

## Введение

В современном мире Интернет продолжает развиваться и всё большее распространение получают связанные данные в связи с внедрением в Интернет соответствующих принципов [6]. Связность данных обеспечивается в системе взаимосвязанных открытых данных (Linked Open Data). Данная система выступает технологической основой для создания Семантического Веба (Semantic Web), делая данные в сети не только машиночитаемыми, но «машинопонимаемыми».

Для обращения к системе связанных данных (LOD) используется язык SPARQL [1]. Но для людей не знакомых с ним это делает использование LOD абсолютно невозможным (SPARQL имеет высокий порог входа). Для решения этой проблемы исследователями, работающими над данной проблемой, разрабатываются естественно-языковые интерфейсы. Они позволяют обращаться к связным данным с помощью запросов на естественном языке. Но все работы, сделанные в данной области [2-4], в качестве естественного языка предполагают использование английского или его подязыков. Поэтому, целью данной работы ставится разработка интеллектуального естественно-языкового интерфейса для обращения к LOD на русском языке.

В рамках данной работы будет рассмотрено, что такое система LOD, на каких принципах она основана, языковая система описания ресурсов RDF, использующаяся в ней для представления данных, и приведены сведения об уже существующих проектах, в которых применяется система LOD.

Также будет необходимо изучить язык запросов к системе LOD SPARQL, его синтаксис и основные конструкции.

Для выбора представления смысла запроса на естественном языке будут рассмотрены следующие подходы к формальному представлению семантической структуры текста: абстрактное представление смысла (АПС),

грамматика Монтегю и теория К-представлений (концептуальных представлений) В.А. Фомичева [5].

Будут рассмотрены подходы к разработке алгоритмов получения семантического представления текста на ЕЯ. Исследователи, использующие в своих работах семантическое представление, применяют последовательный лексический, синтаксический, семантический анализы с последующим построением семантического представления. В свою очередь в теории К-представлений В.А. Фомичев [5] предлагает новый метод построения семантического представления текста на ЕЯё. В рамках нового метода синтаксический и семантический анализ проводится одновременно и не используется синтаксический уровень представления текста. Упомянутый выше метод предполагает 3 этапа: компонентно-морфологический анализ входного текста, построение матричного семантико-синтаксического представления (МССП) и непосредственно сборка семантического представления текста по его МССП.

Потребуется рассмотрение подходов, применяемые в проектах, осуществляющих обращения к LOD на ЕЯ, для перевода запроса на ЕЯ в запрос на языке SPARQL. В частности, рассматриваются подходы, применяемые французским исследователем Себастиеном Ферре [2;3] и американскими исследователями Нели Златаревой и Девишем Амином в работе [4].

Также будет необходимо разработать алгоритмы для реализации преобразования «ЕЯ-запрос → SPARQL-запрос». Для чего потребуется разработка алгоритмов для реализации преобразований «ЕЯ-запрос → Семантическое представление» и «Семантическое представление → SPARQL-запрос».

## Глава 1

### **Краткие сведения о семантической информационной системе LOD и ее применениях**

LOD (Linked Open Data) – открытые связанные данные – система, подразумевающая использование методов публикации и связывания структурированных данных в Интернете. Основывается на четырех принципах, введенных Тимом Бернерс-Ли [6]:

1. Необходимо использование URI для объектов реального мира и абстрактных понятий, а не только для веб-документов и цифровых ресурсов;
2. Необходимо использование HTTP URI, что позволит использовать протокол HTTP для доступа к ресурсам;
3. Необходимо использование RDF (Resource Description Framework) в качестве единой модели данных для публикации данных;
4. Необходимо включать RDF утверждения на другие URI таким образом, чтобы была возможность обнаружения связанных данных.

В целом систему LOD можно представить, как огромный размеченный ориентированный граф, состоящий из элементарных графов, являющихся тройками (утверждениями) языка RDF (рисунок 1).



Рисунок 1 – Пример представления LOD

RDF (Resource Description Framework) [7] – фреймворк описания ресурсов. Стоит сказать, что описание ресурсов (автор документа, дата публикации и т.п.) – это предполагаемое использование RDF. На практике он используется (и интерпретируется) как простейший язык для создания распределённых баз знаний. RDF представляет собой совокупность утверждений о ресурсах в виде, удобном для машинной обработки.

Формальная модель RDF состоит из следующих множеств:

- Множество «Ресурсы»;
- Множество «Литералы»;
- Подмножество «Ресурсов» – «Свойства»;
- Множество «Утверждения».

Последнее множество состоит из троек вида:

(субъект, предикат, объект),

где субъект – это элемент множества «Ресурсы»,

предикат – это элемент множества «Свойства»,

объект – это элемент либо множества «Ресурсы», либо множества «Литералы».

Множество «Ресурсы» состоит из веб-адресов информационных страниц, содержащих описания соответствующих ресурсов, соответственно, множество «Свойства» – содержащих описания соответствующих предикатов. Множество «Литералы» состоит из всевозможных литералов, применение которых допустимо в описании тройки-утверждения.

Также существует языковая система RDFS [8], являющаяся надстройкой над языком RDF. Полное ее название RDF Schema Specification Language, но чаще всего используется сокращение RDF Schema (RDFS – рекурсивный акроним от RDF Schema). Система RDFS расширяет выразительные средства RDF следующими двумя возможностями: выделение и описание подклассов и указание семантических ограничений на атрибуты бинарных отношений, на аргументы и значения функций.

На основе опыта RDF и RDFS в 2004 создается язык проектирования онтологий (баз знаний) Ontology Web Language (OWL) [9]. На его основе на текущий момент разработано огромное количество онтологий по разным предметным областям.

OWL обладает двумя основными отличительными чертами, а именно: имеется много способов описания подклассов объектов и есть возможность указания свойств бинарных отношений (таких как, рефлексивность, транзитивность, симметричность и т.п.) и использования этих свойств при семантическом поиске.

Существует большое количество задач, для решения которых может быть использована система взаимосвязанных данных LOD. В число таковых входят: моделирование и публикация пространственных данных, связывание и публикация данных из научно-исследовательских областей и бизнеса, встраивание данных научных исследований в LOD. Увеличение скорости поиска и анализа информации, в частности научно-исследовательской, медицинской, туристической и бизнес-информации.

Например, LOD используется в геоинформационном направлении. Геоинформационные данные нуждаются в логической и физической взаимосвязанности. Поэтому были организованы и проведены исследования, в ходе которых осуществлялся поиск способов преодоления семантической разнородности информации об географических объектах. В ходе данных исследований было введено понятие Инфраструктуры Пространственных Данных, были разработаны стандарты для построения систем на их основе.

Проекты LinkedGeoData (LGD) и GeoNames являются примерами реализации геоинформационных данных в системе LOD [10].

Проект LGD создавался для обеспечения преобразования и интеграции данных OSM (OpenStreetMap) в инфраструктуру системы LOD. А основным применением OSM является описание данных для визуализации различных карт. Другим проектом, использующим систему LOD, является географическая база данных GeoNames. В ней хранится около десяти миллионов имен географических объектов и состоит из семи с половиной тысяч уникальных элементов доступных для отображения на картах. GeoNames также интегрирует широкий спектр географических данных, такие как названия географических мест и объектов на разных языках, значения высот или глубин, разнообразные характеристики населения и другие, из разных источников.

В Малайзии для интеграции туристических ресурсов и связанности разрозненной информации о туризме была разработана семантическая рекомендательная система по туризму в Малайзии (SMTPS) [11]. Данная система позволяет найти отель, ресторан в заданном месте, узнать об активностях, проводимых в конкретный день, а также просто получить список достопримечательностей. При этом информация, используемая для ответа на запрос, может располагаться на разных информационных ресурсах.

