STUDIA INFORMATICA 2010
Volume 31 Number 2A (89)

Tomasz PIŁKA
Uniwersytet im. Adama Mickiewicza w Poznaniu, Wydział Matematyki i Informatyki

METODY PRZEPROWADZANIA NORMALIZACJI DANYCH XML

Streszczenie. Normalizacja jest procesem mającym na celu uzyskanie schematu danych odpornego na możliwość wystąpienia w instancji powtarzających się danych. Proces ten jest dobrze zbadany dla danych relacyjnych. Wraz z rosnącą popularnością stosowania XML jako formatu wymiany i przechowywania danych normalizacja nabiera ogromnego znaczenia dla danych XML. W procesach integracji danych nieuzasadniona nadmiarowość danych jest szczególnie niepożądana. W pracy analizujemy metody przeprowadzania normalizacji danych XML. Badanie tego problemu w ujęciu XML jest trudniejsze niż dla danych relacyjnych, gdyż w tym przypadku musimy dodatkowo uwzględnić hierarchiczną strukturę danych.

Słowa kluczowe: normalizacja, XML, redundancja danych, postać normalna XML

METHODS OF NORMALIZATION XML DATA

Summary. Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process: eliminating redundant data and ensuring data dependencies. At the work we discuss methods achieving the normalization of data XML. In the case of XML data, this problem is more difficult than for relational data – we must also take the hierarchical data structure into consideration.

Keywords: normalization, XML functional dependency, data redundancy, XML normal form

1. Wstęp

Organizacja i przechowywanie danych to jedno z fundamentalnych zagadnień stawianych twórcom systemów informatycznych. Od systemów takich oczekuje się elastyczności, łat-

wości dostępu, z jednoczesnym zagwarantowaniem poprawności danych. Stąd potrzeba definiowania poprawnych schematów. Problem projektowania i normalizacji baz danych jest jednym z klasycznych tematów omawianych w każdym podręczniku baz danych [1, 13].

Dynamiczny rozwój usług sieciowych pozwalających na współdzielenie i udostępnianie danych spowodował, że coraz większą rolę zaczął odgrywać nowy model przechowywania danych. Format XML, o którym mowa, uznano jako standard w procesach wymiany informacji, jak i integracji danych pochodzących z różnych źródeł. Rosnąca ilość danych udostępnianych w ten sposób sprawia, że problem projektowania schematów dobrej jakości nabiera podobnej wartości jak problem normalizacji danych relacyjnych. Zadanie to jest tym istotniejsze, że obecnie potencjalni klienci systemów wymiany czy integracji danych oczekują, iż przesyłane do nich dane pozbawione będą zbędnych powtórzeń (w tym duplikatów), danych podobnych, a dane zostaną przesłane w krótkim czasie przy niewielkim nakładzie pracy.

Definiowania postaci normalnych dla XML wymaga określenia zasad zachowania spójności danych XML. W literaturze można znaleźć określanie węzłów integralności za pomocą kluczy [6], zależności pomiędzy ścieżkami [6], różnych form zależności funkcyjnych [8, 9, 10]. Pracę nad badaniem zależności funkcyjnych dla danych relacyjnych sięgają lat 70. ubiegłego wieku [1, 13]. Sam problem definiowania postaci normalnych dla XML przedstawiono w wielu pracach. Jedno z pierwszych podejść zostało zaprezentowane przez Arenasa i Libkina w pracy [4], w której wprowadzają definicję zależności funkcyjnej dla XML w ujęcie krotek opartych na poddrzewach. Problem ten, dalej rozwijany był przez S. Kolahiegi [7] i we wspólnych pracach Kolahi i Libkin [8, 9]. M. Vincent [6, 10] zaproponował rozważanie zależności funkcyjnych XML, w którym strony implikacji stanowią ścieżki w dokumencie XML. Inne podejście do otrzymywania danych XML w postaci normalnych prezentuje w swoich pracach Martin Nečaský [11, 12] – w jego rozważaniach proces normalizacji rozpoczyna się od zbudowania modelu ER, który następnie transformowany jest do dokumentu XML. W pracy [12] prezentuje zestawienie metod realizujących proces normalizacji w podobnej technologii.

