ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М. А. БОНЧ-БРУЕВИЧА»

Технологии семантического анализа данных

ПРАКТИКУМ

СПбГУТ)))
САНКТ-ПЕТЕРБУРГ

2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Широкое развитие стандартов и поддержка концепции Linked Data (связанные данные) крупными информационными компаниями определяет тенденции развития будущего WWW (World Wide Web) как глобальной базы данных.

В рамках Linked Data информация описывается в терминах языка RDF (Resource Description Framework), а именно, в виде триплетов, троек вида «субъект-предикат-объект» или квадов (quad) –поименованных графов вида «граф-субъект-предикат-объект».

В соответствии с названием, язык RDF – это средство для описания ресурсов, первоначально был разработан для добавления метаданных к веб-ресурсам, а потом стал языком описания семантической информации ресурсов.

Данный язык является машиночитаемым, что позволяет достаточно просто выполнять кодирование, обмен и повторное использование структурированных метаданных между различными программами

Модель данных RDF предполагает распределенное хранение информации на различных web-серверах с интегрированным или внешним специализированным хранилищем триплетов (Triplestore).

Для доступа к хранилищам триплетов используется протокол и язык запросов SPARQL (SPARQL Protocol and RDF Query Language).

SPARQL - это рекурсивный акроним для SPARQL Protocol And RDF Query Language, который описывается набором спецификаций из W3C [1].

Язык SPARQL представляет собой язык описания запросов к RDF данным, а также протокол передачи запросов и получения ответов. Источниками RDF-данных к которым направляются запросы могут быть любые RDF-документы или базы знаний (доступные с помощью endpoint).

SPARQL как протокол определяет правила взаимодействия программы-клиента SPARQL с сервером обработки SPARQL запросов.

. В настоящее время активно ведутся исследования по использованию технологий Semantic Web в таких областях, как электронные библиотеки, интеграция и поиск информации в сети Интернет и системы управления знаниями. Перспективность данного направления развития информационных технологий подтверждается результатами исследований как зарубежных, так и российских ученых.

Основная цель учебного пособия предоставить студентам информацию об основных принципах формирования RDF хранилищ и научить их использовать язык запросов SPARQL для извлечения данных из этих хранилищ данных.

Ограниченный объем пособия не позволяет достаточно подробно осветить все вопросы, касающиеся важнейших проблем разработки и использования RDF-хранилищ данных, SPARQL-запросов, поэтому для получения дополнительных сведений рекомендуется обратиться к соответствующим источникам, список которых включает ___ наименования. При работе над пособием авторы неоднократно осуществляли заимствование из этих источников определений, примеров и методов изложения.

В заключение авторы благодарят рецензентов и редакторов за внимательное прочтение рукописи и замечания, способствовавшие улучшению качества предлагаемого пособия

1. Основы RDF

1.1. RDF модель хранения данных

RDF (Resource Description Framework) – это модель описания связанных данных, которая позволяет технологии Семантического веба интерпретировать информацию, представленную в сети.

Общая идея, на которой основана модель RDF, заключается в следующем: всё, что существует в мире (будь то физический предмет или абстрактное понятие), имеет определенные свойства, а любое свойство имеет конкретные значения. Значит, описать любую сущность можно с помощью элементарных выражений, которые перечисляют эти свойства и определяют их значения.

Основу модели RDF представляет трехчастное утверждение, или триплет, следующего вида:

Субъект – Предикат (или свойство) – Объект (значение свойства).

Триплет можно соотнести с простым предложением вида

Подлежащее – Сказуемое – Дополнение.

Например, утверждение «*Книга написана Л.Н. Толстым*» в RDF-терминологии можно представить следующим образом: субъект – «*Книга*», предикат – «*написана*», объект – «Л.Н. *Толстой*».

Такое выражение принято представлять в виде графа, в котором субъект и объект – это узлы, а предикат изображается дугой или иной соединительной линей, направленной от субъекта к объекту (рис.1).

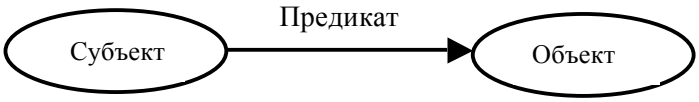


Рис.1. Структура триплета RDF-графа

Субъект в данном случае является описываемым ресурсом, предикат – это наименование свойства ресурса, а объект – значение свойства описываемого ресурса

Описание реального объекта представляет собой совокупность триплетов (рис.2).

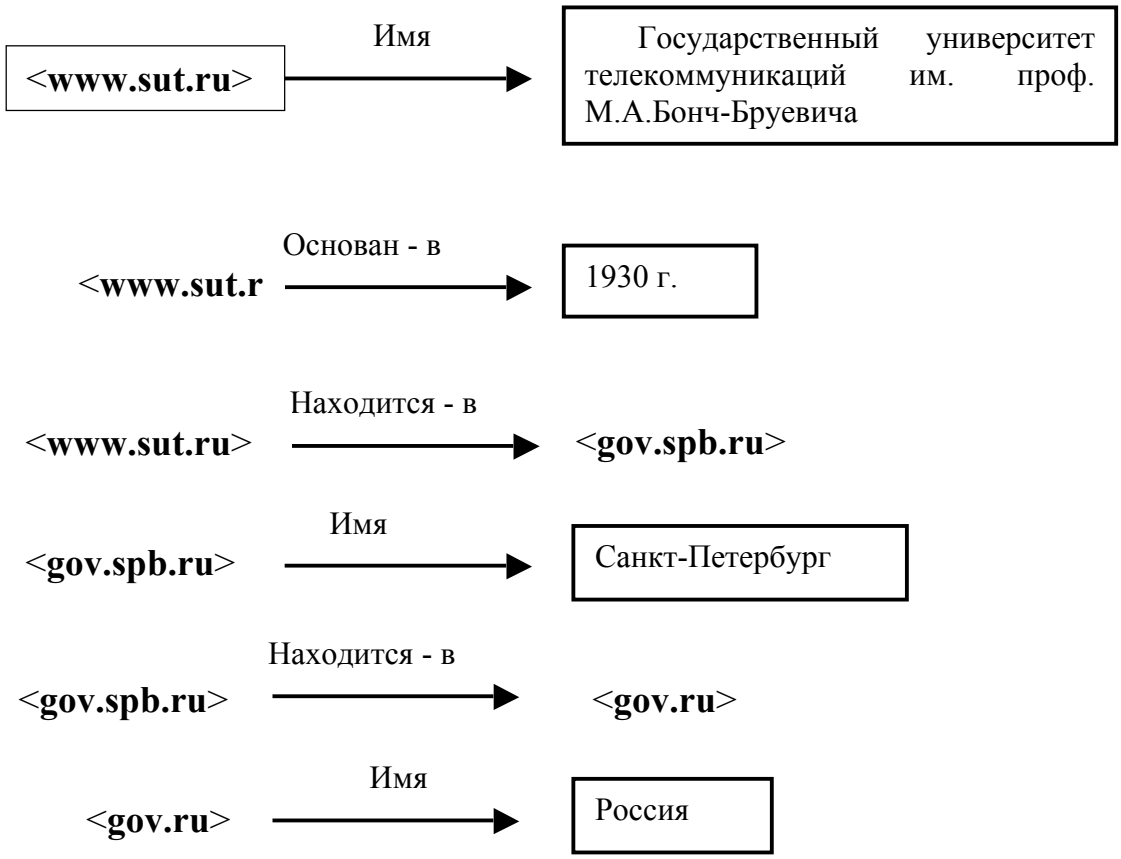


Рис.2. Описание объекта СПбГУТ в виде совокупности триплетов

Все триплеты RDF документа объединяются в общий RDF – граф (рис.3).

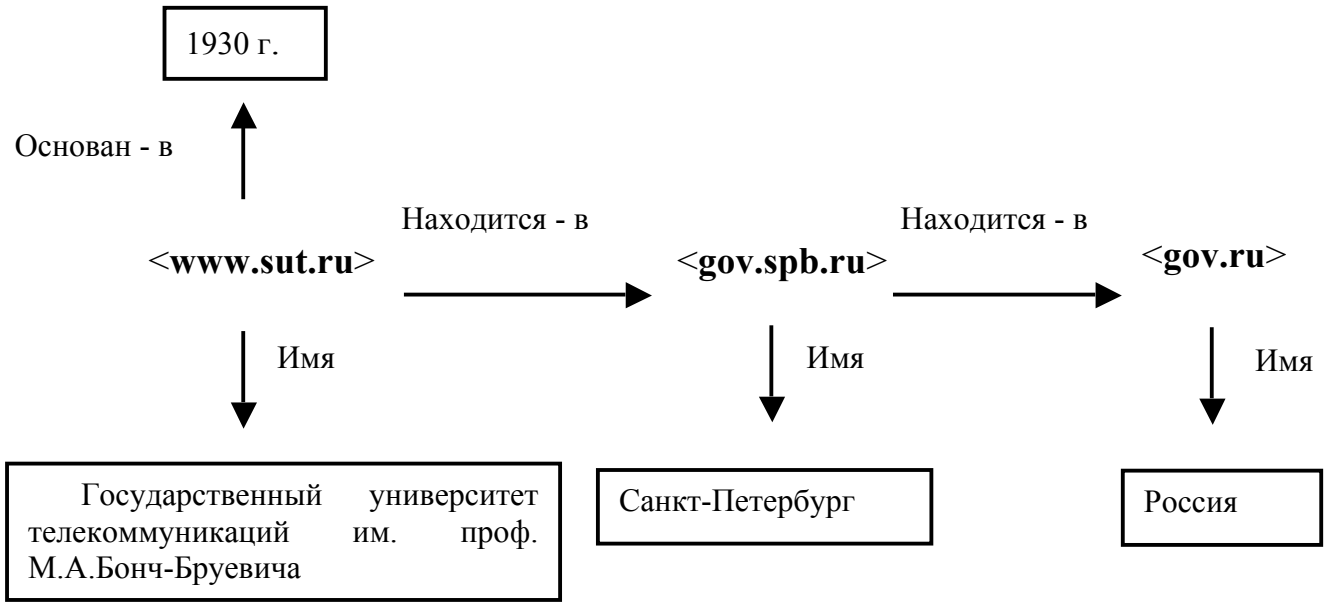


Рис.3. Объединение RDF триплетов в RDF-граф

Все RDF-графы объединяются между собой связями, образуя общий граф RDF документов (Giant Global Graph, рис. 4).

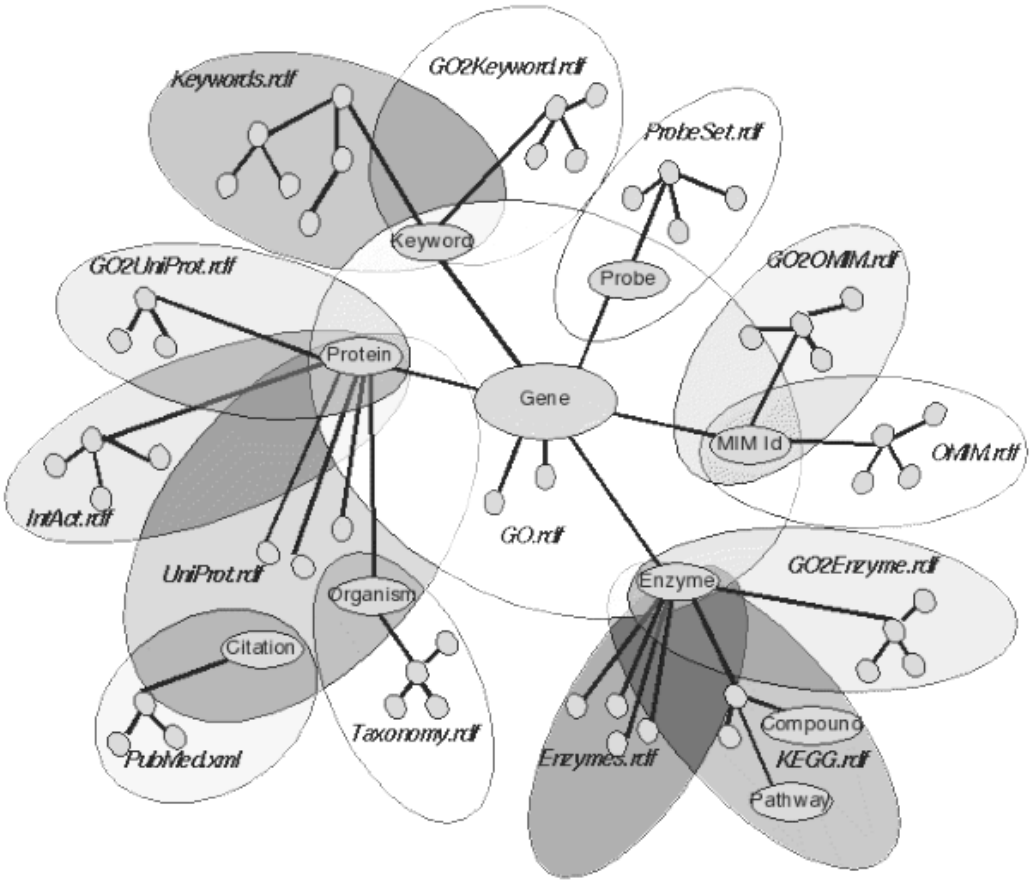


Рис. 4. Оъединенный граф RDF документов (Giant Global Graph (GGG)).

Объединенные RDF-графы позволяют с помощью логических выводов генерировать новые, несуществующие до этого триплеты (рис. 5)

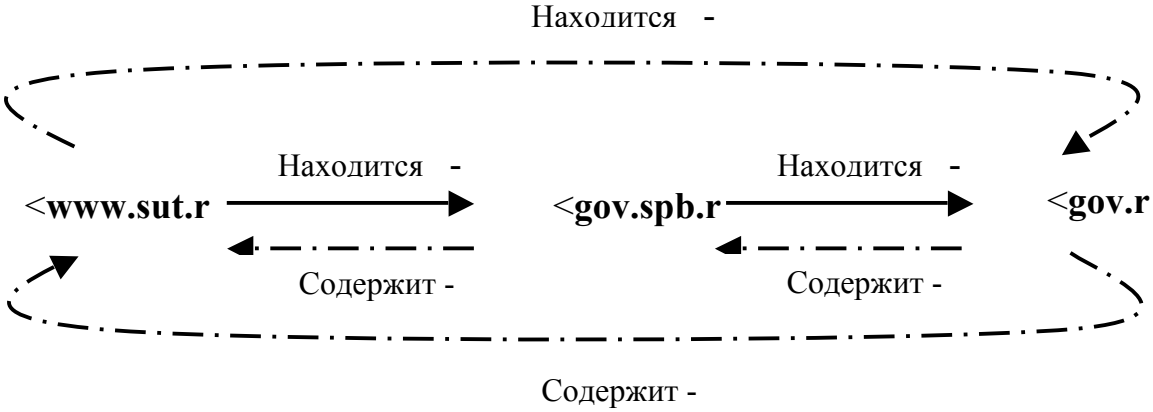


Рис. 5. Логические выводы на базе RDF-графа с генерацией новых триплетов, где ⟶ новые свойства субъектов

1.2. Состав RDF – триплетов

Основными элементами RDF-триплета является ресурс и его свойства. Наиболее рациональным способом указания вида необходимого клиенту ресурса является, например, *имя файла*, *каталог*, в котором хранится этот файл и *имя сервера*, на котором хранится данная файловая система.

Комбинацию *сервер-каталог-ресурс* называют унифицированным локатором ресурса (Uniform Resource Locator) или URL. Например, http://www.learningsparql.com/resources/index.html является URL с указанием конкретного протокола, который используется для доступа к данному ресурсу. Очевидно, что URL могут быть использованы только по отношению к электронным ресурсам.

Для обозначения ресурсов реального мира была заработана спецификация, определяющая универсальное имя ресурса (Universal Resource Names) URN, которое не связано с адресом расположения ресурса в интернете. Например, *urn:isbn:9660313837* – является URN ресурса «Книга "Капитал. Том I"», определённого по коду ISBN, а *urn:oid: 1.2.643* – является URN ресурса «Страна Россия», определённого по коду OID.

URN также обладают определенными недостатками основными из которых являются следующие:

- не для всех типов ресурсов созданы схемы кодирования;
- нет данных о месте расположения ресурса, то есть, невозможно перейти по коду как по ссылке и получить информацию о ресурсе.

Для того, чтобы получить возможность использования как электронных ресурсов, так и ресурсов мира вещей в дальнейшем предложено использовать универсальный идентификатор ресурса (Uniform Resource Identifier) URI.

В общем случае, URI представляет собой некую последовательность символов, идентифицирующую абстрактный или физический ресурс.

К недостаткам URI относится необходимость использования при их записи ограниченного набора только латинских символов и знаков препинания.

Эту проблему позволяет решить стандарт IRI (Internationalized Resource Identifier) — международный идентификатор ресурсов, в котором можно без проблем использовать символы Юникода, и которые не ущемляют права других языков.

Так язык запросов SPARQL ориентирован на использование IRI, так как это позволяет использовать при формировании запросов и ответов многоязычные формы описания ресурсов.

Узлы графов RDF документов представляют собой субъекты и объекты триплетов. Как правило, все узлы имеют имена, представляющие собой имя ресурса (URI) или для объектов - литералы (строки или другие типизированные значения). Кроме того, для структуризации данных, используются пустые (анонимные, безымянные) узлы (рис.6).

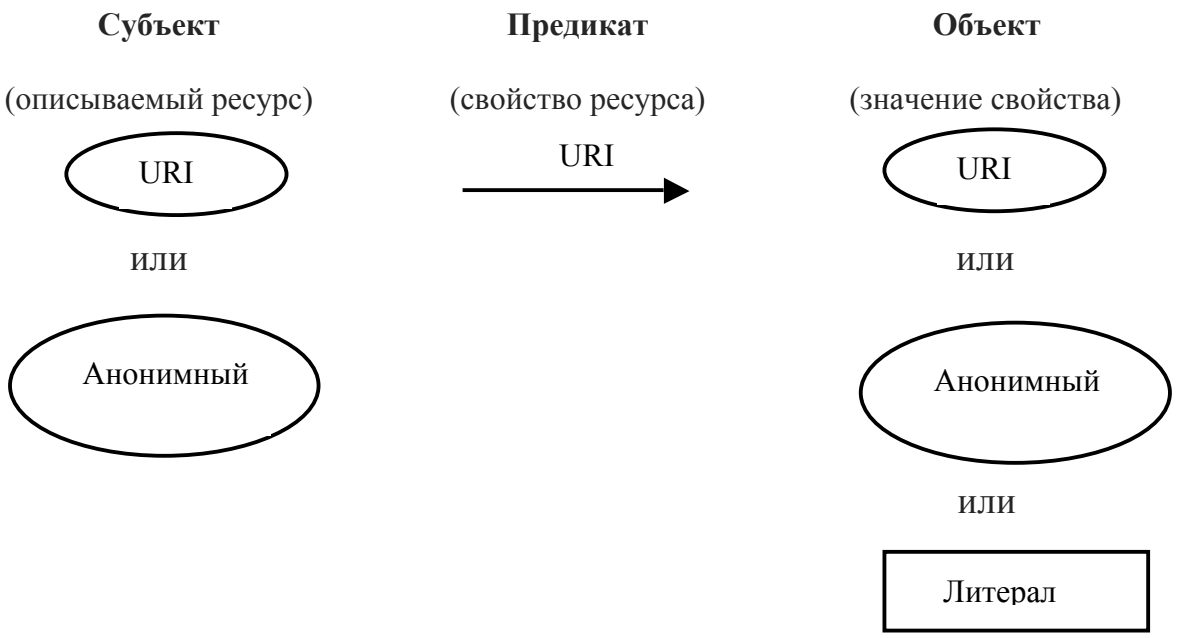


Рис. 6. Возможные значения элементов триплета

В качестве URI может быть использован URL документа, описывающий ресурс. Например, для ресурса "Кремль" можно использовать URL документа с его описанием **http://www.kreml.ru**/ – официальный сайт кремля.

В то же время ресурс Кремль является объектом реального мира, а не электронным документом. Поэтому в RDF документах в качестве URI ресурса "Кремль" могут быть использованы следующие формы

<**http://www.kreml.ru**/> - URI сайта "Кремль" и

<**http://www.kreml.ru#about**> - URI самого объекта "Кремль".

Согласно принятым соглашениям, полные URI, в составе RDF документов, заключаются в угловые скобки.

Анонимные или безымянные узлы (blank nodes или bnodes) указывают на то, что эти узлы в графе не имеют собственного имени, в

силу того, что автор документа не знает имя узла, не хочет или не может его предоставить. Это эквивалентно высказыванию "Петр дружит с кем-то, но я не знаю с кем".

