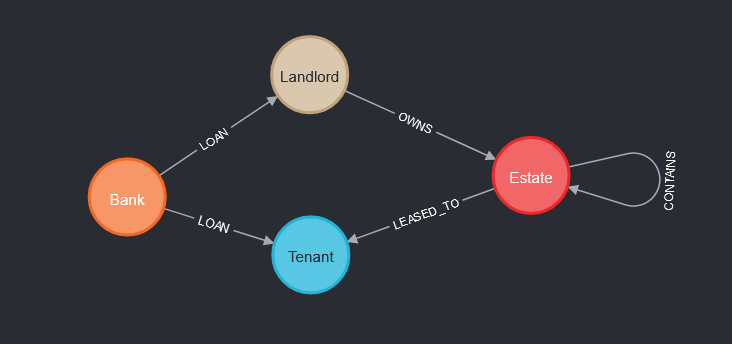
# Cvičení Neo4j

## Pracovní databáze

Pro cvičení jsem vymyslel jednoduchou databázi nájemníků (tenant) a pronajímatelů (landloard) v Praze a Brně. V databázi si držíme banky (id, název, kód), nájemníky a pronajímatele (jméno, příjmení, email, telefon) a nemovitosti (typ, adresa, cena). Některé osoby jsou jak nájemníci tak pronajímatelé. Banky můžou půjčovat peníze pronajímatelům (hypotékou či přes nemovitostní fond) a nájemníkům (jednorázová půjčka). U půjček nás zajímá částka a úroková sazba. Pronajímatel může vlastnit nemovitosti a nájemník jí může mít pronajatou (s danou měsiční cenou). Nemovitost může být byt, dům, garáž, sklep nebo pozemek, je daná nějakou adresou, a má nějakou cenu. Nemovitost může mít navíc další nemovitost jako součást sama sebe (například k bytu patří nějaká externí garáž). Databáze je udělaná primárně jako grafová, ale zároveň je dělaná tak, aby všechny úlohy šly udělat i v MongoDB pro srovnání. Z tohoto důvodu zde nebude moc úloh orientovaných čistě na práci s grafy.

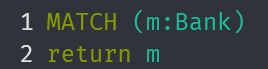
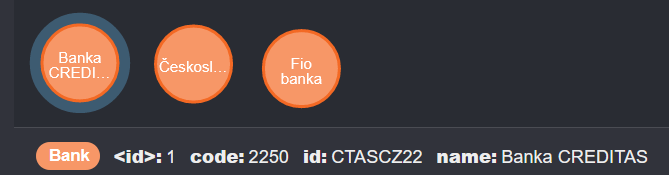
Create script je v souboru: neo4j.txt

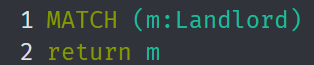
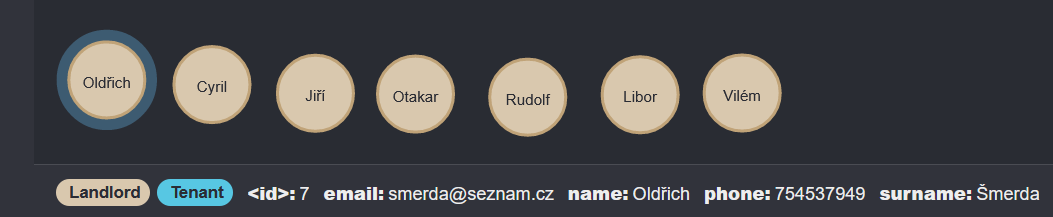
## Schéma databáze

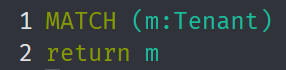


### Úkol 1 – Seznámení s daty

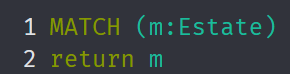
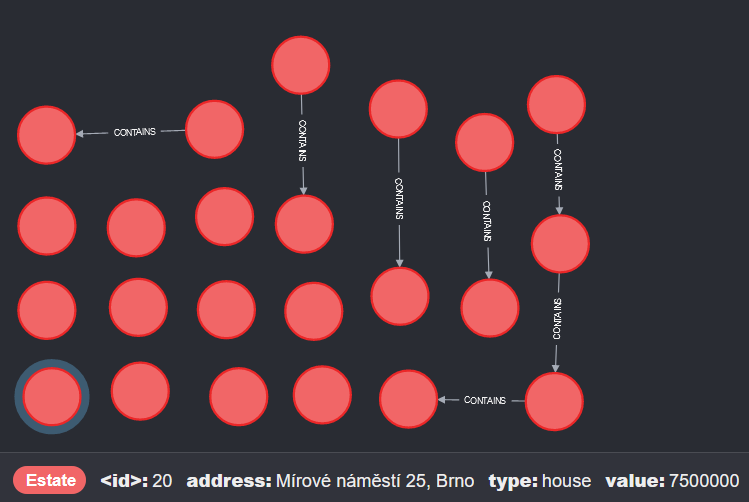
Postupně vypište všechny banky, pronajímatele, nájemníky a nemovitosti. Podívejte se, jaké mají data a poté najděte pronajímatele s telefoním číslem 745082420.

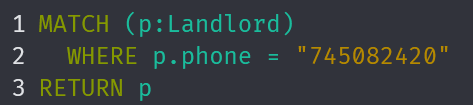
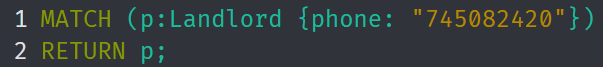
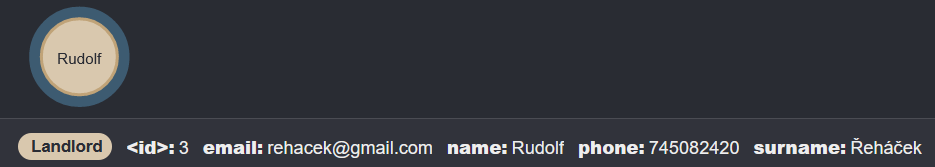
Banky:  
  


Pronajímatelé:  
  


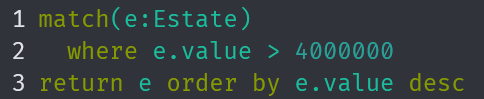
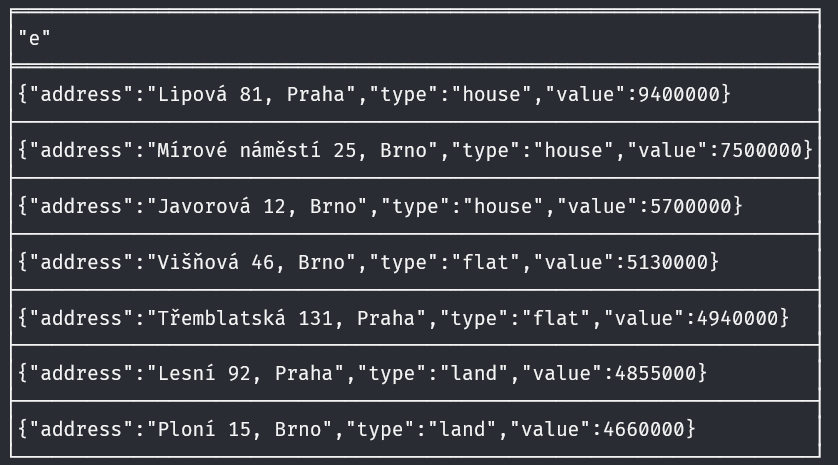
Nájemníci:  
  


V databázi máme 3 osoby (Oldřich, Otakar, Cyril), kteří jsou pronajímateli i nájemníky.

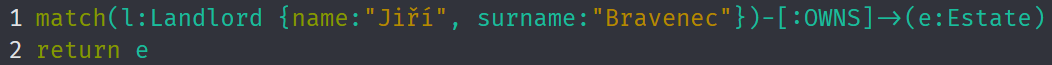
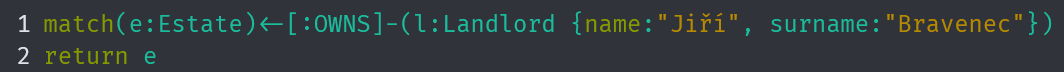
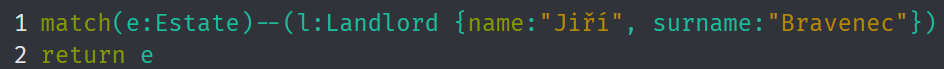
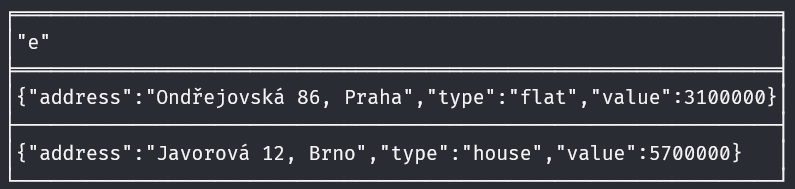
Nemovitosti:  
  
  
V databázi máme 4 nemovitosti (byty), které pod sebou mají další nemovitost (garáž) a jednu supernemovitost (byt + pozemek + garáž + sklep).

