

Budowanie schematów relacyjnych dla danych OWL z wykorzystaniem silników wnioskujących

Maciej Falkowski, Czesław Jędrzejek
Instytut Automatyki i Inżynierii Informatycznej, Politechnika Poznańska,

falkowski81@gmail.com, czeslaw.jedrzejek@put.poznan.pl

Abstrakt. Ogromny wzrost liczby danych, którą przetwarzają dzisiejsze systemy związane z Internetem, a także luźna struktura tych danych wymagają rozwinięcia technik i narzędzi służących semantycznemu ich oznaczeniu i integracji. Techniki te umożliwiają stosowanie nowego rodzaju zapytań do danych, niemożliwych do obsłużenia w obecnych systemach. Co prawda ontologie i związane z nimi narzędzia były początkowo rozwijane z myślą o danych w sieci Web (SemanticWeb), ale ich zastosowanie jest uniwersalne, np. przy integracji heterogenicznych baz danych. Ontologie i w ogólności dane semantycznie oznaczone muszą być składowane – naturalnymi narzędziami do składowania danych są natomiast relacyjne bazy danych. Nie tylko ontologie mogą korzystać z baz danych, ale również bazy danych mogą wykorzystywać ontologie. Możliwości zaawansowanego wyszukiwania danych wykorzystującego zależności opisywane przez ontologie są na tyle duże, że producenci baz danych zaczęli w swoich produktach implementować dla nich wsparcie. W artykule przedstawiona jest idea ontologii na tle schematów relacyjnych baz danych. Następnie omówione są korzyści, jakie płyną z jej wykorzystania na przykładzie prostego zastosowania. W dalszej kolejności omawiany jest przykład implementacji bazujący na narzędziach wprowadzonych w Oracle 10g oraz przedstawiono koncepcję implementacji zbliżonej funkcjonalności za pomocą widoków bazodanych. Artykuł kończy porównanie obydwu metod składowania i przetwarzania ontologii.

Informacja o autorze. Prof. dr hab. inż. Czesław Jędrzejek - w początkowym okresie pracy związany z AGH i UJ w Krakowie. Przez okres 10 lat odbywał staże naukowe i pracował jako Visiting Professor kolejno na kilku uczelniach w USA. W latach 1999-2004 zajmował stanowisko Wiceprezesa Zarządu firmy ITTI w Poznaniu. Jest autorem lub współautorem około 150 publikacji. Kierował kilkudziesięcioma projektami dla wiodących operatorów oraz dostawców sprzętu telekomunikacyjnego w Polsce w zakresie ewolucji sieci i usług, inżynierii ruchu w sieciach teleinformatycznych oraz wykonania, integracji i wdrożenia systemów informatycznych. Od 2003 r. zajmuje stanowisko profesora w Instytucie Automatyki i Inżynierii Informatycznej Politechniki Poznańskiej w Poznaniu. Realizował kilka projektów europejskich dotyczących aplikacji informatycznych.

Mgr inż. Maciej Falkowski k jest pracownikiem Instytutu Automatyki i Inżynierii Informatycznej Politechniki Poznańskiej. W kręgu jego zainteresowań znajduje się szereg zagadnień związanych z semantyką danych, technologiami semantycznymi oraz przetwarzaniem metadanych.

1. Ontologie a schematy relacyjne

Pojęcie ontologii w dziedzinie nauk informatycznych zdobywa ostatnio dużą popularność. Ontologię najczęściej definiuje się jako formalną specyfikację konceptualizacji pewnej dziedziny. W tej definicji można się doszukać pewnej analogii do relacyjnego schematu bazy danych, który również jest pewną konceptualizacją jakiejś dziedziny. Różnice pomiędzy nimi wynikają z odmiennych zastosowań ontologii i schematów relacyjnych. Konceptualizacja do schematów relacyjnych ma na celu przygotowanie struktur, które w wydajny sposób będą mogły przechowywać dane o instancjach bytów dziedziny. Modelowanie zdeterminowane jest poprzez specyfikę i ograniczenia systemów bazodanowych i ukierunkowane na wydajność przetwarzania w przewidywanych zastosowaniach. Stosuje się tu podział schematów, normalizację i inne techniki, które dopasowują koncepcyjny schemat do wymogów przetwarzania. Dobrze zdefiniowania, formalna semantyka schematów ma w tym przypadku drugorzędne znaczenie, gdyż celem zastosowań są pojedyncze, zamknięte systemy i aplikacje, których autorzy są często autorami samych schematów. W takim przypadku znaczenie poszczególnych tabel i kolumn i ich odwzorowanie w rzeczywistość musi być znane jedynie małej grupie osób związanych z implementacją. Celem autora ontologii jest natomiast stworzenie struktury, której odbiorcami będą szerokie grupy ludzi i systemów. Można na nią patrzeć jako na zbiór konceptów, które opisują byty określonej domeny. Koncepty te są najczęściej pogrupowane w hierarchie za pomocą relacji subsumcji zachodzącej między tymi konceptami. Zastosowaniem ontologii jest automatyczne przetwarzanie maszynowe zawartej w niej wiedzy dziedzinowej i zależności pomiędzy konceptami. Dlatego pierwszorzędne znaczenie mają w tym przypadku takie cechy, jak: ścisła, formalna specyfikacja, precyzyjnie i jednoznacznie określona semantyka konceptów oraz zapis w języku, który pozwala na wydajne przetwarzanie (wnioskowanie).

