.Person = 'Bob '
```

Aj na takomto jednoduchom dotaze si čitateľ môže všimnúť, že zápis v Neo4j je oveľa kratší a prehľadnejší. S postupne sa zvyšujúcou komplexitou dotazu sa aj samotný zápis v MySQL stáva oveľa viac komplexným.

V tomto prípade samozrejme záleží na preferencií a subjektívnom názore používateľa a platí, že práca s join príkazmi môže byť aj napriek ich dĺžke niektorým z nich oveľa pohodlnejšia.

3.4.3 **Flexibilita a agilita**

V posledných rokoch sa čoraz viac do popredia dostávajú agilné metodiky. Umožňujú rýchly vývoj softwaru a zároveň pohotovú reakciu na zmenu požiadaviek zákazníka už v priebehu vývojového cyklu. (Wikipédia, 2022b) Grafové databázy umožňujú pridávať nové uzly, vzťahy či označenia (labels) do už existujúcej štruktúry bez jej narušenia. Vďaka tejto flexibilite nemusí byť dopredu pevne stanovená schéma, čo podporuje agilný prístup. Tieto vlastnosti tak presne zapadajú do dnes nastaveného trendu dopytu. V relačnej databáze je nutné pred pridávaním záznamov najskôr definovať schému, následné zásahy či jej pozmenenie môže byť problematické. (Eifrem, Robinson a Weber, 2015)

3.4.4 **Nedostatky**

Aj napriek vyššie zmienenému dáva veľa používateľov stále prednosť relačným databázam (DB-Engines, 2023). Jedným z dôvodov môže byť aj fakt, že užívateľ je zvyknutý na prácu s dobre zavedenou a ľahko pochopiteľnou dátovou platformou a nevidí dôvod na zmenu, a na učenie sa novému systému. Môžeme predpokladať,

že prednosť grafovým databázam dajú skôr nováčikovia, ktorí sa ešte s prostredím relačných databáz nestretli. Z vlastnej skúsenosti si však dovolím tvrdiť, že pre ľudí, ktorí využívajú relačné databázy a jazyk SQL by mal byť prechod na tie grafové pomerne pohodlný.

Ďalšou nevýhodou oproti relačným databázam je menšia základňa používateľov. Ak narazíte na problém, môže byť ťažšie nájsť podporu. Tento problém by sa mal časom s rastúcim počtom užívateľov odstrániť. Zároveň však webová stránka Neo4j na adrese https://neo4j.com/ ponúka pestrú škálu dokumentov a návodov, z ktorých je možné pri riešení čerpať.

4 Vlastná práca

Praktická časť tejto práce sa zameriava návrhom malej databázovej aplikácie určenej pre časť školskej evidencie v prostredí grafovej a relačnej databázy. Použitými nástrojmi sú MySQL a Neo4j, a využívanými jazykmi SQL, a Cypher Query Language. Doplnená je o dátové modely oboch databázových systémov.

4.1 Popis školskej evidencie

Na škole evidujeme údaje o študentoch, ich semestrálnych projektoch, učiteľoch a vyučovaných predmetoch.

O každom študentovi vieme, ako sa volá, v akom je ročníku, a kedy sa narodil. Dokážeme zistiť, na akom semestrálnom projekte pracuje. Študent študuje niekoľko predmetov a môže pracovať na viacerých semestrálnych projektoch. Môže byť zapísaný do predmetu, ale ešte nebyť zapísaný k projektu.

Semestrálny projekt má svoj názov. Vieme zistiť k akému vyučovanému predmetu patrí. Projekt musí patriť vždy iba k jednému predmetu, v rámci predmetu môže byť pridelených viacero projektov.

Na konkrétnom semestrálnom projekte môžu pracovať viacerí študenti naraz. Semestrálny projekt nemusí mať prideleného žiadneho študenta (napríklad, ak si danú tému nikto nevybral).

Predmet je vypísaný pre určitý počet študentov a má svoj názov. Každý predmet je vyučovaný iba v určitom ročníku a semestri. Predmet môže vyučovať viacero učiteľov.

Učiteľ môže učiť viacero predmetov a môže byť vedúcim viacerých projektov. Oproti tomu, projekt má vždy práve jedného vedúceho. O učiteľovi vieme, ako sa volá, a kedy sa narodil.

4.2 Dátové modely

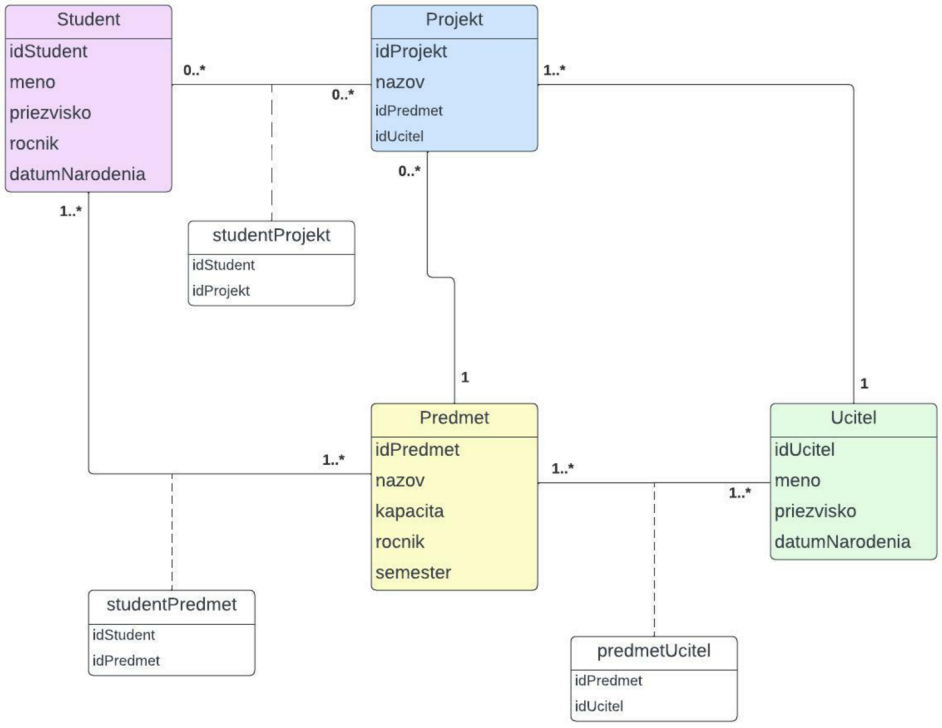
Cieľom dátového modelovania je navrhnúť kvalitnú dátovú štruktúru pre konkrétnu aplikáciu. Keďže aplikácia bude vytvorená v prostredí relačnej aj grafovej databázy, úvod tejto časti obsahuje oba dátové modely.

4.2.1 Relačný dátový model

Model je tvorený sadou entít, ktoré sú medzi sebou poprepájané pomocou väzieb s príslušnou kardinalitou. Každá entita je popísaná svojim názvom, obsahuje množinu atribútov a identifikátor. Našou entitou je napríklad „Student“ a jej inštanciami sú konkrétni študenti, ktorých na škole evidujeme. Kardinalita popisuje obmedzenie v počte inštancií druhej entity, ktoré môžu mať vzťah s akoukoľvek inštanciou entity prvej a naopak. Pri vzťahu M:N, keď jedna inštancia má vzťah s viac inštanciami inej entity a opačne, sa využíva asociačná trieda, ktorá obsahuje dva cudzie kľúče.

Náš relačný dátový model obsahuje tri asociačné triedy („studentProjekt“, „studentPredmet“ a „predmetUcitel“). V modeli môžeme taktiež vidieť väzbu 1:N medzi projektom a učiteľom, a projektom a predmetom. Cudzí kľúč patrí tej entite, ktorej inštancia môže mať vzťah iba s jednou inštanciou druhej entity. Napríklad, ak vieme, že projekt môže mať práve jedného učiteľa, cudzí kľúč „idUcitel“ pridávame do tabuľky „Projekt“ a tak zabránime rastu tabuľky „Ucitel“ do šírky.

Pre tvorbu relačného dátového modelu aplikácie bol využitý webový nástroj LucidChart.

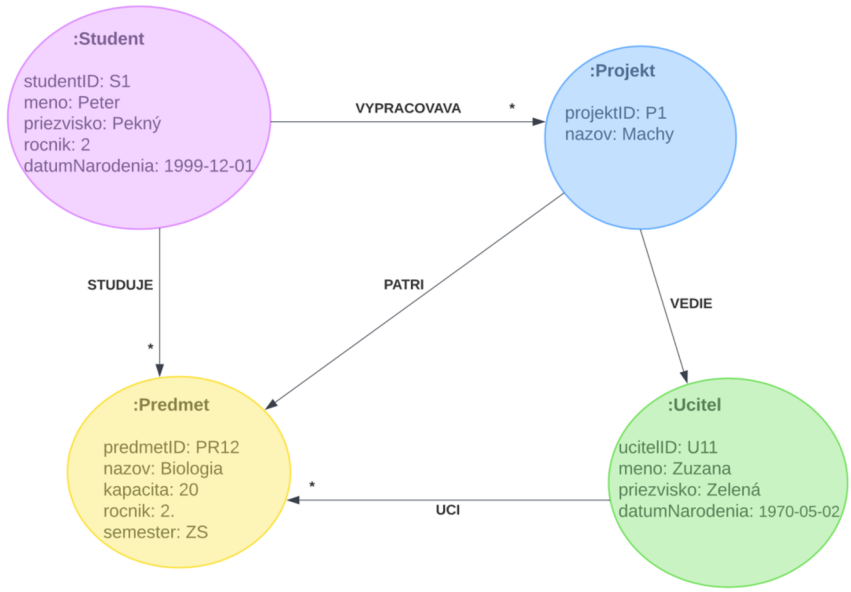


Obrázok 6 - Relačný dátový model

4.2.2. **Grafový dátový model**

Na rozdiel od relačného modelu, grafový je tvorený množinou uzlov a hrán medzi nimi. Relačná entita je v grafovom modeli reprezentovaná pomocou označenia (label) uzlu. Používanie labelov je veľmi užitočné, pretože od seba dokážu odlíšiť také uzly, ktoré obsahujú rovnaké vlastnosti. Samotný uzol je inštancia entity, a teda akoby jeden riadok v tabuľke. Stĺpce tabuľky (atribúty) sú v grafovom prostredí reprezentované vlastnosťami uzlu. Asociačné triedy a cudzie kľúče sa v grafovom modeli neuplatňujú, vzťahy sú riešené priamo. Vzťahy medzi uzlami sú pomenované a majú smer.

Pre tvorbu grafového modelu aplikácie bol rovnako použitý webový nástroj LucidChart.



Obrázok 7 - Grafový dátový model

4.3. Prostredie relačnej databázy MySQL

Pre porovnanie výhod a nevýhod oboch databázových systémov bolo najskôr potrebné zapísať rovnaké dáta do oboch databáz. V relačnej databáze nám podľa modelu zobrazeného na obrázku č. 6 vzniklo sedem tabuliek. Tabuľky obsahujúce údaje o študentoch, projektoch, učiteľoch, predmetoch a tri väzbové tabuľky – studentPredmet, studentProjekt a predmetUcitel.

Tabuľky sa v prostredí MySQL vždy vytvárali príkazom CREATE TABLE. V prípade tabuľky, evidujúcej údaje o študentoch, vyzeral dotaz nasledovne:

```
CREATE TABLE Student (idStudent VARCHAR(255) NOT NULL, meno VARCHAR(255), priezvisko VARCHAR(255), rocnik INT, datumNarodenia DATE, PRIMARY KEY (idStudent));
```

Hneď na začiatku, pri vytváraní tabuľky je potrebné definovať dátové typy a ich veľkosť. Nižšie sú pre názornosť zapísaných záznamov priložené tabuľky (okrem väzbových), ktoré vznikli exportom údajov z MySQL do Excelu. Každá tabuľka má pre lepšiu orientáciu farebne zvýraznené atribúty podľa vyššie uvedeného dátového modelu.

Pre vypísanie celého obsahu každej tabuľky bol použitý dotaz:

SELECT * FROM ‘konkrétna tabuľka‘;

Tabuľka „Student“ obsahuje záznamy o 15 študentoch.

Tabuľka 1 - záznamy o študentoch v MySQL

idStudent	meno	priezvisko	rocnik	datumNarodenia
S1	Peter	Pekný	2	01.12.1999
S2	Dominika	Ďaďová	3	18.08.2000
S3	Alžbeta	Mičáňová	1	15.01.2003
S4	Soňa	Beňová	1	16.02.2003
S5	Andrej	Adamec	3	11.07.2001
S6	Tadeáš	Baláž	2	27.11.2002
S7	Lívia	Fedorová	2	30.09.2002
S8	Roman	Bárta	3	13.06.2001
S9	Gregor	Bukovský	1	17.03.2003
S10	Matej	Fabián	3	02.05.2000
S11	Daniel	Červený	2	30.07.2002
S12	Zita	Drobná	3	29.07.2000
S13	Monika	Bosáková	1	10.02.2003
S14	Lenka	Antalová	1	25.01.2003
S15	Alojz	Capek	3	20.06.2000

Tabuľka „Predmet“ zobrazuje informácie o 5 vyučovaných predmetoch.

Tabuľka 2 - záznamy o predmetoch v MySQL

idPredmet	nazov	kapacita	rocnik	semester
PR11	Biochémia	20	1	ZS
PR12	Biológia	25	2	LS
PR13	Histológia	15	3	ZS
PR14	Fyziológia	20	2	LS
PR15	Latinčina	30	1	ZS

Tabuľka „Ucitel“ obsahuje záznamy o 10 učiteľoch, ktorí na škole vyučujú predmety.

Tabuľka 3 - záznamy o učiteľoch v MySQL

idUcitel	meno	priezvisko	datumNarodenia
U11	Zuzana	Zelená	02.05.1970
U12	Juraj	Jasný	01.03.1965
U13	Mária	Mocná	04.08.1975
U14	Michal	Malý	12.11.1986
U15	Tereza	Tyrolská	04.10.1990
U16	Peter	Planý	07.07.1987
U17	Sandra	Slušná	02.12.1991
U18	Lukáš	Lesný	09.01.1968
U19	Róbert	Rýchly	06.09.1978
U21	Jarmila	Ječná	03.04.1982

Tabuľka „Projekt“ zaznamenáva údaje o 17 projektoch. V tejto tabuľke si môžeme všimnúť, že projekt patrí vždy iba jednému predmetu a má prideleného práve jedného učiteľa. Rovnako si však vieme všimnúť, že jeden učiteľ môže byť vedúcim viacerých projektov, ako je tomu napríklad u učiteľa U11, ktorý je vedúcim projektov Machy, Dýchacia sústava a Sacharidy.

Tabuľka 4 - záznamy o projektoch v MySQL

idProjekt	nazov	idPredmet	idUcitel
P1	Machy	PR12	U11
P2	Časovanie slovies	PR15	U14
P3	Epitely	PR13	U15
P4	Rozmnožovanie rastlín	PR12	U12
P5	Bunka	PR12	U13
P6	Dýchacia sústava	PR14	U11
P7	História jazyka	PR15	U16
P8	Žľazy	PR14	U12
P9	Sacharidy	PR11	U11
P10	Vitamíny	PR11	U18
P11	Antibiotiká	PR11	U21
P12	Fyziológia krvi	PR14	U13
P13	Riasy	PR12	U17
P14	Farbivá	PR13	U12
P15	Techniky mikroskopovania	PR13	U18
P16	Cerebellum	PR14	U19
P17	Podstatné mená	PR15	U14

Každá väzbová tabuľka obsahuje dva cudzie kľúče podľa tabuliek, ktoré spája. Všetci študenti sú zapísaní do predmetu podľa ročníka, v ktorom sa nachádzajú (jednotlivé predmety sú vyučované iba v konkrétnych ročníkoch). Študenti môžu byť zapísaní do predmetu, ale ešte nebyť zapísaní k projektu.

4.3.1. **Dotazy v MySQL**

Nasledujúca časť obsahuje zápis dotazov a výsledky, ktoré ponúkla relačná databáza MySQL.

1. Dotaz - vypísať všetky predmety od jedného učiteľa

V tomto dotaze sme chceli vypísať všetky predmety, ktoré vyučuje konkrétny učiteľ. Keďže máme iba 10 učiteľov, podmienku sme ohraničili menom a priezviskom učiteľa. Ak by sme mali učiteľov viac a mohla by nastať zhoda mena a priezviska u niektorých z nich, vyhľadávanie by sme prispôsobili na použitie identifikátora.