Использование системы LOD в бизнесе позволяет предприятием быстрее и проще получать доступ к интересующим законодательным актам, действующим на территории страны, передавать избранный контент и информацию на аутсорсинг, для поддержки специалистами соответствующей предметной области [12].

Большое количество научной информации хранится в книгах, а книги в свою очередь в библиотеках. При всем своем удобстве библиотеки представляют собой огромный массив данных, в котором отдельно взятому человеку трудно ориентироваться, не говоря уже о поиске литературы по конкретной тематике. Для того чтобы пользователи библиотек имели быстрый и максимально возможный полный доступ, была разработана Классификационная система [13]. Классификационная система обеспечивает семантическую интеграцию библиотечных данных, посредством публикации последних в виде LOD. Классификационная система работает на основе данных Российской государственной библиотеки.

## Основные конструкции языка запросов SPARQL

SPARQL (рекурсивный акроним SPARQL Protocol and RDF Query Language) – язык запросов к данным, представленным в формате RDF. SPARQL для LOD является таким же инструментом, как и SQL для реляционных баз данных. SPARQL имеет сходную с SQL форму запроса, а именно «SELECT FROM WHERE», т.е. «Что выбрать», «В каком ресурсе» и «Каким параметрам должно соответствовать» [1].

Простейший запрос SPARQL представлен в листинге 2. В качестве данных, к которым применяется запрос, используется RDF граф, приведенный в листинге 1.

### Листинг 1 – RDF-тройка

```
<http://example.org/book/book1>  
<http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
```

### Листинг 2 – Простейший запрос на языке SPARQL

```
SELECT ?title WHERE {http://example.org/book/book1  
<http://purl.org/dc/elements/1.1/title> ?title . }
```

Результатом выполнения данного запроса представлен в таблице 1.

Таблица 1 – Результат выполнения SPARQL-запроса, представленного в листинге 2.

title
"SPARQL Tutorial"

Запрос приведенный выше осуществляет поиск книги по заданному графу RDF. Запрос состоит из двух частей: части SELECT, в которой определяются переменные, которые будут отображаться в результатах запроса; и части WHERE, в которой представлен шаблон графа для сопоставления с графом данных, в данном запросе шаблон состоит из графа с одной переменной.



RDF предоставляет возможность использования литералов трех варианты литералов трех вариантов: с пометкой (тегом) языка, литерал со стандартным типом данных и литерал с произвольным типом данных (возможен еще с четвертый тип литерала – простой литерал без тегов языка и типов, но в следующих трёх примерах запросов SPARQL он рассматриваться не будет). Пример данных с использование трёх типов литералов приведены в листинге 3.

### Листинг 3 – RDF граф с использованием литералов

```
@prefix dt:    <http://example.org/datatype#> .
@prefix ns:    <http://example.org/ns#> .
@prefix :      <http://example.org/ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .

:x    ns:p    "cat"@en .
:y    ns:p    "42"^^xsd:integer .
:z    ns:p    "abc"^^dt:specialDatatype .
```

В листингах 4 и 5 приведены на первый взгляд похожие запросы, но запрос в листинге 4 не будет иметь соответствие в указанных выше данных (поэтому результат выполнения запроса будет пустым), в отличие от запроса в листинге 5, поскольку литералы «cat» и «cat@en» не являются одинаковыми. Результат выполнения запроса в листинге 5 представлен в таблице 2.

### Листинг 4 – Запрос с литералом без тега языка

```
SELECT ?v WHERE { ?v ?p "cat" }
```

### Листинг 5 – Запрос с некорректным литералом

```
SELECT ?v WHERE { ?v ?p "cat@en" }
```

Талица 2 – Результат выполнения SPARQL-запроса, представленного в листинге 5.

v
< http://example.org/ns#x>

В листинге 6 приведен пример запроса (результат запроса представлен в таблице 3) для поиска литерала со стандартным типом. При использовании в литерале стандартного типа возможно использование сокращенной формы записи, например, просто 23 вместо «"23"^^<http://www.w3.org/2001/XMLSchema#integer>». В этом случае задача определения конкретного типа возлагается на программное обеспечение реализующее выполнение запроса.

Листинг 6 – Пример запроса с литералом со стандартным типом

```
SELECT ?v WHERE { ?v ?p 42 }
```

Талица 3 – Результат выполнения SPARQL-запроса, представленного в листинге 6.

v
< http://example.org/ns#y>

Следующий пример — это запрос с литералом с произвольным типом (листинг 7). В данном случае литерал должен указываться полностью в явном виде, но при выполнении такого запроса обработчик запроса не обязан знать о указанном типе данных, поиск производится по сравнению литерала.

Листинг 7 – Пример запроса с литералом с произвольным типом

```
SELECT ?v WHERE { ?v ?p
"abc"^^<http://example.org/datatype#specialDatatype> }
```

Результат выполнения запроса представлен в таблице 4.

Таблица 4 – Результат выполнения SPARQL-запроса, представленного в листинге 7.

v
<http://example.org/ns#z>

Возможности ограничения выборки искомых ресурсов будет показано на основе данных, указанных в листинге 8.

Листинг 8 – Данные для демонстрации запросов с ограничениями на искомую выборку

```
@prefix dc:    <http://purl.org/dc/elements/1.1/> .
@prefix :      <http://example.org/book/> .
@prefix ns:    <http://example.org/ns#> .

:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```

Язык SPARQL позволяет ограничивать выборку по содержанию строковыми литералами определенных значений, по числовым литералам на основе арифметических признаков (больше, меньше и т.п.) также возможны ограничения по логическому типу и по дате (до указанной даты, после и т.п.).

В листингах 9 и 10 приведены запросы, ограничивающий выборку, по содержимому строкового литерала.

Листинг 9 – Запрос, ограничивающий выборку, ресурсами, название которых начинается с подстроки «SPARQL»

```

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

SELECT  ?title

WHERE   { ?x dc:title ?title

          FILTER regex(?title, "^SPARQL")

        }

```

Результат выполнения указанного запроса к данным из листинга 8 представлен в таблице 5.

Таблица 5 – Результат выполнения SPARQL-запроса, представленного в листинге 9.

title
"SPARQL Tutorial"

Листинг 10 – Запрос, ограничивающий выборку, ресурсами, название которых содержит подстроку «web»

```

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

SELECT  ?title

WHERE   { ?x dc:title ?title

          FILTER regex(?title, "web", "i" )

        }

```

Результат выполнения запроса к данным из листинга 8 представлен в таблице 6.

Талица 6 – Результат выполнения SPARQL-запроса, представленного в листинге 10.

title
"The Semantic Web"

В листинге 11 приведен пример запроса, ограничивающего выборку на основе значений численных литералов.

Листинг 11 – Запрос с ограничением выборки по числовому литералу

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

PREFIX ns: <http://example.org/ns#>

SELECT ?title ?price

WHERE { ?x ns:price ?price .

        FILTER (?price < 30.5)

        ?x dc:title ?title . }
```

Результат выполнения при применении запроса к данным из листинга 8 представлен в таблице 7.

Талица 7 – Результат выполнения SPARQL-запроса, представленного в листинге 11.

title	price
"The Semantic Web"	23

Запросы с ограничением по логическому типу и дате имеют структуру запроса аналогичную запросам с ограничением по числовому типу.

Язык запросов SPARQL предоставляет возможность для объединения данных в одну логическую сущность, которая будет использоваться в запросе.

Для иллюстрации данной возможности используются данные представленные в листинге 12.

