DJ2 – grafové jazyky

slajdy k přednášce NDBI001

Jaroslav Pokorný

Obsah

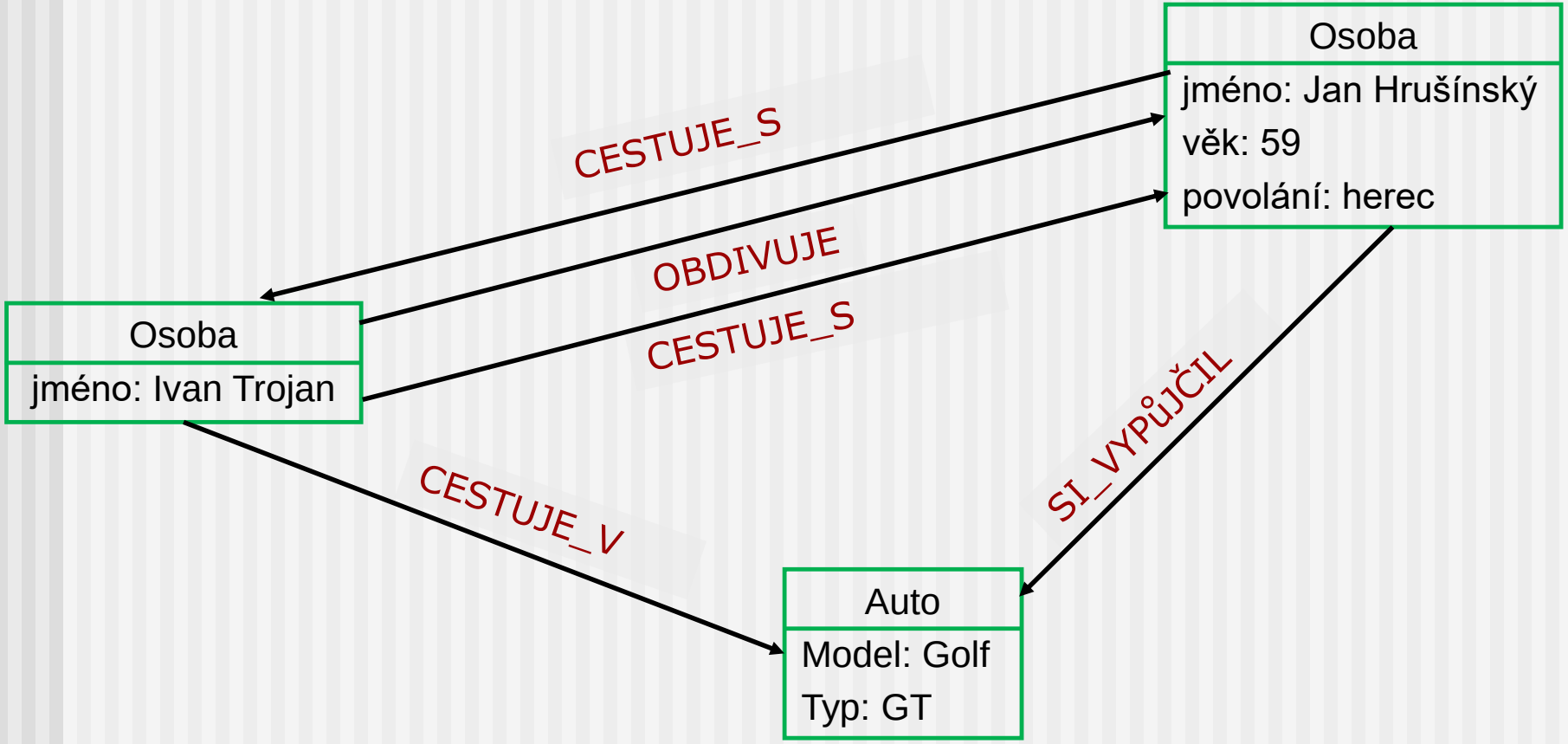
Motivace

- Grafové databáze:
  - data reprezentovaná v podobě grafu
  - kolekce více grafů
- Aplikační oblasti:
  - hypertext
  - sémantický web
  - sociální sítě (Facebook, Twitter, ...)
  - dopravní sítě
  - sémantické asociace (vyšetřování trestné činnosti)
  - biologické sítě
  - ... atd.

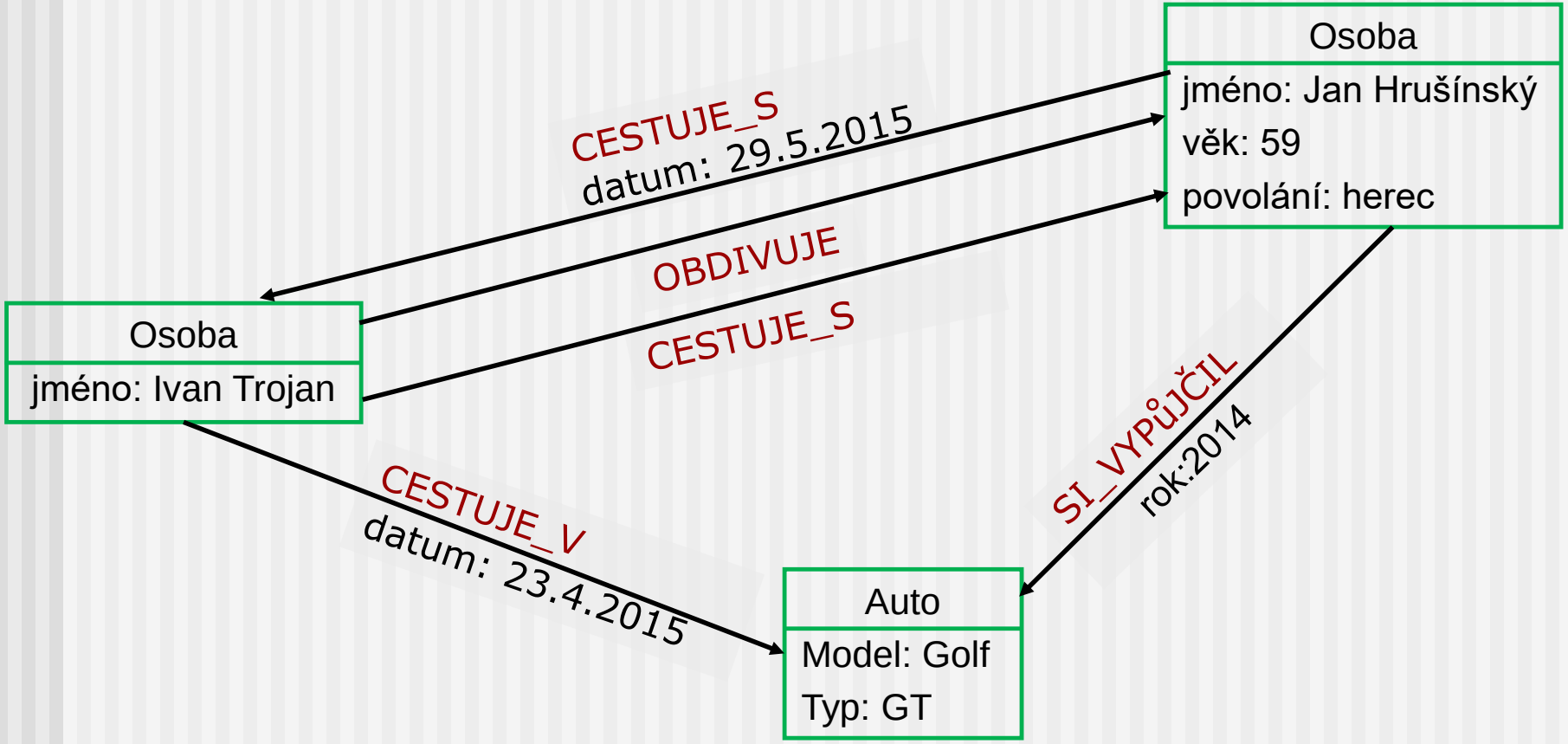
Datový model

- Orientovaný nebo neorientovaný graf (vážený graf, s ohodnocenými hranami, někdy i hypergraf, multigraf)
- Konceptuálně:
  - uzly s vlastnostmi (atributy)
  - označené vztahy s vlastnostmi (atributy)
- Nejčastější případ: orientovaný atributový graf (spíše multigraf) s označenými hranami. Nověji: i uzly mají označení.

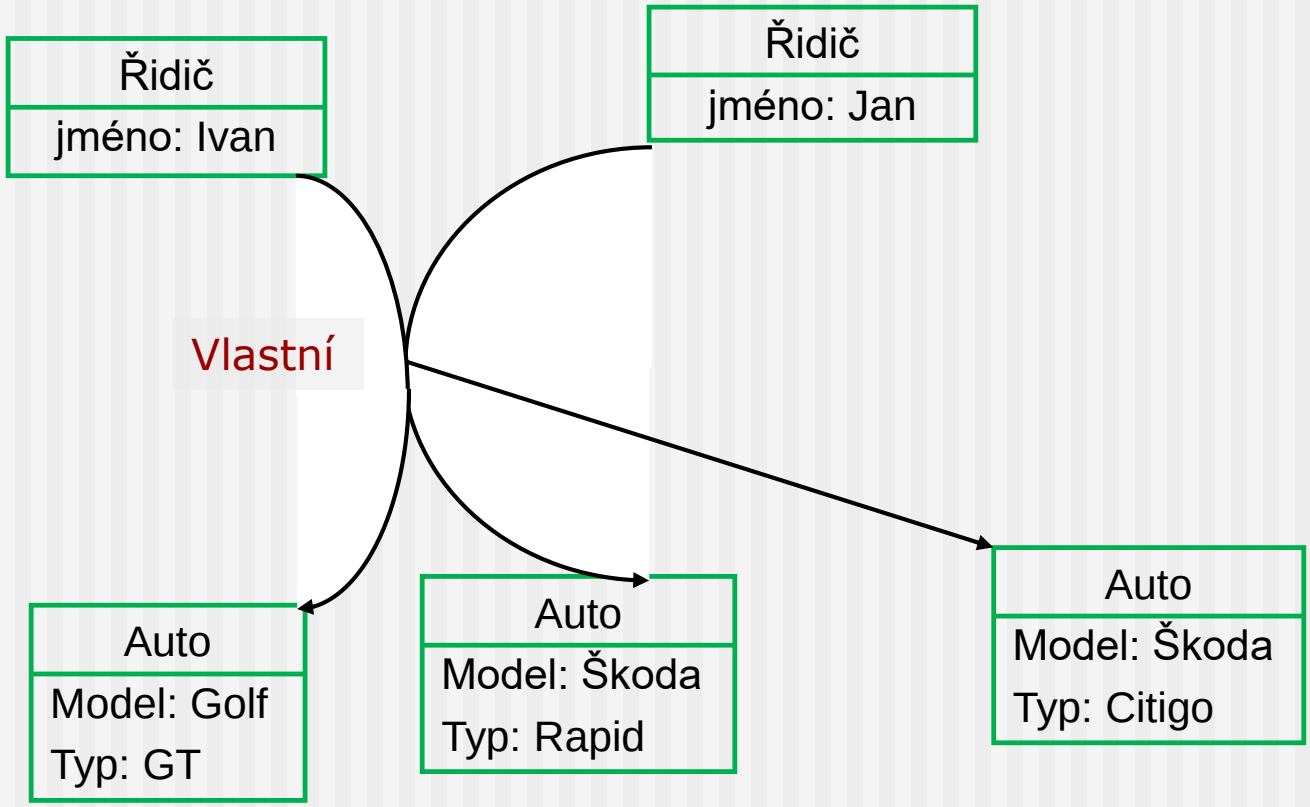
Příklad orientovaného atributového multigrafu



