národní
úložiště
šedé
literatury

Odhadování struktury a asociativní úložiště dat

Řimnáč, Martin
2006

Dostupný z http://www.nusl.cz/ntk/nusl-35654

Dílo je chráněno podle autorského zákona č. 121/2000 Sb.

Tento dokument byl stažen z Národního úložiště šedé literatury (NUŠL).

Datum stažení: 31.07.2024

Další dokumenty můžete najít prostřednictvím vyhledávacího rozhraní nusl.cz .

Odhadování struktury a asociativní úložiště dat

doktorand:
ING. MARTIN ŘIMNÁČ

Ústav informatiky AV ČR
Pod Vodárenskou věží 2
182 07 Praha 8

Fakulta elektrotechnická
České vysoké učení technické
Technická 2

166 27 Praha 6 - Dejvice

rimnacm@cs.cas.cz

školitel:
ING. JÚLIUS ŠTULLER, CSC.

Ústav informatiky AV ČR
Pod Vodárenskou věží 2

182 07 Praha 8

stuller@cs.cas.cz

obor studia:
Databázové systémy

Práce byla podpořena projektem 1ET100300419 programu Informační společnost "Inteligentní modely, algoritmy, metody a nástroje pro vytváření sémantického webu" a částečně i výzkumným záměrem AV0Z10300504 "Informatika pro informační společnost: Modely, algoritmy, aplikace".

Abstrakt

Odhad struktury dat získaných například z webových zdrojů lze využít jednak pro uložení dat, tak pro netriviální dotazování nad těmito daty. Článek rozšiřuje metodu odhadu struktury dat získávající odpovídající schéma relačního modelu ze vstupních dat a popisuje metodu uložení dat pomocí jednoduchého asociativního úložiště dat právě na základě odhadnutého modelu. Článek diskutuje dvě možné implementace úložiště: první uchovávající data jako instance funkčních závislostí, druhou uchovávající pouze instance funkčních závislosti mezi jednoduchými atributy rozšířenou o podporu komplexních atributů pomocí metainformace.

1. Úvod

Struktura dat představuje soubor všeobecně platných vztahů nad danou množinou dat, což následně vede k popisu každého prvku z této množiny jednotným způsobem na základě zvoleného modelu dat. Tento model může například vycházet z teorie relačních databází, přičemž vztahy mezi atributy jsou popisovány pomocí funkčních závislostí.
Odpovídající množinu funkčních závislostí lze získat postupným testováním, např. pomocí naivního algoritmu, přičemž se testuje vztah mezi každým prvkem z potenční množiny atributů a jedním konkrétním atributem, což vede na algoritmus s nepolynomiální složitostí.
Odpovídající data mohou být následně uložena jako množina instancí nalezených funkčních závislostí, navrhované úložiště tedy musí umožnit pokrýt i instance funkčních závislostí s komplexním atributem, což vede na komplikovanější mechanismus práce s úložištěm.

Článek popisuje návrh zjednodušení výše nastíněného úložiště inspirované formátem RDF (Resource Description Framework) [1], který je používaný v prostředí sémantického webu [2]. Výhodou tohoto formátu je jeho jednoduchost a vysoká popisovací schopnost; data jsou uložena ve formě uspořádaných trojic $< objekt, vlastnost, hodnota >$, nebo alternativně pomocí trojic $< objekt, vztah, objekt >$, odpovídající binárnímu predikátu. Data uložená v tomto formátu ze své podstaty nemusejí být funkčního charakteru.

Podobné úložiště založené na popisu funkčními závislostmi bude organizováno pomocí asociačních pravidel mezi dvěma atomickými entitami včetně pojmenování tohoto vztahu. Takové úložiště bude dále pomocí metainformací rozšířeno o podporu popisu vztahů mezi komplexními atributy bez nutnosti ukládat i odpovídající instance, což vede na redukci paměťových nároků úložiště.

Institucionální repozitář AV ČR http://hdl.handle.net/11104/0135724

Článek mimo jiné ukazuje, že takové řešení je možné použít za předpokladu, že vstupními data již obsahují nebo budou rozšířena o atomický atribut, který představuje primární klíč všech objektů.

