Mapowanie schematów i reformułowanie zapytań w systemie semantycznej integracji danych

Tadeusz Pankowski[1,2], **Jakub Marciniak**[2], **Tomasz Piłka**[2]

Streszczenie: W pracy omawiamy problemy semantycznej integracji danych w środowisku P2P. W tym podejściu każdy z *partnerów* (ang. *peer*) dysponuje własnym schematem danych oraz mapowaniami (odwzorowaniami) między swoim schematem a schematami wybranych partnerów. Celem integracji jest udzielenie odpowiedzi na zapytanie kierowane do dowolnego partnera. Odpowiedź winna zawierać dane związane zarówno z odpytywanym partnerem, jak i z wszystkimi partnerami powiązanymi z nim bezpośrednio lub pośrednio. Początkowe zapytanie musi być odpowiednio reformułowane (przeformułowywane) i przesyłane między partnerami, a uzyskiwane odpowiedzi cząstkową muszą być scalane i przesyłane w odwrotnym kierunku aż do partnera początkowego. Kluczowe role w tym procesie odgrywają: mapowanie schematów, reformułowanie zapytań i scalanie danych. Opisana metoda zaimplementowana została w systemie SIX-P2P (*Semantic Integration of XML data in P2P environment*).

Słowa kluczowe: semantyczna integracja danych, XML, P2P, mapowanie schematów, reformułowanie zapytań.

1. Wprowadzenie

Semantyczna integracja danych (SID) rozumiana jest jako dostarczanie danych pamiętanych w różnorodnych źródłach dla potrzeb realizacji wspólnego zadania. Technologia SID rozpatrywana jest przy tym jako naturalne rozwinięcie technologii baz danych włączające dorobek programowania rozproszonego, zarządzania wiedzą, usług sieciowych i semantycznej sieci Web (Haas, 2007). Semantyczna integracja może być zrealizowana poprzez: *integrację materializującą*, *integrację wirtualną* (*federację*) lub *indeksowanie*.

Materializacja polega na utworzeniu bazy zintegrowanych danych (hurtowni danych, magazynu danych). Odmianami materializacji są replikacja i keszowanie (ang. *caching*): *replikacja* to tworzenie kopii danych oraz ich synchronizacji z danymi źródłowymi według określonych strategii; *keszowanie* polega na gromadzeniu wyników wykonywanych zapytań w celu ich późniejszego wykorzystania do udzielania odpowiedzi na te same lub inne zapytania.

W *integracji wirtualnej* tworzona jest wirtualna reprezentacja integrowanych danych. Dane są prezentowane poprzez *schemat docelowy*, którego powiązanie ze *schematami źródłowymi* zdefiniowane jest za pomocą *mapowań*. W zależności od

[1] Politechnika Poznańska, Instytut Automatyki i Inżynierii Informatycznej, pl. M. Skłodowskiej-Curie 5, 60-965 Poznań, e-mail: tadeusz.pankowski@put.poznan.pl

[2] Uniwersytet im. Adama Mickiewicza, Wydział Matematyki i Informatyki, ul. Umultowska 87, 61-614 Poznań

sposobu wykonywania zapytań formułowanych względem schematu docelowego mówimy o *wymianie danych* (ang. *data exchange*) i *reformułowaniu zapytań* (ang. *query reformulation*). W przypadku wymiany danych odpowiednie zbiory danych źródłowych materializowane są według schematu docelowego i na tak utworzonej bazie danych wykonywane jest zapytanie docelowe. Przy reformułowaniu zapytań zapytanie docelowe jest przekształcane w zbiór zapytań, które mogą być wykonane bezpośrednio na danych źródłowych. Uzyskane w ten sposób odpowiedzi częściowe są następnie odpowiednio *scalane* (ang. *merged*). W przypadku integracji w środowisku P2P nie ma jednego wyróżnionego schematu docelowego, a rolę takiego schematu może pełnić schemat związany z dowolnym partnerem (ang. *peer*). Wówczas zarówno wymiana danych, jak i reformułowanie zapytań propagowane są między partnerami zgodnie ze ścieżkami semantycznych powiązań wynikających z mapowań między schematami.

Indeksowanie jest techniką integracji polegającą na utrzymywaniu informacji umożliwiającej dotarcie do pełnych danych. Tworzone są indeksy słów kluczowych zawierające adresy URL dokumentów, w których te słowa są zawarte. Dokumenty te pobierane są dynamicznie dopiero wtedy, gdy wykonywane jest zapytanie.

Praktycznym celem integracji danych jest umożliwienie sprawnego tworzenia nowych aplikacji wymagających danych z wielu źródeł (Haas, 2007). Realizacja tego celu wymaga rozwiązania wielu problemów, takich jak (Haas, 2007), (Haas i in., 2005), (Halevy i in., 2006): (a) określenie najlepszego źródła informacji dla danych potrzeb, (b) utworzenie odpowiedniego interfejsu do integrowanych danych lub ich schematów, (c) formułowanie zapytań do różnorodnych źródeł danych i opracowywanie planów ich efektywnego wykonania, (d) czyszczenie i rozwiązywanie konfliktów w celu uzyskania spójnego zbioru danych, (e) postępowanie z danymi niepewnymi i śledzenie ich pochodzenia, (f) identyfikowanie tych samych obiektów w różnych źródłach, (g) uwzględnienie wymagań bezpieczeństwa, poufności i wiarygodności danych.