```
SELECT pr.nazov, u.meno, u.priezvisko
FROM Predmet pr
JOIN predmetUcitel pu ON pr.idPredmet = pu.idPredmet
JOIN Ucitel u ON pu.idUcitel = u.idUcitel
WHERE u.priezvisko = 'Jasný' AND u.meno = 'Juraj';
```

Databáza nám vrátila tri riadky s predmetmi, ktoré vyučuje Juraj Jasný, viď tabuľka č.5.

Tabuľka 5 - výsledky dotazu č.1 v MySQL

nazov	meno	priezvisko
Biológia	Juraj	Jasný
Histológia	Juraj	Jasný
Fyziológia	Juraj	Jasný

2. Dotaz – vypísať všetky projekty v danom predmete

V tomto dotaze sme chceli vrátiť všetky projekty, ktoré sú vypísané v predmete histológia s ich ID a ID učiteľa, ktorý je ich vedúcim.

```
SELECT p.idProjekt AS 'ID Projektu', p.nazov AS 'Názov Projektu', p.idUcitel AS
        'ID Vedúceho projektu', pr.nazov as 'Názov Predmetu'
FROM Projekt p
JOIN Predmet pr ON p.idPredmet = pr.idPredmet
WHERE pr.nazov = 'Histologia';
```

Podľa výsledku uvedeného v tabuľke č. 6 môžeme vidieť, že v predmete histológia sú vypísané tri projekty – „Epitely“, „Farbivá“ a „Techniky mikroskopovania“.

Tabuľka 6 - výsledky dotazu č.2 v MySQL

ID Projektu	Názov Projektu	ID Vedúceho projektu	NázovPredmetu
P3	Epitely	U15	Histológia
P14	Farbivá	U12	Histológia
P15	Techniky mikroskopovania	U18	Histológia

3. Dotaz – vypísať všetky projekty od daného učiteľa v danom predmete

V tomto dotaze sme chceli vypísať všetky projekty, ktorých vedúci je učiteľ Malý, no nie vo všetkých predmetoch, ktoré učí, ale len v predmete latinčina.

```
SELECT p.nazov AS 'Názov projektu',u.meno AS 'Meno vedúceho projektu',
        u.priezvisko AS 'Priezvisko Vedúceho projektu', pr.nazov AS 'Názov
        predmetu'
FROM Projekt p
JOIN Ucitel u ON p.idUcitel = u.idUcitel
JOIN Predmet pr ON p.idPredmet = pr.idPredmet
WHERE u.priezvisko = 'Malý' AND pr.nazov = 'Latinčina';
```

MySQL nám správne poskytla výsledok vo forme 2 riadkov s názvami projektov „Podstatné mená“ a „Časovanie slovies“ pre Michala Malého, ktoré sú zaznamenané v tabuľke č. 7.

Tabuľka 7 - výsledky dotazu č.3 v MySQL

Názov projektu	Meno vedúceho projektu	Priezvisko Vedúceho projektu	Názov predmetu
Podstatné mená	Michal	Malý	Latinčina
Časovanie slovies	Michal	Malý	Latinčina

4. Dotaz – vypísať všetky projekty vo všetkých predmetoch študenta

V nasledujúcom príkaze sme chceli zistiť, aké sú všetky projekty, na ktoré je zapísaný konkrétny študent vo všetkých jeho predmetoch, ktoré aktuálne navštevuje. Opäť, keďže sa jedná o pomerne malú evidenciu s unikátnymi menami, pre obmedzenie sme si vybrali meno a priezvisko študenta.

```
SELECT s.meno AS 'Meno študenta', s.priezvisko AS 'Priezvisko študenta', pr.nazov
        AS 'Názov predmetu', p.nazov AS 'Názov projektu'
FROM Projekt p
JOIN Predmet pr ON p.idPredmet = pr.idPredmet
JOIN studentProjekt sp ON p.idProjekt = sp.idProjekt
JOIN Student s ON s.idStudent = sp.idStudent
WHERE s.meno = 'Peter' AND s.priezvisko = 'Pekný';
```

Podľa výsledku môžeme vidieť, že Peter Pekný študuje dva predmety a na oboch už má zapísaný projekt. Ak by sme si overili správnosť s tabuľkou „Student“, môžeme si všimnúť, že Peter Pekný sa nachádza v 2. ročníku a ak podľa tabuľky „Predmet“ skontrolujeme, aké predmety sa v tomto ročníku vyučujú, odpovede sa nám budú zhodovať. V 2. ročníku sa totiž vyučuje práve biológia a fyziológia.

Tabuľka 8 - výsledky dotazu č.4 v MySQL

Meno študenta	Priezvisko študenta	Názov predmetu	Názov projektu
Peter	Pekný	Biológia	Machy
Peter	Pekný	Fyziológia	Dýchacia sústava

Ak by sme zmenili meno študenta na Tadeáša Baláža, vypíše sa nám iba jeden projekt – „Machy“, na ktorý je prihlásený a jeden predmet, v ktorom tento projekt vypracováva, aj keď študuje predmety dva. Predmet fyziológia nám MySQL vo výsledkoch nezobrazí, pretože v tomto prípade u študenta Baláža neexistuje žiadny vzťah medzi inštanciou

Fyziológia v tabuľke Predmet a hociktorou inštanciou z tabuľky Projekt. Študent v druhom navštevovanom predmete zapísaný projekt ešte nemá.

Tabuľka 9 - výsledky upraveného dotazu č.4 v MySQL s priezviskom „Baláž“

Meno študenta	Priezvisko študenta	Názov predmetu	Názov projektu
Tadeáš	Baláž	Biológia	Machy

Ak by sme chceli, aby nám databáza ponúkla vo výsledkoch aj druhý predmet, na ktorý je študent prihlásený, museli by sme túto požiadavku realizovať dvomi príkazmi. Oddelene by sme si v jednom dotaze vypísali všetky predmety, ktoré navštevuje tento študent a v ďalšom všetky projekty, ktoré vypracováva. Následne by bolo nutné ich spojiť pomocou špecializovaného software. Dotazy by vyzerali takto:

```
SELECT s.meno AS 'Meno študenta', s.priezvisko AS 'Priezvisko študenta', pr.nazov
        AS 'Názov predmetu'
FROM studentPredmet spt
JOIN Predmet pr ON spt.idPredmet = pr.idPredmet
JOIN Student s ON s.idStudent = spt.idStudent
WHERE s.meno = ‘Tadeáš’ AND s.priezvisko = 'Baláž';
```

