

# Biblioteka SDL – narzędzie semantycznego przetwarzania danych relacyjnych

Jarosław Bąk, Czesław Jędrzejek  
Centrum Doskonałości w dziedzinie Telematyki, Instytut Automatyki i Inżynierii Informatycznej

*e-mail: jaroslaw.bak@put.poznan.pl, czeslaw.jedrzejek@put.poznan.pl*

**Abstrakt.** W pracy przedstawiono opis i zastosowanie biblioteki programistycznej SDL (ang. Semantic Data Library). Narzędzie umożliwia integrację relacyjnych baz danych, ontologii w formacie OWL wraz z regułami SWRL oraz silnika wnioskującego Jess. Na półautomatyczną integrację wspomnianych elementów, składa się sposób odwzorowania schematu relacyjnej bazy danych na pojęcia ontologiczne oraz sposób generacji skryptów w języku Jess Language będącego wejściem silnika wnioskującego. Poprzez połączenie metadanych semantycznych w postaci ontologii wraz z danymi pochodzącymi z relacyjnej bazy danych uzyskuje się dane z explicite zdefiniowaną semantyką. Utworzony w ten sposób model danych semantycznych umożliwia zadawanie zapytań do bazy danych o relacje, które nie są bezpośrednio w niej odzwierciedlone. Odpowiedzi na zapytania definiowane w oparciu o pojęcia ontologiczne realizowane są poprzez wykorzystanie procesu automatycznego wnioskowania. Zaprezentowany został opis metody wnioskowania hybrydowego realizowanej przy użyciu silnika wnioskującego Jess, na którą składają się mechanizmy wnioskowania wstecz oraz w przód. Metoda wnioskowania bazuje na podziale pojęć i reguł na dwie grupy: pojęcia i reguły bazodanowe odpowiedzialne za interakcję z relacyjną bazą danych oraz pojęcia i reguły ontologiczne umożliwiające automatycznie wnioskowanie i udzielanie odpowiedzi na zadawane zapytania. Proces wnioskowania i zadawania zapytań zarządzany jest przez funkcje biblioteki SDL. Posiada ona możliwość zadawania złożonych zapytań składających się z wielu pojęć ontologicznych, które mogą zawierać szereg ograniczeń nakładanych na zmienne występujące w zapytaniach. Językiem zapytań jest język Jess Language. Zaprezentowany został także przykład zastosowania biblioteki SDL korzystającej z ontologii powstającej w ramach projektu Polskiej Platformy Bezpieczeństwa Wewnętrznego, która zawiera definicje pojęć rdzennych i reguł dotyczących procedury przestępczego i procedury jego wykrywania. Przedstawione zostało również porównanie narzędzia SDL z jego poprzednimi wersjami. Poruszone zostały zagadnienia do-tyczące wydajności działania narzędzia SDL w porównaniu z efektami działania narzędzi: silnika wnioskującego KAON2 oraz języka SQL.

**Informacja o autorach.** 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 roku 1994 związał się Francusko-Polską Wyższą Szkołą Nowych Technik Informatyczno-Komunikacyjnych (EFP) w Poznaniu. W latach 1999-2004 zajmował stanowisko Wiceprezesa Zarządu firmy ITTI w Poznaniu. Jest autorem lub współautorem ponad 100 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. Jarosław Bąk jest asystentem w Instytucie Automatyki i Inżynierii Informatycznej Politechniki Poznańskiej. W kręgu jego zainteresowań znajduje się szereg zagadnień związanych z semantyką danych, technologiami semantycznymi oraz przetwarzaniem metadanych i wnioskowaniem.



## 1. Wprowadzenie

Rozwój sieci semantycznych systemów informatycznych oraz zwiększanie się wymagań użytkowników doprowadziły do wniosków, z których jednoznacznie wynika, iż dane bez dodatkowych informacji je opisujących nie są już wystarczające, aby mogły być przetwarzane w sposób inteligentny przez maszyny. Informacje pozyskiwane z relacyjnych baz danych muszą zawierać dodatkowe metadane służące procesom przetwarzania, analizowania i wnioskowania. Dołączenie do danych relacyjnych warstwy konceptualnej (ontologii) umożliwia utworzenie bazy wiedzy, której zakres obejmuje wiedzę ekspertów w danej dziedzinie. Utworzony w ten sposób model danych wymaga powstania odpowiednich narzędzi umożliwiających efektywną realizację zapytań i wnioskowania.

### 1.1. Cel pracy

Celem pracy jest opracowanie metody reprezentowania i przetwarzania danych sieci semantycznej umożliwiającej efektywną realizację zapytań wykorzystujących wnioskowanie. Niezależnie od dominacji komercyjnej baz relacyjnych, istnieje coraz szersza klasa zastosowań (wyszukiwanie w sieci, przetwarzanie tekstu, dane strumieniowe lub naukowe), dla których logiczna reprezentacja RDF jest wygodniejsza [AMMH], [SMAHHH]. Tworzona biblioteka SDL (*ang. Semantic Data Library*) umożliwia dołączanie informacji semantycznej w postaci ontologii do danych relacyjnych. Utworzony w ten sposób model danych semantycznych pozwala na zadawanie zapytań do relacyjnej bazy danych o relacje niebędące bezpośrednio w niej odzwierciedlone. Odpowiedzi na zapytania realizowane są na bazie pojęć i reguł ontologicznych biorących udział w procesie automatycznego wnioskowania hybrydowego (na które składa się wnioskowanie w przód oraz wstecz). Biblioteka SDL umożliwia realizowanie zapytań do relacyjnych baz danych w języku Jess Language [Jess] oraz integrację:

- modelu konceptualnego wiedzy reprezentowanego w postaci ontologii,
- reguł przetwarzanych przez silnik wnioskujący,
- silnika wnioskującego,
- danych zawartych w relacyjnej bazie danych.

Potrzeba realizacji systemu wyniknęła w pracach przygotowawczych Systemu Analizatora Faktów i Związków [PPBW]. Dotychczas istniejące systemy (SweetRules [SwRu], OWLJessKB [OWLJK]) integrują lub próbują integrować tylko niektóre elementy spośród wyżej wymienionych. Obecnie istniejące systemy wnioskujące posiadające możliwość interakcji z bazą danych, wczytują całą zawartość bazy danych (partiami lub całościowo) do pamięci operacyjnej, co w znacznym stopniu spowalnia działanie mechanizmu wnioskującego. Stąd wynikła potrzeba zbudowania uniwersalnego narzędzia, łączącego zintegrowane funkcjonalności ontologii, wnioskowania oraz baz danych. Zastosowanie reguł wraz z ontologiami powinno umożliwić odkrywanie wiedzy, którą bardzo trudno uzyskać w przypadku samych zapytań SQL. Znacznie silniejszym pod względem wnioskowania w przypadku ogólnym, ale nie posiadającym pewnych funkcjonalności biblioteki SDL jest narzędzie KAON2 [KAON2] (w wersji profesjonalnej komercjalizowane przez firmę Ontoprise), którego działanie będzie stanowić porównanie w testach wydajnościowych.

Niniejszy artykuł stanowi kontynuację i rozwinięcie prac przedstawionych w [BaJe07, BaJe08].