W pracy tej przedstawiamy problem normalizacji danych XML. Dokonujemy przeglądu możliwych postępowań oraz prezentujemy własne rozwiązania. W trakcie przeprowadzania normalizacji, poza usuwaniem redundancji ważnym elementem jest analizowanie anomalii występujących w trakcie wstawiania, modyfikacji i usuwania danych. Element ten nie jest jednak omawiany w tej pracy. Układ dalszej części pracy jest następujący: w kolejnym rozdziale prezentujemy przykład motywujący do prowadzenia badań. W trzecim rozdziale przybliżamy proces normalizacji relacyjnych baz danych. W kolejnych rozdziałach przedstawiamy normalizację schematów XML – dla istniejącego schematu oraz schematu wprowadzanego z modelu ER, proponujemy określenie warunku koniecznego na możliwość przeprowadzania normalizacji danych XML. Podsumowanie pracy zawarte w rozdziale szóstym.

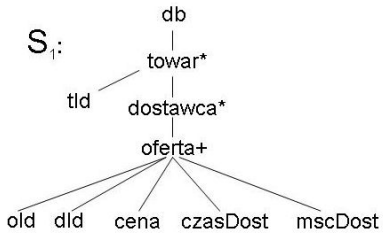
2. Motywacja

Powszechność stosowania formatu XML sprawia, że ilość danych, jaka jest w nim przetrzymywana, rośnie w zawrotnym tempie. Podobnie jak to miało miejsce dla danych relacyjnych, również tu pojawił się problem projektowania poprawnych struktur (schematów) umożliwiających przechowywanie danych. Oczekujemy, że przechowywane dane będą pozbawione redundancji oraz odporne na znane anomalie w trakcie operacji na nich.

W przypadku danych XML problem z redundancją danych może wystąpić w związku z niewłaściwym zaprojektowaniem schematu. Dodatkowym utrudnieniem jest hierarchiczna struktura danych, która nie występuje w przypadku relacyjnym. Warto jednak tu zaznaczyć, że właśnie dzięki hierarchiczności danych XML czasami jesteśmy w stanie zniwelować problemy, jakie występowały dla danych relacyjnych.

Rozpatrzmy zatem następujący przykład fragmentu bazy danych przechowującej informacje dotyczące zamówień i dostawców oferujących pewne towary zawarte w zamówieniach. Mianowicie, niech *towar* jest opisany przez pewien identyfikator (*tId*), miejsce dostawy (*mscDost*), czas dostawy (*czasDost*) i cenę (*cena*). Każdy towar dostarczany jest przez pewnego dostawcę, dodatkowo przyjmijmy założenie, że w dane miejsce towar może dostarczać tylko jeden dostawca.

Dla tak opisanej rzeczywistości możemy zbudować następujący schemat:



Rys.1 Przykładowy schematów XML

Fig. 1 Sample XML Schema

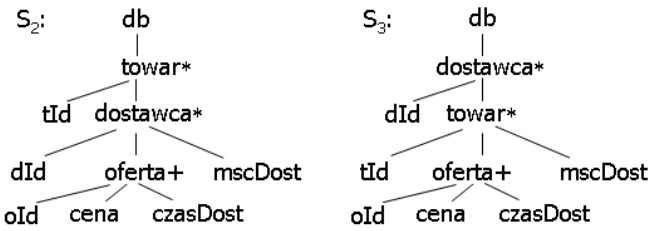
Umieszczony na rysunku 1 schemat pokazuje możliwą hierarchiczną reprezentację umiejscowienia głównych elementów opisywanej rzeczywistości: *towar – dostawca – oferta*. Warto podkreślić, że odwzorowując ten schemat na model relacyjny jedną z możliwości byłoby utworzenie poszczególnych encji: towar, dostawca, oferta.

W schemacie S_1 zachodzą następujące więzy integralności: wszystkie elementy *towar* będące potomkami tego samego elementu *dostawca* muszą mieć taką samą wartość elementu *tId*; element *mscDost* zależy funkcyjne od wartości identyfikatora danej partii (*pId*) oraz identyfikatora dostawcy (*dId*) – oznacza to, że wszystkie towary danego dostawcy dostarczane w ramach tej samej partii muszą mieć to samo miejsce dostarczenia. Wartość elementu *mscDost* determinuje wartość identyfikatora dostawcy (*dId*). Oznacza to, że jeżeli znamy

miejsce dostawy, to dokładnie wiemy, który dostawca dostarczał tam towar, dopuszczamy tu, że jeden dostawca może dostarczać towary do wielu miejsc.

