

XIV Konferencja PLOUG  
Szczyrk  
Październik 2008

# Nowe technologie semantyczne w Oracle 11gR1 i ich interfejs programistyczny

Maciej Falkowski  
Centrum Doskonałości w dziedzinie Telematyki,  
Instytut Automatyki i Inżynierii Informatycznej,  
Politechnika Poznańska

*e-mail: maciej.falkowski@put.poznan.pl*

**Abstrakt.** Artykuł omawia nowości z zakresu semantyki danych i przetwarzania danych RDF (Resource Description Framework) wprowadzone w wersji 11g bazy danych Oracle. Do nowości tych należą głównie rozszerzone możliwości wnioskowania, realizowane przy pomocy samej bazy danych nowymi zestawami reguł. Bardzo ciekawym udogodnieniem jest też dostarczenie programistycznego API dla języka Java, w którym zaimplementowano popularne interfejsy do danych semantycznych, znane z narzędzia Jena – Jena Graph oraz Jena Model. Dzięki temu możliwe jest łatwe zintegrowanie źródła danych (bazy Oracle) z zaawansowanymi systemami wnioskującymi, takimi jak Pellet czy RacerPro. Takie połączenie daje duże możliwości rozwoju aplikacji wymagających zaawansowanych operacji na danych oraz integracji danych z różnych źródeł. W artykule przedstawiono przykłady elementów takiej aplikacji oraz wyniki pomiarów wydajności wnioskowania i zapytań przeprowadzone w różnych konfiguracjach bazy danych.

**Informacja o autorze.** Mgr inż. Maciej Falkowski jest pracownikiem Instytutu Automatyki i Inżynierii Informatycznej Politechniki Poznańskiej. Jego zainteresowania zawodowe dotyczą semantycznej reprezentacji danych, ontologii oraz ich związków z bazami danych.



## 1. Dane ukierunkowane semantycznie

W ostatniej dekadzie miał miejsce gwałtowny rozwój informatyzacji między innymi w dziedzinach składowania i przesyłania danych. Setki razy wzrosły pojemności dysków twardych (z kilku do kilkuset gigabajtów) a także przepływność łączy internetowych (z kilkudziesięciu kilobitów do kilku megabitów). Postęp w tej dziedzinie jest jednak nie tylko ilościowy, ale również jakościowy. Dzięki rozwojowi Internetu oraz technologii sieciowych systemy informatyczne z wyizolowanych stały się szeroko dostępne, często są częścią większej całości i współpracują z innymi systemami. Zupełnie nowym zjawiskiem jest też ogólnodostępny Internet, który można uznać za wielką, rozproszoną bazę danych. Współczesna technika daje jednak nie tylko nowe możliwości, stawia również nowe wymagania. Jednym z nich jest sprawne składowanie, przetwarzanie i przesyłanie dużych ilości danych cyfrowych. W dziedzinie dostępu do danych jednym z kluczowych zagadnień koniecznych do rozwiązania jest wyspecyfikowanie semantyki (znaczenia) danych. Techniki wyszukiwania i dostępu do danych, które sprawdzały się w niedużych, lokalnych systemach nie są wystarczające w obecnej rzeczywistości. W związku z tymi problemami powstał projekt Semantic Web [BHL01], którego celem jest opracowanie technik pozwalających maszynom na sprawne, automatyczne przetwarzanie danych z uwzględnieniem ich semantyki. Potencjalnych zastosowań jest bardzo wiele. W kontekście Internetu pozwoliłoby to na ogromne usprawnienie wyszukiwania informacji. Język HTML działa wyłącznie w warstwie prezentacji. Używając obecnych technik, bazujących na słowach kluczowych, nie jest na przykład możliwe uzyskanie informacji o piłkarzach zadawszy zapytanie o sportowców. Systemy nie znają zależności, że piłkarz to sportowiec, nie znają semantyki wyrażen. Z kolei opis semantyczny wprowadzony do Web Serwisów pozwoliłby agentom na automatyczną kompozycję łańcucha usług w celu zrealizowania jakiegoś zadania, czy na przykład wyszukiwanie alternatywnego usługodawcy w przypadku awarii obecnego. W dziedzinie baz danych problem semantyki pojawia się podczas odpytywania oraz integrowania danych z różnych źródeł. Przykładem może być tutaj kolumna Data w tabeli produktów spożywczych – w jednej bazie danych może ona oznaczać datę produkcji, w innej natomiast datę przydatności do spożycia. Podobnie jak język HTML również schematy relacyjne nie dają możliwości formalnego i bezpośredniego wyrażania semantyki danych.