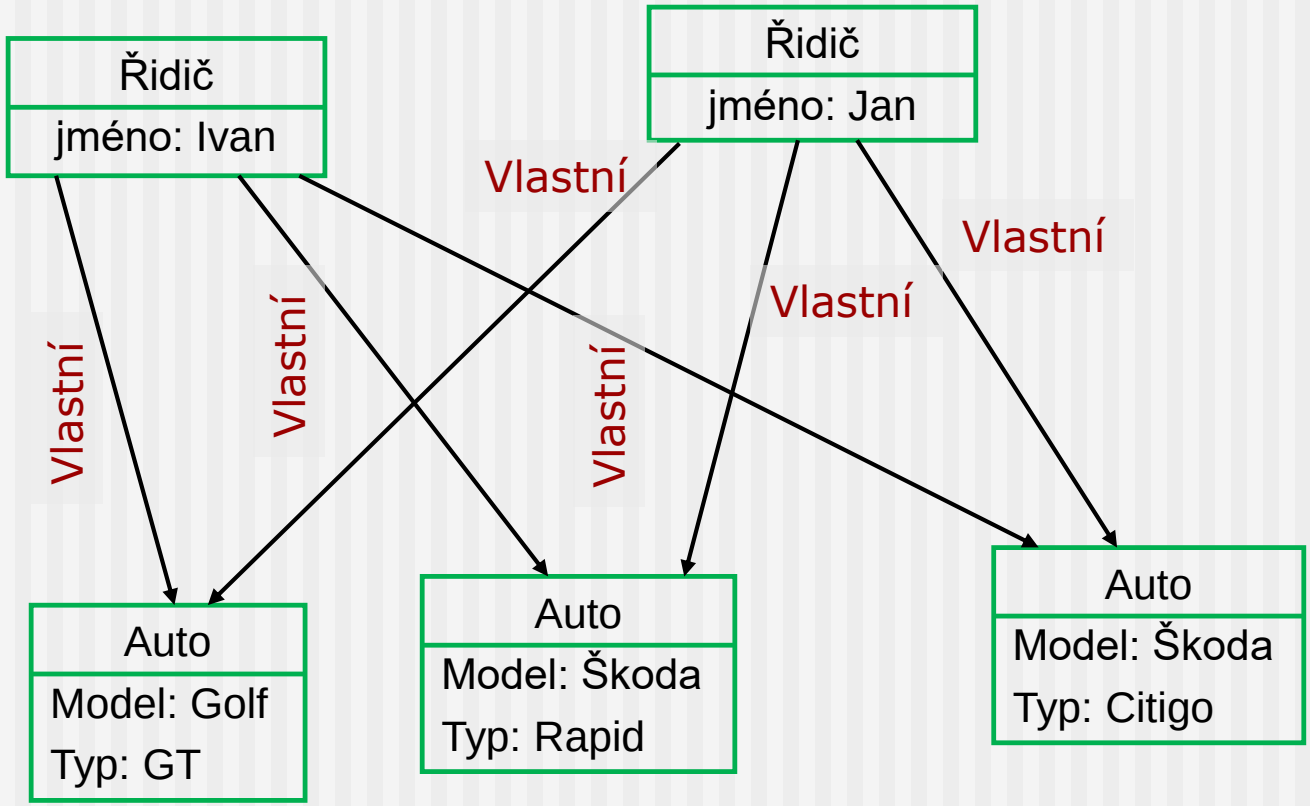
Příklad orientovaného atributového multigrafu



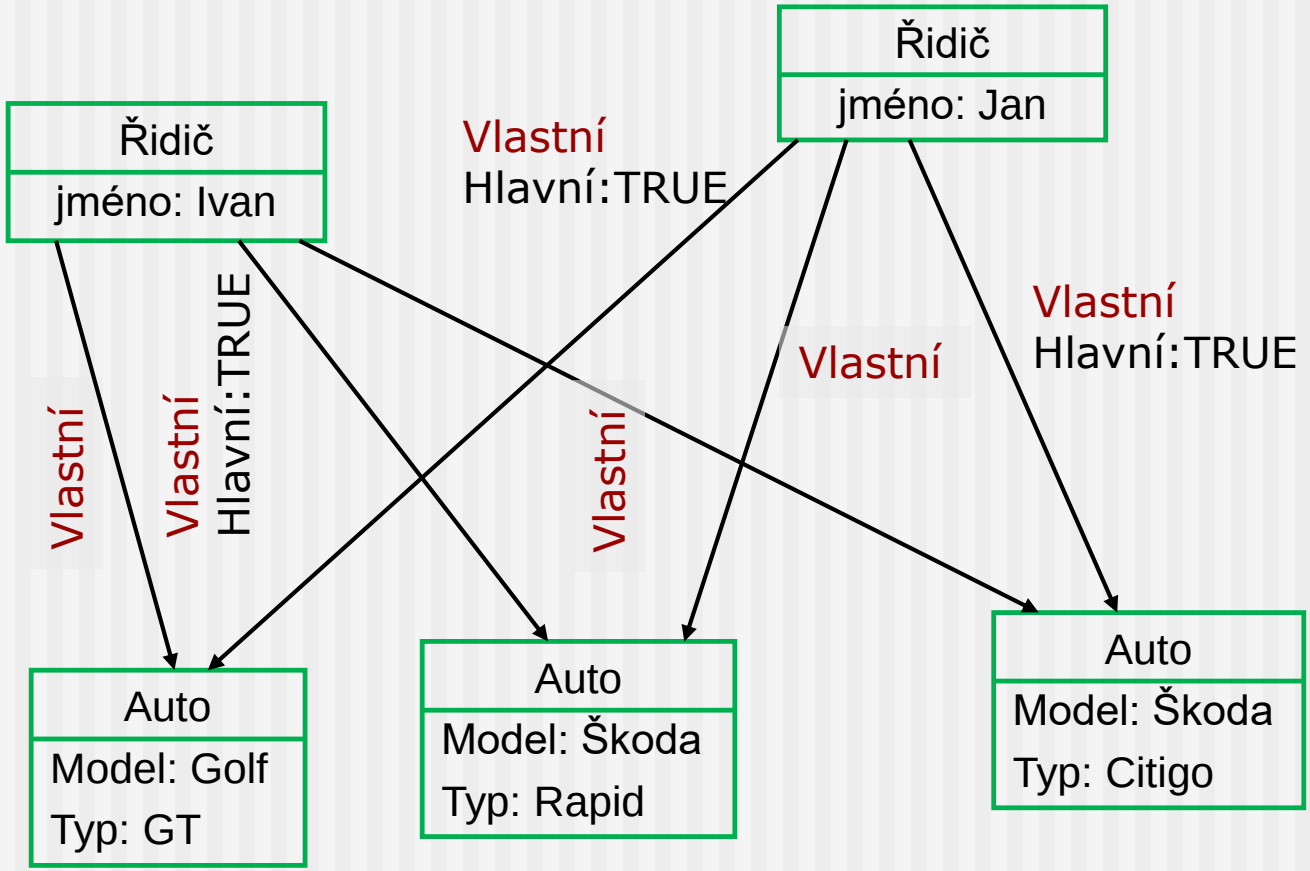
Příklad hypergrafu



Ekvivalentní multigraf



Sémanticky bohatší multigraf



Formálněji

Df: Databázový graf $G = (V, E, N, \Sigma, \varphi, \lambda, A, Att)$ je orientovaný, ohodnocený atributový multigraf, kde V je konečná množina uzlů s identifikátory vytvořenými z konečné abecedy N, E je množina hran a φ je incidenční funkce z E do $V \times V$. Značky hran jsou vytvořeny nad abecedou Σ, λ je funkce z E do Σ označující hrany. A je množina attributů (vlastností) reprezentovaná dvojicemi (A_i, $hodnota_{ij}$). Att je zobrazení přiřazující každému uzlu/hraně podmnožinu (event. prázdnou) atributů z A. Identifikátorům uzlů se rovněž říká značky (značky uzlů).

Pz.: Definice dovoluje databázové grafy s různými množinami atributů uzlů/hrana stejných typů. To se vyskytuje v praxi, spec. v GSŘBD bez schématu. Často jsou definovány domény atributů. Pak $value_{ij} \in dom(A_i)$.

Grafové databáze

- Pomocí relačních SŘBD, Datalogu
- Prostředky pro grafové dotazy v SQL:
  - konečný počet spojení ve výrazu dotazu
  - rekurzivní SQL
- Prostředky pro grafové dotazy v Datalogu:
  - D1: Najdi tranzitivní uzávěr relace PRACUJE_PRO

    POD_NAD(x,y):-PRACUJE_PRO(x,y)

    POD_NAD(x,y):-PRACUJE_PRO(x,z), POD_NAD(z,y)
- Prostředky pro grafové dotazy v XML

(Nativní) grafové databáze

- Obvykle nemají schéma
- Umožňují efektivní uložení dat bez relací a pomalého SQL nad nimi
  - každý uzel zná své sousedy (vlastnost index-free adjacency),
  - se zvyšujícím se počtem uzlů, cena lokálního kroku zůstává stejná (předp. indexace pro vyhledávání)
- GSŘBD, GDB – často alternativní pojmy

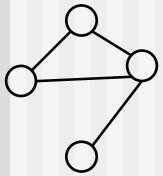
Grafové databáze

- Přirozená formulace grafových problémů v dotazovacím jazyku
- ACID transakce
- Př.: Neo4j (2004), Pregel (2008), HypergrafDB (2009), Sparksee (2008), Titan (2012), OrientDB (2010), Giraph (2010), …

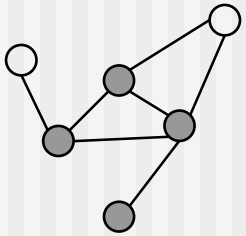
Dotazy nad grafy

- tradiční:
  - nejkratší cesta
  - dosažitelnost
  - isomorfismus podgrafu
  - Page Rank
  - klastrování
  - kritická cesta
- novější:
  - vyhledávání podle klíčových slov
  - vyhledávání v grafu
  - grafové porovnávání vzorů
  - dolování vzorů v grafu
  - detekce anomálií
  - Skyline na grafech
  - OLAP na grafech
  - agregace grafu