W tej pracy rozważamy problemy mapowania schematów i reformułowania zapytań w systemach integracji danych w środowisku P2P. Układ pracy jest następujący: W rozdziale 2 przytaczamy propozycję definiowania mapowań dla integracji relacyjnych baz danych. W rozdziale 3 pokazujemy, w jaki sposób można defi-niować mapowania w przypadku schematów XML. Rozdział 4 definiuje klasę zapytań i proponuje metodę ich reformułowania. Dyskusja różnych strategii propagowania zapytań w systemie P2P i wynikających stąd konsekwencji przedstawiona jest w rozdziale 5. Uwagi dotyczące implementacji omawianych metod w systemie SIX-P2P scharakteryzowane są w rozdziale 6. Rozdział 7 podsumowuje pracę.

2. Mapowanie schematów

Kluczowym problemem w procesie integracji danych, niezależnie od przyjętej strategii integracji, jest zdefiniowanie *mapowania schematów* (ang. *schema mapping*). Mapowanie schematów jest specyfikacją określającą jak wystąpienia jednego schematu (*schematu źródłowego*) są transformowane w wystąpienia innego schematu (*schematu docelowego*). Pierwszym problemem wymagającym rozwiązanie jest wy-

bór języka służącego do zdefiniowania mapowania - *języka mapowania*.

Mapowania schematów rozważane były zarówno w odniesieniu do danych relacyjnych, jak i do danych XML. W odniesieniu do relacyjnych baz danych, problem odwzorowywania schematów sformułowany został przez Fagina i in. (Fagin i in., 2005), (Fagin i in., 2004). Dla schematu relacyjnego $R = (R_1, ..., R_k)$, gdzie każdy symbol relacyjny R_i ma przypisaną dodatnią liczbę całkowitą a_i zwaną *arnością* symbolu, instancją I jest funkcja (interpretacja) przypisująca każdemu symbolowi relacyjnemu R_i a_i-członową relację R_i^I. Pojęcia te można rozszerzyć na dane XML, gdzie schemat dany jest za pomocą DTD lub XML Schema, a instancją schematu jest dokument XML reprezentowany zgodnie z modelem DOM (Arenas, Libkin, 2005), (Pankowski i in., 2007), (XML Schema, 2004) i spełniający schemat.

Mapowania schematów wyrażane są za pomocą formuł logicznych zwanych formułami *STD* (ang. *source-to-target dependencies*) (Abiteboul i in., 1995). Formuły *STD* były wykorzystywane dla potrzeb badania teorii zależności (zależności funkcyjnych i zależności zawierania) w relacyjnych bazach danych (Abiteboul i in., 1995). Ogólna postać formuły STD w logice pierwszego rzędu, *FO STD*, (ang. *first order STD*) jest następująca:

$$\forall \mathbf{x}(\phi(\mathbf{x}) \Rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y})) \tag{1}$$

gdzie: $\mathbf{x}$ jest wektorem zmiennych źródłowych; $\phi(\mathbf{x})$ jest koniunkcją formuł definiują-cych wiązania zmiennych w strukturach źródłowych (w podejściu relacyjnym są to formuły relacyjne o postaci $R(x_1, ..., x_m)$, a w przypadku XML są to *formuły drzewiaste* (ang. *tree-pattern formulas*) omawiane w następnym podrozdziale) i równości o postaci $x = x'$; a $\psi(\mathbf{x}, \mathbf{y})$ jest koniunkcją formuł definiujących wiązania zmiennych w strukturach docelowych.

W wyniku skolemizacji, formuły STD można przekształcić w formuły drugiego rzędu:

$$\exists \mathbf{f} \forall \mathbf{x}, \mathbf{y}(\phi(\mathbf{x}) \wedge \kappa(\mathbf{x}, \mathbf{y}) \Rightarrow \psi(\mathbf{x}, \mathbf{y})) \tag{2}$$

gdzie $\mathbf{f}$ jest wektorem symboli funkcyjnych, a $\kappa(\mathbf{x}, \mathbf{y})$ jest koniunkcją równości o postaci $x = f(x_1, ..., x_k)$ lub $y = f(x_1, ..., x_k)$, gdzie $x, x_1, ..., x_k \in \mathbf{x}$, a $y \in \mathbf{y}$.

3. Mapowanie schematów XML

W przypadku specyfikowania odwzorowań dla danych XML formuły definiujące wiązanie zmiennych przyjmują postać *formuł drzewiastych*.

Definicja 1. *Niech L oznacza zbiór etykiet, niech $top \in L$ będzie etykietę szczytową (najbardziej zewnętrzną), a $x, x_1, x_2, ...$ zmiennymi tekstowymi. Wówczas wyrażenie π o składni:*

$$\pi ::= /top[E]; \qquad E ::= l = x \mid l[E] \mid E \wedge E,$$

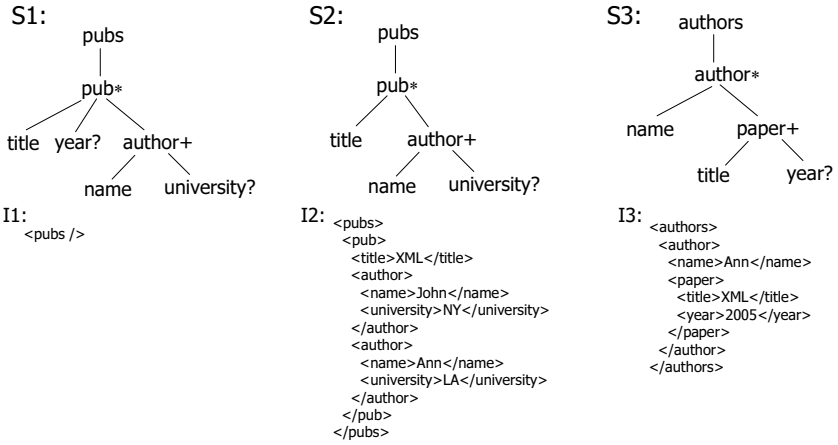
nazywamy (prostą) formułą drzewiastą (nad L), a wyrażenie δ

$$\delta ::= /top[E]; \qquad E ::= l = x \mid l(x_1, ..., x_n)[E] \mid E \wedge E.$$

nazywamy formułą drzewiastą z kluczem (nad L).