Do rozwiązania tych problemów mogą się przyczynić technologie semantycznego oznaczania danych, powstające w projekcie Semantic Web, takie jak model danych RDF [KlCa04] czy języki wyrażania ontologii RDFS [BrGu04] i OWL [PHH04].

RDF (ang. Resource Description Framework) to prosty model danych opracowany w projekcie Semantic Web z myślą o wyrażaniu metadanych związanych z semantyką, jednak przyjął się także w innych dziedzinach. Najmniejszą, atomową daną jest w nim trójka  $\langle S, P, O \rangle$ , gdzie S oznacza podmiot, na temat którego wyrażane jest stwierdzenie, P oznacza predykat (właściwość, relację), natomiast O oznacza obiekt relacji (wartość właściwości). Przykładową trójkę pokazano na rysunku 1.



Rysunek 1. Przykład trójki RDF

Tak zapisaną informację można rozumieć jako binarną relację, w której *podmiot* posiada *właściwość obiekt* (lub występuje w *relacji z obiekt*). Dla przykładowej trójki może to być Jan Kowalski ma numer telefonu +48123456789.

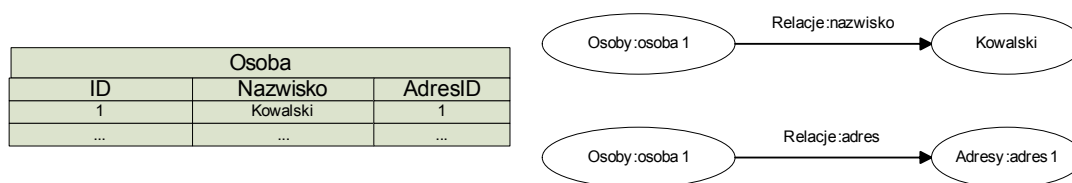
Zarówno podmioty stwierżeń, ich relacje, jak i obiekty są zgodnie z RDF identyfikowane poprzez URI. Każdy podmiot i relacja musi mieć nazwę będącą URI, wyjątkiem jest obiekt (ostatni element trójki RDF), który może mieć postać albo URI, albo literału. Literały służą do wyrażania

np. wartości liczbowych czy napisów. Zapis w postaci URI pozwala na łatwe zidentyfikowanie pochodzenia danego obiektu lub predykatu (poprzez nazwę serwera) i uzyskanie informacji na temat jego znaczenia, co jest bardzo istotne z punktu widzenia interoperacyjności. Dla skrócenia zapisu i łatwiejszego używania URI mogą być skracane za pomocą prefiksów dla przestrzeni nazw, na przykład dla trójki z rysunku 1:

```
xmlns:ludzie="http://people.com",
xmlns:relacje="http://relacje.pl",
<ludzie:JanKowalski, relacje:maNrTelefonu, +48123456789>.
```

URI użyte w podmiocie lub obiekcie może być kojarzone z identyfikatorami tabel relacyjnych. Połączenie podmiotu i obiektu relacją odpowiada więc zależności klucza obcego modelu relacyjnego. Identyfikatory w modelu relacyjnym są lokalne dla tabel - identyfikacja bytu (krotki) w modelu relacyjnym wymaga podania tabeli oraz wartości identyfikatora. W RDF natomiast identyfikatory są globalne, a każdy byt posiada swoje unikalne URI.