Листинг 12 – Данные для демонстрации запросов с использованием объединения данных

```
@prefix dc:    <http://purl.org/dc/elements/1.1/> .

@prefix :      <http://example.org/book/> .

@prefix ns:    <http://example.org/ns#> .

:book1  dc:title  "SPARQL Tutorial" .

:book1  ns:price  42 .

:book2  dc:title  "The Semantic Web" .

:book2  ns:price  23 .
```

Пример запроса с использованием объединения данных представлен в листинге 13, а результат его выполнения в таблице 8.

Листинг 13 – Запрос с использованием объединения данных

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

PREFIX :      <http://example.org/book/>

PREFIX ns:    <http://example.org/ns#>

SELECT ?book ?title ?price {

    VALUES ?book { :book1 :book2 }

    ?book dc:title ?title ;

    ?book ns:price ?price .

}
```

Таблица 8 – Результат выполнения SPARQL-запроса, представленного в листинге 13.

book	title	price
<http://example.org/book/book1>	"SPARQL Tutorial"	42
<http://example.org/book/book2>	"The Semantic Web"	23

### **Основные современные подходы к формальному описанию семантической структуры текстов на естественном языке (ЕЯ)**

Одним из подходов к формальному описанию семантической структуры текстов на ЕЯ является Абстрактное представление смысла (АПС) [14;15]. В рамках данного подхода каждое предложение представляется в виде корневого, направленного ациклического графа с метками на ребрах (отношения между понятиями) и листьях (сами понятия). В качестве понятий в АПС используются английские слова («boy» как в примере выше), фреймы из PropBank («believe-01») или ключевые слова, включающие специальные типы сущностей: даты, региона, расстояния и др, и союзы связки: «и», «или».

АПС учитывает все слова предложения, но при этом он является абстрактным, то есть один граф может представлять несколько разных предложений на естественном языке со схожим смыслом. В качестве примера можно привести следующий набор предложений:

- The boy desires the girl to believe him;
- The boy desires to be believed by the girl;
- The boy has a desire to be believed by the girl;
- The boy's desire is for the girl to believe him.

У перечисленных выше предложений будет одинаковое описание семантической структуры (листинг 14).

## Листинг 14 – Пример АПС