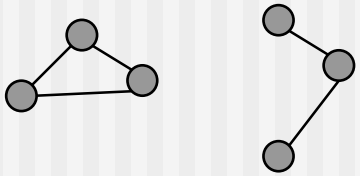
Další typy dotazů



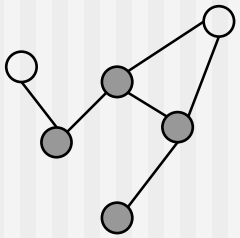
graf dotazu



dotaz na přesnou shodu



dotaz na shody
v supergrafu



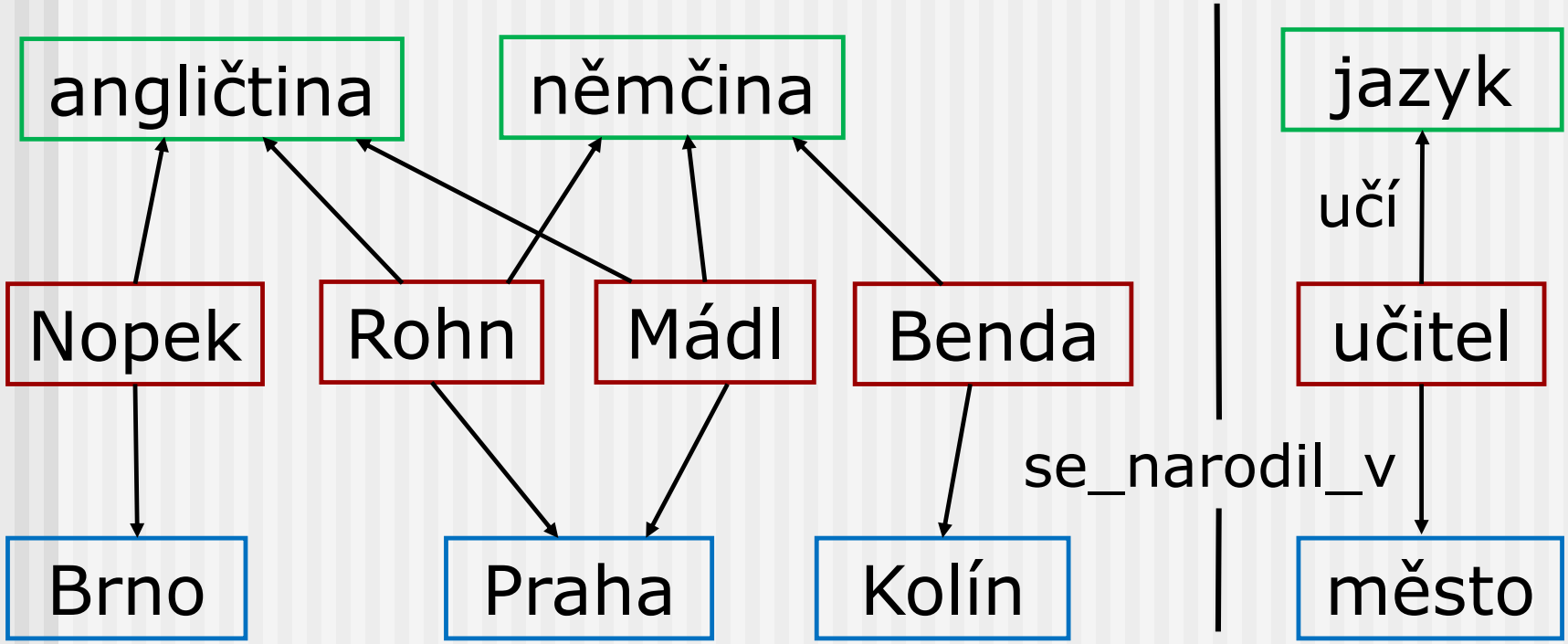
dotaz na podobnost
(vyžaduje míru podobnosti)

Odpovědi

- Závisí na dotazovacím jazyku a zdali je databáze jeden graf nebo více grafů
- Odpověď může být:
  - množina grafů, ve kterých je shoda s dotazem,
  - množina podgrafů, na které je shoda s dotazem
  - množin dosazení do proměnných v dotazu

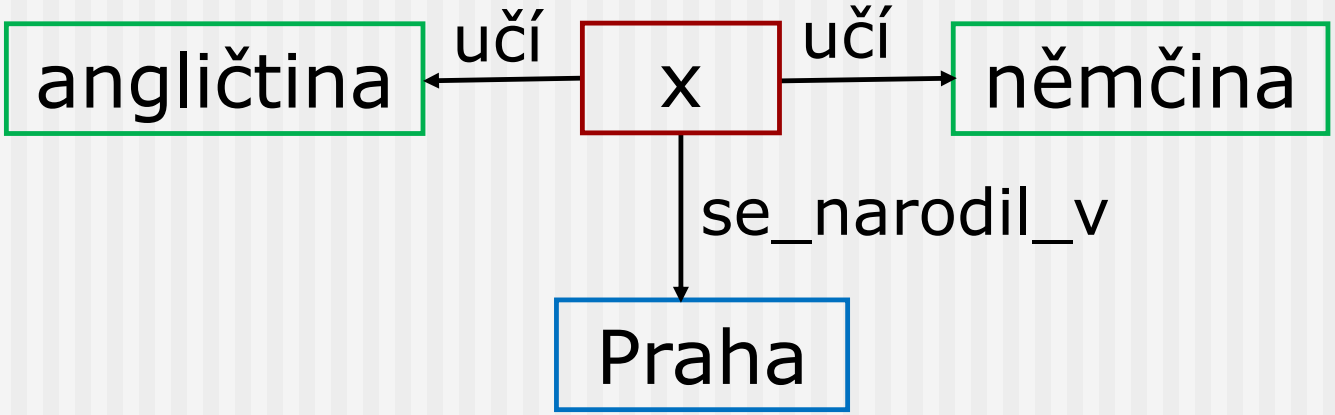
Příklad grafu

Graf učitelů, kteří učí jazyky a města, ve kterých se narodili.



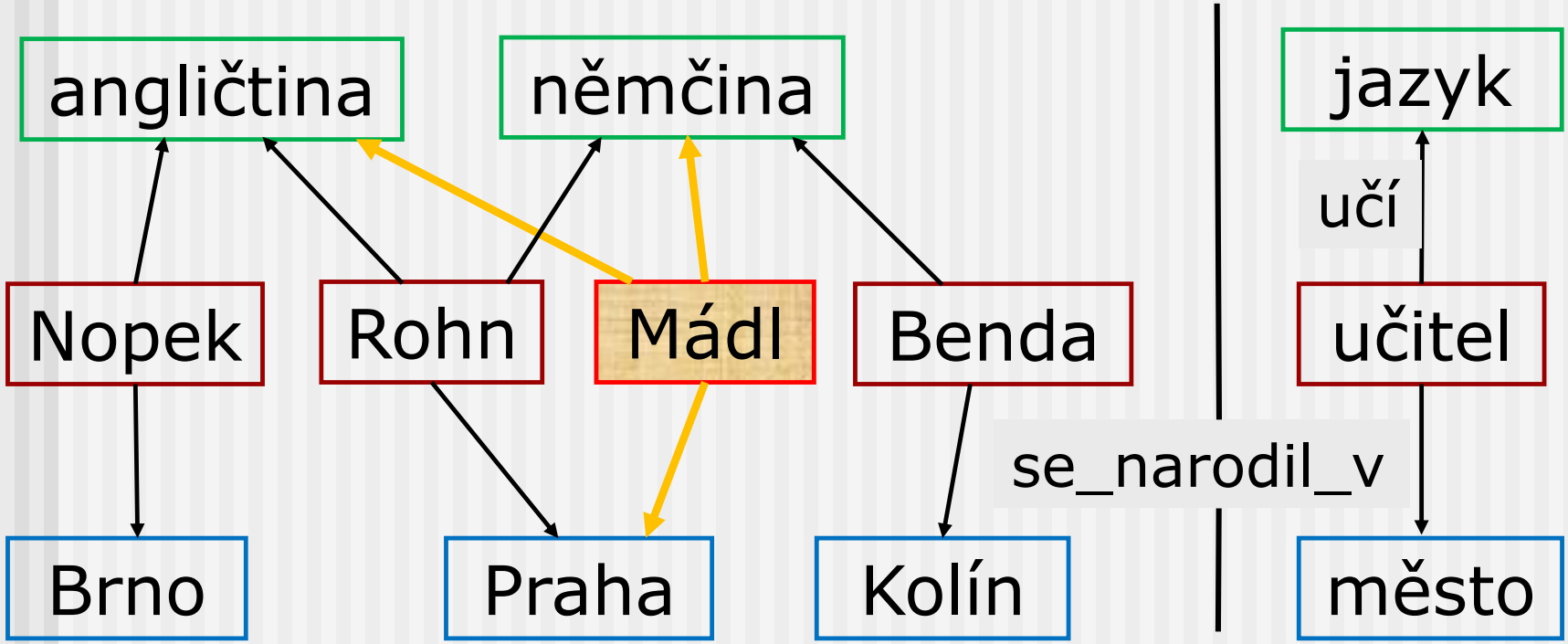
Příklad dotazu

Kteří učitelé narození v Praze učí angličtinu a němčinu?

