REPORT LETO 2021 - MATÚŠ HEDERA

Označenie
Budem používať označenie *Horný Index* a *Dolný Index*, zavedené v reporte zo zimného semestra.

Nech $predicates(q)$ je množina všetkých predikátov, od ktorých závisí predikát q. Im zodpovedajúce vrcholy v grafe definície budem značiť modrou farbou.

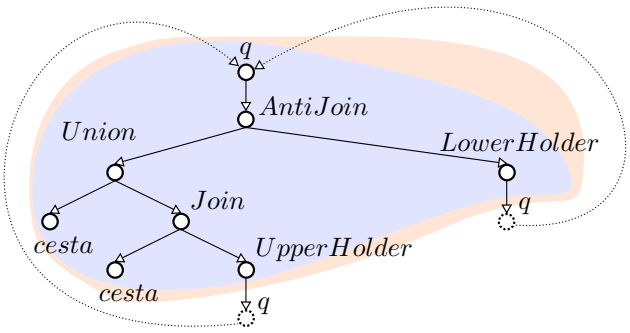
Potom nech $predicates(q, i) := \{p^j | p \in predicates(q) \wedge q^i\}$ (Pre všetky $p \in predicates(q)$ je ich horný index j jednoznačne určený voľbou indexu i - postup je podrobnejšie popísaný v reporte zimného semestra).

Potom nech $predicates(q, i, c) := \{p_c^i | p \in predicates(q)\}$. Im zodpovedajúce vrcholy v grafe definície budem značiť oranžovou farbou.

Cieľ letného semestra
Implementovať rozšírenie pre projekt, ktoré umožní vypočítať rekurzívne dotazy. Na počiatku semestra sme si položili otázku: "Ako reprezentovať rekurzívny dotaz?". Je to súbor pravidiel, podobne ako nerekurzívny dotaz. Aby ale bolo možné realizovať top-down výpočet, pridal som metódy:
$q.instance(i, c)$ - vypočíta predikát q_c^i
$q.getC(i)$ - pomocou opätovných volaní $instance(i, c)$ vypočíta *Dolný Index* uzáveru predikátu q^i
$q.getI()$ - pomocou opätovných volaní $getC(i)$ vypočíta *Horný Index* predikátu q, spĺňajúci podmienku ukončenia

LowerHolder a UpperHolder

Objekty *UpperHolder* a *LowerHolder* slúžia ako pointer na rekurzívny predikát. Je to spôsob, ktorým sa detekuje vnáranie sa do rekurzie pri prechádzaní stromu definície. Prechodom cez *UpperHolder* sa zníži *Dolný Index* - slúži na výpočet uzáveru pomocou naivnej evaluácie. *LowerHolder* sa použije vždy keď q a pointer na q nie sú pod rovnakým horným indexom (to znamená, keď sa prešlo cez *AntiJoin*). Prechod cez *LowerHolder* neznižuje *Horný Index*, pretože uzávery rôznych *Horných Indexov* sú navzájom nezávislé - pod negatívnou stranou *AntiJoin*-u sa znova počíta uzáver pre znížený *Horný Index*. Množina vrcholov pre $predicates(q)$ je oddelená od zvyšku grafu definície vrcholmi typu *UpperHolder* a *LowerHolder* (aby sa nevytvorila nekonečná rekurzívna reťaz). Množina vrcholov pre $predicates(q, i, c)$ je oddelená od zvyšku grafu definície vrcholmi typu *UpperHolder* a *AntiJoin*.



Obr. 1: Definícia (konkrétny program je uvedený nižšie)