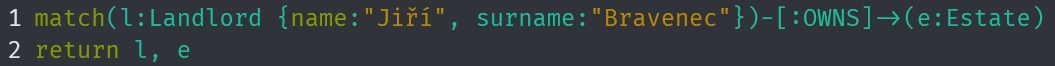
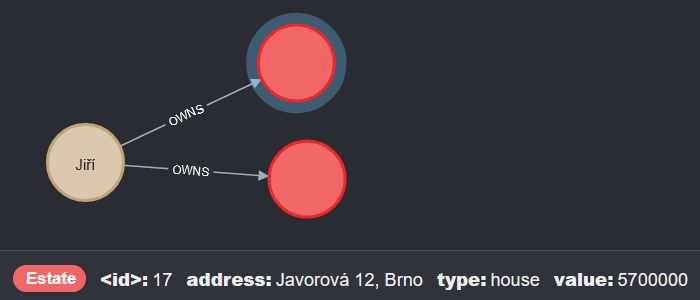
Pronajímatel s telefoním číslem 745082420:  
  
nebo  


### Úkol 2 – Najděte nemovitosti dražší než 4 miliony a seřaďte je podle ceny od největší

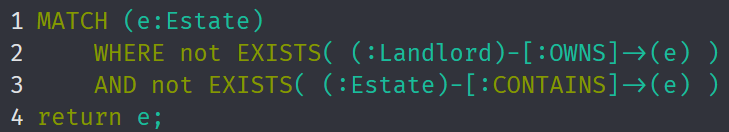
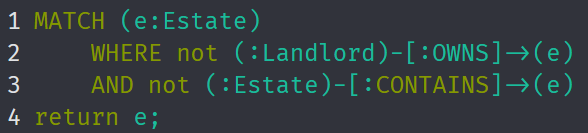
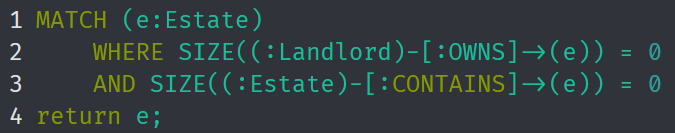
  


### Úkol 3 – Najděte byty, které vlastní Jiří Bravenec

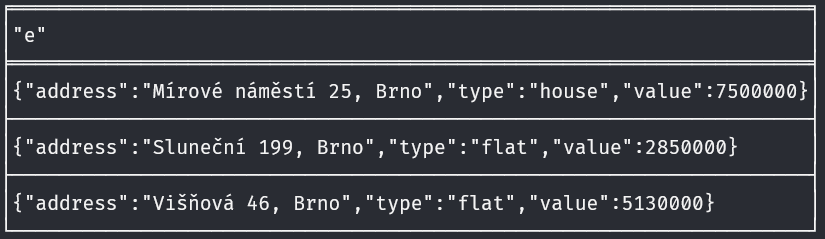
  
nebo  
  
nebo  
  


Grafově:  
  
nebo  
  


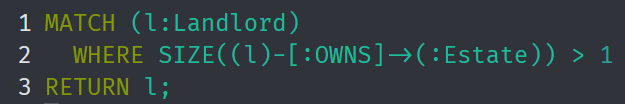
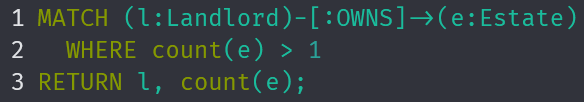
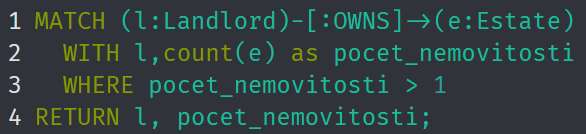
### Úkol 4 – Najděte byty, které nikdo nevlastní (neznáme majitele), podnemovitosti neuvažujte

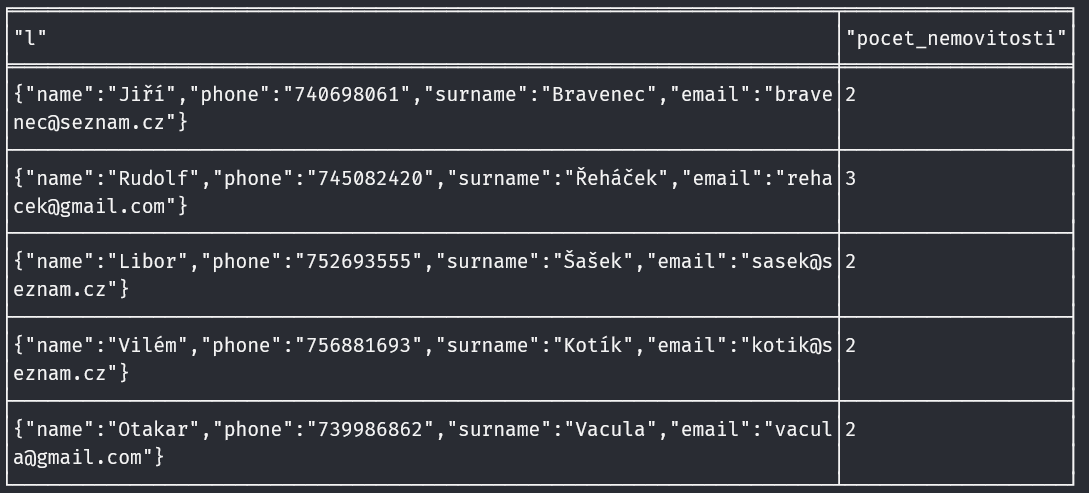
  
nebo  
  
nebo  


Podmínka AND … filtruje pryč podnemovitosti, které nemáme uvažovat. Pokud bychom je chtěli uvažovat, stačilo by k returnu přidat vypsání všech podnemovitostí (tím by se vypsali jen podnemovitosti nevlastněné nemovitosti). V Mongu by to už ale bylo horší.

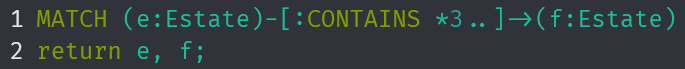
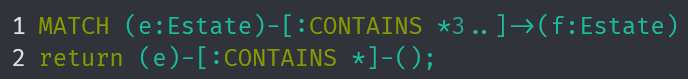
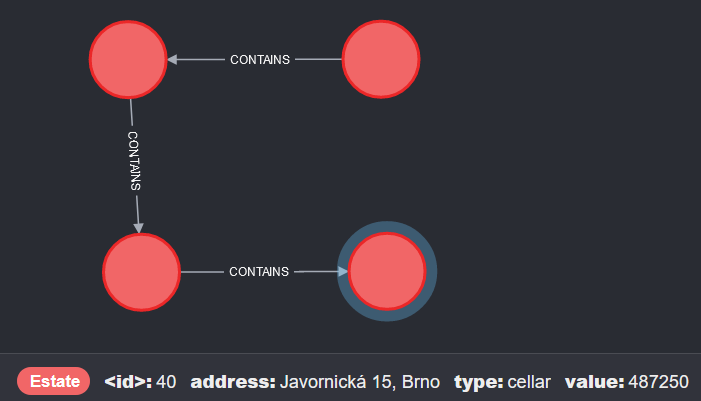