Wyrażenie $l(x_1, ..., x_n)$ określa klucz poddrzewa typu l. Wówczas każdej różnej krotce wartości $(x_1, ..., x_n)$ jedno-jednoznacznie odpowiada poddrzewo o korzeniu w wierzchołku typu l. Jeśli klucz nie jest jawnie podany, to domyślnie przyjmujemy, że każde poddrzewo zależy od wszystkich zmiennych występujących w formule.

W dalszym ciągu, dla uproszczenia rozważań, przyjmujemy, że atrybuty w danych XML reprezentowane są za pomocą elementów. Zakładamy także, że żaden element w schemacie nie ma definicji rekurencyjnej. Wówczas każdy schemat może być przedstawiony jako drzewo. Na rysunku 1 podane są przedstawienia graficzne trzech schematów XML S_1, S_2 i S_3, oraz ich instancje, odpowiednio I_1, I_2 i I_3. Tak rozumiane schematy mogą być reprezentowane za pomocą formuł drzewiastych. Jeśli dodatkowo chcemy uwzględnić klucze zdefiniowane w XSD (XML Schema, 2004), to używamy formuły drzewiastej z kluczem.



Rysunek 1. Drzewa schematów XML $S1$, $S2$, $S3$ oraz ich instancje $I1$, $I2$ i $I3$

Przykład 1. *Schematy z rysunku 1 można zapisać jako formuły drzewiaste (formuła π_i reprezentuje schemat S_i):*

$$\pi_1(x_1, x_2, x_3, x_4) := /pubs[pub[title = x_1 \wedge year = x_2 \wedge author[name = x_3 \wedge university = x_4]]]$$
$$\pi_2(x_1, x_2, x_3) := /pubs[pub[title = x_1 \wedge author[name = x_2 \wedge university = x_3]]]$$
$$\pi_3(x_1, x_2, x_3) := /authors[author[name = x_1 \wedge paper[title = x_2 \wedge year = x_3]]]$$

Uwzględnienie informacji o kluczu wymaga użycie formuły drzewiastej z kluczem.

Przykład 2. *Dla schematu S_2 formuła drzewiasta z kluczem może mieć jedną z następujących postaci:*

$$\delta_2(x_1, x_2, x_3) := /pubs[pub(x_1)[title = x_1 \wedge author(x_1, x_2)[name = x_2 \wedge university = x_3]]] \ lub$$
$$\delta_2'(x_1, x_2, x_3) := /pubs[pub(x_1, x_2)[title = x_1 \wedge author(x_1, x_2)[name = x_2 \wedge university = x_3]]].$$

Formuła δ_2' mówi, że każde poddrzewo */pubs/pub* jest jednoznacznie identyfikowane przez parę wartości (x_1, x_2), a więc jest tyle poddrzew ile różnych par tych wartości. Dla każdej pary $(title, author)$ tworzone jest więc oddzielne poddrzewo. W przypadku formuły δ_2 każde poddrzewo */pubs/pub* jest jednoznacznie identyfikowane przez wartości (x_1), a więc wszyscy współautorzy grupowani są w obrębie publikacji, którą napisali. Podobnie możemy zdefiniować $\delta_1(x_1, x_2, x_3, x_4)$ i $\delta_3(x_1, x_2, x_3)$ specyfikujące strukturę i klucze dla schematów S_1 i S_3.

Jeśli $\pi(x_1, ..., x_n)$ jest formułą drzewiastą reprezentującą schemat S, to instancja I tego schematu jest w pełni określona przez zbiór Ω wartościowań zmiennych występujących w π. Każde wartościowanie $\omega \in \Omega$ przypisuje zmiennym wartości tekstowe lub wartość pustą $\perp$, jeśli wartość tekstowa nie jest określona.

Przykład 3. *Instancje schematów z rysunku 1 reprezentowane są przez następujące zbiory wartościowań:*
$\Omega_1 := \{(x_1 : \perp, x_2 : \perp, x_3 : \perp, x_4 : \perp)\}$,
$\Omega_2 := \{(x_1 : XML, x_2 : John, x_3 : NY), (x_1 : XML, x_2 : Ann, x_3 : LA)\}$,
$\Omega_3 := \{x_1 : Ann, x_2 : XML, x_3 : 2005)\}$

Formuły drzewiaste można wykorzystać do zdefiniowania mapowań (odwzorowań) między schematami. Mapowanie jest wówczas implikacją typu (2), gdzie po lewej stronie występuje koniunkcja formuły drzewiastej reprezentującej schemat źródłowy oraz zbioru równości określających zależności między zmiennymi, a po prawej formuła drzewiasta z kluczem reprezentująca schemat docelowy (Pankowski i in., 2007).