Definícia vs. Výpočtový strom
V projekte "Datalog->RA" doteraz triedy implementujúce rozhranie *Operator* poskytovali metódu *next()*. Inak povedané, definície predikátov a ich výpočtové stromy boli rovnaké objekty. S pridaním rekurzívnych predikátov do projektu sa objavuje potreba novej abstrakcie nad týmito objektami. Totiž definícia rekurzívneho predikátu nie je strom, ale graf obsahujúci cykly (nejaký predikát sa v definícii zopakuje). Teraz rozhranie *Operator* poskytuje metódu *instance()* vracajúcu objekt typu *ConcreteOperator*. Rozhranie *ConcreteOperator* poskytuje metódu *next()*, rozhranie *Operator* nie. Tým sa dosiahlo oddelenie definície predikátu od jeho výpočtového stromu. Výhody oddelenia spočívajú v tom, že niektoré vrcholy môžu mať interný stav (napríklad *Union* si pamätá, či má nasledujúci tuple vrátiť z ľavého/pravého syna). Okrem toho pri prechode grafom definície vytvátame na viacerých miestach nové stromy výpočtu (napríklad pod negatívnou stranou *AntiJoin*-u sa znova počíta uzáver pre znížený *Horný Index*).

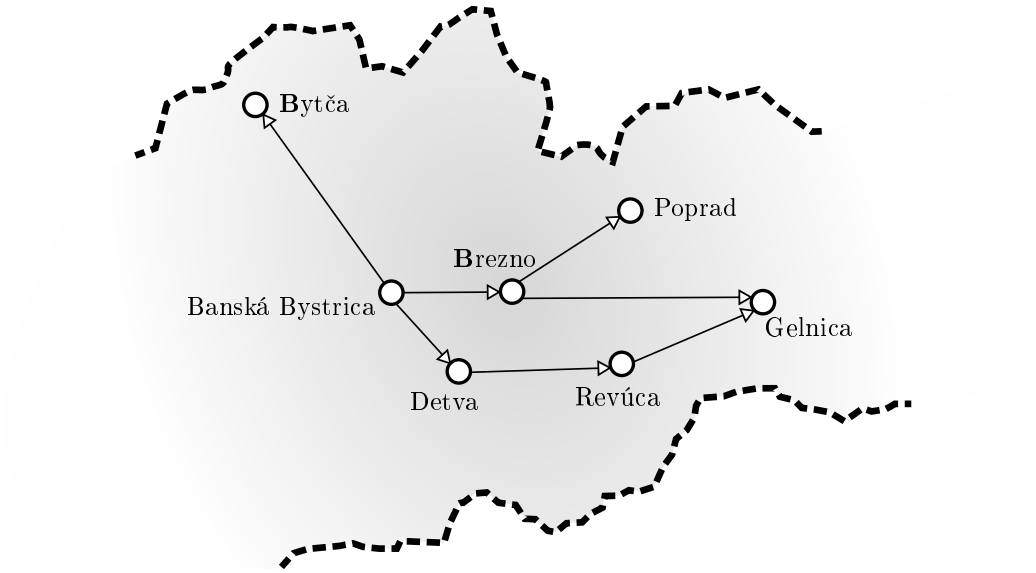
Hľadanie minimálnych indexov

V reporte zimného semestra som spomínal, že o ukončovaní predikátu q nestačí rozhodovať len lokálne, ale je potrebné overiť podmienku ukončenia aj pre všetky predikáty r, od ktorych q závisí - to znamená pre $\forall r \in$ predicates(q) hľadáme *Horný Index* I a jemu zodpovedajúci *Dolný Index* C, pre ktoré platí:

$$
\begin{aligned}
&I = min\{\ i \mid \\
&\quad i >= 3 \\
&\quad \wedge \\
&\quad (\forall\ t:\ t\ \in\ r_C^i\ \Rightarrow\ t\ \in\ r_C^{i-2}) \\
&\quad \wedge \\
&\quad (\forall\ t:\ t\ \in\ r_C^{i-2}\ \Rightarrow\ t\ \in\ r_C^i) \\
&\quad \wedge \\
&\quad C = min\{\ c \mid \\
&\qquad c >= 2 \\
&\qquad \wedge \\
&\qquad [\forall\ p_c^i\ \in\ predicates(r,\ i,\ c)\ \forall\ p_{c-1}^i\ \in\ predicates(r,\ i,\ c-1) \\
&\qquad\quad \forall\ p_c^{i-2}\ \in\ predicates(r,\ i-2,\ c)\ \forall\ p_{c-1}^{i-2}\ \in\ predicates(r,\ i-2,\ c-1)\ : \\
&\qquad\qquad (\forall\ t:\ t\ \in\ p_c^i\ \Rightarrow\ t\ \in\ p_{c-1}^i)\ \wedge\ (\forall\ t:\ t\ \in\ p_c^{i-2}\ \Rightarrow\ t\ \in\ p_{c-1}^{i-2})] \\
&\quad \} \\
&\}
\end{aligned}
$$