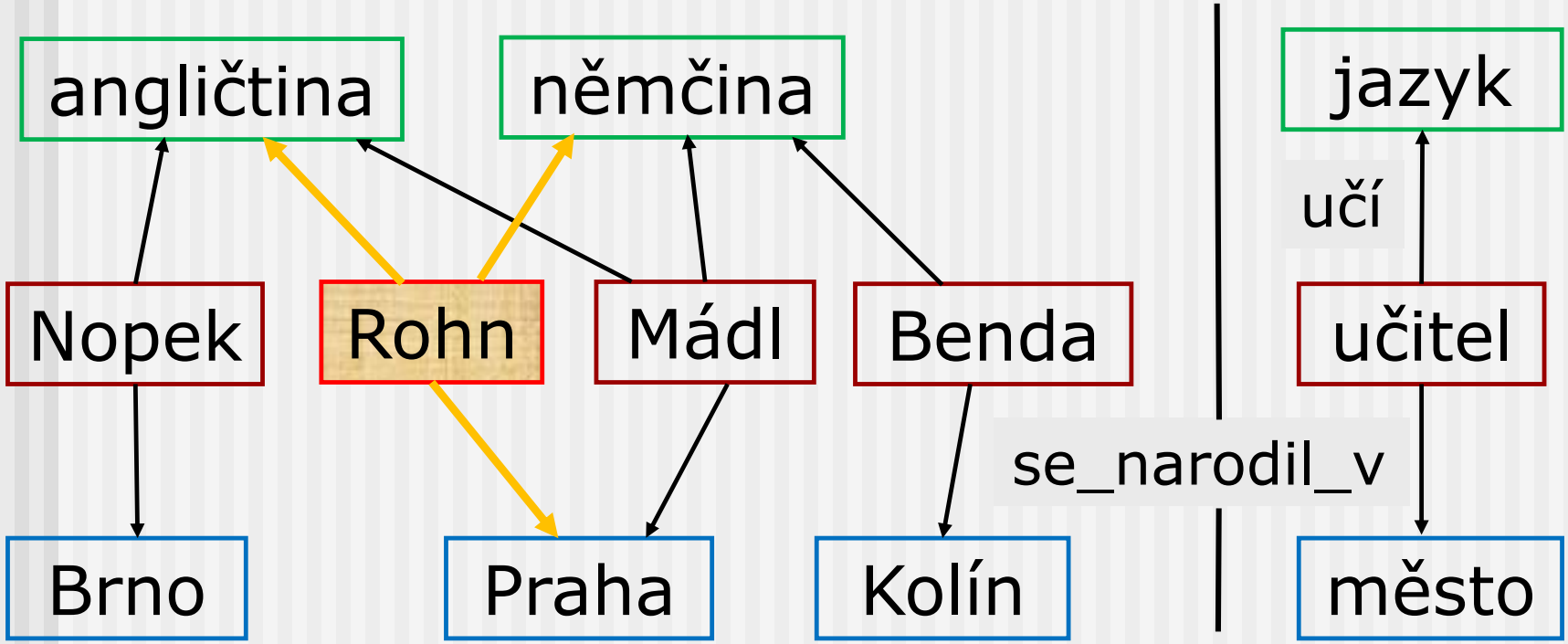


kde x je proměnná.
Jde o jazyky s grafovými vzory (např. G, GraphLog)

Shoda ve dvou podgrafech



Shoda ve dvou podgrafech



Konjunktivní dotaz

- konjunktivní dotaz nad abecedou Σ:

$$\text{ans}(z_1,\ldots,z_n) \leftarrow \wedge\ (x_i,\ a_i,\ y_i),\ 1 \le i \le m$$

kde $m > 0$, x_i a y_i bud' uzlové proměnné nebo konstanty ($1 \le i \le m$), $a_i \in \Sigma$ ($1 \le i \le m$), a z_i je jedna z x_j nebo y_j ($1 \le i \le n$, $1 \le j \le m$).
Odpovědí na dotaz je množina n-tic uzlů.

Je-li hlava tvaru ans() jde o dotaz typu ANO/NE

Př: ans(*x*) ← (*x, učí, němčina*), (*x, učí, angličtina*), (*x, se_narodil_v, Praha*)

…v jiných dotazovacích jazycích

- à la SQL/OQL (Lorel, RQL):

```
SELECT X
FROM X.učí Y, X.učí Z, X:narodil_se_v W
WHERE   Y = angličtina AND Z = němčina AND
        W = Praha
```

W, X, Y a Z jsou proměnné

Symboly:
| disjunkce
· konkatenace
r* uzávěr)

Dotaz s regulární cestou

- dotaz s regulární cestou nad abecedou Σ:

$$\text{ans}(x, y) \leftarrow (x, r, y),$$

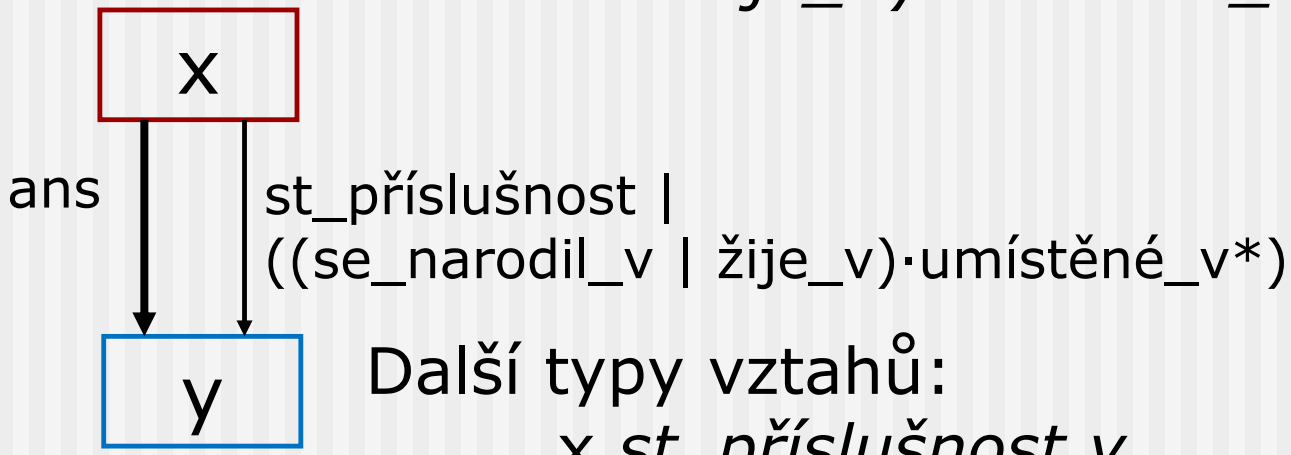
kde x a y jsou uzlové proměnné r je regulární výraz nad Σ.

Odpovědí na dotaz je množina dvojic uzlů (x,y) takových, že existuje cesta z x do y a posloupnost označení hran na té cestě vyhovuje r.

Příklad dotazu s regulární cestou

Najdi učitelé a místa k nim vztažená?

ans(*x, y*) ← (*x, st_příslušnost* | ((*se_narodil_v* | *žije_v*)·*umístěné_v**), *y*)



Další typy vztahů:

x *st_příslušnost y*

x *žije_v y*

x *umístěné_v x* //relace *být částí*

Dotaz s jednoduchou regulární cestou

- dotaz s jednoduchou regulární cestou nad abecedou Σ:

$$\text{ans}(x, y) \leftarrow (x, r, y),$$

kde x a y jsou uzlové proměnné r je regulární výraz nad Σ. Cesta je jednoduchá, neobsahuje-li žádný opakující se uzel.

Odpovědí na dotaz jsou dvojice uzlů (x,y) takové, že existuje jednoduchá cesta z x do y a posloupnost označení hran na té cestě vyhovuje r.

Konjunktivní dotaz s regulárními cestami

- konjunktivní dotaz s regulárními cestami nad abecedou Σ:

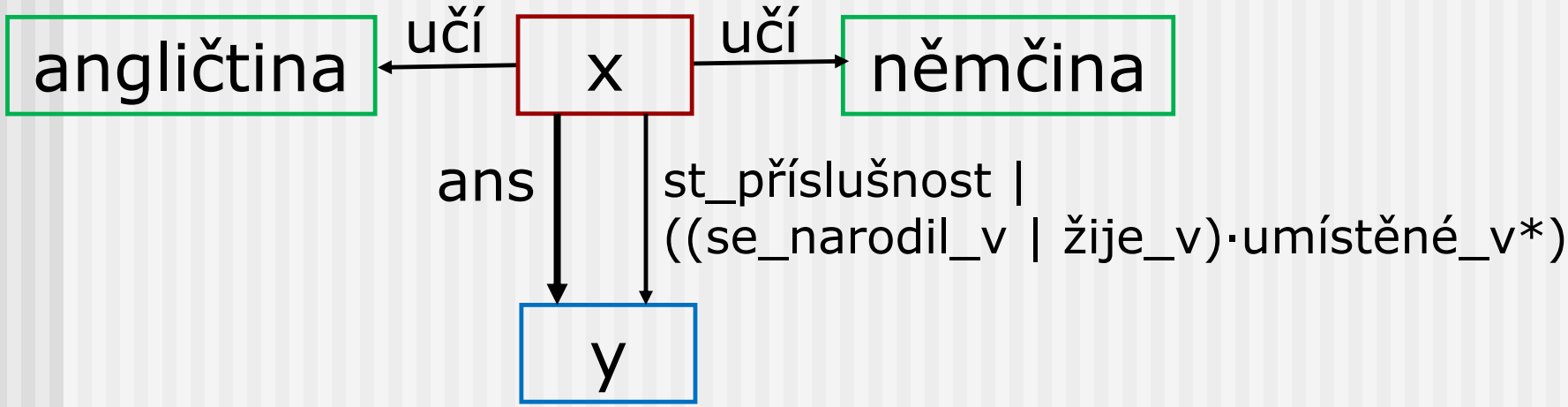
$$\text{ans}(z_1,\ldots,z_n) \leftarrow \wedge\ (x_i, r_i, y_i),\ 1 \le i \le m$$

kde r_i jsou regulárními výrazy nad Σ, z_i jsou x_j nebo y_j. Odpovědí na dotaz je množina n-tic uzlů.

Př: ans(*x*, *y*) ← (*x*, *učí*, *němčina*), (*x*, *učí*, *angličtina*),
(*x*, *st_příslušnost* |
((*se_narodil_v* | *žije_v*)·*umístěné_v**), *y*)

Příklad konjunktivního dotazu s regulárními cestami

Najdi učitelé, kteří učí angličtinu a němčinu, a místa k nim vztažená?



Další možnosti

- rozšířený konjunktivní dotaz s regulárními cestami nad abecedou Σ:
- Př.: Najdi *x* a *y*, přičemž cesta z *y* k *x* je stejná jako z *y* k Rohnovi?

Př: ans(x, y) ← ($Rohn$, π, y), (x, π, y), ($\Sigma^*\pi$)

Kde π je proměnná pro cestu, Σ^* označuje jakoukoliv posloupnost označení hran.

Pz.: užitečné v RDF pro porovnávání sémantických asociací. Jde o vztahy mezi cestami.

Složitost vyhodnocování

- Q(G) - výsledek vyhodnocení dotazu Q nad grafem G
- Problém vyhodnocování dotazu: patří uzel (dvojice uzlů) do Q(G)?
- Složitost:
  - kombinovaná - Q a G jsou součástí vstupu
  - dotazová - Q je vstupem, G je pevně dán
  - datová - G je vstupem, Q je pevně dán

Pz.: Protože Q je většinou krátký a G velký uvažuje se často datová složitost.

Složitost vyhodnocování

- Kombinovaná složitost i dotazová složitost problému vyhodnocování konjunktivních dotazů je ekvivalentní hledání isomorfismu podgrafů, což je NP-úplný problém.

  Datová složitost je PTIME.
- Problém vyhodnocování dotazů s regulární cestou má PTIME kombinovanou složitost.
- Problém vyhodnocování dotazů s jednoduchou regulární cestou je NP-úplný problém.