```
SELECT s.meno AS ‘Meno študenta’, s.priezvisko AS ‘Priezvisko študenta’,
pr.nazov
        AS 'Názov predmetu', p.nazov AS 'Názov projektu'
FROM Projekt p
JOIN Predmet pr ON p.idPredmet = pr.idPredmet
JOIN studentProjekt sp ON p.idProjekt = sp.idProjekt
JOIN Student s ON s.idStudent = sp.idStudent
WHERE s.priezvisko = 'Baláž';
```

5. Dotaz – vypísať, ktorí študenti zapísaní na daný predmet ešte nemajú pridelený projekt

V databáze evidujeme študentov, ktorí na predmet už zapísaní sú, no zatiaľ v ňom nemajú pridelený projekt. Mená týchto študentov si chceme vypísať. V relačnej databáze to urobíme, napríklad v predmete latinčina, nasledovne:

```
SELECT s.idStudent AS 'ID', s.meno AS 'Meno študenta', s.priezvisko AS 'Priezvisko
        študenta', pr.nazov AS 'Názov predmetu'
FROM Predmet pr
JOIN studentPredmet stpr ON pr.idPredmet = stpr.idPredmet
JOIN Student s ON s.idStudent = stpr.idStudent
WHERE pr.nazov = 'Latinčina' and s.idStudent
NOT IN
        (SELECT sp.idStudent FROM studentProjekt sp
        JOIN Projekt p ON p.idProjekt = sp.idProjekt
        JOIN Predmet pr ON pr.idPredmet = p.idPredmet
        WHERE pr.nazov = 'Latinčina');
```

Po vykonaní tohoto dotazu dostaneme nasledovný výsledok:

Tabuľka 10 - výsledky dotazu č.5 v MySQL

ID	Meno študenta	Priezvisko študenta	Názov predmetu
S13	Monika	Bosáková	Latinčina
S14	Lenka	Antalová	Latinčina

Odpoveďou na dotaz, ktorí študenti študujú predmet Latinčina, no zatiaľ v ňom nemajú zapísaní projekt, sú študentky Bosáková a Antalová.

Pre overenie správnosti dotazu konštruujeme opačný dotaz, a teda vypísanie všetkých študentov, ktorí študujú predmet Latinčina a projekt v ňom pridelený majú.

```
SELECT s.idStudent AS 'ID', s.meno AS 'Meno študenta', s.priezvisko
        AS 'Priezvisko študenta', pr.nazov AS 'Názov predmetu', p.nazov AS 'Názov
        projektu'
FROM studentProjekt sp
JOIN Student s ON s.idStudent = sp.idStudent
JOIN Projekt p ON p.idProjekt = sp.idProjekt
JOIN Predmet pr ON pr.idPredmet = p.idPredmet WHERE pr.nazov = 'Latinčina';
```

Podľa výsledku môžeme vidieť, že predošlý dotaz poskytol správnu odpoveď, pretože na predmet sú zapísaní piati študenti prvého ročníka a iba nasledovní traja z nich – Mičáňová, Beňová a Bukovský, majú pridelený projekt.

Tabuľka 11 - výsledky dotazu, ktorý slúži na overenie správnosti dotazu č.5 v MySQL

ID	Meno študenta	Priezvisko študenta	Názov predmetu	Názov projektu
S3	Alžbeta	Mičáňová	Latinčina	Časovanie slovies
S4	Soňa	Beňová	Latinčina	Časovanie slovies
S9	Gregor	Bukovský	Latinčina	História jazyka

4.4 Prostredie grafovej databázy Neo4j

V grafovej databáze Neo4j boli najskôr vytvorené jednotlivé uzly, ktoré mali spoločný label. Labely boli štyri - Student, Projekt, Predmet a Ucitel. Každý uzol predstavoval konkrétnu inštanciu entity. Následne boli medzi uzly pridané vzťahy.

Uzly sa v prostredí Neo4j vytvárali príkazom CREATE. V prípade uzlu „Zuzana Zelená“ s označením „Ucitel“ vyzeral dotaz nasledovne:

```
CREATE (u1: Ucitel {ucitelID: 'U11', meno: 'Zuzana', priezvisko: 'Zelená',
datumNarodenia: date('1970-05-02')})
```

Pre tvorbu vzťahu medzi konkrétnymi uzlami bol používaný dotaz, ktorý zahrňoval kľúčové slová MATCH, WHERE, CREATE a RETURN.

Napríklad, ak sme chceli zapísať vzťah, ktorý určoval, že študentka Ďaďová študuje predmet histológia, použili sme nasledovný dotaz:

```
MATCH (s:Student), (pr:Predmet)
WHERE s.priezvisko = "Ďaďová" and pr.nazov = "Histológia"
CREATE (s)-[w:STUDUJE]->(pr)
RETURN type(w)
```

Ako už bolo spomenuté vyššie v texte, keďže ide o malú evidenciu, kde sme si istí, že sa nevyskytujú rovnaké priezviská u viacerých ľudí – či už študentov alebo učiteľov, mohli sme priradiť predmet študentke podľa jej priezviska.

Ak by sme chceli použiť identifikátor pri priradení projektu k predmetu, príkaz by vyzeral takto:

```
MATCH (p:Projekt), (pr:Predmet)
WHERE p.projektID= "P17" and pr.predmetID = "PR15"
CREATE (p)-[w:PATRI]->(pr)
RETURN type(w)
```

Oproti relačnej databáze MySQL, kde sme museli dopredu definovať aj asociačné triedy studentPredmet, studentProjekt a predmetUcitel, aby sme mohli realizovať dotazy aj medzi tabuľkami so vzťahmi M:N, v grafovej databáze sme tieto väzby definovali priamo spojením dvoch konkrétnych uzlov.

Keďže údaje, ktoré sú zapísané v oboch databázach, už boli poskytnuté v tabuľkách č. 1 – 4 (kapitola 4.3, strany 27-28 tejto práce), nebudú v tejto časti už znovu zobrazené.

V Neo4j je možné zobraziť výsledok pomocou grafu, tabuľky alebo textu. Pre zachovanie prehľadnosti a orientácie entít boli uzly patriace jednému labelu zvýraznené rovnakou farbou ako atribúty v jednotlivých tabuľkách relačnej databáze a vyššie uvedenom dátovom modeli.

4.4.1. **Dotazy v Neo4j**

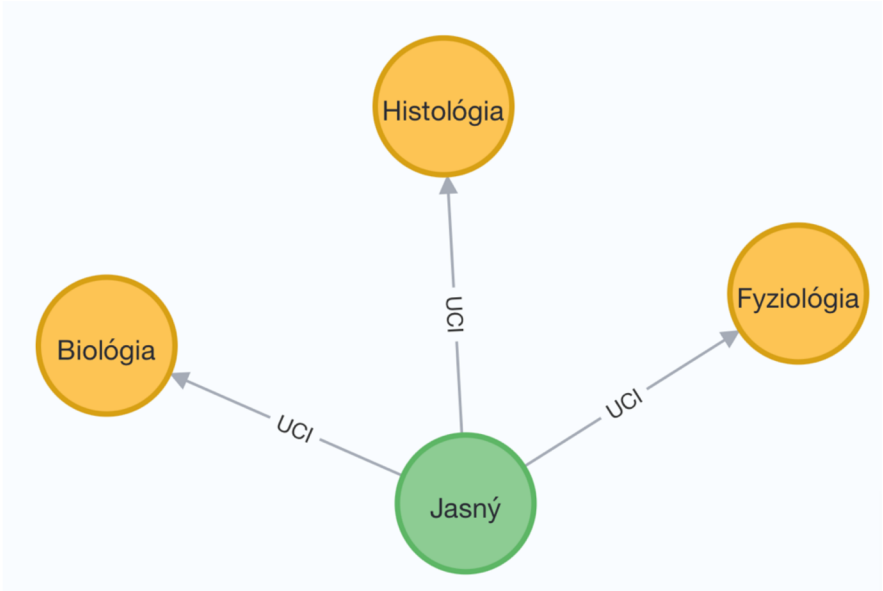
Nasledujúca časť obsahuje zápis dotazov a výsledky, ktoré ponúkla grafová databáza Neo4j.

1. Dotaz - vypísať všetky predmety od jedného učiteľa

Pre vypísanie všetkých predmetov, ktoré učí učiteľ Jasný sme použili nasledujúci dotaz. Rovnako ako v MySQL, podmienku sme ohraničili priezviskom učiteľa.