### 1.2. Wykorzystywane narzędzia

Biblioteka SDL została zaimplementowana w języku Java, ze względu na jego powszechną dostępność oraz fakt, iż większość aplikacji semantycznych implementowanych jest właśnie w tym

języku. Dodatkowym wyznacznikiem był fakt, iż narzędzie Jess [Jess] zostało również zaimplementowane w języku Java, co pozwala na dostęp do funkcji języka Java z poziomu języka skryptowego Jess Language i odwrotnie. Silnik wnioskujący Jess (Java Expert System Shell) jest rozwijany w laboratoriach rządowych Sandia, w Stanach Zjednoczonych od 1995 roku. Biblioteka pozwala na wnioskowanie za pomocą algorytmu Rete [Rete]. Dzięki modyfikacjom w działaniu algorytmu narzędzie umożliwi wnioskowanie w przód oraz wstecz. Wnioskowanie regresywne wymaga jednak dokładnego wyspecyfikowania, które reguły i szablony (będące odpowiednikiem klas) w nim uczestniczą. Jess, jako silnik reguł, pozwala na konstruowanie oprogramowania, które posiada elementy sztucznej inteligencji, zapisane w faktach i regułach. Narzędzie to jest udostępniane nieodpłatnie na okres 2 lat w celach badawczych. W pracy wykorzystano najnowszą wersję narzędzia: 7.1. Biblioteka Jess znajduje zastosowanie w wielu systemach eksperckich oraz regułowych. Dzięki możliwości zastosowania wnioskowań wstecz i w przód, istnieje możliwość wykorzystania jej w praktycznie dowolnym systemie, wymagającym mechanizmów wnioskowania.

Językiem do definiowania ontologii obsługiwanych przez bibliotekę SDL jest język OWL (Web Ontology Language) [OWL]. Jest to najpowszechniej używany standard zapisu będący rekomendacją konsorcjum W3C. Język OWL można wzbogacać o reguły języka SWRL (Semantic Web Rule Language) [SWRL], które zwiększają siłę wyrazu ontologii. Edytorem ontologii, który wykorzystywany był w jej tworzeniu jest edytor Protégé [Protégé]. Wykorzystywany jest również interfejs API edytora wraz z narzędziem Jena2 [Jena2].

Serwer baz danych stanowi produkt firmy Microsoft – MS SQL Serwer 2005 (obsługiwane są również inne wersje tej rodziny serwerów). Połączenie z bazą danych realizowane jest poprzez sterowniki JDBC (Java Database Connectivity). Warto zaznaczyć, iż drobna modyfikacja fragmentów implementacji biblioteki SDL pozwoli na dopasowanie jej do innych serwerów relacyjnych baz danych, takich jak Oracle czy MySQL.

W ramach testów porównawczych i wydajnościowych wykorzystane zostało narzędzie KAON2 [KAON2]. Jest to jedyne znane nam narzędzie łączące możliwości ontologii, reguł i relacyjnych baz danych. Biblioteka ta realizuje zapytania zadawane w języku SPARQL [SPARQL]. KAON2 umożliwia wnioskowanie przy użyciu ontologii i reguł, jednak w przypadku użycia relacyjnej bazy danych wymagane jest ręczne utworzenie pliku XML, który odwzorowuje podstawowe pojęcia na kolumny tabeli relacyjnej (łączące klucz główny z inną kolumną). Plik ten odpowiada za instancje pojęć ontologicznych w bazie danych i jest odpowiednikiem ontologii OWL, jednak zapisanym w formacie języka XML.

Narzędzie KAON2 jest udostępniane nieodpłatnie w celach badawczych. Jest ono podstawą działania narzędzia OntoBroker OWL [OntBro] przeznaczonego do celów komercyjnych; jest ono dystrybuowane przez firmę Ontoprise [OntPri]. Dodatkiem do narzędzia OntoBroker OWL jest narzędzie OntoStudio umożliwiające semi-automatyczne mapowanie struktur danych oraz integrację danych z heterogenicznymi źródłami [OntStu].

## 2. Architektura oraz funkcje realizowane przez bibliotekę SDL

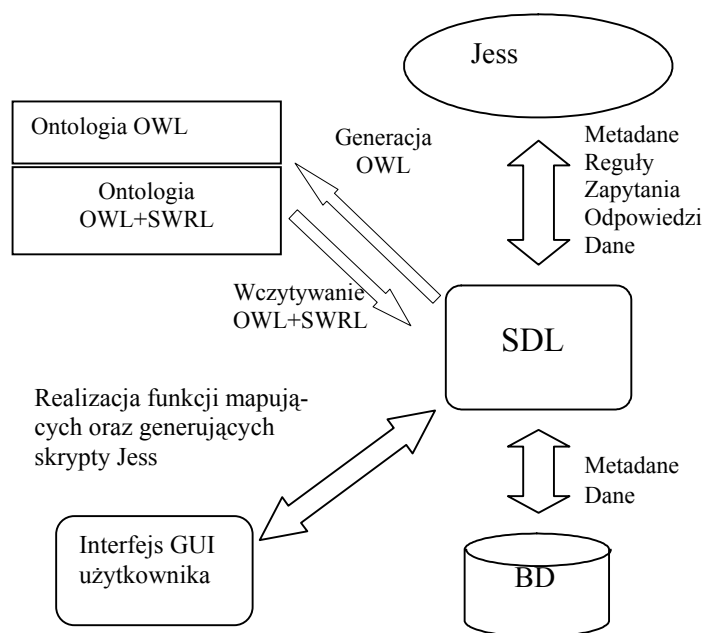
### 2.1. Architektura biblioteki SDL

Prezentowana architektura środowiska semantycznego przetwarzania danych składa się z następujących modułów:

- Jess – Java Expert System Shell,
- Ontologia OWL – ontologia zawierająca metadane semantyczne opisujące bazę danych, zapisana w języku OWL,

- Ontologia OWL+SWRL – ontologia wzbogacona nowymi pojęciami oraz regułami w języku SWRL,
- Interfejs GUI użytkownika – interfejs graficzny umożliwiający semi-automatyczne mapowanie obiektów ontologicznych na relacyjne oraz generację skryptów w języku Jess Language,
- BD – baza danych SQL Server 2005 (biblioteka obsługuje również inne wersje serwera).

Architektura systemu została przedstawiona na Rys. 1, na którym kierunek strzałek określa przepływ informacji pomiędzy modułami.



Rys. 1. Architektura narzędzia semantycznego przetwarzania danych - biblioteki SDL

## 2.2. Funkcje realizowane przez bibliotekę SDL

Przed biblioteką SDL postawionych zostało wiele wymagań dotyczących automatyzacji procesów transformacji oraz efektywnej realizacji złożonych zapytań. Do najważniejszych realizowanych funkcji należą:

- odczytywanie schematu relacyjnej bazy danych,
- generowanie ontologii OWL na podstawie schematu relacyjnej bazy danych,
- generowanie skryptu w języku Jess Language, zawierającego meta opis struktury bazy danych,
- przetwarzanie plików w formacie OWL wraz z regułami SWRL [SWRL],
- odwzorowywanie wszystkich konceptów ontologicznych, właściwości oraz reguł na skrypt w języku Jess Language,
- generowanie zapytań Jess na podstawie reguł SWRL,
- manipulowanie silnikiem Jess z poziomu języka Java,
- dodawanie danych z relacyjnej bazy danych, odpowiadających struktutom szablonów w języku Jess z poziomu języka Java,

- zadawanie zapytań do relacyjnej bazy danych,
- zadawanie złożonych zapytań do silnika reguł w języku Jess Language,
- zarządzanie procesem wnioskowania w przód,
- zarządzanie procesem wnioskowania wstecz,
- zarządzanie procesem wnioskowania hybrydowego,
- udostępniony interfejs graficzny ułatwiający korzystanie z niektórych funkcji biblioteki,
- udostępniony interfejs API biblioteki umożliwiający dostęp do większości funkcji (dużo większy zakres funkcjonalności niż w przypadku interfejsu graficznego).