Taka reprezentacja nie jest jednak dobra. Informacje o dostawcy są powielane w opisie każdego towaru oferowanego przez tego dostawcę. Podobnie z miejscem dostawy danego towaru (*mscDost*) – z warunków integralności przyjętych powyżej wiemy, że wszystkie towary dostawcy dostarczane w ramach jednej partii dostarczane są w to samo miejsce, stąd też informacja ta przechowywana przy towarach jest nadmiarowo powtarzana. Występuje tu, zatem redundancja danych.

Schemat z rysunku 1 nie jest jedynym możliwym przedstawieniem opisywanej rzeczywistości. Elementy *towar*, *dostawca*, *oferta* możemy ułożyć w innym uporządkowaniu hierarchicznym. Zostało to przedstawione na rysunku 2.



Rys. 2. Inna możliwa reprezentacja schematu z rys 1
Fig. 2. Another sample representation schema from figure 1

Ułożenie elementów zaprezentowane na schematach S_2 i S_3 również nie jest najlepsze – dla schematu S_2 dostawca może dostarczać wiele towarów, mamy zatem niepotrzebne powtarzanie informacji o dostawcy. W przypadku schematu S_3 – ten sam towar może być dostarczany przez wielu sprzedawców, stąd niepotrzebne powtarzanie informacji o towarze.

Pojawia się więc pytanie*: W jaki sposób dokonywać projektowania schematu XML, aby był on poprawny z punktu widzenia problemu redundancji danych?* Postępowanie umożliwiające wybór schematu, który będzie redundancji, duplikatów danych czy też odporny na anomalię zostanie opisane w dalszej części pracy.

3. Normalizacja danych relacyjnych

Celem przeprowadzania normalizacji schematów relacyjnych jest uzyskanie schematu o określonej formie, najczęściej normalizacja jest przeprowadzana do uzyskania trzeciej postaci normalnej (3PN) bądź postaci normalnej Boyce-Codda (PNBC). Proces ten w ogólności polega na dekompozycji schematu wejściowego.

Przeprowadzanie dekompozycji powinno odbywać się z zachowaniem wszystkich danych i jednoczesnym zachowaniem zależności funkcyjnych. Dla schematu relacyjnego $\boldsymbol{R} = (U, F)$,

gdzie U jest skończonym zbiorem atrybutów, a F jest skończonym zbiorem zależności funkcyjnych nad U oraz $U_1, U_2 \subseteq U, U_1 \cup U_2 = U$. Schematy $\boldsymbol{R}_1 = (U_1, F_1)$ i $\boldsymbol{R}_2 = (U_2, F_2)$ są bezstratną dekompozycją schematu $\boldsymbol{R} = (U, F)$ wtedy i tylko wtedy, gdy:

- dekompozycja zachowuje dane, jeżeli dla każdej instancji $\boldsymbol{R}$ schematu $\boldsymbol{R}$ w wyniku złączenia projekcji schematu $\boldsymbol{R}$ na *U1* i *U2* otrzymamy relację równą $\boldsymbol{R}$:

  $$R = \pi_{U_1}(R) \rhd\lhd \pi_{U_2}(R),$$

- dekompozycja zachowuje zależności funkcyjne, jeżeli:

  $$F^+ = (F_1 \cup F_2)^+,$$

  gdzie $F_1^+ = \{X \rightarrow Y \mid X \rightarrow Y \in F \wedge X \cup Y \subseteq U_1\}$, analogicznie dla F_2^+.

Przypomnijmy definicję zależności funkcyjnych w przypadku danych relacyjnych [13]:

Definicja 3.1. Niech dana będzie relacja $\boldsymbol{R}(U)$ i niech $X, Y \subseteq U$. Zależnością funkcyjną nazywamy wyrażenie postaci: $X \rightarrow Y$.

Mówimy wówczas, że Y zależy funkcyjnie od X lub, że X determinuje funkcyjnie Y.

Mówimy, że w $\boldsymbol{R}$ spełniona jest zależność funkcyjna $X \rightarrow Y$ (oznaczamy to jako $R \models X \rightarrow Y$), jeśli dla każdych dwóch krotek r_1 i $r_2 \in R$ zachodzi:

$$r_1[X] = r_2[X] \Rightarrow r_1[Y] = r_2[Y]$$

Inna reprezentacja danych XML i relacyjnych uniemożliwia bezpośrednie przeniesienie definicji zależności funkcyjnej dla danych relacyjnych na przypadek danych XML. Powyższa definicja określona jest dla płaskich, tabularycznych danych, w przypadku XML musimy uwzględnić również hierarchiczność danych.