```
MATCH (u:Ucitel)-[:UCI]-(pr:Predmet)
WHERE u.priezvisko = 'Jasný'
RETURN u, pr
```

Databáza nám vrátila nasledujúci graf, na ktorom môžeme vidieť, že učiteľ Jasný vyučuje tri predmety, a to biológiu, histológiu a fyziológiu.

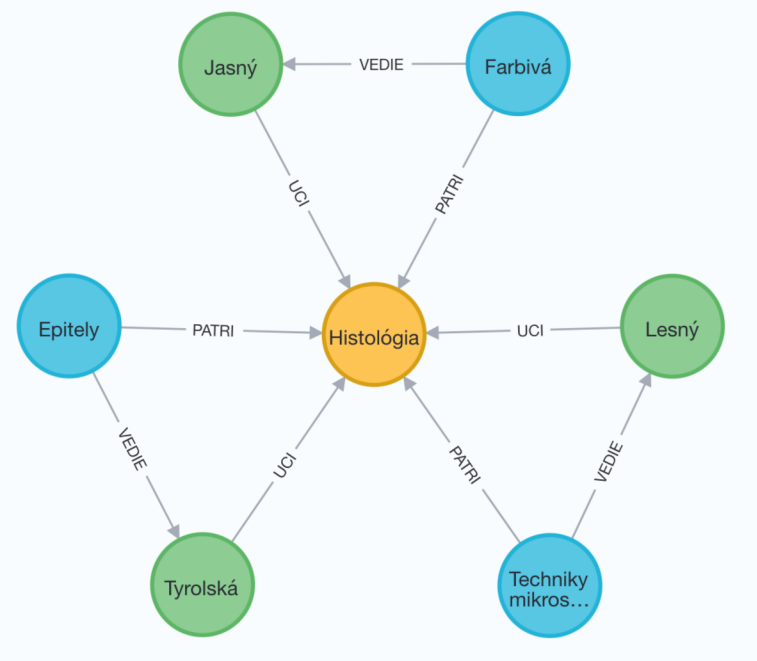


Obrázok 8 - výsledky dotazu č.1 v Neo4j

2. Dotaz – vypísať všetky projekty v danom predmete

Zisťovali sme, ktoré všetky projekty sú vypísané v predmete histológia.

```
MATCH (p:Projekt)-[:PATRI]-(pr:Predmet),(p:Projekt)-[:VEDIE]-(u:Ucitel)
WHERE pr.nazov = 'Histológia'
RETURN p, pr,u
```



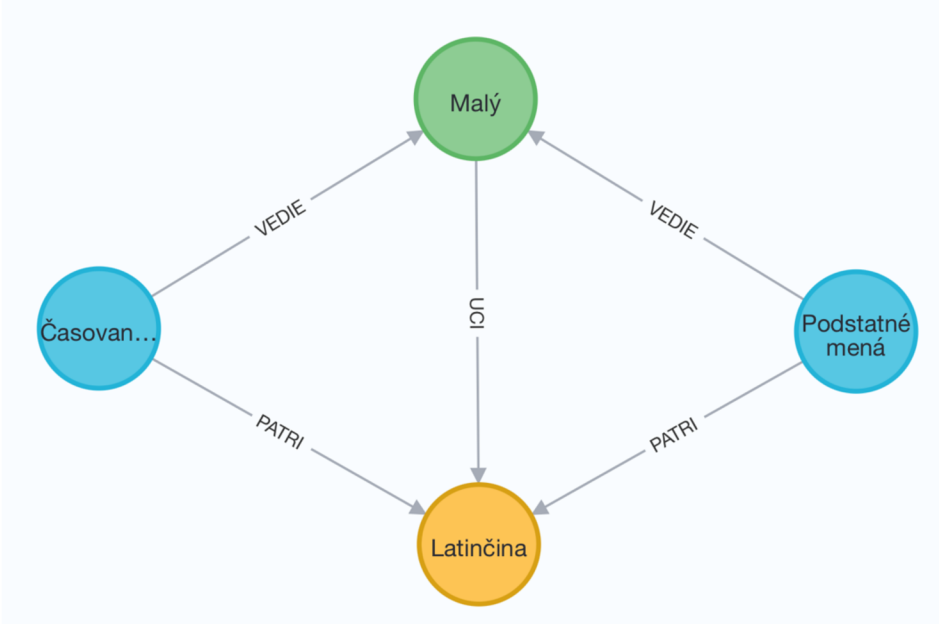
Obrázok 9 - výsledky dotazu č.2 v Neo4j

Podľa výsledku uvedeného na obrázku č. 9 môžeme vidieť, že v predmete histológia sú vypísané tri projekty – „Epitely“, „Farbivá“ a „Techniky mikroskopovania“.

3. Dotaz – vypísať všetky projekty od daného učiteľa v danom predmete

Pomocou nasledujúceho dotazu sme boli schopní zobraziť projekty, ktorých vedúcim je učiteľ Malý v predmete latinčina.

```
MATCH (p:Projekt)-[:VEDIE]-(u:Ucitel), (u:Ucitel)-[:UCI]-(pr:Predmet)
WHERE pr.nazov =  'Latinčina' and u.priezvisko =  'Malý'
RETURN p, u, pr
```



Obrázok 10 - výsledky dotazu č.3 v Neo4j

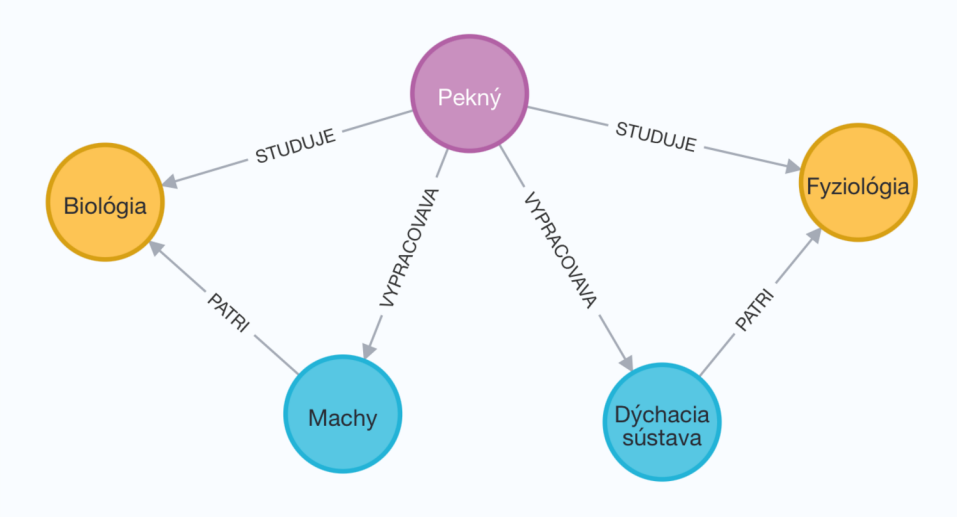
Na základe výsledku reprezentovaného obrázkom č. 10 vidíme, že tento učiteľ je vedúcim dvoch projektov - „Podstatné mená“ a „Časovanie slovies“.

4. Dotaz – vypísať všetky projekty vo všetkých predmetoch študenta

V tomto dotaze sme zisťovali, na ktoré všetky projekty je zapísaný študent Pekný v rámci všetkých jeho predmetov.