```
(w / want-01
:ARG0 (b / boy)
:ARG1 (b2 / believe-01
:ARG0 (g / girl)
:ARG1 b))
```

В этом примере проявляется возможность АПС абстрагироваться от синтаксической структуры, что позволяет выделять предложения с разной синтаксической структурой, но с одинаковой семантической.

Хоть АПС и не зависит от синтаксических особенностей, но он рассчитан для описания семантической структуры предложений только на английском языке и без взаимосвязи с предыдущими предложениями.

Также существует подход к формальному описанию семантической структуры текста называемый Грамматика Монтегю [16]. В рамках данного подхода семантическая структура описывается с помощью формальных инструментов логики предикатов первого порядка и лямбда-исчисления. При разработке своей грамматики автор руководствовался собственным утверждением о том, что естественные и формальные языки в существенных своих свойствах не отличаются друг от друга, что в свою очередь позволяет описывать их семантику и синтаксис в рамках одной математической модели. В качестве примера можно привести СП предложения «there is man»:

$$S \rightarrow \text{there is NP } \{ \text{np } \lambda x.\text{true} \} \text{ “there is } [_{\text{NP}} \text{ a man}]”.$$

Данному подходу свойственны недостатки аналогичные недостаткам АПС, а именно то, что Грамматика Монтегю рассчитана на описание предложений на английском языке и она не располагает формальными инструментами для описания дискурсов. Также стоит отметить, что грамматика изложена автором в трудно воспринимаемой форме и является трудной для реализации в программном виде из-за вычислительной сложности.



Выше указанным подходам свойственна изначальная ориентация на описания высказываний (в Грамматике Монтегю еще есть возможность описывать вопросы, но этого все равно недостаточно). А как бы то не было, еще существуют выражения, соответствующие целям, действиям, определениям, назначениям вещей, поэтому должен быть и формальный аппарат, позволяющий описывать смысловую структуру таких выражений.

Последний рассматриваемый и применяемый в данной работе подход к формальному описанию семантической структуры текстов на ЕЯ – теория К-представлений (концептуальных представлений) В.А. Фомичева [5]. Данная теория описывает математическую модель, позволяющую формально описать семантическое представление текстов (дискурсов) на ЕЯ. При этом данный подход не обладает недостатками АПС и грамматики Монтегю, которые могут описать СП предложения предложений на подъязыках английского. С помощью теории К-представлений возможно описать СП текстов в первую очередь на русском, а также на английском, немецком и французском языках. Также в рамках данного подхода возможно, строить представления как фраз, выражающих высказывания, так и повествовательных текстов; представления целей (выраженных неопределенными формами глаголов с зависимыми словами) и вопросов; строить и различать обозначения единиц, соответствующих объектам, ситуациям, процессам в реальном мире и понятиям, характеризующим эти объекты, ситуации и процессы; строить и различать обозначения: объектов и множеств объектов, понятий и множеств понятий.

В качестве примера К-представления можно привести семантическое представление следующего предложения: «Сколько раз в этом году запрашивался учебник Коробова?». Для данного предложения будет построено следующее СП:

Вопрос ( $x_1$ , ( $x_1 \equiv \text{Колич-элемент}(\text{все запросы} * (\text{Время, текущий-год})$   
(Предмет-запроса, не учебник \* (Автор, не человек \* (Фамилия, “Коробов”)  
:  $x_2$ )))))).

### **Основные подходы к разработке семантически-ориентированных ЕЯ-интерфейсов для взаимодействия с прикладными интеллектуальными системами**

В работах зарубежных исследователей (B. Nethravathi et al 2020, Lee M. Christensen et al 2009, Chuan Wang et al 2015, Sylvain Pogodalla 2004) [17-20] в основном применяется совпадающий в значимых частях подход для построения семантического представления текста на ЕЯ. Он в общем виде состоит из последовательно выполняемых лексического, синтаксического, семантического анализов и непосредственного построения представления.

В ходе лексического анализа осуществляется деление текста на лексические единицы и определяются их морфологические признаки. Определение признаков проводится как с помощью запросов к разнообразным лингвистическим банкам с соответствующей информацией, так и с помощью средств лексического анализа собственной разработки.

На следующем этапе, синтаксического анализа, определяются взаимосвязи между лексическими единицами и строится синтаксическое дерево.

Далее на основе данных полученных на предыдущих этапах проводится семантический анализ, в ходе которого определяется смысловая структура входного текста.

И на основе результатов семантического анализа проводится непосредственно построение семантического представления текста.

В свою очередь Владимир Александрович Фомичев в своей научной монографии «Формализация проектирования лингвистических процессоров» в рамках своей теории К-представлений формулирует новый метод

преобразования ЕЯ-текста в СП текста [5] (Фомичев 2005). Данный метод предназначен для разработки ЕЯ-интерфейсов для взаимодействия с прикладными интеллектуальными системами. Указанный выше метод состоит из трех частей (этапов) преобразования.

На первом этапе осуществляется компонентно-морфологический анализ входного текста. По входному тексту строится не менее одного компонентно-морфологических представления (КМП), каждое из которых представляет из себя пару (набор) классифицирующего и морфологического представлений текста. Последнее представляет возможные значений морфологических признаков для лексических единиц входной текста. Чаще всего отдельным фразам соответствует единственное КМП. В ситуации неоднозначного разбиения на значащие единицы или невозможности однозначного определения части речи какой-либо единицы пользователю задаются уточняющие вопросы, ответы на которые позволяют снять неоднозначность.

Второй этап предполагает построение матричного семантико-синтаксического представления (МССП). На этом этапе с каждым словом связывается одно из возможных значений и устанавливаются смысловые отношения между единицами текста.

МССП сначала недоопределенно и неопределенность снимется шаг за шагом. Для снятия неопределенности используется информация из лингвистической базы данных (ЛДБ) о возможных способах соединения единиц текста в лингвистически допустимые сочетания.

На заключительном, третьем этапе производится сборка семантического представления текста, которое является его К-представлением, по его МССП.

С помощью описанного выше метода возможно устанавливать смысловые соотношения в сочетаниях типа: «Глагол + Предлог +

Существительное», «Глагол + Существительное», «Существительное1 + Предлог + Существительное2», «Число + Существительное», «Прилагательное + Существительное», «Существительное1 + Существительное2», «Причастие + Существительное», «Причастие + Предлог + Существительное», «Вопросительно-относительное местоимение или местоименное наречие + Глагол», «Предлог + Вопросительно-относительное местоимение + Глагол».

Довольно важно отметить, что данный метод учитывает многозначность слов, что чрезвычайно важно при проектировании ЕЯ-интерфейсов для взаимодействия с прикладными интеллектуальными системами. А также не предполагает использование синтаксического уровня представления текста (как следствие и синтаксического анализа), хоть синтаксический уровень и используется в течение долго времени как отечественными, так и зарубежными исследователями.

