**Redukce datových modelů**

Řimnáč, Martin
2007

# Redukce datových modelů

*doktorand:*
ING. MARTIN ŘIMNÁČ
Ústav informatiky AV ČR, v. v. i.
Pod Vodárenskou věží 2
182 07 Praha 8
rimnacm@cs.cas.cz

*školitel:*
ING. JÚLIUS ŠTULLER , CSC.
Ústav informatiky AV ČR, v. v. i.
Pod Vodárenskou věží 2
182 07 Praha 8
stuller@cs.cas.cz

obor studia:
Databázové systémy

Práce byla podpořena projektem č. 1ET100300419 programu Informační společnost (Tématického programu II Národního programu výzkumu v ČR: Inteligentní modely, algoritmy, metody a nástroje pro vytváření sémantického webu), záměrem AV0Z10300504 “Computer Science for the Information Society: Models, Algorithms, Applications” a projektem 1M0554 Ministersva školství, mládeže a tělovýchovy ČR ”Pokročilé sanační technologie a procesy”.

**Abstrakt**

Příspěvek se zabývá aspekty optimalizace paměťových nároků binárního úložiště atributově anotovaných dat na základě transitivní redukce systému funkčních závislostí. Tento systém buď může být předem daný modelem, v tomto případě se ukazuje, že je možné optimalizaci použít jednorázově; a nebo tento model je inkremetálním způsobem odhadován a pak je vhodné již jednou naoptimalizované úložiště pouze upravovat opět inkrementálním způsobem. V poslední sekci se příspěvek zaobírá rozborem nejednoznačnosti výsledku včetně detailního rozboru vlastností základních konfigurací částí modelu způsobující tuto nejednoznačnost. V neposlední řadě je analyzována složitost dílčích operací v úložišti.

## 1. Úvod a motivace

Studium principů paměti je v rámci psychologie studováno od středověku, 19. století přináší první experimenty s pamětí (Ebbinghaus), které byly následovány experimenty s pamětí v širším pojetí (Pavlov, Thorndike, Lashley), převážně z hlediska schopnosti učení se - již v této době byla studována vazba mezi pamětí jako prostředkem pro uchování znalostí a procesy učení jako způsoby pro vkládání nových znalostí. Paměť je obvykle dělena podle nejrůznějších kritérií, například podle trvání uchování znalosti (krátkodobá, dlouhodobá paměť), či podle její funkce (asociativní, sémantická, fonologická smyčka); ukazuje se však, že paměť funguje spíše jako jeden celek nežli propojení více bloků, každý odděleně reprezentující určitý typ paměti [1].

Mnohé z těchto hypotéz a experimentálních výsledků lze zúročit v oblasti umělé inteligence. Jednou z takových podoblastí je studium schopnosti paměti samočinně strukturovat uložené znalosti. Zcela jistě paměť musí vážit mezi svou kapacitou a časem, za který je možné hledanou znalost vybavit.

Stejným směrem se vydává i tento příspěvek, avšak místo paměti lidské používá pro paměť formalismu binárních matic [2], které mají přímou návaznost na formáty sémantického webu [3].

Předpokládejme, že uvažujeme atributově anotovaná data se známou strukturou popsanou schématem, či v případě, že schéma není dostupné, odhadnutou z dat [4, 5, 6]. Tato struktura nechť pokrývá minimálně množinu atributů včetně jejich aktivních domén a množinu platných funkčních závislostí; samotná data jsou pak uložena v úložišti jako instance funkčních závislostí. V mnohých případech je možné na základě tranzitivních pravidel redukovat množinu všech platných funkčních závislostí a tím i počet jejich instancí. Tato redukce podstatným způsobem ovlivňuje efektivitu uložení znalostí a tedy i kapacitu paměti nutnou pro uchování předmětných dat.

Mezními případy jsou reprezentace:

- *s minimální dobou vybavení* - pokrývající všechny platné funkční závislosti, příp. jejich instance
- *s minimálními nároky na paměť* - pokrývající minimální počet instancí funkčních závislostí, avšak bez možnosti dosažení finálního výsledku v jednom kroku

Tato problematika je známa z teorie grafů jako tranzi-

tivní redukce, příp. uzávěr grafu [7, 8]. Poznamenejme, výsledek tranzitivní redukce je nejednoznačný.

Příspěvek proto zavádí přídavné kritérium, jenž zajišťuje optimalizaci úložiště s ohledem na minimální nároky na kapacitu paměti při zachování veškerých znalostí. Takovou redukci je možné provést jak na úrovní instancí, tak na úrovni zobecněných popisů, v případě tohoto příspěvku redukcí množiny platných funkčních závislostí.