### Úkol 5 – Najděte pronajímatele, kteří vlastní alespoň 1 byt

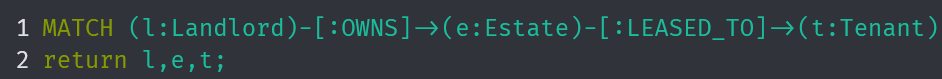
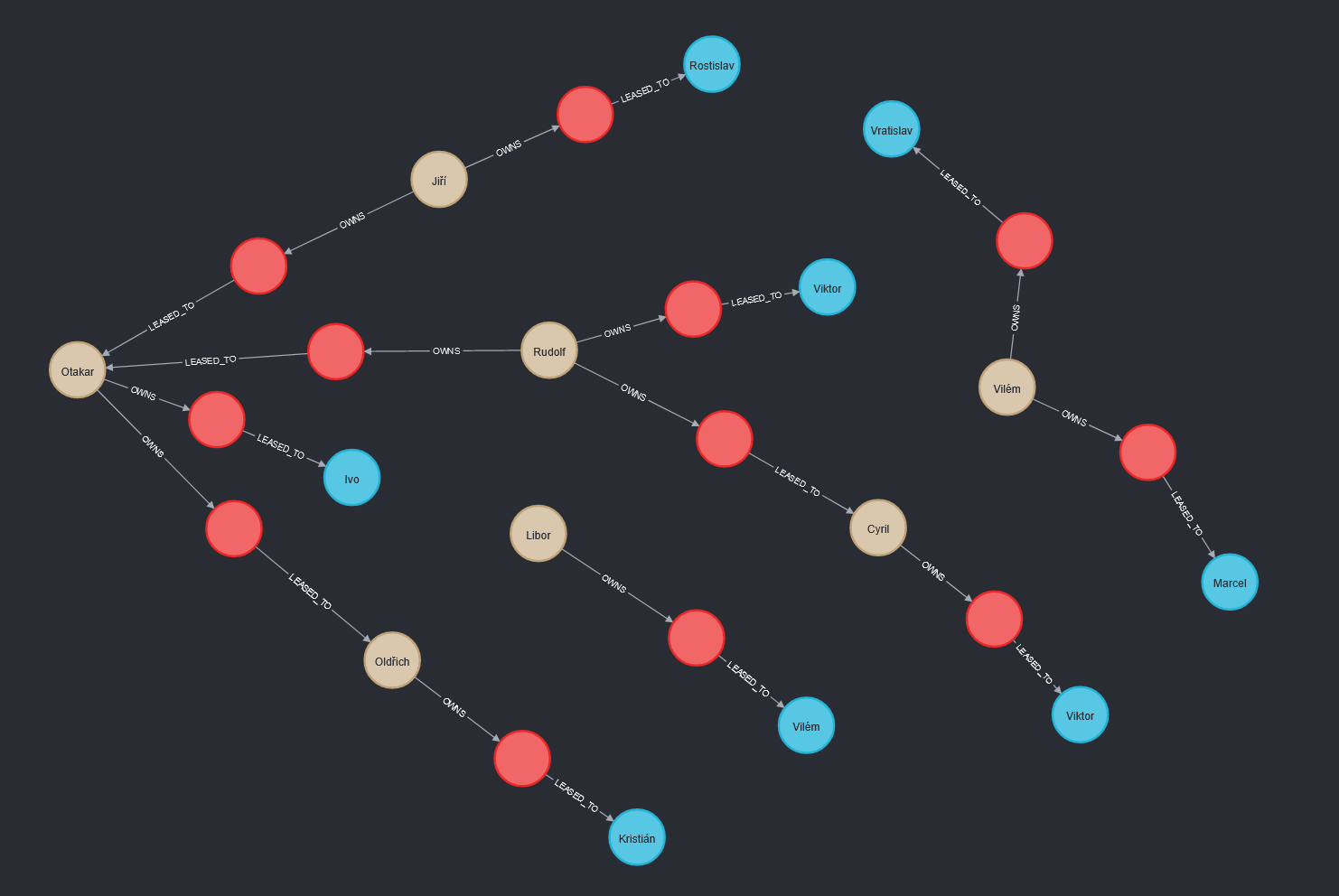
  
Kdybychom ovšem chtěli vypsat i konkrétní počet, údaj už nemáme, co zkusit count:  
  
Pokud to tak zkusíme, dostaneme chybu, že agregační funkci takto použít nelze. Musíme použít klauzuli WITH, a výsledek dotazu „poslat“ do dalšího dotazu, kde ji vyfiltrujeme přes WHERE:  




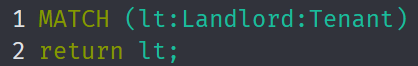
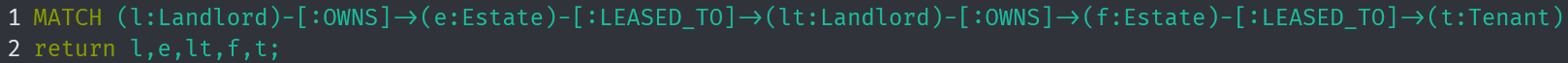
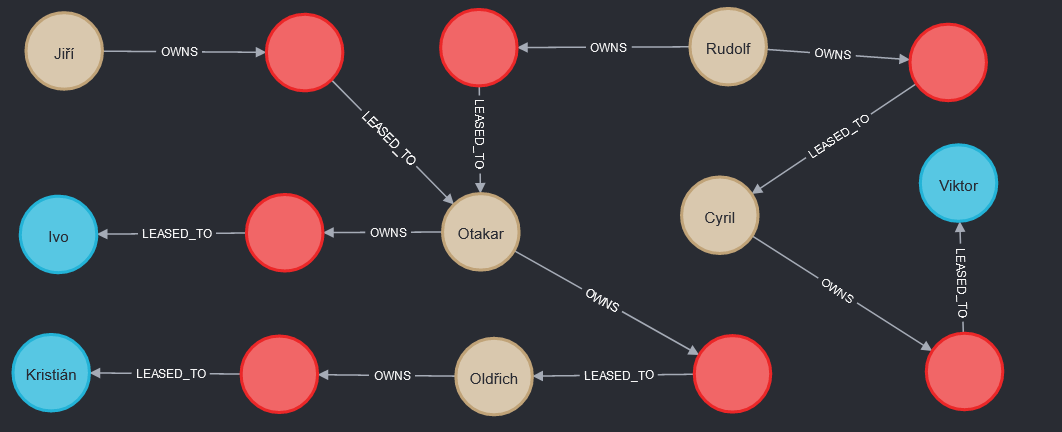
### Úkol 6 – Najděte všechny supernemovitosti (nemovitosti, které pod sebou mají další nemovitosti a ty mají další a tak dále hloubky 4 a více) např. dům->pozemek->garáž->sklep a zobrazte celou tuto nemovitost jako graf

Nemovitosti hloubky 4 a více mají 3 a více „CONTAINS“ uzlů, můžeme proto napsat:  
  
Tím ovšem dostaneme pouze krajní hrany. Navíc nemovitost může být „strom“ a mít další podnemovitosti menší hloubky, které chceme vypsat. Celou nemovitost tedy vypíšeme jako:  
  


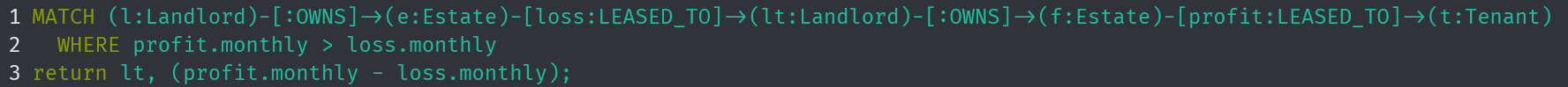
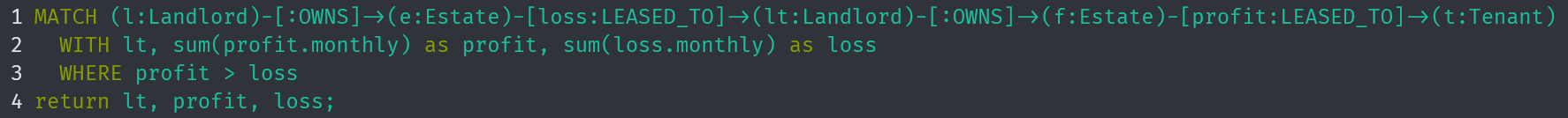
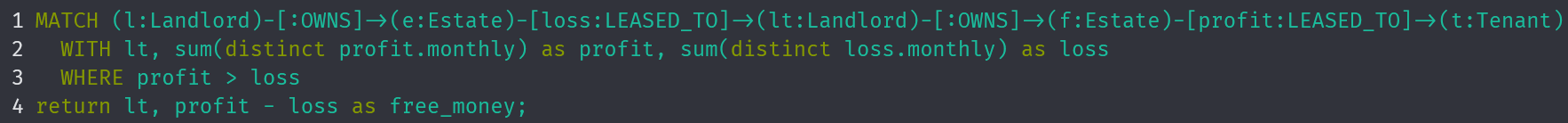
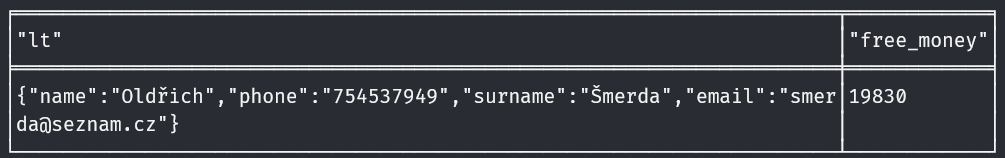
### Úkol 7 – Vypište vztahy mezi pronajímateli a nájemníky do grafu

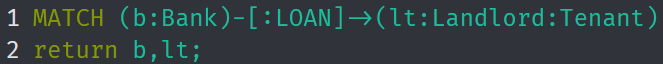
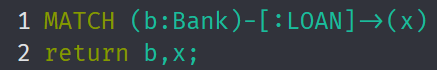
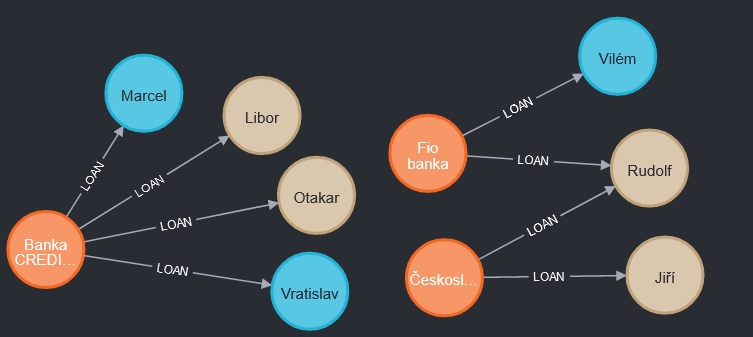
### Úkol 8 – Vypište takové lidi, kteří zároveň vlastní nemovitost a jsou nájemníky v jiné

Pokud by stačilo pouze vypsat osoby, stačí velmi jednoduchý:  
  