Definicja 2. *Mapowanie schematu S_i na schemat S_j jest wyrażeniem o postaci*

$$m_{ij} := \pi_i(\mathbf{x}) \wedge \kappa(\mathbf{x}, \mathbf{y}) \Rightarrow \delta_j(\mathbf{x}, \mathbf{y}), \tag{3}$$

gdzie domyślnie wszystkie zmienne są kwantyfikowane uniwersalnie, a symbole funkcyjne – egzystencjalnie (zgodnie ze wzorem (2)) oraz:

- $\pi_i(\mathbf{x})$ – *formuła drzewiasta reprezentująca schemat S_i;*
- $\kappa(\mathbf{x}, \mathbf{y})$ – *koniunkcja równości atomowych, tj. równości o postaci $x = x'$, $y = x$, $y = c$, $y = f(x_1, ..., x_n)$, $x, x', x_i \in \mathbf{x}$, $y \in \mathbf{y}$, c - stała (w tym $\perp$), określających zależności między wartościami zmiennych;*
- $\delta_j(\mathbf{x}, \mathbf{y})$ – *formuła drzewiasta z kluczem reprezentująca schemat docelowy.*

Przykład 4. *Przykłady mapowań między schematami S_1, S_2 i S_3:*

$$\begin{aligned}
m_{11}(x_1, x_2, x_3, x_4) &:= \pi_1(x_1, x_2, x_3, x_4) \wedge \kappa(x_1, x_2, x_3, x_4) \\
&\quad \Rightarrow \delta_1(x_1, x_2, x_3, x_4), \\
m_{21}(x_1, x_2, x_3, y) &:= \pi_2(x_1, x_2, x_3) \wedge \kappa(x_1, y, x_2, x_3) \Rightarrow \delta_1(x_1, y, x_2, x_3), \\
m_{23}(x_1, x_2, x_3, y) &:= \pi_2(x_1, x_2, x_3) \wedge \kappa(x_1, y, x_2, x_3) \Rightarrow \delta_3(x_2, x_1, y), \\
m_{32}(x_1, x_2, x_3, y) &:= \pi_3(x_1, x_2, x_3) \wedge \kappa(x_2, x_3, x_1, y) \Rightarrow \delta_2(x_2, x_1, y),
\end{aligned}$$

gdzie:

$$\kappa(x_1, x_2, x_3, x_4) := x_2 = f_Y(x_1) \wedge x_4 = f_U(x_3).$$

Znaczenie mapowań objaśnia przykład:

$$m_{23}(x_1, x_2, x_3, y) := \pi_2(x_1, x_2, x_3) \wedge \kappa(x_1, y, x_2, x_3) \Rightarrow \delta_3(x_2, x_1, y) = \\ /pubs[pub[title = x_1 \wedge author[name = x_2 \wedge university = x_3]]] \\ \wedge\, y = f_Y(x_1) \wedge x_3 = f_U(x_2) \\ \Rightarrow /authors[author[name = x_1 \wedge paper[title = x_2 \wedge year = y]]].$$

W mapowaniu m_{23} występują cztery zmienne, w tym trzy *źródłowe*, x_1, x_2, x_3, i jedna *docelowa* y. Zmienne x_3 i y są zmiennymi *zależnymi*, gdyż ich wartości są funkcyjnie zależne od, odpowiednio, x_2 i x_1. W tym przypadku konkretne postaci tych zależności nie są znane, ale wiadomo, że istnieją pewne funkcje oznaczane przez f_Y i f_U. W dalszym ciągu przyjmujemy, że są to funkcje Skolema. Wówczas, jeśli na przykład x_1 ma wartość $''XML''$, to $f_Y(x_1)$ przyjmuje wartość tekstową (term) $''f_Y(XML)''$, a więc wartość jest konkatenacją nazwy funkcji i jej argumentów.

Jak pokażemy w rozdziale 5, takie termy mogą być wykorzystane do odkrywania pewnych *brakujących danych* (ang. *missing data*). W ostatecznej postaci drzewa XML elementy o takich wartościach mogą być usuwane.

Definicja 3. *Niech $\pi(\mathbf{x})$ będzie formułą drzewiastą (z kluczem jawnym lub domyślnym) i niech atom $l = x$ występuje w $\pi(\mathbf{x})$. Wówczas etykietę l nazywamy typem zmiennej x, co oznaczamy $type_{\pi(\mathbf{x})}(x) = l$. Typem formuły drzewiastej nazywamy zbiór złożony z typów wszystkich jej zmiennych, tj.*

$$type(\pi(\mathbf{x})) = \{type_{\pi(\mathbf{x})}(x) \mid x \in \mathbf{x}\}.$$

Na przykład: $type_{\pi_2(x_1,x_2,x_3)}(x_1) = title$, $type(\delta_3) = \{name, title, year\}$.

4. Zapytania i reformułowanie zapytań

Definicja 4. *Niech Φ będzie formułą, w której występują równości atomowe zbudowane nad etykietami z pewnego zbioru L, zmiennymi z pewnego zbioru X, termami o postaci $f(x_1, ..., x_n)$ i stałymi. Niech $L' \subseteq L$ i $X' \subseteq X$. Projekcją formuły Φ na zbiór L' (na zbiór X'), co zapisujemy $\Pi_{L'}(\Phi)$ (lub $\Pi_{X'}(\Phi)$) nazywamy taką największą podformułę Φ' formuły Φ, w której dla każdej równości atomowej każda etykieta należy do L' (każda zmienna należy do X').*