Vyhodnocení dotazu s regulární cestou

Př.: ans(*x, y*) ← (*x, st_příslušnost* | ((*se_narodil_v* | *žije_v*)·*umístěné_v**), *y*)

Možnost pomocí DATALOGu:

asoc(X,Y) ← se_narodil_v(X,Y)

asoc(X,Y) ← žije_v(X,Y)

je_částí(X,Y) ← umístěné_v(X,Y)

je_částí(X,Y) ← umístěné_v(X, Z), je_částí(Z,Y)

ans(X,Y) ← st_příslušnost(X,Y)

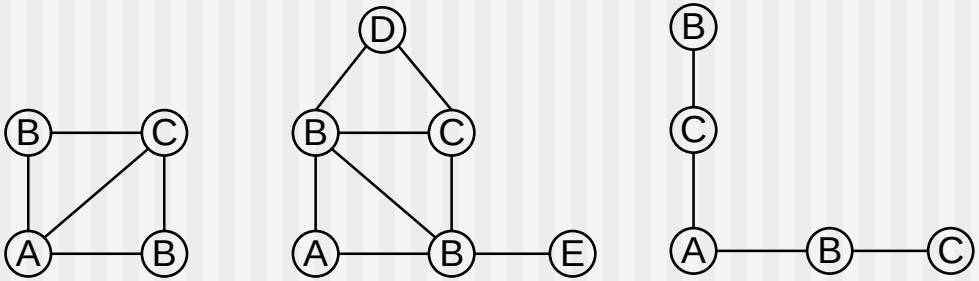
ans(X,Y) ← asoc(X,Y)

ans(X,Y) ← asoc(X, Z), je_částí(Z,Y)

Pz.: podobně jako v jazyku Graphlog

Implementace: indexace

- Př.: index GaphGrep (Sasha et al, PODS'02)
- Přepoklad: neorientované grafy s označenými uzly. Uzly mají Id.
- index založený na cestách z označení uzlů a z Id.
  - Proveď enumeraci všech cest délky <= L všech grafů v DB
  - Pro každou cestu ulož počet jejich výskytů ve všech grafech DB do hašovací tabulky.
  - Zde: jen cesty z označení uzlů. Obsahuje-li cesta *p* hranu *e*, pak se *e* na *p* vyskytuje pouze jednou.

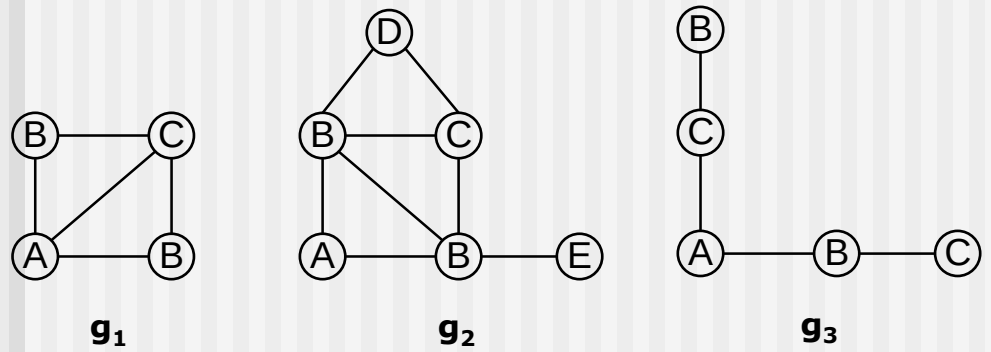


klíč	g_1	g_2	g_3
h(CA)	1	0	1
…			
h(ABCB)	2	2	0

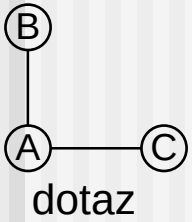
index

Implementace: indexace

- Filtrování: zahašuj všechny cesty dotazu (délky <= L)
  - v indexu nalezni grafy – kandidáty (pozor na kolize h)
  - eliminuj grafy, kde počet výskytů nějaké cesty je menší počet výskytů té cesty v grafu dotazu
  - proveď a verifikuj, tj. kontroluj isomorfismus



Key	g_1	g_2	g_3
h(AB)	2	2	1
h(AC)	1	0	1
h(BAC)	2	0	1



AB:1
AC:1
BAC:1

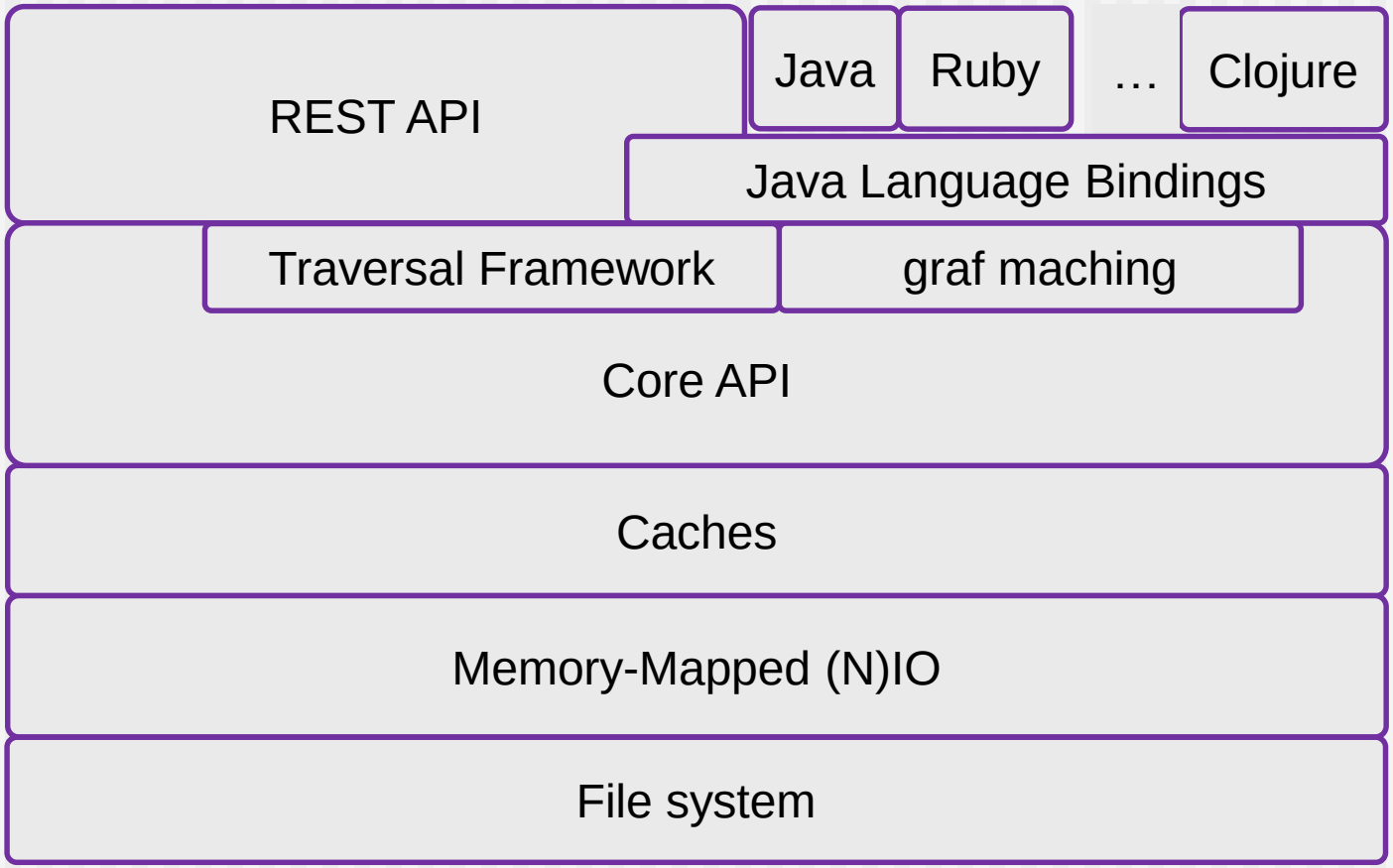
kandidáti
= {g1, g3}

verifikace

Neo4j: je NoSQL databáze

- Vlastnosti: ACID transakce, vysoká dostupnost, šálovatelnost (miliardy uzlů, vztahů), rychlé dotazování,
- Dotazy:
  - Programováním prostřednictvím Java API REST
  - vazba na jazyky Neo4.rb (JRuby), Gremlin, …
  - Deklarativním jayzkem Cypher (SPARQL je navržen pro jiný datový model)
- Management uzlů a vztahů
- Indexace
- Průchod grafem
- Hledání cest
- Porovnávání vzorů

Neo4j: logická architektura



Základní pojmy

- uzly
  - vlastnosti
    - dvojice klíč-hodnota
- vztahy
  - počáteční uzel
  - koncový uzel
  - vlastnosti
    - dvojice klíč-hodnota
- Indexy
  - pro uzly
  - pro vztahy

Dotazovací jazyk Cypher

Inspirovaný SQL:

CREATE: vytvoří uzel, vztah; např.

CREATE(n:OSOBA { jméno: ’Jan’}) //OSOBA je label, n je proměnná pro nový uzel

identifikátory – jména přiřazená částem grafu: n, A, r, jméno, osoba

Pz.: jde vlastně o proměnné

START: (volitelné) vstupní body v grafu (indexem nebo ID) pro vzorek grafu

- Např. START n=node(*), START n=node(3),
  START n=node:Osoba(jméno=‘Ivan Trojan’]

dotazování příkladem

MATCH: vzorek grafu, vázaný ke vstupním bodům ve START je zadán pomocí jedné nebo několika cest

Značení: uzly: (a) nebo () (anonymní uzly)

hrany - ->, <- -, - -

Dotazovací jazyk Cypher

- vztažené uzly (neorientované) pomocí - -
- vztažené uzly vstupující (vystupující) pomocí - - > (< - -)
- vztahy vstupující (vystupující) pomocí proměnné, např. [r], nebo jménem vztahu (označená hrana), např. (a) - [:friend] -> (b) spojené cesty, např. (a)- - > (b) - - > (c), (a) - [*] -> (b), (a) - [*1..4] -> (b), (a) - [?] -> (b)

- Funkce: např. nodes (relationships) vrací všechny uzly (vztahy) na cestě
- Pokročilé dotazování: vestavěné grafové algoritmy
  - shortestPath
  - allSimplePaths
  - allPaths
  - dijkstra (volitelně s cost_property a default_cost parameters)

Dotazovací jazyk Cypher

WHERE: filtrovací kritéria (AND, OR, NOT, porovnání, regulární výrazy, …)

Agregační funkce: COUNT, SUM, AVG, MAX, MIN, COLLECT

RETURN: tvar odpovědi

ORDER BY: (podobné SQL) DESC, …

LIMIT: omezení výstupu (řádky)