Aby specyfikacja konceptualizacji była formalna, konieczny jest język do jej wyrażania. Zarówno schematy baz danych, jak i ontologie są przetwarzane przez maszyny, więc formalny język jest dla ich wyrażania niezbędny. W dziedzinie baz danych mogą do tego służyć diagramy związków encji czy skrypty języka SQL definiujące tabele. Do zapisu ontologii istnieje wiele języków, jednak obecnie wiodącymi są propozycje W3C, czyli najprostszy język RDFS [BrGu04] oraz znacznie go rozszerzająca rodzina języków OWL [PHH04]. W językach tych podstawowym konceptem jest koncept klasy. Klasa jest rozumiana jako pewien nazwany zbiór bytów. Pewną analogią do klasy w dziedzinie baz danych jest tabela. Tabela relacyjna określa właściwości zawartych w niej bytów (krotek) poprzez swoje kolumny. Semantycznym odpowiednikiem kolumn w omawianych językach jest pojęcie właściwości (inaczej relacji). Informacje w RDFS i OWL wyrażane są w modelu danych RDF [KlCa04]. Atomową częścią jest w nim trójka, składająca się z podmiotu, relacji i obiektu. Informację relacyjną o tym, że krotka o ID równym 5 z tabeli Osoba ma w kolumnie Wiek wartość 26 można wyrazić w modelu RDF jako trójkę (Osoba_5, maWiek, 26). Byty w schematach relacyjnych są identyfikowane dwuetapowo – poprzez wybór tabeli a następnie wybór konkretnej krotki za pomocą identyfikatora. W przytoczonym przykładzie tych dwóch elementów identyfikacji utworzono jeden globalny identyfikator.

Subsumcja zachodząca między dwoma klasami oznacza, że klasa subsumowana jest jakąś specjalizacją klasy subsumującej. Na przykład klasa matka może być specjalizacją klasy kobieta określoną w ten sposób, że matka to taka kobieta, która posiada dziecko. Z takiej definicji wynika, że każda matka to kobieta, a więc klasa Kobieta subsumuje klasę Matka. Subsumcja to podstawowa relacja ontologiczna, której w brakuje w tradycyjnych systemach bazodanowych. Duża różnica pomiędzy ontologiami a schematami relacyjnymi istnieje też w rozumieniu związku klasy z właściwością (ontologia) a tabeli z kolumną (schemat relacyjny). W tym drugim przynależność bytu do danej tabeli określa, jakie ten byt może (lub musi) posiadać cechy, nazwane poprzez kolumny tej tabeli. Nie jest możliwe w tym modelu przypisanie krotce wartości cechy, która nie ma odpo-

wiednika w jakiejś kolumnie. W ontologiach bazujących na OWL model danych jest dużo bardziej elastyczny i związek między klasą bytu a jego właściwościami jest inny. Klasa nie definiuje możliwych dla jej indywiduów właściwości, ale raczej właściwości danego bytu określają jego przynależność do klas. Na przykład jeśli wiadomo jest, że dany obiekt jest klasy kobieta i wiadomo, że posiada dziecko, to z tych informacji wynika, że obiekt ten jest też klasy matka. Byty mogą być instancjami dowolnej liczby klas. Przedstawiona różnica jest kluczowa w problematyce konstruowania schematów relacyjnych dla danych OWL.

Jak już wspomniano, w schematach relacyjnych jawne i formalne wyspecyfikowanie semantyki nie jest kluczowe, gdyż grupa bezpośrednich odbiorców takiego schematu jest stosunkowo wąska i do przekazania znaczeń poszczególnych tabel i kolumn nie są konieczne specjalne narzędzia. Ontologie natomiast w zamierzeniu mają być podstawą do semantycznego oznaczania danych dostępnych szerokim gronom użytkowników na całym świecie. Prosty przykład może być właściwość ‘data’ produktów spożywczych – można ją interpretować jako datę przydatności do spożycia, datę produkcji albo jeszcze inaczej.

Ostatnią z najważniejszych różnic pomiędzy schematami relacyjnymi a ontologiami jest siła wyrazu języków, służących do ich zapisu. Języki zapisu ontologii z rodziny OWL bazują logice deskrypcyjnej. Dzięki temu zapisy wykonane w tych językach mogą być interpretowane zgodnie z zasadami tej logiki, co umożliwi automatyczne ich przetwarzanie w systemach wnioskujących i wyciąganie dodatkowych informacji na podstawie już istniejących. Ta właściwość pozwala na wygodne i efektywne budowanie hierarchii klas i właściwości oraz ich odpytywanie z uwzględnieniem wnioskowania. Prosty przykład wnioskowania może obejmować wnioskowanie o przynależności do klas. Wiedząc, że klasa ssak jest podklasą klasy zwierzę, a klasa pies jest podklasą klasy ssak, można wywnioskować, że pies jest zarówno ssakiem i zwierzęciem. Istnieje wiele implementacji systemów wnioskujących, pozwalających efektywnie korzystać z ontologii.