V případě, že použijeme inkrementální algoritmus pro odhad struktury dat [2], je nutné při jakékoli změně provést reoptimalizaci. Z tohoto důvodu příspěvek navrhuje detekovat pouze neoptimální části a provést nápravu inkrementálním způsobem.

Poslední část příspěvku ukazuje, že požadavek na minimální paměťové nároky sám o sobě nevede na jednoznačné řešení optimalizační úlohy. Víceznačnost výsledku může být omezena použitím libovolné z předepsaných konfigurací. Dílčí vlastnosti těchto konfigurací jsou analyzovány a následně porovnány.

### 1.1. Binární úložiště dat

*Úložiště dat* je systém pro uchování a následné vyhledávání dat. Úložiště $\mathscr{R}$ vedle samotných dat $\mathscr{I}$ obsahuje i jejich model $\mathscr{M}$. Úložiště, jehož všechna data $\mathscr{I}$ splňují požadavky kladené modelem $\mathscr{M}$, se nazývá *konzistentní*.

*Binární úložiště* atributově anotovaných dat je úložiště, jehož data jsou uložena pomocí množiny implikací mezi elementy $e \in \mathscr{E}$ - dvojicemi atribut hodnota. Model $\mathscr{M}$ pro potřeby tohoto příspěvku zahrnuje:

- množinu atributů $\mathscr{A}$,
- množinu hodnot $\mathscr{D}$
- množinu aktivních domén jednotlivých atributů $\{\forall A \in \mathscr{A} : \mathscr{D}_\alpha^{\mathscr{R}}(A) \subseteq \mathscr{D}\}$
- množinu platných (unárních) funkčních závislostí $\mathscr{F} \subseteq \mathscr{A} \times \mathscr{A}$, na jejichž základě je možné jednoznačně dovodit hodnotu atributu na pravé straně z hodnoty atributu na levé straně.

Přepokládejme, že pro každý element $e \in \mathscr{E}$, atribut $A \in \mathscr{A}$ a hodnotu $v \in \mathscr{D}$ existuje index ke každému prvku jednoznačně přiřazující přirozené číslo. Pak je možné nadefinovat úložiště pomocí binárních matic namísto množin:

- matici instancí $\Phi = [\phi_{ij}]$

$$\phi_{ij} = \begin{cases} 1 & \text{pokud } e_i \rightarrow e_j \in \mathscr{I} \\ 0 & \text{jinak} \end{cases} \quad (1)$$

- matici funkčních závislostí $\Omega = [\omega_{ij}]$

$$\omega_{ij} = \begin{cases} 1 & \text{pokud } A_i \rightarrow A_j \in \mathscr{F} \\ 0 & \text{jinak} \end{cases} \quad (2)$$

- matici aktivních domén atributů $\Delta = [\delta_{ij}]$

$$\delta_{ij} = \begin{cases} 1 & \text{pokud } v_j \in \mathscr{D}_\alpha(A_i) \\ 0 & \text{jinak} \end{cases} \quad (3)$$

Úložiště je pak možné definovat jako:

$$\mathscr{R} = [\Phi, \mathscr{M}], \text{ kde } \mathscr{M} = [\Omega, \Delta, \mathscr{A}, \mathscr{D}] \quad (4)$$

Úložiště je konzistentní, pokud implikace $\Phi$ pokrývají pouze instance funkčních závislostí $\Omega$, tedy

$$\forall \phi_{ij} = 1 : \phi'_{ij} = 1, \text{ kde } [\phi'_{ij}] = \Delta\Omega\Delta^T \quad (5)$$

Objekty nechť jsou popsány pomocí množiny elementů $t \subset \mathscr{E}$. V dalším textu se omezme na popis objektů stejného druhu, tj. každý objekt je popsán pomocí všech atributů $A \in \mathscr{A}$ a prázdné hodnoty nejsou přípustné. Navíc (jako silnější podmínku) požadujme, aby každý atribut byl v záznamu $t$ pokryt právě jedním elementem (tj. v rámci záznamu nelze dělit neatomické atributy)

$$\forall t : ||t|| = ||\mathscr{A}|| \quad (6)$$
$$\forall t : \forall e_i \in t \nexists e_j \in t, e_j \neq e_i : \mathscr{A}(e_i) = \mathscr{A}(e_j)$$

Toto omezení mimo jiné přináší:

$$\forall \omega_{ij} = 1 : \mathscr{D}_\alpha(A_i) \geq \mathscr{D}_\alpha(A_j) \quad (7)$$