### **Основные подходы к разработке интеллектуальных интерфейсов для преобразования запроса к LOD на ЕЯ в запросы на языке SPARQL**

Аналогичными работа по разработке интеллектуальных интерфейсов для преобразования запроса к LOD на ЕЯ в запросы на языке SPARQL занимается французский исследователь Себастьян Ферре (Sébastien Ferré 2012, 2013). Одной из его разработок является SQUALL. Он описан в двух работах Ферре: «SQUALL: a Controlled Natural Language for Querying and Updating RDF Graphs» [2] и «SQUALL: A Controlled Natural Language as Expressive as SPARQL 1.1.» [3].

SQUALL является контролируемым языком, т.е. такой версией естественного языка, которая получена с помощью ограничения в использовании грамматической вариативности, определенных речевых оборотов. Перечисленные выше ограничения проводятся с целью устранения (уменьшения) многозначности и сложности, что в свою очередь обеспечивает

формальную логическую основу, т.е. формальные семантику и синтаксис. А язык с формальными семантикой и синтаксисом может быть сопоставлен с другим формальным языком. При разработке языка SQUALL Ферре придерживался следующих двух требований:

1. язык должен обладать выразительностью сопоставимой с выразительностью языка SPARQL;
2. обладая выразительностью SPARQL, язык должен быть высокоуровневым и иметь естественный синтаксис, абстрагированный от низкоуровневых понятий, таких как реляционная алгебра, которая используется в языке SPARQL.

Иными словами, французский исследователь разработал по факту естественно-языковой аналог SPARQL.

Для перевода из языка SQUALL в язык SPARQL используется промежуточное представление – грамматика Монтегю. Для упрощения представления и дальнейшего перевода в SPARQL некоторые конструкторы заменяются на понятия, которые они отображают.

Перевод из промежуточного языка в SPARQL осуществляется с помощью сравнения логических конструкций с конструкциями языка SPARQL.

Формулы с конструкторами переводятся в запросы SPARQL, другие типы формул переводятся в запросы обновления SPARQL. Встреча при переводе переменной предполагает создание новой переменной SPARQL. Предикаты преобразуются в фильтры SPARQL, агрегации – в агрегативные подзапросы SPARQL.

В работе «Natural Language to SPARQL Query Builder for Semantic Web Applications» [4] (Neli Zlatareva, Devansh Amin 2021) применяется подход отличный от подхода Ферре. В данной работе не разрабатывался специальный

язык, а уклон сделан на выделении необходимых для запроса на языке SPARQL смысловых структур из синтаксического дерева зависимостей.

Первым этапом является распознавание именованных сущностей, содержащихся в запросе на ЕЯ. Далее строится синтаксическое дерево зависимостей, на основе которого далее определяются предикаты, которые будут использоваться при построении запроса на языке SPARQL. Следующим шагом применяется классификатор типа запроса для определения типа запроса на ЕЯ, что далее повлияет на структуру запроса на языке SPARQL (возможны три типа запроса: единичный факт, единичный факт с конкретным типом и вопрос). На следующем шаге производится сопоставление предикатов и сущностей, полученных на предыдущих шагах, с предикатами и сущностями, определёнными в онтологии приложения. Также на этом этапе преодолеваются лексические пробелы и смысловые разрывы (появляются из-за содержательной разницы или разницы между представлением онтологии и информацией, требуемой в запросе). Для этого применяется алгоритм, обеспечивающий сравнение на основе косинусного сходства. На заключительном этапе на основе предикатов и сущностей, предоставленных на предыдущем этапе, строится запрос SPARQL.

## **Выводы**

В данной главе сделан обзор актуальности и сфер применения Системы взаимосвязанных открытых данных (LOD). Использование LOD позволяет получать требуемую информацию быстрее и точнее, чем при использовании классических способов поиска информации.

Рассмотрены основные конструкции языка SPARQL.

Проанализированы основные подходы для формального представления семантической структуры текста на естественном языке. Наиболее подходящим подходом для использования в данной работе является теория К-

представлений В.А. Фомичева, поскольку она предоставляет формальный аппарат для описания семантической структуры русского языка.

Проведен анализ подходов к разработке семантически-ориентированных ЕЯ-интерфейсов для взаимодействия с прикладными интеллектуальными системами. Для использования в данной работе был выбран новый метод преобразования текста в его семантическое представление предложенный В.А. Фомичевым.

Рассмотрены подходы к переводу запроса на естественном языке в запрос на языке SPARQL. Ни один из имеющихся подходов не обеспечивает использование запросов на русском языке.

## **Глава 2 Разработка алгоритмов для реализации преобразования вида «ЕЯ-запроса → SPARQL-запрос»**

### **Разработка логической структуры лингвистической базы данных**

Для преобразования текста на ЕЯ в его семантическое представление с помощью нового метода преобразования, предложенного В.А. Фомичевым (Формализация проектирования лингвистических процессоров, 2005) предполагается использование промежуточного представления – матричное семантико-синтаксическое представление (МССП) [5]. В начальный момент времени МССП недоопределенно и неопределенность снимается последовательно, шаг за шагом на основе информации из лингвистической базы данных (ЛБД) о возможных способах соединения единиц текста в лингвистически корректные сочетания. Для этого в ЛБД должна содержать информацию, позволяющую устанавливать возможные семантические отношения следующих сочетаний: «Глагол + Предлог + Существительное», «Глагол + Существительное», «Существительное1 + Предлог + Существительное2», «Число + Существительное», «Прилагательное + Существительное», «Существительное1 + Существительное2», «Причастие + Существительное», «Причастие + Предлог + Существительное», «Вопросительно-относительное местоимение или местоименное наречие + Глагол», «Предлог + Вопросительно-относительное местоимение + Глагол».

Для выполнения поставленных условий ЛБД должна состоять из 4 компонентов: морфологической базы данных, лексико-семантического словаря, словаря глагольно-предложных семантико-синтаксических фреймов и словаря предложных семантико-синтаксических фреймов 333.

Морфологическая база данных хранит информацию о конкретной словоформе в виде набора значений, следующих параметров: части речи, подкласса части речи, падежа, числа, рода, залога, времени, наклонения, вида,



лица, возвратности. Для одной словоформы возможно наличие нескольких наборов. Например, со словом "книги" может быть связано три набора значений морфологических признаков (если "книги" - словоформа в единственном числе, то эта словоформа находится в родительном падеже; если "книги" - словоформа во множественном числе, то она может быть, как в именительном, так и в винительном падежах).

Лексико-семантический словарь является одним из основных компонентов ЛБД, ставящим в соответствие единицам текстов единицы семантического уровня. В общем виде его структуру можно представить следующим образом: (Номер набора, Лексема, Часть речи лексемы, Сем, Семантические координаты, Комментарий).

Номер набора служит для однозначной идентификации набора, что позволит обеспечить организацию циклов в алгоритме преобразования текста в семантическое представление.

Лексема – элемент множества лексем из морфологической базы данных.