2. Wsparcie dla technologii semantycznych w Oracle Spatial 10g – SDO_RDF_MATCH

Użyteczność danych semantycznych i ich związek z bazami danych zauważyli producenci oprogramowania bazodanowego. Ontologie służą do opisywania pewnych klas zależności między bytami, a we współczesnym świecie ogromna większość informacji gromadzona jest w relacyjnych bazach danych. Korzystanie z tych danych i ich odpytywanie może być wspomagane poprzez użycie ontologii. Jedną z wiodących na rynku implementacją jest produkt Oracle Spatial 10g umieściła elementy wykorzystujące model danych RDF i pozwalające przechowywać oraz wykorzystywać w zapytaniach proste ontologie. W tym punkcie przedstawiony zostanie system operowania na ontologiach firmy Oracle oraz pokazane zostaną korzyści, jakie standardowe aplikacje bazodanowe w codziennych zastosowaniach mogą odnieść po włączeniu w procesy przetwarzania danych zawartych w ontologiach.

W tym celu rozważmy system wspomagający użytkowników w wyborze restauracji. System taki może posiadać bazę danych lokali wraz z serwowanymi przez nie rodzajami potraw. W przykładzie przyjęto, że rodzaje te określane są przez narodowość kuchni, z której się wywodzą. Restauracje mogą więc serwować dania chińskie, polskie, włoskie itp. Niech rozważany system posiada tabelę Restauracje jak w Tabeli 1, w której poszczególnym restauracjom przypisane są obsługiwane kuchnie narodowe.

Tabela 1. Tabela Restauracje

Restauracje	
ID restauracji	Kuchnia
1	amerykańska
2	meksykańska

Restauracje	
ID restauracji	Kuchnia
2	amerykańska
3	portugalska
3	polska

System operujący na takich informacjach może działać sprawnie pod warunkiem, że klient poda dokładnie, jakiej kuchni (jednej lub wielu) oczekuje. Przykładowe zapytanie SQL do takiego schematu:

```
SELECT * FROM Restauracje WHERE Kuchnia = 'amerykańska'
```

Taki system nie uwzględnia jednak podobieństw, jakie występują pomiędzy pewnymi kuchniami. Np. zapytanie:

```
SELECT * FROM Restauracje WHERE Kuchnia = 'latynoamerykańska'
```

nie zwróciłoby żadnego wyniku, mimo, że kuchnia meksykańska, obsługiwana przez jedną z restauracji, zalicza się do kuchni latynoamerykańskich. Powiązania tego typu bardzo dobrze można wyrazić za pomocą ontologii i relacji subsumcji. Przykładowe powiązania między kuchniami (taksonomię) dla przykładu pokazuje Rysunek 1.



Rys. 1. Ontologia kuchni

Kategorie (klasy) umieszczono w owalach, natomiast instancje klas oznaczone są samym tekstem. Klasy tworzą hierarchię coraz bardziej szczegółowych określeń, a pewne kuchnie należą do więcej niż jednej klasy. Gdyby w zapytaniu wykorzystać te dodatkowe informacje, zapytania klientów mogłyby być pełniej obsługiwane, a system pozwalał na zadawanie ogólniejszych zapytań, włączając w nie nie tylko same nazwy kuchni, ale również ich klasy. Gdyby w bazie danych w jakiś sposób odwzorowany był powyższy graf zależności, odpowiedź na pytanie o kuchni latynoamerykańskie byłaby taka, jak przedstawiona w Tabeli 2.

Tabela 2. Odpowiedzi na zapytanie uwzględniające ontologię

ID restauracji	Kuchnia
2	meksykańska
3	portugalska

Informacje dotyczące klas i ich hierarchii można zaimplementować w bazie danych bez używania dodatkowych narzędzi. Wiedzę dziedziczną na temat kategoryzacji kuchni narodowych można umieścić w specjalnie przygotowanych tabelach i tak zmodyfikować zapytania SQL, aby te dodatkowe informacje uwzględnić. O ile dla prostych przypadków, jak ten w przykładzie, może to

być stosunkowo łatwe, o tyle w bardziej złożonych systemach problem może okazać się nie trywialny. W obydwu przypadkach warto rozważyć użycie dedykowanego systemu, jakim jest część Oracle Spatial 10g.

Na rozszerzenie oprogramowania Oracle w kierunku danych semantycznych i ontologii składają się dwa główne, powiązane ze sobą elementy. Pierwszy z nich to możliwość importowania i składowania danych semantycznych RDF w bazie, bez potrzeby budowania własnych schematów. Model RDF różni się od modelu relacyjnego, jednak możliwe są odwzorowania między nimi i dane z jednego modelu można bez straty konwertować na drugi i odwrotnie. W tym zakresie Oracle zastosował najprostszą, ale i najbardziej uniwersalną technikę – tzw. vertical table [DPB07], z pewnymi dodatkowymi modyfikacjami, jak normalizacja nazw. Dzięki temu dane takie jak ontologie mogą być bezpośrednio wczytywane przez system bazodanowy. Aby w bazie danych Oracle umożliwić korzystanie z tych funkcji, należy wydać jako administrator polecenia:

```
CREATE TABLESPACE rdf_tblspace
DATAFILE '/oradata/orcl/rdf_tblspace.dat' SIZE 1024M REUSE
AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED
SEGMENT SPACE MANAGEMENT AUTO;
```