Kdybychom chtěli i informace o tom, kdo je pronajal a komu osoba pronajímá, můžeme využít například:  
  


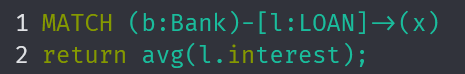
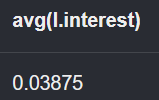
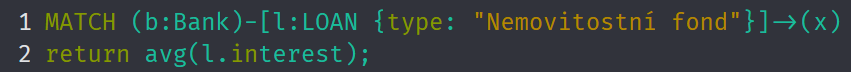
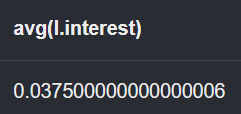
### Úkol 9 – Vypište lidi, kteří profitují na tom, že jsou nájemníky někde jinde a vypište jejich profit, daně zanedbejte

Zajímá nás tedy rozdíl v příjmech z pronájmu vs ztráty z nájmu, první nástřel:  
  
Je špatně, protože jeden člověk může vlastnit i bydlet ve více nemovitostech, tohle by fungovalo pouze pokud by mohl vlastnit maximálně jednu. Zkusíme tedy využít sumu:  
  
V tomto případě je to také špatně, protože se cesty překrývají a profity i ztráty se započítávají vícekrát. Musíme tedy využít distinct:  
  


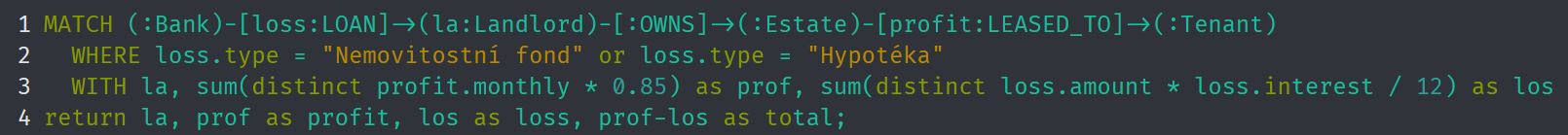
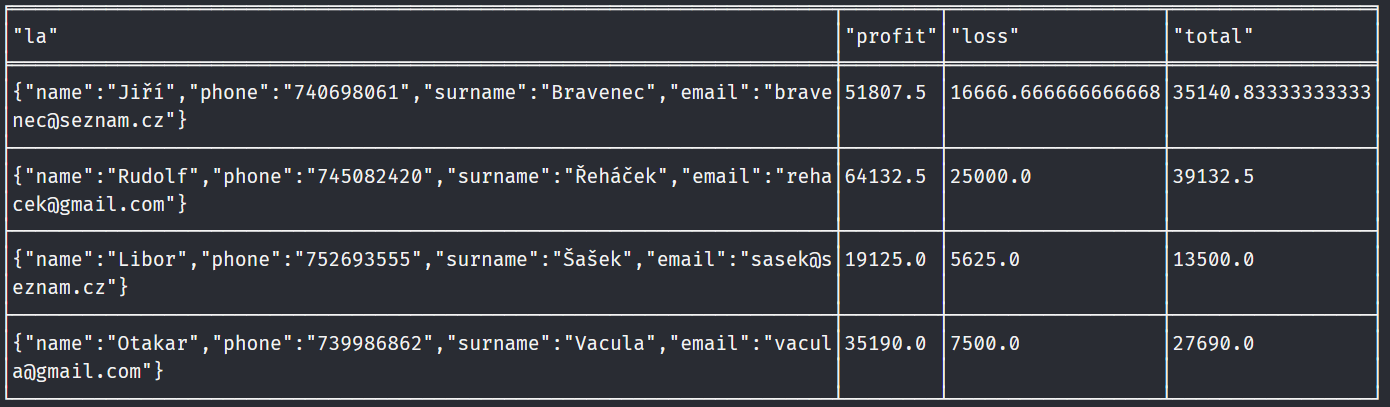
### Úkol 10 – Zobrazte v grafu všechny půjčky

Půjčky si můžou vzít jak nájemníci, tak pronajímatelé, můžeme zkusit:  
  
Takto ovšem nedostaneme „Landlord or Tenant“, ale „Landlord and Tenant“, jelikož chceme všechny, můžeme použít obecné x:  
  


### Úkol 11 – Zjistěte průměrný úrok u všech půjček a u nemovitostních fondů

Obecně:  
  
  
Fond (drobné problémy s doublem):  
  


### Úkol 12 – Zjistěte, jestli pronajímatelé profitují na svých půjčkách, výdaje zanedbejte, daň z příjmu předpokládejte 15% bez odpisů

Budeme postupovat podobně jako u zjišťování profitů nájemník – pronajímatel, akorát musíme udělat trochu jiné výpočty:  
  
Zajímají nás tedy půjčky z fondu nebo hypotéka, a potřebujeme zjistit profit, což je součet všech pronajatých bytů (bez duplicit) \* 0.85 což nám strhne 15%. Ztrátu vypočítáme jako celkovou výši půjčky \* úrok / 12, protože úroky jsou ročně. Pak už vše stačí jen vypsat.  


# Cvičení MongoDB

## Pracovní databáze

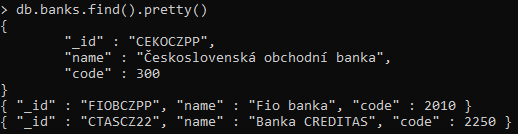
Pro srovnání MongoDB a Neo4j zůstává databáze téměř stejná (snažil jsem se ji co nejlépe převést do JSONu, jak si myslím, že by to mělo být). Převedení z Neo4j do MongoDB je následující: Uzly (Bank, Tenant, Landlord, Estate) jsou kolekce „entit“. Jelikož v Neo4j může být uzel Tenant is Landlord zároveň, rozhodl jsem se udělat kolekci People, která má vždy atribut type, což je pole stringů, které určuje, jestli je Person „tenant“ nebo „landlord“ nebo obojí. V Neo4j jsme měli také několik „relací“ (hran). Hrana LOAN odpovídá many-to-many vazbě a proto je v MongoDB řešena jako samostatná kolekce, která má vždy id banky a id osoby + ostatní vlastnosti, které byly obsaženy v hraně. Hrany OWNS, LEASED\_TO a CONTAINS odpovídají one-to-many (byt patří vždy pouze jedné osobě a může ho pronajímat vždy jen jedna osoba, nemovitost patří vždy pouze pod jednu nemovitos [ale nemovitost má více podnemovitostí]) a proto je tohle řešené pomocí pole, kde každá položka je vazba (má id na osobu / nemovitost a další atributy dle potřeby hrany). Databáze byla dělaná pro Neo4j a proto některé úlohy mají poměrně krkolomné řešení, na které se Mongo nehodí (nebo jsem nepřišel na jednodušší řešení).

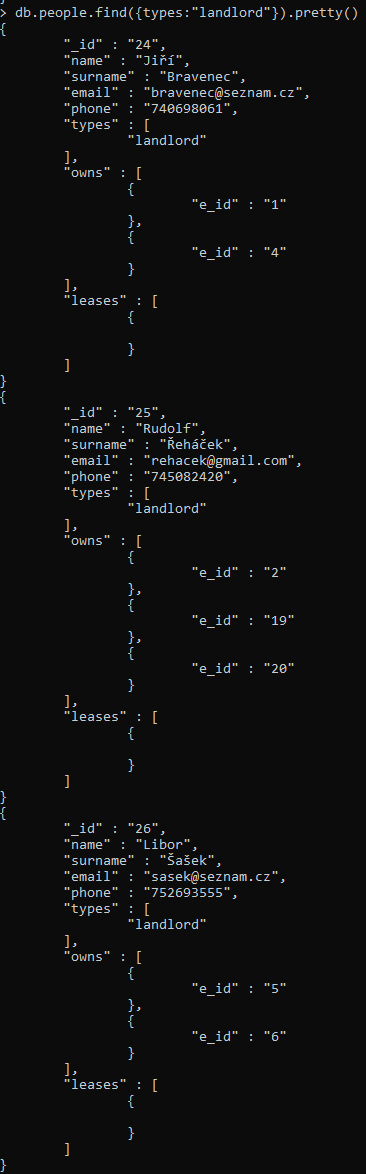
Create script je v souboru: mongo.txt

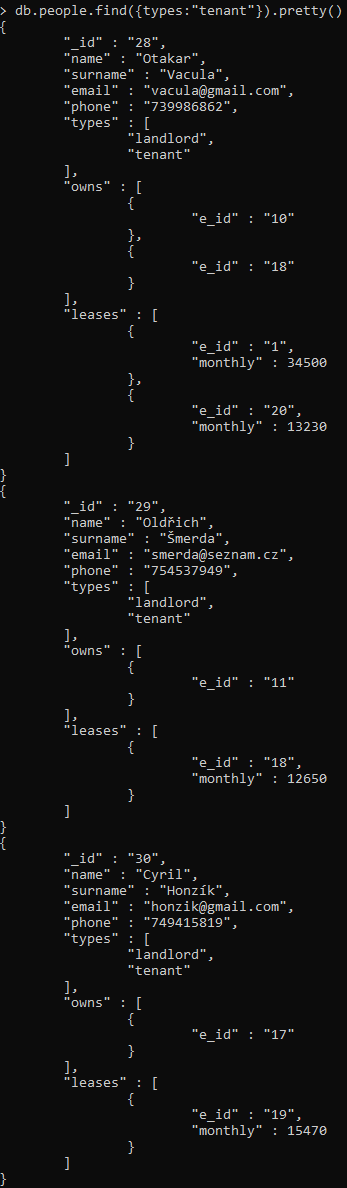
## Schéma databáze („entity“) neboli jak by to vypadalo v aplikaci