Definicja 3.2. Schemat jest w trzeciej postaci normalnej (3NP), jeżeli dla każdej zależności funkcyjnej $X \rightarrow A \in F^+$ zachodzą następujące warunki:

X jest nadkluczem (zawiera klucz) albo A jest atrybutem kluczowym.

Z drugiego warunku wynika, że tylko atrybuty kluczowe mogą funkcyjnie determinować zbiór atrybutów, który nie jest kluczem. Schemat jest w postaci normalnej Boyce-Codda, jeżeli spełniony jest pierwszy z przedstawiających warunków.

Normalizacja polega na sukcesywnej dekompozycji schematu relacji na jej projekcje. Proces ten dokonywany jest w wyniku analizowania zależności funkcyjnych. W efekcie otrzymujemy dobrze zaprojektowany schemat pozbawiony zbędnych redundancji i odporny na anomalie wstawiania, usuwania i modyfikacji danych.

W praktycznych pracach normalizację dla danych relacyjnych przeprowadza się do otrzymania trzeciej postaci normalnej. Nie gwarantuje to całkowitego usunięcia redundancji, za-

pewnia za to zachowanie wszystkich zależności funkcyjnych. Przeprowadzenie normalizacji do PNBC gwarantuje całkowite wyeliminowanie redundancji, jednak niektóre zależności mogą zostać utracone [14].

4. Normalizacja istniejącego schematu XML

W przypadku relacyjnych danych algorytmy przeprowadzania procesu normalizacji są dobrze zbadane, nie można ich jednak przenieść na przypadek danych XML. W dotychczasowych pracach zaproponowano kilka podejść do definiowania postaci normalnych dla XML [2, 4, 9]. Propozycja algorytmu przeprowadzającego normalizację umieszczona jest w pracy [2]. Algorytm ten zostanie przybliżony w dalszej części pracy.

W pracy wprowadzimy definicję postaci normalnej dla XML, wykorzystując do definiowania zależności funkcyjnych formuły drzewiaste (*tree-pattern formula, TPF*) [14, 15, 16]. Wcześniej wprowadzimy niezbędne definicje danych XML, zależności funkcyjnych. Zakładamy także, że żaden element w schemacie nie ma definicji rekurencyjnej. Wówczas każdy schemat może być przedstawiony jako drzewo. Na rysunku 1 podano reprezentację graficzną danych XML w postaci schematu S_1. Schematy danych XML najczęściej definiowane są z wykorzystaniem języków DTD bądź XSD [15]; w tym paragrafie do definiowania schematów wykorzystamy formuły drzewiaste.

4.1. Drzewa XML

Dane XML możemy zdefiniować jako nieuporządkowane, nieetykietowane drzewo (drzewo XML) nad skończonym zbiorem etykiet *L*, zbiorem wartości tekstowych $Str \cup \{\perp\}$, gdzie symbol $\perp$ oznacza wartość *null*.

Definicja 4.1 [Drzewo XML]: Drzewem XML I nazywamy układ:

$$I = (r, N^e, N^t, child, \lambda, \nu),$$

gdzie:

- *r* jest *wierzchołkiem wyróżnionym – korzeniem drzewa*, N^e jest skończonym zbiorem wierzchołków elementowych, a N t jest skończonym zbiorem wierzchołków tekstowych;
- $child \subseteq (\{r\} \cup N^e) \times (N^e \cup N^t)$ – to relacja wprowadzająca strukturę drzewiastą do zbioru $\{r\} \cup N^e \cup N^t$ *t*, gdzie *r* jest korzeniem drzewa, a każdy wierzchołek elementowy ma przynajmniej jednego potomka (który również jest albo wierzchołkiem elementowym, albo wierzchołkiem tekstowym), wierzchołki tekstowe są liśćmi;

- $\lambda : N^e \to L$ – to funkcja etykietująca wierzchołki elementowe ich nazwami (etykietami);
- $\nu : N^t \to Str \cup \{\perp\}$ – to funkcja etykietująca wierzchołki tekstowe wartościami ze zbioru *Str* bądź wartością $\perp$.