Сем – соответствующая лексеме семантическая единица. Для глаголов, причастий, деепричастий семантическая единица, связанная с соответствующим отглагольным существительным. Например, глагол "поступить" имеет два значения: поступление абитуриента в учебное заведение; поступление физического объекта на какой-то пространственный объект (например, товара на склад). Поэтому началом одного из наборов возможного лексико-семантическим словаря будет последовательность элементов (n1, поступить, глагол, поступление1), а началом другого набора - последовательность (n2, поступить, глагол, поступление2). Для прилагательного Сем представляется в виде предиката. Например, для лексемы «зеленый» семом будет (Цвет(z1, зеленый)).

Семантические координаты сущности, характеризуемой Семом, определяют различную смысловую нагрузку, которая может быть у нее, в разных контекстах. Например, если Сем – фирма, то возможными семантическими координатами может быть набор (Интеллектуальная система, Пространственный объект, Организация).

Лексико-семантический словарь может включать в себя следующие наборы:

(208, поступить, глаг, поступление1, сит, «поступить в вуз»),

(209, поступить, глаг, поступление2, сит, nil, nil, «поступил груз»),

(311, алюминиевый, прилаг, Материал(z1, алюминий), физ.об),

(358, зеленый, прилаг, Цвет(z1, зелен), физ.об),

(411, пассажирский, прилаг, Назначение(z1, перемещение1 \* (Объект1, опред множ \* (Кач-состав, человек))), дин.физ.об).

В словаре предложных семантико-синтаксических фреймов содержатся наборы, описывающие возможные семантические отношения либо между двумя связанными существительными, либо между двумя существительными, связанными предлогом.

Словарь предложных семантико-синтаксических фреймов состоит из наборов следующего вида: (Номер набора, предлог (в том числе нулевой), Возможная семантическая координата первого существительного, Возможная семантическая координата второго существительного, Падеж второго существительного, Обозначение смыслового отношения, Пример отношения (необязателен)).

В качестве примера словаря семантико-синтаксических фреймов можно привести следующие наборы (фреймы):

(1, 'от', вещество, болезнь, 2, Против1, 'таблетки от гриппа');

(2, 'от', вещество, дин.физ.об, 2, Против2, 'мазь от комаров');

(3, 'от', физическое явление, физ.об, 2, Эффект1, 'тень от дома').

## **Алгоритм построения семантического представления входного запроса на естественном языке**

### **Постановка задачи**

На вход ЕЯ-интерфейса поступает запрос. Будем рассматривать только запросы пользователя, представимые в виде

Фрагмент1 Сущ1 Предлог (возможно, пустой)

Фрагмент2 Сущ2 Фрагмент3,

где Фрагмент 1 является либо пустой цепочкой, либо последовательностью прилагательных,

Сущ1 – существительное,

Фрагмент 2 является либо пустой цепочкой, либо последовательностью прилагательных,

Сущ2 – существительное,

Фрагмент3 является либо пустой цепочкой, либо искусственным именем, либо словосочетанием, определяющим сравнение с числом (например, «меньше 50000» или «не больше 60»).

Примеры входных запросов:

Запрос1 = "одноместные многоцелевые боевые самолеты российского производства",

Запрос2 = "экспериментальные летательные аппараты Китая",

Запрос3 = "широкофюзеляжные самолеты компании Airbus",

Запрос4 = " планета с самым большим радиусом",

Запрос5 = "частные аэропорты Германии",

Запрос6 = "канадские города с населением меньше 50000".

Форму семантического представления (СП) входного запроса дает теория К-представлений В.А. Фомичева [5].

Возможным К-представлением Запроса3 является выражение  
самолёт(Тип-фюзеляжа,=,Широкофюзеляжный)  
(Производитель,=,Airbus).

### Описания вспомогательных алгоритмов

Для работы, описанных далее алгоритмов необходимы морфологические и классифицирующее представления входного запроса.

Морфологическое представление можно представить в виде двумерного массива со структурой, представленной в таблице 9.

Таблица 9 – Структура морфологического представления запроса

<b>base</b>	<b>morph</b>
в	1
самолет	2

В столбце «base» содержатся лексемы слов, входящих в запрос, а в столбце «morph» числовые коды (иными словами номера) наборов морфологических признаков, связанных с соответствующими словами из запроса.

Классифицирующее представление можно представить также в виде двумерного массива со структурой, представленной в таблице 10.

Таблица 10 – Структура классифицирующего представления запроса

<b>unit</b>	<b>tclass</b>	<b>subclass</b>	<b>mcoord</b>
в	предлог	-	1

самолет	существительное	сущ-нарицательное	2
---------	-----------------	-------------------	---

В столбце «unit» указывается словоформа, в «tclass» - обозначение части речи, в «subclass» – обозначение подкласса части речи и в «mcoord» – номер набора морфологических признаков.

### Описание функции *Dictionary-form*

**Вход:**  $R_m$  – морфологическое представление входного запроса  $Req$ ,  $R_c$  – классифицирующее представление (КлП) запроса  $Req$ ,  $p$  – целое – позиция какой-то словоформы в КлП  $R_c$ , т.е.  $1 \leq p \leq \text{lentext}$ , где  $\text{lentext} = \text{Длина}(T)$  – количествозаполненных строк двумерного массива  $R_c$ .

**Значение:** лексема (базовая форма) слова в позиции  $p$  массива  $R_c$ , т.е. слова  $R_c[p, \text{unit}]$ .

**Пример.** Пусть  $Req = \text{Запросб} = \text{"недорогое вращающееся кресло для школьника"}$ ,  $p = 5$ . Тогда  $R_c[p, \text{unit}] = \text{«школьника»}$ ,  $\text{Dictionary-form}(R_m, R_c, p) = \text{«школьник»}$ .

### Описание функции *Input-line*

**Вход:**  $\text{base}$  - лексема (базовая форма) некоторого слова,  $\text{Arls}$  – двумерный массив – проекция лексико-смантического словаря  $\text{Lsdic}$  на входной запрос.

**Значение:** целое – наименьший номер  $k$  такой строки массива  $\text{Arls}$ , что  $\text{Arls}[k, \text{lec}] = \text{base}$ , т.е. лексема  $\text{base}$  расположена в массиве  $\text{Arls}$  на пересечении строки с номером  $k$  и столбца с индексом  $\text{lec}$ .

### Описание функции *Modif-form*

**Вход:** строка вида  $R(z, b)$ , где  $R$  - бинарный реляционный символ, т.е. имя отношения с двумя атрибутами, и – имяфункции с одним аргументом.

**Значение:** строка вида  $(R, b)$  в первом случае и строка вида  $(H, d)$  во втором случае.

**Пример.**  $\text{Modif-form}(\text{Вес}(z, 3/\text{тонна})) = (\text{Вес}, 3/\text{тонна})$ .

### Описание алгоритма *Construct-sem-image*

**Вход:**  $R_m$  – морфологическое представление входного запроса  $Req$ ,  $R_c$  – классифицирующее представление (КлП) запроса  $Req$ ,  $j$  – целое – позиция какого-то прилагательного в КлП  $R_c$ ,  $m$  – целое – позиция некоторого прилагательного в КлП  $R_c$ , причем  $1 \leq j \leq m$ , и если  $j < m$ , то в позициях от  $j$  до  $m$  расположены только прилагательные.

**Выход:** output – строка.

#### Алгоритм

Начало

Цикл по  $k$  от  $j$  до  $m$

Начало attribute := Dictionary-

form( $R_m$ ,  $R_c$ ,  $k$ ) $q$  := Input-line(Arls,

attribute)

sem-item := Arls[ $q$ , sem]

Если  $k = j$  то Output := Modif-form(sem-item)

Иначе Output := Output + Modif-form(sem-item)

{Здесь + - обозначение операции конкатенации (или сцепления) строк}

кесли

конец

### Описание алгоритма *Discover-conc-relat*

**Вход:**  $R_m$  – морфологическое представление входного запроса  $Req$ ,  $R_c$  – классифицирующее представление (КлП) запроса  $Req$ , position1 – целое – позиция первого существительного Сущ1, position2 – целое – позиция второго существительного Сущ2, prep – строка – предлог, относящийся к Сущ2 (возможно, пустой предлог nil).

**Выход:** semrel – строка – обозначение семантического отношения, реализующегося в сочетании (Сущ1, prep, Сущ2); conc-noun1 – строка –

обозначение семантической единицы, ассоциированной с Сущ1 в рассматриваемом запросе; *conc-noun2* – строка - обозначение семантической единицы, ассоциированной с Сущ2 в рассматриваемом запросе.