2. Odhadování struktury dat

Motivací pro `odhadování struktury dat` [3] je efektivní uložení dat získaných pomocí specializovaných nástrojů [4, 5] automaticky extrahujících data z webové stránky. Vhodným zdrojem dat pro tyto metody bývají takzvané `produktové katalogy`; webové stránky prezentující technické informace, přičemž formát většinou odpovídá nějakému implicitnímu popisu dat.

Připomeňme, že úloha odhadu struktury dat má význam pouze na `extensionální úrovni`, tedy výsledek úlohy je vždy svázán s konkrétními vstupními daty C (extensionální úroveň výsledku je značena horním indexem, např. M^C). Naopak, definování úlohy na intesionální úrovni je principielně nemožné.

Různé metody pro odhad struktury dat jsou založeny na `naivním algoritmu` uvedeném například v [7]: algoritmus postupně prochází všechny prvky z množiny atributů $A \in \mathscr{A}^C$ (cyklus 1) implikovanou vstupními daty C a pro každý prvek z potenční množiny atributů $X \in \mathscr{P}(\mathscr{A}^C - \{A\})$ (cyklus 2) testuje existenci funkčního zobrazení $f : \mathscr{D}_\alpha^C(X) \rightarrow \mathscr{D}_\alpha^C(A)$ (test 3), přičemž symbol $\mathscr{D}_\alpha^C(A)$ značí aktivní doménu atributu A vzhledem k C. Na základě existence této funkce (na extensionální úrovni vzhledem k danému C) predikuje možnost existence funkční závislosti $(X \rightarrow A)$ na úrovni intensionální.

$$
\begin{aligned}
&M^C = \emptyset \\
&\text{for } \forall A \in \mathscr{A}^C && (1)\\
&\quad \text{for } \forall X \in \mathscr{P}(\mathscr{A}^C - A) && (2)\\
&\quad\quad \text{if } \exists \mathscr{I} : \mathscr{D}_\alpha^C(X) \rightarrow \mathscr{D}_\alpha^C(A) \text{ then} && (3)\\
&\quad\quad\quad M^C := M^C \cup \{X \rightarrow A\}
\end{aligned}
$$

Označíme-li n počet atributů v cyklu (1) a m počet objektů ve vstupní kolekci, složitost tohoto algoritmu je dána převážně velikostí potenční množiny 2^{n-1} (cyklus 2), někdy bývá diskutována [9] i složitost testu (3) funkční závislosti $O((nm) \cdot m)$. Celková složitost naivního algoritmu tedy je:

$$O(n \cdot 2^{n-1} \cdot nm^2) = O(2^{n-1}n^2m^2) \tag{4}$$

Předmětem výzkumu v této oblasti, často označované jako `functional dependency discovery` [3, 6, 7, 8, 9, 10, 11], je redukce počtu testovaných funkčních závislostí. Formálně hledáme množinu $P' \subset P = \mathscr{P}(\mathscr{A}^C - A)$, která bude implikovat shodný model jako původní P. Převážně se jedná o zavedení postupné dekompozice vedoucí k modelu bez triviálních funkčních závislostí M_T.

$$A_{i1} \rightarrow A_j \in M \rightsquigarrow A_{i2} \rightarrow A_j \in M_T \;\; \forall A_{i2} \supset A_{i1} \tag{5}$$

Někdy bývá zohledněno i hledání minimální množiny [3] funkčních závislostí $M_S \subset M$ implikující stejný model díky transitivitě funkčních závislostí.

$$A_i \rightarrow A_j \in M_S \wedge A_j \rightarrow A_k \in M_S \rightsquigarrow A_i \rightarrow A_k \in M \tag{6}$$

3. Asociativní úložiště dat

Předpokládejme, že pomocí zvolené metody odhadneme strukturu dat na základě vstupních dat C. Diskutujme nyní nejjednodušší možný způsob, jak data uložit [12].