Dane XML przedstawiające konkretną reprezentację schematu wygodnie jest rozpatrywać jako drzewo I o schemacie S nad zbiorem zmiennych **x** jako parę (S, Ω), gdzie S jest schematem TPF, a Ω jest zbiorem wartościowań zmiennych w **x**. Wartościowanie $\omega \in \Omega$ jest funkcją przydzielającą zmiennym z **x** wartości z $Str \cup \{\perp\}$, tzn. ω: $\mathbf{x} \to Str \cup \{\perp\}$. Parę (S, Ω) nazywamy opisem danych XML.

Drzewo XML I spełnia opis (S, Ω), co oznaczamy jako $I \models (S, \Omega)$, jeśli I spełnia (S, ω) dla każdego wartościowania $\omega \in \Omega$. Zauważmy, że opis (S, Ω) reprezentuje pewną klasę instancji schematu S dla tego samego zbioru wartości Ω.

4.2. Zależności funkcyjne dla danych XML

Definicje zależności funkcyjnych dla XML korzystają z klasycznej definicji zależności funkcyjnych dla danych relacyjnych. Uwzględnienie struktury danych wymusiło wprowadzenie pewnych modyfikacji. W literaturze wyróżniamy dwa główne nurty badań prowadzonych nad określaniem zależności funkcyjnych dla XML. W pierwszym [2÷5] zależności funkcyjne oparte są na notacji krotek drzewiastych budowanych z wykorzystaniem ścieżek na podstawie definicji DTD (*tree tuple*), w drugim definicja korzysta ze ścieżek bazowanych na hierarchii elementów (*closest node*) [10]. Definicja wprowadzona w tym paragrafie wykorzystuje TPF:

Definicja 4.2 [Zależność funkcyjna XML]. Zależnością funkcyjną XML (*XFD*) nad zbiorem etykiet L i zbiorem zmiennych **x** nazywamy następujące wyrażenie:

$$fd ::= /P[C]/.../P\,[C],$$
$$P ::= l \mid P/l,$$
$$C ::= TRUE \mid P = x \mid C \wedge ... \wedge C,$$

gdzie $l \in L$, a x są zmiennymi w **x**. Jeśli nazwy zmiennych w **x** są jednoznaczne, to możemy zależność funkcyjną zapisać jako $fd(\mathbf{x})$.

Przykładowa zależność funkcyjna dla schematu S_1 ma więc postać:

$$f(x_1,x_2) = /zamowienie/partia[pId = x_1]/dostawca[towar/dId = x_2]$$

Zależności funkcyjne zgodne z powyższą definicją są wyrażeniami XPath [17], które dla danego wartościowania zmiennych ω zwracają sekwencję obiektów. Obiektami zwracanymi w wyniku obliczania zależności funkcyjnych mogą być zarówno całe węzły danych, jak i wartości tekstowe. Umożliwia nam to określenie spełniania zależności funkcyjnych. Dla

danej instancji danych XML $I = (S, \Omega)$ spełnia XML-ową zależność funkcyjną $f(x_1,\ldots x_k)$, jeżeli dla każdego wartościowania $\omega \in \Omega$ zmiennych funkcja $f(x_1,\ldots x_k)$ zwraca singleton.

Można zatem wprowadzić następującą definicję spełniania zależności funkcyjnych XML, dla danych których schemat jest TPF.

Definicja 4.3 [Spełnianie zależności funkcyjnych]. Niech I jest instancją schematu TPF $S(\mathbf{x})$, a f zależnością funkcyjną XML zdefiniowaną nad S. Instancja danych XML I spełnia zależność funkcyjną f (oznaczamy to jako: $I \models f$), jeżeli dla każdego wartościowania ω zmiennych w **x** zachodzi następująca implikacja:

$$I \models (S, \omega) \Rightarrow count([\![f(\omega)]\!]) \leq 1,$$

gdzie $[\![f(\omega)]\!]$ jest wynikiem f obliczonym przy wartościowaniu ω.

4.3. Postać normalna dla XML

Przyjmijmy, że przez (S, F) będziemy oznaczać schemat TPF S wraz ze zbiorem zależności funkcyjnych zdefiniowanych nad S. Domknięcie (S, F) oznaczmy przez $(S, F)^+$. Jest nim schemat TPF S wraz ze zbiorem zależności funkcyjnych, które mogą być wyprowadzone z (S, F).