Powyższe polecenia utworzą odrębną przestrzeń tablicową, w której przechowywane będą tabele związane z danymi ontologicznymi. W przestrzeni tej należy utworzyć strukturę dla przechowywania danych, bez której operacje dodawania są nieaktywne:

```
EXECUTE SDO_RDF.CREATE_RDF_NETWORK('rdf_tblspace');
```

Następnie trzeba utworzyć tabelę, w której będą przechowywane dane modelu RDF. Trójkom, czyli pojedynczym, atomowym porcjom informacji modelu RDF przypisano nowy typ danych – SDO_RDF_TRIPLE_S. Tabela do ich przechowywania musi posiadać kolumnę właśnie tego typu. Zalecane jest ponadto, aby miała ona oprócz kolumny reprezentującej trójki, także kolumnę ID. Podane poniżej polecenie tworzy tabelę COUISINE_RDF_DATA o dwóch polach, id i triple.

```
CREATE TABLE COUISINE_RDF_DATA (id NUMBER, triple SDO_RDF_TRIPLE_S);
```

Na bazie tej tabeli można w ostatnim kroku utworzyć model RDF. Parametrami polecenia są kolejno: nazwa modelu, nazwa tabeli, gdzie model będzie przetrzymywany, oraz nazwa kolumny w tej tabeli, która odpowiada za składowanie trójek.

```
EXECUTE SDO_RDF.CREATE_RDF_MODEL('COUISINE', 'COUISINE_RDF_DATA', 'TRIPLE');
```

Terminem model w nomenklaturze RDF określa się zbiór danych. Można tworzyć wiele różnych modeli o różnym przeznaczeniu – osobny model dla systemu restauracji, osobny z kategoriami towarów itp. Tak przygotowany model można wypełnić danymi RDF, na przykład ontologią.

Przygotowanie tabeli i modelu dla ontologii to pierwszy krok w kierunku ich wykorzystania w systemie. Kolejne, niezbędne elementy to możliwość wnioskowania oraz możliwość zadawania zapytań. Wnioskowanie może odbywać się na podstawie danych z jednego lub kilku modeli oraz jednego lub wielu zestawów reguł. Firma Oracle dołączyła zestawy reguł odpowiedzialne za wnioskowanie w najprostszym systemie – RDFS. W procesie wnioskowania zostają na podstawie reguł oraz dostępnych danych wywodzone nowe dane. W przypadku ontologii taki proces z opisów poszczególnych pojęć tworzy ich taksonomię.

W Oracle stosowany jest algorytm wnioskowania wprzód. Działa on w ten sposób, że w fazie wstępnej, przez zadaniem jakiegokolwiek zapytania, zostają wywiedzione i zapamiętane wszystkie możliwe nowe fakty. Fakty te są następnie wykorzystywane podczas szukania odpowiedzi. Kluczowe jest to, że wnioskowanie jest przeprowadzane tylko raz. Przed wykorzystaniem danych

wzbogaconych o dodatkowe, wywnioskowane fakty, należy dla nich utworzyć tzw. indeks polececiem:

```
SDO_RDF_INFERENCE.CREATE_RULES_INDEX(
  'rdfs_rlx_cuisine',
  SDO_RDF_Models('cuisine'),
  SDO_RDF_Rulebases('RDFS')
);
```

Indeks ten to właśnie baza wywnioskowanych danych. Argumenty to kolejno: nazwa indeksu, modele w których przeprowadzane jest wnioskowanie, oraz zestawy reguł. Dla każdej kombinacji zbioru modeli i zbioru reguł, które mają być wykorzystane w zapytaniach, trzeba utworzyć osobny indeks. Operacja tworzenia indeksu związana jest z przeprowadzeniem wnioskowania; w rzeczywistości indeks jest niczym innym, jak zbiorem wszystkich stwierdzeń, które można wywnioskować z danych źródłowych i zestawu reguł. Model działania jest więc taki, że wnioskowanie odbywa się jednokrotnie, w momencie wydania polecenia utworzenia indeksu, a przy późniejszym odpytywaniu modeli wykorzystywane są wyniki tego procesu, składowane w osobnej tabeli. Zaletą tego podejścia są bardzo małe opóźnienia przy odpytywaniu. Do wad można natomiast zaliczyć długi czas przygotowywania indeksu i duże wymagania pamięciowe. Ponadto każda zmiana danych źródłowych lub reguł unieważnia indeks – trzeba go przebudować. W przypadku danych lub reguł, które zmieniają się często takie ograniczenie może powodować mniejszą przydatność praktyczną. Jednak w przypadku ontologii, które pozostają niezmiennie w czasie, nie ma to dużego znaczenia.

