Opracowanie i implementacja podstaw optymalnego przechowywania i transferowania danych medycznych (ontologie)

Patryk Bąk, Marek Kendziorek, Wojciech Inglot

AGH Kraków, 2013

1 Abstrakt

Wprowadzenie: Systemy medyczne należą do jednych z najbardziej rozbudowanych systemów w branży IT. Cechują się one nie tylko ogromnymi ilosciami danych, które przetwarzają, ale również bardzo dużą złożonoscią budowych tych danych oraz relacji, które między nimi zachodzą. Własnie branża e-health jako jedna z pierwszych zainteresowała się zastosowaniem ontologii oraz ich reprezentacji w zagadnieniu reprezentowania danych. Niniejszy artykuł porusza problem zastosowania optymalnych technologii i metodologii do implementacji systemu opartego o ontologię z naciskiem na przechowywanie oraz transfer ontologii.

Rozwiązanie: Zaproponowane przez nas rozwiązanie problemu oparte zostało na języku programowania java oraz opartych na nim technologiach i narzędziach. Do transferu danych wykorzystano FUSEKI, które oferuje transfer danych protokołem REST z wykorzystaniem języka zapytań SPARQL. Dodatkowo przy definiowanu ontologii zaproponowano wykorzystanie opartego i interfejsy i stałe słownika danych.

Wnioski: Zaproponowana przez nas implementacja reprezentacji oraz transferu danych medycznych posiada wiele cech pożądanych przy projektowaniu tego typu systemów. Rozwiązanie to cechuje się bardzo dużą skalowalnoscią zarówno od strony wielkosci samego systemu jak i możliwosci rozbudowy przechowywanych w nim danych. Zastosowanie architektury typu SOA przy budowie systemu medyczneo pozwala na łatwą, modułową rozbudowę takiego systemu oraz ułatwioną integrację z innymi systemami medycznymi, zupełnie niezależnymi od siebie.

2 Wstep

2.1 Ontologia

Ontologia jest formalną reprezentacją pewnej dziedziny wiedzy. Składają się na nią zapis zbiorów pojęć i relacji między nimi. Zapis ten tworzy schemat pojęciowy, który jest opisem danej dziedziny wiedzy. Może służyć jednocześnie jako podstawa do wnioskowania o właściwości opisywanych ontologią pojęć.

Pod pojęciem ontologii mogą się kryć różne struktury wiedzy. Ich przeznaczenie czy zakres stosowania może być wieloraki.

Ontologie możemy podzielić ze względu na stopień ich formalizacji:

- nieformalne
  - predefiniowane słownictow
  - słowniki
  - tezarusy
  - taksonomie
- formalne
  - ontologie oparte na danych
  - ontologie oparte na logice

Podział następuje także ze względu na zakres stosowania:

- ontologie wysokiego poziomu (*upper ontologies*)
- ontologie dziedzinowe (*domain ontologies*)
- ontologie aplikacyjne

Istnieje szereg języków pozwalających na zapis lub wspierających ontologie:

- OWL (*Ontology Web Language*)
- RDF (*Resource Description Framework*)
- RDF Vocabulary Description Language (RDF Schema, RDFS)
- OCML (Operational Conceptual Modeling Language)
- XML (Extensible Markup Language)

2.2 OWL

OWL (Ontology Web Language) jest opartym na składni XML językiem służącym do opisu ontologii. Stanowi on rozszerzenie języka RDF (Resource Definition Language). Istnieją 3 odmiany OWL: OWL Lite, OWL DL oraz OWL Full. W 2004 roku język ten został uznany za standard przez organizację W3C.

OWL został stworzony do przetwarzania informacji o obiektach oraz relacji między nimi, nie do przechowywania tych informacji w formie czytelnej dla człowieka. OWL jest zbudowany na bazie RDF, tak więc języki te są ze sobą w pełni kompatybilne. OWL posiada jednak znacznie mocniejszą składnię.