Definicja 4.4 [Postać normalna XML]. Niech S będzie schematem TPF, a F zbiorem zależności funkcyjnych nad S. (S, F) jest w postaci normalnej XML (*XNF*) wtedy i tylko wtedy, gdy dla każdej zależności funkcyjnej $f/l \in (S, F)^+$ zachodzi również zależność $f \in (S, F)^+$, czyli

$$(S, F) \text{ jest w XNF} \Leftrightarrow (f/l \in (S, F)^+ \Rightarrow f \in (S, F)^+)$$

Zależność ta rozumiana jest w następujący sposób: przyjmijmy, że $f(x_1,\ldots x_k)/l$ jest zależnością funkcyjną, a I jest instancją danych schematu S. W instancji I spełniona jest zależność funkcyjna $f(x_1,\ldots, x_k)/l$, jeżeli dla każdego wartościowania ω układu zmiennych $(x_1,\ldots, x_k)$ znajduje się co najwyżej jedna wartość typu $type(f/l)$. Zatem, dla każdego wartościowania $(x_1,\ldots, x_k)$ powinniśmy mieć tylko jedną wartość f/l, czyli może być tylko jedno poddrzewo typu $type(f)$ oznaczone przez wyrażenie $f(x_1,\ldots, x_k)$. Zauważmy, że spełnienie warunku XNF pozwala na zniwelowanie problemu redundancji danych [14, 16].

4.4. Algorytm normalizacji – Decomposition Algorithm

W tym paragrafie przybliżamy założenia normalizacji opisane w pracy M. Arenasa [2], gdzie umieszczony jest kompletny algorytm. Algorytm ten na wejście przyjmuje schemat XML w postaci definicji DTD oraz zbiór zależności funkcyjnych $\sum$. Ze zbioru zależności funkcyjnych $\sum$ wybierane są zależności wprowadzające anomalie – zależności te nie spełniają implikacji podanej w definicji postaci normalnej dla XML. Algorytm bazuje na dwóch krokach wykonywanych dopóki schemat nie zostanie sprowadzony do postaci normalnej dla XML, tzn. dopóki wszystkie

zależności wprowadzające anomalie nie zostaną usunięte. Każdy krok algorytmu wykonywany jest na jednej zależności, następnie cała definicja schematu jest przebudowywana, aby uwzględnić wprowadzoną zmianę:

1. Przeniesienie atrybutu (*Moving Attributes*).
2. Tworzenie nowego elementu (*Creating New Element Types*).

Należy tu jednak podkreślić, że wynik działania algorytmu jest ściśle uzależniony od wejściowego schematu XML, dlatego w kolejnym paragrafie proponujemy wstępne przygotowanie takiego schematu. Drugim istotnym elementem jest fakt, że algorytm ten nie uwzględnia możliwości zamiany kolejności występowania elementów w hierarchii schematu XML, rozszerzenie to pozwala na przeprowadzenie w niektórych przypadkach pełnej normalizacji.

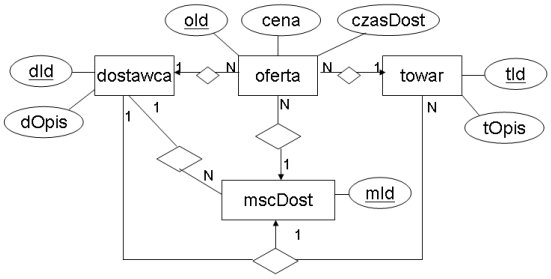
5. Normalizacja XML – od modelu ER do postaci normalnej

Jedną z metod przeprowadzania normalizacji dla danych XML jest zbudowanie modelu ER (*Entity-Relationship*), który będzie odzwierciedlał opisywany fragment rzeczywistości. W literaturze dostępne są rozważania takiego podejścia [12], gdzie relacyjny model ER rozszerzany jest do modelu ER dla XML. Następnie na modelu tym stosowane są algorytmy transformacji do schematu XML.

Nasza metoda bazuje na analizie zależności funkcyjnych występujących pomiędzy atrybutami i encjami w schemacie ER. Na rysunku 3 przedstawiono schemat relacyjny ER odpowiadający schematowi XML, umieszczonemu na rysunku 1.

