OSTIS-2013
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

ПРАКТИЧЕСКИЕ АСПЕКТЫ РЕАЛИЗАЦИИ ПОИСКОВЫХ ИНТЕРФЕЙСОВ СЕМАНТИЧЕСКИХ ПРИЛОЖЕНИЙ LINKED DATA

Галушка И.Н.[*], Завгородний В.В.**, Солошич С.Н.[*], Щербак С.С.[*]

[*] *Кременчугский национальный университет имени Михаила Остроградского, г. Кременчуг, Украина*

sergey.shcherbak@gmail.com

***Днепродзержинский государственный технический университет г. Днепродзержинск, Украина*

Рост популярности концепции связанных данных наряду с описанием информации в виде триплетов RDF обусловил необходимость исследования процедур взаимодействия с хранилищами триплетов. В результате разработан универсальный поисковый интерфейс, обеспечивающий визуальное построение запросов к хранилищам триплетов. Визуальные запросы автоматически преобразуются в язык запросов SPARQL, который используется для доступа к хранилищам триплетов. После выполнения запроса пользователь получает контекст с триплетами, соответствующими искомым с учетом заданных ограничений на предикаты и объекты.
Ключевые слова: поиск; связанные данные; шаблон реализации; триплет

ВВЕДЕНИЕ

Широкое развитие стандартов и поддержка концепции Linked Data крупными ИТ компаниями определяет тенденции развития будущего WWW как глобальной базы данных, в которой можно через специализированные поисковые интерфейсы получать доступ к структурированным представлениям данных, документов, распределенных по Web, для решения задач поиска и подобных.

В рамках Linked Data информация описывается в терминах языка RDF (англ. Resource Description Framework), а именно, в виде триплетов, троек вида «субъект-предикат-объект» или квадов (quad) – поименованных графов вида «граф-субъект-предикат-объект». Далее, если это не будет приводить к противоречию, будем использовать понятие «триплет» как для понятия «триплет», так и для понятия «квад».

Модель данных RDF предполагает распределенное хранение объектов и их схем, при их наличии, на различных web-серверах с интегрированным или внешним специализированным хранилищем триплетов (Triplestore). Для доступа к хранилищам триплетов используется протокол и язык запросов SPARQL (англ. SPARQL Protocol and RDF Query Language), который является в некотором смысле аналогом языка SQL, применяемого для обработки данных реляционных баз. Использование подобных языков запросов подразумевает наличие специализированных знаний для эффективного их применения, что не способствует ни популяризации языка SPARQL, как средства извлечения данных, ни увеличению распространенности хранилищ триплетов.

В данной работе предлагается подход к увеличению эффективности поиска на основе хранилищ триплетов путем уменьшения сложности процедур динамического формирования запросов на языке SPARQL и применения шаблонов запросов для генерации пользовательского поискового интерфейса.

1. Поисковые пользовательские интерфейсы к хранилищам триплетов

Поиск информации в рамках Linked Data основывается на использовании данных, представленных в виде объектов, принадлежащих к некоторой предметной области. В рамках такого подхода содержимое документа представляется как совокупность объектов, объединенных некоторым контекстом. Для определения контекста объектов будем использовать понятие «граф» квада. Тогда поиск информации сводится к идентификации по заданным критериям (пользовательским ограничениям) триплетов графа и выводу

контекстов с триплетами, соответствующих идентифицированным.

Для создания эффективных поисковых интерфейсов, «дружественных» к пользователю, необходима разработка типовых решений – паттернов реализаций, которые могли бы позволить визуально конструировать запросы к поименованным графам хранилища триплетов и не требовала бы от разработчика или пользователя программных систем специализированных знаний о языке SPARQL.

На примере веб-приложения рассмотрим архитектуру классического решения на основе хранилища триплетов и языка запросов SPARQL (рис.1).