Do elementów języka OWL należą takie atrybuty jak:

- Class - definiuje grupę indywidualnosci, które posiadają pewne wspólne cechy. Klasy można organizować w hierarchie za pomocą atrybutu subClassOf.
- rdf:Property - okresla pewne relacje między indywidualnosciami np. samochod może posiadać drzwi, osoba moze posiadać dziecko albo rodzica.
- rdfs:domain - ogranicza indywidualności, do których można zastosować relację (property). Jeżeli relacja łączy jedną indywidualność z drugą, i jednocześnie posiada ona domenę na określoną klasę, to obie indywidualności muszą należeć do tej klasy.
- rdfs:range - ogranicza nie tylko indywidualności, ale także wartość relacji (property)
- Individual - indywidualności są instancjami klas, które są połączone pewnymi relacjami (property).

Wszystkie powyższe atrybuty są częścią specyfikacji OWL Lite.

2.3 Protege

Protege jest otwarto-źródłowym narzędziem stworzonym przez Stanford Medical Informatics. Jak przy większości narzędzi modelujących, architektura Protege jest przejrzyście podzielona na dwie części - model i widok. Modele są wewnętrzną reprezentacją mechanizmy ontologii i baz wiedzy. Widoki zapewniają interfejs użytkownika, dzięki któremu można wyświetlać i manipulować modelami.

Protege posiada możliwość wczytania, edycji i zapisywania w wielu popularnych dla ontologii formatach, np:

- RDF
- OWL
- XML
- UML
- Bazy relacyjne

2.4 SPARQL

2.5 REST

REST jest protokołem komunikacji między aplikacjami, który tak naprawdę jest tylko pewną dodatkową warstwą nałożoną na protokół HTTP. Budowanie zapytań restowych z poziomu języków programowania jest tak samo proste, jak zrozumienie działania protokołu HTTP. Istnieje wiele bibliotek i prostych frameworków do budowania serwisów, odbierających i wysyłających zapytania HTTP. Należą do nich np Jersey-RS, czy zbudowana na jego bazie trochę większa biblioteka Apache CXF. Biblioteka Fuserki, będąca częścią specyfikacji Apache Jena oferuje możliwość transferu danych właśnie protokołem REST, przy użyciu języka zapytań SPAQRL.

Ze względu na swoją prostotę i oparcie o standard HTTP, wybór tego protokołu jako sposobu komunikacji między aplikacjami jest bardzo dobrą decyzją. Umożliwia to znakomitą skalowalność naszego systemu, poprzez rozdzielenie go na szereg indywidualnych aplikacji, które nie muszą być napisane przy użyciu tej samej technologii, muszą jedynie współdzielić tą metodę komunikacji oraz ewentualnie sposób serializacji danych.

3 Implementacja rozwiązania

Rozwiązanie zaprezentowane w niniejszym artykule zostało napisane w języku Java, z wykorzystaniem Ontology API biblioteki Apache Jena. Biblioteka ta pozwala na mapowanie różnego rodzaju reprezentacji ontologii na klasy Java. Do transferowania naszych danych wykorzystamy bibliotekę Apache CXF, a dokładniej jej moduł REST, który pozwoli nam na przesyłanie zserializowanych danych przez protokół HTTP.

3.1 Reprezentacja danych

Ontology API biblioteki Apache Jena pozwala między innymi na budowanie oraz importowanie modeli ontologii. Stworzenie nowego modelu ontologii oraz zaimportowanie do niego już istniejącej ontologii zapisanej w formacie OWL przedstawiono na rysunku.

3.2 Transfer danych

W projekcie uruchomiony został serwer HTTP Fuseki. Pozwala on na przechowywanie i udostepnienie danych wczytanych z plikow OWL poprzez zapytania HTTP. W jego wewnętrznej bazie danych budowane są grafu do których aplikacje klienckie odwołują się w zapytania SPARQL.

Instalacja Instalacja Jeny i Fuseki ogranicza się do dodania do zmiennych środowiskowych ścieżek do paczek.

```
export JENA=
    /Users/XXXX/Fuseki/apache-jena-2.11.0/
export FUSEKI=
    /Users/XXXX/Fuseki/jena-fuseki-1.0.0/
```

Uruchomienie serwera Standardowo serwer FUSEKI uruchamiany jest pod wirtualną domeną „localhost"i na porcie 3030.