```
MATCH (s:Student)-[:VYPRACOVAVA]-(p:Projekt),(s:Student)-[:STUDUJE]-
        (pr:Predmet)
WHERE s.priezvisko = 'Pekný'
RETURN p, s, pr
```

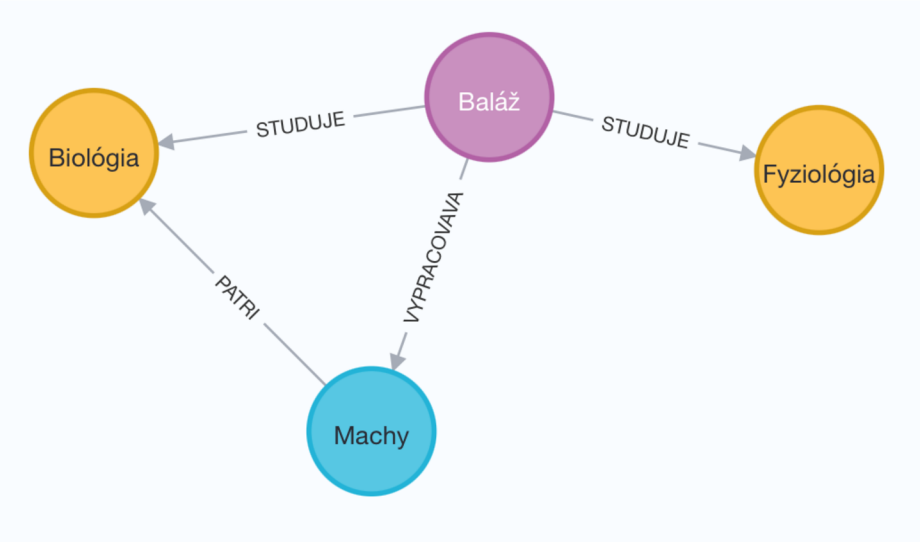
Databáza nám poskytla výsledok, podľa ktorého môžeme vidieť, že študent Pekný študuje dva predmety. V oboch predmetoch má zapísaný jeden projekt. Projekt „Machy“ v predmete biológia a projekt „Dýchacia sústava“ v predmete fyziológia.



Obrázok 11 - výsledky dotazu č.4 v Neo4j

Ak aj v tomto prípade tak, ako v MySQL zmeníme meno študenta na Tadeáša Baláža, vypíše sa nám iba jeden projekt – „Machy“, na ktorý je prihlásený, aj keď je zapísaný do predmetov dvoch. Podľa výsledku vidíme, že v predmete fyziológia ešte projekt nevypracováva. Na rozdiel od MySQL, v prostredí Neo4j stačí túto požiadavku realizovať jedným dotazom.

```
MATCH (s:Student)-[:VYPRACOVAVA]-(p:Projekt),(s:Student)-[:STUDUJE]-
        (pr:Predmet)
WHERE s.priezvisko = 'Baláž'
RETURN p, s, pr
```



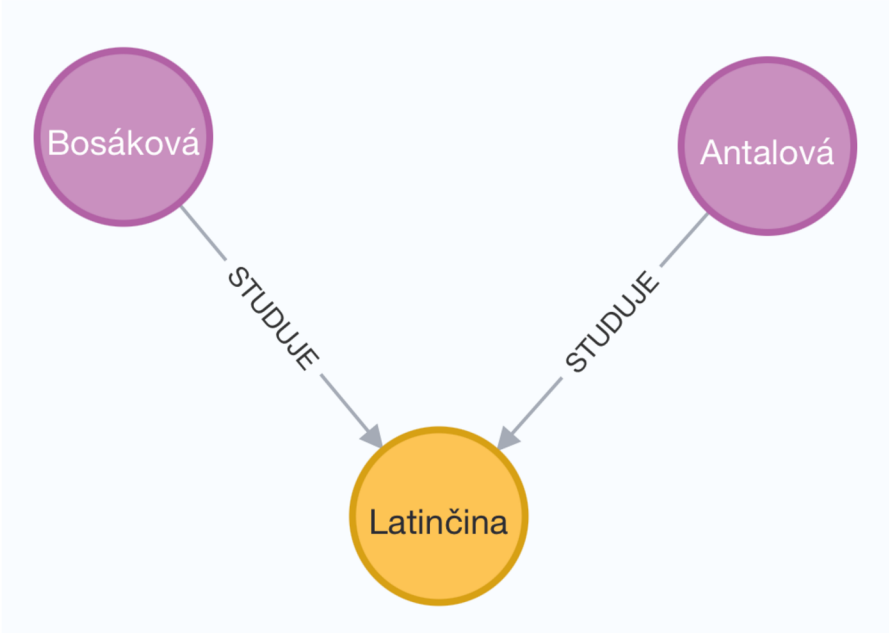
Obrázok 12 - výsledky upraveného dotazu č.4 v Neo4j s priezviskom „Baláž“

5. Dotaz – vypísať, ktorí študenti zapísaní na daný predmet ešte nemajú pridelený projekt

Ak si želáme zistiť, ktorí študenti v našej evidencií sú zapísaní na konkrétny predmet, no zatiaľ v ňom nemajú pridelený projekt, realizujeme dotaz týmto spôsobom.

```
MATCH (s:Student)-[:STUDUJE]-(pr:Predmet {nazov:'Latinčina'} )
WHERE NOT
        (s)-[:VYPRACOVAVA]-(:Projekt)-[:PATRI]->(pr:Predmet{nazov:'Latinčina'})
RETURN *
```

Podľa obrázku č. 13 môžeme vidieť, že v predmete latinčina zatiaľ nemajú pridelený projekt dve študentky, Antalová a Bosáková.

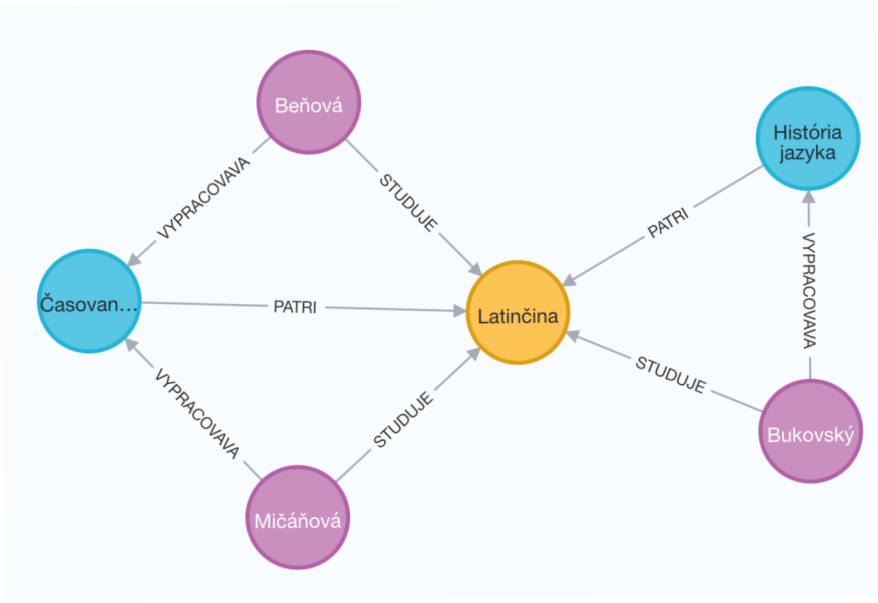


Obrázok 13 - výsledky dotazu č.5 v Neo4j

Ak by sme si chceli overiť správnosť výsledku vypísaním všetkých študentov, ktorí predmet latinčina navštevujú a zároveň v ňom už majú zapísaný projekt, tak, ako tomu boli pri použití MySQL, dotaz uvedený nižšie by nám poskytol túto odpoveď.

Projekt majú zatiaľ zapísaný traja študenti – Beňová, Bukovský a Mičáňová, ani jeden z nich sa však nezhoduje s uzlami „Antalová“ a „Bosáková“.