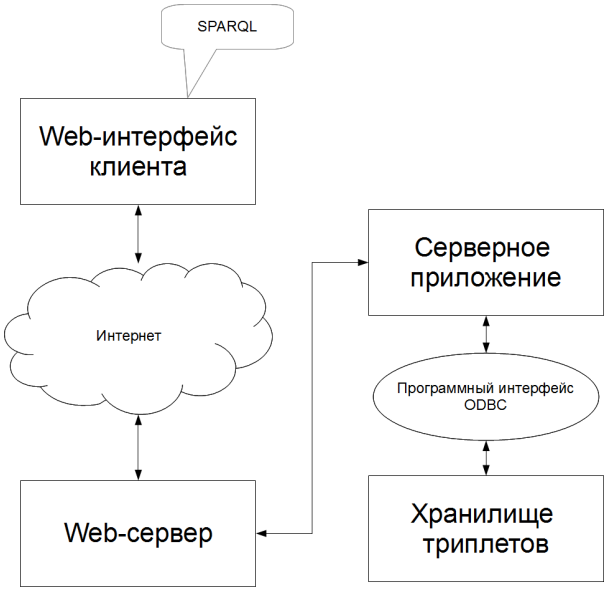

Рисунок 1 – Архитектура веб-приложения на основе хранилища триплетов

Запрос пользователя на языке SPARQL передается от web-приложения к хранилищу триплетов, где запрос выполняется, а результаты отсылаются обратно клиенту.

Ключевая особенность классического решения в необходимости формирования зачастую в ручную запроса на языке SPARQL, что требует не только специализированных знаний о языке SPARQL, но и знаний о структуре объектов, содержащихся в хранилище триплетов.

Одним из приложений, реализующих подобную функциональность является веб-интерфейс isql компании OpenLink, применяемый в Dbpedia, который наряду с универсальностью доступа к данным, отличается стабильностью работы.

Рассмотрим архитектуру предлагаемого решения, исключающего взаимодействие пользователя с хранилищем триплетов через SPARQL (рис.2).

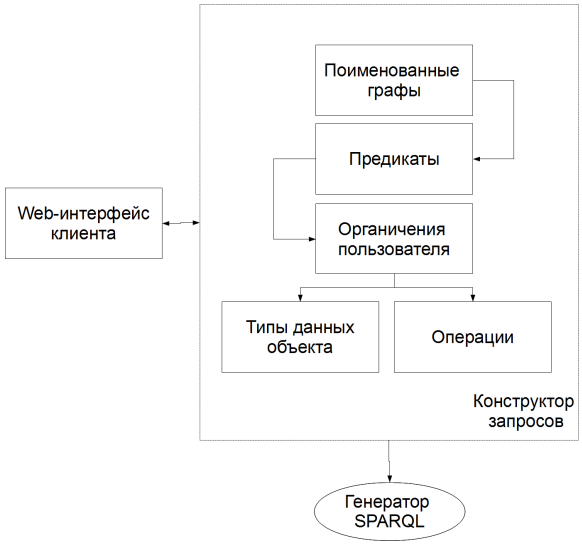

Рисунок 2 – Архитектура веб-интерфейса к хранилищу триплетов на основе конструктора запросов SPARQL

В рамках такого решения пользователь через веб-интерфейс получает доступ к перечню поименованных графов хранилища триплетов, выбор из которого позволит автоматически получить информацию о топологии графа, а именно о составляющих его предикатах и типах данных объектов, что может быть использовано для формирования поискового запроса путем установления пользовательских ограничений (фильтров) на выводимые данные с хранилища триплетов.

Ограничения пользователя определим как набор правил с условиями вывода объектов поименованного графа, типы которых определяют семантику применяемых операций сравнения. В качестве основы для условий вывода определим перечень операций, а именно «=» (равно), «>» (больше), «<» (меньше), которые могут быть применены к объектам.

Семантика операций «=», «>», «<» определяется типом объектов поименованного графа, т.е. каким именно образом будет реагировать конструктор запросов на символьные имена операций «=» «>» «<» определяется типом объектов поименованного графа.

Для целочисленного типа объекта (xsd:integer) операции «=», «>», «<» имеют следующий смысл:

«=» – сравниваемые объекты равны;

«>» – объект, с которым проводится сравнение больше сравниваемого;

«<» – объект, с которым проводится сравнение меньше сравниваемого;

Для строкового типа (xsd:string) под действиями «=», «>», «<» будем подразумевать следующее :

«=» – сравниваемые объекты равны;

«>» – сравниваемый объект является подстрокой объекта, с которым проводится сравнение;

«<» – объект, с которым проводится сравнение является подстрокой сравниваемого;

Действия «=», «>», «<» для объектов типа «xsd:float» (число с плавающей точкой) имеют такую же семантику, как и соответствующие операции с целочисленным типом объектов.