При наличии утверждения о том, что эти узлы не имеют имени, нужно иметь в виду следующее. Во-первых, сущность реального мира, на которую ссылается данный узел, не обязана быть безымянной. Друг Петра из примера все-таки имеет какое-то имя. Во-вторых, определяя узел как "безымянный", подразумевается использование концепции именования сущностей с помощью URI.

На самом деле, анонимные узлы в RDF документе, как правило, имеют "локальные" идентификаторы, на которые можно ссылаться только внутри данного документа.

Кроме того, анонимные узлы используются для структурирования данных в составе RDF документа.

Пример 1. Приведенный документ представляет записи в адресной книге, где субъект **ab:i0432** имеет предикат **ab:address** (рис.7), значение которого имеет особенный префикс пространства имен: подчеркивание. Это значение в свою очередь, имеет свои собственные значения, описывающие отдельные компоненты адреса Ричарда.

```
# filename: ex041.ttl
@prefix ab: <http://learningsparql.com/ns/addressbook#> .

ab:i0432 ab:firstName "Richard" ;
         ab:lastName "Mutt" ;
         ab:homeTel "(229) 276---5135" ;
         ab:email "richard49@hotmail.com" ;
         ab:address _:b1 .

_:b1 ab:postalCode "49345" ;
     ab:city "Springfield" ;
     ab:streetAddress "32 Main St." ;
     ab:region "Connecticut"
```

Рис. 7. RDF записи адресной книги

Префикс с подчеркиванием означает, что это пустой узел. Его цель в данном случае состоит в выделении определенной группы значений предикатов и объектов.

_:b1 используется как локальное имя для указания места выделенной группы значений, когда необходимо обратиться к этой группе троек из других частей этого набора данных.

RDF программное обеспечение, которое читает эти данные, может игнорировать значение b1, но оно должно помнить, что Ричард (или ресурс ab:i0432) имеет свойство ab:address со значением b1, которое указывает на четыре других свойства и их значения (рис.8).

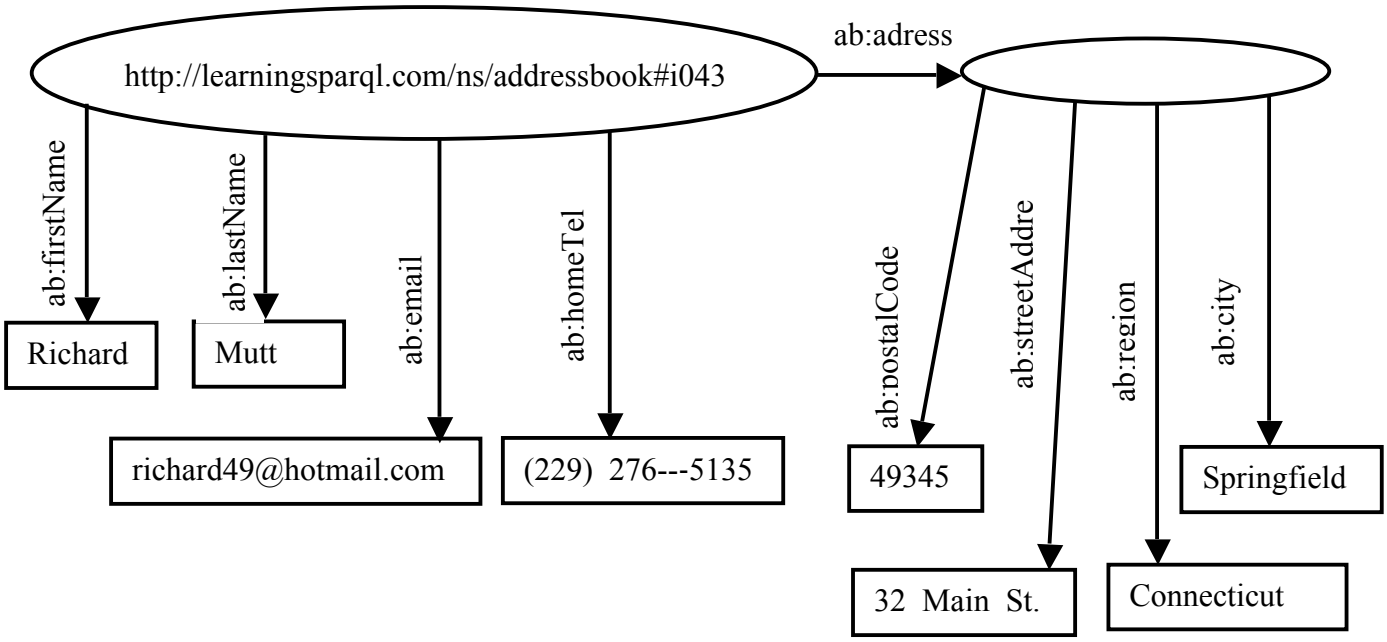


Рис. 8. Граф RDF записи адресной книги с пустым узлом

Литеральные значения представляют собой просто текст, который может быть использован вместо объектов в RDF-триплетах. В отличии от имен (то есть URI), которые заменяют собой сущности в реальном мире, литеральные значения — это просто текстовые данные вставленные в граф. Литеральные значения используются для того, чтобы установить связь между людьми и их именами, книгами и их номерами ISBN, и т.д.

Литеральные значения могут дополняться одним из двух возможных элементов метаданных. Первый это дескриптор языка, который указывает, на каком языке представлено текстовое значение литерала.

Выражение **'chat'@en** представляет собой литеральное значение **'chat'** с английским дескриптором языка, или **'chat'@fr** то же самое значение литерала, но с французским дескриптором языка.

Литералы с одинаковым текстом, но разными кодами языка не равны друг другу (рис.9).

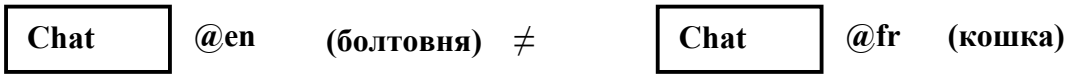


Рис. 9. Использование дескриптора языка для литералов

Кроме того, литеральные значения могут быть помечены с помощью URI, идентифицирующего тип данных (типизированные литералы). Тип данных указывает, как интерпретировать текст литерала: как число, URI, дату или время, и др. Типом данных может быть любой URI, но обычно для этого используются типы данных, определенные в XML Schema.

Для обозначения типа данных обычно используется запись следующего вида: после литерального значения в кавычках следуют два символа "крышка" ('^^') и URI, определяющий тип данных (может быть указан в сокращённой форме).

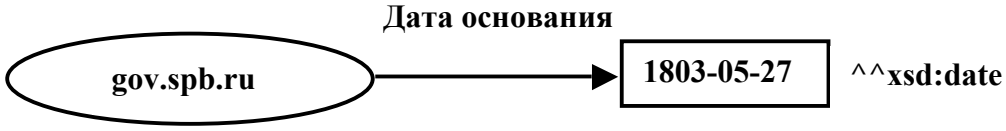


Рис. 10. Литерал с указанием типа данных

Типизированные литералы с разным текстом могут быть равны, если соотносятся с одним и тем же значением (рис.11).

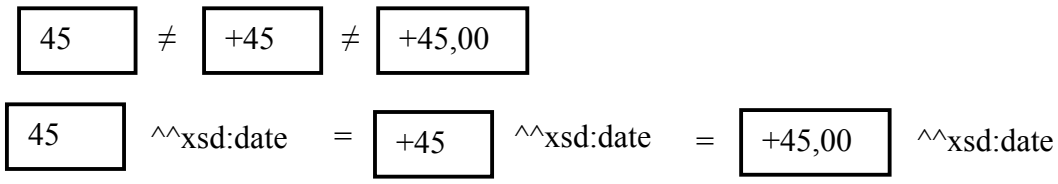


Рис. 11. Соотношения типизированных и нетипизированных литералов

1.3. RDF – классы

Ресурсы, используемые в RDF моделях, объединяются в группы, называемые классами. Члены таких классов называются экземплярами класса. Сами классы также являются ресурсами и идентифицируются соответствующими URI.

Принадлежность ресурса классу задается встроенным предикатом *rdf:type* (рис. 12).

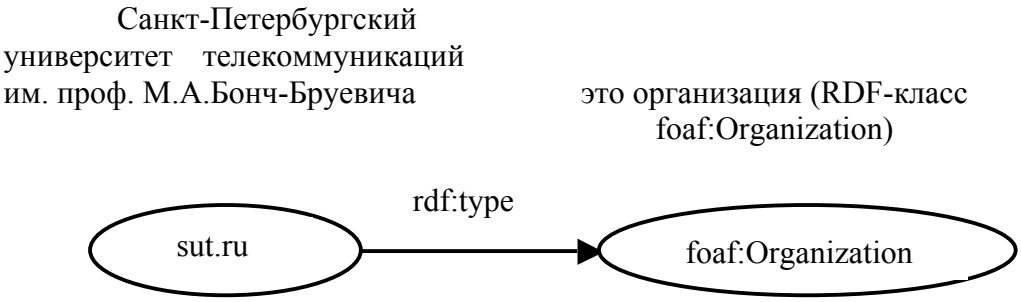


Рис. 12. Использование предиката rdf:type для задания класса

RDF содержит несколько встроенных классов, таких как *rdf:Statement*-для задания реификации; *rdf:Bag, rdf:Seq, rdf:Alt, rdf:List* - для создания раличных контейнеров и списков и др. [1].

1.3.1. Реификация

Реификация является утверждением об утверждении.

Для создания утверждения об утверждении создается ресурс, относящийся к классу *rdf:statement* и имеющий три свойства:

- *rdf:subject* - субъект утверждения;
- *rdf:predicate* - предикат;
- *rdf:object* - объект.

Так, например, утверждение о том, что "Мюллер думает, что Штирлиц является гражданином Третьего рейха" имее следующий вид (рис.13).

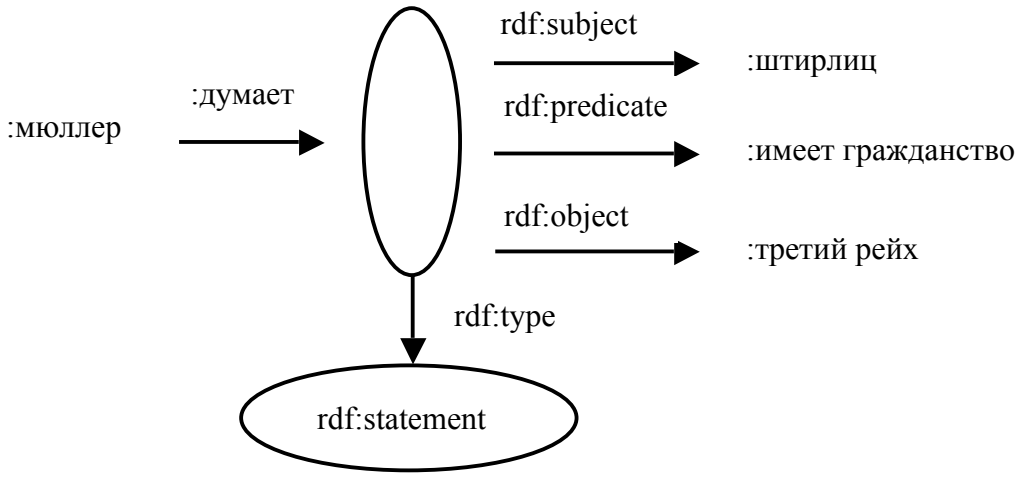


Рис. 13. Использование класса rdf:Statement

1.3.2. RDF-контейнеры

Одним из факторов существенного повышения мощности языка RDF являются средства представления структур данных, образуемых из описаний отдельных ресурсов. В RDF поддерживаются два вида базовых структур данных – контейнеры и коллекции. Эти средства RDF позволяют группировать ресурсы (и литералы) и использовать их в высказываниях как единое целое.

В RDF предусмотрена возможность использования следующих трех типов контейнеров:

• **bag** (рис. 14) — неупорядоченный набор (Apple основал Стив Джобс, Стив Возняк и Рональд Уэйн.);

• **alt**— набор возможных альтернатив (столицей XXIII Зимних Олимпийских игр будет Анси, Мюнхен или Пхёнчхан.);

• **seq**— упорядоченный набор (в случае отставки президента США его замещают по порядку: вице-президент, спикер Палаты представителей, временный председатель Сената, государственный секретарь, и т.д.).

Bag-контейнер представляет собой неупорядоченную группу ресурсов или литералов (с возможностью повторений), для которой не вводится порядковых отношений между элементами.

В приведенном примере контейнер (как целое) описывает пустой узел, на который (как на объект) ссылается свойство **«:основан»**. Для указания принадлежности некоторых сущностей (элементов) к контейнеру в RDF используется специальное обозначение свойств ресурса-контейнера, имеющее общий вид **«rdf:_n»,**

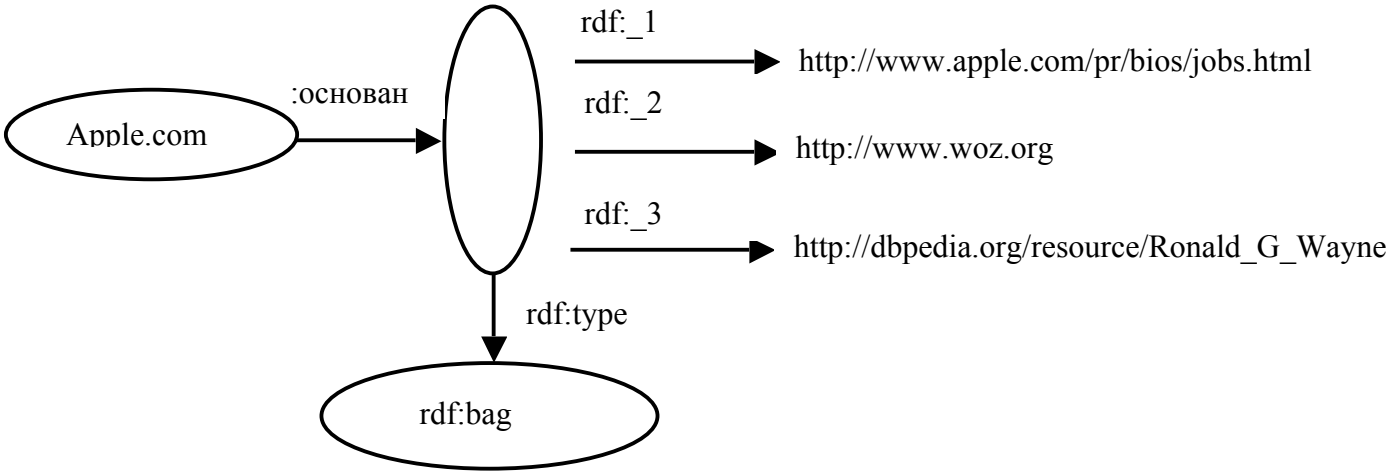


Рис. 14. Использование контейнера типа Bag

где n – произвольное десятичное положительное целое, большее нуля. Обязательным при этом является определение свойства **«rdf:type»,** задающего тип контейнера.

Наборы альтернатив **«rdf:alt»** представляют собой неупорядоченные группы ресурсов или литералов, среди которых не могут встречаться повторения. При этом результатом обработки набора альтернатив всегда является выбор одного из элементов контейнера (альтернативы). Особенностью **«rdf:alt»** является то, что элемент, соответствующий свойству **«rdf:_1»**, имеет предопределенное назначение и считается «альтернативой по умолчанию».

Последним, третьим типом контейнеров являются последовательности **«rdf:seq»**, которые определяют группы линейно упорядоченных элементов (возможно, повторяющихся), представляющих собой ресурсы или литералы.

Следует отметить, что с точки зрения модельного (в виде графа) представления все виды RDF-контейнеров полностью аналогичны. Различие заключается лишь в способах интерпретации этих структур данных и в указании типа контейнера в качестве объекта для предиката **«rdf:type»**.

1.3.3. RDF- коллекции

Коллекции являются базовыми структурами данных в RDF и представляют собой фактически линейную списочную структуру, связывающую (через специальные свойства пустых узлов-субъектов) в единую последовательность различные триплеты.

Коллекции являются закрытыми группами, т.е. они содержат только те элементы, которые были перечислены и больше никаких других по определению. (Согласно канонической христианской традиции, Троица состоит из Бога-отца, Бога-Сына и Святого Духа и ни из кого больше по определению).

В отличие от контейнеров, в которых один ресурс-субъект (обозначающий контейнер в целом) связан различными свойствами с различными объектами, коллекции связывают узлы одними и теми же стандартными свойствами, определяющими порядок следования элементов.

На рис. 15 изображен RDF-граф, описывающий использование коллекции.

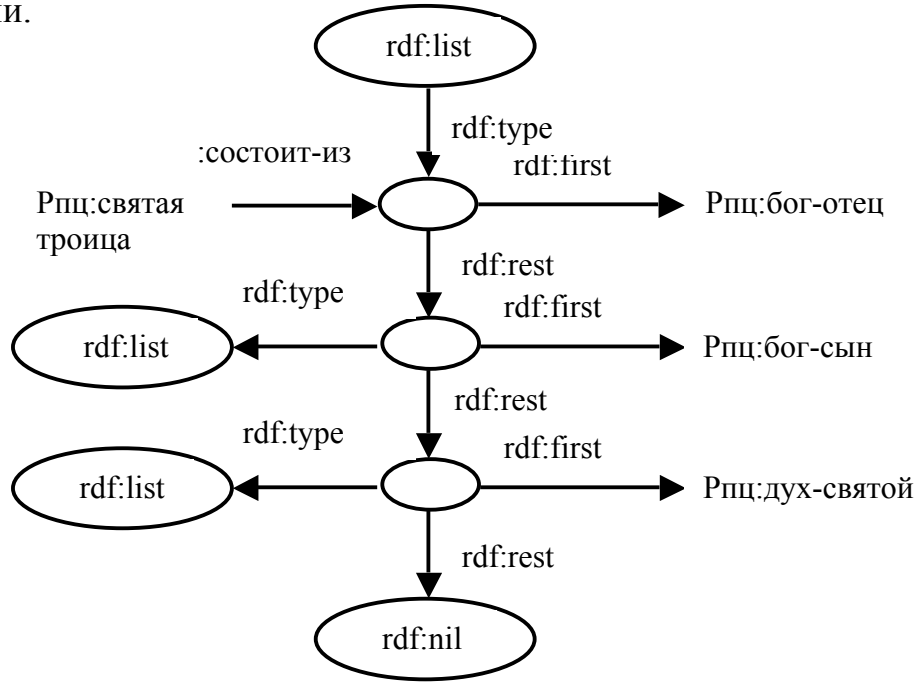


Рис. 15. RDF модель с использованием коллекции

Из примера видно, что в RDF-графе отсутствуют явные указания на то, что в модели использована коллекция. Здесь все элементы коллекции являются объектами одного и того же предопределенного свойства (предиката) – **«rdf:first»** и имеют в качестве субъекта своего триплета пустой узел, который связывается со следующим аналогичным пустым узлом и

другим предопределенным свойством **«rdf:rest»**. Агрегирование всех элементов в единую структуру данных происходит за счет того, что все пустые узлы, формирующие основной список, неявно (без отображения в графе) получают зарезервированное в RDF свойство **«rdf:type»**, значение которого становится равным предопределенному символу **«rdf:list»**. Признак окончания списка задается в коллекциях с использованием предопределенного символа – **«rdf:nil»**.

1.4. Сериализация RDF

Сериализация - это технический термин обозначающий технологию сохранения RDF в виде строки байтов, которая в дальнейшем может быть сохранена на магнитном диске или другом запоминающем устройстве . Использование термина "сериализация", а не "файла", объясняется тем, что существуют операционные системы, которые не используют термин "файл", но на практике все RDF сериализации до сих пор были текстовыми файлами, которые использовали различные форматы (синтаксисы) представления троек.

Существуют следующие форматы сериализации:

• RDF/XML—стандартный формат на базеXML.

• N-Triples—простой формат.

• Turtle—удобный формат.

• JSON-LD—формат на базе JSON.

• RDF и Microdata—формат RDF-разметки HTML- страниц.

Ниже рассматриваются только стандартный формат на базеXML и наиболее удобный для понимания формат Turtle.

1.4.1. RDF/XML- формат описания RDF графов

Формат описания RDF задается нотацией RDF/XML (документ консорциума W3C), регламентирующий описание RDF-графов с использованием языка XML. Принципиальная возможность отображения семантики RDF в терминах XML определяется следующими основными причинами:

1) язык RDF базируется на формальной (т. е. строго определенной) семантике, описываемой средствами абстрактного синтаксиса, задающего не конкретный вид элементов и конструкций отображаемого языка, а их модельное представление и способ логической интерпретации. В качестве подобной синтаксической формы для RDF может выступать XML.