Definicja 5. *Niech $m_{ij} := \pi_i(\mathbf{x}) \wedge \kappa(\mathbf{x}, \mathbf{y}) \Rightarrow \delta_j(\mathbf{x}, \mathbf{y})$ będzie mapowaniem schematu S_i w schemat S_j. Niech ponadto dane będą: (1) kwalifikator zapytania $\phi(\mathbf{x}')$, gdzie $\mathbf{x}' \subseteq \mathbf{x}$; (2) zbiór etykiet docelowych $K \subseteq type(\delta_j(\mathbf{x}, \mathbf{y}))$. Niech ponadto $\mathbf{z} \subseteq (\mathbf{y}, \mathbf{x})$ będzie zbiorem tych zmiennych, których typ należy do K lub są konieczne do wyznaczenia takich zmiennych. Wówczas zapytaniem nad schematem S_i do schematu S_j nazywamy wyrażenie $q_{ij} := query_{m_{ij}}(\phi(\mathbf{x}'), K)$, gdzie*

$$query_{m_{ij}}(\phi(\mathbf{x}'), K) = \Pi_{\mathbf{x}' \cup \mathbf{z}}(\pi_i(\mathbf{x})) \wedge \Pi_{\mathbf{x}' \cup \mathbf{z}}(\kappa(\mathbf{x}, \mathbf{y})) \wedge \phi(\mathbf{x}') \Rightarrow \Pi_K(\delta_j(\mathbf{x}, \mathbf{y})).$$

Przykład 5. *Zapytaniem nad schematem S_1 do S_1 jest na przykład:*

$$q_{11} := query_{m_{11}}(x_3 = \text{“}John\text{”}, \{title, year, name\}) = \\ /pubs[pub[title = x_1 \wedge year = x_2 \wedge author[name = x_3]]] \\ \wedge x_2 = f_Y(x_1) \wedge x_3 = \text{“}John\text{”} \Rightarrow \\ /pubs[pub(x_1)[title = x_1 \wedge year = x_2 \wedge author(x_1, x_3)[name = x_3]]]$$

W rozważanej przez nas klasie systemów integracji danych zapytanie jest formułowane względem pewnego schematu docelowego. Aby uzyskać pełną odpowiedź zapytanie to musi być przesyłane dalej do wszystkich jego partnerów, ci partnerzy przesyłają je z kolei do swoich partnerów itd. W ten sposób zapytanie może dotrzeć do wszystkich źródeł, które mogą dostarczyć jakąś składową ostatecznej odpowiedzi. Uzyskane odpowiedzi cząstkowe są krok po kroku scalane i przesyłane do początkowego schematu docelowego, względem którego sformułowano zapytanie. Odpowiedź jest ostatecznie redagowana i udostępniona użytkownikowi.

Ważnym problemem w omówionym procesie jest *reformułowanie* zapytań. Przypuśćmy, że zapytanie q_{ij} jest sformułowane w pojęciach schematu S_i i zwraca odpowiedź o strukturze zgodnej z pewną projekcją schematem S_j. Jeśli istnieje źródło S_k powiązane z S_i, to zapytanie q_{ij}, po odpowiednim przereformułowaniu, musi być najpierw wykonane w S_k. Uzyskana odpowiedź jest następnie scalana z instancją schematu S_i (dzięki czemu mogą być odkryte dane, które oddzielnie nie istnieją ani w S_k ani w S_i, ale w wyniku ich scalania mogą być wywnioskowane, patrz rozdz. 5). W końcu na wyniku scalania wykonywane jest zapytanie q_{ij} i wynik jest przekazywany do S_j (który może być schematem docelowym lub jest jednym z etapów pośrednich prowadzących do schematu docelowego).

Definicja 6. *Niech* $q_{ij} := \pi'_i \wedge \kappa' \wedge \phi \Rightarrow \delta'_j$ *będzie zapytaniem nad schematem* S_i *do schematu* S_j, *a* $m_{ki} := \pi_k \wedge \kappa \Rightarrow \delta_i$ *niech będzie mapowaniem* S_k *do* S_i. *Przeformułowaniem zapytania* q_{ij} *względem mapowania* m_{ki} *nazywamy zapytanie*

$$q_{ki} := rewrite_{m_{ki}}(q_{ij}) = query_{m_{ki}}(\phi[\mathbf{x} \rightarrow \sigma(\mathbf{x})], type(\pi'_i)),$$

gdzie $\phi[\mathbf{x} \rightarrow \sigma(\mathbf{x})]$ *powstaje z* $\phi(\mathbf{x})$ *w ten sposób, że każde wystąpienie zmiennej* x *w* ϕ *zastępowane jest zmienną* y, $y = \sigma(x)$, *jeśli* $type_{\pi'_i}(x) = type_{\delta_i}(y)$. *W ten sposób tworzony jest kwalifikator zapytania z takim podstawieniem zmiennych, które jest zgodne z przypisywaniem zmiennych stosowanym w mapowaniu* m_{ki}.

Przykład 6. *Dla zapytania* q_{11} *z przykładu 5, uzyskujemy następujące przeformułowania:*

$$\begin{aligned}
q_{21} = \;& /pubs[pub[title = x_1 \wedge author[name = x_2]]] \wedge y = f_Y(x_1) \wedge \\
& \quad x_2 = \text{“}John\text{”} \Rightarrow \\
& /pubs[pub(x_1)[title = x_1 \wedge year = y \wedge author(x_1, x_2)[name = x_2]]] \\
q_{32} = \;& /authors[author[name = x_1 \wedge paper[title = x_2]]] \wedge x_1 = \text{“}John\text{”} \Rightarrow \\
& /pubs[pub(x_2)[title = x_2 \wedge author(x_2, x_1)[name = x_1]]] \\
q_{31} = \;& /authors[author[name = x_1 \wedge paper[title = x_2 \wedge year = x_3]]] \wedge \\
& \quad x_3 = f_Y(x_2) \wedge x_1 = \text{“}John\text{”} \Rightarrow \\
& /pubs[pub(x_1)[title = x_1 \wedge year = x_3 \wedge author(x_1, x_2)[name = x_2]]] \\
q_{23} = \;& /pubs[pub[title = x_1 \wedge author[name = x_2]]] \wedge y = f_Y(x_1) \wedge \\
& \quad x_2 = \text{“}John\text{”} \Rightarrow \\
& /authors[author(x_2)[name = x_2 \wedge paper(x_2, x_1)[title = x_1 \wedge year = y]]]
\end{aligned}$$

Następujący algorytm tłumaczy każde zapytanie q_{ij}, gdzie po prawej stronie implikacji jest formuła drzewiasta z domyślną definicją kluczy, na język XQuery (XQuery 1.0, 2002) (w dialekcie zaimplementowanycm w MS SQL Server 2005).