Model danych RDF w swojej strukturze (trójki) jest bardzo prosty. Wynikającą z tego zaletą jest jego wysoka interoperacyjność w porównaniu z modelem relacyjnym. W tym drugim dane są nierozzerwalnie związane ze schematem, natomiast w RDF schemat jest zawsze trójkowy. Dzięki temu można wymieniać same dane, bez konieczności uzgadniania ich schematu. Inną przyczyną prostoty modelu RDF jest jego przeznaczenie do wyrażania informacji z bardzo różnorodnych źródeł, gdzie często zdarzać się mogą informacje niepełne lub niespójne. W takim środowisku niepraktyczne byłoby wyrażanie informacji za pomocą rozbudowanych struktur, jak tabele relacyjne. Gdyby użyć w tym celu modelu relacyjnego wartości wielu kolumn mogłyby być nieznanne i musiałyby być wypełniane NULLami. Przykład tych samych danych wyrażonych relacyjnie oraz w RDF przedstawia rysunek 2.



Rysunek 2. Przykład danych wyrażonych relacyjnie i poprzez RDF

Koncepcję powiązania danych RDF z relacyjnymi zaproponował Tim Berners-Lee [BHL01]: tabele relacyjne można utożsamić z klasami, pojedynczą krotkę tabeli z informacjami o konkretnym bycie, a klucz główny tabeli może posłużyć do zbudowania identyfikatora dla bytu. Nazwy kolumn odpowiadają nazwom relacji RDF, natomiast komórki krotki to wartości tych relacji. W przykładzie z rysunku 2 wartości krotki z tabeli o trzech kolumnach zostały wyrażone za pomocą dwóch trójek. Pierwsza trójka łączy byt *Osoby:osoba1* relacją *nazwisko* z wartością „Kowalski”. Druga trójka łączy ten sam podmiot *Osoby:osoba1* relacją *adres* z bytem o identyfikatorze *Adresy:adres1*. Nazwa identyfikatora w obydwu przypadkach powstała z połączenia nazwy tabeli i wartości jej klucza głównego, odpowiednio tabela *Osoba* i tabela *Adres*, nie uwzględniona na rysunku, do której odnosi się klucz obcy *AdresID*. Używanie nazw relacji do wyrażania informacji jest bardziej elastyczne niż tabele i kolumny. Relacje są samodzielnymi bytami i funkcjonują niezależnie (a nie jak kolumny, które są częścią tabel), dzięki temu wprowadzenie nowego typu informacji w RDF polega na użyciu w znanej strukturze (trójce) nowej nazwy relacji. W modelu relacyjnym ta sama czynność wymaga dodania nowej kolumny lub nowej tabeli do systemu.

Relacje pełnią kluczową rolę w wyrażaniu informacji RDF – tą samą rolę co w modelu relacyjnym schemat bazy danych. Oderwanie nazw relacji od schematu pozwala na łatwe ich gromadzenie i publikowanie. W tym zakresie ogromne znaczenie mają ontologie, które mogą być uważane za rozwinięte słowniki pojęć i klas. Dzięki nim łatwe stało się budowanie hierarchii pojęć (klas i relacji) opartych na zawieraniu się (subsumcji), a także definiowanie dodatkowych zależności między pojęciami i nakładanie na nie więzów. Najprostszym językiem definiowania ontologii

związanym z RDF jest RDFS (ang. Resource Description Framework Schema). Wprowadza on między innymi pojęcie klas i pozwala na przeprowadzanie prostego wnioskowania, na przykład o tym, że piłkarz to sportowiec. Na bazie RDFS zbudowano rodzinę języków OWL (ang. Web Ontology Language) o zróżnicowanej sile wyrazu – najprostszy OWL Lite, oparty o pewną logikę deskrypcyjną OWL DL oraz najsilniejszy, jednak nierozstrzygalny OWL Full. Języki te funkcjonują obecnie jako standardowe i ogromna większość powstających ontologii bazuje właśnie na nich. W kontekście przetwarzania i integracji danych ontologie stanowią bardzo dobre narzędzie do specyfikowania semantyki (dzięki predykatom takim jak `rdf:Description`, służącym do opisywania znaczeń pojęć) w standardowy i spójny sposób, a ponadto dzięki używaniu URI są przystosowane do współdzielenia pomiędzy wieloma użytkownikami.

W tabeli 1 zebrano podstawowe cechy odróżniające model danych RDF od modelu relacyjnego.