2) язык XML является одним из классических представителей языков с расширяемой (открытой) семантикой. Этот язык предоставляет возможность вводить в свои конструкции новые синтаксические элементы, которые могут интерпретироваться произвольным образом. Т. е. XML может рассматриваться как «посредник» – средство «транспортировки» семантической информации, которая может быть востребована программными приложениями, способными ее «понять» (интерпретировать) тем или иным способом. Соответственно, XML может являться посредником и при передаче семантической информации RDF.

Рассмотрим основные средства нотации RDF/XML на примере.

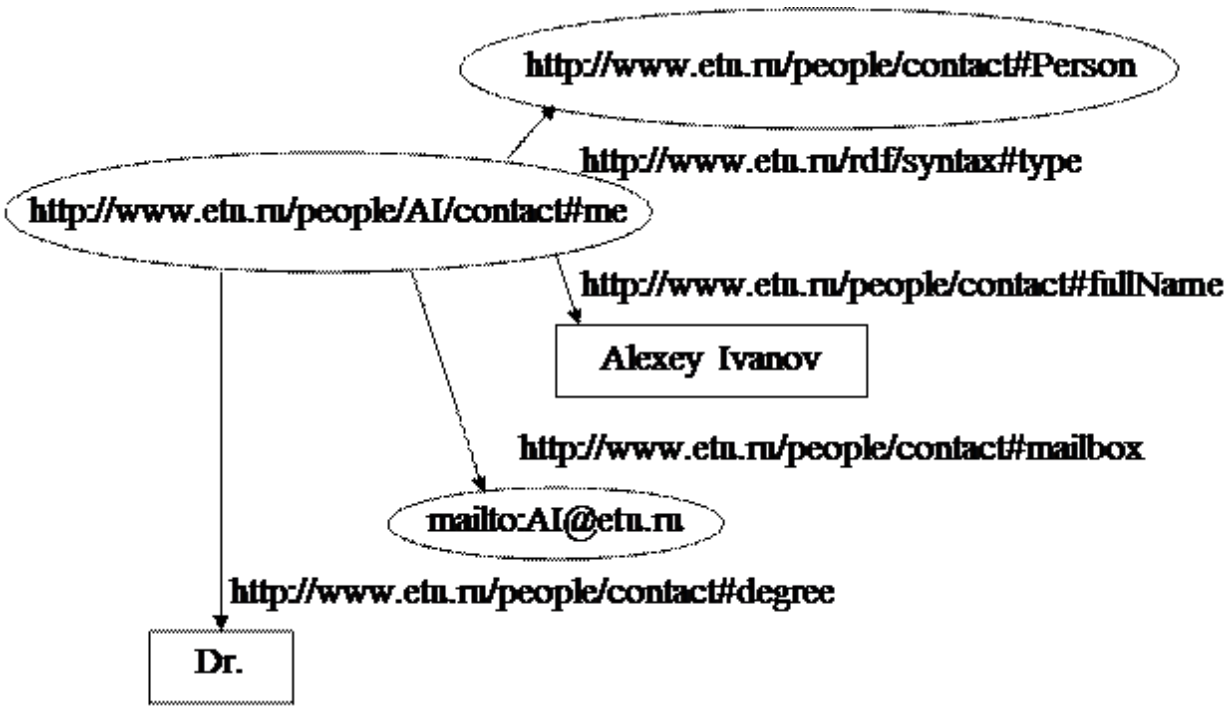


Рис. 16. Пример RDF графа

В соответствии с правилами синтаксиса RDF/XML RDF-граф, изображенный на рис. 16, может быть описан на XML следующим образом (рис.17).

```
# Пример 1
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.etu.ru/rdf/syntax#"
xmlns:contact="http://www.etu.ru/people/contact#">
<contact:Person rdf:about="http://www.etu.ru/people/AI/contact#me">
<contact:fullName>Alexey Ivanov</contact:fullName>
<contact:mailbox rdf:resource="mailto:AI@etu.ru"/>
<contact:degree>Dr.</contact:degree>
</contact:Person>
</rdf:RDF>
```

Рис. 17. Пример XML записи RDF графа

В XML вложению метаданных RDF соответствует специальный элемент rdf:RDF (ограниченный в тексте метками **«rdf:RDF»** и **«/rdf:RDF»**).

Этот элемент в свою очередь содержит последовательность XML-элементов (которые и образуют в совокупности RDF-документ) двух типов – элементов-узлов и элементов-свойств.

Элемент-узел связывается с тэгом **«rdf:Description»**, который в ряде случаев может опускаться или заменяться в целях упрощения записи. В элементы-узлы вкладываются описания элементов-свойств, которые описывают исходящие из узла дуги. Каждый элемент выделяется «операторными скобками» вида «<X» и «/X>» (в закрывающей скобке X может опускаться, если элемент не содержит вложенных описаний, как в строке **«<contact:mailbox …/»** Примера 1), где X является идентификатором элемента. Помимо этого внутри скобок могут использоваться описания атрибутов элемента (в примере определениям атрибутов соответствуют предложения **«xmlns:rdf=…»**, **«xmlns:contact=…»**, **«rdf:about=…»** и **«rdf:resource=…»**.

Идентификатор элемента представляет собой URI, который, в общем случае, может состоять из префикса (строки, предшествующей символу «:») и локального имени (остатка строки, расположенного справа от разделителя «:»).

Префикс задает пространство имен (namespace), используемое элементом, а локальное имя – непосредственное определение идентификатора в этом пространстве.

Наличие именованного префикса позволяет точно указывать область видимости любого идентификатора в RDF- и XML-документах. Так в примере пространству имен **«http://www.etu.ru/people/contact#»**

ставится в соответствие префикс **«contact»**. С учетом введения префиксов в XML-нотации употребляется термин «квалифицированное имя» (**qualified name – QName**) ресурса, например – **«contact:fullName»**, которому соответствует полная запись URI-ссылки **«http://www.etu.ru/people/contact#fullName»**. Такие имена могут использоваться как равноправные XML-тэги.

В RDF механизм префиксов имеет и более глубокое значение. Здесь префикс является основным средством указания словарей, описывающих содержание тех или иных терминов в соответствующем документе, средством определения контекстной зависимости терминов, а также средством разделения общезначимых понятий и понятий, специфичных для приложения, или различных трактовок (интерпретаций) одной и той же сущности разными «участниками общения».

Определение префиксов RDF-документа может быть реализовано в виде задания атрибутов **«xmlns:…=…»** (XML-namespace) элемента **«rdf:RDF»**. Обязательным является определение префикса **«rdf:»**, который задает местонахождение основного словаря – словаря служебных терминов самого языка RDF (в приведенном ранее примере этот префикс описан строкой **«<rdf:RDF xmlns:rdf="http://www.etu.ru/rdf/syntax#"»**).

Аналогичный словарь RDFS, который является подмножеством RDF, задается префиксом **«rdfs:»** (RDF Schema).

В тексте примера 1 все описание RDF-графа строится на базе описания одного «родительского» элемента-узла, сопоставленного URI-ссылке **«http:// www.etu.ru/people/AI/contact#me»**, и четырех элементов-свойств этого узла – **«#type»**, **«#fullName»**, **«#mailbox»** и **«#degree»**.

Каждый из элементов-свойств определяет в свою очередь по одному «дочернему» узлу графа, который задает значение соответствующего свойства. Так, литерал **«Alexey Ivanov»** является значением свойства **«#fullName»**, а ресурс **«mailto: AI@etu.ru»** – значением свойства **«#mailbox»**. Для описания каждого из свойств (вместе со значением свойства) внутрь XML-элемента, характеризующего «родительский» узел, вкладывается элемент-свойство. Исключение составляет свойство **«#type»**, которое является предопределенным символом RDF и позволяет в данном случае выразить явную спецификацию свойства **«#type»** (в виде **«<rdf:type resource="http://www.etu.ru/people/AI/contact# Person"/>»**) подстрокой **«contact:Person»** и заменить стандартный тэг <rdf:Description>, применяемый для описания элементов-узлов, на тэг **<contact: Person>** (иными словами, заменить описание самостоятельного элемента-свойства на описание тэга в элементе-узле).

Характеристика «родительского» узла в примере 1 реализована с использованием зарезервированного атрибута **«rdf:about»**, в котором

может быть задан произвольный ресурс, сопоставляемый с узлом RDF-графа. Другой стандартный атрибут **«rdf:resource»** предназначен для задания значений свойств или атрибутов элементов-узлов в тех случаях, когда эти значения описываются идентификаторами ресурсов (не литералами) и не являются субъектами каких-либо других триплетов.

Характерной особенностью RDF-графов является возможность структуризации данных не только для сущностей (например, информация об Алексее Иванове в рассмотренном ранее примере была структурирована по свойствам «Полное имя», «Адрес e-mail», «Ученая степень», «Тип – сотрудник»), но и для свойств. Пусть в примере (рис. 16) требуется задать для Иванова обобщенное свойство «Контакты», не имеющее однозначного значения, но имеющее подсвойства – адрес электронной почты и рабочий телефон (контекст предыдущего примера изменен для демонстрации использования обычных URI в качестве идентификаторов ресурсов). Для представления таких ситуаций в RDF-моделях предусмотрен механизм пустых узлов – «анонимных ресурсов» (рис. 18).

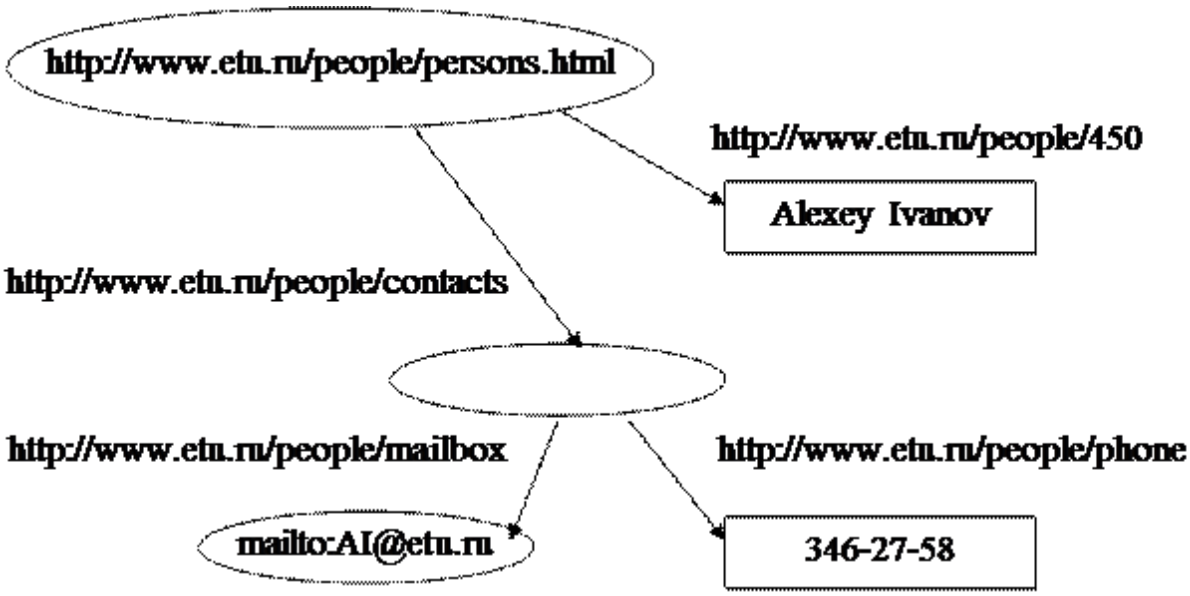


Рис. 18. Пример RDF графа с пустым узлом

Из рис. 18 видно, что абстрактное свойство «Контакты» **(«contacts»)** не определяет конкретных объектов-носителей свойства. Определенные значения имеют только подсвойства этого свойства – **«mailbox»** и **«phone»**. Соответственно, узлу, связанному со стартовым узлом «persons.html» предикатом **«contacts»**, не сопоставляется никакая сущность. Для описания подобных узлов в RDF/XML предусмотрена возможность использования специального атрибута XML-элементов –

«rdf:nodeID». XML-текст, реализующий описание RDF-графа на рис. 18 приведен в примере 2 (рис.19).

```
# Пример 2
<?xml version="1.0"?>
# Описание пространств имен:
<rdf:RDF xmlns:rdf="http://www.etu.ru/rdf/syntax#"
xmlns:people="http://www.etu.ru/people/">
# Описание стартового узла:
<rdf:Description rdf:about = "http://www.etu.ru/people/persons.html">
# Свойство «Фамилия» для стартового узла:
<people:450>Alexey Ivanov</people:450>
# Абстрактное свойство «Контакты» для стартового узла:
<people:contacts rdf:nodeID="abc"/>
</rdf:Description>
# Описание пустого узла:
<rdf:Description rdf:nodeID="abc">
# Описание свойств пустого узла:
<people:mailbox rdf:resource="mailto:AI@etu.ru"/>
<people:phone>346-27-58</ people:phone>
</rdf:Description>
</rdf:RDF>
```

Рис. 19. Пример XML записи RDF графа с пустым узлом

В приведенном тексте атрибут **«rdf:nodeID»** сначала используется для присвоения фиктивного имени **"abc"** пустому узлу в элементе-узле **«rdf:Description»**, а затем – для ссылки на это имя в элементах-свойствах **«mailbox»** и **«phone»**. Сам идентификатор **"abc"** будет известен при этом только в пределах текущего RDF-документа.

Для определения расширенной области видимости такого идентификатора следует использовать альтернативный атрибут **«rdf:ID»**, который устанавливает «известность» имени в пределах пространства имен, определяемого так называемым базовым URI, т. е. URI стартового RDF-документа, ссылающегося на текущий. Если бы рассматриваемый в примере идентификатор **"abc"** был приведен в атрибуте **«rdf:ID»** при значении базового URI **«http://www.etu.ru/people»**, то он бы интерпретировался как **«http://www.etu.ru/people#abc»** (базовый URI может также явно задаваться в RDF-документе атрибутом **«xml:base=…»** заголовка элемента **«rdf:RDF»**).

Следует отметить, что необходимость в фиктивной идентификации возникает только в случае множественных подсвойств. Если бы подсвой-

ство **«mailbox»** свойства **«contacts»** отсутствовало бы, XML-фрагмент, описывающий соответствующий маршрут в исходном RDF-графе, мог бы выглядеть таким образом (рис.20):

```
<rdf:Description rdf:about = "http://www.etu.ru/people/persons.html">
<people:contacts>
<rdf:Description>
<people:phone>346-27-58</ people:phone>
</rdf:Description>
</people:contacts>
</rdf:Description>
```

Рис. 20. Пример XML записи RDF графа без пустого узла

Для приведенного XML-текста (т. е. при пустом узле и значении свойства «phone», выражаемом литералом) допускается и более лаконичное описание свойства внутри элемента **«rdf: Description»:**

<people:contacts people:phone="346-27-58"/>

Таким образом, если в заголовке элемента-узла отсутствуют **«rdf:about»** и **«rdf: nodeID»**, узел считается пустым. Если же из этого узла исходит несколько дуг RDF-графа, фиктивное именование узла (через атрибут **«rdf:nodeID»**) необходимо, так как иначе невозможно указать на тот факт, что через данный узел проходит несколько путей к подсвойствам.

В заключение следует отметить, что существует три основных способа описания субъектов и объектов в RDF-триплетах:

1) описание сущности с использованием атрибута **«rdf:about»** внутри элемента **«rdf:Description»** (т. е. элемента-узла);

2) описание сущности в свойстве другой сущности с использованием атрибута **«rdf:resource»** (если данная сущность соответствует узлу, не имеющему исходящих дуг; в этом случае свойство не имеет XML-элемента, соответствующего его значению, что отображается отсутствием каких-либо символов в описании закрывающей метки тэга, например в строке **<people:mailbox rdf:resource="mailto:AI@etu.ru"/>**);

3) описание пустого узла в элементе **«rdf:Description»**, который либо не имеет атрибута **«rdf:about»**, либо имеет атрибут **«rdf:nodeID»** (или **«rdf:ID»**).

1.4.2. Turtle - формат описания RDF графов

Turtle (Terse RDF Triple Language) - формат для сериализации графов RDF (модель описания ресурсов).

Синтаксис. URI в формате Turtle записываются в угловых скобках.

<http://dbpedia.org/resource/Leipzig>

Литералы заключаются в кавычки.

"Лейпциг" @ ru
"51.333332" ^ ^ xsd:float

В записях RDF моделей с использованием формата Turtle тройки (субъект – предикат – объект) разделяются точками.

<http://dbpedia.org/resource/Leipzig> <http://www.w3.org/2000/01/rdf-schema#label> "Leipzig"@de .
<http://dbpedia.org/resource/Leipzig>
<http://dbpedia.org/property/hasMayor >
<http://dbpedia.org/resource/Burkhard_Jung>.

Пробелы и переносы строк за пределами идентификаторов ресурсов игнорируются.

В Turtle для сокращения записей URI обычно используются префиксные объявления.

Например, **@prefix dbr: <http://dbpedia.org/resource/>.**

С учетом использования префиксов исходная запись

<http://dbpedia.org/resource/Leipzig> <http://www.w3.org/2000/01/rdf-schema#label> "Leipzig"@de .

может быть преобразована в запись следующего вида

@prefix dbr: <http://dbpedia.org/resource/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema> .
dbr:Leipzig rdfs:label "Leipzig"@de .

Вопросы для самопроверки

1.В утверждении "Окно покрашено белой краской" назовите части, которые относятся к субъекту, предикату и объекту.

2. Какая часть триплета при представлении его в виде графа изображается соединительной линией (дугой)?

3. Назовите основное отличие URI от IRI.

4.Может ли предикат быть литералом?

5. Какой из элементов триплета может быть литералом?

6. Какой из элементов триплета может быть пустым узлом?

7. Для какой цели используется пустой узел в RDF данных?

8. Для какой части триплета может быть использован дескриптор языка?

9. Какие типы контейнеров используются?

10.Чем отличается контейнер в RDF данных от коллекции?

11. Назовите основные форматы сериализации RDF данных.

2. Язык запросов SPARQL

Протокол SPARQL и язык запросов RDF – SPARQL (SPARQL Protocol and RDF Query Language), разработанный консорциумом W3C, определяет протокол для передачи запросов клиентов обработчикам запросов и язык формирования запросов на выборку и модификацию данных в RDF хранилищах данных. Рекомендации поязыку SPARQL были приняты консорциумом W3C в январе 2008 года.

2.1. Элементы языка SPARQL

Запрос SPARQL является текстовым документом в кодировке Unicode, либо в другой кодировке, поддерживаемым программным средством обработки запросов. Файлы с запросами SPARQL имеют расширение **.rq**. Значение MIME-типа данных для такого файла задается как **"application/sparql-query"**.

Для языка SPARQL определены следующие элементы:

1. IRI;
2. Уточненные имена;
3. Пробельные символы;
4. Литералы;
5. Переменные запроса;

6. Пустые узлы;
7. Шаблоны триплетов;
8. Базовые шаблоны графов;
9. Ключевые слова;
10. Комментарии.

1. В языке SPARQL используется интернационализированный идентификатор ресурса – IRI (Internationalized Resource Identifier), позволяющий, в отличие от универсального идентификатора ресурса – URI (Uniform Resource Identifier), задавать в идентификаторе не только латинские буквы, цифры и некоторое другие символы, но и символы кодировки Unicode (в частности, буквы кириллицы). В SPARQL определено следующее ограничение для IRI: в идентификаторе не должны использоваться символы

"<", ">", '"', пробел, "{", "}", "|", "\", "^" и **"`".**

В языке SPARQL IRI заключаются в угловые скобки "<" и ">", например:

<**http://purl.org/dc/elements/1.1/>.**

Ссылки IRI в запросе SPARQL могут быть как абсолютными, так и относительными.

2. Уточненные имена задаются в виде: префикс-пространства-имен:локальное-имя, например, **dc:creator**.

Если для локального имени задано пространство имен по умолчанию, то префикс пространства имен не указывается, например, **:name**.

3. В SPARQL используются следующие пробельные символы: символ пробела (код "20"), символ горизонтальной табуляции (код "09"), символ возврата каретки (код "0D") и символ перехода на новую строку (код "0A").

4. Литералы в SPARQL заключаются в двойные или одиночные апострофы, за которыми может следовать метка языка, образуемая символом **"@"**, за которым задается код языка.