Przedstawione powyżej funkcje składają się na proces integracji relacyjnej bazy danych, ontologii i silnika wnioskującego.

Obecna wersja biblioteki wykorzystuje mechanizm wnioskowania hybrydowego podczas procesu wykonywania zapytań (większość poprzednich wersji również wykorzystywało mechanizm wnioskowania hybrydowego - [BąJę07, BąJę08]).

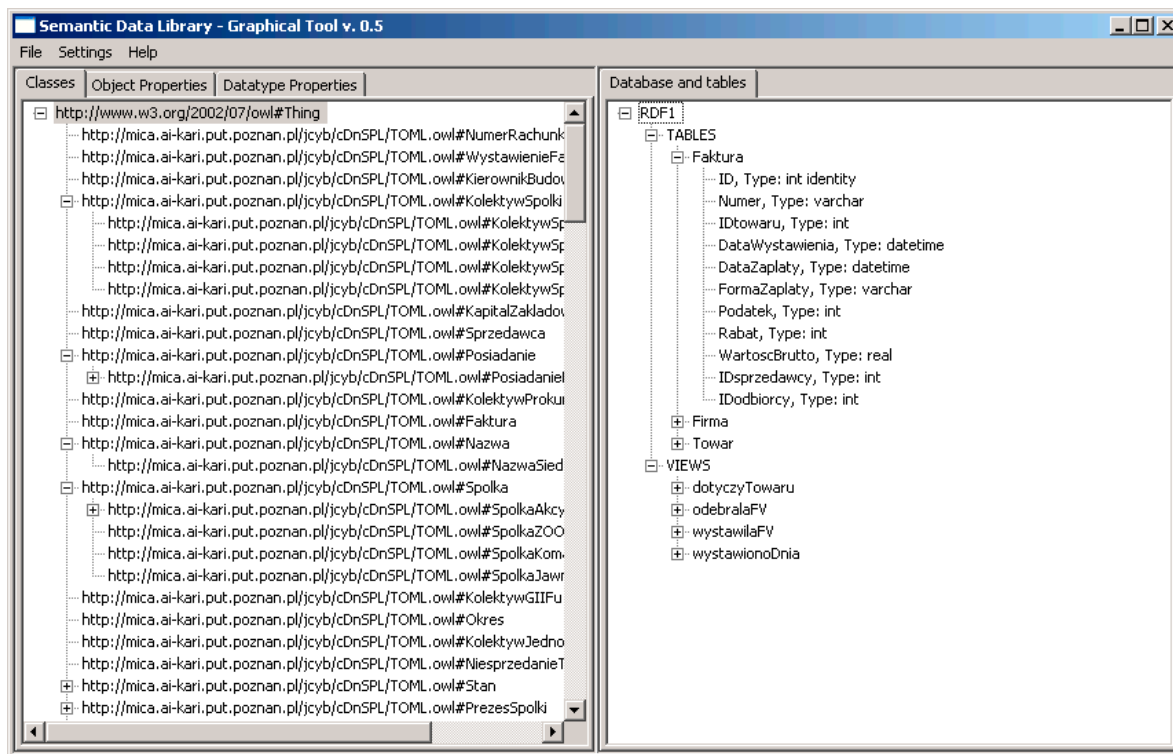
Biblioteka SDL posiada ograniczenia dotyczące możliwości wykorzystania ontologii. Obecna wersja wykorzystuje ontologię wyłącznie jako zhierarchizowaną siatkę pojęć i właściwości. Obsługiwane są standardy OWL i SWRL wraz z ograniczeniami nakładanymi na zmienne w języki SWRLB [SWRLB]. Przed rozpoczęciem procesu integracji warto przeprowadzić tzw. wnioskowanie w terminologii przy użyciu narzędzi do tego służących, np. Pellet [Pellet]. Za pomocą wnioskowania generowana jest bardziej zhierarchizowana postać ontologii, gdyż wzięte pod uwagę zostają wszystkie konstrukcje języka OWL oraz ograniczenia nakładane na właściwości i klasy. Dodatkowym atutem wnioskowania w terminologii jest sprawdzenie spójności oraz poprawności ontologii.

### 2.3. Funkcje realizowane z poziomu interfejsu graficznego

W celu łatwiejszego wykorzystania niektórych funkcjonalności biblioteki SDL zaimplementowany został interfejs graficzny użytkownika. Do funkcji wykonywanych za jego pomocą należą:

- wczytywanie pliku OWL i wyświetlenie hierarchii pojęć (funkcja *Open OWL File*),
- wyświetlanie schematu relacyjnej bazy danych w postaci drzewa zawierającego tabele, widoki i poszczególne kolumny (funkcja *Connect to database*),
- konfiguracja połączenia z relacyjną bazą danych (funkcja *Set Settings*),
- otwieranie pliku Jess przy pomocy notatnika systemowego (funkcja *Open Jess File*),
- mapowanie pojęć ontologicznych na kolumny tabel w relacyjnej bazie danych (funkcje *Create Direct SQL Query* oraz *Create Mapping*),
- zapis wykonanego mapowania do pliku w języku Jess (funkcja *Generate Jess Mapping File*),
- transformacja ontologii OWL na skrypt w języku Jess służący do wnioskowania w przód (funkcja *Transform OWL to Forward Jess File*),
- transformacja ontologii OWL na skrypt w języku Jess służący do wnioskowania wstecz (funkcja *Transform OWL to Backward Jess File*),
- łączenie skryptów: mapującego bazę danych na pojęcia ontologiczne oraz skryptu ontologii OWL w formacie języka Jess (funkcja *Merge Jess From OWL and Mapping file*).

Na Rys. 2. został przedstawiony interfejs graficzny biblioteki SDL. W lewym oknie znajdują się 3 zakładki reprezentujące odpowiednio hierarchie: klas, relacji binarnych pomiędzy elementami klas (ang. *Object Properties*) i relacji binarnych pomiędzy elementami klas a danymi (ang. *Datatype Properties*). W prawym oknie znajduje się schemat relacyjnej bazy danych wraz z tabelami i widokami, przy czym zawarte są tam również typy danych przechowywanych w kolumnach (nie są one jednak istotne w procesie transformacji).