```
MATCH (s:Student)-[:STUDUJE]-(pr:Predmet),(s:Student)-[:VYPRACOVAVA]-
        (p:Projekt), (p:Projekt)-[:PATRI]-(pr:Predmet)
WHERE pr.nazov = 'Latinčina'
RETURN s, pr, p
```



Obrázok 14 - výsledky dotazu, ktorý slúži na overenie správnosti dotazu č.5 v Neo4j

5 Výsledky a diskusia

Nasledujúca časť prezentuje a hodnotí rozdiely, výhody, a nevýhody oboch použitých technológií na základe vykonávaných dotazov, ktoré boli v praktickej časti doplnené o tabuľky, obrázky a časti kódu.

Na základe výsledkov mojej práce by som za výhody Neo4j a naopak nevýhody MySQL označila dĺžku zápisu a s ním súvisiacu náročnosť samotnej konštrukcie, a rýchlosť vykonania dotazu, ktoré ovplyvňujú objem potrebnej programátorskej práce.

Už na prvý pohľad je zrejmé, že dĺžka zápisu dotazu je v prípade relačnej databázy väčšia a tým pádom je potrebné nad jeho konštrukciou stráviť viac času. Pri štvrtom upravenom dotaze, zameranom na študenta, ktorý vypracováva iba jeden projekt, aj keď je zapísaný na predmety dva, je dokonca v MySQL nutné realizovať požiadavku dvomi dotazmi a následne ich spájať pomocou špecializovaného software. V prostredí grafovej databáze sa táto požiadavka realizuje jedným krátkym príkazom.

Ak je pre dosiahnutie požadovaného výsledku nutné dotaz zapisovať väčším počtom znakov a riadkov v kóde, môže to pre niektorých užívateľov znamenať zvýšenie náročnosti konštrukcie samotného dotazu a zníženie jeho prehľadnosti.

Z vlastnej skúsenosti môžem povedať, že príkazy v Neo4j sa mi vďaka ich kratšiemu zápisu konštruovali ľahšie a napísať príkazy v MySQL, ktoré často spájali naraz tri tabuľky a viac, mi zabralo viac času a trvalo mi dlhšie, kým som sa dopracovala k správnemu zápisu. Čas potrebný k zápisu dotazu a náročnosť jeho konštrukcie môžu v praxi výrazne ovplyvniť objem programátorskej práce a s tým súvisiaci priebeh vývoja.

Pri porovnávaní som sa zamerala aj na výkonnosť, a teda rýchlosť spracovania dotazu a zobrazenia výsledku jednotlivými databázami. Ku každému dotazu som mala k dispozícii výsledný čas v milisekundách.

Podľa podmienok, v ktorých som pracovala, nedokážem objektívne konštatovať, ktorá z databáz bola rýchlejšia, pretože každá z nich bola spustená v odlišnom prostredí.

Pre prácu s relačnou databázou som využívala nástroj MySQL Workbench bežiaci na desktope, na rozdiel od práce s grafovou databázou, pre ktorú som využívala Neo4j spustený v prehliadači. Prihliadnuc k týmto podmienkam a výsledkom mojej práce môžem však povedať, že dotazy v Neo4j neboli pomalšie, práve naopak.

Mnou navrhnutá časť študijnej evidencia bola malá a nad obomi databázami som realizovala pomerne jednoduché dotazy, preto si dovolím porovnanie doplniť o experiment realizovaný Jonasom Partnerom a Aleksou Vukotic, ktorý som opisovala už v teoretickej časti (kapitola 3.4.1, strany 18-19 tejto práce) a ktorého výsledky nie sú v rozpore s mojim vyššie uvedeným tvrdením.

Experiment aplikovali na prostredie sociálnej siete, v ktorej sa medzi dátami nachádzalo veľké množstvo väzieb. S postupným zvyšovaním náročnosti dotazu a hĺbky väzieb, do ktorej museli databázy zájsť pri jeho vyhodnotení, začala rýchlosť vykonávania dotazov u relačnej databázy klesať. Pri treťom stupni hĺbky sa oproti grafovej znížila o 180-krát s trvaním 30 sekúnd, pri štvrtom bola Neo4j takmer o 1 135 - krát rýchlejšia a MySQL poskytla výsledky dotazu po necelých 26 minútach.

Na druhej strane, za stále pretrvávajúcu výhodu relačných databázových prostriedkov považujem veľkú základňu užívateľov spojenú s dobre zabehnutou databázovou platformou.

Relačné databázy sú využívané denne už niekoľko desiatok rokov a stále výrazne dominujú v rôznych rebríčkoch popularity. DB-Engines Ranking prideľuje každému databázovému prostriedku skóre podľa kritérií, medzi ktoré patrí napríklad počet zmienení vo vyhľadávačoch a ich frekvencia, počet pracovných ponúk súvisiacich s daným databázovým prostriedkom alebo počet ich zmienení na sociálnych sieťach. Podľa posledného zverejnenia na začiatku marca 2023 sa na prvých troch priečkach umiestnili relačné databázové prostriedky. MySQL patrilo druhé miesto. S popularitou relačných databáz úzko súvisí aj veľkosť užívateľskej základne.

Grafové databázy naberajú na popularite postupne. Neo4j skončila v tomto rebríčku na 20. mieste, preto môžeme predpokladať, že jej aktuálna základňa užívateľov bude menšia. V praxi to môže v prípade výskytu problému pri práci s databázou znamenať spomalenie priebehu vývoja. Keďže je základňa menšia, môže byť časovo náročnejšie nájsť technickú pomoc a úspešne problém vyriešiť. (DB-Engines, 2023)

Domnievam sa, že grafovému databázovému prostriedku dajú prednosť skôr nováčikovia, ktorí sa so žiadnym databázovým prostriedkom ešte nestretli, nie sú zaťažení mentálnym modelom, a nemajú s čím porovnávať.

Užívatelia, ktorí pracujú s relačnými databázami sú zvyknutí na ich špecifické fungovanie. Na základe vytvoreného mentálneho modelu sa pri stretnutí s grafovou databázou snažia uplatňovať už zabehnuté algoritmizácie a práca s ňou im preto nemusí vyhovovať.

Výsledky tejto práce sú limitované menším množstvom dát, s ktorými som pracovala, pri väčšej vzorke by sa jednotlivé porovnania mohli mierne líšiť. Avšak ak by sme v nadväzujúcej štúdií predpokladali väčšie množstvo dát a komplexnejšie vzťahy medzi nimi, výsledky takejto štúdie by mohli ešte viac podčiarknuť tie, ktoré vyplynuli z tejto práce.

Oba databázové prostriedky vykazujú výhody aj nevýhody. Z bežného užívateľského hľadiska, ak do úvahy neberieme prácu s nadmerným množstvom dát s veľkým počtom väzieb, sa však nedá jednoznačne konštatovať, ktorá platforma je lepšia. Každý užívateľ je individuálny, pri práci s databázou môže uprednostňovať odlišné chovanie či inú formu zobrazovania výsledkov.

6 Záver

Cieľom bakalárskej práce bolo navrhnúť, vytvoriť a otestovať malú databázovú aplikáciu pre časť študijnej evidencie v prostredí grafovej a relačnej databáze.

Práca bola rozdelená na dve časti. V teoretickej časti boli popísané nástroje databázových technológií. Ponúkla stručné predstavenie oboch databáz s cieľom ozrejmiť základné rozdiely medzi nimi. Predstavený bol aj jazyk Cypher Query Language.