Примеры литералов: **"abcd"**, **'Строка 1'@ru**, **"String 1"@en**.

Если литералы содержат апострофы или символ перевода на новую строку, то они заключаются в три двойных или одиночных апострофа, например:

'"Символ '<' задает начало IRI а символ '>' – его окончание.'",

. В литералах можно задавать следующие специальные символы:

'\t' – горизонтальная табуляция (код "U+0009");

'\n' – переход на новую строку (код "U+000A");

'\n' – возврат каретки (код "U+000D");

'\b' – удаление предыдущего символа (код "U+0008");

'\f' – переход к новой странице (код "U+000C");

'\"' – двойной апостроф (код "U+0022");

"\'" – одиночный апостроф (код "U+0027");

'\\' – обратный слеш (код "U+005C").

В SPARQL можно также задавать типизированные литералы, т.е. литералы, содержащие данные определенного типа. Для этого после литерала вставляются символы "^^", после которых либо задается ссылка на IRI для типа данных, либо задается уточненное имя типа данных, например:

"ТИ-81"^^<http://www.someInstitute.edu/ns/groupDatatype>

или

"2008-09-01"^^xsd:date.

В SPARQL в качестве типов литералов можно использовать следующие типы схемы XML: **xsd:integer, xsd:decimal, xsd:float, xsd:double, xsd:string, xsd:boolean, xsd:dateTime и xsd:date.**

Кроме этого, при вызове функций и выполнении операторов в аргументах функций можно использовать следующие числовые типы схемы XML: **xsd:nonPositiveInteger, xsd:negativeInteger, xsd:long, xsd:int, xsd:short, xsd:byte, xsd:nonNegativeInteger, xsd:unsignedLong, xsd:unsignedInt, xsd:unsignedShort, xsd:unsignedByte** и **xsd:positiveInteger.**

Для целых чисел (тип **xsd:integer**), чисел с десятичной точкой (тип **xsd:decimal**), чисел в экспоненциальной форме (тип **xsd:double**), а также булевских значений **true or false** (тип **xsd:boolean**) можно просто задать значение литерала без указания типа и без ограничивающих апострофов, например, **15, 2.72, 4.7e-3** или **true.**

В SPARQL определен также нулевой литерал (ресурс **rdf:nil**), который в сокращенной форме записывается как "( )".

5. Переменные запроса задаются в SPARQL как идентификаторы XML, перед которыми задается префикс **"$"** или префикс "**?**", например **$x1** или **?x1** (это одна и та же переменная). Переменные в SPARQL имеют глобальную область видимости, т.е. доступны в любом месте запроса.

6. Пустые узлы задаются в следующем виде **_:метка-узла**, например,

_:emptyNode1 dc:creator "Иванов И.И."^^xsd:token .

Если пустой узел в запросе один, его метку можно не указывать, заключив предикат и объект в квадратные скобки, например,

[dc:creator "Иванов И.И."^^xsd:token] .,

либо указав скобки "**[]**" перед предикатом и объектом, например,

[] dc:creator "Иванов И.И."^^xsd:token ..

Пустой узел без метки может задаваться как субъект следующего шаблона триплета, например:

[**dc:creator "Иванов И.И."^^xsd:token**]

dc:contributor "Петров П.П."^^xsd:token .,

что эквивалентно следующим двум триплетам:

_:emptyNode1 dc:creator "Иванов И.И."^^xsd:token

_:emptyNode1 dc:contributor "Петров П.П."^^xsd:token .

Пустой узел без метки может также задаваться как объект предыдущего шаблона триплета, например:

prog:statProg dc:subject

[dc:creator "Иванов И.И."^^xsd:token] .,

что эквивалентно следующим двум триплетам:

prog:statProg dc:subject _:emptyNode1

_:emptyNode1 dc:creator "Иванов И.И."^^xsd:token .

7. В SPARQL используются следующие пробельные символы: символ пробела (код U+0020"), символ горизонтальной табуляции (код "U+0009"), символ возврата каретки(код "U+000D") и символ перехода на новую строку (код "U+000A").

8. Шаблоны триплетов состоят из трех компонент: субъекта, предиката и объекта.

Субъект триплета может быть либо переменной, либо литералом, либо пустым узлом, либо ссылкой на IRI (IRIref).

Предикат триплета может быть либо переменной, либо ссылкой на IRI (IRIref).

Объект триплета может быть либо переменной, либо литералом, либо пустым узлом, либо ссылкой на IRI (IRIref).

При записи шаблонов триплетов можно использовать следующие сокращения:

1. Если несколько триплетов имеют общий субъект, то первый триплет

записывается полностью, а для остальных триплетов записывается только предикат и объект, при этом триплеты отделяются друг от друга точкой с запятой (символ **";"**);

2. Если несколько триплетов имеют общий субъект и предикат, то первый триплет записывается полностью, а для остальных триплетов записывается только объект при этом триплеты отделяются друг от друга запятой (символ ",");

3. Коллекции RDF можно записать в форме списка элементов коллекции, заключенного в круглые скобки ("(" и ")"), при этом элементы

коллекции отделяются друг от друга пробелами, а элементы коллекции также могут быть коллекциями;

4. Вместо предиката **rdf:type** можно использовать сокращение **a**.

. Ниже приведены примеры использования сокращений при записи шаблонов триплетов:

1. При использовании сокращения типа 1 триплеты

prog:statProg dc:creator "Иванов И.И."^^xsd:sting.
prog:statProg dc:contributor "Петров П.П."^^xsd:sting ..

можно записать следующим образом:

prog:statProg dc:creator "Иванов И.И."^^xsd:sting;
dc:contributor "Петров П.П."^^xsd:sting ..

2. При использовании сокращения типа 2 триплеты

prog:statProg dc:creator "Иванов И.И."^^xsd:sting .
prog:statProg dc:creator "Сидоров С.С."^^xsd:sting ..

можно записать следующим образом:

prog:statProg dc:creator "Иванов И.И."^^xsd:sting,
"Петров П.П."^^xsd:sting ..

3. При использовании сокращения типа 3 триплеты

_:n0 rdf:first 1 ;
rdf:rest _:n1 .
_:n1 rdf:first _:n2 .
_:n2 const:two 2 .
_:n1 rdf:rest _:n3 .
_:n3 rdf:first _:n4 .
_:n4 rdf:first 3 ;
rdf:rest _:n5 .
_:n5 rdf:first 4 ;
rdf:rest rdf:nil .
_:n3 rdf:rest rdf:nil .

можно записать следующим образом:

(1 [const:two 2] (3 4)).

4. При использовании сокращения типа 4 триплет

?x1 rdf:type :MyClass

можно записать следующим образом:

?x1 a :MyClass.

8. Базовый шаблон графа запросов формируется с использованием набора шаблонов триплетов.

9. В SPARQL определены следующие ключевые слова (табл.1):

Таблица 1

Ключевые слова языка запросов SPARQL

BASE	**SELECT**	**ORDER BY**	**FROM**	**GRAPH**	**STR**	**isURI**
PREFIX	**CONSTRUCT**	**LIMIT**	**FROM NAMED**	**OPTIONAL**	**LANG**	**isIRI**
	DESCRIBE	**OFFSET**	**WHERE**	**UNION**	**LANGMATCHES**	**isLiteral**
	ASK	**DISTINCT**		**FILTER**	**DATATYPE**	**REGEX**
		REDUCED		**a**	**BOUND**	**true**
					sameTERM	**false**

Ключевые слова, за исключением **a**, могут задаваться в любом регистре, т.е. как прописными, так и строчными буквами.

Назначение указанных в табл.1 ключевых слов будет описано ниже.

10. Комментарии в SPARQL начинаются с символа **"#"** и заканчиваются концом строки.

2.2. Общая структура SPARQL запроса

Запрос SPARQL имеет следующую структуру:

пролог,

запрос-SPARQL.

Пролог, если он есть в документе, содержит одно или несколько предложений объявлений префиксов для пространств имен, а также может содержать одно предложение объявления базового IRI.

Объявление префиксов для пространств имен сопоставляет префикс заданному пространству имен и имеет следующий синтаксис:

PREFIX префикс: <IRI>, например,

PREFIX dc: <http://purl.org/dc/elements/1.1/>.

Если параметр "префикс" не задан, объявление задает префикс пространства имен, используемый по умолчанию, например:

PREFIX : <http://www.someInstitute.edu/groupId#>.

Объявление базового IRI определяет IRI, относительно которого будут задаваться IRI в документе, например,

BASE <http://www.someInstitute.edu/>.

Запрос-SPARQL может быть одним из следующих видов запросов:

- запрос **SELECT;**
- запрос **CONSTRUCT**;
- запрос **ASK**;
- запрос **DESCRIB**E.

Исходными данными, которые обрабатывает запрос (файл с расширением **.rq**), являются файлы документов RDF, RDFS или OWL, записанные с использованием либо нотации N3 (файлы с расширением .n3), либо нотации Turtle (файлы с расширением .ttl), либо на языке RDF/XML (файлы с расширениями **.rdf** или **.owl**). Общая структура организации обработки данных запросом приведена на рис. 21.

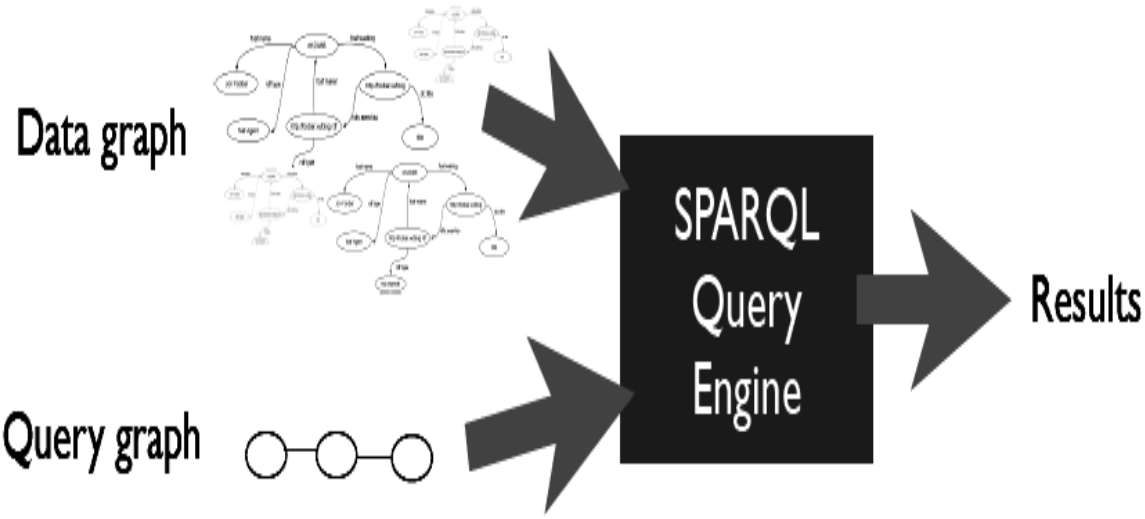


Рис. 21. Общая структура организации обработки данных запросами SPARQL

Форма вывода результата запроса полностью определяется видом запроса.

2.3. Запрос SELECT

Общая структура запроса SELECT имеет следующий вид (рис.22).

В запросе обязательно задается список переменных, состоящий либо из одной, либо нескольких переменных.

Вместо списка переменных запрос может содержать символ "*" (все переменные запроса). Кроме того в запросе могут быть указаны исходные данные которые обрабатываются запросом.

```
PREFIX  foo: <http://example.com/resources/>
# префиксные объявления (пролог)
FROM   …
# источники запроса
SELECT   …
# состав ответа на запрос (список переменных)
WHERE { …}
# шаблон запроса
ORDER BY   …
# модификаторы запроса
```

Рис. 22. Общая структура запроса типа SELECT

Для переменных, заданных в списке, либо для всех переменных запроса (если задан символ **"*"**) в результате обработки данных формируется набор соответствий переменных запроса со связанными с ними ресурсами исходных данных.

Результат запроса может быть выведен либо в текстовом виде, либо в формате XML.

При выводе в текстовом виде результат представляется в виде таблицы соответствия, в которой в заголовке выводятся имена переменных запроса (без символа**"?"** или **"$"** в начале имени), а в столбцах выводятся значения связанных с этими переменными ресурсов. Количество строк для переменной равно количеству найденных соответствий (для разных переменных количество строк может быть разным). Если для какой-либо переменной соответствий не найдено, количество строк для нее равно 0.

При выводе в формате XML таблица соответствия выводится в виде документа XML, структура которого представлена на рис. 23.

Как видно из приведенной структуры, корневым элементом документа является элемент **sparql**, в котором задана ссылка на пространство имен элементов и атрибутов документа. В элемент **sparql** вложены элементы **head** (заголовок) и **results** (результаты). В атрибутах **name** (имя) вложенных в **head** элементов **variable** (переменная) задаются имена всех переменных.

В **head** могут быть также вложены элементы **link** (связь), в атрибуте **href** (ссылка) которых задаются ссылки на файлы, содержащие дополнительные метаданные о результатах запроса. Элементы **link** могут быть заданы после любого элемента **variable**.

Вложенные в элемент results элементы **result** (результат) описывают строки таблицы. Во вложенных в элемент элементах **binding** (связывание) в атрибуте **name** (имя) задается имя переменной, соответствующее значение которой задается в содержимом элемента **binding.**

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
<head>
<variable name="имя-переменной-1"/>
<variable name="имя-переменной-2"/>
…
<variable name="имя-переменной-n"/>
<link href="имя-файла"/>
</head>
<results>
<result>
<binding name="имя-переменной-1">значение-1-1</binding>
<binding name="имя-переменной-2">значение-1-2</binding>
…
<binding name="имя-переменной-n">значение-1-n</binding>
</result>
<result>
<binding name="имя-переменной-1">значение-2-1</binding>
<binding name="имя-переменной-2">значение-2-2</binding>
…
<binding name="имя-переменной-n">значение-2-n</binding>
</result>
…
<result>
<binding name="имя-переменной-1">значение-m-1</binding>
<binding name="имя-переменной-2">значение-m-2</binding>
…
<binding name="имя-переменной-n">значение-m-n</binding>
</result>
</results>
</sparql>
```

Рис. 23. Общая структура ответа на запрос в виде XML документа

В зависимости от типа значения в содержимом элемента **binding** задаются следующие элементы:

- **<uri>**значение-URI**</uri>** – содержимым является URI;
- **<literal>**литерал**</literal>** – содержимым является литерал;
- **<literal** xml:lang="код-языка">литерал**</literal>** – содержимым является литерал на заданном языке;
- **<literal** datatype="тип-данного">литерал**</literal>** – содержимым является литерал заданного типа (типизированный литерал);

- <bnode>метка-узла**</bnode>** – содержимым является метка пустого узла.

2.4. Шаблон запроса

Условие поиска соответствия в запросе задается шаблоном группы графов:

WHERE *шаблон-группы-графов,*

где *шаблон-группы-графов* заключен в фигурные скобки ("{" и "}") и может содержать следующие компоненты:

- пустой шаблон;
- базовый шаблон графа;
- вложенные шаблоны группы графов;
- дополнительный шаблон группы графов;
- объединение шаблонов группы графов;
- шаблон **GRAPH**;
- фильтр.

Ключевое слово **WHERE** в условии может быть пропущено.

Пустой шаблон соответствует любому графу, включая пустой граф, с только одним решением, не связывающим переменную запроса.

Например, запрос

SELECT $book
{}

выведет следующую таблицу результата (рис. 24):

Рис. 24. Результат выполнения запроса с пустым графом

Рассмотрим примеры построения SPARQL запросов к данным представленным в исходном документе в нотации Turtle (Рис. 27).

Запрос-1. Запрос -1 должен найти и вывести в таблицу ответов для экземпляра книги book5 название книги (переменная- **bookTitle**) и автора (переменная- **bookAuthor**) имеет вид, представленный на рис. 25 .

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?bookTitle ?bookAuthor
WHERE {
:book5 dc:creator ?bookAuthor;
dc:title ?bookTitle;
dc:type ?bookGenre .
}
```

Рис. 25 . Запрос - 1

Результат выполнения запроса к исходному документу в текстовом представлении имеет следующий вид (рис.26) и представляет найденные связи для указанных переменных при фиксированном субъекте.

bookTitle	bookAuthor
"Приключения Робинзона Крузо"@ru	"Дефо Даниэл"@ru

Рис. 26. Результат выполнения Запроса -1

Запрос-2. Запрос-2 должен найти и вывести в таблицу ответов для экземпляра книги book1 название книги (переменная- **bookTitle**), автора (переменная- **bookAuthor**) и жанр произведения (переменная- **bookGenre**). Текст запроса представлен на рис. 28.

В разделе SELECT запроса нет перечисления переменных подлежащих выводу в ответе. Символ "*" показывает, что ответ на запрос будет включать значения всех переменных, которые использовались при построении графа запроса.

На рис. 29 представлены найденные связи для всех переменных , которые использовались при построении запроса-2 (в том порядке, в котором они использовались) при фиксированном субъекте.

Запрос - 3 должен найти и вывести в таблицу ответов все экземпляры книг, описанных в исходном документе (переменная- **book)** и указать наименование этих книг (переменная- **bookTitle**). Текст запроса представлен рис. 30.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix : <http://www.libRegistry.org/ns/lib#> .
        # Экземпляр book1 класса Book
:book1 a <http://www.libRegistry.org/ns/lib#Book>;
        # Идентификатор ISBN книги
dc:identifier "02419-0";
        # Название книги
dc:title "Десять негритят"@ru;
        # Автор книги
dc:creator "Кристи Агата"@ru;
        # Жанр книги
dc:type "детектив"^^xsd:string;
        # Издатель книги
dc:publisher "Эксмо"@ru;
        # Дата издания книги
dc:date "2003-03-09"^^xsd:date .
        # Экземпляр book2 класса Book
:book2 a <http://www.libRegistry.org/ns/lib#Book>;
dc:identifier "966-00-0785-5";
dc:title "Кобзар"@uk;
dc:creator "Шевченко Тарас"@uk;
dc:type "классика"^^xsd:string;
dc:publisher "Наукова думка"@uk;
dc:date "2002-10-16"^^xsd:date .
        # Экземпляр book3 класса Book
:book3 a <http://www.libRegistry.org/ns/lib#Book>;
dc:identifier "0-14-062015-3";
dc:title "Robinson Crusoe"@en;
dc:creator "Defoe Daniel"@en;
dc:type "классика"^^xsd:string;
dc:publisher "Penguin"@en;
dc:date "2005-06-19"^^xsd:date .
        # Экземпляр book4 класса Book
:book4 a <http://www.libRegistry.org/ns/lib#Book>;
dc:identifier "5-367-00019-3";
dc:title "Убийство в доме викария"@ru;
dc:creator "Кристи Агата"@ru;
dc:type "детектив"^^xsd:string;
dc:publisher "Амфора"@ru;
dc:date "2006-01-28"^^xsd:date .
        # Экземпляр book5 класса Book
:book5 a <http://www.libRegistry.org/ns/lib#Book>;
dc:identifier "966-7047-42-3";
dc:title "Приключения Робинзона Крузо"@ru;
dc:creator "Дефо Даниэл"@ru;
dc:type "классика"^^xsd:string;
dc:publisher "Эксмо"@ru;
dc:date "2007-03-28"^^xsd:date .
```

Рис. 27. Исходный документ в формате Turtle

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT *
WHERE {
:book1 dc:creator ?bookAuthor;
dc:title ?bookTitle;
dc:type ?bookGenre .
}
```

Рис. 28. Запрос - 2

bookAuthor	bookTitle	bookGenre
"Кристи Агата"@ru	"Десять негритят"@ru	"детектив"^^xsd:string

Рис. 29 . Результат выполнения Запроса - 2

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?book ?bookTitle
WHERE {
?book dc:title ?bookTitle .
}
```

Рис. 30. Запрос - 3

Результат выполнения Запроса – 3 представлен на рис. 31,

где представлены найденные связи для заданных переменных для субъекта и объекта при фиксированном значении предиката.

book	bookTitle
:book5	"Приключения Робинзона Крузо"@ru
:book4	"Убийство в доме викария"@ru
:book3	"Robinson Crusoe"@en
:book2	"Кобзар"@uk
:book1	"Десять негритят"@ru

Рис. 31 . Результат выполнения Запроса-3

Запрос - 4 должен найти и вывести в таблицу ответов обозначения (переменная - **bookCha**r)и значения (переменная- **bookValue**).параметров книги 1 , описанной в исходном документе. Текст запроса представлен рис. 32. Результат выполнения запроса показан на рис. 33.

В таблице ответа представлены найденные связи для всех предикатов и объектов фиксированного субъекта.

Запрос - 5 должен найти и вывести в таблицу ответов книги автора " Кристи Агата " @ru. Текст запроса представлен рис. 33. . Результат выполнения запроса показан на рис. 34.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?book ?bookTitle
WHERE {
?book dc:title ?bookTitle .
}
```