Algorytm 1

Wejście: Zapytanie $q_{ij} := \pi_i \wedge \phi \Rightarrow \pi_j$, gdzie: $\pi_i := /top'[E']$, $\pi_j := /top[E]$.
Wyjście: Zapytanie w XQuery względem schematu S_i i zwracające odpowiedź zgodną z π_j.

$$ToXQuery(/top'[E'] \wedge \phi \Rightarrow /top[E]) = \begin{array}{l} <top>\{ \\ \quad \textbf{for } \$v \textbf{ in } /top', \\ \quad\quad \tau(\$v, E') \\ \quad\quad \textbf{where } \phi \\ \quad \textbf{return} \\ \quad\quad \rho(E)\} \\ </top> \end{array}$$

where:

1. $\tau(v, l = x) = \$x$ **in if** $(\$v[l])$ **then** $string(\$v/l[1])$ **else** $''null''$,
2. $\tau(v, l[E]) = \$v'$ **in if** $(\$v[l])$ **then** $\$v/l$ **else** $/$, $\tau(v', E)$,
3. $\tau(v, E_1 \wedge \cdots \wedge E_k) = \tau(v, E_1), \cdots, \tau(v, E_k)$,
4. $\rho(l = x) = <l>\{\$x\}</l>$
5. $\rho(l[E]) = <l>\rho(E)</l>$
6. $\rho_E(E_1 \wedge \cdots \wedge E_k) = \rho(E_1) \ \cdots \ \rho(E_k)$

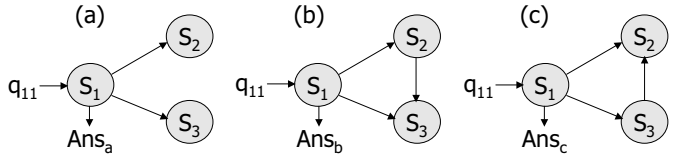
Dla zapytanie q_{31} algorytm powyższy tworzy następujące zapytanie w języku XQuery:

```
<pubs>{ for $_v in /authors,
  $_v1 in if ($_v[author]) then $_v/author else /,
    $x_1 in if ($_v1[name]) then string($_v1/name[1]) else "null",
    $_v22 in if ($_v1[paper]) then $_v1/paper else /,
      $x_2 in if ($_v22[title]) then string($_v22/title[1]) else "null",
      $x_3 in if ($_v22[year]) then string($_v22/year[1]) else "null"
  where $x_1 ="John"
  return
  <pub>
     <title>{$x_2}</title>
     <year>{$x_3}</year>
     <author>
        <name>{$x_1}</name>
     </author>
  </pub> }
</pubs>
```

5. Strategie wykonywania zapytań w systemie SIX-P2P

W tym rozdziale przedyskutujemy różne możliwe strategie wykonywania zapytań w systemie P2P. Pokażemy, że wybór strategii przetwarzania (sposób propagacji zapytań między partnerami) oraz stosowany sposób scalania danych cząstkowych ma istotny wpływ na ostateczną odpowiedź oraz na koszt przetwarzania.

Rozważmy różne możliwe strategie udzielania odpowiedzi na zapytanie q_{11} (przy-kład 5). Strategie te przedstawiono graficznie na rys. 2. W każdym przypadku zapytanie q_{11} formułowane jest względem schematu S_1, a następnie można postępować według jednej z poniższych strategii:



Rysunek 2. Trzy różne strategie udzielania odpowiedzi na zapytanie q_{11}

Strategia (a). Zapytanie q_{11} przesyłane jest do S_2 i S_3. Tam jest przeformułowane do, odpowiednio, zapytań q_{21} i q_{31}. Zapytania te są wykonywane na I_2 i I_3 i odpowiedzi $q_{21}(I_2)$ oraz $q_{31}(I_3)$ zwracane są do S_1. Tam odpowiedzi scalane ze sobą i z instancją I_1. Na tak scalonych danych wykonywane jest zapytanie q_{11} i w rezultacie uzyskujemy końcową odpowiedź Ans_a. Zapisujemy to następująco ($\oplus$ oznacza operację scalania, *merge*, dokumentów XML, tj. operację wyznaczania *supremum* lub *najmniejszego kresu górnego* (Pankowski, Hunt, 2005):

$$\begin{aligned}
Ans_a &= q_{11}(I_1 \oplus Ans_{21} \oplus Ans_{31}),\\
Ans_{21} &= q_{21}(I_2) = \{(title : XML, year : \bot, name : John)\},\\
&\quad\ q^D_{21}(I_2) = \{(year : f_Y(XML))\},\\
Ans_{31} &= q_{31}(I_3) = \{\},\\
Ans_a &= \{(title : XML, year : \bot, name : John)\}.
\end{aligned}$$

Zapytanie $q^D_{21}(I_2)$, pochodne względem $q_{21}(I_2)$ zwraca odpowiedzi zawierające tylko pola odpowiadające zmiennym pochodnym (zmiennej y typu $year$ w rozważanym przypadku). Jako wartości tych zmiennych udostępniane są wartości termów Skolema z równości definiujących zależności dla zmiennych pochodnych (występujące w formule κ). Dane te są następnie wykorzystane do odkrywania pewnych brakujących danych (patrz poniżej strategia (c)).