Na výstupe výpočtu q sú len tuples patriace do všetkých troch posledných iterácii *Horného indexu* q_C^I, $q_{C_{i-1}}^{I-1}$, q_C^{I-2}. Keďže $q_C^I = q_C^{I-2}$ (podmienka ukončenia), výstup výpočtu q je ekvivalentný $q_{C_{i-1}}^{I-1} \cap q_C^I$.

Príklad



Obr. 2: Mapa databázy

V databáze máme jediný predikát *cesta*(odkiaľ, kam) reprezentujúci orientovaný graf:
cesta("Banská Bystrica", "Brezno").
cesta("Banská Bystrica", "Detva").
cesta("Banská Bystrica", "Bytča").
cesta("Brezno", "Poprad").
cesta("Brezno", "Gelnica").
cesta("Detva", "Revúca").
cesta("Revúca", "Gelnica").
Nech q je množina miest, kam sa vieme dostať z Banskej Bystrice bez toho, aby sme prechádzali cez mestá, ktoré nemajú jedinečné začiatočné písmeno v množine q. Chceme vypočítať množinu q.

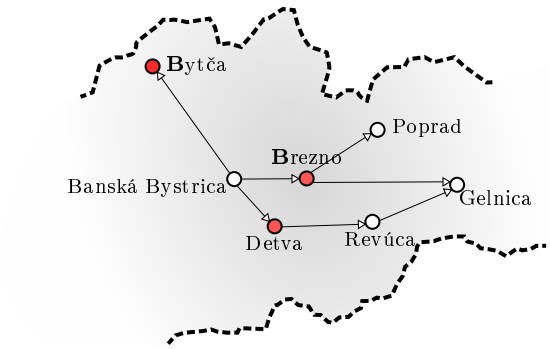
Zaujímavé sú mestá Bytča a Brezno. Do oboch sa z Banskej Bystrice možno dostať, preto ich dočasne umiestnim do množiny q. Hneď ich z nej ale odstránim, pretože sa obe začínajú na písmeno B. V momente, keď ich odstránim sa ale v množine q nenachádza žiadne mesto na písmeno B, takže by som ich mal podľa definície vložiť nazad do q - to je moment, kedy sa vnárame do rekurzie. Okrem toho počítame rekurzívne aj nepriame cesty ako tranzitívny uzáver.

Dotaz q môžeme zapísať v relačnej algebre ako:

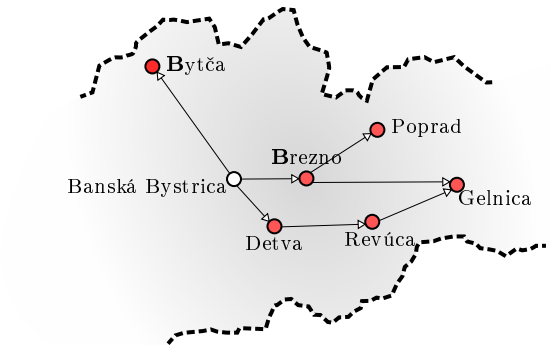
$$\texttt{priame} = \pi_{\texttt{kam}} \ (\ \sigma_{\texttt{odkiaľ=Banská Bystrica}}\ \texttt{cesta}\ )$$
$$\texttt{q} = \rho_{\texttt{q1}}\ [\ \texttt{priame} \cup \pi_{\texttt{cesta.kam}}\ (\ \texttt{q} \bowtie_{\texttt{q.kam=cesta.odkiaľ}}\ \texttt{cesta}\ )\ ]\ \triangleright_{\texttt{q1.kam.charAt(0)=q2.kam.charAt(0)}}\ \rho_{\texttt{q2}}\ \texttt{q}$$