Rys. 2. Interfejs graficzny biblioteki SDL

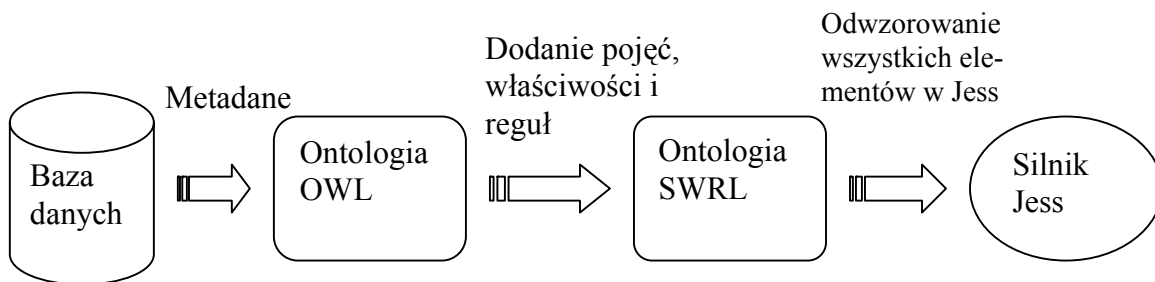
## 2.4. Proces integracji relacyjnej bazy danych, ontologii i silnika Jess

Proces integracji ma na celu przekształcenie powiązanych danych relacyjnych, ontologii OWL i reguł SWRL na format języka Jess Language. Możliwości biblioteki Jess sprawiły, że jest ona punktem końcowym całej integracji, w którym dokonywane jest wnioskowanie oraz uruchamiane są zapytania. Reprezentacja danych w jednym formacie pozwala na wykorzystanie w pełni możliwości systemu wnioskującego.

Proces integracji może być wykonywany dwojako. Pierwsza metoda została przedstawiona na Rys. 3. W tym przypadku proces integracji składa się z 3 etapów:

- etapu transformacji relacyjnej bazy danych na pojęcia i relacje w języku OWL; jest on wykonywany automatycznie; efektem tego etapu jest plik OWL,
- etapu tworzenia ontologii oraz mapowania obiektów ontologicznych na bazodanowe za pomocą reguł w języku SWRL; efektem jest plik OWL wraz z regułami SWRL,
- etapu transformacji ontologii OWL (z regułami SWRL) na skrypt w języku Jess Language.

Szerszy opis powyższej metody integracji można znaleźć w [BaJę07] oraz [BaJę08].



Rys. 3. Proces integracji realizowany przez bibliotekę SDL bez użycia interfejsu graficznego

Druga metoda integracji wykorzystuje funkcjonalność zawartą w interfejsie graficznym biblioteki SDL. Proces przebiega następująco:

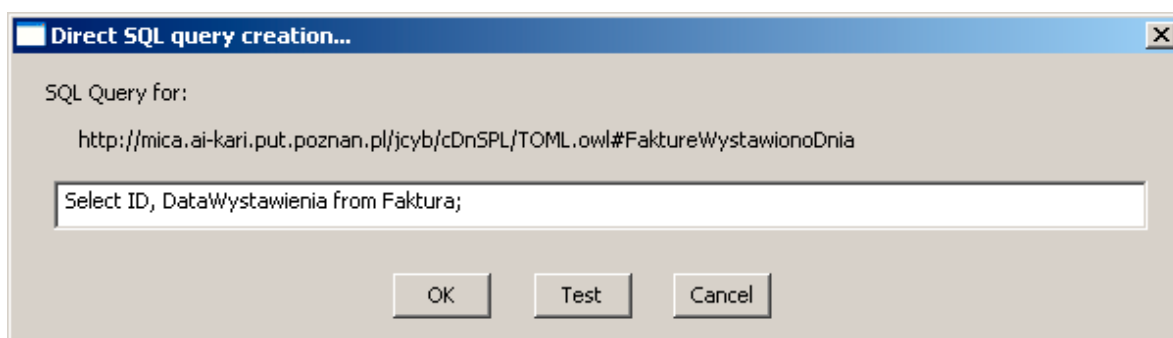
1. Utworzoną wcześniej ontologię wczytuje się za pomocą opcji *Open OWL File*. Efektem będzie wyświetlenie 3 drzew w 3 zakładkach (*Classes*, *Object Properties*, *Datatype Properties*).
2. Następnie za pomocą opcji *Connect to database* należy połączyć się z relacyjną bazą danych (ustawienia dotyczące połączenia z bazą można ustalić za pomocą opcji *Set Settings* - Rys. 4.).
3. Następnie za pomocą opcji *Create Direct SQL Query* tworzy się zapytanie do relacyjnej bazy danych, które mapuje się na pojęcia z ontologii. Przykładowo, jeśli chcemy zamapować pojęcie *FakturaWystawionoDnia* formułujemy zapytanie: *Select ID, DataWystawienia from Faktura*; i klikamy OK. (Rys. 5.). Istnieje możliwość przetestowania zapytania poprzez kliknięcie przycisku *Test*. Rezultatem jego działania jest liczba odpowiedzi na zapytanie. Istnieje możliwość zadawania bardziej złożonych zapytań zawierających klauzule *Where*, *And* i inne.
4. Kolejnym krokiem jest wygenerowanie skryptu reguł mapujących w języku Jess za pomocą funkcji *Generate Jess Mapping File*. Każda reguła mapująca zawiera odpowiednie zapytanie SQL. Wynikiem jest plik w języku Jess Language (odpowiada on wnioskowaniu wstecz).
5. Następnie należy wygenerować plik ontologii zapisanej w formacie języka Jess Language za pomocą funkcji: *Transform OWL to Backward Jess File* (skrypt wnioskowania wstecz).
6. Ostatnim krokiem jest połączenie plików: mapującego oraz ontologicznego (tylko skrypt służący do wnioskowania wstecz), za pomocą funkcji *Merge Jess From OWL and Mapping file*. Tak skonstruowany plik nadaje się do wczytania przez system wnioskujący biblioteki SDL.

W procesie integracji istotny jest fakt, iż podane zapytanie SQL służy do skonstruowania 4 reguł (w przypadku mapowania właściwości) lub 2 reguł (w przypadku mapowania klasy), z których każda odpowiedzialna jest za pobranie informacji z relacyjnej bazy danych. Przykładowo, dla przedstawionej wcześniej właściwości *FakturaWystawionoDnia* tworzone są następujące zapytania SQL:

- *Select ID, DataWystawienia from Faktura*; dla nieznanego ID i *DataWystawienia*,
- *Select ID from Faktura Where DataWystawienia = ?y*; dla nieznanego ID, ale znanej wartości w kolumnie *DataWystawienia*,
- *Select DataWystawienia from Faktura Where ID = ?x*; dla nieznanego wartości w kolumnie *DataWystawienia*, ale znanej wartości ID,
- *Select ID, DataWystawienia from Faktura Where ID = ?x AND DataWystawienia = ?y*; dla znanych wartości zarówno ID jak i *DataWystawienia*.



Rys. 4. Ustawienia dotyczące połączenia z relacyjną bazą danych



Rys. 5. Przykład mapowania właściwości FakturaWystawionoDnia na zapytanie SQL

Transformacja ontologii OWL na skrypt w języku Jess przebiega w następujący sposób:

- wszystkie pojęcia zamieniane są na szablony (ang. *template*) języka Jess, np. klasa Faktura oraz właściwość FakturaWystawionoDnia będą zamienione na szablony:

```
(deftemplate Faktura (slot name))
(deftemplate FakturaWystawionoDnia (slot domain) (slot range))
```

- reguły zapisane w języku SWRL zamieniane są na reguły języka Jess; np. reguła `FakturaWystawionoDnia(?x, ?d1), FakturaWystawionoDnia(?y, ?d2), swrlb:equal(?d2, ?d1) → WystawioneWTymSamymDniu(?x, ?y)`

zostanie zamieniona na:

```
(defrule WystawioneWTymSamymDniu
  (need-WystawioneWTymSamymDniu (domain ?x) (range ?y))
  (FakturaWystawionoDnia (domain ?x) (range ?d1))
  (FakturaWystawionoDnia (domain ?y) (range ?d2))
  (test (eq ?d1 ?d2))
=>
```

```
(assert (WystawioneWTymSamymDniu (domain ?x) (range ?y))))
```

- powyższa reguła podobnie jak w przypadku reguł mapujących zamieniona zostanie na 4 reguły w przypadku właściwości i 2 reguły w przypadku klasy,
- instancje pojęć zawartych w ontologii są pomijane.