Pro začátek neuvažujme funkční závislosti s komplexními atributy. Za tohoto předpokladu množina možných levých stran funkčních závislostí se redukuje pouze na $P' = \mathscr{A}^C$. Díky tomuto omezení

Institucionální repozitář AV ČR http://hdl.handle.net/11104/0135724

získáváme polynomiálně složitý algoritmus pro odhad modelu, navíc bez nutnosti testování funkčních závislostí s komplexními atributy.

$$O((n \cdot n) \cdot (m)m) = O(n^2 m^2) \tag{7}$$

Navrhněme úložiště Φ popisující instance funkčních závislostí M^{Φ}. Základním modelovacím prvkem zvolíme `element` představující dvojici atribut-hodnota $(A; a)$. Na základě vstupních dat přiřadíme množinu elementů:

$$E = \{e = (A; a) : \quad A \in \mathscr{A}^C, a \in \mathscr{D}_{\alpha}^C(A)\} \tag{8}$$

$$|E| \leq nm \tag{9}$$

Úložiště představuje množinu instancí funkčních závislostí, na něž můžeme nahlížet jako na asociativní pravidla mezi elementy

$$\Phi = \{e_i \rightarrow e_j : e_i, e_j \in E\} \tag{10}$$

Tato representace je nápadně podobná organizaci RDF dokumentů (pro $e_i = (A_i; a_{i'}), e_j = (A_j; a_{j'})$) a je na ni převeditelná.

$$\begin{array}{l} < A_i \qquad \text{rdf:about='}a_{i'}\text{'} > \\ \qquad\quad < A_j \text{ rdf:resource='}a_{j'}\text{'}/ > \\ < /A_i > \end{array} \tag{11}$$

Úložiště Φ je `konzistentní`, pokud

$$\forall a_{j'1} \forall a_{j'1} \nexists a_{j'2} \neq a_{j'1} : (A_i; a_{i'}) \rightarrow (A_j; a_{j'1}) \in \Phi \wedge (A_i; a_{i'}) \rightarrow (A_j; a_{j'2}) \in \Phi \tag{12}$$

Analogicky množina elementů $X \subseteq E$ je konzistentní, pokud

$$\forall a_{j'1} \forall a_{j'1} \nexists a_{j'2} \neq a_{j'1} : (A_j; a_{j'1}) \in X \wedge (A_j; a_{j'2}) \in X \tag{13}$$

Pokud úložiště Φ obsahuje pouze instance funkčních závislostí M^{Φ}, je vždy konzistentní. Pak je možné z úložiště odvodit strukturu dat na základě pravidla:

$$\forall A_i, A_j \in \mathscr{A}^C \; \exists a_{i'} \in \mathscr{D}_{\alpha}^C(A_i), a_{j'} \in \mathscr{D}_{\alpha}^C(A_j) : (A_i; a_{i'}) \rightarrow (A_j; a_{j'}) \in \Phi \rightsquigarrow A_i \rightarrow A_j \in M^{\Phi} \tag{14}$$

Zpětně, struktura dat implikuje pozice v úložišti, které mohou nést informaci.

$$\forall A_i, A_j \in \mathscr{A}^C : A_i \rightarrow A_j \notin M^{\Phi} \rightsquigarrow \forall a_{i'} \in \mathscr{D}_{\alpha}^C(A_i), \forall a_{j'} \in \mathscr{D}_{\alpha}^C(A_j) : (A_i; a_{i'}) \rightarrow (A_j; a_{j'}) \notin \Phi \tag{15}$$

Ze vztahu (14) vyplývá, že není nutné externě udržovat informaci o množině funkčních závislostí platných na úložišti Φ, neboť ji lze získat nejvýše v $O(n^2 m^2)$.