Program začne výpočet q výpočtom uzáveru q^3. Dopracuje sa k q_3^3 = {Bytča, Brezno, Detva, Revúca, Poprad, Gelnica}. Teraz ho chce porovnať s q^1, preto vypočíta uzáver q^1. Dopracuje sa k q_3^1 = {Bytča, Brezno, Detva, Revúca, Poprad, Gelnica}. Keďže $q_3^3 = q_3^1$ (a predikát *cesta* tiež triviálne spĺna podmienku ukončenia, keďže je EDB) výpočet skončí. Vypočíta uzáver q^2, dopracuje sa k q_4^2={Detva, Revúca, Gelnica}. Na výstup pošle $q_3^3 \cap q_4^2$={Detva, Revúca, Gelnica}.

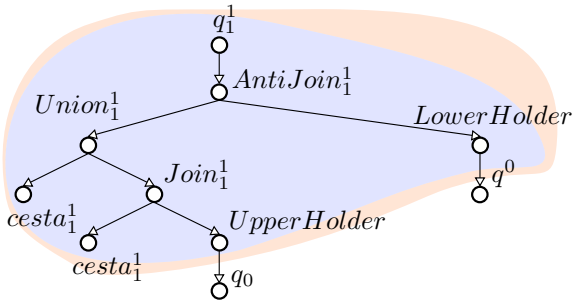
Nižšie som pre vizualizáciu miery opakovania výpočtov rozkreslil niektoré medzivýpočty.