Tabela 1. Cechy modelu relacyjnego i modelu RDF

Model relacyjny	Model RDF
Rozbudowana struktura tabel	Prosta struktura trójek
Dane wyrażane za pomocą krotek grupujących wiele informacji, konieczność wprowadzenia od razu całej krotki	Dane wyrażane za pomocą trójek, wyrażających pojedynczą informację
Krotki, utożsamiane z bytami, identyfikowane za pomocą nazwy tabeli oraz wartości identyfikatora	Identyfikator bytu w postaci URI, jednoznaczny i unikalny w skali całego systemu
Kolumny, służące do wyrażania danych, ściśle związane z tabelami.	Relacje i klasy służące do wyrażania danych są bytami niezależnymi, pogrupowane w hierarchie i słowniki (ontologie), można je stosować dowolnie.
Rozszerzalność modelu o nowe dane wymaga zmiany schematu i może być utrudniona	Bardzo łatwa rozszerzalność o nowe rodzaje danych poprzez dodanie nowych relacji do słownika
Wysoka wydajność wynikająca z przyjętych ograniczeń co do modelu danych dziedziny (np. że osoba może mieć jedno imię)	Niższa wydajność, niewielkie ograniczenia modelowanej dziedziny (można wprowadzać dowolnie wiele dowolnych relacji).
Dobrze sprawdza się w zastosowaniach hermetycznych, w obrębie jednej organizacji	Zaprojektowany z myślą o interoperacyjności, danych rozproszonych i pochodzących z różnych, niezależnych źródeł. Ontologia jako narzędzie integracji.

## 2. Dane semantyczne w Oracle

Oracle jest pierwszym dużym producentem, który wprowadził do swojego produktu obsługę danych RDF i technologii z nim związanych. W roku 2006 r. w wersji 10g bazy danych pojawiły się nowe typy danych odpowiadające trójkom RDF. Typy te to `SDO_RDF_TRIPLE_S` oraz `SDO_RDF_TRIPLE`. Pierwszy z nich jest używany do przechowywania danych, drugi natomiast do ich wyświetlania. Oprócz samych typów danych wprowadzono również możliwość wnioskowania za pomocą reguł ontologii RDF(S), a także funkcję tablicową `SDO_RDF_MATCH` (w wersji 11g zmieniono nazwę na `SEM_MATCH`). Funkcja ta pozwala na odpytywanie danych RDF zawartych w grupach (zwanym modelami), z wykorzystaniem wcześniej przeprowadzonego wnioskowania i wykorzystywanie odpowiedzi w zwykłym zapytaniu SQL. Dzięki takiemu roz-

wiązaniu aplikacje pracujące z danymi relacyjnymi mogą być łatwo poszerzane o możliwości, jakie daje RDF i wnioskowanie. Przykład zapytania wykorzystującego funkcję SEM\_MATCH:

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

Zapytanie to zaczerpnięto z pracy [FaJe07], gdzie przedstawiono również z większymi szczegółami możliwości zastosowań funkcji semantycznych bazy Oracle. Największą nowością w wersji 11g w stosunku do poprzedniej jest udostępnienie interfejsu programistycznego języka Java do funkcji semantycznych Oracle. Zostało to zrealizowane za pomocą implementacji dobrze znanych interfejsów pakietu Jena2 [WSKR03] – Jena Graph oraz Jena Model. Interfejs Graph służy do operacji na zbiorach trójek i jest zaimplementowany jako klasa GraphOracleSem, natomiast interfejs Jena Model rozszerza graf o interpretację ontologiczną. Pozwala on między innymi na odnajdywanie relacji i klas nadrzędnych czy zadawanie zapytań z uwzględnieniem wnioskowania. Jest zaimplementowany jako klasa ModelOracleSem i umiejscowiony, podobnie jak GraphOracleSem w przestrzeni oracle.spatial.rdf.client.jena. Aby z nich skorzystać należy przeprowadzić aktualizację bazy o komponent „Jena Oracle Adaptor”, dostępną dla baz w wersji 10g oraz 11g.

W dalszej części pokazane zostaną wybrane elementy funkcjonalne bazy danych związane z RDF i jej adaptera dla Jena obrazowane przykładami. Za przypadek testowy posłuży aplikacja umożliwiająca śledzenie przepływów dokumentów, towarów i usług pomiędzy różnymi podmiotami. Dla uproszczenia jedynym rodzajem danych będą dane o fakturach. Na przykładach pokazany zostanie proces ładowania danych, definiowania reguł przetwarzania, wnioskowanie prowadzące do powstania nowych danych oraz odpytywanie zgromadzonej wiedzy.