Drugim elementem, pozwalającym efektywnie korzystać z przechowywanych RDFowych danych, jest funkcja tablicowa SDO_RDF_MATCH. Umożliwia ona pełne i efektywne wykorzystanie potencjału danych semantycznych i ontologii oraz ich włączanie do tradycyjnych zapytań SQL. Korzystanie z danych ontologicznych nieodłącznie wiąże się z wnioskowaniem – do systemu dołączono zestawy reguł odpowiadających logice wnioskowania RDF oraz RDFS, dzięki czemu ontologie zapisane w tych językach są w pełni użyteczne. Dodatkowo istnieje możliwość definiowania własnych zestawów reguł i ich stosowania wraz z tymi dostarczonymi, co znacznie zwiększa elastyczność systemu.

System firmy Oracle pozwala w łatwy sposób nie tylko wprowadzać takie dodatkowe informacje (zapisane w postaci ontologii), ale także na łatwe formułowanie zapytań uwzględniających ontologiczne dane. W zapytaniach używany jest trójkowy model danych RDF. Do wyrażenia subklasowości pomiędzy pojęciami służy w nim relacja rdf:subClassOf. Przykładowe zapytanie o kuchnię latynoamerykańską mogłoby wyglądać jak poniżej.

```
SELECT * FROM Restauracje r
WHERE r.Kuchnia IN
TABLE (SDO_RDF_MATCH(
  '?kuchnia rdf:subClassOf latynoamerykańska',
  SDO_RDF_Models('cuisine'),
  SDO_RDF_Rulebases('RDFS'),
  null,
  null));
```

Funkcja SDO_RDF_MATCH zwraca tabelę powstałą poprzez zadanie zapytania do wyspecyfikowanego modelu (zbioru danych RDF). W tym przypadku zapytanie jest o wszystkie kuchnie, które są podklasą kuchni latynoamerykańskiej w modelu 'cuisine'. Wnioskowanie o podklasowości odbywa się na zasadach RDFS. Argumentami funkcji SDO_RDF_MATCH są kolejno: szukana w ontologii zależność, zbiór odpytywanych modeli, zbiór wykorzystanych reguł, zbiór aliasów dla przestrzeni nazw oraz zbiór filtrów. Wzorzec zależności ma składnię podobną do SPARQL [PrSe07] – składa się ze zbioru trójek, w każdej trójce zmienne zaczynają od znaku zapytania. W wyniku wykonania tej funkcji powstaje tabela, w której każdej zmiennej odpowiada kolumna.

Krotki tej tabeli to wartościowania zmiennych z zapytania, powstałe w procesie poszukiwania dopasowań do wzorca w przeszukiwanym modelu.

Dodatkowe dane o zależnościach między kuchniami i ich kategoriach można by wyrazić w modelu relacyjnym i samodzielnie wprowadzić w pewnej postaci do bazy, a następnie wykorzystać w zapytaniach, jednak wbudowany mechanizm znacznie ułatwia zarówno wprowadzanie danych jak i późniejsze zadawanie zapytań. Równoważne powyższemu zapytanie wyrażone w samym SQL mogłoby wyglądać jak poniżej:

```
SELECT * FROM Restauracje
WHERE Kuchnia IN
(SELECT term1 FROM relationships
START WITH
term2 = 'latynoamerykańska' AND
relation = 'subClassOf'
CONNECT BY
PRIOR term1 = term2 AND
relation = 'subClassOf');
```

System musiałby oczywiście posiadać tabelę 'relationships', w której zapisane byłyby relacje subsumcji między klasami. Powyższe zapytanie wykorzystuje tranzytywne domknięcie, rozszerzenie wprowadzone w SQL'99, które pozwala na tworzenie tabel na podstawie rekursywnych zapytań. Postać SQL dla tak prostego zapytania nie jest trywialna, a dla bardziej rozbudowanych przypadków z wieloma wystąpieniami warunków SQL i skomplikowanego zapytania w dziedzinie ontologii skonstruowanie samodzielnie zapytania byłoby stosunkowo trudne.

3. Alternatywne rozwiązanie – implementacja reguł wnioskowania w definicjach widoków

Omówiony system firmy Oracle pozwala włączyć do zapytań SQL dane pochodzące z ontologii i wykorzystać je do budowania nowego typu warunków. Istnieje wiele alternatywnych dla tego podejścia rozwiązań pozwalających na wzbogacanie standardowych zapytań wiedzą zawartą w ontologiach. Rozwiązania te wykorzystują zewnętrzne silniki wnioskujące, translatory zapytań lub inne narzędzia.