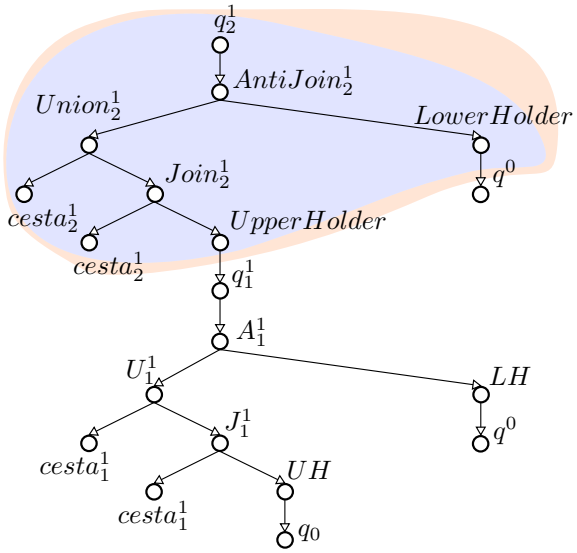
Obr. 3: Mapa q_1^1



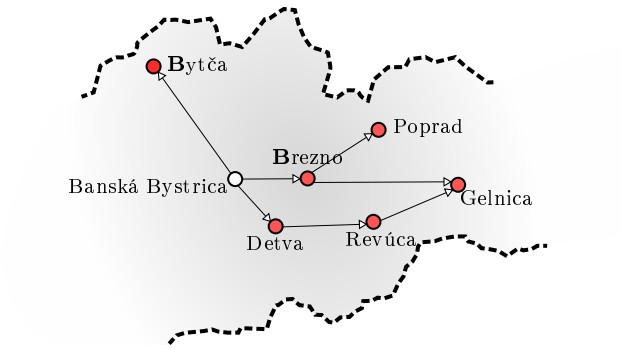
Obr. 4: Mapa q_2^1



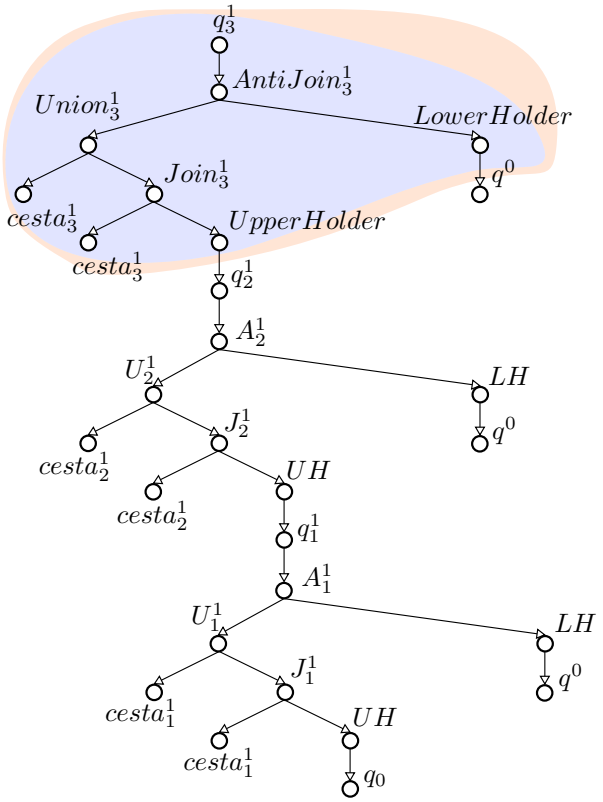
Obr. 5: Mestá do vzdialenosti 1



Obr. 6: Mestá do vzdialenosti 2

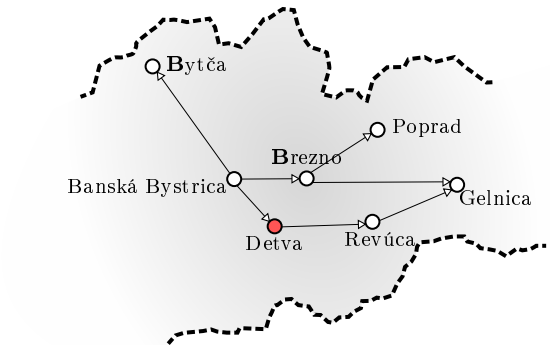


Obr. 7: Mapa q_3^1

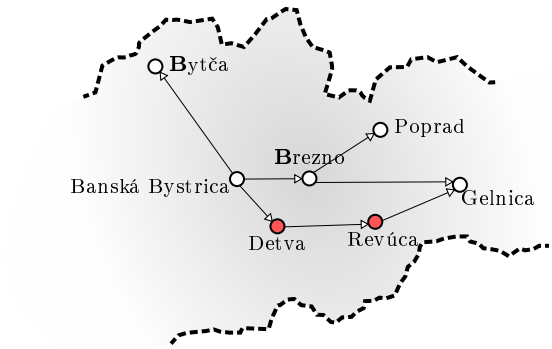


Obr. 8: Mestá do vzdialenosti 3

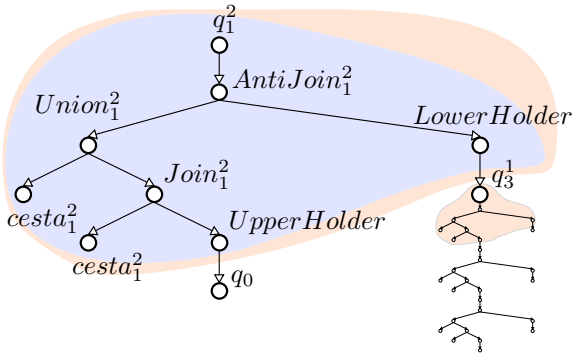
Výpočet q^1 skončí, pretože q_3^1 ={Bytča, Brezno, Detva, Revúca, Poprad, Gelnica} = q_2^1 ={Bytča, Brezno, Detva, Revúca, Poprad, Gelnica}