Strategia (b). Tym różni się od strategii (a), że partner S_2 propaguje zapytanie do S_3 i czeka na odpowiedź. Po uzyskaniu odpowiedzi $q_{32}(I_3)$ scala ją z instancją I_2, na wyniku scalenia wykonuje zapytanie q_{21} i odpowiedź przekazuje do S_1. Partner S_3 obsługuje dwa zapytania: q_{11} otrzymane od S_1 i q_{21} otrzymane od S_2. Zapytania te reformułuje otrzymując odpowiednio q_{31} (przeformułowanie q_{11} względem m_{31}) oraz q_{32} (przeformułowanie q_{21} względem m_{32}). Odpowiedzie $q_{31}(I_3)$ i $q_{32}(I_3)$ przesyła odpowiednio do S_1 i S_3. Łatwo pokazać, że stosowanie tej strategii daje ten sam wynik co w przypadku (a). Jednak koszty jej realizacji są większe.

$$Ans_b = \{(title : XML, year : \bot, name : John)\}.$$

Strategia (c). Tym różni się od strategii (b), że partner S_3 propaguje zapytanie do S_2 i czeka na odpowiedź. Dalsze kroki są analogiczne jak w przypadku (b).

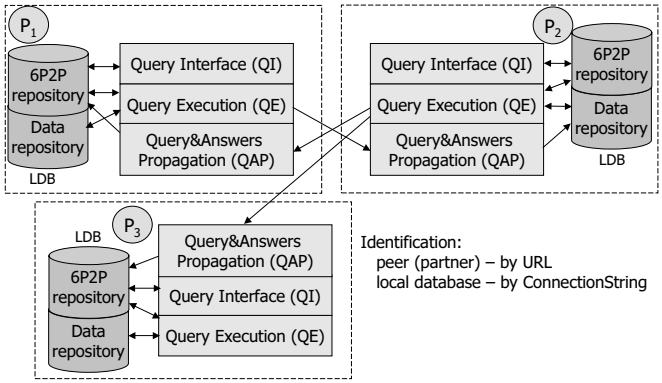
$$\begin{aligned}
Ans_c &= q_{11}(I_1 \oplus Ans_{21} \oplus Ans_{31}),\\
Ans_{23} &= q_{23}(I_2) = \{(title : XML, year : \bot, name : John)\},\\
&\quad\ q^D_{23}(I_2) = \{(year : f_Y(XML))\},\\
I_3 &= \{(title : XML, year : 2005, name : Ann)\}\\
&\quad\ I^D_3 = \{(year : f_Y(XML))\}\\
Ans_{31} &= q_{31}(I_3 \oplus Ans_{23}) =\\
&= \{(title : XML, year : 2005, name : John)\}\\
Ans_c &= \{(title : XML, year : 2005, name : John)\}.
\end{aligned}$$

Przy obliczaniu $I_3 \oplus Ans_{23}$ stosujemy metodę odkrywania (wywnioskowania) wartości pola *year* dla danej $(year : \perp)$ w odpowiedzi $q_{23}(I_2)$. Z analizy wartości *year* w zbiorach $q_{23}^D(I_2)$, I_3 i I_3^D wynika, że wartością tą jest 2005. Wynika stąd, że stosowanie strategii (c) daje lepszy wynik niż dwóch poprzednich.

Jak widać na rozważanych przykładach istotnymi decyzjami, które muszą być podjęte w procesie integracji danych jest: (1) określenie strategii propagacji zapytań; i (2) zdefiniowanie procesu scalania danych w taki sposób, aby możliwe było odkrywanie danych, które jawnie nie istnieją w żadnym ze scalanych źródeł (Pankowski, 2006).

6. Uwagi o architekturze i implementacji systemu SIX-P2P

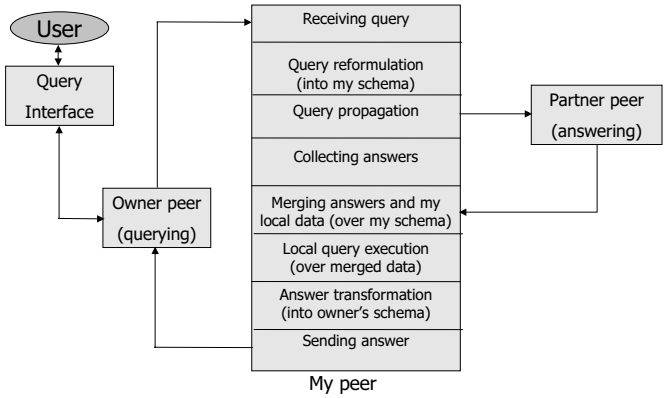
Realizacja omówionej metody przetwarzania realizowana jest w systemie SIX-P2P. Ogólna architektura systemu przedstawiona jest na rysunku 3, a architektura oprogramowania na rysunku 4.