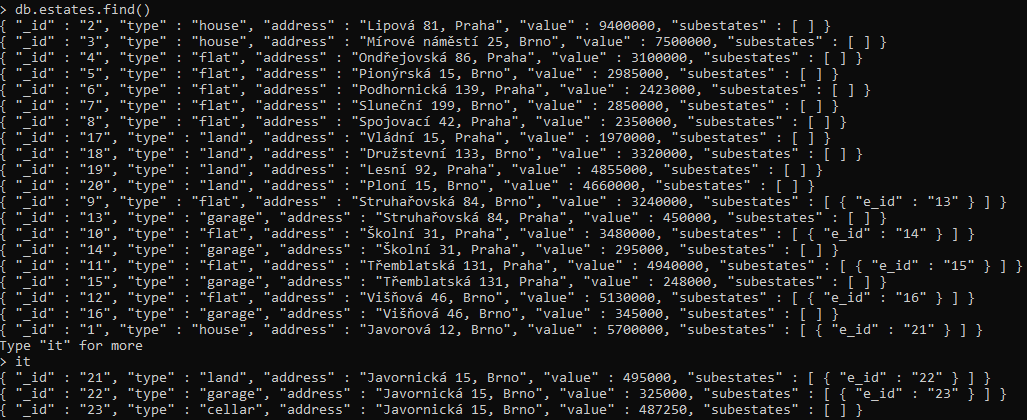
### Úkol 1 – Seznámení s daty

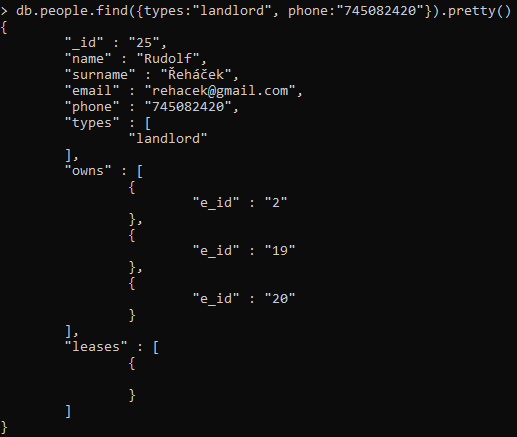
Postupně vypište všechny banky, pronajímatele, nájemníky a nemovitosti. Podívejte se, jaké mají data a poté najděte pronajímatele s telefoním číslem 745082420.

Banky:  


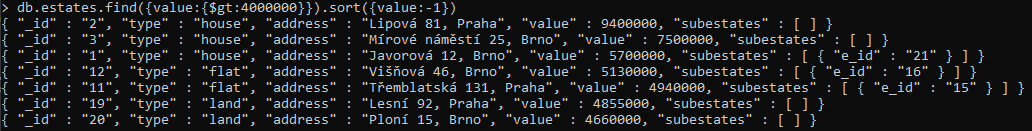
Pronajímatelé:  
  

Nájemníci:  
    
V databázi máme 3 osoby (Oldřich, Otakar, Cyril), kteří jsou pronajímateli i nájemníky.

Nemovitosti:  
  
V databázi máme 4 nemovitosti (byty), které pod sebou mají další nemovitost (garáž) a jednu supernemovitost (byt + pozemek + garáž + sklep).

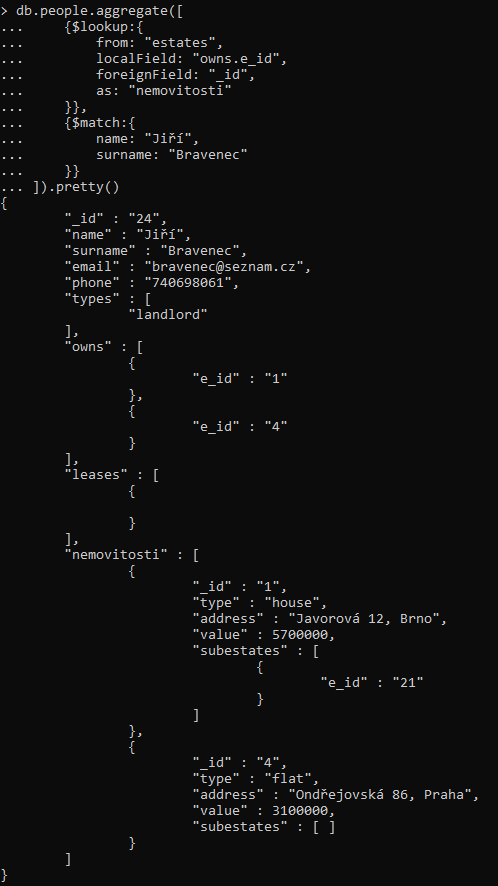
Pronajímatel s telefoním číslem 745082420:  


### Úkol 2 – Najděte nemovitosti dražší než 4 miliony a seřaďte je podle ceny od největší



### Úkol 3 – Najděte byty, které vlastní Jiří Bravenec

Jelikož Person a Estates mají one-to-many vazbu, musíme udělat něco jako join, tedy vytahat si z Estates všechny nemovitosti, které se shodují s e\_id v owns[]. Pak už stačí jednoduchý selekt a projekce.  

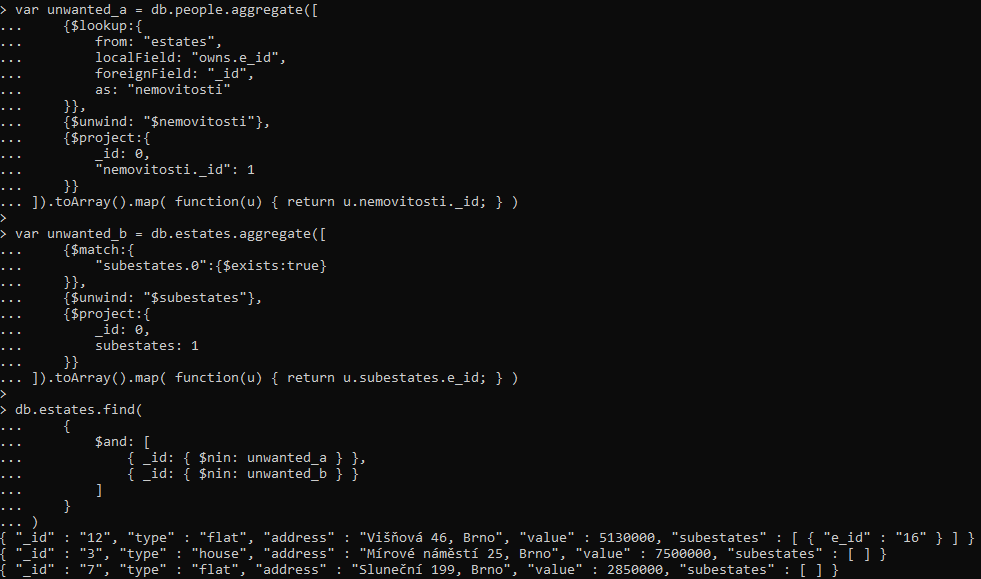

Pokud neuděláme projekci, dostaneme data i s informacemi o vlastníkovi:  


### Úkol 4 – Najděte nemovitosti, které nikdo nevlastní (neznáme majitele), podnemovitosti neuvažujte

Uděláme si proměnou unwanted\_a, do které dáme id všech nemovitostí, které někdo vlastní. V agregaci používáme unwind, který z pole nemovistostí udělá jednotlivé hodnoty a ostatní nakopíruje. Pokud bychom měli třeba dvojici (\_id: 1, nemovitosti: [5, 6]), unwind na nemovitost by udělal dvě samostatné dvojice (1, 5), (1, 6) a díky tomu k nim jde jednoduše přistupovat přes tečkovou notaci.