3.1. Odvozování - generalizace

Nadefinujme nyní generalizační odvozovací mechanismus jako zobrazení G:

$$G(\Phi) : E \rightarrow E \tag{16}$$

Toto zobrazení $G(\Phi)$ na zadanou množinu elementů (ve smyslu omezujících podmínek dotazu) $X \subseteq E$ vrátí množinu elementů $Y \subseteq E$, které jsou podmínkami na základě Φ implikovány. Pro úložiště respektující popis pomocí množiny funkčních závislostí platí:

$$X \subseteq Y \tag{17}$$

Vztah (17) platí díky tomu, že úložiště pokrývá i triviální funkční závislosti typu $A \rightarrow A \; \forall A \in \mathscr{A}^C$ (definice viz (5)). Důsledkem je fakt, že mechanismus generalizace je v takovém případě monotónní.

Institucionální repozitář AV ČR http://hdl.handle.net/11104/0135724

Odvozovací mechanismus je založen na pravidle

$$(A_i; a_{i'}) \in X \wedge (A_i; a_{i'}) \rightarrow (A_j; a_{j'}) \in \Phi \rightsquigarrow (A_j; a_{j'}) \in Y \tag{18}$$

Odvozování dalších (implikovaných) faktů pomocí pravidla (18) z X je realizováno přes vlastnost tranzitivity (6) funkčních závislostí.

Protože úložiště pokrývá pouze instance funkčních závislostí (14), je zaručena konzistence úložiště (12). Generalizační mechanismus zaručuje, že na konzistentní vstup (13) bude vrácen konzistentní výstup.

Toto tvrzení vychází z předpokladu, že množina X je konzistentní. Neboť dle (17) $X \subseteq Y$, pak část odpovídající X je rovněž konzistentní. Stačí tedy ukázat konzistenci množiny $Y - X$. Aby došlo k aktivaci nějakého dalšího y na základě pravidla (18) aplikovaném na nějaké konkrétní $x \in X$, díky konzistenci úložiště Φ odvozené y musí být rovněž konzistentní vůči všem elementům v Y.

4. Komplexní atributy a úložiště

Uvažujeme funkční závislost atributu A_r na komplexním atributu $\{A_{l_1} \dots A_{l_{n'}}\}$:

$$\{A_{l_1} \dots A_{l_{n'}}\} \rightarrow A_r \in M \tag{19}$$

Diskutujme nyní dvě varianty vyjádření vztahu mezi komplexními atributy (funkční závislost s komplexním atributem na levé straně).

- Uložení instance funkční závislosti s komplexním atributem.
- Uložení metainformace o existenci funkční závislosti s komplexním atributem.

4.1. Komplexní atributy v úložišti

Tato varianta předpokládá, že úložiště je schopno pojmout informaci o komplexním atributu a jeho instanci. Prakticky takové úložiště lze implementovat jako množinu uspořádaných trojic

$$\{< id, \{(A_{l_1}; a_{l_1}) \dots (A_{l_{n'}}; a_{l_{n'}})\}, (A_r; a_r) >\} \tag{20}$$

kde id je identifikátor instance, $\{(A_{l_i}; a_{l_i})\} \subseteq \mathscr{P}(E)$ je hodnota komplexního atributu (tj. hodnoty representující podmínku aktivace instance komplexního atributu) a $(A_r; a_r) \in E$ je fakt reprezentující důsledek aktivace.

Na takové úložiště implementované pomocí atomických atributů (tj. splňující požadavky 1NF) bude potřeba $n' + 1$ instancí dvojic mezi identifikátorem id a elementy $(A_i; a_{i'})\, \forall i \in \{l_1 \dots l_{n'}\} \cup \{r\}$.

Funkční závislost mezi jednoduchými atributy je (jako triviální důsledek) namapována na 2 dvojice.