Ze schematu umieszczonego na rysunku 3 wynikają następujące zależności funkcyjne:

```
oId → dId, tId,mId
dId, tId → mId
mId → dId
dId → dOpis
tId → tOpis
oId → cena czasDost
```



Rys. 3. Schemat ER odpowiadający schematowi S_1
Fig. 3. ER schema corresponding to schema S_1

Dla uproszczenia rozumowania przyjmijmy, że miejsce dostawy (*mId*) oznaczymy nazwą zapisanej powyżej encji *mscDost*. Procedura przeprowadzania transformacji tego schematu, do schematu XML jest dwuetapowa:

1. Na podstawie modelu ER tworzymy wejściowy schemat XML, do schematu jako elementy przenosimy nazwy encji, atrybutów i zależności funkcyjnych występujących pomiędzy atrybutami kluczowymi. Utworzony tak schemat **musi** spełniać warunek konieczny, umożliwiający sprowadzenie schematu do postaci normalnej. Warunek ten podajemy poniżej.
2. Schemat otrzymany w pierwszym etapie poddawany jest działaniu kroku: „*tworzenie nowego elementu*" zdefiniowanego w algorytmie Decomposition Algorithm (DA) z pracy M. Arenasa [1]. W ten sposób usuwamy zależności funkcyjne psujące dany schemat.

Warunek konieczny: Niech $f := \varphi(A_1,\ldots, A_k)/B$ jest zależnością funkcyjną typu $q/l/B$ nad schematem XML *S*. Mówimy, że schemat *S* spełnia warunek sprowadzenia do postaci normalnej, jeżeli zbiór składający się z etykiet wszystkich terminalnych potomków *q* jest funkcyjnie zależny od zbioru wszystkich etykiet terminalnych elementów występujących w *f*.

Procedura przeprowadzania normalizacji omawianego schematu jest zatem następująca:

Krok 1

Korzystając ze zbioru zależności funkcyjnych i kluczy dla schematu podanego na rys. 3 (zbiór ten umieszczony jest powyżej) i stosując warunek konieczny przeprowadzania normalizacji, podany powyżej, otrzymujemy następujący schemat DTD:

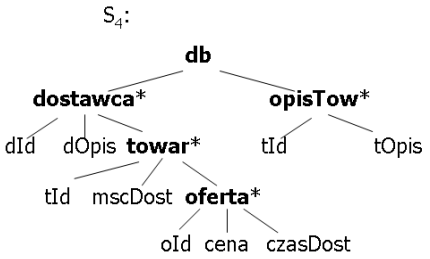
```
db → dostawca*
dostawca → dId dOpis towar*
towar → tId tOpis mscDost oferta*
oferta → oId cena  czasDost
```

Krok 2

DTD otrzymane w poprzednim kroku jest wejściowym schematem dla algorytmu DA. Szukamy anomalnej zależności funkcyjnej. W omawianym przykładzie zależnością taką jest: *tId* → *tOpis*. Stosujemy uproszczony algorytm DA. W efekcie otrzymujemy następującą definicję DTD:

```
db → dostawca* opisTow*
dostawca → dId dOpis towar*
towar → tId mscDost oferta*
oferta → oId cena  czasDost
opisTow → tId tOpis
```

W omawianym przykładzie była to jedyna zależność wprowadzająca anomalię, otrzymujemy zatem końcowy schemat przedstawiony na rysunku 4. Schemat ten, co można sprawdzić, jest w postaci normalnej dla XML.



Rys. 4. Wynik transformacji schematu ER do schematu będącego w postaci normalnej dla XML
Fig. 4. The result of transformation ER schema to XML schema in XNF

6. Podsumowanie

W pracy przedstawiono przegląd propozycji przeprowadzania normalizacji danych XML. Możemy wyróżnić dwa główne podejścia otrzymywania znormalizowanego schematu XML: budowanie schematu XML będącego już w postaci normalnej na podstawie modelu ER i przeprowadzanie normalizacji istniejącego schematu. Metody te są na tyle różne, że nie można ich porównywać. Wybór odpowiedniego podejścia uwarunkowany jest wejściowym zbiorem danych. W zaprezentowanych rozważaniach zastosowaliśmy notację TPF, która korzysta z semantyki języka XPath [17]. Wprowadzona definicja XNF oparta jest na założeniach przedstawionych w pracy [2].

Rosnąca popularność stosowania XML wymusza konieczność opracowania efektywnych metod pozwalających na zapewnienie poprawności danych. Problem ten jest szczególnie istotny w systemach integracji danych pochodzących z różnych źródeł. Dalsze badania dotyczyć będą konstruowania algorytmów pozwalających na przeprowadzanie normalizacji danych XML, z uwzględnieniem możliwości zmian w hierarchii elementów.