CREATE: vytváří uzly a vztahy

DELETE: odstraňuje uzly, vztahy a vlastnosti

SET: dává hodnoty do vlastností

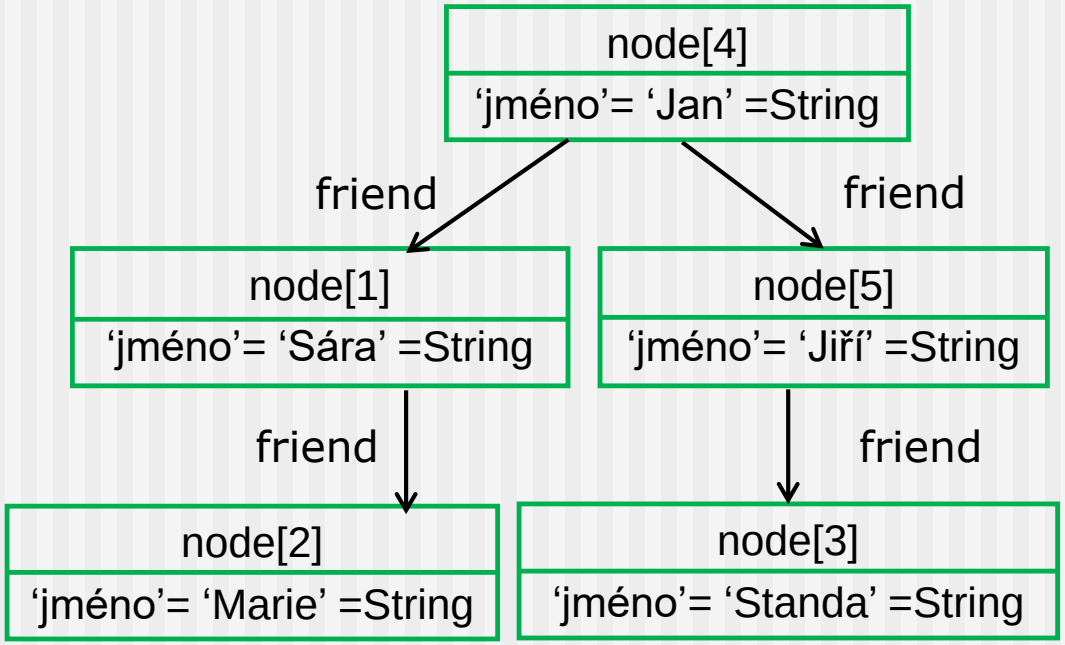
FOREACH: provádí aktualizační akce jednou pro každý prvek seznamu

WITH: rozdělí dotaz do více různých částí

Dotaz na sousedy uzlu

x
node[4]{jméno->"Jan"}
node[3]{jméno->"Standa"}

Odpověď



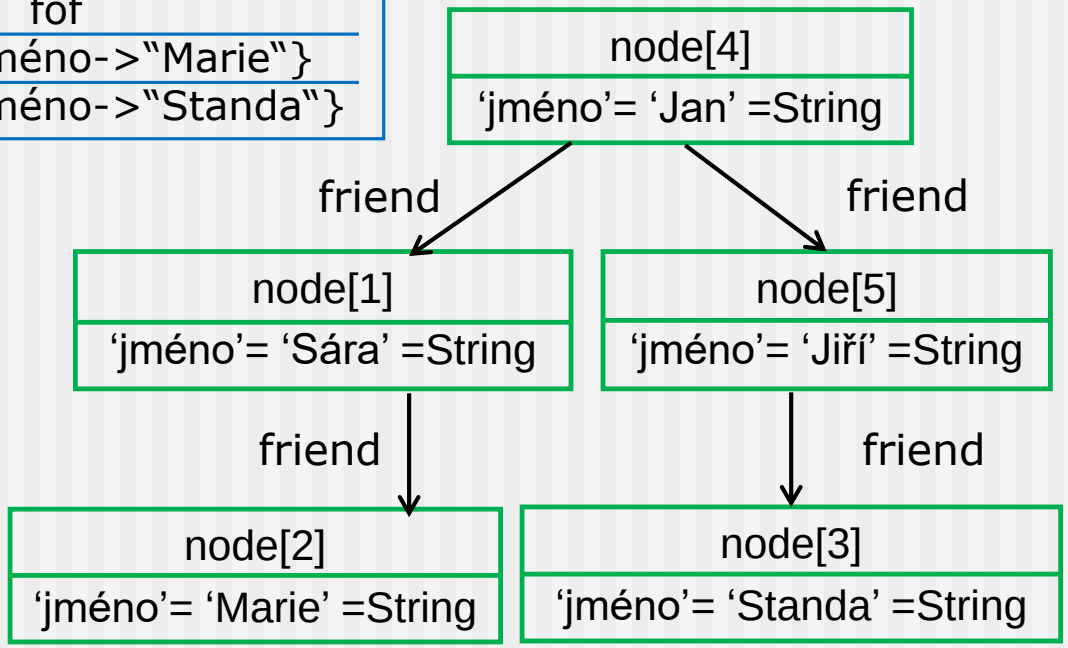
Dotaz

```
START n = node(5)          // funkce node vrací uzel daného čísla
MATCH (n) - - (x)          // lépe používat vlastní Id
RETURN x
```

Dotaz friend-of-friend

jan	fof
node[4]{jméno->"Jan"}	node[2]{jméno->"Marie"}
node[4]{jméno->"Jan"}	node[3]{jméno->"Standa"}

Odpověď



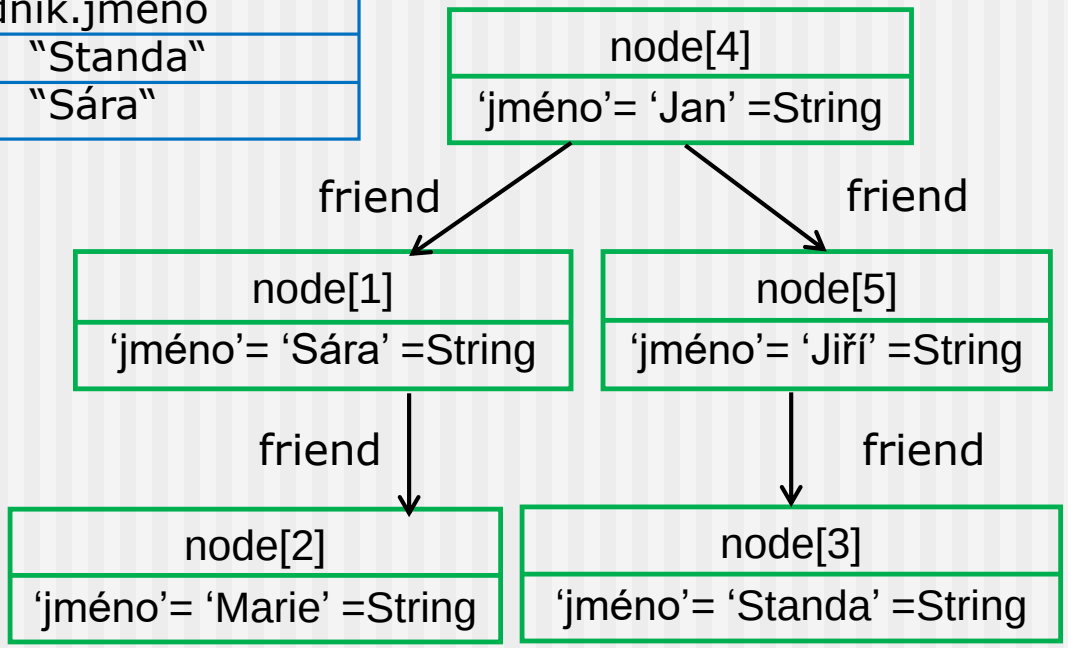
Dotaz

```
START jan = node:node_auto_index(jméno='Jan')
MATCH jan- [:friend] -> () - [:friend] -> fof
RETURN jan, fof
```

Dotaz s regulárním výrazem

uživatel	následník.jméno
node[5]{jméno->"Jiří"}	"Standa"
node[4]{jméno->"Jan"}	"Sára"

Odpověď

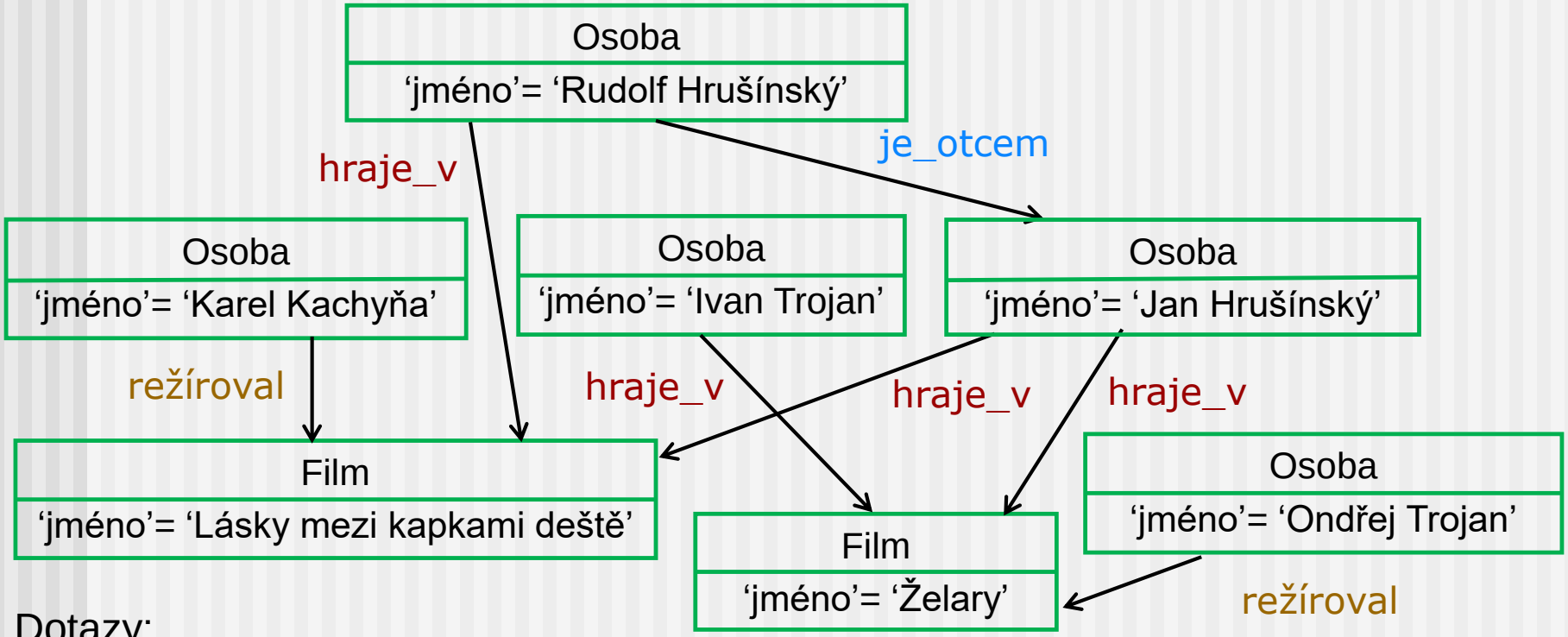


Dotaz

```
START uživatel=node(5,4,1,2,3)
MATCH uživatel - [:friend] -> následník
WHERE následník.jméno = ~/S.*/          //značení: ~/reg_výraz/
RETURN uživatel, následník.jméno
```