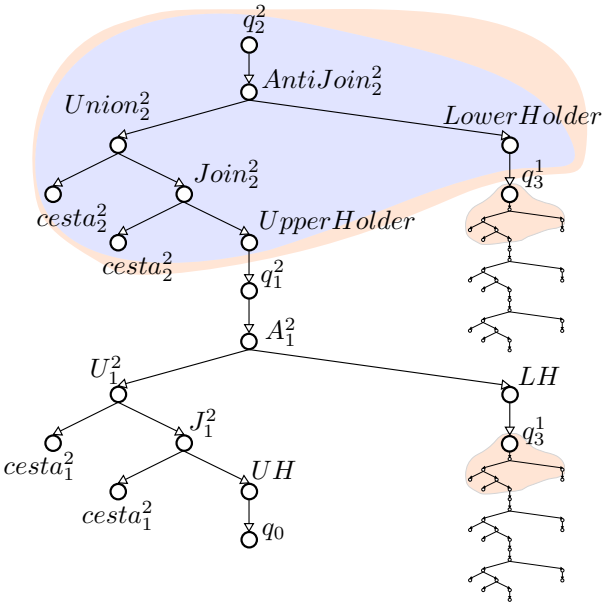
Obr. 9: Mapa q_1^2



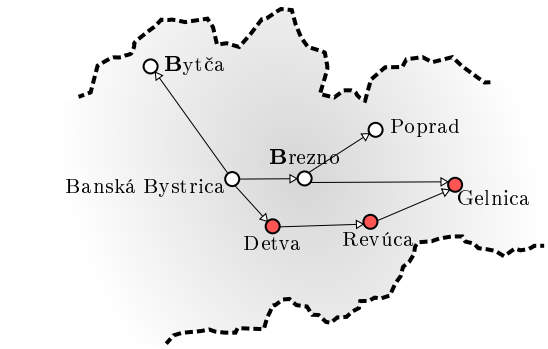
Obr. 10: Mapa q_2^2



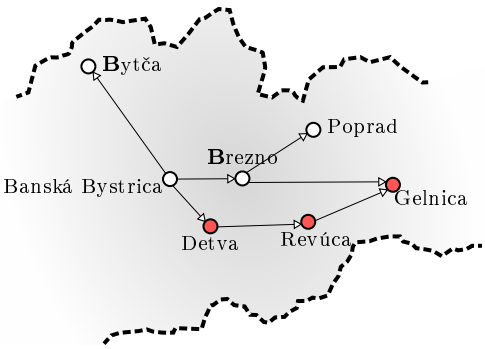
Obr. 11: Mestá do vzdialenosti 1



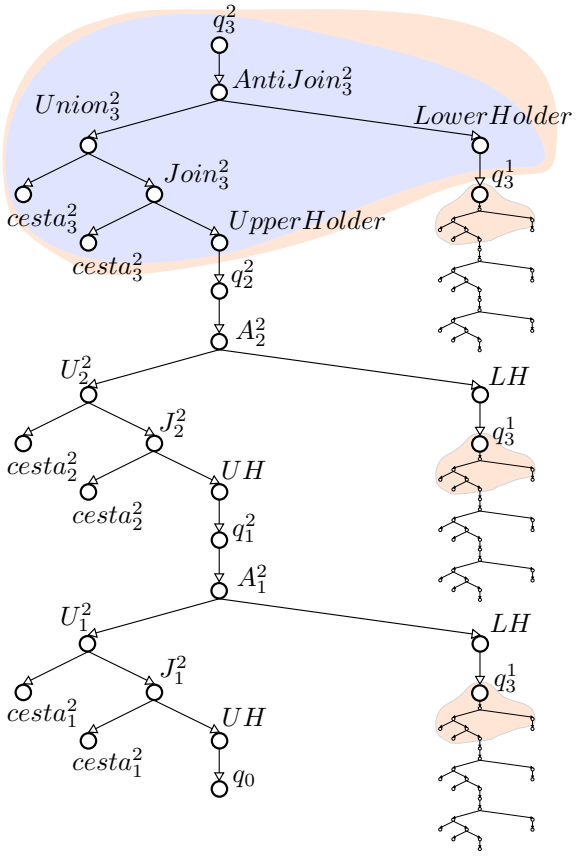
Obr. 12: Mestá do vzdialenosti 2



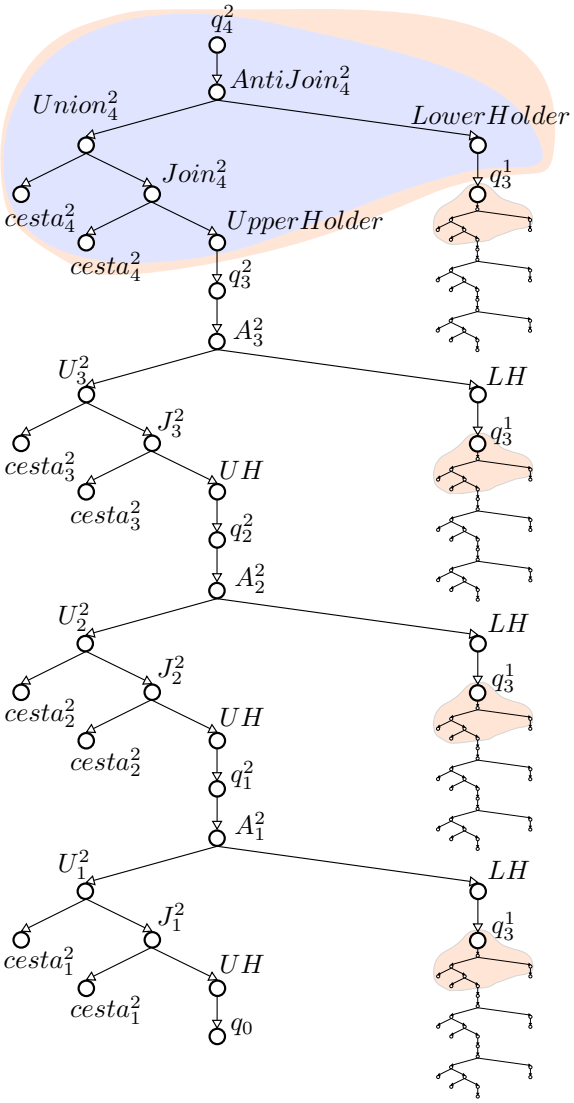