### 3. Metoda zadawania zapytań

Zapytania obsługiwane przez narzędzie SDL zadawane są w języku Jess Language. Silnik wnioskujący Jess posiada własne mechanizmy zadawania zapytań, jednak w przypadku realizacji mechanizmu wnioskowania wstecz nie są one w stanie udzielić odpowiedzi na złożone zapytania (składające się z więcej niż jednego pojęcia). Fakt ten wynika ze sposobu działania silnika Jess w trybie wnioskowania wstecz (w trybie wnioskowania w przód nie występują tego typu ograniczenia).

Realizacja zapytania została podzielona na dwa silniki wnioskujące Jess, przy czym pierwszy z nich to silnik wnioskujący wstecz, który odpowiada za gromadzenie danych z bazy danych i przekazywanie ich na bieżąco do drugiego silnika, który stanowi silnik wnioskujący w przód, służący do udzielenia odpowiedzi na całe zapytanie. Oba skrypty uczestniczące w procesie wnioskowania, wstecz oraz w przód, mogą być automatycznie generowane z poziomu interfejsu graficznego lub programowo poprzez interfejs API biblioteki SDL.

Proces odpowiedzi na zapytanie składa się z następujących kroków:

1. Wykonaj zapytanie w silniku wnioskującym w przód i zwróć rezultat.
2. Jeżeli brak odpowiedzi na zapytanie:
  - a) Zlicz liczbę właściwości/klas występujących w zapytaniu; zapamiętaj ją jako liczbę uruchomień silnika wnioskującego wstecz –  $R$ .
  - b) W silniku wnioskującym wstecz wykonaj  $R$  razy:
    - Zadaj zapytanie szczątkowe (jedna właściwość/klasa) biorąc pod uwagę zależności z innymi zmiennymi oraz ograniczenia nakładane na zmienne – nastąpi aktywacja odpowiednich reguł.
    - Wykonaj wnioskowanie wstecz – pobrane zostaną dane z relacyjnej bazy danych.
    - Ponownie zadaj to samo zapytanie szczątkowe – wyniki przekaz do silnika wnioskującego w przód.
  - c) Wykonaj zapytanie w silniku wnioskującym w przód i zwróć rezultat.

Etap 1 oraz 2c to etapy wnioskowania w przód, natomiast etapy 2a i 2b to etapy wnioskowania wstecz. Zapytania formułowane są w języku Jess Language. Odpowiedzią na zapytanie jest zbiór wartościowań wszystkich zmiennych występujących w zapytaniu.

Rozdział kolejny prezentuje przykład zastosowania biblioteki SDL wraz z porównaniem wydajności narzędzi: KAON2 i SQL.

### 4. Przykład zastosowania biblioteki SDL

W celu zademonstrowania działania biblioteki SDL zostały utworzone relacyjne bazy danych dotyczące informacji o fakturach oraz wykorzystana została ontologia formułowana w ramach projektu Polskiej Platformy Bezpieczeństwa Wewnętrznego dotycząca konceptualizacji przedsiębiorstw gospodarczych [CyPPBW]. Ontologia ta zawiera ponad 300 pojęć, na które składają się klasy oraz właściwości. Należy dodać, że system nie działa na zasadzie przesiewania milionów dokumentów, ponieważ takie dane nie istnieją.

Istnieje wstępna informacja o przelewach z Głównego Inspektoratu Informacji Finansowej o transakcjach podejrzanych dostarczana przez banki na podstawie ustawy o przeciwdziałaniu wprowadzaniu do obrotu finansowego wartości majątkowych pochodzących z nielegalnych lub nieujawnionych źródeł z dnia 16 listopada 2000 r. Obecnie istnieje projekt nowelizacji ustawy o przeciwdziałaniu praniu pieniędzy oraz finansowaniu terroryzmu, mający na celu przede

wszystkim implementację Dyrektywy 2005/60/WE Parlamentu Europejskiego i Rady z dnia 26 października 2005 r. w sprawie przeciwdziałania korzystaniu z systemu finansowego w celu prania pieniędzy oraz finansowania terroryzmu (Dz. Urz. UE. L 309 z 25.11.2005 r.). Z podejrzanymi przelewami kojarzone są dokumenty, najczęściej faktury, a często też dokumenty odbioru prac oraz dokumenty potwierdzające faktyczne a nie fikcyjne wykonanie pracy.

Wygenerowanych zostało 5 relacyjnych baz danych zawierających informacje o:

- 100 fakturach,
- 200 fakturach,
- 300 fakturach,
- 500 fakturach,
- 1000 fakturach.

Każda baza danych zawiera 3 tabele oraz 4 niematerializowane widoki (Rys. 6 i 7). Bazy danych nie zawierają indeksów ani kluczy głównych. Wcześniejsze testy pokazują jednak, iż metody optymalizacji bazy danych działają pozytywnie nie tylko dla zapytań w języku SQL, ale również dla zapytań realizowanych przez narzędzia: SDL i KAON2.

Faktura			Towar		
	Column Name	Condensed Type		Column Name	Condensed Type
	ID	int		ID	int
	Numer	varchar(50)		Nazwa	nvarchar(150)
	IDtowaru	int			
	DataWystawienia	datetime			
	DataZapłaty	datetime			
	FormaZapłaty	varchar(150)			
	Podatek	int			
	Rabat	int			
	WartoscBrutto	real			
	IDSprzedawcy	int			
	IDodbiorcy	int			

Firma		
	Column Name	Condensed Type
	ID	int
	Nazwa	varchar(150)

Rys. 6. Tabele w relacyjnej bazie danych zawierającej informacje o fakturach

Widok dotyczyTowaru	Widok wystawionoDnia
Faktura, typ: varchar(50)	Faktura, typ: varchar(50)
Towar, typ: varchar(50)	Data, typ: datetime
Widok wystawilaFV	Widok odebralaFV
Firma, typ: varchar(50)	Firma, typ: varchar(50)
Faktura, typ: varchar(50)	Faktura, typ: varchar(50)

Rys. 7. Niematerializowane widoki w relacyjnej bazie danych zawierającej informacje o fakturach

Poszczególne widoki wiążą informacje:

- fakturę z produktem, którego dotyczy – widok dotyczyTowaru,
- fakturę z datą wystawienia – widok wystawionoDnia,
- firmę wystawiającą fakturę z numerem faktury – widok wystawilaFV,
- firmę, która odebrała fakturę, czyli była stroną kupującą – widok odebrałaFV.

Zostały utworzone skrypty mapujące pojęcia ontologiczne na relacje bazodanowe. Przy pomocy interfejsu graficznego zostały utworzone odpowiednie skrypty silnika wnioskującego: w przód oraz wstecz wraz z mapowaniem.

Dla silnika wnioskującego KAON2 została utworzona dodatkowa ontologia mapująca odpowiednie kolumny z tabel relacyjnych na pojęcia ontologiczne.