### Алгоритм

*base1* := Dictionary-form(*Rm*, *Rc*, *position1*)

{*base1* - базовая форма (лексема) первого существительного} *base2* := Dictionary-form(*Rm*, *Rc*, *position2*)

{*base2* - базовая форма (лексема) второго существительного}

Пусть *narfrp* – количество строк в массиве *Arfrp*. Двумерный массив *Arfrp* строится по словарю предложных семантико-синтаксических фреймов *Frp*.

Массив *Arfrp* содержит все шаблоны (или фреймы) из *Frp*, «привязанные» к предлогу *prep*.

Например, по запросу «НЕДОРОГОЙ КРЕМ ОТ КОМАРОВ» может быть построен следующий массив *Arfrp* с количеством строк *narfrp* = 4:

(1, 'от', вещество, болезнь, 2, Против1, 'таблетки от гриппа');

(2, 'от', вещество, дин.физ.об, 2, Против2, 'мазь от комаров');

(3, 'от', физическое явление, физ.об, 2, Эффект1, 'тень от дома');

(4, 'от', изделие, модельер, 2, Дизайн, 'сумка от Valentino').

В цикле по номеру *m* строки массива *Arfrp* (*m* изменяется от 1 до *narvfr*), где строка *m* является записью упорядоченного набора вида

**(*m*, *prep*, *sort1*, *sort2*, *grc*, *rel*, *expl*),**

выполняются следующие действия:

### *Действие 1:*

В цикле по строкам двумерного массива *Arls* (проекция лексико-семантического словаря *Lsdic* на входной запрос), состоящего из записей упорядоченных наборов вида

$$(i, lec, pt, sem, st_1, \dots, st_k, commelentext) ,$$

находится очередная строка с номером *line1*, для которой *base1* = *lec*.

Затем проверяется следующее Условие 1: среди сортов, *st<sub>1</sub>, ..., st<sub>k</sub>* найдется сорт, являющийся конкретизацией сорта *sort1* (в частности, такой сорт может совпадать с *sort1*).

### *Действие 2:*

Аналогичное Условие 2 проверяется для *base2* (т.е. для базовой формы второго существительного) во вложенном цикле по строкам двумерного массива *Arls*.

### *Действие 3:*

(Выполняется в случае, когда Условие 1 истинно и Условие 2 истинно для некоторой строки массива *Arls* с номером *line2*):

Проверяется, может ли Сущ2 быть в грамматическом падеже с числовым кодом *grc*.

Если ДА, то в выходную строку *semrel* рассматриваемого алгоритма добавляется обозначение смыслового отношения *rel* из рассматриваемой строки с номером *m* массива *Arfrp*.

Кроме того, выполняются присваивания

*conc-noun1* := *Arls*[*line1*, *sem*],

*conc-noun2* := *Arls*[*line2*, *sem*].



## Описание головного модуля целевого алгоритма

**Вход:** T – входной запрос на ЕЯ, Lingb – лингвистический базис

**Выход:** Semrepres – возможное K-представление входного запроса.

### Алгоритм SemParsing

Начало

Построить компонентно-морфологическое представление входного запроса, т.е. построить Rm – морфологическое представление входного запроса и Rc -классифицирующее представление (КлП) входного запроса.

lentext := Длина (T) – количество элементарных значащих единиц входного текста (совпадает с количеством заполненных строк КлП Rc).

Построить двумерные массивы Arls, Arfrp – проекции на входной запрос T соответственно лексико-семантического словаря Lsdic и словаря предложносемантико-синтаксических фреймов Fgr.

В цикле по k от 1 до lentext построить одномерный массив wordnouns, состоящий из существительных входного запроса, и сформировать значение целочисленной переменной numbwordnouns – количество существительных во входном запросе.

{Комментарий. Тогда wordnouns [1] – позиция первого существительного в запросе, wordnouns [2] - позиция второго существительного в запросе и т.д.}

Если (numbwordnouns < 1) или (numbwordnouns > 2)

То Вывод («Неправильный запрос»)

Иначе вызвать алгоритм TwoWordnounsRequestsParsing

Кесли

конец

## Описание основной подсистемы головного модуля

### Алгоритм NounOneNounTwoConnection

**Вход:** Rm – морфологическое представление входного запроса Req, Rc – классифицирующее представление (КлП) запроса Req, wordnouns – массив позиций существительных в запросе.

**Выход:** Semrepres – построенное семантическое представление.

**Условие вызова:** numbwordnouns = 2

#### Алгоритм

Начало

position1 := wordnouns[1],

position2 := wordnouns[2]

{Комментарий. Здесь position1 и position2 – позиции во входном запросе соответственно первого и второго существительных}

posprep := 1 + position1

если Rc[posprep, tclass] ≠ предлог, то prep := 'nil'

иначе prep := Rc[posprep, unit]

{Комментарий. Таким образом, значением переменной prep является предлог, относящийся ко второму существительному. Если же такого предлога нет, то prep присваивается значение nil (пустой предлог)}.

Find-sem-role(position1, position2, prep, role, conc-noun1, conc-noun2)

Если position1 = 1 {Т.е. нет прилагательных перед первым существительным}

To description1 := conc-noun1

Иначе

Construct-sem-image(Rm, Rc, 0, position1 – 1, Characteristics1)

description1 := conc-noun1 + '\*' + Characteristics1

Кесли

Semrepres := description1

{Здесь + - обозначение операции конкатенации (или сцепления) строк}

Если position2 - posprep = 1 {Т.е. нет прилагательных между предлогом и вторым существительным}

To description2 := conc-noun

Иначе

Construct-sem-image(Rm, Rc, posprep + 1, position2 – 1, Characteristics2)

description2 := conc-noun2 + '\*' + Characteristics2

Кесли

Если после существительного в позиции pos2 следует искусственное имя artif-name ("Airbus", "Тойота" и т.д.)

To description2 := description2 + '\*' (Назв,' + artif-name +')'

Кесли

Если после существительного в позиции pos2 следует словосочетание сравнения “меньше 50000”, “больше 60” и т.п.

To description2 := description2 + “(” + <название отношения (“меньше”, “больше»)> + “,” + <значение, с которым проводится сравнение (“50000”, “60”)> + “)”

Semrepres := Semrepres + (role, description2)

Вернуть Semrepres

Конец

## **Алгоритм построения SPARQL-запроса по К-представлению входного запроса**

### **Постановка задачи**

На вход генератора SPARQL-запросов поступает семантическое представление входного запроса на естественном языке. Входное семантическое представление имеет следующую структуру:

$A (B_1, R_1, C_1) (B_2, R_2, C_2) \dots (B_n, R_n, C_n),$

где  $A$  – обозначение понятия на русском языке (самолёт, автомобиль, компания и т.д.),

$B_1, B_2, \dots, B_n$  – имена смысловых параметров представления на русском языке,

$R_1, R_2, \dots, R_n$  – имена бинарных отношений на русском языке,

$C_1, C_2, \dots, C_n$  – обозначения значения параметра или второго атрибута отношения на русском языке.

На основе поступившего на вход представления строится SPARQL-запрос, где  $A$  определяет тип искомых сущностей, а из троек  $(B, R, C)$  определяются значения параметров, которыми должны обладать искомые сущности.

Примеры семантических представлений, которые могут быть на входе:

Представление1 = "самолёт (Экипаж, =, 1) (Тип, =, Многоцелевой боевой) (Производитель, =, Россия) ",

Представление2 = "лет-аппарат (Тип, =, экспериментальный) (Принадлежность, =, Китай) ",

Представление3 = "самолёт (Тип-фюзеляжа, =, Широкий) (Производитель, =, Airbus)",

Представление4 = "планета (Радиус, =, #макс#) ",

Представление5 = "аэропорт (Тип-Аэропорта, =, Частный) (Расположение, =, Германия)",

Предствление6 = "город (Страна, =, Канада) (Население, меньше, 50000)".

Пример SPARQL запроса для Представления1 представлен в листинге 14.

Листинг 14 – SPARQL-запрос, построенный на основе Представления1