Rysunek 3. Ogólna architektura systemu SIX-P2P

Każdy partner (ang. *peer*) w systemie posiada własną lokalną bazę daych złożoną z dwóch części: repozytorium danych przeznaczonych do udostępniania oraz repozytorium 6P2P, w którym pamiętane są informacje niezbędne do realizacji funkcji semantycznej integracji danych (a więc informacje o partnerach, mapowaniach schematów, otrzymanych odpowiedziach cząstkowych, itp.). Za pomocą interfejsu zapytań (QI) użytkownik formułuje zapytanie. Wewnętrzna reprezentacja zapytań ma postać zgodną z definicją 5. Moduł wykonywania zapytań (QE) steruje procesem reformułowania zapytań, propagacji zapytań do partnerów, scalaniem wyników częściowych i wnioskowaniem w celu odkrycia brakujących danych, a także przekazywaniem odpowiedzi częściowych do partnerów. Wykorzystuje w tym celu funkcje systmu zarządzania bazą danych. Komunikacja między partnerami (QAP) odbywa się z wykorzystaniem technologii usług sieciowych Web Services.

Oprogramowanie systemu SIX-P2P zostało zrealizowane w środowisku C# .NET 2.0 i MS SQL Server 2005. Poszczególne warstwy na rysunku 4 pokazują zadania realizowane przez oprogramowanie.



Rysunek 4. Architektura oprogramowania w systemie SIX-P2P

7. Podsumowanie

W pracy przedstawiliśmy formalne podstawy definiowania mapowań schematów i reformułowaniem zapytan w systemie semantycznej integracji danych w środowisku P2P. Podstawową ideą tego podejścia jest przedstawienie schematu XML za pomocą formuł drzewiastych. Wyróżniliśmy klasę formuł drzewiastych uwzględniających możliwość definiowania klucza schematu. Pokazaliśmy, w jaki sposób formuły drzewiaste mogą być wykorzystane do definiowania mapowań oraz do zdefiniowania ważnej klasy zapytań do danych XML. W definicji mapowania uwzględniamy również specyfikację zależności między wartościami danych, co ma istotne znaczenie w procesie odkrywania tzw. „brakujących danych". Zaproponowaliśmy metodę reformułowania zapytań, wykorzystywaną w przypadku ich propagowania w środowisku P2P. Przedstawiliśmy algorytm translacji zapytań języka XQuery.

Rozważyliśmy strategie propagacji zapytań i scalania odpowiedzi cząstkowych w procesie integracji danych. Odpowiedź na zapytanie udzielana jest poprzez propagowanie zapytania do wszystkich możliwych partnerów. Partnerzy źródłowi stają się następnie partnerami docelowymi i propagują zapytania dalej do swoich partnerów itd. Odpowiedzi na zapytania są zbierane, scalane i sukcesywnie przesy-łane do końcowego partnera docelowego. Jak pokazaliśmy na przykładzie, zarówno odpowiedź, jak i koszt jej uzyskania może zależeć od obranej strategii przetwarzania (tj. od tego, do jakich partnerów i w jakiej kolejności będziemy propagować zapytanie). Otwiera to nowe obszary badawcze w zakresie poszukiwania optymalnych strategii realizacji zapytań w systemie P2P.

W pracy omówiliśmy i zaproponowaliśmy nowe rozwiązania w zakresie modeli i metod semantycznej integracji danych w środowisku P2P, w szczególności w zakresie: (1) definiowania mapowań schematów; (2) reformułowania zapytań na podstawie zdefiniowanych mapowań: (3) scalania odpowiedzi częściowych. Opisana metoda zaimplementowana została w systemie SIX-P2P realizowanym w ramach grantu Ministerstwa Nauki i Szkolnictwa Wyższego nr 1553/T02/2006/31.

Literatura

Abiteboul, S., Hull, R., Vianu, V. (1995) Foundations of Databases. *Addison-Wesley*, Reading, Massachusetts.

Arenas, M., Libkin, L. (2005) XML Data Exchange: Consistency and Query Answering. *PODS Conference*, 13–24.

Fagin, R., Kolaitis, P. G., Popa, L. (2005) Data exchange: getting to the core. *ACM Trans. Database Syst.* **30**, 1, 174–210.

Fagin, R., Kolaitis, P. G., Popa, L., Tan, W. C. (2004) Composing Schema Mappings: Second-Order Dependencies to the Rescue. *PODS*, 83–94.

Haas, L. M. (2007) Beauty and the Beast: The Theory and Practice of Information Integration. *ICDT*, Lecture Notes in Computer Science **4353**, 28–43.

Haas, L. M., Hernández, M. A., Ho, H., Popa, L., Roth, M. (2005) Clio grows up: from research prototype to industrial tool. *SIGMOD*, 805–810.

Halevy, A. Y., Rajaraman, A., Ordille, J. J. (2006) Data Integration: The Teenage Years. *VLDB*, 9–16.

Pankowski, T. (2006) Management of executable schema mappings for XML data exchange. *Database Technologies for Handling XML Information on the Web*, Lecture Notes in Computer Science **4254**, 264–277.

Pankowski, T., Cybulka, J., Meissner, A. (2007) XML Schema Mappings in the Presence of Key Constraints and Value Dependencies. *ICDT 2007 Workshop EROW'07* CEUR Workshop Proceedings Vol. 229, 1–15.

Pankowski, T., Hunt, E. (2005) Data Merging in Life Science Data Integration Systems. *Intelligent Information Systems*, Advances in Soft Computing, Springer Verlag, 279–288.

XML Schema Part 1: Structures (2004) `www.w3.org/TR/xmlschema-1`.

XQuery 1.0: An XML Query Language. W3C Working Draft (2002) `www.w3.org/TR/xquery`.

Schema mappings and query reformulation in semantic data integration systems

In the paper we discuss the problem of semantic integration of data in P2P environment. In this setting each peer has its own database schema as well as mappings between the schema and some partners. The goal of integration is to answer a query formulated against any partner. We show how tree-pattern formulas can be used to specify mappings and reformulate queries in such settings. In particular, we show that the result of query answering may depend on a strategy of query propagation in P2P environment.