Jednym z pomysłów na implementację tego typu funkcjonalności jest reprezentacja wszystkich danych ontologicznych w specjalnie do tego przygotowanych tabelach i konstruowanie zapytań z wykorzystaniem zawartych w nich danych. Instancjom każdej klasy przypisuje się dedykowane tabele, a następnie stosuje widoki do zbudowania hierarchii pojęć odpowiadającej hierarchii w ontologii. Dla każdej klasy należy wygenerować osobną tabelę o nazwie powstałej według ustalonego schematu na podstawie nazwy klasy. Dzięki temu budując zapytania wiadomo, w jakiej tabeli szukać instancji danej klasy. Tabela klasy powinna zawierać pole, w które będą wpisywane nazwy (identyfikatory) jej instancji. Tak powstała struktura pozwala przechowywać w relacyjnej bazie danych informacje o instancjach klas, nie pozwala jednak na wykorzystanie reguł związanych z subsumcją. Do utworzenia taksonomii opartej na subsumcji należy w następnym kroku użyć silnika wnioskującego, na przykład RacerPro. Na podstawie wygenerowanej przez niego hierarchii klas i właściwości można w dalszych krokach zdefiniować widoki dla każdej klasy. Nazwy widoków również muszą być konstruowane według ustalonego schematu. Na elementy widoku dedykowanego klasie C powinny składać się wszystkie krotki tabeli dedykowanej klasie C oraz wszystkie krotki klas, które są subsumowane przez klasę C. Te drugie znajdują się w widokach odpowiednich klas. W ten sposób logika zawierania się klas może być zaimplementowana w definicjach widoków. Przykład fragmentu definicji struktury dla hierarchii z rysunku 1 przedstawiono poniżej.

```
--tabele odpowiadające klasom
CREATE TABLE europejska ( Nazwa VARCHAR(50) )
CREATE TABLE zachodnia ( Nazwa VARCHAR(50) )
CREATE TABLE kuchnia ( Nazwa VARCHAR(50) )
CREATE TABLE latynoamerykańska ( Nazwa VARCHAR(50) )
CREATE TABLE dalekowschodnia ( Nazwa VARCHAR(50) )
CREATE TABLE azjatycka ( Nazwa VARCHAR(50) )
CREATE TABLE południowoazjatycka ( Nazwa VARCHAR(50) )

--widoki - implementacja hierarchii klas
CREATE VIEW europejska_v AS ( SELECT * from europejska)
CREATE VIEW zachodnia_v AS ( SELECT * from zachodnia UNION SELECT * FROM eu-
ropejska_v)
CREATE VIEW kuchnia_v AS ( SELECT * from kuchnia UNION SELECT * FROM zachod-
nia_v UNION SELECT * FROM azjatycka_v UNION SELECT * FROM latynoamerykańska_v)
CREATE VIEW latynoamerykańska_v AS ( SELECT * from latynoamerykańska)
CREATE VIEW dalekowschodnia_v AS ( SELECT * from dalekowschodnia)
CREATE VIEW azjatycka_v AS ( SELECT * from azjatycka UNION SELECT * FROM dale-
kowschodnia_v UNION SELECT * FROM południowoazjatycka_v)
CREATE VIEW południowoazjatycka_v AS ( SELECT * from południowoazjatycka)
```

Definicje tabel oraz widoków można łatwo wygenerować automatycznie. Ostatnim krokiem jest wypełnienie go instancjami, po czym zapytanie o kuchnie latynoamerykańskie można sformułować jak poniżej:

```
SELECT * FROM Restauracje
WHERE Kuchnia IN
(SELECT Nazwa FROM latynoamerykańskie_v)
```

W ten sposób utworzono system, którego funkcjonalność w przykładowym zastosowaniu odpowiada funkcjonalności narzędzia Oracle. Jednak możliwości ekspresji ontologii nie ograniczają się jedynie do prostej relacji subsumcji. Podstawowym pytaniem w dziedzinie ontologii jest pytanie o przynależność indywiduum do pewnej klasy. Przynależność ta może być deklarowana explicite lub wnioskowana na podstawie zasad logiki deskrypcyjnej. Jeśli jest zadeklarowana, wtedy przy wypełnianiu schematów instancje klas trafiają do odpowiednich tabel i nie ma problemu z określeniem takiej przynależności. W przypadku wnioskowanej przynależności do klasy, proces wnioskowania może opierać się na dwóch rodzajach stwierdzeń ontologicznych. Pierwszy rodzaj to stwierdzenia dotyczące subsumcji klas, które było stosowane w przedstawianych do tej pory przykładach. Wnioskowanie w tym trybie polega na sprawdzeniu jaka jest zadeklarowana klasa (lub klasy) bytu i sprawdzeniu, czy któraś z tych klas jest podklasą klasy użytej w zapytaniu. Aby uzyskać odpowiedź, wystarczy znajomość taksonomii, takiej jak na rysunku 1 i takiej, jak w zaimplementowanych widokach.

Drugi rodzaj stwierdzeń ontologicznych, które mogą być wykorzystywane we wnioskowaniu to definicje klas. Klasy mogą być definiowane jako sumy, różnice czy przecięcia innych klas (np. klasa Kwadrat może być przecięciem klas Prostokąt i WielokątForemeny). Takie definicje również łatwo jest zaimplementować w postaci widoków. Sumę klas można wyrazić za pomocą słowa kluczowego UNION, natomiast przecięcie i różnicę za pomocą słowa kluczowego IN. Definicja widoku kuchni amerykańskich jako zachodnich, ale nie europejskich mogłaby wyglądać jak poniżej:

```
CREATE VIEW amerykańskie_v AS ( SELECT * from zachodnia_v WHERE Nazwa NOT IN
(SELECT * FROM europejska_v))
```