// vytvoř uzel
CREATE (Osoba {jméno: ‘Zdeněk Svěrák})

Dotazy: na vztahy, s alternativou

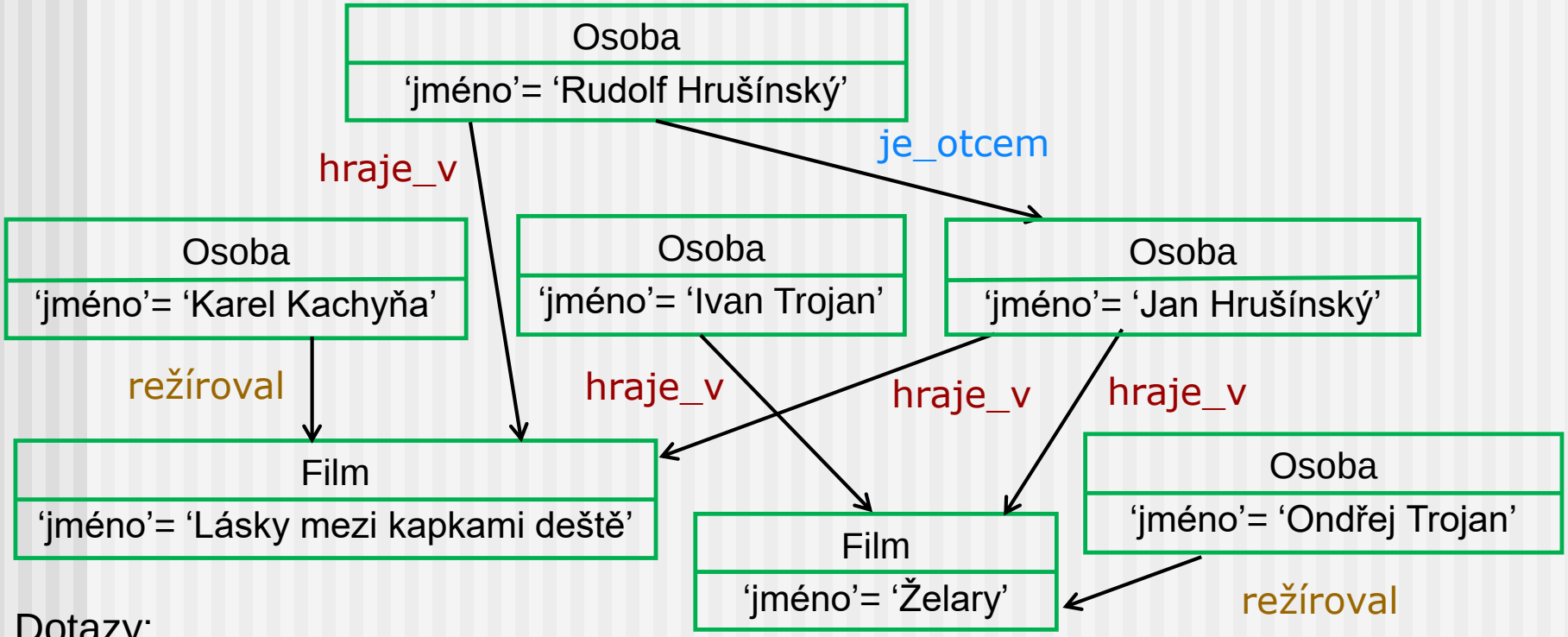


Dotazy:

MATCH (x {jméno: 'Rudolf Hrušínský' })-[r]->y)
RETURN r

MATCH (x)-[:hraje_v | režíroval] ->film{jméno:’Želary’})
RETURN x

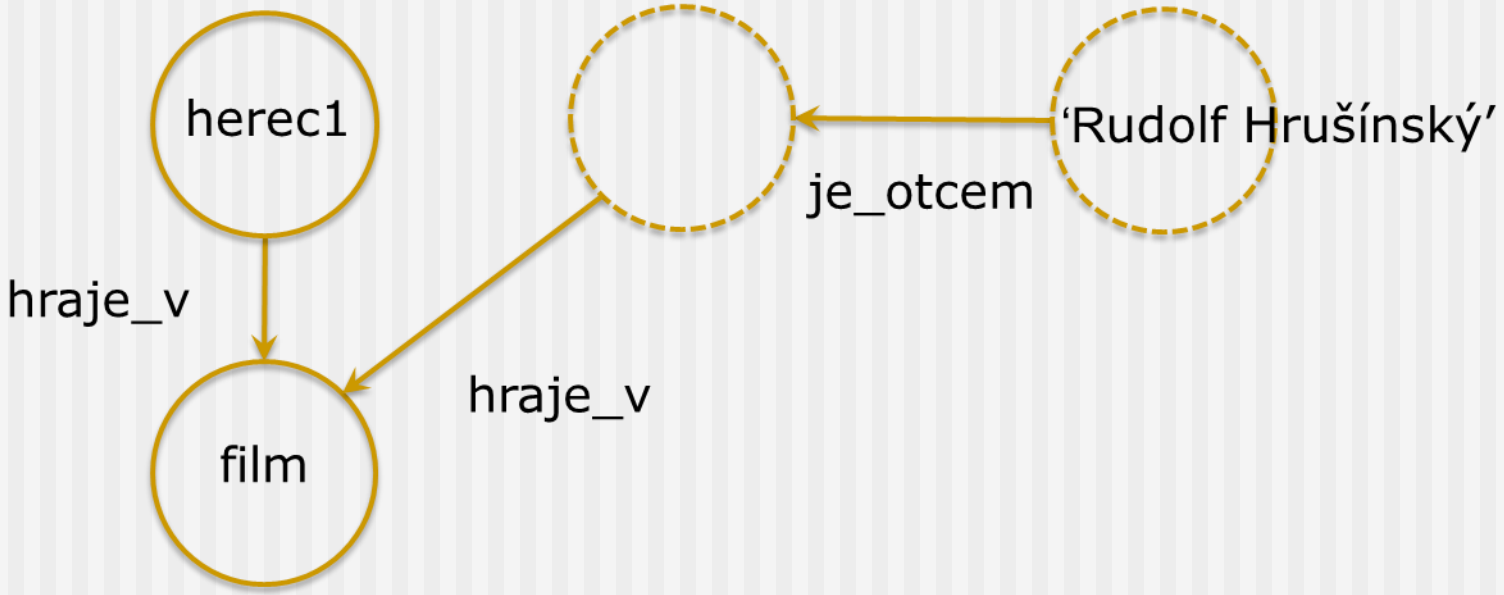
Dotazy na přímé následníky



Dotazy:

```
// najdi film vytažený k Janu Hrušínskému
START x=node:Osoba(jméno =‘Jan Hrušínský’)
MATCH (x)- ->(film)
RETURN film.jméno
```

Dotaz se složitější cestou



// najdi herce, který hraje ve stejném filmu jako syn Rudolfa Hrušínského

MATCH herec1-[:hraje_v]->film<-[:hraje_v]-()<-[je_otcem]-herec2

WHERE herec2.jméno = 'Rudolf Hrušínský'

RETURN herec1.jméno

Pokročilé konstrukty

- NULL pro reprezentaci chybějící nebo nedefinované hodnoty
- HAS(n.jméno) – musí existovat vlastnost jméno
- Konstrukty ALL, ANY, NONE, SINGLE pro práci s kolekcemi
- skalární funkce LENGTH, TYPE, ID, …
- funkce pro kolekce
- matematické funkce,
- funkce pro práci s řetězci
- IO

Aktualizace

- Přidej hodnotu vlastnosti

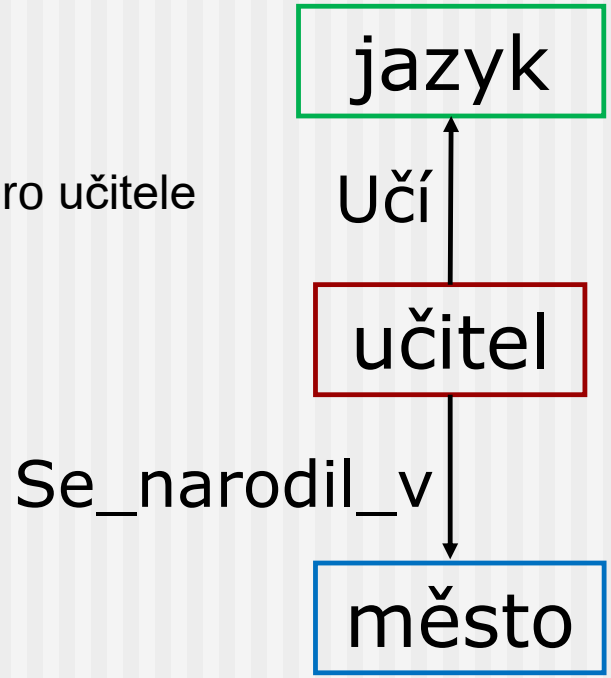
```
MATCH (n:OSOBA)
WHERE n.jméno='Ivan Trojan'
SET n.země='Česká republika'
RETURN n AS Jan, r., m
```

Modelování grafových databází

- Schéma grafové databáze

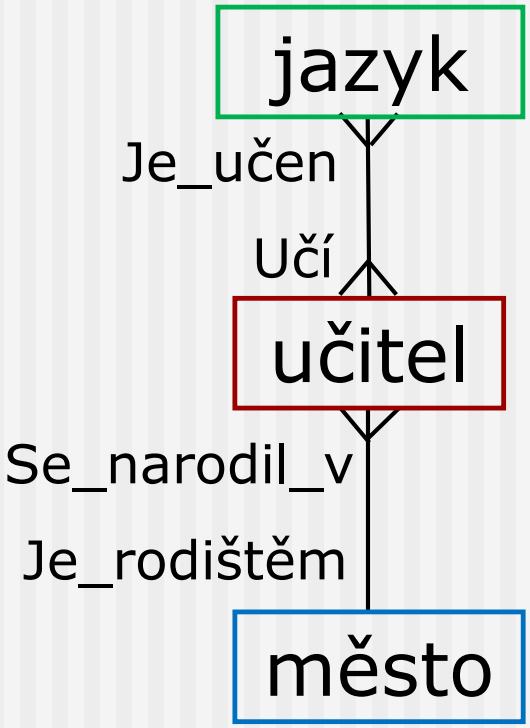
  Mělo by specifikovat strukturu + definice atributů + IO

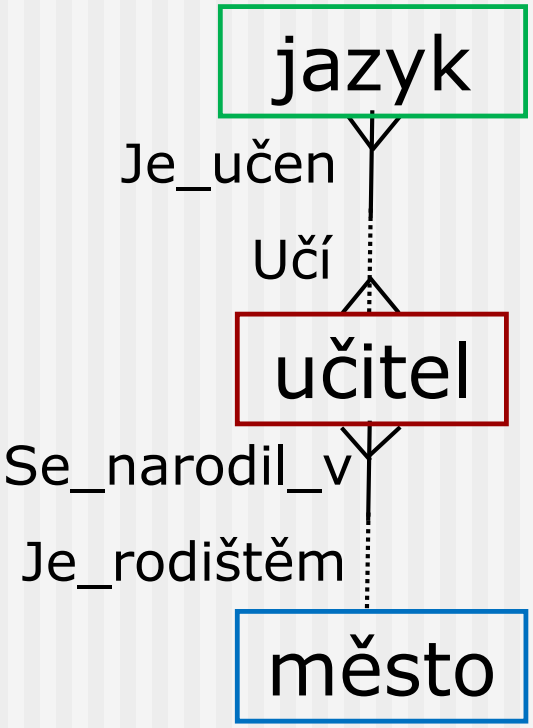
  - Příklady atributů:
    - Název pro jazyk, Rok_narození, #U_ID pro učitele
    - Místnost pro učí
- „slabé“ schéma GDB
  - neobsahuje kardinality,
    - Učí učitel více jazyků?
    - Může jazyk učit více učitelů?
  - neobsahuje závislosti mezi atributy,
  - obsahuje částečně: domény atributů, omezení atributů.