Podstawową klasą, za pomocą której obiekty programistyczne komunikują się z bazą danych jest klasa Oracle. Odpowiada ona za połączenie z instancją bazy danych, uwierzytelnienie itp. Każdy graf, a zatem i model posiada w bazie danych odpowiadające mu tabele, w których wprowadzane do tego modelu trójki są przechowywane. Jeśli podczas konstrukcji nowego modelu te tabele nie zostaną wskazane, to oprogramowanie automatycznie je utworzy:

```
Oracle oracle = new Oracle(DB_URL, USER, PASS);
ModelOracleSem model = ModelOracleSem.createOracleSemModel(oracle, modelName);
```

Do utworzonego modelu można załadować dane z plików. API umożliwia trzy metody dodawania danych (inkrementalną, batch oraz bulk), a obsługiwane formaty to między innymi RDF/XML oraz NTRIPLE [GrBe04]. W rozpatrywanym przypadku dane dotyczą faktur i są zapisane zgodnie z ontologią [CyMa]. Wycinek schematu ontologii dotyczący faktury przedstawiono na rysunku 3. Szarym tłem zaznaczono atrybuty, które mają odzwierciedlenie w rzeczywistych atrybutach faktury (jak jej numer, towar, kwota itp.). Pozostałe atrybuty są związane z dodatkowymi informacjami, które mogą być w pewnych przypadkach przydatne w projektowanym systemie.



```

INSERT INTO mdsys.semr_faktury VALUES(
  'odebralaFV_rule',
  '(?_t1 :klasyfikuje ?x) (?_t2 :maJakoBeneficjenta ?_t1) (?_t3
:jestZreifikowanaCzynnoscia ?_t2) (?_t4 :jestKlasyfikowanyPrzez ?_t3) (?_t5
:jestWynikiem ?_t4) (?y :jestKlasyfikowanyPrzez ?_t5)',
  NULL,
  '(?x :odebralaFV ?y)',
  SEM_ALIASES(SEM_ALIAS('',' http://mica.ai-
kari.put.poznan.pl/~jcyb/cDnSPL/TOML.owl# ')));

INSERT INTO mdsys.semr_faktury VALUES(
  'wystawilaFV_rule',
  '(?_t1 :klasyfikuje ?x) (?_t2 :jestCelemDzialania ?_t1) (?_t3
:jestZreifikowanaCzynnoscia ?_t2) (?_t4 :jestKlasyfikowanyPrzez ?_t3) (?_t5
:jestWynikiem ?_t4) (?y :jestKlasyfikowanyPrzez ?_t5)',
  NULL,
  '(?x :wystawilaFV ?y)',
  SEM_ALIASES(SEM_ALIAS('',' http://mica.ai-
kari.put.poznan.pl/~jcyb/cDnSPL/TOML.owl# ')));

```

Zdefiniowane w ten sposób zbiory reguł można aplikować do danych w celu wywnioskowania nowych faktów. Oracle stosuje model wnioskowania w przód, który polega na tym, że dla danego zbioru danych i zbioru reguł wywnioskowane i zapamiętywane są wszystkie możliwe nowe fakty. Dla każdego zestawu danych i reguł istnieje w systemie tzw. indeks regułowy, który przechowuje wywnioskowane trójki. Indeks ten jest uzupełnieniem modelu bazowego, wykorzystywanym podczas odpytywania. Można go utworzyć poleceniem:

```

execute SEM_APIS.CREATE_RULES_INDEX('faktury_idx', SEM_Models('m_fakt100'),
SEM_Rulebases('faktury'));

```

Proces tworzenia indeksu regułowego może być czasochłonny. Przeprowadziliśmy testy tworzenia indeksu z zaprezentowanym zbiorem reguł oraz zestawami danych o różnej wielkości. Komputer testowy posiada procesor Intel T2400 oraz 3 GB pamięci RAM i pracuje pod kontrolą systemu Windows XP. Wyniki przedstawia tabela 2.