Inną możliwą definicją klasy jest definicja poprzez wymagane właściwości jej instancji, jak w przytaczanym już przykładzie matki – matka to kobieta, która ma co najmniej jedno dziecko, gdzie fakt posiadania dzieci wyrażany jest za pomocą właściwości, np. maDziecko. Aby odpowiedzieć na tego typu zapytania nie wystarczy jedynie analiza taksonomii klas. Konieczna jest również analiza i wnioskowanie w ich definicjach - opisach warunków, jakie muszą spełniać in-

dywidua, aby można je było zaliczyć do instancji danej klasy. Zaprezentowane do tej pory schematy nie pozwalają na przechowywanie informacji o relacjach. Ponieważ wszystkie relacje (właściwości) w RDF są binarne, tabele im dedykowane muszą posiadać dwie kolumny – podmiot relacji (jego identyfikator) oraz obiekt relacji. Przykładowa tabela do przechowywania relacji maDziecko:

```
CREATE TABLE maDziecko ( podmiot VARCHAR(50), obiekt VARCHAR(50))
```

Z taką tabelą w systemie można utworzyć widok matka_v (zakładając, że istnieje widok kobieta_v) następująco:

```
CREATE VIEW matka_v AS SELECT Nazwa FROM kobieta_v WHERE Nazwa IN (SELECT podmiot FROM maDziecko)
```

Taka definicja widoku pozwala przypisać do klasy byty na podstawie ich właściwości. Właściwości te muszą więc być znane systemowi wnioskującemu, który na nich operuje. W przypadku implementacji firmy Oracle oznacza to konieczność włączenia tych danych do modelu RDF, wczytania ich do tablicy trójek oraz utworzenia dla nich indeksu, czyli przeprowadzenia wnioskowania. Takie postępowanie może być problematyczne w przypadku, gdy wnioskowanie miało by objąć również dane natywnie przechowywane w modelu relacyjnym, gdyż oznacza de facto konieczność transformacji danych z modelu relacyjnego na model RDF. Transformacja do modelu RDF oraz ponowne załadowanie tych samych danych, ale w innym modelu może być z różnych przyczyn kłopotliwe. Po pierwsze, oznacza to wykonanie kopii wszystkich danych, a co za tym idzie co najmniej dwukrotny wzrost objętości bazy, ponadto powoduje problemy synchronizacji obydwu kopii i inne.

Omawiany alternatywny system w zaprezentowanej do tej pory postaci również posiada podobne ograniczenia. Jednak implementacja całkowicie oparta na SQL pozwala dużo ściślej zintegrować logikę ontologii z danymi relacyjnymi bez potrzeby przekształcania danych pomiędzy modelami. Jak już wcześniej wspomniano, odpowiednikami właściwości RDF mogą być kolumny schematów relacyjnych. Można więc tak zdefiniować widok dedykowany danej właściwości, aby znalazły się w nim dane pochodzące z odpowiednich kolumn tabel relacyjnych. Dzięki temu można włączyć w przetwarzanie ontologiczne dane w postaci relacyjnej bez zmiany ich formatu. Jako przykład rozważmy bazę danych o dwóch tabelach:

Osoba			
ID	Nazwa	Wiek	Płeć

Potomstwo	
IDrodzica	IDdziecka

Załóżmy, że ontologia dotyczy powiązań i zależności rodzinnych i występują w niej takie relacje jak maSyna, maCórke, maDziecko, maWiek, maPłeć i klasy Mężczyzna, Kobieta, Rodzic, Ojciec, Matka, OsobaPełnoletnia i Dziecko.

Zaczynając od relacji, można dla nich utworzyć widoki w następujący sposób:

```
create view maSyna AS SELECT o1.nazwa AS podmiot, o2.nazwa AS obiekt
FROM osoba o1, osoba o2, potomstwo p
WHERE o1.id = p.rodzic AND o2.id = p.idDziecka AND o2.Płeć = 1
```

```
create view maCórke AS SELECT o1.nazwa AS podmiot, o2.nazwa AS obiekt
FROM osoba o1, osoba o2, potomstwo p
WHERE o1.id = p.rodzic AND o2.id = p.idDziecka AND o2.Płeć = 0
```

```
create view maWiek AS SELECT Nazwa AS podmiot, Wiek AS obiekt FROM Osoba
create view maPłeć AS SELECT Nazwa AS podmiot, Wiek AS obiekt FROM Osoba
```

Powyższe widoki przypisują dla bytów właściwości na podstawie danych zawartych w tabelach relacyjnych. W ten sposób dane te zostają wciągnięte w proces wnioskowania ontologicznego

bez konieczności transformacji modelu na RDF. Definicja kolejnej właściwości, maDziecko, jest uogólnieniem już istniejących i zdefiniowanych poprzez widoki:

```
create view maDziecko AS SELECT * FROM maSyna UNION SELECT * FROM maCórkę
```