Obr. 13: Mapa q_3^2



Obr. 14: Mapa q_4^2

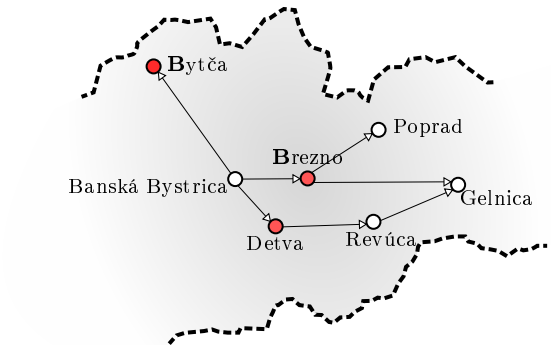


Obr. 15: Mestá do vzdialenosti 3

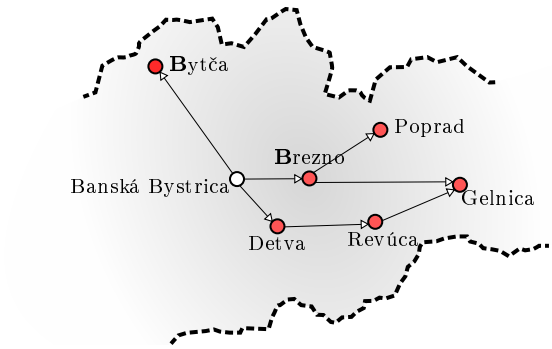


Obr. 16: Mestá do vzdialenosti 4

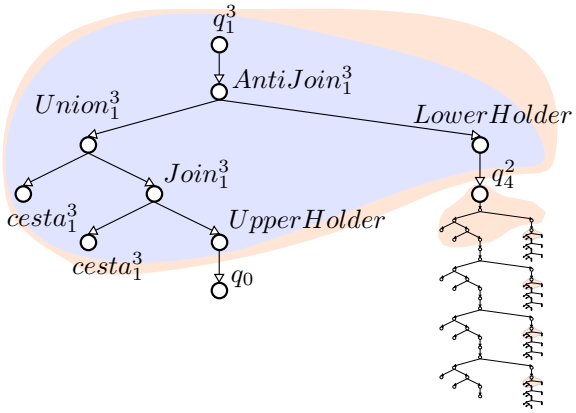
Výpočet q^2 skončí, pretože q_4^2 ={Detva, Revúca, Gelnica} = q_3^2 ={Detva, Revúca, Gelnica}