Tabela 2. Czas budowania indeksów regułowych

Nr zestawu danych	Liczba faktur	Liczba trójek	Liczba wywnioskowanych faktów	Średni czas wnioskowania [s]
1	100	3700	400	2
2	200	7300	800	3
3	300	10900	1200	5
4	500	18100	2000	30
5	1000	36100	4000	99
6	2000	72100	8000	346

Czas potrzebny na zbudowanie indeksu silnie rośnie wraz ze wzrostem wielkości danych i już przy dwóch tysiącach faktur i prostym zestawie reguł system wymaga ponad pięciu minut. Taką procedurę trzeba wykonać jednokrotnie, dopóki nie zostaną zmienione dane lub reguły i konieczna będzie przebudowa.

Reguły posłużyły do wprowadzenia nowych relacji, których można używać w zapytaniach do modelu. Po zbudowaniu indeksu zapytania uwzględniające reguły można zadawać w języku SQL, posługując się funkcją tablicową SEM\_MATCH, lub poprzez obiekty Java w natywnym dla RDF języku zapytań SPARQL [PrSe07]. Zapytanie SPARQL składa się z trzech głównych części: bloku SELECT, w którym deklarowane są poszukiwane zmienne, bloku WHERE, który zawiera poszukiwany wzorzec, oraz bloku FILTER, w którym mogą być nakładane ograniczenia na zmienne występujące we wzorcu. Sposób definiowania zapytania w funkcji SEM\_MATCH jest bardzo podobny. W projektowanej aplikacji jednym z zapytań jest zapytanie o łańcuch przepływu faktur pomiędzy firmami, gdzie faktury dotyczą tego samego towaru i są wystawiane w porządku chronologicznym. Przykład w wersji SPARQL (pierwszy) oraz SQL z SEM\_MATCH (drugi) pokazano poniżej:

```
PREFIX a: <http://mica.ai-kari.put.poznan.pl/~jcyb/cDnSPL/TOML.owl#>
SELECT ?fv1 WHERE {
  <http://example.com#firma1> a:wystawilaFV ?fv1.
  ?fir2 a:odebralaFV ?fv1.
  ?fir2 a:wystawilaFV ?fv2.
  ?fir3 a:odebralaFV ?fv2.
  ?fv1 a:dotyczyTowaru ?t1.
  ?fv2 a:dotyczyTowaru ?t1.
  ?fir3 a:wystawilaFV ?fv3.
  ?fv3 a:dotyczyTowaru ?t1.
  ?fv1 a:wystawionoDnia ?d1.
  ?fv2 a:wystawionoDnia ?d2.
  ?fv3 a:wystawionoDnia ?d3.
  FILTER (?d1 < ?d2 && ?d2 < ?d3)}

SELECT fir2, fir3
FROM TABLE(SEM_MATCH(
'(<http://example.com#firma1> :wystawilaFV ?fv1)
(?fir2 :odebralaFV ?fv1)
(?fir2 :wystawilaFV ?fv2)
(?fir3 :odebralaFV ?fv2)
(?fir3 :wystawilaFV ?fv3)
(?fv1 :dotyczyTowaru ?t1)
(?fv2 :dotyczyTowaru ?t1)
(?fv1 :wystawionoDnia ?d1)
(?fv2 :wystawionoDnia ?d2)
(?fv3 :wystawionoDnia ?d3)',
SEM_Models('m_fakt100'),
SEM_Rulebases('faktury'),
SEM_ALIASES(SEM_ALIAS('', 'http://mica.ai-
kari.put.poznan.pl/~jcyb/cDnSPL/TOML.owl#')),
'(d1 < d2 and d2 < d3)'));
```