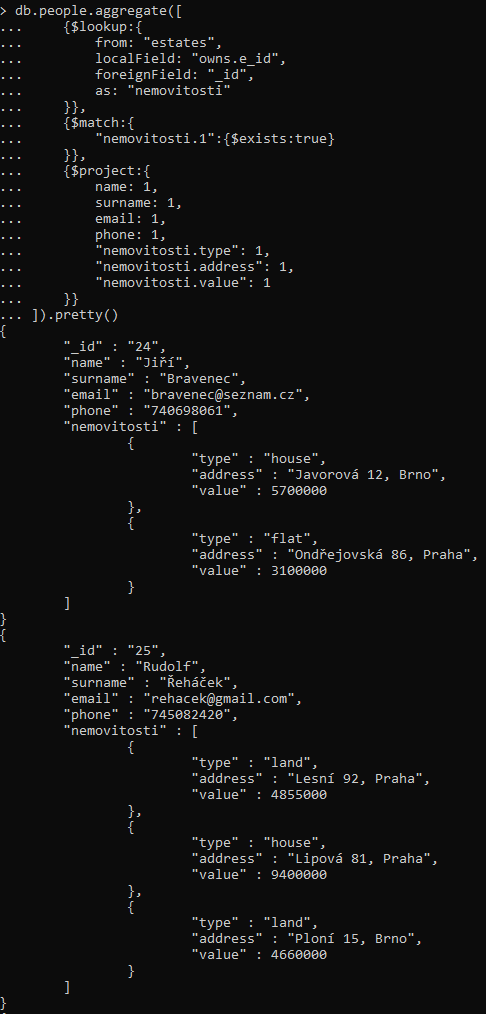
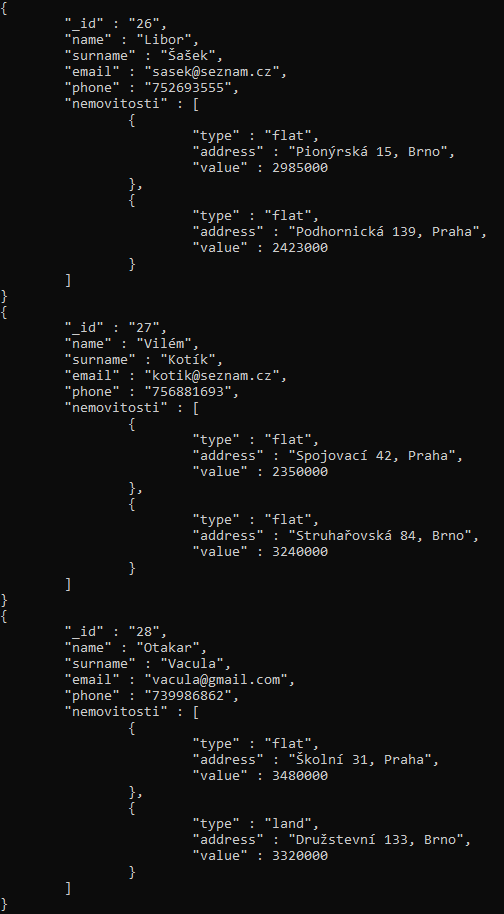
Poté uděláme v podstatě to samé pro unwanted\_b, akorát tam přijdou id všech podnemovitostí. Vezmeme tedy každou nemovitost, podíváme se, jestli má nějakou podnemovitost a pokud ano, přidáme ji do výsledku.

Na konec stačí projet kolekci estates a vyfiltrovat všechny nemovitosti, které mají id v unwanted\_a nebo unwanted\_b a zbydou nám pouze ty, které nikdo nevlastní.

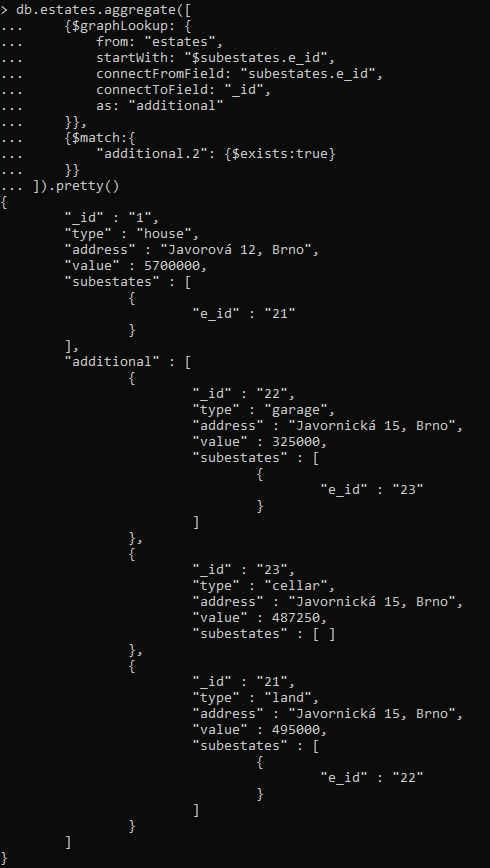


### Úkol 5 – Najděte pronajímatele, kteří vlastní více než 1 byt

Uděláme opět join lidí s vlastněnými nemovitostmi. V tomto joinu budeme mít pole nemovitostí. Více než jeden je v podstatě 2 a více, a jelikož se indexuje od nuly, pokud na indexu 1 v poli nemovitostí existuje hodnota, má pronajímatel alespoň 2 byty. Pak dle libosti vyfiltrujeme jen zajímavé parametry.

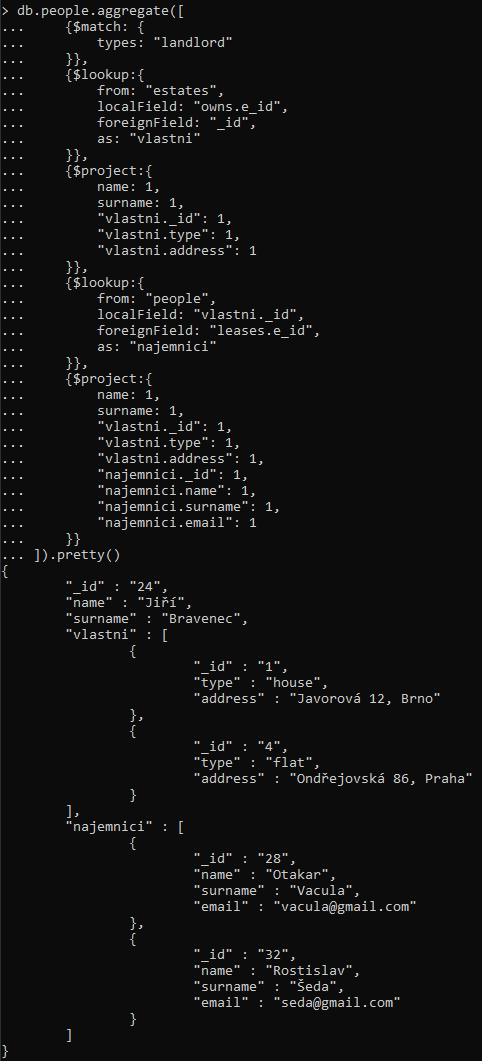
 

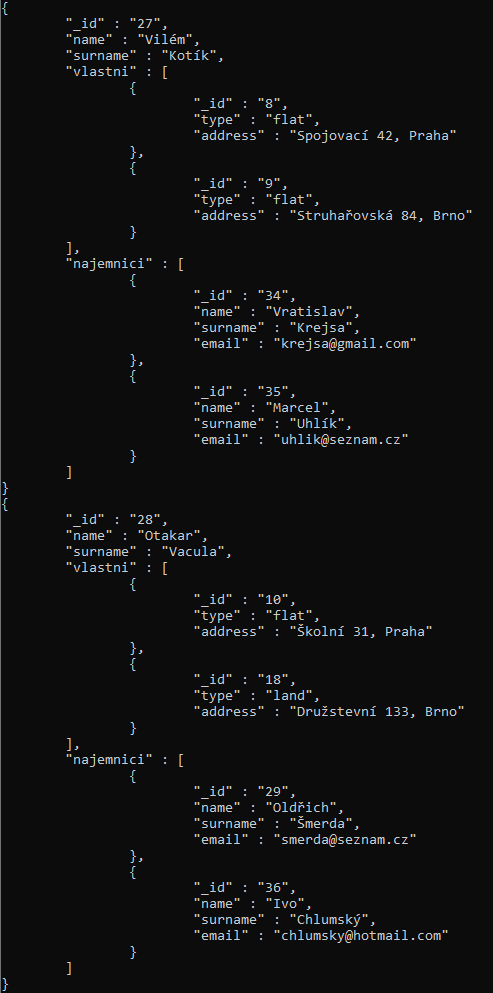
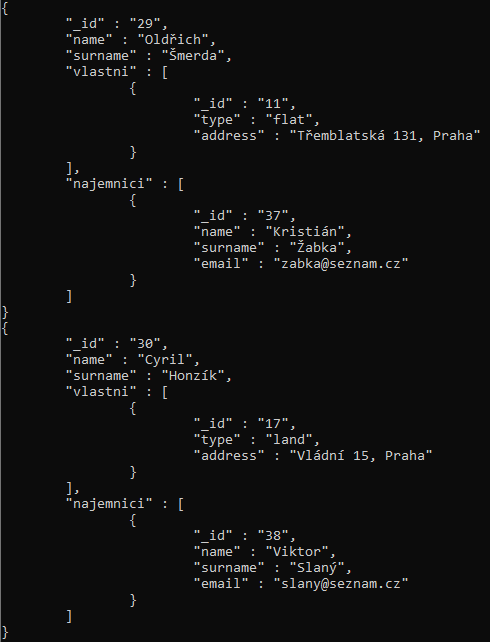
### Úkol 6 – Najděte všechny supernemovitosti (nemovitosti, které pod sebou mají další nemovitosti a ty mají další a tak dále hloubky 4 a více) např. dům->pozemek->garáž->sklep a zobrazte celou tuto nemovitost