4.2. Komplexní atribut jako metainformace

Způsob uvedený v předchozí sekci vede na poměrně komplikovaně organizované úložiště. Pokusme se tedy navrhnout jednodušší způsob. Modelování funkční závislosti s komplexním atributem (19) se rozpadá na dva případy. Pokud:

- existuje atribut A_k takový, že $A_k \rightarrow A_{l_i} \in M^\Phi$ pro $\forall i = 1 \dots n'$, pak taková závislost bude v asociativním úložišti Φ interpretována jako soubor instancí funkčních závislostí s atributem A_k na levé straně a atributy A_{l_i} a A_r na straně pravé. Tedy:

  $$\forall i = 1 \dots n' \, \exists A_k : A_k \rightarrow A_{l_i} \in M \rightsquigarrow A_k \rightarrow A_{l_i} \in M^\Phi \wedge A_k \rightarrow A_r \in M^\Phi \tag{21}$$

  Informace o existenci funkční závislosti s komplexním atributem v M je zachycena jako metainformace pomocí faktu

  $$\{A_{l_1} \dots A_{l_n}\} \rightarrow A_r \in M^\Phi_\odot \tag{22}$$

Institucionální repozitář AV ČR http://hdl.handle.net/11104/0135724

- takový atribut A_k neexistuje, asociativní úložiště nemá žádný prostředek, jak svázat hodnotu komplexního atributu $\{A_{l_i}\}$ na levé straně s hodnotou atributu A_r na pravé straně a tedy tento vztah nebude úložištěm pokryt.

Existence atributu A_k je tedy podmínkou nutnou, aby vůbec došlo k zachycení vztahu mezi atributy. V praxi tuto podmínku triviálně zaručíme, pokud každému záznamu ve vstupní kolekci přiřadíme jednoznačný identifikátor A_I. Z definice funkční závislosti vyplývá, že $A_I \rightarrow A$ pro $\forall A \in \mathscr{A}^C$, tedy druhá varianta nemůže nastat, neboť v takovém případě minimálně $A_k = A_I$.

4.3. Instance funkčních závislostí s komplexním atributem

Diskutujme nyní úpravy asociativního úložiště (20) ve smyslu (21) tak, aby bylo možné rozšířit generalizační mechanizmus (17 -18) o podporu komplexních atributů na základě metainformace v $M^{\Phi}_{\odot}$.

Předpokládejme, že podmínka existence klíče (21) je splněna. Ukažme, že za tohoto předpokladu není potřeba ukládat instance funkční závislosti s komplexním atributem (19), neboť generalizační krok lze provést na základě znalosti hodnoty/hodnot společného atributu A_k, která je odvozena "zpětně" z hodnoty komplexního atributu.

$$\{A_{l_1} \dots A_{l'_n}\} \rightarrow A_r \in M^{\Phi}_{\odot} \wedge \forall (A_{l_i}; a_{l_i}) \in X \rightsquigarrow (A_r; a_r) \in Y \tag{23}$$

Zaveďme aktivaci komplexního atributu. Úložiště obsahuje instance

$$(A_k; a_k) \rightarrow (A_{l_i}; a_{l_i}) \in \Phi \qquad i = 1 \dots n' \tag{24}$$

$$(A_k; a_k) \rightarrow (A_r; a_r) \in \Phi \tag{25}$$

Generalizační krok spočívá v nalezení množiny možných elementů $(A_k; a_k)$ relevatních k $(A_r; a_r)$ tak, že $a_k \in K(X, A_k)$, přičemž

$$K(X, A_k) = \{a_k : \left\{ \begin{array}{l} \{A_{l_1} \dots A_{l'_n}\} \rightarrow A_r \in M^{\Phi}_{\odot} \wedge A_k \rightarrow A_r \in M^{\Phi} \\ \forall i = 1 \dots n' : (A_{l_i}; a_{l_{i'}}) \in X \wedge (A_k; a_k) \rightarrow (A_{l_i}; a_{l_{i'}}) \in \Phi \end{array} \right. \} \tag{26}$$

Nyní se generalizační mechanismus rozpadá na tři případy. Pokud:

- $|K(X, A_k)| = 0$, pak není funkční závislost $\{A_{l_1} \dots A_{l'_n}\} \rightarrow A_r$ pokryta instancemi buďto v X nebo v úložišti Φ a tudíž nemůže být aktivována.
- $|K(X, A_k)| = 1$, neboli $\{A_{l_1} \dots A_{l'_n}\} \rightarrow A_k \in M$, pak hodnotu a_r ze vztahu (23) určíme pomocí instance (25). V tomto případě atributy A_r a A_k mohou splývat.
- $|K(X, A_k)| > 1$, neboli $\{A_{l_1} \dots A_{l'_n}\} \rightarrow A_k \notin M$, tedy předchozí mechanismus nelze použít. Za této situace ovšem $\{A_{l_1} \dots A_{l'_n}\} \rightarrow A_r \in M^{\Phi}_{\odot}$ a $A_k \rightarrow A_r \in M^{\Phi}$, pak ale musí nutně $(A_k; a_k) \rightarrow (A_r; a_r) \in \Phi \ \forall a_k \in K(X, A_k)$ vést na jedinou hodnotu a_r (neboť v opačném případě by $\{A_{l_1} \dots A_{l'_n}\} \rightarrow A_r \notin M$).

Tedy, instance funkčních závislosti s komplexním atributem není potřeba ukládat za předpokladu existence nějakého společného klíčového atributu A_k, neboť se podle výše zmíněného postupu dají odvodit z instancí funkčních závislostí mezi jednoduchými atributy.

Pokud bychom místo generalizace dotazu volili jeho specializaci (tj. chtěli bychom získat seznam objektů splňující podmínky dané dotazem), je uvažování komplexních atributů v odvozovacím mechanizmu bezpředmětné, neboť zaručeně dojde k aktivaci atributu A_k, který je vždy specializovanějším nežli libovolný z atributů A_{l_i}. V okamžiku, kdy budeme chtít pomocí generalizace atributů určujících objekty získat jejich popis (tj. konkrétní množiny dvojic $(A_i; a_{i'})$), dojde nejprve k aktivaci atributu A_k a v následném kroku pak k aktivacím atributů A_{l_i} i A_r, tj. v tomto případě není nutné uvažovat rozšíření (23) generalizačního mechanismu.

Naopak rozšíření (23) je nutné uvažovat v případě, že X obsahuje instance všech atributů A_{l_i}, avšak neobsahuje aktivaci A_k, což v praxi nastává poměrně často.

Institucionální repozitář AV ČR http://hdl.handle.net/11104/0135724

4.4. Úložiště s metainformacemi

Diskutujme nyní možnost, jak implementovat úložiště s vlastnostmi odpovídajícími předchozím dvěma odstavcům.

Neboť M^{Φ} obsahuje funkční závislosti pouze mezi jednoduchými atributy, není nutné uchovávat identifikátor instance, postačuje přímo uchovávat jen dvojice elementů $(A_i; a_{i'}) \rightarrow (A_j; a_{j'}) \in \Phi$ například pomocí množiny uspořádaných dvojic

$$\{< (A_i; a_{i'}), (A_j; a_{j'}) >\} \tag{27}$$

Funkční závislosti s komplexními atributy jsou uloženy v $M^{\Phi}_{\odot}$, způsob uložení je analogický k (20) s tím rozdílem, že se uchovává informace pouze o funkční závislosti, nikoliv o jejích instancích.

$$\{< id, \{A_{l_1} \dots A_{l_{n'}}\}, A_j >\} \tag{28}$$

4.5. Případová studie - porovnání

Porovnejme závěrem obě navržené implementace úložiště podle počtu záznamů nutných k uložení informace, první popisující instance potencionálně komplexních atributů a druhý, obsahující pouze instance funkčních závislostí s jednoduchými atributy. Pro jednoduchost uvažujme, že již na začátku ukládaní máme k dispozici model popisující vstupní kolekci dat.