Таким образом, пользователь может формировать правила вывода содержимого того или иного поименованного графа хранилища триплетов. На основе полученных правил может быть автоматически сформирован типовой SPARQL-запрос к хранилищу триплетов, практические аспекты создания которого рассмотрены в разделе 2.

2. Практические аспекты реализации поискового пользовательского интерфейса к хранилищу триплетов

Архитектура веб-интерфейса к хранилищу триплетов на основе конструктора запросов SPARQL (рис. 2) подразумевает выполнение набора предопределенных действий с помощью SPARQL для динамического получения топологии поименованного графа. К таким действиям относятся:

- получение списка поименованных графов;
- получение списка предикатов в поименованном графе;
- получение типа данных предиката.

Для получения списка всех имеющихся поименованных графов в хранилище триплетов можно использовать следующий запрос на языке SPARQL:

```
SELECT distinct ?g WHERE
{ GRAPH ?g { ?s ?p ?o} }
```

Для получения конкретного поименованного графа или группы, имя которых соответствует некоторому критерию можно использовать следующий запрос на языке SPARQL:

```
SELECT distinct ?g WHERE {
GRAPH ?g
   { ?s ?p ?o . Filter(regex(?g, <http://shcherbak.net/>))
   } }
```

где из перечня всех графов выбираются те, у которых http://shcherbak.net/ встречается в имени графа;

Для получения списка предикатов в поименованном графе можно использовать следующий запрос на языке SPARQL:

```
SELECT distinct ?p
FROM <http://shcherbak.net/User>
{ ?s ?p ?o }
```

где из графа <http://shcherbak.net/User> будут выбраны все уникальные предикаты и возвращены в виде набора в качестве результата выборки.

Для получения типа предиката можно использовать следующий запрос на языке:

```
SELECT datatype(?o)
FROM <http://shcherbak.net/User>
WHERE {
?s ns:date_of_ birthday ?o }
```

где для предиката ns:date_of_birthday графа http://shcherbak.net/User будет получен тип объекта.

На основе подобных запросов можно строить произвольные запросы на выборку с поименованных графов, например:

```
sparql SELECT ?s ?p ?o
FROM <http://shcherbak.net/User>
WHERE { ?s ?p ?o   FILTER ( ?s = ns:2 ) }
```

где с графа http://shcherbak.net/User будут выбраны триплеты пользователя ns:2.

Кроме того, в качестве ограничений пользователя могут выступать языковые теги (lang tags). В этом случае, для вывода информации с графа http://shcherbak.net/User можно установить фильтр на вывод объектов на русском языке, у которых языковой тег установлен в значение «ru»:

```
SELECT ?name ?second_name
FROM <http://shcherbak.net/User>
WHERE {
 ?s ns:name  ?name.
  ?s  ns:second_name  ?second_name.
  FILTER ( ?name = 'Иван'@ru && ?second_name = 'Иванов'@ru ) }
```

Для поиска объекта с учетом нескольких пользовательских ограничений можно использовать следующий SPARQL-запрос:

```
PREFIX ns:<http://shcherbak.net/>
SELECT  ?o
FROM <http://shcherbak.net/User>
WHERE
{  ?s ns:name ?o
   FILTER regex(?o, "Иван",  "i" )
   FILTER regex(?o, "Петров",  "i" ) }
```

Реализация поискового интерфейса на основе вышеприведенных запросов (рис.3) предоставляет пользователю вспомогательную информацию, которая может значительно облегчить процедуру составления запроса к хранилищу триплетов и уменьшить время его составления.

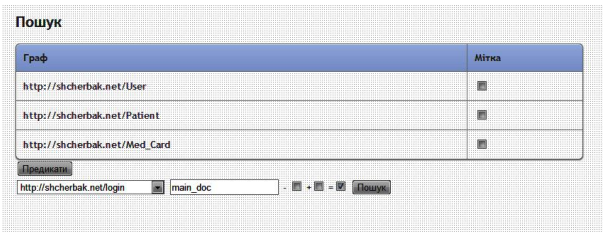

Рисунок 3 – Пример поискового интерфейса

Процедура составления запроса сводится к выбору поименованных графов, представляющих интерес в контексте поиска и наложении

ограничений на предикаты и значения выводимых в результате поиска объектов.

Результаты поиска группируются по принадлежности к некоторому субъекту в табличном представлении (Рис. 4). Таким образом, на клиенте из множества триплетов, полученных в результате выполнения запроса SPARQL на сервере, компонуется запись в удобном для чтения человеком виде.