Рис. 32. Запрос - 4

bookChar	bookValue
dc:date	"2003-03-09"^^xsd:date
dc:publisher	"Эксмо"@ru
dc:type	"детектив"^^xsd:string
dc:creator	"Кристи Агата"@ru
dc:title	"Десять негритят"@ru
dc:identifier	"02419-0"
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	:Book

Рис. 33. Результат выполнения Запроса - 4

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?book ?bookTitle
WHERE {
?book dc:creator "Кристи Агата"@ru;
dc:title ?bookTitle .
}
```

Рис. 34. Запрос - 5

book	bookTitle
<http://www.libRegistry.org/ns/lib#book4>	"Убийство в доме викария"@ru
<http://www.libRegistry.org/ns/lib#book1>	"Десять негритят"@ru

Рис. 35. Результат выполнения Запроса - 5

В таблице представлены найденные связи для заданных в запросе переменных и фиксированного объекта "Кристи Агата"@ru.

Запрос - 6 должен найти и вывести в таблицу ответов перечень названий книг с указанием их типа. Текст запроса представлен рис. 36. . Результат выполнения запроса показан на рис. 37. В таблице ответа редставлены найденные связи для заданных в запросе переменных при фиксированных предикатах.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?book ?bookTitle ?bookType
WHERE {
?book dc:title ?bookTitle .
?book dc:type ?bookType .
}
```

Рис. 36. Запрос - 6

book	bookTitle	bookType
:book5	"Приключения Робинзона Крузо"@ru	"классика"^^xsd:string
:book4	"Убийство в доме викария"@ru	"детектив"^^xsd:string
:book3	"Robinson Crusoe"@en	"классика"^^xsd:string
:book2	"Кобзар"@uk	"классика"^^xsd:string
:book1	"Десять негритят"@ru	"детектив"^^xsd:string

Рис. 37. Результат выполнения Запроса - 6

2.5. Использование вложенных и дополнительных шаблонов группы графов в SELECT запросах

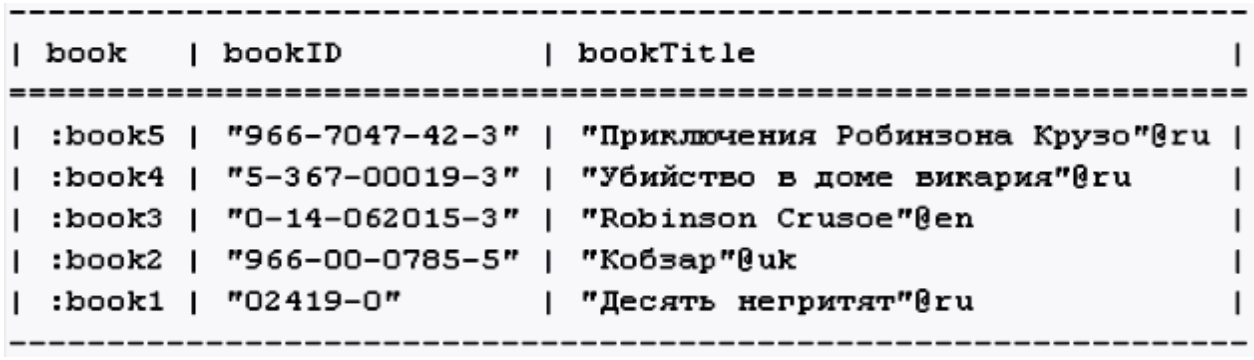
Вложенные шаблоны группы графов содержат внутри условия одну или несколько заключенных в фигурные скобки групп графов.

Пример использования вложенного шаблона группы графов в условии запроса SELECT представлен на рис. 38.

На рис.39 представлен ответ на запрос – 6, который отражает Все найденные связи для заданных в запросе переменных в двух вложенных шаблонов групп графов при фиксированных предикатах.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?book ?bookID ?bookTitle
WHERE {
{?book dc:identifier ?bookID .}
{?book dc:title ?bookTitle .}
}
```

Рис. 38. Запрос - 6



book	bookID	bookTitle
:book5	"966-7047-42-3"	"Приключения Робинзона Крузо"@ru
:book4	"5-367-00019-3"	"Убийство в доме викария"@ru
:book3	"0-14-062015-3"	"Robinson Crusoe"@en
:book2	"966-00-0785-5"	"Кобзар"@uk
:book1	"02419-0"	"Десять негритят"@ru

Рис. 39. Результат выполнения Запроса - 6

Запросы с использованием базовых шаблонов графов выводят связи переменных только в том случае, если триплет полностью соответствует указанному в запросе шаблону.

Однако в некоторых случаях желательно вывести связь для переменной, даже если триплет частично соответствует шаблону в запросе.

Для решения этой задачи используется дополнительный шаблон группы графов, который имеет следующий вид:

шаблон **OPTIONAL** {***шаблон***}

Рассмотрим пример использования дополнительного шаблона графов в SELECT-запросе.

Если в исходном документе (рис. 27) закомментировать строку

dc:type "детектив"^^xsd:string;

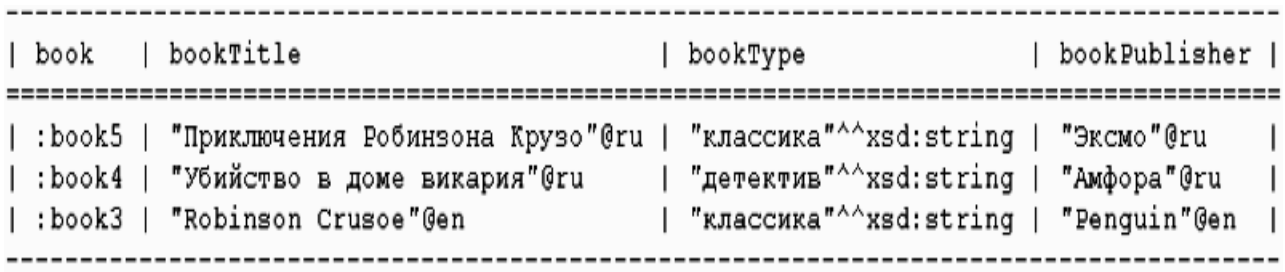
для экземпляра **book1** и строку

dc:publisher "Наукова думка"@uk;

для экземпляра book2, то запрос – 7, представленный на рис. 40 выведет результат который в текстовом представлении имеет следующий вид (рис. 41).

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?book ?bookTitle ?bookType
WHERE {
?book dc:title ?bookTitle;
dc:type ?bookType;
dc:publisher ?bookPublisher .
}
```

Рис. 40. Запрос - 7



book	bookTitle	bookType	bookPublisher
:book5	"Приключения Робинзона Крузо"@ru	"классика"^^xsd:string	"Эксмо"@ru
:book4	"Убийство в доме викария"@ru	"детектив"^^xsd:string	"Амфора"@ru
:book3	"Robinson Crusoe"@en	"классика"^^xsd:string	"Penguin"@en

Рис. 41. Результат выполнения Запроса - 7

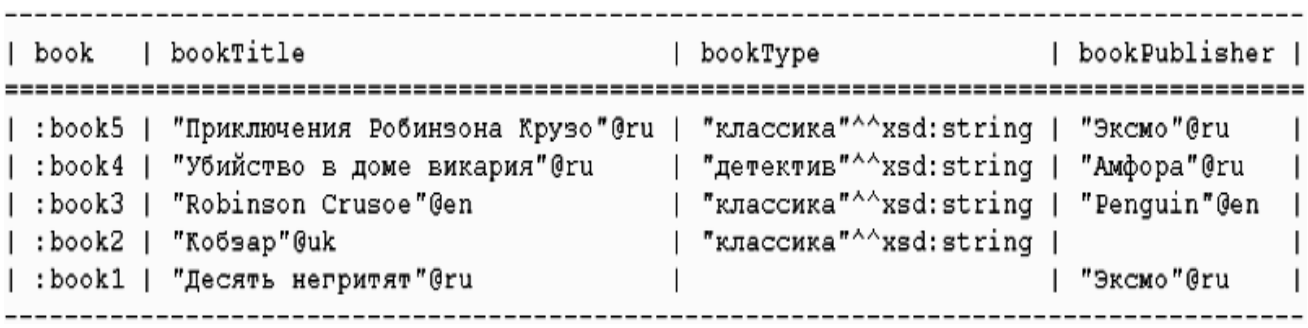
Как показывает результат работы запроса, экземпляры book1 и book2 не вошли в результирующую таблицу, поскольку они не полностью соответствуют шаблону запроса.

Если все же необходимо учесть эти экземпляры в результирующей таблице, то необходимо задать отсутствующие в них триплеты в дополнительных шаблонах группы графов, изменив запрос следующим образом (рис. 42).

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?book ?bookTitle ?bookType ?bookPublisher
WHERE {
?book dc:title ?bookTitle .
OPTIONAL {?book dc:type ?bookType}
OPTIONAL {?book dc:publisher ?bookPublisher}
}
```

Рис. 42. Запрос - 8

В этом случае будет получен следующий результат (рис. 43).



book	bookTitle	bookType	bookPublisher
:book5	"Приключения Робинзона Крузо"@ru	"классика"^^xsd:string	"Эксмо"@ru
:book4	"Убийство в доме викария"@ru	"детектив"^^xsd:string	"Амфора"@ru
:book3	"Robinson Crusoe"@en	"классика"^^xsd:string	"Penguin"@en
:book2	"Кобзар"@uk	"классика"^^xsd:string	
:book1	"Десять негритят"@ru		"Эксмо"@ru

Рис. 43. Результат выполнения Запроса - 8

Из таблицы ответа видно, что ячейки таблицы, для которых не найдены связи, выводятся пустыми.

Аналогичную задачу решает и способ объединения шаблонов группы графов. В этом случае появлятся возможность получить соответствие шаблону графа запроса в том случае, если при обработке запроса получено соответствие хотя бы для одного из заданных альтернативных шаблонов.

Для задания альтернативных шаблонов используется следующий способ:

{шаблон-1} **UNION** ***{шаблон-2}*** **UNION {шаблон-3} …** .

Заданные в объединении шаблоны последовательно проверяются на соответствие до тех пор, пока один из шаблонов не соответствует триплету исходного документа, либо пока последовательность шаблонов не завершится.

Рассмотрим пример использования объединения шаблонов графов в SELECT-запросе.

В исходном документе (рис. 27) используется версия, заданная в пространстве имен **http://purl.org/dc/elements/1.1/** с префиксом **dc**.

Предположим, что экземпляр **book1** задан с использованием версии **Dublin Core 1.0** в пространстве имен **http://purl.org/dc/elements/1.0/.**

Добавим в прологе документа предложение

@prefix dc10: <http://purl.org/dc/elements/1.0/> и заменим все префиксы **dc** для экземпляра **book1** префиксами **dc10**.

Тогда результат обработки запроса (рис. 44) не будет содержать связей для экземпляра book1.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?book ?bookTitle ?bookPubDate
WHERE {
?book dc:title ?bookTitle;
dc:date ?bookPubDate .
}
```

Рис. 44. Запрос - 9

Однако, если задать запрос с использованием альтернативных шаблонов (рис. 45),

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?book ?bookTitle ?bookPubDate
WHERE {
{{?book dc:title ?bookTitle}
UNION {?book dc10:title ?bookTitle}}
{{?book dc:date ?bookPubDate} UNION
{?book dc10:date ?bookPubDate}}
}
```

Рис. 45. Запрос - 10

то в таблицу результата будут включены и связи для экземпляра book1(рис.46).

book	bookTitle	bookPubDate
:book5	"Приключения Робинзона Крузо"@ru	"2007-03-28"^^xsd:date
:book4	"Убийство в доме викария"@ru	"2006-01-28"^^xsd:date
:book3	"Robinson Crusoe"@en	"2005-06-19"^^xsd:date
:book2	"Кобзар"@uk	"2002-10-16"^^xsd:date
:book1	"Десять негритят"@ru	"2003-03-09"^^xsd:date

Рис. 46. Результат выполнения Запроса - 10

2.6. Использование фильтров

Фильтр определяет дополнительные ограничения на выводимые в ответе данные и реализуется следующим образомс
:

FILTER ***ограничения***.

Параметр ***ограничения*** может принимать одну из следующих форм:

- **выражение;**
- **вызов встроенной функции;**
- **вызов внешней функции**.

Выражение заключается в круглые скобки "(" и ")" и может содержать в качестве компонент переменные, литералы, ссылки на IRI (IRIref), уточненные имена, вызовы встроенных функций, а также другие выражения.

В выражениях можно использовать операции отношения: **равно** ("="), **не равно** ("!="), **больше** (">"), **меньше** ("<"), **больше или равно** (">="), **меньше или равно** ("<="), а также логические операции: **И** ("&&"), **ИЛИ** ("||") и **НЕ** ("!").

Для числовых литералов определены также одноместные арифметические операции: **плюс** ("+") и **минус** ("-"), а также двуместные арифметические операции: **сложение** ("+"), **вычитание** ("-"), **умножение** ("*") и **деление** ("/").

В языке **SPARQL** определены встроенные функции следующих типов:

- функции проверки;
- функция преобразования;
- функция определения типа данных;
- языковые функции;
- функции выполнения логических операций;
- функция поиска с использованием регулярного выражения.

К функциям проверки относятся следующие функции: **bound()**, **isIRI()**,**isURI(), isBlank(), isLiteral(),RDFterm-equal() и sameTerm().**

Функция **bound(переменная)** проверяет, является ли переменная, заданная в качестве аргумента функции, связанной. Если это условие выполняется, функция возвращает true, иначе возвращает false.

Функции **isIRI(аргумент)** и **isURI(аргумент)** проверяют, является ли

аргумент **IRI или URI**. Если эти условия выполняются, функции возвращают true, иначе возвращают false.

В качестве аргумента могут быть заданы **IRI**, **URI, литерал**, **переменная** или **пустой узел.**

Функция **isBlank(аргумент)** проверяет, является ли аргумент пустым узлом. Если это условие выполняется, функция возвращает true, иначе возвращает false. В качестве аргумента могут быть заданы **IRI**, **URI**, **литерал**, **переменная** или **пустой узел**.

Функция **isLiteral(аргумент)** проверяет, является ли аргумент литералом. Если это условие выполняется, функция возвращает true, иначе возвращает false. В В качестве аргумента могут быть заданы **IRI**, **URI**, **литерал**, **переменная** или **пустой узел**.

Функция **RDFterm-equal(аргумент-1, аргумент-2)** возвращает true, если оба аргумента являются эквивалентными, иначе возвращает false. В

качестве аргументов могут быть заданы **литералы, переменные, пустые узлы IRI** или **URI**.

Два литерала являются эквивалентными при выполнении следующих условий: они посимвольно равны; у обоих либо есть, либо нет меток языков; если метки языков есть, то они одинаковые; у обоих либо есть, либо нет типов данных. Два пустых узла считаются эквивалентными, если они указывают на один и тот же узел. Два IRI или URI считаются эквивалентными, если они посимвольно равны.

Функция **sameTerm(аргумент-1, аргумент-2)** возвращает true, если оба аргумента являются одним и тем же данным, иначе возвращает false. В качестве аргументов могут быть заданы **литералы, переменные, пустые узлы IRI** или **URI**.

Функция преобразования **str(аргумент)** возвращает строковое представление аргумента. В качестве аргумента могут быть заданы **литерал, переменная, IRI** или **URI**.

Функция определения типа **datatype(аргумент)** возвращает уточненный тип данных схемы XML для аргумента. Если аргумент задает простые (нетипизированные) данные, то возвращается тип данных **xsd:string.** В качестве аргумента могут быть заданы **простой** или **типизированный литералы** или **переменная**.

К языковым функциям относятся функции **lang()** и **langMatches()**.

Функция **lang(аргумент)** возвращает строку кода языка аргумента. Если код языка для аргумента не задан, возвращается пустая строка. В качестве аргумента могут быть заданы **литерал** или **переменная**.

Функция **langMatches(метка-языка, диапазон-меток)** возвращает true, если метка-языка входит в диапазон-меток, заданный во втором аргументе, иначе возвращает false. Аргумент диапазон-меток содержит либо список меток языков, отделенных друг от друга символом "-", либо символ "*".

Логические функции **logical-or() и logical-and()**выполняют логические операции над аргументами.

Функция **logical-or(аргумент-1, аргумент-2)** возвращает результат выполнения операции **аргумент-1 || аргумент-2** (true или false). Оба аргумента должны иметь булевский тип.

Функция **logical-and(аргумент-1, аргумент-2)** возвращает результат выполнения операции **аргумент-1 && аргумент-2** (true или false). Обааргумента должны иметь булевский тип.

Функция поиска с использованием регулярного выражения **regex(строка, шаблон, флажки)** вызывает функцию **fn:matches()языка XPath 2.0**. В аргументе **строка** задается строковый литерал или строковую переменную, а в аргументе **шаблон** – шаблон поиска в строке (в

соответствии с правилами задания регулярных выражений). В необязательном аргументе задаются **флажки "s", "m", "i" и "x"** (один флажок или строка, содержащая комбинацию флажков). Функция возвращает true, если в строке найдено соответствие шаблону и false – в противном случае.

В фильтре могут быть заданы вызовы внешних функций. Так, можно использовать функции языка **XPath 2.0**, если в прологе задать предложение **PREFIX fn: <http://www.w3.org/2005/xpath-functions#>.** После этого можно вызвать необходимую функцию XPath 2.0 в фильтре, например:

FILTER fn:starts-with(&name,"И")

Аналогичным образом можно задавать и вызывать другие функции (функции расширения)

Рассмотрим пример использования фильтров в запросах SELECT.

Результат обработки запроса (рис. 47) к исходному документу (рис. 27) в текстовом виде имеет следующий вид (рис.48).

В таблице выведены авторы и наименования книг жанра "классика".

Для поиска и вывода в таблицу авторов и наименований книг, у которых в написании автора книги встречается строка "Defoe"или строка "Дефо" можно использовать следующий запрос (рис. 47).

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?bookAuthor ?bookTitle
WHERE {
?book dc:title ?bookTitle;
dc:creator ?bookAuthor;
dc:type ?bookGenre .
FILTER (?bookGenre = "классика"^^xsd:string)
}
```

Рис. 47. Запрос - 11

```
-----------------------------------------------------------
| bookAuthor           | bookTitle                        |
===========================================================
| "Дефо Даниэл"@ru     | "Приключения Робинзона Крузо"@ru |
| "Defoe Daniel"@en    | "Robinson Crusoe"@en             |
| "Шевченко Тарас"@uk  | "Кобзар"@uk                      |
-----------------------------------------------------------
```

Рис. 48. Результат выполнения Запроса – 11

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?bookAuthor ?bookTitle
WHERE {
?book dc:title ?bookTitle;
dc:creator ?bookAuthor .
FILTER (regex(str(?bookAuthor), "Defoe") ||
regex(str(?bookAuthor), "Дефо"))
}
```

Рис. 49. Запрос - 12

Результат обработки запроса представлен на рис. 50.

Для того, чтобы вывести в таблицу авторов и наименования книг на украинском языке используем следующий запрос (рис. 51).

Результат обработки этого запроса представлен на рис. 50.

bookAuthor	bookTitle
"Дефо Даниэл"@ru	"Приключения Робинзона Крузо"@ru
"Defoe Daniel"@en	"Robinson Crusoe"@en

Рис. 50. Результат выполнения Запроса – 12

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?bookAuthor ?bookTitle
WHERE {
?book dc:title ?bookTitle;
dc:creator ?bookAuthor .
FILTER (lang(?bookTitle) = "uk")
}
```

Рис. 51. Запрос - 13

bookAuthor	bookTitle
"Шевченко Тарас"@uk	"Кобзар"@uk

Рис. 52. Результат выполнения Запроса - 13

Запрос (рис. 53) выводит авторов, наименования книг и даты издания для книг издательства Эксмо, выпущенных после 2004 года. Результат обработки запроса представлен на рис. 54.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?bookAuthor ?bookTitle ?bookPubDate
WHERE {
?book dc:title ?bookTitle;
dc:creator ?bookAuthor;
dc:publisher ?bookPublisher;
dc:date ?bookPubDate .
FILTER (?bookPublisher = "Эксмо"@ru &&
?bookPubDate > "2005-01-01"^^xsd:date)
}
```

Рис. 53. Запрос - 14