### 1.2. Tranzitivní redukce

Vztah mezi dvěma obecnými prvky je tranzitivní, pokud platnost vztahu mezi prvky $[i, j]$ a $[j, k]$ implikuje platnost vztahu mezi prvky $[i, k]$. Pakliže na matici instancí budeme pohlížet jako na incidenční matici grafu, lze tuto úlohu převést do teorie grafů - na hledání tranzitivního uzávěru příslušného grafu. Tímto způsobem je možné docílit minimalizace počtu prvků [9], které musí být uloženy, aniž by došlo ke ztrátě dat. Úloha hledání takové podmožiny se nazývá tranzitivní redukce, úloha inverzní (rekonstruující z redukce úplnou množinu) se nazývá hledání tranzitivního uzávěru. Poznamenejme, že výsledek transitivní redukce není jednoznačný.

Jak množina funkčních závislostí $\mathscr{F}$, tak díky (5) i množina instancí konzistentního úložiště splňují podmínku transitivity. Jednoduše lze nahlédnout, že postačuje redukovat pouze funkční závislosti a následně ponechat

pouze instance těch funkčních závislostí, které odpovídají tranzitivního uzávěru.

Pro binární matice je možné tranzitivní uzávěr vyjádřit jako mocnění redukované matice:

$$X = (X^\flat)^\kappa \qquad (8)$$
$$\kappa = \arg\min_k \{k : (X^\flat)^k = (X^\flat)^{k+1}\}$$

Parametr $\kappa$ představuje počet kroků nutných ke získání plné formy matice (odpovídající tranzitivnímu uzávěru zaručující dosažitelnost výsledku v jednom kroce).

Hledání tranzitivního uzávěru postupným násobením matic je značně neefektivní, složitost operace je $\mathscr{O}(|X|^{\kappa+2})$. Lepším způsobem je hledání uzávěru pomocí:

**Algoritmus 1** *Tranzitivní uzávěr*

*for* $\forall k = 1 \ldots |X|$
  *for* $\forall j = 1 \ldots |X|$
    *for* $\forall i = 1 \ldots |X|$
      *if* $x_{ij} = 1 \wedge x_{jk} = 1$ *then*
        $x_{ik} = 1$;

Tento algoritmus má složitost již nezávislou na parametru $\kappa$: $\mathscr{O}(|X|^3)$, případně přesněji $\mathscr{O}((\sum_{\forall i,j} x_{ij})^2)$.

## 2. Optimalizace úložiště

Mějme úložiště $\mathscr{R}$ naplněné daty $\Phi$ odpovídající instancím funkčních závislostí $\Omega$ a předpokládejme, že model obsahuje všechny (tj. tranzitivní uzávěr) platné funkční závislosti. Pokusme se nyní optimalizovat úložiště tak, aby pro uložení všech dat bylo potřeba minimálního počtu instancí, tedy minimalizujeme

$$||\mathscr{I}|| = ||\Phi|| = \sum_{\forall i,j} \phi_{ij} \qquad (9)$$

### 2.1. Počet instancí

Diskutujme nyní počet instancí funkčních závislostí.

Mějme funkční závislost $f = (A_i \rightarrow A_j) \in \mathscr{F}$ reprezentovanou v binární matici jako $\omega_{ij} = 1$. Blok $\mathbf{1}$ v matici $\Delta\Omega_{ij}\Delta^T$, který odpovídá[1] prostoru této funkční závislosti, představuje její všechny možné instance. Takových instancích je celkem

$$||\Delta\Omega_{ij}\Delta^T|| = ||\mathscr{D}(A_i)|| \cdot ||\mathscr{D}(A_j)|| \qquad (10)$$

[1] $\Omega_{ij} = [\omega_{i'j'}] : \omega_{i'j'} = 1$ pokud $i = i', j = j'$, jinak $\omega_{i'j'} = 0$

Díky informaci o existenci funkční závislosti $f$ je možné na základě hodnoty atributu na levé straně jednoznačně určit hodnotu atributu na straně pravé; tedy z tohoto počtu možných instancí je přípustných pouze $||\mathscr{D}(A_i)||$. Jinými slovy, znalost o existenci funkční závislosti $f$ redukuje počet přípusných instancí ze všech možných celkem o

$$\frac{||\mathscr{D}(A_i)||}{||\Delta\Omega_{ij}\Delta^T||} = \frac{1}{||\mathscr{D}(A_j)||}\ \% \qquad (11)$$