W celu sprawdzenia wydajności biblioteki SDL zostały sformułowane 3 ekwiwalentne zapytania w językach:

- Jess Language dla biblioteki SDL,
- SQL dla serwera MS SQL 2005,
- SPARQL dla silnika wnioskującego KAON2.

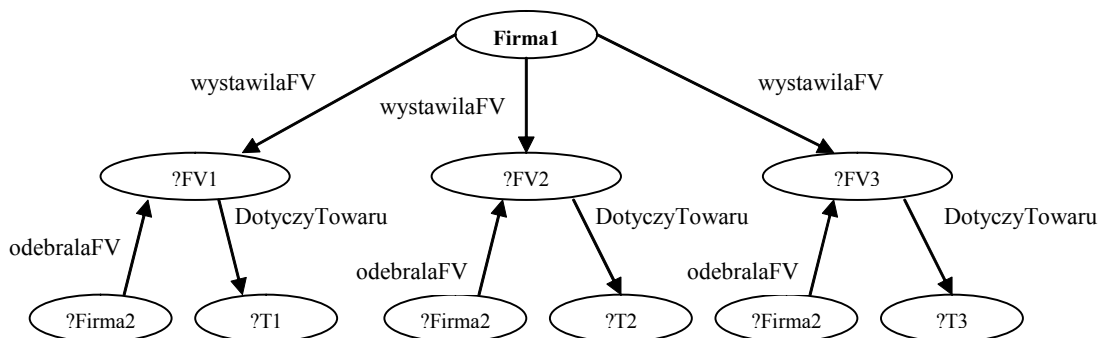
Pierwsze z zapytań dotyczy firm, którym firma *Firma1* wystawiała faktury na 3 różne towary. Znalezienie takich firm sugeruje, iż są one partnerami firmy *Firma1*. Drugie zapytanie dotyczy łańcucha faktur wystawionych pomiędzy 3 firmami na ten sam produkt, gdzie pierwszą firmą w łańcuchu jest *Firma1* a daty widniejące na fakturach muszą występować w porządku rosnącym. Taki łańcuch obrazuje sieć pośredników towaru. Ostatnie, trzecie zapytanie dotyczy zamkniętego łańcucha faktur na ten sam towar, gdzie odbierająca ostatnią fakturę firma (*Firma1*) jest również pierwszą w łańcuchu a daty muszą występować w porządku rosnącym. Zamknięty łańcuch faktur sugeruje, iż istnieje prawdopodobieństwo popełnienia przestępstwa karnoskarbowego – karuzeli VAT.

Zapytania w postaci grafowej zostały przedstawione na Rys. 8. Zmienne są poprzedzone znakiem „?”; nad strzałkami widnieją nazwy właściwości zaczerpnięte z ontologii. Każde zapytanie było wykonywane sześć razy, a uśrednione wyniki zostały zaprezentowane w Tabeli 1. Skrót PW – oznacza pierwsze wykonanie zapytania, natomiast średnia obliczana jest na podstawie kolejnych 5 uruchomień zapytania.

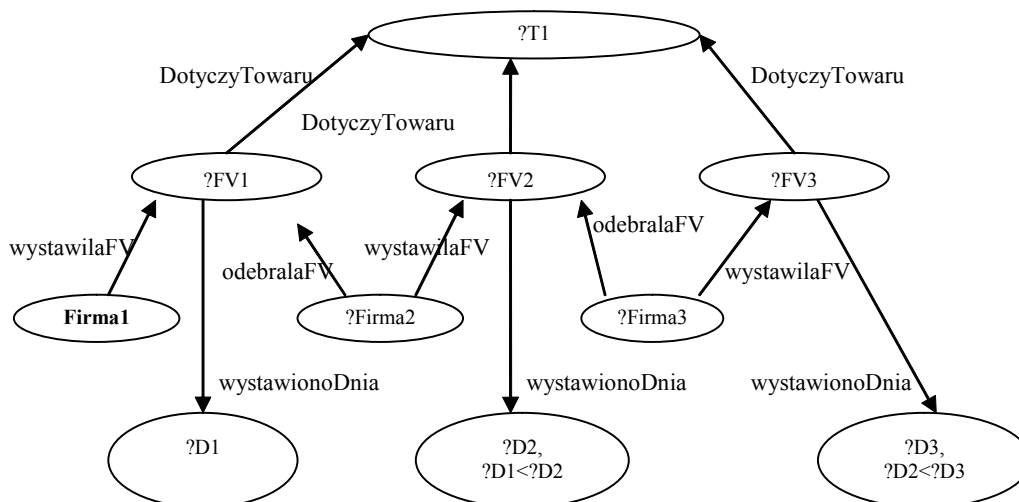
Testy wykonywane były na komputerze klasy PC o następującej konfiguracji:

- procesor Intel Core2 Duo 2GHz, 4MB Cache,
- pamięć RAM wielkości 2GB, 667 MHz,
- serwer relacyjnych baz danych Microsoft SQL Server 2005.