```
-------------------------------------------------------------------------------
| bookAuthor       | bookTitle                        | bookPubDate           |
===============================================================================
| "Дефо Даниэл"@ru | "Приключения Робинзона Крузо"@ru | "2007-03-28"^^xsd:date |
-------------------------------------------------------------------------------
```

Рис. 54. Результат выполнения Запроса - 14

2.7. Определение исходных документов

Исходными данными для запроса на языке SPARQL являются графы, заданные в документах RDF и OWL. В предыдущих примерах использовался только один граф (исходный документ), причем его местоположение в запросе не задавалось.

В SPARQL имеется возможность прямого задания набора исходных документов, которые обрабатываются запросом.

Эти данные задаются помощью компонента ***исходные-данные*** и имеют одну из следующих форм:

FROM IRIref

или

FROM NAMED IRIref,

где **IRIref** – ссылка **IRI** или **URI** на местоположение исходного документа.

Набор исходных документов (набор данных RDF), который содержит один граф (один исходный документ) без имени называется

графом по умолчанию и должен обязательно присутствовать в наборе данных.

В наборе могут быть заданы и другие графы (исходные документы), называемые именованными графами. Если именованные графы присутствуют в наборе данных, то их местоположения задаются с помощью формы **FROM NAMED IRIref**.

Если заданы только местоположения именованных графов, то предполагается, что граф по умолчанию является пустым графом.

При выполнении запроса граф, используемый для поиска соответствия, называется активным графом. В предыдущих примерах использовался только граф по умолчанию, поэтому этот граф и являлся активным графом.

Если необходимо для части запроса сделать активным один или несколько именованных графов, то в условии запроса необходимо задать шаблон **GRAPH**, который имеет следующую форму:

GRAPH *параметр шаблон-группы-графов*

где ***параметр*** – либо **IRIref**, либо переменная, диапазоном значений которой являются все именованные графы, заданные в запросе, а ***шаблон-группы-графов*** определен в предыдущем разделе пособия.

Рассмотрим пример использования исходных документов в запросе типа SELECT.

Дополнительно к документу в нотации Turtle (рис. 27, файл bookLib.ttl), введем второй и третий документы Turtle. В этидокументах добавим свойство **dc:language**, задающее язык, на котором написана книга.

Второй документ **Turtle** (файл bookLib-1.ttl) содержит две книги и имеет вид, представленный на рис. 55.

Третий документ **Turtle** (файл bookLib-2.ttl) содержит одну книгу и представлен на рис. 56.

Запрос- 15 (рис. 57) обрабатывает файлы bookLib.ttl и bookLib-1.ttl. Однако строка запроса **GRAPH file:///D:/ARQ/bookLib-1.ttl** определяет активным лишь один граф данных.

Результат обработки запроса к исходным документам в текстовом представлении имеет следующий вид (рис. 58). То есть, в таблицу выведена информация об авторах, наименования книг и жанрах для графа данных, описанногов файле

file:///D:/Library/ARQ/bookTest-1.ttl.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix : <http://www.libRegistry.org/ns/lib#> .
        # Экземпляр book1-1 класса Book
:book1-1 a <http://www.libRegistry.org/ns/lib#Book>;
dc:identifier "5-699-07331-0";
dc:title "Трое в одной лодке, не считая собаки"@ru;
dc:creator "Джером Джером"@ru;
dc:language "русский"^^xsd:string;
dc:type "классика"^^xsd:string;
dc:publisher "Эксмо"@ru
dc:date "2004-11-09"^^xsd:date .
        # Экземпляр book2-1 класса Book
:book1-2 a <http://www.libRegistry.org/ns/lib#Book>;
dc:identifier "966-7047-42-3";
dc:title "Гаррі Поттер і орден Фенікса"@uk;
dc:creator "Джоан Ролінг"@uk;
dc:language "украинский"^^xsd:string;
dc:type "детская"^^xsd:string;
dc:publisher "А-Ба-Ба-Га-Ла-Ма-Га"@uk;
dc:date "2003-12-19"^^xsd:date .
```

Рис. 55. Исходный документ 2 (файл bookLib-1.ttl)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix : <http://www.libRegistry.org/ns/lib#> .
        # Экземпляр book2-1 класса Book
:book2-1 a <http://www.libRegistry.org/ns/lib#Book>;
dc:identifier "5-08-004015-7";
dc:title "Робинзон Крузо"@ru;
dc:creator "Дефо Даниэль"@ru;
dc:language "русский"^^xsd:string;
dc:type "детская"^^xsd:string;
dc:publisher "Детская литература"@ru;
dc:date "2006-05-17"^^xsd:date .
```

Рис. 56. Исходный документ 3 (файл bookLib-2.ttl)

Запрос – 16 (рис. 59) должен вывести в таблицу информацию об авторах, наименованиях книг и языке, на котором написаны книги, для классики и детской литературы сведения о которых заданы во всех трех файлах (bookTest.ttl, bookTest-1.ttl и bookTest-2.ttl).

Результат обработки запроса – 16 представлен на рис. 60.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?bookAuthor ?bookTitle ?bookType
FROM <file:///D:/Library/ARQ/bookLib.ttl>
FROM NAMED <file:///D:/ARQ/bookLib-1.ttl>
WHERE {
GRAPH <file:///D:/ARQ/bookLib-1.ttl>
{
?book dc:title ?bookTitle;
dc:creator ?bookAuthor;
dc:type ?bookType;
dc:language ?bookLanguage .
}
FILTER (?bookLanguage = "украинский"^^xsd:string)
}
```

Рис. 57. Запрос - 15

```
-------------------------------------------------------------------------------------------------------
| bookAuthor      | bookTitle                      | bookType                                         |
=======================================================================================================
| Джоан Ролінг@uk | Гаррі Поттер і орден Фенікса@uk | детская^^http://www.w3.org/2001/XMLSchema#string |
-------------------------------------------------------------------------------------------------------
```

Рис. 58. Результат выполнения Запроса – 15

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.libRegistry.org/ns/lib#>
SELECT ?bookAuthor ?bookTitle ?bookLanguage
FROM NAMED <file:///D:/ARQ/bookLib.ttl>
FROM NAMED <file:///D:/ARQ/bookLib-1.ttl>
FROM NAMED <file:///D:/ARQ/bookLib-2.ttl>
WHERE {
GRAPH ?named
{
?book dc:title ?bookTitle;
dc:creator ?bookAuthor;
dc:type ?bookType .
OPTIONAL {?book dc:language ?bookLanguage}
}
FILTER (?bookType = "классика"^^xsd:string ||
?bookType = "детская"^^xsd:string)
}
```

Рис. 59. Запрос - 16

bookAuthor	bookTitle	bookLanguage
Дефо Даниэль@ru	Робинзон Крузо@ru	русский^^http://www.w3.org/2001/XMLSchema#string
Джоан Ролінг@uk	Гаррі Поттер і орден Фенікса@uk	украинский^^http://www.w3.org/2001/XMLSchema#string
Джером Джером@ru	Трое в одной лодке, не считая собаки@ru	русский^^http://www.w3.org/2001/XMLSchema#string
Дефо Даниэл@ru	Приключения Робинзона Крузо@ru	
"Defoe Daniel"@en	"Robinson Crusoe"@en	
Шевченко Тарас@uk	Кобзар@uk	

Рис. 60. Результат выполнения Запроса - 16

2.8. Использование уточнений в запросе SELECT

Компонент ***уточнение*** в запросе задается либо как ключевое слово **DISTINCT,** либо как ключевое слово **REDUCED.**

Если задано уточнение **DISTINCT**, а в ответе на запрос несколько одинаковых переменных связаны с одинаковыми значениями ресурсов, то из результата удаляются все повторяющиеся значения, за исключением одного.

Если задано уточнение **REDUCED**, а в результате несколько одинаковых переменных связаны с одинаковыми значениями ресурсов, то из результата удаляются несколько повторяющихся значений. Количество удалений может меняться от нуля до n-1, где n – количество повторений.

Результат запроса -17 (рис. 61) к исходному документу (рис.27) в текстовом представлении имеет вид, представленный на рис.62.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?bookAuthor
WHERE {
?book dc:creator ?bookAuthor .
FILTER regex(str(?bookAuthor), "Кристи")
}
```

Рис. 61. Запрос - 17

```
----------------------
| bookAuthor         |
======================
| "Кристи Агата"@ru  |
| "Кристи Агата"@ru  |
----------------------
```

Рис. 62. Результат выполнения Запроса - 17

В ответе автор книги выведен дважды, поскольку этот автор встречается в двух книгах.

Если изменить строку запроса **SELECT ?bookAuthor** на **SELECT DISTINCT ?bookAuthor,** то имя автора будет выведено только один раз, то есть ответ на запрос будет иметь вид, указанный на рис. 63.

```
----------------------
| bookAuthor         |
======================
| "Кристи Агата"@ru  |
----------------------
```

Рис. 63. Результат выполнения Запроса – 17 после модификации

Точно такой же результат будет получен и при использовании **уточнения REDUCED.**

2.9. Использование модификаторов результатта в запросе

Компонент *модификаторы-результата* в запросе позволяет изменить выводимый результат с помощью следующих операций:

- упорядочение выводимых результатов;
- задание индекса, с которого начинается вывод результатов, в последовательности выводимых результатов;
- ограничение количества выводимых результатов в последовательности.

В запросе может быть задана одна или несколько операций из перечисленных операций.

Упорядочение выводимых результатов задается с помощью следующего выражения:

ORDER BY ***условия-упорядочения,***

где ***условия-упорядочения*** задает одно или несколько условий, разделенных пробелом.

Каждое из условий может принимать одну из следующих форм:

ASC ***выражение,***

или

DESC ***выражение,***

или

ограничение,

или

переменная.

Параметр ***выражение*** в первой и второй форме аналогичен выражению в фильтре

Параметр ***ограничение*** в третьей форме аналогичен ограничению в фильтре

Первая, третья и четвертая формы выполняют упорядочение результатов по возрастанию, в соответствии с компаратором, заданным в выражении, ограничении или по значениям переменной.

Вторая форма выполняет упорядочение результатов по убыванию, в соответствии с компаратором, заданным в выражении.

Индекс, с которого начинается вывод результатов, в последовательности выводимых результатов задается с помощью выражения **OFFSET** ***число,*** где число – целое неотрицательное число

(значение 0 – вывод с первого элемента последовательности, значение 1 – со второго и т.д.).

Ограничение количества выводимых результатов в последовательности задается с помощью выражения **LIMIT *число***, где ***число*** – целое положительное число (при значении 0 не выводится ни одного результата). Если заданное число больше количества полученных результатов, выводятся все результаты.

Рассмотрим пример использования модификаторов в запросе SELECT.

Так запрос – 18 (рис. 64) выводит таблицу (рис. 65), где авторы упорядочены по возрастанию (упорядочение выполняется сначала для английского языка, а затем для русского и украинского).

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?bookAuthor ?bookTitle
WHERE {
?book dc:title ?bookTitle;
dc:creator ?bookAuthor .
}
ORDER BY ASC(?bookAuthor)
```

Рис. 64. Запрос - 18

bookAuthor	bookTitle
"Defoe Daniel"@en	"Robinson Crusoe"@en
"Дефо Даниэл"@ru	"Приключения Робинзона Крузо"@ru
"Кристи Агата"@ru	"Десять негритят"@ru
"Кристи Агата"@ru	"Убийство в доме викария"@ru
"Шевченко Тарас"@uk	"Кобзар"@uk

Рис. 65. Результат выполнения Запроса – 18

Если изменить строку запроса **ORDER BY ASC(?bookAuthor)** на **ORDER BY ASC(?bookAuthor) LIMIT 3 OFFSET 1,** то из приведенной выше таблицы будет выведены три книги, начиная со второй книги (рис. 66).

Ответе на запрос-19 (рис. 67) книги Агаты Кристи будут упорядочены по убыванию (рис. 68).

bookAuthor	bookTitle
"Дефо Даниэл"@ru	"Приключения Робинзона Крузо"@ru
"Кристи Агата"@ru	"Десять негритят"@ru
"Кристи Агата"@ru	"Убийство в доме викария"@ru

Рис. 66. Результат выполнения Запроса – 18 после модификации

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?bookAuthor ?bookTitle
WHERE {
?book dc:title ?bookTitle;
dc:creator ?bookAuthor .
}
ORDER BY ?bookAuthor DESC(?bookTitle)
```

Рис. 67. Запрос - 19

bookAuthor	bookTitle
"Defoe Daniel"@en	"Robinson Crusoe"@en
"Дефо Даниэл"@ru	"Приключения Робинзона Крузо"@ru
"Кристи Агата"@ru	"Убийство в доме викария"@ru
"Кристи Агата"@ru	"Десять негритят"@ru
"Шевченко Тарас"@uk	"Кобзар"@uk

Рис. 68. Результат выполнения Запроса – 19

2.10. Запрос CONSTRACT

Запрос **CONSTRUCT** возвращает граф RDF, заданный шаблоном графа, который указан в запросе. Этот результирующий граф формируется (конструируется) следующим образом: для каждого результата в последовательности результатов запроса выполняется подстановка переменных из шаблона графа, а затем все триплеты объединяются в единый граф RDF.

Если в результате такого преобразования получается триплет, содержащий несвязанную переменную или неправильную конструкцию RDF, например, литерал в позиции субъекта или предиката, такой триплет не включается в выходной граф RDF.

Шаблон графа может содержать триплеты без переменных, которые выводятся в выходном графе RDF без изменений.

Запрос **CONSTRUCT** имеет следующую форму.

CONSTRUCT ***шаблон-конструирования исходные-данные условие модификаторы-результата***

Параметр ***шаблон-конструирования*** содержит базовый шаблон графа вз апросе), заключенный в фигурные скобки"{" и "}".

Остальные параметры – такие же, как и в запросе **SELECT**: ***исходные-данные, условие*** и ***модификаторы-результата***

Обязательными параметрами являются ***шаблон-конструирования*** и ***условие.***

Используем запрос **CONSTRUCT** для создания нового графа RDF для списка книг из RDF документа (рис.27). В новом графе для каждой книги на русском или украинском языке будут заданы только свойства **dc:creator** и **dc:title**, а также добавлено свойство **dc:language**.

Сначала формируем запрос -20 , который формирует документ **Turtle** (в файле **bookLibru.ttl**), в котором заданы книги на русском языке (рис. 69).

Следующий запрос – 21 (рис. 71) формирует документ **Turtle** (в файле **bookLibuk.ttl**) , в котором задана книга на украинском языке (рис. 72).

Затем запрос – 22 (рис. 73) соединяет два полученных документа в один документ **Turtle** (рис. 74).

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
CONSTRUCT {?book dc:title ?bookTitle;
dc:creator ?bookAuthor;
dc:language "русский"}
WHERE {
?book dc:title ?bookTitle;
dc:creator ?bookAuthor .
FILTER (lang(?bookAuthor) = "ru")
}
```

Рис. 69. Запрос - 20

```
# ======== CONSTRUCT results
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://www.libRegistry.org/ns/lib#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

:book1
dc:creator "Кристи Агата"@ru ;
dc:language "русский" ;
dc:title "Десять негритят"@ru .

:book4
dc:creator "Кристи Агата"@ru ;
dc:language "русский" ;
dc:title "Убийство в доме викария"@ru .

:book5
dc:creator "Дефо Даниэл"@ru ;
dc:language "русский" ;
dc:title "Приключения Робинзона Крузо"@ru .
# ========
```

Рис. 70. Результат выполнения Запроса – 20

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
CONSTRUCT {?book dc:title ?bookTitle;
dc:creator ?bookAuthor;
dc:language "украинский"}
WHERE {
?book dc:title ?bookTitle;
dc:creator ?bookAuthor .
FILTER (lang(?bookAuthor) = "uk")
}
```

Рис. 71. Запрос – 21

```
# ======== CONSTRUCT results
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://www.libRegistry.org/ns/lib#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

:book2
dc:creator "Шевченко Тарас"@uk ;
dc:language "украинский" ;
dc:title "Кобзар"@uk .
```

Рис. 72. Результат выполнения Запроса – 21

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
CONSTRUCT {?book dc:title ?bookTitle;
dc:creator ?bookAuthor;
dc:language ?bookLang}
FROM NAMED <file:///D:/ARQ/bookLibru.ttl>
FROM NAMED <file:///D:/ARQ/bookLibuk.ttl>
WHERE {
GRAPH ?named {
?book dc:title ?bookTitle;
dc:creator ?bookAuthor;
dc:language ?bookLang .
}
}
```

Рис. 73. Запрос - 22

2.11. Запрос ASK

Запрос **ASK** проверяет, существует ли результат для заданного в тексте запроса шаблона и имеет следующую форму.

ASK ***исходные-данные условие.***

Параметры запроса – такие же, как и в запросе **SELECT**: ***исходные-данные*** и ***условие***. Обязательным параметром является ***условие.***

Результат запроса может быть выведен либо в текстовом виде, либо в формате XML.

Вывод в текстовом виде представляет собой строку, содержащее слово **yes**, если результат запроса существует и **no** – в противном случае.

```
# ======== CONSTRUCT results
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://www.libRegistry.org/ns/lib#book1>
dc:creator "Кристи Агата"@ru ;
dc:language "русский" ;
dc:title "Десять негритят"@ru .

<http://www.libRegistry.org/ns/lib#book2>
dc:creator "Шевченко Тарас"@uk ;
dc:language "украинский" ;
dc:title "Кобзар"@uk .

<http://www.libRegistry.org/ns/lib#book4>
dc:creator "Кристи Агата"@ru ;
dc:language "русский" ;
dc:title "Убийство в доме викария"@ru .

<http://www.libRegistry.org/ns/lib#book5>
dc:creator "Дефо Даниэл"@ru ;
dc:language "русский" ;
dc:title "Приключения Робинзона Крузо"@ru .
# ========
```

Рис. 74. Результат выполнения Запроса – 22

Вывод в формате XML имеет следующий вид (рис. 75).

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
<head></head>
<results>
<boolean>значение</boolean>
</results>
</sparql>
```

Рис. 75. XML формат ответа на запрос типа **ASK**

Параметр ***значение*** в выводе равен **true**, если результат запроса существует и **false** – в противном случае.

Ниже приведен пример использования запроса – 23 (рис. 76) к RDF документу (рис. 27)

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
ASK {?book dc:title "Приключения Робинзона Крузо"@ru}
```

Рис. 76. Запрос - 23

Ответ в в текстовом представлении имеет следующий вид:

```
yes
```

В формате XML этот результат представлен на рис. 77.

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
<head>
</head>
<boolean>true</boolean>
</sparql>.
```

Рис. 77. XML формат ответа на запрос - 23

Запрос – 24 (рис. 78) к исходному документу (рис. 27) в текстовом представлении имеет следующий вид:

```
no
```

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
ASK {?book dc:title "Приключения Тома Сойера"@ru}
```

Рис. 78 Запрос - 24

2.12. Запрос DESCRIBE

Запрос **DESCRIBE** для заданного шаблона графов возвращает как результат граф RDF, содержащий данные о ресурсах, удовлетворяющих требованиям запроса в формате RDF документа.

Запрос **DESCRIBE** имеет следующую форму.

DESCRIBE ***список исходные-данные условие модификаторы-результата.***

В запросе обязательно должен быть задан ***список***, а также либо ***условие***, либо ***исходные-данные.***

Элементами списка являются либо переменные, либо ссылки на IRI(URI) – IRIref. Список может также содержать только символ "*", что подразумевает использовать список всех переменных, заданных в запросе.

Остальные параметры – те же, что в запросе **SELECT**: ***исходные-данные***, ***условие*** и ***модификаторы-результата***.

Запрос – 25 (рис. 79) к исходному RDF документу (рис. 27) выводит документ Turtle , содержащий описание ресурсов (книг), для которых автором является "Кристи Агата" (рис. 80).

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
DESCRIBE ?book
WHERE {?book dc:creator "Кристи Агата"@ru}
```

Рис. 79. Запрос - 25

```
# ======== DESCRIBE results
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://www.libRegistry.org/ns/lib#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-
ns#> .
:book1
        rdf:type :Book ;
        dc:creator "Кристи Агата"@ru ;
        dc:date "2003-03-09"^^xsd:date ;
        dc:identifier "02419-0" ;
        dc:publisher "Эксмо"@ru ;
        dc:title "Десять негритят"@ru ;
        dc:type "детектив"^^xsd:string .
:book4
        df:type :Book ;
        c:creator "Кристи Агата"@ru ;
        dc:date "2006-01-28"^^xsd:date ;
        dc:identifier "5-367-00019-3" ;
        dc:publisher "Амфора"@ru ;
        dc:title "Убийство в доме викария"@ru ;
        dc:type "детектив"^^xsd:string .
# ========
```

Рис. 80. Результат выполнения Запроса – 25

Вопросы для самопроверки

1.Опишите структуру SPARQL – запроса.