Zobecníme-li tuto úvahu na celou množinu funkčních závislostí, pak

$$||\Phi|| = \sum_{\forall\omega_{ij}=1} ||\mathscr{D}(A_i)|| \qquad (12)$$

### 2.2. Vliv znalosti funkčních závislostí

Znalost množiny funkční závislostí $\mathscr{F}$ není pro úložiště $\mathscr{R}$ principielně nutná; odpovídající minimální funkcionalitu by úložiště pokrývalo i za předpokladu, že každý objekt by byl uložen jako množina implikací mající klíčový element, jednoznačně definující konkrétní objekt, na levé straně a elementy popisující vlastnosti objektu na straně pravé.

$$\mathscr{F}' = \{\forall A \in \mathscr{A} : A_{\text{PK}} \rightarrow A\} \qquad (13)$$

Bude-li takové úložiště obsahovat popis $m$ objektů, každý popsán stejnými atributy $\mathscr{A}$, pak počet uložených instancí bude:

$$||\Phi'|| = \sum_{\forall\omega_{ij}=1} = ||\mathscr{A}|| \cdot m = ||\mathscr{A}|| \cdot ||\mathscr{D}_\alpha(A_{\text{PK}})|| \qquad (14)$$

Poměr mezi počtem instancí takového úložiště a úložištěm zohledňujícími funkční závislosti bude označen

$$\nu = \frac{||\Phi||}{||\Phi'||} \qquad (15)$$

Tento poměr bude příznivý (tj. $\nu < 1$), pakliže množina funkčních závislostí nebude obsahovat žádné redundantní závislosti.

Ilustrujme výpočet poměru na umělém příkladu. Mějme množinu atributů $A_{GK} \in \mathscr{A}$. Nechť jsou tyto atributy rozděleny do $g$ disjunktních skupin $\mathscr{A}_G$ po $k$ atributech a nechť aktivní doména všech attributů v jedné skupině je stejná a její velikost je odvozena od velikosti předchozí skupiny, která je $u$ krát větší. Velikost aktivních domén

atributů v první skupině je $d$. Množina funkčních závislostí $\mathscr{F}$ je

$$\{\forall A_{1i} \in \mathscr{A}_1 : A_{\text{PK}} \rightarrow A_i\} \cup \bigcup_{\forall l<g-1} \{A_{li} \rightarrow A_{(l+1)i}\} \tag{16}$$

Srovnáme-li počet instancí s (13), dostáváme

$$\nu = \frac{||\Phi||}{||\Phi'||} = \frac{||\mathscr{D}(A_{\text{PK}})|| + \sum_{v=1}^{g} kd\frac{1}{u^v}}{||\mathscr{D}(A_{\text{PK}})|| \cdot kg} \tag{17}$$

Za předpokladu, že budeme uvažovat $||\mathscr{D}(A_{\text{PK}})|| = d^k$, dostáváme

$$\nu = \frac{1}{g}\Big(\frac{1}{k} + \frac{1}{d^{k-1}\sum_{v=1}^{g} u^v}\Big) \tag{18}$$

Je patrné, že tento poměr bude velmi příhodný a tudíž lze zahrnutí znalosti funkčních závislostí více nežli doporučit. Poznamejme, že $||\mathscr{F}|| = ||\mathscr{F}'||$ a $\mathscr{F}'$ je možné za použití tranzitivních pravidel odvodit z $\mathscr{F}$.

### 2.3. Algoritmus

Hledejme nyní algoritmus, který najde minimální systém funkčních závislostí a zohledňuje i počet instancí tohoto systému. Mějme matici $\Omega$ popisující vstupní neoptimalizovaný systém funkčních závislostí, výstupem algoritmu pak je optimalizovaná matice $\Omega^\flat$.

Základní algoritmus pracuje na principu, že se z pracovní matice postupně vyjímají hrany; pakliže tranzitivní uzávěr (Algoritmus 1) nově vzniklé matice bez hrany je shodný s tranzitivním uzávěrem původní matice, je vyjmutí této hrany přípustné, v opačném případě tato hrana musí být navrácena.

Test se provede postupně pro všechny hrany. Navíc jej lze zjednodušit pouze na potvrzení možnosti odvodit vyjmutou hranu na základě okolních hran.

**Algoritmus 2** *Tranzitivní redukce*

*for* $\forall i, j : \omega_{ij} = 1$
  $\omega_{ij} = 0$; $\Omega' =$ *closure* $(\Omega)$;
  *if* $\omega'_{ij} = 0$ *then*
    $\omega_{ij} = 1$;
$\Omega^\flat = \Omega$;

Složitost tohoto algoritmu pro nalezení tranzitivní redukce je $\mathscr{O}(||\Omega||^3)$. Existují však vylepšení pro silně souvislé grafy [7], které používají pro hledání redukce procházení grafu z náhodně vybraného vrcholu do hloubky. Tyto algoritmy vykazují složitost $\mathscr{O}(||\mathscr{A}|| + ||\Omega||)$, avšak jich nelze použít, neboť systém funkčních závislostí nemusí odpovídat silně souvislému grafu - tedy pro použití této myšlenky je nutné procházet všechny vrcholy - atributy. Složitost pak naroste na $||\mathscr{A}|| \cdot \mathscr{O}(||\mathscr{A}|| + ||\Omega||)$, což principielně vede na $\mathscr{O}(||\mathscr{A}||^3)$.