Pro případovou studii byla zvolena data reprezentující n'-ární součet hodnot atributů $A_1 \dots A_{n'}$, hodnota součtu je pak uložena pod atributem A_S. Každý záznam je rozšířen o primární klíč A_k z důvodů uvedených u (21).

$$\begin{array}{ccccccc} A_k & A_1 & A_2 & \cdots & A_{n'-1} & A_{n'} & A_S \\ 0 & 0 & 0 & & 0 & 0 & 0 \\ 1 & 0 & 0 & & 0 & 1 & 1 \\ 2 & 0 & 0 & & 0 & 2 & 2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 9 & 0 & 0 & & 0 & 9 & 9 \\ 10 & 0 & 0 & & 1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 19 & 0 & 0 & & 1 & 9 & 10 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 10^{n'} & 9 & 9 & & 9 & 9 & 9n' \end{array} \tag{29}$$

Model implikovaný vstupní kolekcí dat je

$$A_k \rightarrow A_S \tag{30}$$

$$A_k \rightarrow A_i \ \forall i = 1 \dots n' \tag{31}$$

$$\{A_1, \dots, A_n\} \rightarrow A_S \tag{32}$$

Každý záznam ze vstupních dat (29) implikuje

- 1 instanci funkční závislosti (30),
- n' instancí funkčních závislostí (31) a
- 1 instanci funkční závislosti (32) s komplexním atributem.

Institucionální repozitář AV ČR http://hdl.handle.net/11104/0135724

Tedy počet instancí se během jednoho kroku změní o:

$$\Delta|I^B| = n' + 1 \tag{33}$$
$$\Delta|I^C| = 1 \tag{34}$$

Pokud budeme uvažovat uložení předmětného záznamu ze vstupní dat do úložiště pokrývající přímo instance komplexních atributů pomocí (20), je potřeba na uložení jedné instance funkční závislosti s komplexním atributem arity a na levé straně potřeba celkem $(a+1)$ záznamů. V tomto konkrétním případě (29) se interní struktury úložiště rozšíří o m záznamů, přičemž

$$m = \Delta|I^B| \cdot 2 + \Delta|I^C| \cdot (n' - 1 + 1) = 3n' + 2 \tag{35}$$

Budeme-li uvažovat úložiště pokrývající pouze instance funkčních závislostí s jednoduchými atributy získaných podle (21) uložených jako (28), každou takovou instanci pokryje 1 záznam v interní struktuře úložiště, tedy přírůstek m' pro tento případ je

$$m' = \Delta|I^B| \cdot 1 = n' + 1 \tag{36}$$

Neboť se jako metainformace ukládá seznam funkčních závislostí s komplexními atributy, zůstává velikost $|M^{\Phi}_{\odot}|$ nezměněna.

Srovnáním (35) a (36) získáme

$$\frac{m'}{m} = \frac{n' + 1}{3n' + 2} \simeq 33\% \tag{37}$$

5. Závěr

Článek se zabývá uložením vstupních dat do úložiště pomocí odhadnuté struktury. Ukazuje, že intuitivně nejjednodušší možný návrh úložiště Φ jako asociativní paměti $(A_i; a_{i'}) \rightarrow (A_j; a_{j'})$ primárně popisující vztahy mezi jednoduchými atributy je postačující, neboť obsahuje veškeré informace nutné pro podporu vztahů platných mezi komplexními atributy za předpokladu (21), že existuje nějaký atribut jednoznačně definující každý záznam ve vstupní kolekci dat. Tento fakt vede k nutnosti uložení pouze instancí funkčních závislostí s jednoduchými atributy, přičemž vztahy mezi komplexními atributy jsou vyjádřeny externě pomocí metainformace $M^{\odot}$.

Ukazuje se, že analýzu těchto vztahů je nutné použít pouze při generalizaci, generalizační mechanismus je nutné rozšířit o pravidlo (23).

V článku prezentovaný výsledek vede ke značným paměťovým úsporám, jednoduššímu návrhu úložiště a v neposlední řadě k odstranění případných duplicit a nekonzistenci dat v úložišti. Vzhledem k metodám odhadu dat se jedná o rozdělení výpočetní složitosti algoritmu (4) na dvě části, první polynomiálně složitou pro odhad struktury (7) a uložení dat (bez uvažování komplexních atributů) a druhou, nepolynomiálně složitou generující příslušná metadata v $M^{\Phi}_{\odot}$. Proto tuto část je ale možné přejmout výsledky z [3, 6, 7, 8, 9, 10, 11]