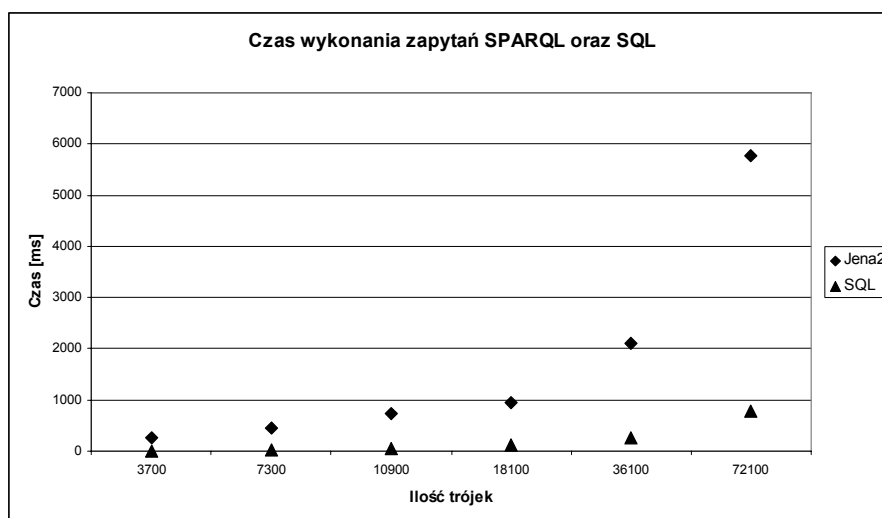
Zapytania różnią się jedynie sposobem zapisu, a w obydwu główną część stanowi trójkowy opis poszukiwanego wzorca. Zaprezentowany przykład dotyczy ciągu, w którym uczestniczą trzy firmy. Niestety, z powodu błędów w implementacji podsystemu zapytań RDF w Oracle rozmiar tekstu zapytania jest bardzo ograniczony i obecnie nie da się przy jego pomocy odpytywać o dłuższe łańcuchy powiązań ani poprzez zapytania SEM\_MATCH, ani poprzez interfejs do obiektów Jena2. Problem ten został już zauważony i ma być poprawiony w najbliższym czasie. Z tymi ograniczeniami przeprowadziliśmy testy porównawcze czasu wykonania zapytań w środowisku Java

(zapytanie SPARQL) oraz bezpośrednio poprzez zapytanie SQL z SEM\_MATCH. Testy przeprowadzono za pomocą tej samej maszyny, co poprzedni test. Czasy wykonania przedstawia tabela 3.

Tabela 3. Czasy wykonania zapytań w środowisku Jena oraz w SQL

Nr zestawu danych	Liczba odpowiedzi	Czas wykonania Jena2 [ms]	Czas wykonania SQL [ms]
1	6	250	10
2	11	460	30
3	30	734	50
4	78	954	110
5	348	2093	250
6	1417	5766	780

Wykonanie tych samych kwerend w środowisku SQL było za każdym razem kilkukrotnie szybsze, niż analogiczne zapytania w środowisku Jena2. Może to być związane z narzutem tworzenia obiektów Java oraz nie do końca dopracowaną implementacją adaptera. W przetestowanym zakresie wielkości danych czas odpowiedzi rósł wprost proporcjonalnie do wielkości danych objętych zapytaniem. Graficzne przedstawienie wyników przedstawia rysunek 4.



Rysunek 4. Wykres zależności czasu odpowiedzi systemów od wielkości danych

Wszystkie dotychczas omówione funkcjonalności dostępne poprzez interfejs programistyczny mają swoje bezpośrednie odpowiedniki w postaci poleceń SQL. Dwa najważniejsze elementy, które stanowią o przewadze możliwości interfejsu programistycznego Jena2 nad tradycyjnym SQL-owym to możliwość wykorzystania dostępnych dla Jena2 systemów wnioskujących oraz szersza możliwość manipulacji zapytaniami i ich wynikami niż za pomocą języka SQL.

Dodatkowe, zewnętrzne systemy wnioskujące, takie jak Pellet, pozwalają na używanie bardziej ekspresywnych ontologii niż wbudowane w Oracle RDF i RDFS, zazwyczaj jednak kosztem wydajności.

Ścisły styk zapytań i odpowiedzi z językiem programowania pozwala na większą ich kontrolę i możliwości manipulacji. Istnieją takie rodzaje zapytań, których nie da się prosto wyrazić w SQL czy SPARQL. W prezentowanym przykładzie aplikacji takim zapytaniem może być żądanie znalezienia łańcucha operacji wystawienia faktur między firmami o długości z danego przedziału.