Hrany mohou být vyjímany v náhodném pořadí. Pakliže budeme prioritně vyjímat funkční závislosti $f = A_i \rightarrow A_j$ s největším počtem instancí $||\mathscr{D}(A_i)||$, součástí tranzitivní redukce zůstanou funkční závislosti s minimálním počtem instancí. Za předpokladu, že vstupní matice je maximálním tranzitivním uzávěrem, vrácený systém funkčních závislostí bude pokrývat nejmenší možný počet instancí.

Složitost seřazení funkčních závislostí $\omega_{ij} = 1$ do posloupnosti dvojic indexů $[[i_1, j_1] \dots [i_{||\Omega||}, j_{||\Omega||}]]$ podle kritéria

$$||\mathscr{D}_\alpha(A_{i_u})||) > ||\mathscr{D}_\alpha(A_{i_v})||) \rightsquigarrow u \succ v \tag{19}$$

je $\mathscr{O}(||\Omega|| \log(||\Omega||))$.

**Algoritmus 3** *Tranzitivní redukce minimalizující počet instancí*

$\Omega^\star =$ *closure* $(\Omega)$;
$o =$ *sort* $(\Omega^\star)$;
$\Omega^\flat =$ *reduce* $(\Omega^\star)$ *respecting* $o$;

Za cenu výpočtu tranzitivního uzávěru vstupní matice $\Omega$ (může být požadováno jako vstupní předpoklad algoritmu) $\mathscr{O}(||\Omega||^2)$ a cenu za seřazení $\mathscr{O}(||\Omega|| \log(||\Omega||))$ není získána libovolná přípusná redukce systému funkčních závislostí, ale taková, která potřebuje k reprezentaci celé znalostní báze minimální paměťový prostor.

### 2.4. Intensionální versus extensionální modely

Díky platnosti (7) je nutné optimalizovat úložiště $\mathscr{R}$ pouze jednou na základě funkčních závislostí $\mathscr{F}$ popsaných intensionálním modelem $M$; jakékoli přidávání dat odpovídajících modelu $M$ nemůže vést ke takové změně, která by způsobila neoptimalitu úložiště. Jinými slovy znalost všech platných funkčních závislostí je silnějším předpokladem nežli znalost velikosti (aktivních) domén atributů.

Mnohé zdroje, zvláště pak v prostředí webu, poskytují pouze data bez jakéhokoli popisu (modelu). Zvláštním případem jsou data bez uvedeného intensionálního modelu, avšak (většinou pomocí atributové anotace) schopné extensionální model explicitně popsat. V tomto případě hovoříme o metodě odhadu modelu - struktury dat.

Tyto metody (označované jako Functional Dependency Discovery) se snaží nejlepším možným způsobem odhadnout systém funkčních závislostí. Podle postupného hledání protipříkladů rozdělují množinu všech možných unárních funkčních závislostí $\overline{\mathscr{F}} = \mathscr{A} \times \mathscr{A}$ na:

- *porušené funkční závislosti* $\mathscr{F}_c$. Za předpokladu bezchybnosti vstupních dat porušená funkční závislost v jakékoli extensi nemůže být součástí intensionálního modelu. Jinými slovy, po nalezení protipříkladu lze s jistotou tvrdit, že daná funkční závislost neplatí. Matici porušených funkčních závislostí budeme označovat $\mho$.

- *neporušené funkční závislosti* $\mathscr{F}$. U těchto funkčních závislostí dosud nebyl nalezen protipříklad, který by je porušil. O těchto funkčních závislostech se můžeme domnívat, že jsou platné (a tedy součástí intensionálního modelu).

Díky neznalosti množiny funkčních závislostí jsou velikosti aktivních domén jedinou použitelnou známou charakteristikou. Ta se může v průběhu vkládání dat do úložiště měnit - její změny často vedou na základě (7) k získání protipříkladu s následnou detekcí porušení funkční závislosti a tím ke změně modelu. Nový model je potřeba znovu optimalizovat.