V závěru článku je uvedeno porovnání dvou popisovaných přístupů k úložišti dat, první je implementované s přímou podporou instancí funkčních závislostí s komplexními atributy a druhá bez této podpory. Obě implementace jsou navrženy tak, aby splňovali podmínku 1NF. Díky tomuto faktu je nutné ukládat jednu instanci funkční závislosti s komplexním atributem pomocí několika záznamů. Případová studie v jednom konkrétním případě (n-ární součet čísel) ukazuje, že úložiště bez pokrytí vztahů mezi komplexními atributy vykazuje asi 66% úsporu paměťových nároků úložiště. Jinými slovy případová studie potvrzuje výhodu pokrytí informace pomocí jednoduchých asociačních pravidel a v případě nutnosti uvažování metainformace o vztazích mezi komplexními atributy.

Závěrem poznamenejme, že z korespondujících důvodů nebyla do formátu RDF zavedena podpora komplexních atributů.

Institucionální repozitář AV ČR http://hdl.handle.net/11104/0135724

Praktické závěry tohoto článku jsou nasnadě, v případě potřeby uložení velkého počtu dat, které lze předpokládat i u extrakce dat z webových stránek, je hledisko paměťových úspor velmi důležité. V tomto konkrétním případě ani neznamená nutnost ztráty informace, ba právě naopak zabraňuje případné nekonzistenci dat vznikající duplicitou mezi instancemi funkčních závislostí s komplexními atributy a odvozovacím mechanismem pro tyto instance.

Literatura

[1] Eric Miller, Ralph Swick, Dan Brickley. "Resource Description Framework". `<http://www.w3.org/RDF/>` [on-line], 2004.

[2] Grigoris Antoniou, Frank van Harmelen. "A Semantic Web Primer". MIT Press, 2004. ISBN: 0-262-01210-3.

[3] M. Řimnáč. Odhad struktury dat a induktivní logické programování In *ITAT 2005: Workshop on Information Technologies, Applications and Theory.* pp. 263-272. ISBN: 80-7097-609-8. 2005

[4] D.W. Ebley, C. Tao, S.W. Liddle. "Automating the extraction of data from HTML tables with unknown structure." In *Data & Knowledge Engineering* 54, pp 3-24. Elsevier. 2005

[5] G. Gottlob, Ch. Koch. " A Formal Comparison of Visual Web Wrapper Generators." In *Proc. of. SOFSEM 2006: Theory and Practice of Computer Science*. LNCS 3831. ISSN 0302-9743. Springer.

[6] P.A. Flach, I. Savnik. "Database Dependency Discovery: A Machine Learnig Approach". In *AI Communications*, Volume 12/3, pp. 139–160. 1999.

[7] H. Mannila, K.J. Räihä. "Dependency Inference". In *Proc. of VLDB*. pp. 155–158. ISBN: 0-934613-46-X. 1987.

[8] H. Mannila, K.J. Räithä. "Algorithms for inferring functional dependencies from relations." In *Data & Knowledge Engineering* 12, pp 83-99. Elsevier. 1994.

[9] J. Kivinen, H. Mannila. "Approximate Inference of Functional Depentencies from Relations". In *Proc. of 4. int. conf. on Database Theory*, Berlin, Germany. pp. 129–149. ISSN: 0304-3975. 1995.

[10] Ch. Giannella, E. Robertson. "On Approximation Measures for Functional Dependencies". In *ADBIS 2002: Advances in databases and information systems*, Elsevier. pp. 483–507. ISSN 0306-4379. 2004.

[11] T. Akutsu, S. Miyano, S. Kuhara. "A Simple Greedy Algorithm for Finding Functional Relations: efficient implementation and average case analysis." Theoretical Computer Science, Volume 292, Issue 2. pp 481–495. ISSN: 0304-3975. 2003.

[12] D. Bednarek, D. Obdrzalek, J. Yaghob, F. Zavoral. "Access Rights Definition and Management in an Information System based on a DataPile Structure". In *ITAT 2004, Workshop on Information Technologies, Application and Theory*, 2004.

Institucionální repozitář AV ČR http://hdl.handle.net/11104/0135724