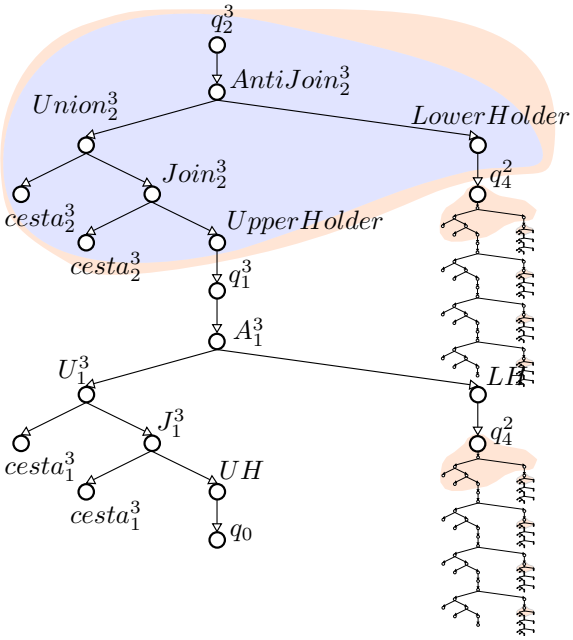
Obr. 17: Mapa q_1^3



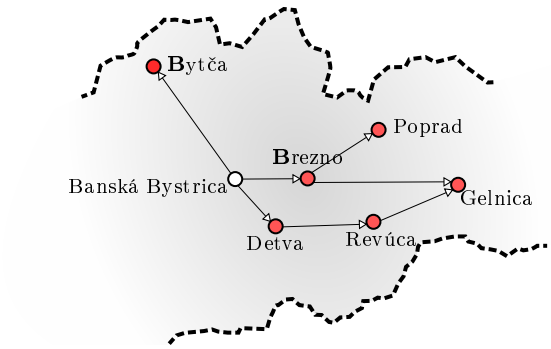
Obr. 18: Mapa q_2^3



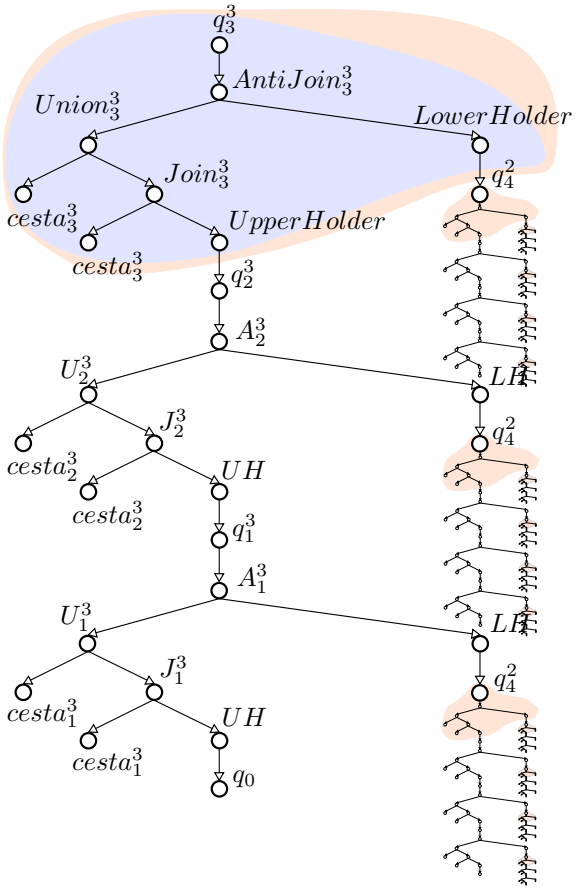
Obr. 19: Mestá do vzdialenosti 1



Obr. 20: Mestá do vzdialenosti 2



Obr. 21: Mapa q_3^3



Obr. 22: Mestá do vzdialenosti 3

Výpočet q^3 skončí, pretože q_3^3 ={Bytča, Brezno, Detva, Revúca, Poprad, Gelnica} = q_2^3 ={Bytča, Brezno, Detva, Revúca, Poprad, Gelnica}