Widoki odpowiadające klasom również można tworzyć tak, aby włączać do nich dane z tabel relacyjnych.

```
CREATE VIEW Mężczyzna AS SELECT Nazwa FROM Osoba WHERE Płeć = 1
CREATE VIEW Kobieta AS SELECT Nazwa FROM Osoba WHERE Płeć = 0
CREATE VIEW Matka AS SELECT Nazwa FROM Kobieta WHERE Nazwa IN (SELECT podmiot
FROM maDziecko)
CREATE VIEW Ojciec AS SELECT Nazwa FROM Mężczyzna WHERE Nazwa IN (SELECT pod-
miot FROM maDziecko)
CREATE VIEW Rodzic AS SELECT Nazwa FROM Matka UNION SELECT Nazwa FROM Ojciec
CREATE VIEW Dziecko AS SELECT Nazwa FROM Osoba o, maWiek w WHERE o.Nazwa =
w.podmiot AND w.obiekt < 18
CREATE VIEW OsobaPełnoletnia AS SELECT Nazwa FROM Osoba o, maWiek w WHERE
o.Nazwa = w.podmiot AND w.obiekt >= 18
```

Przedstawione definicje widoków mogą być wygenerowane automatycznie. Wejściem generatora jest przetworzona przez system wnioskujący ontologia oraz konfiguracja odwzorowania kolumn relacyjnych w odpowiadające im właściwości ontologiczne. Na tej podstawie generowane są widoki, w których definicjach zaszyta zostaje hierarchia klas oraz warunki przynależności indywiduów do tych klas. Dzięki temu łatwe staje się odpytywanie bazy danych uwzględniające kategoryzację obiektów – aby zadać zapytanie o byty jakiejś klasy wystarczy sprawdzić ich obecność w widoku związanym z tą klasą. Dokonanym w przykładach uproszczeniem jest użycie nazw typu napis jako identyfikatorów – można zamiast nich stosować oczywiście dowolne inne typy danych.

4. Podsumowanie

Omówione dwie metody pozwalają wykorzystać dane ontologiczne do odpytywania systemów relacyjnych. Podstawowe różnice między nimi wynikają z metody zastosowanego wnioskowania. W przypadku Oracle SDO_RDF_MATCH jest to wnioskowanie wprzód – wszystkie informacje zostają wywnioskowane zanim może zostać zadane jakiegokolwiek zapytanie. Jest to w pewnych sytuacjach wydajne, ma jednak swoje wady, takie jak konieczność przebudowania indeksu po każdorazowej zmianie danych (ontologii). W przypadku metody opartej na widokach, można mówić o wnioskowaniu wstecz. Wejściem tego rodzaju wnioskowania jest również zapytanie, a polega ono na takim jego transformowaniu, żeby uwzględnione zostały wszystkie reguły przetwarzania. Takiej transformacji w tym przypadku dokonują definicje widoków. Zaletą tej metody jest odporność na jakiegokolwiek zmiany reguł ontologicznych i brak konieczności wyliczania czegośkolwiek przed zadaniem zapytań. W przypadku zmiany w ontologii wystarczy zmienić definicje odpowiednich widoków i system może działać bez większych opóźnień. Największą różnicą jest jednak poziom integracji z danymi relacyjnymi. Rozwiązanie Oracle pozwala w pewien specyficzny sposób zintegrować w zapytaniu SQL wynik przetwarzania ontologii, nie pozwala jednak przetwarzać ontologicznie danych, które są w postaci relacyjnej. Zaprezentowane alternatywne rozwiązanie zbudowane jest w całości o standardowe elementy SQL i dzięki temu można zatrzeć różnicę w przetwarzaniu pomiędzy danymi relacyjnymi a danymi RDF. Wynika to z faktu, że logika przetwarzania związana z ontologią jest sprowadzona na ten sam poziom, co wykonanie właściwego zapytania SQL. Minusem może być słabsza wydajność takiego podejścia, która cechuje wszystkie rozwiązania bazujące na wnioskowaniu wstecz w porównaniu do wnioskowania wprzód, jeśli nie liczyć czasu przetwarzania wstępnego. Jednak z całą pewnością w niektórych zastosowaniach zaprezentowane podejście może okazać się bardziej adekwatne niż użycie SDO_RDF_MATCH.

Praca ta została sfinansowana ze środków na naukę w latach 2006-2009 jako projekt badawczy rozwojowy „Narzędzie wspomagające procedury śledcze wykorzystujące automatyczne wnioskowanie” oraz przez grant Politechniki Poznańskiej 45-083/07/BS.

Bibliografia

- [Pier04] Pierra G.: The PLIB ontology-based approach to data integration, World Computer Congress 2004, Tuluza
- [BrGu04] Brickley D., Guha R.V.: RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, <http://www.w3.org/TR/rdf-schema/>, 2004
- [PHH04] Patel-Schneider P., Hayes P., Horrocks I.: OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation, <http://www.w3.org/TR/owl-absyn/>, 2004
- [KICa04] Klyne G., Carroll J.: Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, <http://www.w3.org/TR/rdf-concepts/>, 2004
- [DPB07] Dehainsala H., Pierra G., Bellatreche L.: OntoDB: An Ontology-Based Database for Data Intensive Applications, Proc. of Database Systems for Advanced Applications (DAS-FAA'2007), Bangkok
- [PrSe07] Prud'hommeaux E., Seaborne A.: SPARQL Query Language for RDF, W3C Candidate Recommendation, <http://www.w3.org/TR/rdf-sparql-query/>, 2007