Praktická časť bola zameraná na návrh samotnej aplikácie v prostredí MySQL a Neo4j a dokumentáciu databázových modelov za pomoci štandardu UML. Grafová aj relačná databáza obsahovali rovnaké dáta a vykonávané boli dotazy, ktoré vracali rovnaký výsledok. V závere práce boli databázy vzájomne porovnané z hľadiska výkonnosti, dĺžky a náročnosti zápisu, a s tým súvisiacou náročnosťou vývoja, a veľkosti základne užívateľov. Porovnanie bolo doplnené o tabuľky, obrázky a časti kódu.

Pri zložitejších dotazoch sa dĺžka a náročnosť konštrukcie zápisu pri použití relačnej databáze zvyšovala a rýchlosť vykonania dotazu znižovala, čo by v praxi mohlo viesť k navýšeniu objemu potrebnej programátorskej práce. Na druhej strane, v rebríčkoch popularity stále výrazne dominujú relačné databázy, a v prípade výskytu problému je viac pravdepodobné, že vďaka väčšej užívateľskej základni bude jeho vyriešenie časovo menej náročné.

Keďže som sa v tejto práci nevenovala iba grafovým databázam, ale aj ich porovnaniu s relačnými, s ktorými som sa stretla už počas štúdia, mohla som tak naraz aplikovať nadobudnuté vedomosti spolu s rozšírením si obzorov o nové.

7 Zoznam použitých zdrojov

BRUGGEN, R. (2014) *Learning Neo4j – Run blazingly fast queries on complex graph datasets with the power of the Neo4j graph database*. First edition. Birmingham: Packt Publishing Ltd. ISBN 978-1-84951-716-4.

DB-Engines (2023) *DB-Engines Ranking – popularity ranking of database management systems.* [online] Dostupné z: https://db-engines.com/en/ranking. [Citované 2023-03-10].

GeeksforGeeks (2022) *Difference between Hierarchical and Network Data Model – GeeksforGeeks*. [online] Dostupné z: https://www.geeksforgeeks.org/difference-between-hierarchical-and-network-data-model/ [Citované 2022-08-13].

GeeksforGeeks (2022) *History of DBMS – GeeksforGeeks*. [online] Dostupné z: https://www.geeksforgeeks.org/history-of-dbms/ [Citované 2022-08-13].

MEIER, A. a KAUFMANN, M. (2019) *SQL & NoSQL databases: models, languages, consistency options and architectures for Big data management.* Wiesbaden: Springer. ISBN 978-3-658-24548-1.

MemgraphDB (2023) *MemgraphDB*. [online] Dostupné z: https://memgraph.com/memgraphdb [Citované 2023-01-30].

MemgraphDB (2023) *Neo4j vs Memgraph – How to Choose a Graph Database?* [online] Dostupné z: https://memgraph.com/blog/neo4j-vs-memgraph. [Citované 2023-01-30].

MemgraphDB (2023) *Getting started with Cypher | Memgraph Docs.* [online] Dostupné z: https://memgraph.com/docs/cypher-manual. [Citované 2023-01-30].

MERUNKA, V. (2002) *Objektový přístup v databázových systémech.* Praha: Credit. ISBN 80-213-0882-6.

MySQL (2022) *MySQL Connectors.* [online] Dostupné z: https://www.mysql.com/products/connector/. [Citované 2022-08-22].

Neo4j: Graph Data Platform (2022) *Neo4j Drivers – Getting Started.* [online] Dostupné z: https://neo4j.com/docs/getting-started/current/languages-guides/#:~:text=Neo4j%20officially%20supports%20the%20drivers,for%20the%20binary%20Bolt%20protocol [Citované 2022-08-20].

Neo4j: Graph Data Platform (2022) *Introduction – Neo4j Cypher Manual*. [online] Dostupné z: https://neo4j.com/docs/cypher-manual/current/introduction/ [Citované 2022-08-15].

Neo4j Graph Data Platform (2022) *How much faster is a graph database, really?* [online] Dostupné z: https://neo4j.com/news/how-much-faster-is-a-graph-database-really/ [Citované 2022-08-14].

Neo4j: Graph Data Platform (2022) *Comparing SQL with Cypher – Developer Guides*. [online] Dostupné z: https://neo4j.com/developer/cypher/guide-sql-to-cypher/ [Citované 2022-08-15].

Neo4j: Graph Data Platform (2022) *Concepts: Relational to Graph – Developer Guides*. [online] Dostupné z: https://neo4j.com/developer/graph-db-vs-rdbms/ [Citované 2022-08-15].

Neo4j: Graph Data Platform (2022) *Neo4j Aura – Fully Managed Cloud Solution*. [online] Dostupné z: https://neo4j.com/cloud/platform/aura-graph-database/?ref=neo4j-home-hero [Citované 2022-08-15].

Neo4j: Graph Data Platform (2022) *The Neo4j Cypher Manual v4.4 - Neo4j Cypher Manual*. [online] Dostupné z: https://neo4j.com/docs/cypher-manual/current/ [Citované 2022-08-15].

Neo4j Graph Data Platform (2022) *Cypher Query Language – Developer Guides*. [online] Dostupné z: https://neo4j.com/developer/cypher/ [Citované 2022-08-14].

One Property Graph Query Language (2019) *The GQL Manifesto*. [online] Dostupné z: https://gql.today/. [Citované 2022-08-21].

PGQL | Property Graph Query Language (2022) *PGQL | Property Graph Query Language.* [online] Dostupné z: https://pgql-lang.org/. [Citované 2022-08-21].

ROBINSON, I., WEBBER, J. a EIFREM, E. (2015) *Graph databases.* Second edition. Sebastopol: O'Reilly Media. ISBN 978-1-491-93089-2.

SASAKI, B. (2018) *Graph Databases for Beginners: Why a Database Query Language Matters (More Than You Think) - Neo4j Graph Data Platform*. [online] Dostupné z: https://neo4j.com/blog/why-database-query-language-matters [Citované 2022-08-14].

Statistics and Data (2022) *The Most Popular Databases – 2006/2022.* [online] Dostupné z: https://statisticsanddata.org/data/most-popular-databases-2006-2022/. [Citované 2023-01-30].

VOSTROVSKÝ, V. (2014) *Vytváření databází v ORACLE.* Praha: Česká zemědělská univerzita v Praze, Provozně ekonomická fakulta. ISBN 978-80-213-1191-6.

VUKOTIC, A., ABEDRABBO, T., FOX, D., PARTNER, J. a WATT, N. (2015) Neo*4j in action*. Shelter Island: Manning Publications. ISBN 978-1-617-29076-3.

Wikipédia (2022) *Databáze - Wikipedia*. [online] Dostupné z: https://cs.wikipedia.org/wiki/Datab%C3%A1ze [Citované 2022-08-14]

Wikipédia (2022). *Agile software development – Wikipedia*. [online] Dostupné z: https://en.wikipedia.org/wiki/Agile_software_development [Citované 2022-08-15].

Wikipédia (2023) Graph Query Language – Wikipedia. [online] Dostupné z: https://en.wikipedia.org/wiki/Graph_Query_Language. [Citované 2023-01-30].

8 Zoznam obrázkov a tabuliek

8.1 Zoznam obrázkov

8.2 Zoznam tabuliek