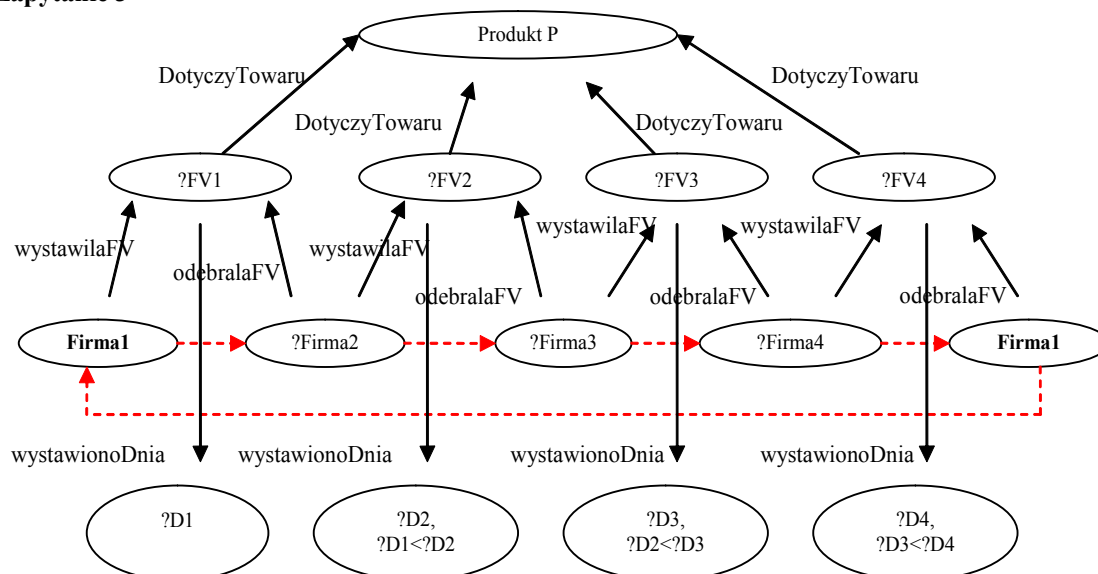
## Zapytanie 1



## Zapytanie 2



## Zapytanie 3



Rys. 8. Postać grafowa testowych zapytań

Tabela 1. Wydajność biblioteki SDL w porównaniu do narzędzi: SQL i KAON2

Liczba faktur w bazie	Nr pytania	SDL [ms]		SQL [ms]		KAON2 [ms]		Liczba wyników
		PW	Średnia	PW	Średnia	PW	Średnia	
100	1	578	<b>553</b>	1 032	<b>19</b>	6 125	<b>578</b>	0
	2	890	<b>853</b>	7 297	<b>63</b>	10 359	<b>550</b>	2
	3	1 031	<b>1 018,8</b>	7 328	<b>18,8</b>	7 844	<b>562,4</b>	0
200	1	704	<b>647</b>	3 453	<b>431,2</b>	4 093	<b>843,8</b>	24
	2	1485	<b>1 459,4</b>	7 656	<b>284,2</b>	11 046	<b>568,8</b>	10
	3	2078	<b>1 984,4</b>	7 500	<b>34,8</b>	8 218	<b>906,2</b>	1
300	1	937	<b>868,6</b>	4 797	<b>2 059,6</b>	4 985	<b>2 134,6</b>	120
	2	2703	<b>2 519</b>	8 610	<b>1 181,2</b>	12 204	<b>609,6</b>	30
	3	3593	<b>3 672</b>	7 703	<b>93,8</b>	9 656	<b>2 209,2</b>	4
500	1	1500	<b>1 431,4</b>	135 422	<b>133 018,2</b>	13 781	<b>10 990,6</b>	774
	2	5360	<b>5 265,6</b>	15 250	<b>7 471,6</b>	12 766	<b>680</b>	183
	3	10656	<b>10 162,4</b>	8 516	<b>828</b>	23 656	<b>15 903,2</b>	25
1000	1	6 906	<b>6 840,6</b>	29 831	<b>24 242</b>	199 125	<b>198 118,8</b>	8502
	2	20 703	<b>16 534,4</b>	9 391	<b>1 589,8</b>	13 610	<b>1640,4</b>	1683
	3	106 375	<b>106 306,4</b>	18 922	<b>6 899,8</b>	425 766	<b>421 922</b>	518

Z przeprowadzonych testów jednoznacznie wynika, iż większość zapytań realizowanych przez SDL wykonuje się szybciej w pierwszym wykonaniu. Wynika to z faktu, iż serwer baz danych otrzymuje proste zapytania, których nie musi w żaden sposób optymalizować. W przypadku zapytań SQL lub KAON2, zadawane zapytanie jest bardzo złożone, jednak jego wyniki i ostateczna forma jest przechowywana przez menadżera zapytań bazy danych, stąd późniejsze wykonania charakteryzują się znacznie krótszym czasem odpowiedzi. W niektórych przypadkach (trzecie zapytanie do bazy 1000 faktur) biblioteka SDL potrzebuje ponad 10 razy więcej czasu na udzielenie odpowiedzi niż w przypadku serwera baz danych. Trwają obecnie prace nad formalną przyczyną tego faktu oraz nad dokładnym sformalizowaniem metody zadawania zapytań przedstawionej w Rozdziale 3. Narzędzie KAON2, w niektórych przypadkach również znacznie odbiega od zapytań realizowanych przez SQL i SDL. W efekcie zauważalny jest fakt, iż wykonanie zapytania, które wykorzystuje dane z explicite zdefiniowaną semantyką, wymaga więcej czasu. W celu lepszego oszacowania wydajności biblioteki SDL należałoby sformułować większą liczbę zapytań o znanej (obliczonej) złożoności. W tym kierunku będą szły dalsze prace związane z narzędziem SDL.

## 5. Poprzednie wersje biblioteki SDL

Obecnie wykonana biblioteka SDL jest piątą wersją systemu wnioskującego posiadającego interakcję z relacyjną bazą danych. Pierwsza wersja zawierała wyłącznie możliwość wnioskowania w przód, co wiązało się z potrzebą wczytywania wszystkich danych do pamięci. W wersji drugiej narzędzie wprowadzono wnioskowanie wstecz oraz stworzono mechanizm wnioskowania hybrydowego. W wersji trzeciej zoptymalizowano mechanizm zadawania zapytań oraz metodę mapowania danych relacyjnych na pojęcia ontologiczne. W wersji czwartej próbowano wykorzystać wyłącznie wnioskowanie wstecz, jednak praktyka pokazała, iż zadawanie złożonych zapytań jest bardzo nieefektywne. Obecna implementacja znacznie poprawia wydajność całego systemu oraz

ponownie wprowadza mechanizm wnioskowania hybrydowego. Najważniejsze różnice pomiędzy poszczególnymi wersjami zostały przedstawione w Tabeli 2.

Tabela 2. Porównanie wersji biblioteki SDL

Wersja	1	2	3	4	5	
<b>Opis</b>	Wnioskowanie w przód, potrzeba wczytania całej bazy danych do pamięci roboczej (RAM). Krótki czas odpowiedzi na zapytanie (po długim czasie wnioskowania). Możliwość zadawania zapytań bez podania wartości początkowych, zarówno dla domain i range.	Wnioskowanie hybrydowe. Silnik wnioskujący wstecz: oddzielne reguły dla dziedziny (domain) i zakresu (range) pojęcia. Dodatkowa reguła dodająca instancje pojęcia do bazy wiedzy. Reguły zawierające pola z bazy danych są automatycznie zamieniane na zapytania SQL (reguły tworzące pojęcia wiążące kolumny). Silnik wnioskujący w przód: służy do przefiltrowania danych pod kątem uwzględnienia ograniczeń nałożonych na zmienne. Wczytywane są całe krotki. Zadawane zapytania muszą posiadać wartość dla domain lub range. Szybsze działanie niż w wersji 1.	Wnioskowanie hybrydowe. Jedna reguła definiująca każde pojęcie. Reguły zawierające pola z bazy danych są automatycznie zamieniane na zapytania SQL (reguły tworzące pojęcia wiążące kolumny z wartością z kolumny) i łączone w jedną regułę (w przypadku takiej samej głowy reguły). Silnik wnioskujący wstecz dostarcza wyłącznie wymagane dane. Silnik wnioskujący w przód pozwala uzyskać odpowiedź na zapytanie. Zapytanie można zadawać wyłącznie z określoną wartością dla dziedziny pojęcia. Najlepsze wyniki czasowe porównywalne z zapytaniami SQL.	Wnioskowanie wstecz. Reguły zawierające pola z bazy danych są automatycznie zamieniane na zapytania SQL. Możliwość wnioskowania z hierarchii pojęć. Interfejs graficzny do obsługi podstawowych funkcji biblioteki. Obsługa tabel oraz widoków w relacyjnej bazie danych.	Wnioskowanie wstecz. Reguły zawierające pola z bazy danych są automatycznie zamieniane na zapytania SQL. Możliwość wnioskowania z hierarchii pojęć. Tworzone są 4 reguły na własność 1, 2 na klasę.	Wnioskowanie hybrydowe. Cztery reguły na własność 1, 2 na klasę. Możliwość wnioskowania z hierarchii pojęć. Interfejs graficzny do obsługi podstawowych funkcji biblioteki. Obsługa tabel oraz widoków w relacyjnej bazie danych.
<b>Zmiany</b>		Dodanie mechanizmu wnioskującego wstecz służącego wydobyciu danych z relacyjnej bazy danych. 3 reguły na pojęcie.	Zamiast 3 reguł jest jedna. Łączenie w jedną regułę zbioru reguł dającego ten sam efekt (np. dodanie tego samego faktu do bazy wiedzy). Pobieranie z bazy danych wyłącznie określonych wartości (nie całych krotek, tylko poszczególne pola: klucz główny-kolumna).	Pobieranie z bazy danych wyłącznie określonych wartości (par: klucz główny-kolumna). Niepotrzebne są już wartości początkowe, aby rozpocząć proces wnioskowania. Dodano możliwość wnioskowania z hierarchii pojęć.	Ułatwienie mapowanie pojęć poprzez interfejs graficzny. Najnowsza wersja biblioteki Jess (po- przednie wersje silnika nie są obsługiwane). Drobne zmiany w architekturze narzędzia.	
<b>Wady</b>	Bardzo niska wydajność w przypadku dużych baz danych. Silnik przeprowadza proces wnioskowania dla wszystkich reguł, co przedkłada się na długi czas jego realizacji.	Wymagana wartość początkowa zapytania (podanie wartości klucza głównego). Zbyt duża liczba reguł dodająca to samo pojęcie do bazy wiedzy (po 3 reguły na pojęcie). Zbyt duża liczba faktów dodawanych do bazy wiedzy, które mają negatywny wpływ na wydajność rozwiązywania.	Wymagana wartość początkowa zapytania wyłącznie dla domain (w przypadku range wymagałoby to zdefiniowania nowej reguły).	Niska efektywność odpowiedzi w przypadku zadawania złożonych zapytań.	Niezadawalająca wydajność w niektórych zapytaniach, pomimo iż wersja 5 jest 4 razy szybsza od wersji 4.	