Użytkownik za każdym razem może definiować inne zakresy interesujących go długości łańcucha, może też kształtować inne parametry mające wpływ na formę zapytania i w takim przypadku za każdym razem należy wygenerować je od nowa. Generator zapytań łatwo można zaimplementować w języku Java. Ponadto otrzymane wyniki można w dalszej części przetwarzać, np. wybierając te najbardziej interesujące bazując na prostych parametrach (np. o najwyższych kwotach itp.), lub też na skomplikowanych systemach wag biorących pod uwagę różne parametry. Dostęp do danych semantycznych z poziomu języka programowania daje w tym polu dużą elastyczność.

### 3. Podsumowanie

Semantyczne składowanie i przetwarzanie danych zdobywa coraz szerszą popularność i Oracle nie jest jedynym dużym producentem baz danych, który zaimplementował wsparcie dla RDF w swoim produkcie. Zespół z IBM opracował zestaw narzędzi [MWLC08] opartych na bazie danych DB2 w wersji 9.1, który umożliwia gromadzenie, wnioskowanie i odpytywanie danych RDF. Podobnie jak w przypadku Oracle, dane mogą być ładowane z plików XML oraz innych formatów. Inne jest w tych narzędziach podejście do składowanych danych. W przypadku produktu Oracle wszystkie dane, zarówno dotyczące konkretnych asercji (np. omawiane w artykule dane o fakturach) jak i ich metadane (ontologie, jak zaprezentowany na rysunku 3 fragment) są traktowane jednakowo i zapisywane w tych samych strukturach. IBM opracował natomiast bardziej złożone metody składowania. Dane ontologiczne posiadają swoje własne schematy, pozwalające na łatwe ich wyodrębnienie i przetwarzanie, także dla samych asercji zaprojektowano osobne schematy w zależności od ich rodzaju. Potencjalnie pozwala to na bardziej zoptymalizowane zadowanie zapytań.

Pokazane funkcje wykorzystania potencjału w explicite wyrażonej semantyce stanowią jedynie fragment ich możliwych zastosowań. Ontologie i format RDF bardzo dobrze nadają się do integracji danych z wielu źródeł, na skalę znacznie większą niż jest to możliwe w obecnych systemach bazodanowych, a wraz z rozwojem i upowszechnianiem się ontologii ten rodzaj przetwarzania będzie jeszcze zyskiwał na popularności i możliwościach. Ciągłe jeszcze trzeba płacić cenę wydajności za rozszerzone możliwości nowych modeli danych, jednak istnieje na tym polu duża aktywność, a włączenie się w ten trend największych laboratoriów i producentów baz danych z pewnością korzystnie wpłynie na tempo i zakres wprowadzania coraz wydajniejszych rozwiązań.

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 45-083/08/DS.

#### Bibliografia:

- [BHL01] Berners-Lee T., Hendler J., Lassila O.: The Semantic Web, Scientific American 5/2001, ISSN 0036-8733
- [KICa04] Klyne G., Carroll J.: Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, <http://www.w3.org/TR/rdf-concepts/>, 2004
- [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
- [PrSe07] Prud'hommeaux E., Seaborne A.: SPARQL Query Language for RDF, W3C Candidate Recommendation, <http://www.w3.org/TR/rdf-sparql-query/>, 2007

- [FaJe07] Falkowski M., Jędrzejek C.: Budowanie schematów relacyjnych dla danych OWL z wykorzystaniem silników wnioskujących, XIII Konferencja użytkowników i deweloperów ORACLE, PLOUG 2007, Kościelisko, 2007
- [WSKR03] Wilkinson K., Sayers C., Kuno H., Reynolds D.: Efficient RDF Storage and Retrieval in Jena2, HP Laboratories Technical Report HPL-2003-266
- [GrBe04] Grant J., Beckett D.: RDF Test Cases, W3C Recommendation, <http://www.w3.org/TR/rdf-testcases/#ntriples>, 2004
- [CyMa] Cybulka J., Martinek J., Police Investigations Management System Based on the Workflow Technology, Jurix 2008, wysłane do publikacji
- [MWLC08] Ma L., Wang C., Lu J., Cao F., Pan Y., Yu Y.: Effective and Efficient Semantic Web Data Management over DB2, SIGMOD 2008.