### 2.5. Inkrementální verze algoritmu

Použití algoritmu 3 je vhodné pro jednorázovou optimalizaci. Předpokládejme, že model se nebude měnit radikálně, vždy zůstane nějaká část beze změny. Diskutujme nyní navržení úprav optimalizovaného uložiště tak, aby bylo optimální i po vložení nového objektu.

Neoptimalita se často odvijí od porušení funkční závislosti $\omega_{ij} = 1$ z redukovaného systému. Tato porušená funkční závislost bude následně vyjmuta ze systému, tj. $\omega'_{ij} = 0$, avšak je nutné zachovat funkční závislosti, které byly pomocí této závislosti odvoditelné na základě tranzitivních pravidel:

$$\forall k : \omega_{jk} = 1 \rightsquigarrow \omega'_{ik} = 1$$
$$\forall l : \omega_{li} = 1 \rightsquigarrow \omega'_{lj} = 1$$

Tento proces probíhá do okamžiku, kdy už žádná z funkčních závislostí v redukovaném systému není porušená.

Je možné ukázat, že nový redukovaný systém $\Omega'$ je optimální možný, pokud před započatím detekce porušených funkčních závislostí byl optimální vůči již aktualizovaným velikostem aktivních domén atributů.

## 3. Nejednoznačnost transitivní redukce

Nalezení tranzitivní redukce je obecně úloha s víceznačným řešením. Z tohoto důvodu je vhodné použít nějaké doplňující kritérium.

Za předpokladu, že použijeme kritérium minimalizující počet instancí, výsledek bude jednoznačný až na atributy spojené s funkčními závislostmi, které tvoří cyklus libovolné délky. Takové skupiny atributů budeme nazývat komponentami:

$$A_i \in \mathscr{C}(A_j) \subseteq \mathscr{A} \text{ if } \exists A_j \in \mathscr{C} : \omega_{ij} = \omega_{ji} = 1 \quad (20)$$

Pokud každou z komponent $\mathscr{C}(A_j)$ nahradíme jedním reprezentativním atributem $A_j$, pak výsledek tranzitivní redukce s minimálním počtem instancí je jednoznačný. Jinými slovy nejednoznačnost celého výsledku je způsobena nejednoznačností tranzitivní redukce komponent včetně nejednoznačnosti výběru reprezentativního atributu, jenž komponentu propojuje s ostatními.

Obecně existuje mnoho konfigurací, jak popsat vztahy uvnitř komponenty. Tři základní, lineární, hvězdicivou a cyklus popišme detailněji, všechny ostatní jsou nějakou kombinací těchto základních konfigurací.

Před výčtem vlastností dílčích konfigurací poznamenejme, že jako důsledek (7) a (20) platí:

$$\forall \mathscr{C} \, \forall A_i, A_j \in \mathscr{C} : ||\mathscr{D}_\alpha(A_i)|| = ||\mathscr{D}_\alpha(A_j)|| = \epsilon \quad (21)$$

Dále, vlastnosti konfigurace $\Omega^X_{\mathscr{C}} = [\omega^\flat_{ij}]$ se odvijí od vstupních a výstupních stupňů vrcholů odpovídající atributům při zachování vzájemné odvoditelnosti mezi všemi atributy v komponentě reprezentované tranzitivním uzávěrem $\Omega^\star_{\mathscr{C}}$. Proto tento účel zaveďme kritérium na počet vrcholů (atributů) mající předepsaný součet stupňů :

$$\mu(\beta) = ||\{\forall A_i \in \mathscr{C} : \sum_{\forall k} \omega^\flat_{ik} + \omega^\flat_{ki} = \beta\}|| \quad (22)$$

### 3.1. Lineární konfigurace

Lineární konfigurace představuje takovou redukci $\Omega^L_{\mathscr{C}}$, která je symetrická a maximalizuje počet vrcholů $\mu(4)$ mající součet stupňů součet stupňů roven $4$:

$$\arg\max_{\mu(4)} \{\Omega^L_{\mathscr{C}} = (\Omega^L_{\mathscr{C}})^T \wedge (\Omega^L_{\mathscr{C}})^{||\mathscr{C}||} = \Omega^\star_{\mathscr{C}}\} \quad (23)$$

Výsledkem takové minimalizace je matice, jejíž prvky jsou:

$$\omega^L_{ij} = \begin{cases} 1 & i = j+1 \vee j = i+1 \\ 0 & \text{jinak} \end{cases} \quad (24)$$