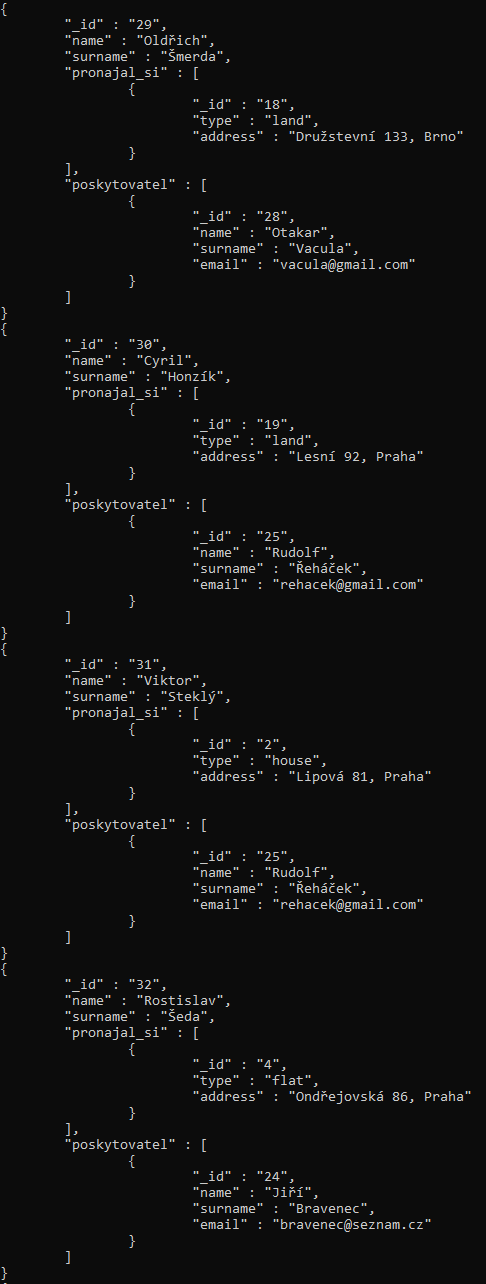
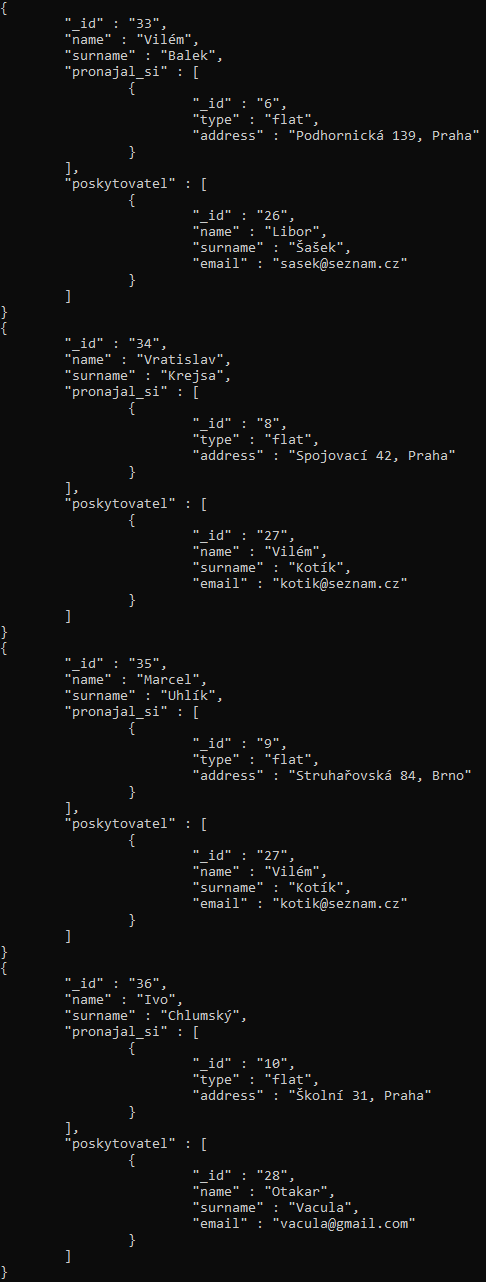
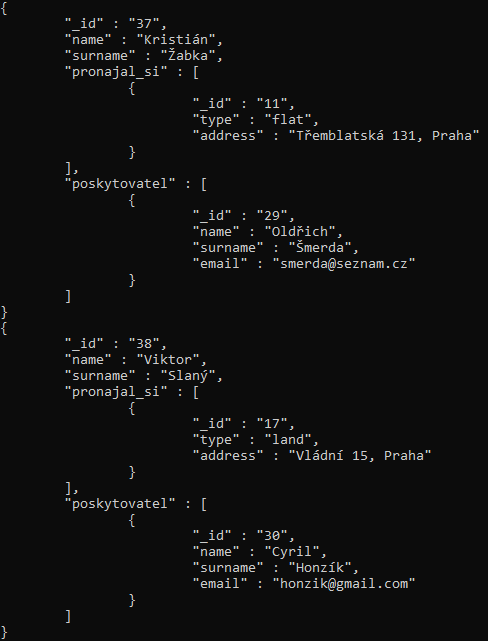
Typická úloha pro grafy a Neo4j, jak se s tím popere Mongo? Mongo má kupodivu funkci graphLookup, která umí rekurzivně procházet „entity“ a přidávat všechny, které projde do pole. Stačí tedy vhodně napsat, jak má průchod vypadat. Procházíme kolekci estates a mapujeme vždy subestates.e\_id na \_id v estates, výsledek dáme do additional. Takto hezky rekurzivně projdeme vše (pozor, mongo má na tohle dle dokumentace limit cca 100 MB paměti).

### Úkol 7 – Vypište vztahy mezi pronajímateli a nájemníky

Musíme udělat 2x join, poprvé z People na Estates (person owns estate) a poté znovu z People na Estates (person rents owned estate). V prvním joinu tedy spojíme osoby s vlastněnými byty a v druhém pak všechny tyto data spojíme znovu s osobami přes vlastněnou nemovitost na pronájem nemovitosti. Pak už stačí vyfiltrovat vše co chceme vypsat, aby toho nebylo moc. Tímto dostaneme vztahy pronajímatelů s nájemníky, když chceme opak, stačí joiny vhodně otočit.

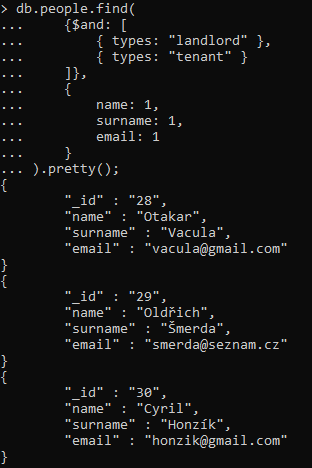
Vztahy pronajímatelů s nájemníky:  
 

Vztahy nájemníků s pronajímateli:  
   

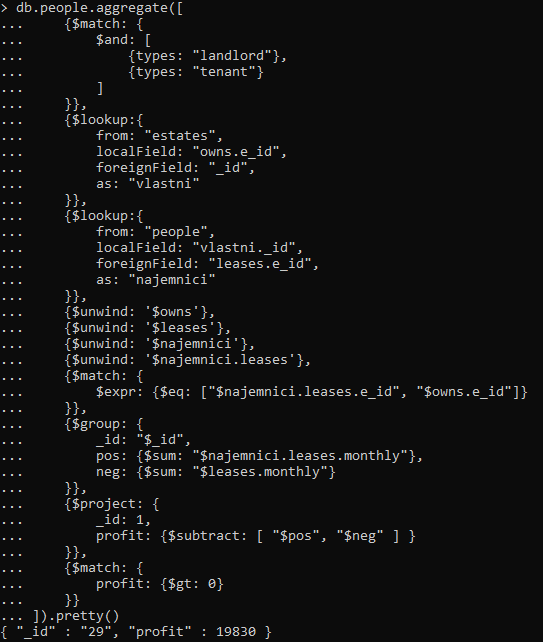
### Úkol 8 – Vypište takové lidi, kteří zároveň vlastní nemovitost a jsou nájemníky v jiné

Jednoduchá verze, stačí se podívat že Person je jak landlord tak tenant. Kdybychom tuto informaci neměli, museli by jsme opět joinovat.