Modelování grafových databází

- Grafové konceptuální schéma

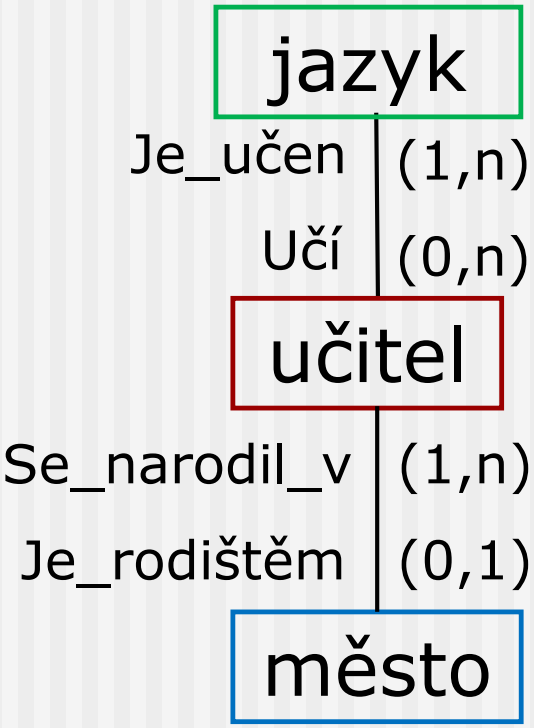




Modelování grafových databází

- Grafové konceptuální schéma

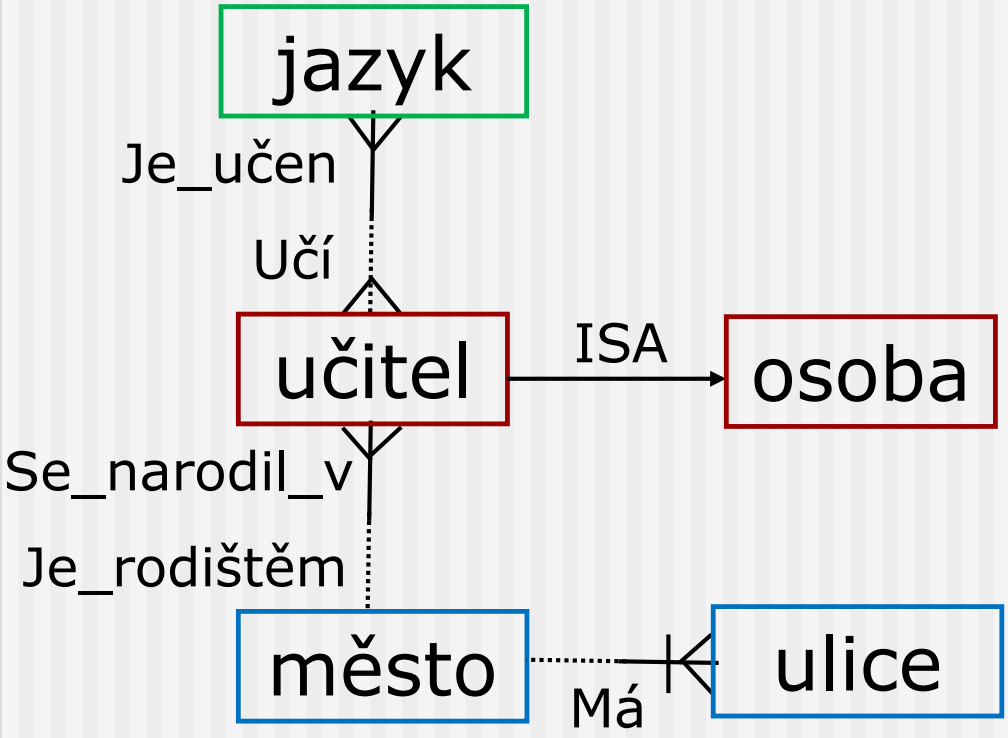
varianta s min-max IO



Modelování grafových databází

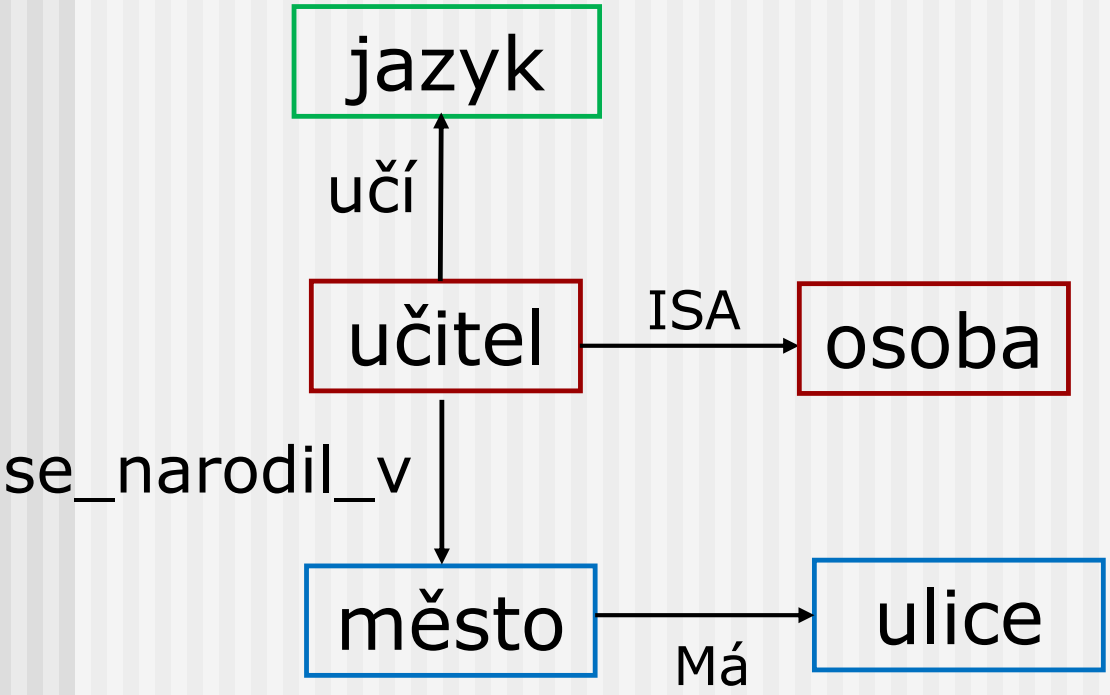
- Grafové konceptuální schéma

varianta s ISA-hierarchiemi a slabými entitami



Modelování grafových databází

- odpovídající schéma grafové databáze
  varianta s ISA-hierarchiemi a slabými entitami



Modelování grafových databází

- (databázové) integritní omezení pomocí vzoru
- Př.: GSŘBD GRAD (má databázový model, nepožívá schéma, umožňuje IO pomocí vzorů)

  IO: každý učitel učící němčinu se narodil po roce 1980.

Literatura: Ghrab, A., Romero, O., Skhiri, S., Vaisman, A., Zimányi, E.: GRAD: On Graph Database Modeling. Cornel University Library, arXiv:1602.00503, 2016.

Zobrazení: konceptuální → databázové

- Převede graf konceptuálního schématu na graf databázového schématu.
- Zobrazení není jednoznačné. Je třeba se rozhodnout:
  - pro orientaci šipky v typu vztahu
  - pro název značky hrany
- U ISA a slabých entit je třeba uvažovat odlišné klíče typů entit (dědí z vyšších členů hierarchie, ze silných typů entit),

Schémata a praxe

Přístup ke schématu grafové databáze v Neo4j:

- Nemá schéma.
- Dílčí možnosti definovat IO:
  - CREATE CONSTRAINT ON (učitel:Učitel)
    ASSERT učitel.#U_ID IS UNIQUE,
  - CREATE CONSTRAINT ON (učitel:Učitel)
    ASSERT exists(učitel.Rok_narození)
    stanovuje, že všechny uzly s jistou značkou mají jistou vlastnost
  - CREATE CONSTRAINT ON ()-[učí:Učí]-()
    ASSERT exists(učí.Místnost)
    stanovuje, že všechny hrany s jistou značkou mají jistou vlastnost

Schémata a praxe

Přístup ke schématu grafové databáze v GSŘBD OrientDB:

- Role schématu může být přesně specifikována
  - *plné schéma* - umožňuje „striktní režim“ na úrovni tříd a množin povinných polí.
  - *bez schématu* – umožňuje třídy bez vlastností. Implicitní je „nestriktní režim“, což znamená, že záznamy mohou mít libovolná pole.
  - *schéma hybridně* – umožňuje třídy s danými poli, ale i záznamy, ve kterých jsou pole definovaná uživatelem. Tomu se někdy říká *smíšené schéma*.
- Schéma grafové databáze může být někdy nevýhodné, např. v dynamickém prostředí, rychle se měnící aplikační doméně apod.

Závěr

Problémy v oblasti GDB a výzvy pro další výzkum

- návrh GDB
- potřeba benchmarků
- vývoj heuristik pro některé obtížné grafové problémy
- pattern matching grafů
- komprese grafů
- integrace grafových dat
- vizualizace grafů
- zpracování proudu grafových dat

Reference

1. Sasha et al: Algorithmics and Applications of Tree and Graph Searching PODS'02
2. Cypher: http://docs.neo4j.org/chunked/stable/cypher-query-lang.html
3. Cypher Cheat Sheet http://assets.neo4j.org/download/Neo4j_CheatSheet_v3.pdf
4. The Neo4 Manual: https://cs.brown.edu/courses/cs227/papers/neo4j.pdf
5. Pokorný, J., Snášel, V.: Big Graph Storage, Processing and Visualization. Chapter 12 in: Graph-Based Social Media Analysis, I. Pitas (Ed.), Chapman and Hall/CRC, pp. 403 – 430, 2015.