Taková matice pak vykazuje:

| | | |
|---|---|---|
| Maximální počet iterakcí | $\kappa$ | $\|\|\mathscr{C}\|\| - 1$ |
| Počet funkčních závislostí | $\|\|\Omega^L_{\mathscr{C}}\|\|$ | $2(\|\|\mathscr{C}\|\| - 1)$ |
| Počet instancí | $\|\|\Phi\|\|$ | $2\epsilon(\|\|\mathscr{C}\|\| - 1)$ |
| Kompresní poměr | $\nu^L$ | $\frac{2\epsilon(\|\|\mathscr{C}\|\|-1)}{\epsilon(\|\|\mathscr{C}\|\|-1)} = 2$ |
| Délka minimálního cyklu | $\sigma^{\ominus}$ | 2 |
| Délka maximálního cyklu | $\sigma^{\oplus}$ | $2(\|\|\mathscr{C}\|\| - 1)$ |
| Maximální počet vložení | $\theta^{\oplus}$ | 0 |
| Maximální počet vyjmutí | $\theta^{\ominus}$ | $\kappa$ |

**Tabulka 1:** Vlastnosti lineární konfigurace

### 3.2. Hvězdicová konfigurace

Hvězdicová konfigurace představuje takovou redukci $\Omega^S_{\mathscr{C}}$, která je symetrická a maximalizuje počet vrcholů $\mu(2)$ se součtem stupňů 2:

$$\arg\max_{\mu(2)}\{\Omega^L_{\mathscr{C}} = (\Omega^S_{\mathscr{C}})^T \wedge (\Omega^S_{\mathscr{C}})^{||\mathscr{C}||} = \Omega^{\star}_{\mathscr{C}}\} \quad (25)$$

Výsledkem takové minimalizace je matice, jejíž prvky jsou:

$$\omega^S_{ij} = \begin{cases} 1 & i = 1 \vee j = 1 \\ 0 & \text{jinak} \end{cases} \quad (26)$$

Taková matice pak vykazuje:

| | | |
|---|---|---|
| Maximální počet iterakcí | $\kappa$ | 2 |
| Počet funkčních závislostí | $\|\|\Omega^L_{\mathscr{C}}\|\|$ | $2(\|\|\mathscr{C}\|\| - 1)$ |
| Počet instancí | $\|\|\Phi\|\|$ | $2\epsilon(\|\|\mathscr{C}\|\| - 1)$ |
| Kompresní poměr | $\nu^L$ | $\frac{2\epsilon(\|\|\mathscr{C}\|\|-1)}{\epsilon(\|\|\mathscr{C}\|\|-1)} = 2$ |
| Délka minimálního cyklu | $\sigma^{\ominus}$ | 2 |
| Délka maximálního cyklu | $\sigma^{\oplus}$ | 4 |
| Maximální počet vložení | $\theta^{\oplus}$ | $(\|\|\mathscr{C}\|\| - 2)$ |
| Maximální počet vyjmutí | $\theta^{\ominus}$ | $(\|\|\mathscr{C}\|\| - 1)$ |

**Tabulka 2:** Vlastnosti hvězdicové konfigurace

### 3.3. Konfigurace cyklus

Konfigurace cyklus představuje takovou redukci $\Omega^C_{\mathscr{C}}$, která narozdíl od předchozích není symetrická a maximalisuje počet vrcholů $\mu(2)$ mající součet stupňů 2:

$$\arg\max_{\mu(2)}\{(\Omega^C_{\mathscr{C}})^{||\mathscr{C}||} = \Omega^{\star}_{\mathscr{C}}\} \quad (27)$$

Výsledkem takové minimalizace je matice, jejíž prvky jsou:

$$\omega^C_{ij} = \begin{cases} 1 & i = j + 1 \\ 1 & i = ||\mathscr{C}|| \wedge j = 1 \\ 0 & \text{jinak} \end{cases} \quad (28)$$

Taková matice pak vykazuje:

| | | |
|---|---|---|
| Maximální počet iterakcí | $\kappa$ | $\|\|\mathscr{C}\|\| - 1$ |
| Počet funkčních závislostí | $\|\|\Omega^C_{\mathscr{C}}\|\|$ | $\|\|\mathscr{C}\|\|$ |
| Počet instancí | $\|\|\Phi\|\|$ | $\epsilon\|\|\mathscr{C}\|\|$ |
| Kompresní poměr | $\nu^C$ | $\frac{\epsilon(\|\|\mathscr{C}\|\|)}{\epsilon(\|\|\mathscr{C}\|\|-1)} \rightsquigarrow 1$ |
| Délka minimálního cyklu | $\sigma^{\ominus}$ | $\|\|\mathscr{C}\|\|$ |
| Délka maximálního cyklu | $\sigma^{\oplus}$ | $\|\|\mathscr{C}\|\|$ |
| Maximální počet vložení | $\theta^{\oplus}$ | $(\|\|\mathscr{C}\|\| - 2)$ |
| Maximální počet vyjmutí | $\theta^{\ominus}$ | $(\|\|\mathscr{C}\|\| - 1)$ |

**Tabulka 3:** Vlastnosti konfigurace cyklus

### 3.4. Porovnání konfigurací

Na základě porovnání parametrů uvedených v předchozích tabulkách je možné zobecnit poznatky o volbě konfigurace.