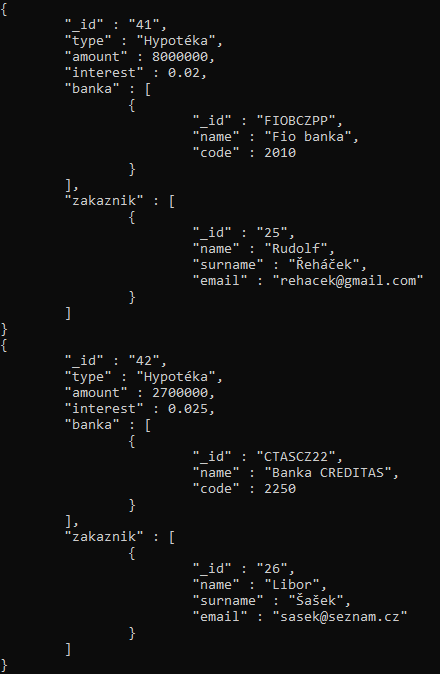
### Úkol 9 – Vypište lidi, kteří profitují na tom, že jsou nájemníky někde jinde a vypište jejich profit, daně zanedbejte

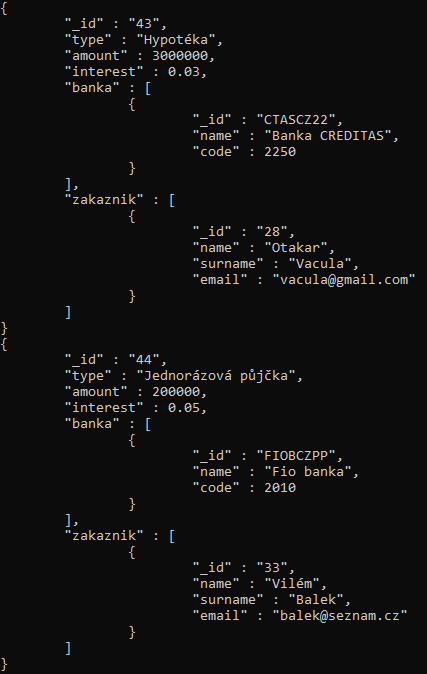
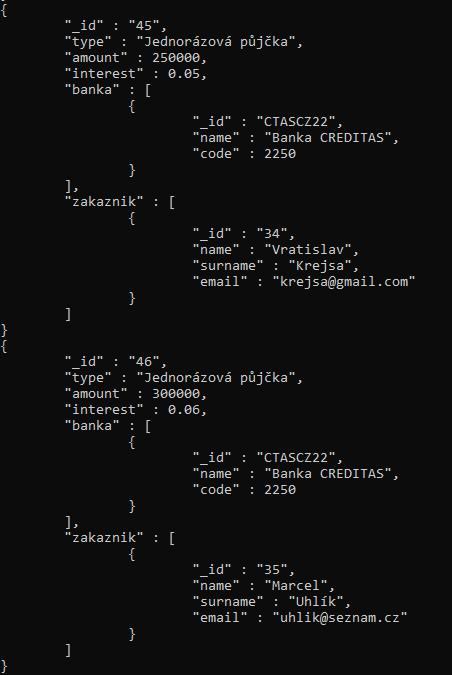
Chceme tedy Person, který je landlord i tenant. Pak si k němu joinem přidáme vlastněné byty a nájemníky těchto bytů. Unwindem zajistíme, že dostaneme něco jako kartézský součin, z kterého pak vyfiltrujeme pouze ty hodnoty, kde nájemník opravdu platí za vlastněný byt (protože nájemník může mít pronajatých více bytů od více osob, vyfiltrujeme to pouze na hodnoty, kde se id vlastněného bytu rovná id pronajatého bytu). Tyto zbylé hodnoty pak opět sjednotíme podle id vlastníka a sečteme měsiční příjmy a ztráty. Na konec uděláme rozdíl a vyfiltrujem jen ty, které mají profit. Pokud bychom chtěli jméno a příjmení, kromě id, museli bychom ještě joinout s Person (pravděpodobně to jde i jinak, ale nepodařilo se mi přes $group propasovat další data).



### Úkol 10 – Zobrazte všechny půjčky

Jelikož půjčky jsou řešeny přes many-to-many vazbu s Bank a Person. Musíme si k půjčce najoinovat obě data a pak vyfiltrovat, co chceme vypsat.

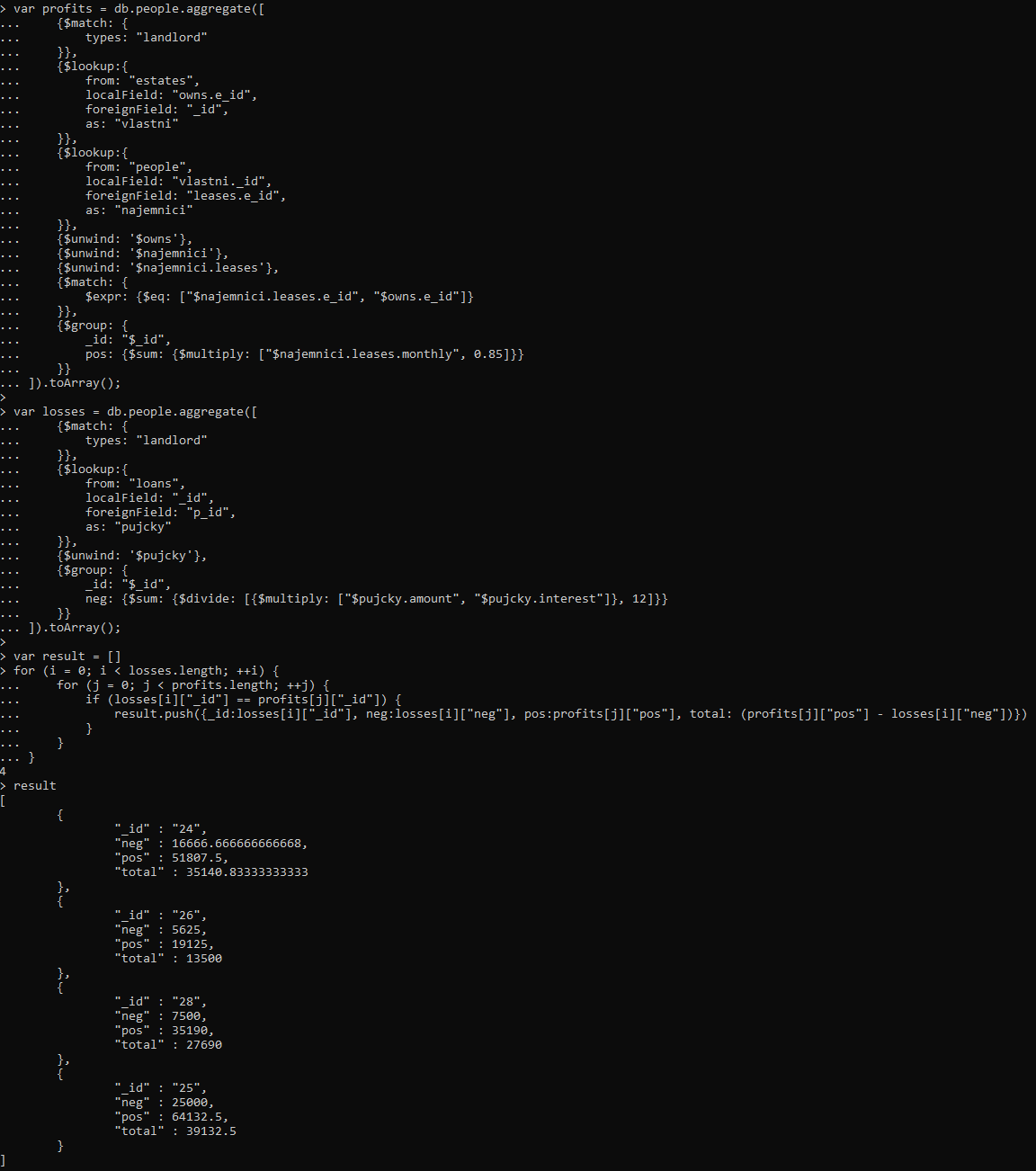
 

### Úkol 11 – Zjistěte průměrný úrok u všech půjček a u nemovitostních fondů

Stačí projet všechny půjčky (group by null) a zavolat průměrovací funkci na úrok.

### 

### Úkol 12 – Zjistěte, jestli pronajímatelé profitují na svých půjčkách, výdaje zanedbejte, daň z příjmu předpokládejte 15% bez odpisů

Musíme si úlohu rozdělit na dvě části, protože kartézský součin přes tolik věcí by produkoval duplicitní data (museli bychom vymyslet chytřejší group by než přes id). Rozdělíme si úlohu na profits a losses. Profits už máme z předchozí úlohy, jenom se trochu upraví a přidá se násobení 0.85 kvůli dani. Ztráty jsou jednodušší, stačí vzít všechny vlastníky nějaké nemovitosti, najoinovat si k nim jejich půjčky a unwindnout je. Pak stačí obyčejný group by podle id (akorát násobíme úrokem a dělíme 12 měsíci). Na konec by asi šlo vymyslet něco chytřejšího, ale pro ukázku co vše v Mongu jde jsem to umlátil JavaScriptem.  


# Závěr

Vytvořil jsem databázi pro Neo4j tak, aby šla rozumně převést do MongoDB. Všechny dotazy (kromě „ukažte jako graf“) jdou udělat jak v Neo4j, tak v MongoDB. Tím že byly dotazy vymýšlené pro Neo4j a tím, že MongoDB moc není dělané na časté joiny a obecně má jiný přístup, tak bylo psaní dotazů pro Mongo o dost náročnější. Práce ukazuje, že je velmi důležité zvolit pro daný problém vhodnou databázi. Obecně se mi pracovalo lépe s Neo4j, ne kvůli typu dotazů, ale kvůli syntaxi. JavaScript syntaxe má strašně symbolů a závorek a byla pro mě méně přehlednější. Na druhou stranu mi připadlo, že MongoDB dotazy šlo psát celkem mechanicky, kdežto u Neo4j jsem nad tím musel přemýšlet.