2. В какой части запроса располагаются предложения объявлений префиксов?

3. Какие данные располагаются в части запроса называемой пролог?

4.Какую функцию выполняет указание SELECT в SPARQL – запросе?

5. Какие условия вывода информации в ответе на запрос определяет указание SELECT *?

6. Какую задачу решает использование ключевого слова OPTIONAL?

7. Для какой цели используется ключевое слово UNION?

8.Какие встроенные функции SPARQL в составе запроса может вызывать FILTER?

9. Каким образом в SPARQL – запросе задается источник данных для обработки?

3. Точки доступа SPARQL

Точка доступа SPARQL — это служба, поддерживающая протокол запросов SPARQL. Точка доступа позволяет пользователю делать запросы к RDF хранилищу. Сервер обрабатывает запрос и возвращает ответ в некотором, обычно машинно-читаемом, формате. То есть, точки доступа SPARQL по сути дела являются API (application programming interface) к базам знаний, а представление результатов обычно реализуется программным обеспечением вызывающей стороны.

Различают два вида точек доступа: ***общего назначения*** и ***локальные.***

Точки доступа общего назначения могут производить запросы по любым указанным RDF-документам, находящимся в сети. А локальные точки доступа способны получать данные только от одного ресурса.

Примеры локальных точек доступа: NASA endpoint, BBC wildlife endpoint, DBPedia endpoint и др.

3.1. Точка доступа DBpedia

DBpedia — является одним из наиболее крупных центров информации в области Linked Data (связанные данные). DBpedia — это проект создания структурированной и постоянно обновляющейся копии Википедии, представленной в языке RDF.

Набор данных DBpedia представляет из себя большую многодоменную онтологию которая была получена из Википедии. Английская версия DBpedia в настоящее время описывает 4,0 миллиона "сущностей" с 470 миллионами "фактов".

Существуют локализованные версии DBpedia на 119 языках. Все эти версии вместе описывают 24,9 миллиона объектов. Полный набор данных DBpedia индексирует и описывает 12,6 миллионов уникальных вещей на 120 различных языках, использует 24,6 миллиона ссылок на изображения и 27,6 миллионов HTML-ссылок на внешние веб-страницы. DBpedia использует 45,0 миллионов ссылок на внешние наборы RDF-данных, 67,0 миллионов ссылок на категории Wikipedia и 41,2 миллиона ссылок на YAGO категории. Этот набор данных состоит из 2,46 миллиарда единиц информации (RDF троек), из которых 470 миллионов были взяты из английского издания Википедии, 1,98 миллиарда были извлечены из других языковых разделов.

Перейти в точку доступа DBpedia можно по ссылке **http://dbpedia.org/sparql/**. Общий вид интерфейса точки доступа приведен на рис. 81, где:

1 – поле, где указан сетевой адрес источника RDF данных;

2 – поле для ввода текста SPARQL запроса;

3 – поле настройки формата представления данных ответа на запрос;

4 – поле настройки времени (в миллисекундах),затрачиваемого на обработку данных;

5 – кнопка запуска в работу программы обработки запроса;

6 – ссылка на источник справочной информации о SPARQL;

7 – ссылка на информацию о предустановленных префиксах, которые могут быть использованы в запросах данного интерфейса;

8 – ссылка на информацию об используемых в данном интерфейсе правилах вывода;

9 – ссылка на альтернативный интерфейс SPARQL.

Рассмотрим задачу поиска в DBpedia информации об авиационных фирмах (компаниях), которые расположены в России.

Сначала определим класс данных в котором сгруппированы данные о фирмах. Для этого используем SPARQL запрос следующего вида

(рис.82). В запросе в качестве субъекта используем пустой узел **[ ],** в обозначении предиката использован предустановленный префикс **rdf:**, что соответствует **http://www.w3.org/1999/02/22-rdf-syntax-ns#**. Результаты обработки запроса помещаются в переменную **?Concept**. Объем предоставляемых в ответе данных ограничивается 100 элементами (**LIMIT 100**).

В полученном ответе найдем класс, который соответствует компаниям (рис. 83).

Если открыть эту ссылку, то можно определить связь с промышленными компаниями (рис. 84).

Откорректируем SPARQL запрос, выделив только компании связанные с ресурсом **dbo:industry** (рис. 85).

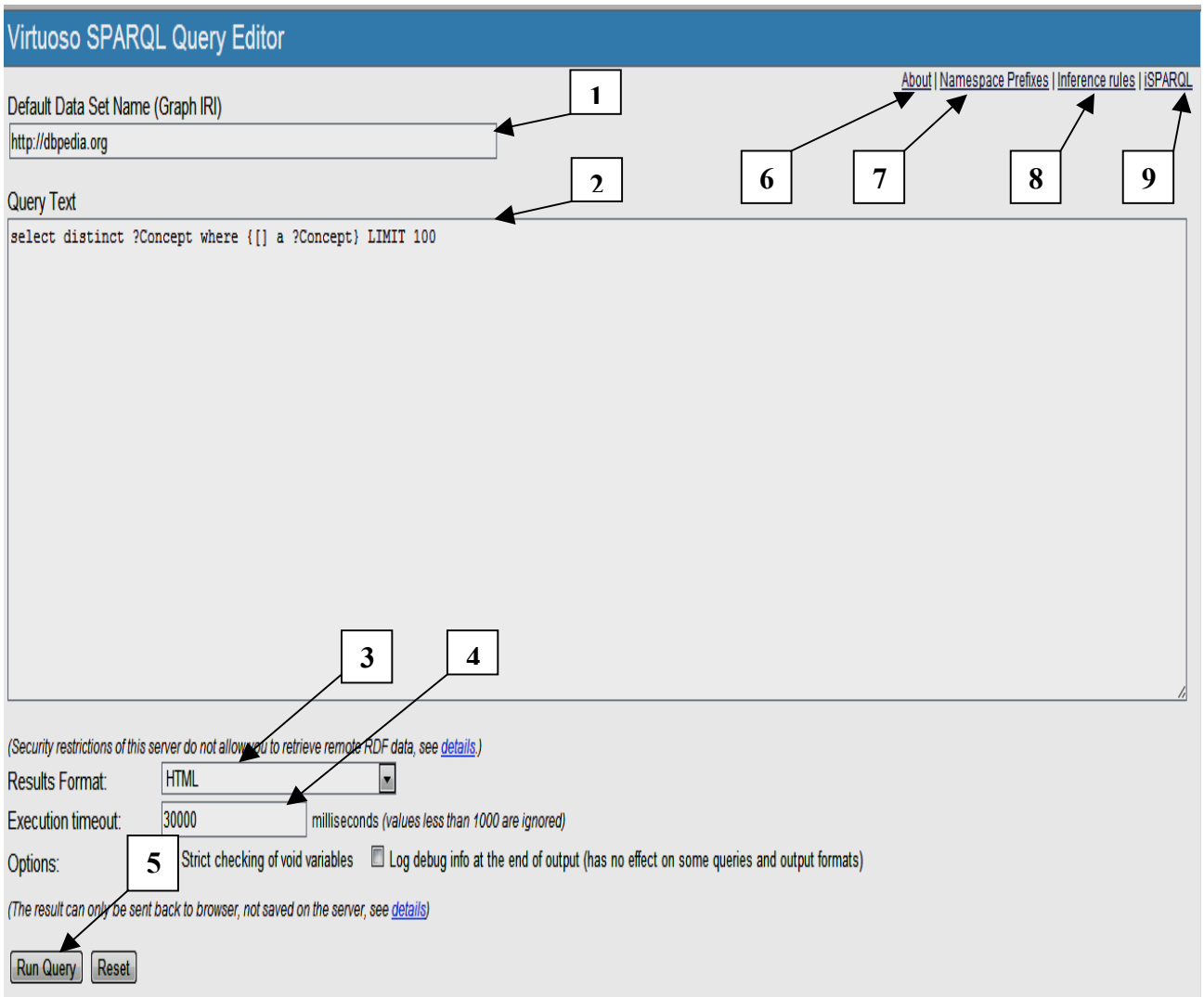


Рис. 81. Общий вид интерфейса точки доступа к RDF хранилищу DBpedia

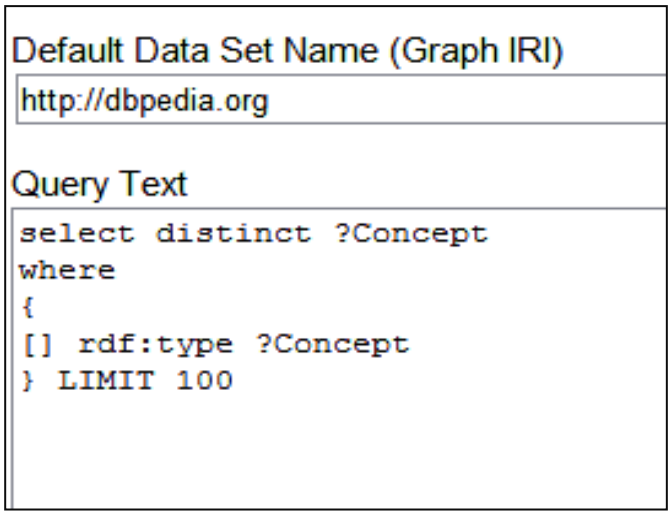


Рис. 82. Текст запроса – 26 к RDF хранилищу DBpedia

http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing
http://dbpedia.org/class/yago/YagoPermanentlyLocatedEntity
http://dbpedia.org/class/yago/PhysicalEntity100001930
http://www.w3.org/2002/07/owl#Thing
http://dbpedia.org/ontology/Company
http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#Agent
http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#SocialPerson
http://www.wikidata.org/entity/Q43229
http://dbpedia.org/ontology/Agent
http://dbpedia.org/ontology/Organisation
http://schema.org/Organization

Рис. 83. Фрагмент данных из ответа на запрос - 26

В ответе на данный запрос можно выделить компанию, которая относится к авиастроительным фирмам (рис. 86).

Чтобы выделить только авиастроительные компании, откроем

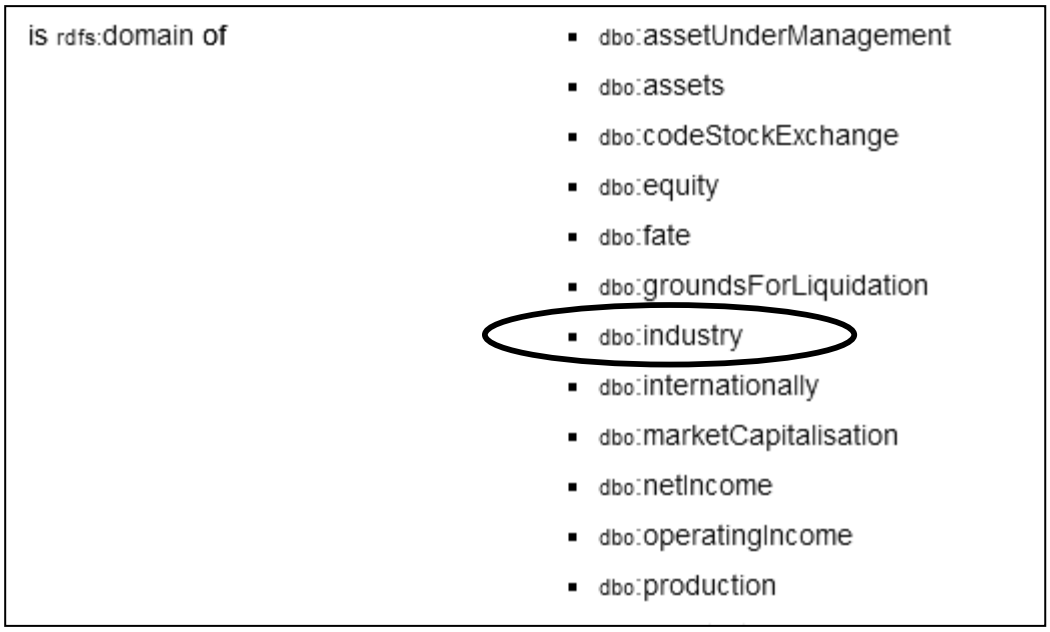


Рис. 84. Фрагмент данных ресурса http://dbpedia.org/ontology/Company

ресурс **http://dbpedia.org/resource/Aero_Nord** и определим значение свойства **dbo:industry**, которое (рис. 87) (**dbr:Aerospace)** позволит решить поставленную задачу и соответствующим образом откорректируем текст запроса (рис.88).

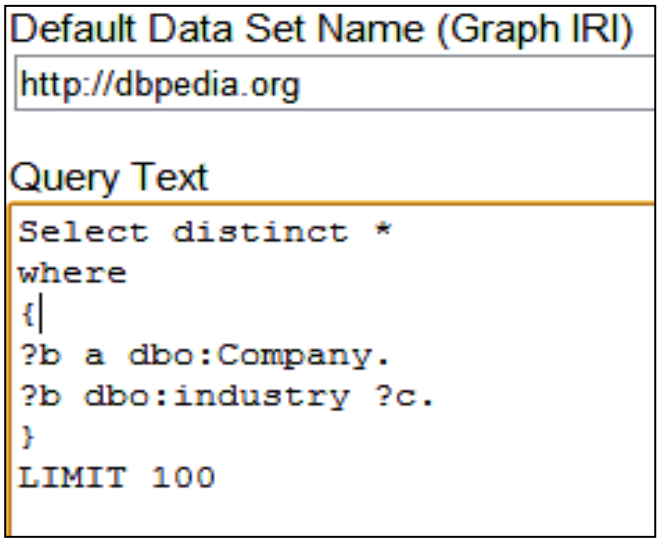


Рис. 85. Текст запроса – 27 к RDF хранилищу DBpedia

http://dbpedia.org/resource/Adgreetz	http://dbpedia.org/resource/Marketing
http://dbpedia.org/resource/Advanced_Business_Solutions	http://dbpedia.org/resource/Enterprise_software
http://dbpedia.org/resource/Aero_Nord	http://dbpedia.org/resource/Aerospace
http://dbpedia.org/resource/Aerochute_International	http://dbpedia.org/resource/Aerospace

Рис. 86. Фрагмент данных из ответа на запрос – 27

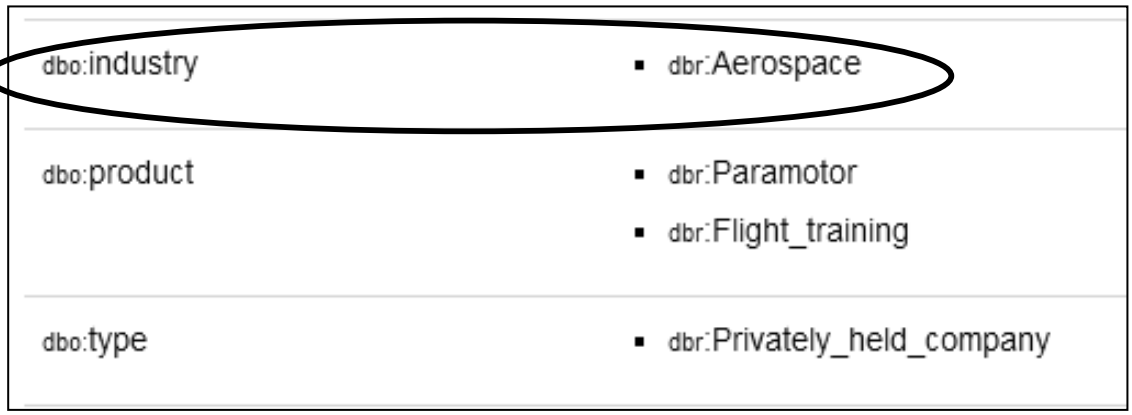


Рис. 87. Фрагмент данных ресурса http://dbpedia.org/resource/Aero_Nord

Обработка запроса позволяет получить список авиастроительных компаний (рис. 89).

Default Data Set Name (Graph IRI)

http://dbpedia.org

Query Text

```
select distinct *
where
{
?b a <http://dbpedia.org/ontology/Company>.
#?b dbo:industry ?c.
?b dbo:industry  dbr:Aerospace.
}
 LIMIT 1000
```

Рис. 88. Текст запроса – 28 к RDF хранилищу DBpedia

b
http://dbpedia.org/resource/Aero_Nord
http://dbpedia.org/resource/Aerochute_International
http://dbpedia.org/resource/Alpha_Unmanned_Systems
http://dbpedia.org/resource/Bilsam_Aviation
http://dbpedia.org/resource/Brumby_Aircraft_Australia
http://dbpedia.org/resource/Govgistics
http://dbpedia.org/resource/Heli-Sport
http://dbpedia.org/resource/InterPlane_Aircraft
http://dbpedia.org/resource/Little_Wing_Autogyros._Inc.
http://dbpedia.org/resource/Modti_inc.
http://dbpedia.org/resource/One_Aviation
http://dbpedia.org/resource/Parascender_Technologies

Рис. 89. Фрагмент данных из ответа на запрос – 28

Определим авиастроительные компании, которые расположены в России. Откроем ресурс **dbp:locationCountry** (рис. 90) и определим свойство **dbp:locationCountry**, значение которого определяет страну расположения компании.

dbp:locationCity	dbr:Albuquerque,_New_Mexico
dbp:locationCountry	dbr:United_States
dbp:logo	dbr:File:One_Aviation_logo.png
dbp:name	One Aviation (en)

Рис. 90. Фрагмент данных ресурса http://dbpedia.org/resource/One_Aviation

Откорректируем текст запроса таким образом, чтобы в таблицу выводились только компании, которые расположены в России (рис. 91) , кроме того, выведем в таблицу данные о городе в котором расположена компания (рис. 92).

Default Data Set Name (Graph IRI)

http://dbpedia.org

Query Text

```
select distinct *
where
{
?b a <http://dbpedia.org/ontology/Company>.
#?b dbo:industry ?c.
?b dbo:industry  dbr:Aerospace.
?b dbo:locationCountry dbr:Russia.
?b dbp:locationCity ?c.
}
 LIMIT 1000
```

Рис. 91. Текст запроса – 29 к RDF хранилищу DBpedia

b	c
http://dbpedia.org/resource/Aviastroitel	http://dbpedia.org/resource/Moscow
http://dbpedia.org/resource/Airbridge_(ultralight_aircraft_manufacturer)	http://dbpedia.org/resource/Moscow
http://dbpedia.org/resource/Krasniye_Kryl'ya	http://dbpedia.org/resource/Taganrog
http://dbpedia.org/resource/Tupolev	http://dbpedia.org/resource/Moscow

Рис. 92. данные ответа на запрос – 29

Для уточнения данных о авиастроительных компаниях можно использовать свойство **dbo:foundationPlace**,в этом случае текст запроса будет выглядеть следующим образом (рис. 93). Результат обработки запроса представлен на рис. 94.

Default Data Set Name (Graph IRI)

http://dbpedia.org

Query Text

```
select distinct *
where
{
?b a <http://dbpedia.org/ontology/Company>.
#?b dbo:industry ?c.
?b dbo:industry  dbr:Aerospace.
{?b dbo:locationCountry dbr:Russia.
?b dbp:locationCity ?c.}
UNION
{?b   dbo:foundationPlace dbr:Moscow.}
}
 LIMIT 1000
```

Рис. 93. Текст запроса – 30 к RDF хранилищу DBpedia

b	c
http://dbpedia.org/resource/Aviastroitel	http://dbpedia.org/resource/Moscow
http://dbpedia.org/resource/Airbridge_(ultralight_aircraft_manufacturer)	http://dbpedia.org/resource/Moscow
http://dbpedia.org/resource/Krasniye_Kryl'ya	http://dbpedia.org/resource/Taganrog
http://dbpedia.org/resource/Tupolev	http://dbpedia.org/resource/Moscow
http://dbpedia.org/resource/Yakovlev	
http://dbpedia.org/resource/Polikarpov	
http://dbpedia.org/resource/Tupolev	
http://dbpedia.org/resource/Mikoyan	

Рис. 94. Данные ответа на запрос – 30

3.2. SPARQL запросы к FOAF файлу

FOAF (Friend of a Friend - "друг друга") является онтологией описания человека, выполненной с использованием машиночитаемого языка для описания людей, групп и отношений между ними.

Основная идея **FOAF** заключается в хранении информации в формате, который не только понятен людям, но и легко обрабатывается компьютерами. Стандартный набор свойств **FOAF** однозначно описывает отношения между объектами.

Вся совокупность документов **FOAF** является связанной информационной системой, то есть представляет собой единую информационную базу, где. каждая страница **FOAF** связана с другими **FOAF**-документами.