Mezi výhody lineární konfigurace patří fakt, že počet funkčních závislostí redukovaného systému je stejně jako u počtu všech odvoditelných funkčních závislostí v průběhu procesu odhadování nerostoucí. Jinými slovy tato konfigurace nevyžaduje tak časté odvozování instancí nově vložených funkčních závislostí - pouze porušené nahrazuje platnými. Druhou výhodou je, že délka minimálního cyklu je 2. Toho lze s výhodou využít při detekci cyklu. Cykly mohou způsobovat nežádoucí postupné deaktivace elementů při dotazování, jenž využívá zobecnění binárních matic na matice s hodnotami prvků z celého intervalu $< 0, 1 >$.

Naopak bezespornou výhodou hvězdicové konfigurace je minimální počet kroků nutných k dosažení finálního výsledku. Tento počet se týká jak fáze vyhledávání, tak iterakcí pro detekci porušených funkčních závislostí ve fázi odhadu struktury dat. Tato výhoda je však zaplacena množstvím možných úprav (vložení) při detekci porušené funkční závislosti.

Konfigurace cyklus vykazuje výhodné vlastnosti předchozích dvou, navíc se kompresní poměr blíží 1 (oproti 2 u předchozích konfigurací způsobený požadavkem na symetrii), avšak konfiguraci tvoří jeden cyklus délky $\kappa = ||\mathscr{C}||$. Tedy tato konfigurace je velmi vhodná pro ukládání dat do úložiště, avšak dotazování potřebuje největší počet iterací a detekce cyklů pro účely zmíněné výše je výpočetně náročná.

## 4. Závěr

Článek se zabývá možnostmi redukcí počtu instancí funkčních závislostí a to jak v případě, že tyto funkční závislosti jsou popsány v intensionálním modelu, tak v průběhu procesu odhadování modelu z dat na extensionální úrovni. Ukazuje se, že optimalizaci postačí v prvním případě provést jednorázově, avšak v případě po-

stupného odhadování vlastností dat je nutné optimalizaci provést po vložení každého nového záznamu.

Z tohoto důvodu je navrženo detekovat pouze neoptimální části a ty následně optimalizovat. Tyto neoptimalizované části jsou vždy svázány s porušením některé z funkčních závislostí (takové jsou jako neplatné z modelu postupně vyjímány). Příspěvek z tohoto důvodu zavádí pojem komponent - porušení funkční závislosti v komponentě vede na její rozdělení.

Druhotně příspěvek ukazuje, jak přesnější znalost o struktuře dat vede ke snížení počtu víceznačných řešení optimalizační úlohy. Vedle toho na umělém příkladu ukazuje výhody znalosti platných funkčních závislostí pro efektivní uložení dat do úložiště.

**Literatura**

[1] A. Baddeley, *Vaše paměť*. Books, 1999.

[2] M. Řimnáč, "Data structure estimation for rdf oriented repository building," *cisis*, vol. 0, pp. 147–154, 2007.

[3] G. Antoniou and F. v Harmelen, *A Semantic Web Primer*. MIT Press, 2004.

[4] H. Mannila and K. Räithä, "Design by example: An applications of armstrong relations.," *Journal of computer and system sciences*, vol. 33, pp. 129–141, 1986.

[5] H. Mannila and K. Räithä, "Algorithms for inferring functional dependencies from relations," *Data & Knowledge Engineering*, vol. 12, pp. 83–99, 1994.

[6] P. A. Flach and I. Savnik., "Database dependency discovery: A machine learning approach," *AI Communications*, vol. 12/3, pp. 139–160, 1999.

[7] K. Simon, "Finding a minimal transitive reduction in a strongly connected digraph within linear time," in *WG '89: Proceedings of the 15th International Workshop on Graph-Theoretic Concepts in Computer Science*, (London, UK), pp. 245–259, Springer-Verlag, 1990.

[8] K. Simon, "On minimum flow and transitive reduction," in *ICALP '88: Proceedings of the 15th International Colloquium on Automata, Languages and Programming*, (London, UK), pp. 535–546, Springer-Verlag, 1988.

[9] M. Kryszkiewicz, "Concise representations of association rules," in *Pattern Detection and Discovery: ESF Exploratory Workshop*, vol. LNCS 2447, 2002.