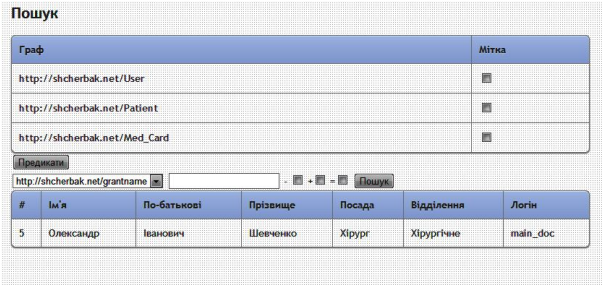

Рисунок 4 – Пример вывода результатов поиска в табличном представлении

Язык RDF позволяет формировать произвольной сложности графовые структуры для представления данных, что составляет как в вычислительном плане, так и в плане эффективного использования проблемы по организации взаимодействия через интерфейс ODBC (англ. Open Database Connectivity) с хранилищем триплетов, поэтому рассмотрим ограничения предложенного решения.

Предполагается, что данные, содержащиеся в хранилище триплетов, структурированы с использованием принципов объектно-ориентированного проектирования, а именно предполагается, что поименованный граф является контейнером (классом) для множества объектов этого класса, однозначно идентифицируемых по субъекту триплета.

Ограничения подразумевают исключительно более удобную работу с хранилищем триплетов для разработчиков программного обеспечения, использующих в своей работе реляционные системы управления базами данных. Без учета ограничений предложенное решение является работоспособным, но реальный смысл генерируемых данных в качестве результатов поиска изменится.

Для учета ограничений данные об объектах предлагается создавать поименованные графы на основе типовых, предлагаемых ниже, SPARQL запросов.

Для создания поименованных графов можно использовать следующий SPARQL-запрос:

create graph <http://shcherbak.net/User>

где создается граф пользователя «http://shcherbak.net/User» в текущем хранилище триплетов.

Для добавления триплетов в поименованный граф можно использовать следующий SPARQL-запрос:

```
INSERT DATA INTO
<http://shcherbak.net/User>
{  ns:log1 ns:login "dxfg".
 ns:log1 ns:firstname "Иван".
 ns:log1 ns:lastname  "Иванович".
 ns:log1 ns:grantname  "Иванов".}
```

ЗАКЛЮЧЕНИЕ

В данной работе предложена архитектура универсального поискового интерфейса, который обеспечивает визуальное построение и выполнение запросов к хранилищам триплетов на языке SPARQL. После выполнения запроса пользователь получает контекст с триплетами, соответствующими искомым с учетом заданных ограничений на предикаты и объекты.

Практическая реализация решения создана на языке программирования PHP и прошла успешную апробацию на базе хранилища триплетов OpenLink Virtuoso.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

[DuCbarme, 2011] DuCbarme B. Learning SPARQL / B.DuCbarme. - O'Reilly Media: 2011. - 258 c.

[Powers, 2008] Powers S. Practical RDF / S. Powes. - O'Reilly Media: 2008. - 352 c.

[Холзнер, 2004] Холзнер С. XML. Энциклопедия, 2-е издание / Стивен Холзнер. - СПб.:Питер, 2004. - 1101 с.:ил.

[Бек, 2008] Бек К. Шаблоны реализации корпоративных приложений.: Пер. с англ. - М.:ООО "И. Д. Вильямс", 2008. - 176 с.:ил.

PRACTICAL ASPECTS OF SEARCH INTERFACE IMPLEMENTATION FOR SEMANTIC APPLICATIONS BASED ON LINKED DATA

Galushka I.N. *, Zavgorodniy V. V. **, Soloshish S.N. *, Shcherbak S.S. *

**Kremenchuk Mykhailo Ostrohraskyi National University, Kremenchuk, Ukraine*

sergey.shcherbak@gmail.com

*** Dneprodzerzhinsk state technical university, Dneprodzerzhinsk, Ukraine*

The increase of linked data popularity along with information description in the form of RDF-triplets caused the necessity of studying the procedures for interaction with triple stores. A unique search interface have been developed, which enables visual querying the triple stores. These visual queries are automatically converted into SPARQL query language that is used for triple store accessing. After the query is executed, a user gets the desired context with triplets according to the set constraints for predicates and object