## 6. Podsumowanie

W niniejszej pracy przedstawione zostało narzędzie semantycznego przetwarzania danych, biblioteka SDL. Biblioteka ta posiada szeroki zakres funkcji umożliwiających operacje na metadanych (ontologia OWL) oraz na danych semantycznych (danych wzbogaconych o informację semantyczną). Posiada ponadto interfejs graficzny, który w znaczny sposób ułatwia wykonywanie najważniejszych funkcji transformujących ontologię oraz mapujących jej pojęcia na relacje bazodanowe do postaci skryptu w języku Jess Language. Integracja relacyjnej bazy danych, ontologii OWL wraz z regułami SWRL umożliwia zadawanie zapytań w języku silnika wnioskującego Jess. Proces realizacji zapytań wykorzystuje mechanizmy wnioskowania wstecz oraz w przód, a całość procesu udzielania odpowiedzi jest zarządzana przez bibliotekę SDL.

Wnioskowanie hybrydowe realizowane w procesie realizacji zapytań ma za zadanie zwiększyć wydajność procesu wnioskowania silnika Jess wykorzystującego relacyjną bazę danych. Zastąpienie potrzeby wczytania wszystkich danych z bazy, trybem "na żądanie" (w procesie wnioskowania wstecz), pozwala na kilkunastokrotne zmniejszenie czasu potrzebnego na uzyskanie odpowiedzi. Zaletą takiego podejścia jest skalowalność. Wynika ona z faktu, iż nie jest wczytywana cała baza danych, a jedynie jej fragmenty istotne w procesie wykonywania zapytania.

Obecna implementacja biblioteki SDL oraz przeprowadzone testy wydajnościowe wskazują, że połączenie technologii relacyjnych baz danych z silnikiem wnioskującym przy użyciu ontologii tworzy nową jakość i dostarcza nam narzędzie, które pozwala na efektywne zadawanie bardzo złożonych zapytań do bazy danych. Wnioskowanie hybrydowe pozwala na usunięcie wad istniejących silników wnioskujących takich jak np. KAON2, których szybkość w wielu przypadkach jest niezadowolająca. Wykorzystywany w pracy Jess jest uznawany za jeden z najszybszych silników wnioskujących. Ze względu na wykorzystywany algorytm Rete [Rete] fakty poddawane procesowi wnioskowania muszą się znajdować w pamięci RAM. Powoduje to, że silnik ten może działać na relatywnie małej porcji danych. Zastosowanie wnioskowania hybrydowego usuwa wspomnianą niedogodność.

Kolejnym etapem prac nad wnioskowaniem hybrydowym będzie dalsza poprawa jego efektywności oraz sformalizowanie procesu udzielania odpowiedzi, co skutkować będzie opracowaniem bardziej wydajnych algorytmów wykonywania zapytań.

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

- [AMMH] Daniel Abadi, Adam Marcus, Samuel Madden, Katherine Hollenbach, [Scalable Semantic Web Data Management Using Vertical Partitioning](#), VLDB 2007, pp.411-422.
- [BąJę07] Bąk J., Jędrzejek C.: Wnioskowanie hybrydowe w relacyjnej bazie danych, wykorzystujące ontologię OWL wzbogaconą regułami języka SWRL, Materiały konferencyjne PLOUG2007, Kościelisko, październik, 2007, ISSN 1641-211, str. 285-300
- [BąJę08] Bąk J., Jędrzejek C., Wnioskowanie hybrydowe w relacyjnej bazie danych wykorzystujące podejście semantyczne, Konferencja BDAS'08 (Bazy Danych: Aplikacje i Systemy), Ustroń 27-30 maja 2008, str. 335-350
- [CyPPBW] Ontologia PPBW, <http://mica.ai-kari.put.poznan.pl/~jcyb/PPBW.owl>
- [Frie03] Friedman-Hill E., „Jess in Action”, 2003, Manning Publications Co.
- [Jena2] Biblioteka Jena2, <http://jena.sourceforge.net/>

- [Jess] Java Expert System Shell, <http://www.jessrules.com/>
- [KAON2] Narzędzie KAON2, <http://kaon2.semanticweb.org/>
- [MSS04] Motik B., Sattler U., Studer R., Query Answering for OWL-DL with Rules. Proc. of the 3rd International Semantic Web Conference (ISWC 2004), Hiroshima, Japan, November, 2004, pp. 549-563
- [OntBro] Narzędzie OntoBroker OWL, <http://www.ontoprise.de/de/en/home/products/ontobroker.html>
- [OntPr] Firma Ontoprise, <http://www.ontoprise.de/de/en/home.html>
- [OntStu] Narzędzie OntoStudio, <http://www.ontoprise.de/de/en/home/products/ontostudio.html>
- [OWL] Web Ontology Language (OWL) Guide Version 1.0, <http://www.w3.org/TR/owl-features/>
- [OWLJK] <http://edge.cs.drexel.edu/assemblies/software/owljesskb/>
- [Pellet] Narzędzie wnioskujące Pellet, <http://pellet.owlid.com/>
- [PPBW] Polska Platforma Bezpieczeństwa Wewnętrznego, „Narzędzie wspomagające procedury śledcze wykorzystujące automatyczne wnioskowanie”, <http://www.ppbw.pl>.
- [Protégé] Protégé Editor, <http://protege.stanford.edu/>
- [RDF] RDF, <http://www.w3.org/RDF/>
- [Rete] Algorytm Rete w narzędziu Jess, <http://herzberg.ca.sandia.gov/jess/docs/71/rete.html>
- [SMAHHH] Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos Nabil Hachem Pat Helland, The End of an Architectural Era (It's Time for a Complete Rewrite), VLDB 2007, pp: 1151
- [SPARQL] Język zapytań SPARQL, <http://www.w3.org/TR/rdf-sparql-query/>
- [SwRu] SweetRules, <http://sweetrules.projects.semwebcentral.org/>
- [SWRL] SWRL: A Semantic Web Rule Language Combining OWL and RuleML, <http://www.w3.org/Submission/SWRL/>
- [SWRLB] SWRLB, <http://www.w3.org/2003/11/swrlb>