Документы **FOAF** основываются на синтаксисе **XML** и используют конвенцию описания ресурсов **RDF**. **FOAF** определяет набор полезных классов и свойств, которые могут использоваться совместно с обычными **RDF**-словарями и онтологиями.

Ранее рассмотренный запрос – 26 определяет состав классов в RDF. Анализ состава этих классов позволяет выделить класс **dbo:Person** (рис. 95), который связан с ресурсом **FOAF** (рис. 96).

В качестве примера использования данных **FOAF** рассмотрим задачу поиска людей по нескольким параметрам, в частности найдем

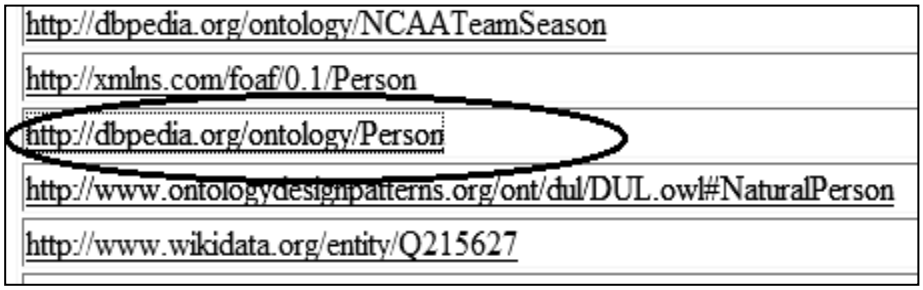


Рис. 95. Данные ответа на запрос – 26

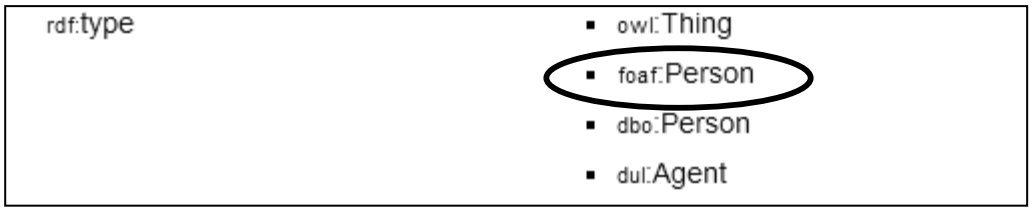


Рис. 96. Связь DBpedia с ресурсом FOAF

данные на людей которые имеют определенные заслуги перед обществом (награждены медалями, премиями и др.), выделим среди них наших

соотечественников и получим краткое описание характера наград. А затем определим только тех, кто награжден Большой золотой медалью М.В. Ломоносова.

Сформируем запрос – 31, котрый выведет в таблицу ресурсы, описывающие представителей класса **foaf:Person,** которые имеют награды и краткое описание наград (рис. 97). Описания наград выведем с использованием оператора FILTER, разрешив вывод описаний только на русском языке. Далее выявим среди награжденных наших соотечественников (запрос – 32 на рис. 98). Для этого используем свойство элементов класса **foaf:Person**, которое обозначается как **dbp:nationality** и оператор . FILTER, который позволяет выводить в таблицу только те субъекты у которых значение свойства **dbp:nationality** будет содержать фрагмент литерала **"rus".**

Результаты обработки последних запросов не приводятся в пособии, поскольку они занимают большой объем и имеют неудобный формат вывода. Читателям предлагается самим проделать описанные упражнения и получить информацию соответствующую составленным запросам.

Анализ свойства показывает, что одним из возможных значений этого свойства является значение **dbr:Soviet_Union**, поэтому для учета всех награжденных соотечественников необходимо учитывать и субъекты с данным значением свойства. Откорректируем последний запрос с учетом последних данных (см. запрос – 33, рис. 99).

Default Data Set Name (Graph IRI)

http://dbpedia.org

Query Text

```
select distinct *
where
{
?personal a foaf:Person.
?personal dbo:award ?award.
?award dbo:abstract ?abstracta.
FILTER(langMatches(lang(?abstracta), "ru")).
}

LIMIT 100
```

Рис. 97. Текст запроса – 31 к RDF хранилищу данных

Default Data Set Name (Graph IRI)

http://dbpedia.org

Query Text

```
select distinct *
where
{
?personal a foaf:Person.
?personal dbo:award ?award.
?award dbo:abstract ?abstracta.
FILTER(langMatches(lang(?abstracta), "ru")).
?personal dbo:nationality ?na.
FILTER regex(?na, "RUS", "i").

}

LIMIT 1000
```

Рис. 98. Текст запроса – 32 к RDF хранилищу данных

Default Data Set Name (Graph IRI)

http://dbpedia.org

Query Text

```
select distinct *
where
{
?personal a foaf:Person.
?personal dbo:award ?award.
?award dbo:abstract ?abstracta.
FILTER(langMatches(lang(?abstracta), "ru")).
{?personal dbo:nationality ?na.
FILTER regex(?na, "RUS", "i").}
UNION
{?personal dbo:nationality ?na.
FILTER regex(?na, "Sov", "i").}

}

LIMIT 1000
```

Рис. 99. Текст запроса – 33 к RDF хранилищу данных

Чтобы определить, какие из субъектов награждены Большой золотой медалью М.В.Ломоносова используем свойство **dbo:award** в последнем запросе, которому присвоим значение **dbr:Lomonosov_Gold_Medal** (запрос – 34, рис. 100).

Результаты послученные по последнему запросу показывают, что поставленная задача выполнена.

Default Data Set Name (Graph IRI)

http://dbpedia.org

Query Text

```
select distinct *
where
{
?personal a foaf:Person.
?personal dbo:award ?award.
?personal dbo:award   dbr:Lomonosov_Gold_Medal.
?award dbo:abstract ?abstracta.
FILTER(langMatches(lang(?abstracta), "ru")).
{?personal dbo:nationality ?na.
FILTER regex(?na, "RUS", "i").}
UNION
{?personal dbo:nationality ?na.
FILTER regex(?na, "Sov", "i").}

}

LIMIT 1000
```

Рис. 100. Текст запроса – 34 к RDF хранилищу данных

3.3. SPARQL запросы с использованием Dublin Core

Дублинское ядро (Dublin Core, DC)- это набор элементов метаданных, смысл которых описан вербально и зафиксирован в спецификациях его стандартов.

Комбинация значений этих элементов Dublin Core представляет собой некоторую запись, описывающую информационный ресурс.

Целью такого описания является обеспечение широкого доступа к сетевым информационным ресурсам.

Дело в том, что в связи со значительным ростом ресурсов интернет сети, становится все сложнее найти необходимый для использоваия ресурс.

В настоящее время существует ряд каталогов ресурсов интернет, однако они являются и не полными, и не точными, поскольку интернет постоянно меняется. Возникают одни ресурсы, закрываются другие, меняют содержание третьи и т.д.. Поэтому любой каталог без надежного механизма корректировки устаревает в момент его создания.

С 2005 года словарь Dublin Core представлен и в формате RDF и является основой для описания ресурсов в сети интернет.

Первоначальная версия Dublin Core включала 13 элементов и была предложена на состоявшемся в 1995 году в Дублине (США) симпозиуме, организованном Online Computer Library Center (OCLC) и National Center for Supercomputing Applications (NCSA). Эта версия решала задачи описания информационных ресурсов библиотечных систем, информационных ресурсов интернета, электронных документов и т.п. Развитие Dublin Core поддерживается специально учрежденной для этой цели организацией - Dublin Core Metadata Initiative (Инициатива по метаданным Дублинского ядра, DCMI).

Действующая версия спецификаций Дублинского ядра DC 1.1 была принята в качестве стандарта DCMI в 1999 г., а впоследствии приобрела статус официальных стандартов Организации национальных стандартов информационных технологий США – NISO (стандарт NISO Z39.85-2001) и Международной организации стандартизации – ISO (стандарт ISO: 15836-2003).

Набор элементов Dublin Core разработан на английском языке, но созданы версии и на многих других языках, включая финский, норвежский, тайваньский, японский, португальский, немецкий, греческий, индонезийский и испанский и др. Правительство ряда стран утвердило Dublin Core в качестве национального стандарта метаданных.

В России с 1 июля 2011 года действует ГОСТ Р 7.0.10-2010 (ИСО 15836:2003) «Национальный стандарт Российской Федерации. Система стандартов по информации, библиотечному и издательскому делу. Набор элементов метаданных „Дублинское ядро“».

Словарь Dublin Core разделён на два уровня:

- простой (неквалифицированный, simple), состоящий из 15 элементов;
- компетентный (квалифицированный, qualified), состоящий из 18 элементов и группы квалификаторов, которые уточняют семантику элементов для повышения полезности поиска ресурсов.

Простой *набор элементов метаданных Дублинского ядра* (Dublin Core Metadata Element Set; DCMES) состоит из 15 следующих элементов метаданных:

1. Title — название (это наименование, под которым известен данный ресурс);
2. Creator — создатель (человек (автор), организация или служба, несущие первичную ответственность за создание ресурса);
3. Subject — тема (предметное содержание ресурса, перечень ключевых слов к информационному ресурсу);
4. Description — описание (обзор содержания ресурса, может содержать аннотацию, оглавление, ссылку на полнотекстовый реферат, а также другие данные);
5. Publisher — издатель (человек, организация или служба, ответственные за предоставление ресурса пользователям);
6. Contributor — внёсший вклад, соисполнитель (человек, организация или служба, ответственные за вклад в содержание ресурса);
7. Date — дата (дата некоторого события в жизненном цикле ресурса, обычно ассоциируется с датой создания или предоставления ресурса в пользование);
8. Type — тип (природа или жанр содержания ресурса, например, для статьи это может быть следующий вариант - <meta name="DC.type" content="**Text**" />);
9. Format — формат документа (физическое или форматное представления ресурса, например - <meta name="DC.format" content="**text/html**" />);
10. Identifier — идентификатор ресурса и, (однозначная в пределах данного контекста ссылка на ресурс);
11. Source — источник (ссылка на ресурс, послуживший источником данных);
12. Language — язык (язык изложения содержания ресурса);
13. Relation — отношения (ссылка на связанный (родственный) ресурс);
14. Coverage — покрытие (локализация и границы тематики ресурса, как правило, определяет пространственную локализацию (географическое наименование или координаты), отрезок времени (наименование периода, дата или интервал дат) или юрисдикцию (наименование административной единицы).;
15. Rights — авторские права (сведения о правах на использование и управление ресурсом, обычно элемент содержит положение об управлении ресурсом или ссылку на службу, где можно получить эту информацию).

Квалифицированный (компетентный) *набор элементов метаданных Дублинского ядра*, помимо 15 вышеперечисленных, может включать:

- Audience — аудитория (зрители);
- Provenance — происхождение;
- RightsHolder — правообладатель.

Все элементы описания факультативны (опциональны) и могут повторяться. Элементы метаданных могут появляться в любом порядке".

Полная информация по определениям элементов и отношениям между ними описана в *Реестре метаданных Дублинского ядра* (Dublin Core Metadata Registry).

Рассмотрим задачу поиска библиографической информации в базе данных Британской Национальной Библиотеки (BNB). Доступ к точке доступа, работающей с указанным ресурсом можно осуществить по ссылке **http://bnb.data.bl.uk/flint-sparql.** Интерфейс точки доступа BNB мало отличается от описанного ранее интерфейса точки доступа DBpediaи не требует дополнительных пояснений.

Например, найдем 50 книг, изданных в Москве. Сначала определим место издания книг, это можно сделать, задав следующую тройку

?place rdfs:label "Moscow" .

Переменная **?place** имеет свойство со значением места издания **"Moscow"**.

На следующем этапе определим публикации у которых свойство "место издания" (<http://purl.org/NET/c4dm/event.owl#place>) имеет значение переменной **?place**, которое определено предыдущей тройкой.

?publication event:place ?place.

Публикации, удовлетворяющие этому условию будут записаны в переменную **?publication**.

Следующий этап решает задачу выделения книг с заданным свойством в фонде BNB.

?book blt:publication ?publication.

То есть, в переменную **?book** записываются все ресурсы обладающие свойством <http://www.bl.uk/schemas/bibliographic/blterms#publication>, значение которого равно значению переменной **?publication**.

Далее для ресурсов, записанных в переменную ?book, которые уже удовлетворяют поставленным требованиям (изданы в г. Москва и существуют в BNB) определим идентификационный номер **(?isbn**) и наименование книги (**?title)**.

?book bibo:isbn10 ?isbn.

?book dct:title ?title.

При этом используем предикаты
<http://purl.org/ontology/bibo/isbn10> и <http://purl.org/dc/terms/ title />

В таблицу результатов обработки запроса выведем наименование (URI) ресурса, идентификационный номер ресурса и наименование книги.

Для записи текста запроса сначала определим предикаты для используемых ресурсов.

```
PREFIX bibo: <http://purl.org/ontology/bibo/>
PREFIX blt: <http://www.bl.uk/schemas/bibliographic/blterms#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX event: <http://purl.org/NET/c4dm/event.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

В операторе SELECT определим переменные подлежащие выводу в таблицу и ключевое слово distinct, чтобы исключить вывод повторяющихся результатов.

```
SELECT distinct ?book ?isbn ?title
```

В шаблон графов поместим все ранее определенные тройки и ограничим количество выводимой информации пятидесятью объектами.

```
WHERE
{

  ?place rdfs:label "Moscow" .
  ?publication event:place ?place.
  ?book blt:publication ?publication;
        bibo:isbn10 ?isbn;
        dct:title ?title.
  }
LIMIT 50
```

Фрагмент результатов работы данного запроса представлен на рис. 101.


Query Results

Visual Results Mode

book	isbn	title
http://bnb.data.bl.uk/id/resource/008086275	0714717991	Stories
http://bnb.data.bl.uk/id/resource/008086161	0714716650	Champions in the making
http://bnb.data.bl.uk/id/resource/008086243	0714717622	Molecules without chemical bonds : essays on chemical topology
http://bnb.data.bl.uk/id/resource/008086345	0714719285	Ulyanovsk : a guide
http://bnb.data.bl.uk/id/resource/008086173	0714716820	Afghanistan : past and present

Рис. 101. Результат обработки запроса к ресурсам BNB

Рассмотрим другую задачу поиска книг по определенной тематике.

Найдем книги по кристаллографии и выведем их параметры (URI, идентификационный номер и наименование) в таблицу.

Текст запроса представлен на рис. 102, результаты запроса – на рис. 103.

```
PREFIX bibo: <http://purl.org/ontology/bibo/>
PREFIX dct: <http://purl.org/dc/terms/>

SELECT ?book ?isbn ?title WHERE
{
  ?book dct:subject <http://bnb.data.bl.uk/id/concept/lcsh/Crystallography>;
        bibo:isbn13 ?isbn;
        dct:title ?title.
}
LIMIT 50
```

Рис. 102. Текст запроса на поиск книг по тематике "кристалография" из фонда BNB

book	isbn	title
http://bnb.data.bl.uk/id/resource/015351099	9780470699614	NMR crystallography
http://bnb.data.bl.uk/id/resource/016196249	9781439887622	Structure and properties of fat crystal networks
http://bnb.data.bl.uk/id/resource/017050047	9781634637916	Crystals and crystal growth
http://bnb.data.bl.uk/id/resource/017164005	9780198738688	The Basics of Crystallography and Diffraction
http://bnb.data.bl.uk/id/resource/017164005	9780198738671	The Basics of Crystallography and Diffraction

Рис. 103. Фрагмент результатов запроса на поиск книг по тематике "кристалография" из фонда BNB

Вопросы для самопроверки

.Какие функции выполняет точка доступа (SPARQL end point)?

2. Какие типы точек доступа реализованы в WWW?

3. Каким образом сформировано RDF хранилище данных DBpedia?

4. С помощью какого запроса можно получить информацию о всех классах RDF хранилище данных DBpedia?

5. Какая информация хранится в FOAF файле?

6. Как можно определить место расположения объекта с помощью SPARQL – запроса?

7. Каким образом ресурсы FOAF связаны с ресурсами DBpedia?

8. Назовите основные функции элементов Dublin Core.

9.Какова структура словаря Dublin Core?

10. Определите набор троек запроса, определяющего место публикации ресурса.

4. Задания для практической работы

Цель выполнения заданий для практической работы состоит в приобретении навыков использования семантических веб-ресурсов. Для выполнения практических заданий используйте ресурс: http://dbpedia.org/sparql. При возникновении вопросов полезно использовать материалы книги «Изучаем SPARQL» и примеры из [7].

Задание 1

Вопрос 1. Что вычисляет этот запрос?

```
SELECT COUNT(?subject)
WHERE {?subject ?predicate ?object}
LIMIT 100
```

Вопрос 2. Какой параметр в предыдущем запросе «не работает»?
Изучите результаты SPARQL запроса:

```
SELECT ?predicate (COUNT(*)AS ?frequency)
WHERE {?subject ?predicate ?object}
GROUP BY ?predicate
ORDER BY DESC(?frequency)
LIMIT 100
```

Вопрос 3. Что подсчитано в столбце frequency? Почему предикат http://www.w3.org/1999/02/22-rdf-syntax-ns#type в первой строке таблицы?

Вопрос 4. Используя префикс PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema# составьте запрос демонстрирующий контент, определяемый предикатом rdfs:label.

Отчет по заданию 1 должен содержать:

- тексты всех составленных SPARQL запросов;
- скриншоты результатов запросов;
- ответы на вопросы 1 - 4.

Задание 2

Составьте SPARQL запрос, который находит 100 пар субъект-объект, связанных предикатом <http://purl.org/dc/terms/subject>.

Найдите все классы, используемые в dbpedia. Подсчитайте общее количество классов.

Отчет по заданию 2 должен содержать:

- тексты всех составленных SPARQL запросов;
- скриншоты результатов запросов;

Задание 3

Для объекта http://www.w3.org/2002/07/owl#Class в точке доступа http://dbpedia.org/sparql составьте SPARQL запрос показывающий 20 пар субъект-предикат.

Повторите этот запрос для получения 2000000 пар субъект-предикат.

Вопрос 1. Что означает предикат https://www.w3.org/1999/02/22-rdf-syntax-ns#type ?
Вопрос 2. Что означают субъекты?

Для трех произвольных субъектов из предыдущего результата сформируйте запрос, который выведет свойство rdfs:label.

Вопрос 3. Прокомментируйте полученные результаты.

Отчет по заданию 3 должен содержать:

- тексты всех составленных SPARQL запросов;
- скриншоты результатов запросов;
- ответы на вопросы 1 - 3.

Задание 4

Составьте SPARQL запрос, который отыщет все страны (country) мира и выведит их названия.

Пояснения:

Название страны можно прочитать в http://www.w3.org/2000/01/rdf-schema#label. Страна - это объект со следующим URI: http://dbpedia.org/ontology/Country.

Вопрос 1. Сколько всего стран найдено?

Дополните предыдущий SPARQL запрос триплетом, который для каждой страны укажет количество жителей.

Пояснение:

Используйте свойство - это http://dbpedia.org/property/populationEstimate.

Дополните предыдущий SPARQL запрос фильтром, который ограничит выбор только странами с количеством жителей более 50 миллионов.

Вопрос 2. Сколько таких стран найдено?

Отчет по заданию 4 должен содержать:

- тексты всех составленных SPARQL запросов;
- скриншоты результатов запросов;
- ответы на вопросы 1 и 2.

Задание 5

Составьте SPARQL запрос, который отыщет все страны (country) мира и выведит их названия только на русском языке.

Подсчитайте их общее количество.

Определите на каких языках мира включены определения (dbo:abstract) понятий университет (university) и компьютер (computer) в семантический веб-ресурс <http://dbpedia.org/resource/>. Перечислите эти языки.

Отчет по заданию 4 должен содержать:

- тексты всех составленных SPARQL запросов;
- скриншоты результатов запросов;
- ответы на вопросы.

СПИСОК ЛИТЕРАТУРЫ

1. RDF Primer. W3C Working Draft. http://www.w3.org/TR/rdf-primer
2. Resource Description Framework (RDF): Concepts and Abstract Syntax / Graham Klyne (ed), Jeremy J. Carroll (ed) //W3C Recommendation 10 February 2004. [Электронный ре-сурс]. – Электрон. текстовые данные. – Режим доступа: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/
3. RDFSutie Project. http://139.91.183.30:9090/RDF/
4. Язык запросов SPARQL для RDF. Рекомендация W3C, http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/
5. The Friend of a Friend (FOAF) project. [Электронный ре-сурс]. – Электрон. текстовые данные. – Режим доступа: http://www.foaf-project.org/
6. Dublin Core Activity. http://dublincore.org
7. https://wiki.base22.com/display/btg/SPARQL+Query+Examples