Uruchomienie serwera FUSEKI:

```
./fuseki-server --update
    --loc=/Users/~/MyTDB/ /ds &
```

Załadowanie pliku z ontologią Załadowania pliku OWL (poprzez konsolową komendę):

```
./s-put
    http://localhost:3030/ds/data
    default
    /Users/XXXX/RIMV3OWL.owl
```

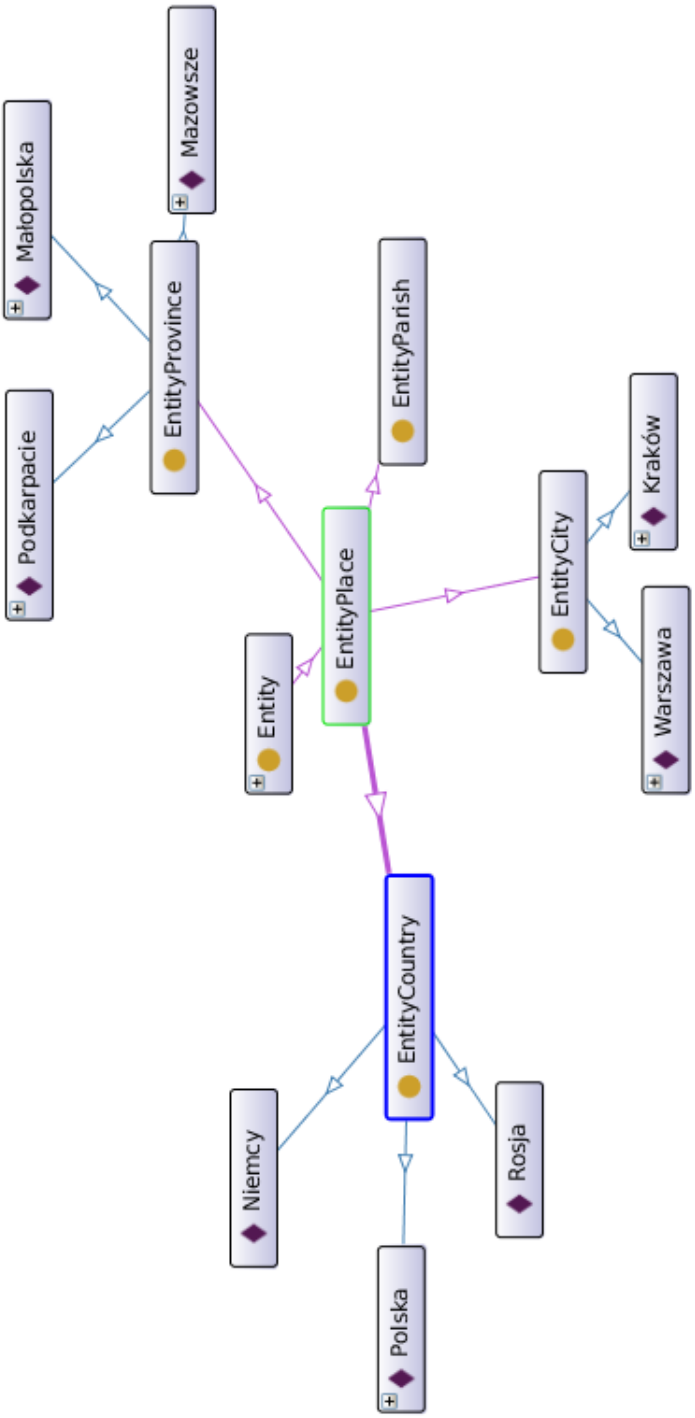
Tworzenie zapytania SPARQL Wysłanie zapytania SPARQL wypisującego „koncepty"z załadowanego grafu:

1. Tworzenie zapytania.

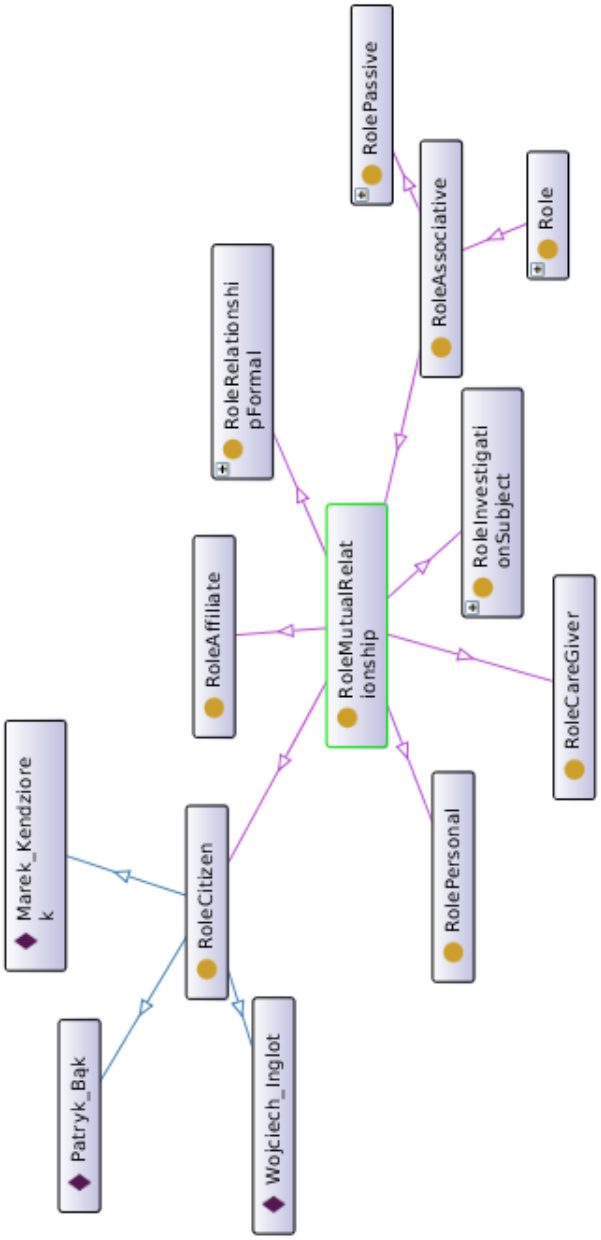
   ```
   select distinct ?Concept
       where {[] a ?Concept}
   ```

2. Tworzenie zapytania HTTP POST, gdzie:
   - jako URI podajemy sufix prowadzący do punktu końcowego (eng. endpoint) i doklajemy końcówkę /query
   - w parametrze POST/GET „query"podajemy stworzone zapytanie
   - opcjonalne parametry: output (np. „json"), default-graph-uri
3. W efekcie otrzymujemy zapytanie (dla punktu końcowego „ds":

```
http://localhost:3030/ds/sparql?
query=select+distinct++%3FConcept
+where+%7B%5B%5D+a+%3FConcept%7D
&default-graph-uri=
&output=json
```



Rysunek 1. Przykład Ontologii Opartej o HL7 (EntityPlace)



Rysunek 2. Przykład Ontologii Opartej o HL7 (Role)

4 Podsumowanie

Zaproponowana przez nas implementacja reprezentacji oraz transferu danych medycznych posiada wiele cech pożądanych przy projektowaniu tego typu systemów. Rozwiązanie to cechuje się bardzo dużą skalowalnoscią zarówno od strony wielkosci samego systemu jak i możliwosci rozbudowy przechowywanych w nim danych. Zastosowanie architektury typu SOA przy budowie systemu medyczneo pozwala na łatwą, modułową rozbudowę takiego systemu oraz ułatwioną integrację z innymi systemami medycznymi, zupełnie niezależnymi od siebie. Idąc dalej można skorzystać również z możliwosci oferowanych przez narzędzia jeszcze wyższych rzędów, takie jak szyny biznesowe, dzięki którym integracja mogłaby być jeszcze skuteczniejsza, nie naruszając jednoczesnie architektur oraz implementacji poszczególnych systemów.

W kwestii reprezentacji danych, wykorzystanie takiego frameworku jak Apache Jena pozwala na intuicyjne operowanie na dowolnej reprezentacji ontologii, ich rozbudowę w srodowisku programowania java, oraz ich graficzną reprezentację. Wykorzystanie tutaj w pracy srodowiska java jednoczesnie czyni znacznie prostszą integrację tego rozwiązania z opisanym wczesniej mechanizmem transferowania informacji, który również napisany jest w tym języku.

Acknowledgments

Authors would like to thank YYYYY.

Literatura

1. A. Einstein, On the movement of small particles suspended in stationary liquids required by the molecular-kinetic theory of heat, Annalen der Physik 17, pp. 549-560, 1905.