BIBLIOGRAFIA

1. Abiteboul S., Hull R., Vianu V.: Foundations of Databases. Reading, Massachusetts, Addison-Wesley, 1995.
2. Arenas M.: Normalization theory for XML. SIGMOD Record, Vol. 35, No. 4, 2006, s. 57÷64.
3. Arenas M., Libkin L.: XML Data Exchange: Consistency and Query Answering. PODS Conference, 2005, s.13÷24.
4. Arenas M., Libkin L.: A normal form for XML documents. ACM Trans. Database Syst., Vol. 29, 2004, s. 195÷232.

5. Arenas M., Libkin L.: An information-theoretic approach to normal forms for relational and XML data. J.ACM, Vol. 52, No. 2, 2005, s. 246÷283.
6. Jixue L., Millist V., Chengfei L.: Functional Dependencies, from Relational to XML. Lecture Notes in Computer Science, Vol. 2890, 2003, s. 1063÷1079.
7. Kolahi S.: Dependency-Preserving Normalization of Relational and XML Data. DBPL, LNCS, G. M. Bierman and C. Koch, Eds., Springer, Vol. 3374, 2005, s. 247 ÷261.
8. Kolahi S.: Dependency-preserving normalization of relational and XML data. Journal of Computer and Systems Sciences, Vol. 73, No. 4, 2007, s. 636÷647.
9. Libkin L.: Normalization theory for XML. LNCS, Vol. 4704, 2007, s.1÷13.
10. Millist V., Jixue L.: Checking Functional Dependency Satisfaction in XML. LNCS, 2003ol. 3671, 2005, s. 4÷17.
11. Nečaský M.: Conceptual model based normalization of xml view. Dateso, 28 Annual International Workshop of Databases, TExts, Specifications and Objects, CUR-WS, Vol. 330, 2008, s. 13÷24.
12. Nečaský M.: Conceptual model for XML. Phd thesis, Carles Univeristy, 2008.
13. Pankowski T.: Podstawy baz danych. Wydawnictwo Naukowe PWN, Warszawa, 1992.
14. Pankowski T., Piłka T.: Transformation of XML Data into XML Normal Form. Informatica, Vol. 33, No. 4, 2009, s. 417÷430.
15. Pankowski T., Cybulka J., Meissner A.: XML Schema Mappings in the Presence of Key Constraints and Value Dependencies. ICDT 2007 Workshop EROW, CEUR Workshop Proceedings. CEUR-WS.org, Vol. 229, 2007, s. 1÷15.
16. Piłka T., Pankowski T.: Zależności funkcyjne w danych XML. Studia Informatica, red.: S. Kozielski, B. Małysiak-Mrozek, P. Kasprowski, D. Mrozek, Vol. 30-2A (83), 2009, s. 7÷19.
17. XML Path Language (XPath) 2.0. 2006, www.w3.org/TR/xpath20 .

Recenzent: Prof. dr hab. inż. Stanisław Kozielski

Wpłynęło do Redakcji 31 stycznia 2010 r.

Abstract

Normalization as a way of producing good database designs is a well understood topic for relational data. For relational database design, normalization is a systematic way of ensuring that a database structure is suitable for general-purpose querying and free of certain undesir-

able characteristics – insertion , update, and deletion anomalies. In this case, preservation of all dependencies requires 3NF but then some redundancy is present. Further normalization to BCNF eliminates redundancies but does not preserve dependencies. This problem was also investigated for XML data, where is more difficult because of hierarchical data structure. On the other side, thanks to hierarchical nature of XML, we often can achieve both properties.

In the paper we discuss methods of designing XML normal form (XNF) for XML data. We define the notion of satisfaction of XML functional dependence by an XML tree. To define XNF we use the approach proposed by M. Arenas, whose some proposition are used in normalization algorithm based on conceptual model. We discuss various issues connected with normalization. We discuss, how the concept of database normalization can be used in the case of XML data. This semantic information is then used to check whether the design has some desirable properties, and if this is not the case, it is also used to convert the poor design into an equivalent well-designed database.

Adres

Tomasz PIŁKA: Uniwersytet im. Adama Mickiewicza, Wydział Matematyki i Informatyki, ul. Umultowska 87, 61-614 Poznań, Polska, tomasz.pilka@amu.edu.pl .