```
select distinct ?var1
```

```

where {

    values ?var2 {dbo:Aircraft} .

    ?var1 rdf:type ?var2 .

    values ?p3 {dbp:crew} .

    values ?v3 {1} .

    ?var1 ?p3 ?v3 .

    values ?p4 {dbo:type} .

    values ?v4 {dbr:Multirole_combat_aircraft} .

    ?var1 ?p4 ?v4 .

    values ?p5 {dbo:manufacturer dbo:origin} .

    values ?v5 {dbr:Russia} .

    ?var1 ?p5 ?v5 .

}

```

### **Проблема неоднозначности имен предикатов**

В ходе разработки алгоритма преобразования запроса на естественном языке в запрос на языке SPARQL пришлось столкнуться с рядом проблем, потребовавших дополнительного исследования.

В рамках одной онтологии возможно одновременное наличие различных именовании одного и того же предиката, что не позволяет делать унифицированные по структуре запросы на языке SPARQL для запросов на естественном языке даже с одинаковой структурой. Например, в онтологии DBpedia в информации о городе предикат, связывающий данный город с количеством жителей, может иметь одно из следующих именовании: «population», «populationTotal», «p», «pop2010census» (пример представлен в таблице 11). Причем предикат «p» при описании города может использоваться как в значении «количество населения», так и в значении «название района

города». Описанная выше проблема делает затруднительным программное построение запроса на языке SPARQL.

Таблица 11 – Предикаты «Население» у разных городов

Город	Предикат
Оттава	population
Москва	populationTotal
Ульяновск	p
Северодвинск	pop2010census

Проанализировав работы других исследователей, можно выделить два применяемых способа решения описанной выше проблемы. Первый заключается в создании собственной онтологии, зачастую на основе данных других онтологий, но со своими классами объектов и предикатами. Поскольку такая онтология создается непосредственно для целевой системы, в нее закладываются необходимая семантическая структура и система имен, что позволяет унифицировать генерацию запроса на языке SPARQL. Но данный подход требует больших затрат времени на разработку собственной онтологии, поэтому в рамках данной работы применяться не будет.

Также в проектах, целью которых является обращение к системе LOD на естественном языке, применяются онтологии, использующие строго описанный набор классов и предикатов. Примером такой онтологии является YAGO, система классов и типов которой основана на наборе классов и предикатов Schema (специальная онтология, описывающая логическую структуру семантической связанности объектов реального мира). Данная онтология предоставляет возможность генерации унифицированных по структуре запросов на языке SPARQL. Но онтология YAGO содержит большое количество слабо связанных данных, т.е. некоторая часть данных об объектах реального мира просто не указана или указана не корректно. Например, в описании городов связь со страной, в которой находится данных город, указывается с помощью предиката комментария, значение которого

является строка приблизительно следующего содержания: «город в России» или «Казахстанский город». Хотя, придерживаясь принципов LOD, необходимо было сделать предикат, например, с название «страна», значением которого являлась бы ссылка (URI) на описание необходимой страны.

### **Принципы преобразования параметров запросов к LOD**

Для преодоления проблемы неоднозначности именования параметров (отношений и их значений) в онтологии, к которой предназначается запрос, в рамках данной работы использовался описанный далее подход.

Суть упомянутого подхода заключается в предварительном связывании параметров, использующихся в исходном запросе, с соответствующими параметрами, используемыми в онтологии. Для этого в рамках базы данных организуется таблица связи, которая будет содержать информацию о связи параметров и отношений К-представления и онтологии (пример таблицы связи представлен в таблице 2). При этом, одному параметру исходного запроса может быть поставлено в соответствие несколько параметров, использующихся в онтологии. Например, отношению «Колич-Жителей», обозначающему количество жителей конкретного города (страны и т.п.), в соответствие могут быть поставлены отношения «population», «populationTotal», «p» и «pop2010census» онтологии DBpedia. А значению «Россия» – «dbr:Russia».

Таблица 2 – Связь параметров в К-представлении с соответствующими параметрами в онтологии

<b>Параметр в К-представлении</b>	<b>Параметр в онтологии</b>
Колич-Жителей	population
	populationTotal
	p
	pop2010census
Россия	dbr: Russia

При построении SPARQL-запроса по К-представлению запроса на естественном языке для перевода параметров запроса в термины онтологии организуется запрос к базе данных, содержащей таблицу связи параметров, возвращающий все возможные параметры, используемые в онтологии, соответствующие данному параметру исходного запроса. Далее, с помощью специальной конструкции языка SPARQL из них образуется множество, выступающее в запросе самостоятельной смысловой единицей. Например, отношения «population», «populationTotal», «р» и «pop2010census» в запросе будут выступать не в качестве отдельных отношений, а в качестве одного отношения со значением «количество жителей».

### **Описание вспомогательных алгоритмов для построения SPARQL-запроса**

#### **Описание функции *isSpeacialConst***

**Вход:** maybeConst – строка, которую необходимо проверить, не является ли она специальной константой

**Значение:** логическое значение: Истина, если строка является специальной константой, иначе Ложь

**Пример:** isSpecialConst(“#макс#”) = Истина  
isSpecialConst(“самолет”) = Ложь.

#### **Описание функции *sortOrder***

**Вход:** const – специальная константа («#макс#» или «#мин#»).

**Значение:** строка, определяющая порядок сортировки. В случае константы «#макс#» – “desc”, «#мин#» – “asc”.

**Пример:** sortOrder(“#макс#”) = “desc”,  
sortOrder(“#мин#”) = “asc”.

#### **Описание функции *Sign***

**Вход:** signName – название отношения R, определяющего знак («меньше», «больше», «не меньше» или «не больше»).



**Значение:** строка-обозначение знака сравнения.

**Пример:** Sign(“меньше”) = “<”,

Sign(“больше”) = “>”, Sign(“не меньше”) = “>=” или Sign(“не больше”) = “<=”.

### **Описание функции *translate***

**Вход:** input – название или значение параметра К-представления, для которого необходимо найти соответствующие в онтологии,  
context – модуль разрешения имен.

**Значение:** список соответствующих названий или значений параметра, использующих в онтологии

**Пример:** translate(“Колич-жителей”) = [“dbo:p”, “dbo:population”, “dbo:populationTotal”, “dbo:pop2010census”].

### **Описание функции *createValues***

**Вход:** values – список обозначений параметров и значений, используемых в онтологии.

**Значение:** строка с перечислением обозначений в формате “{Обозначение1 Обозначение2 ... ОбозначениеN}”

**Пример:** createValues([dbo:plane, dbo:Aircraft]) = “{dbo:plane dbo:Aircraft}”.

### **Описание функции *createHeader***

**Вход:** values – список обозначений понятия А, используемых в онтологии.

**Значение:** строка заголовка SPARQL-запроса

**Пример:** createHeader([dbo:plane, dbo:Aircraft]) = “select distinct ?var1 where {values ?var2 {dbo:plane dbo:Aircraft} . ?var1 rdf:type ?var2 .”

Т.е. с помощью этой функции задается тип искомой сущности в SPARQL-запросе.

### **Описание функции *createEqTriple***

**Вход:** predicates – список обозначений предикатов, используемых в онтологии и соответствующих конкретному отношению К-представления,  
valuesPredicate – список обозначений значения, которое могут принимать предикаты, указанные в predicates,

numPredicate – номер тройки, создаваемой в данной функции (необходим для создания переменных уникальных для этой тройки).

**Значение:** строка, описывающая тройку, в которой устанавливается соответствие значений предикатов целевым значениям.

**Пример:** `creatyEqTriple([dbo:origin, dbo:country], [dbr:Russia], 2) = “values ?p2 { dbo:origin dbo:country } . values ?v2 { dbr:Russia } . ?var1 ?p2 ?v2 .”`

### **Описание функции *createEqTripleWithSortVar***

**Вход:** predicates – список обозначений предикатов, используемых в онтологии и соответствующих конкретному отношению К-представления, numPredicate – номер тройки, создаваемой в данной функции (необходим для создания переменных уникальных для этой тройки).

**Значение:** строка, описывающая тройку, в которой устанавливается соответствие значений предикатов переменным, по которым будет проводится сортировка.

**Пример:** `createEqTripleWithSortVar([dbo:radius, dbo:r], 4) = “values ?p4 { dbo:radius dbo:r } . ?var1 ?p4 ?var4 .”`

### **Описание функции *createCompareTriple***

**Вход:** predicates – список обозначений предикатов, используемых в онтологии и соответствующих конкретному отношению К-представления, numPredicate – номер тройки, создаваемой в данной функции (необходим для создания переменных уникальных для этой тройки), comparisonSign – знак сравнения, comparisonValue – строка, содержащая значение, которым необходимо сравнивать).

**Значение:** строка, описывающая тройку, в которой устанавливается сравнение значений предикатов с целевым значением.

**Пример:** `createCompareTriple([dbo:radius, dbo:r], 5, “<”, “10000”) = “values ?p5 { dbo:radius dbo:r } . ?var1 ?p5 ?var5 . filter (?var5 < 10000) .”`

## Описание основного алгоритма построения SPARQL-запроса

### Описание алгоритма buildSparql

**Вход:** repr – семантическое представление входного запроса вида A (B1, R1, C1) (B2, R2, C2) ... (Bn, Rn, Cn),

context – модуль разрешения имен.

**Значение:** SPARQL-запрос, построенный на основе входного семантического представления.

#### Алгоритм:

Начало

Входное семантическое представление разбирается на составные части ().

entity = A

triples = список троек (B, R, C)

translatingEntity = translate(entity, context)

requestHeader = createHeader(translatingEntity)

numVar = 3; {Комментарий: переменные с номерами 1 и 2 используются в заголовке запроса}

requestBody = ""

В цикле по списку троек triples выполняются следующие действия:

Действие 1:

Текущая тройка из строки преобразуется в массив triple следующим образом

triple[0] = B

triple[1] = R

triple[2] = C

Действие 2:

Подготовка значений предиката.

Если triple[2] возможно преобразовать в число

To translatingValues = triple[2]

Кесли

Если triple[2] является специальной константой (например, #макс#)

To sorting = sortOrder(triple[2]) + “(?s” + numVar + “)”

Иначе translatingValues = translate(context, triple[2])

Кесли

Действие 3:

translatingPredicate = translate(context, triple[0])

Если triple[1] = “=”

То

Если triple[2] является специальной константой

To requestBody = requestBody +  
createEqTripleWithSortVar(translatingPredicate, numVar)

Иначе requestBody = requestBody + createEqTriple(translatingPredicate,  
translatingValues, numVar)

Кесли

Иначе

requestBody = requestBody + createCompareTriple(translatingPredicate, numVar,  
Sign(triple[1]), triple[2])

Кесли

numVar = numVar + 1

Конец цикла

request = requestHeader + requestBody + “}”

Если sorting не пустая строка

To request = request + sorting + “limit 1”

Кесли

Вернуть request

Конец

## **Выводы**

В данной главе были разработаны алгоритмы для реализации преобразования вида «ЕЯ-запрос → SPARQL-запрос».

Для этого была разработана логическая структура лингвистической базы данных. ЛБД состоит из трех компонент: морфологической базы данных, лексико-семантического словаря и словаря предложных фреймов.

Разработан описан алгоритм построения К-представления входного запроса на русском языке.

Разработан и описан алгоритм построения SPARQL-запроса по семантическому представлению (К-представлению) входного запроса на русском языке. В ходе разработки данного алгоритма пришлось столкнуться с проблемами неоднозначности и недостаточной связанности онтологий, потребовавшими дополнительного исследования. Для преодоления указанных проблем был предложен принцип преобразования параметров запросов к LOD.

## Глава 3

### Используемые средства разработки

В качестве средства разработки для приложения была выбрана платформа .NET (использовалась версия .NET5), на ее основе предоставляются простые в использовании и при этом отличающиеся большим спектром возможностей средства для разработки оконных приложений; приложений, работающих с базами данных и приложений, реализующих выполнение SPARQL-запросов.

В качестве языка программирования был выбран C#, как основной объектно-ориентированный язык платформы .NET.

Для разработки оконного интерфейса была выбрана система построения клиентских приложений WPF. WPF на данный момент является самым современным и широко используемым средством разработки.

Для реализации обращения к базам данных используется компонент .NET LINQ и LINQ to Entities, которые предоставляют возможности написания запросов к базам данных, корректность которых можно проверить на этапе компиляции и статического анализа. Это достигается за счет интеграции LINQ в язык C#.

Для определения морфологических признаков используется библиотека DeerpMorphy. Ее работа основана на нейронной сети, и она предоставляет возможности определения всех морфологических признаков (части речи, падежа, рода, лица), а также начальной формы (лексемы).

Для реализации лингвистической базы данных выбрана СУБД PostgreSQL, по не скольким причинам: есть ее реализации от отечественных разработчиков, также она предоставляет возможность создавать базы данных неограниченного размера и обеспечивает целостность данных на высоком уровне.

## Описание реализованной схемы лингвистической базы данных

### Морфологическая база данных

Для морфологической базы данных, схема представлена на рисунке 1.

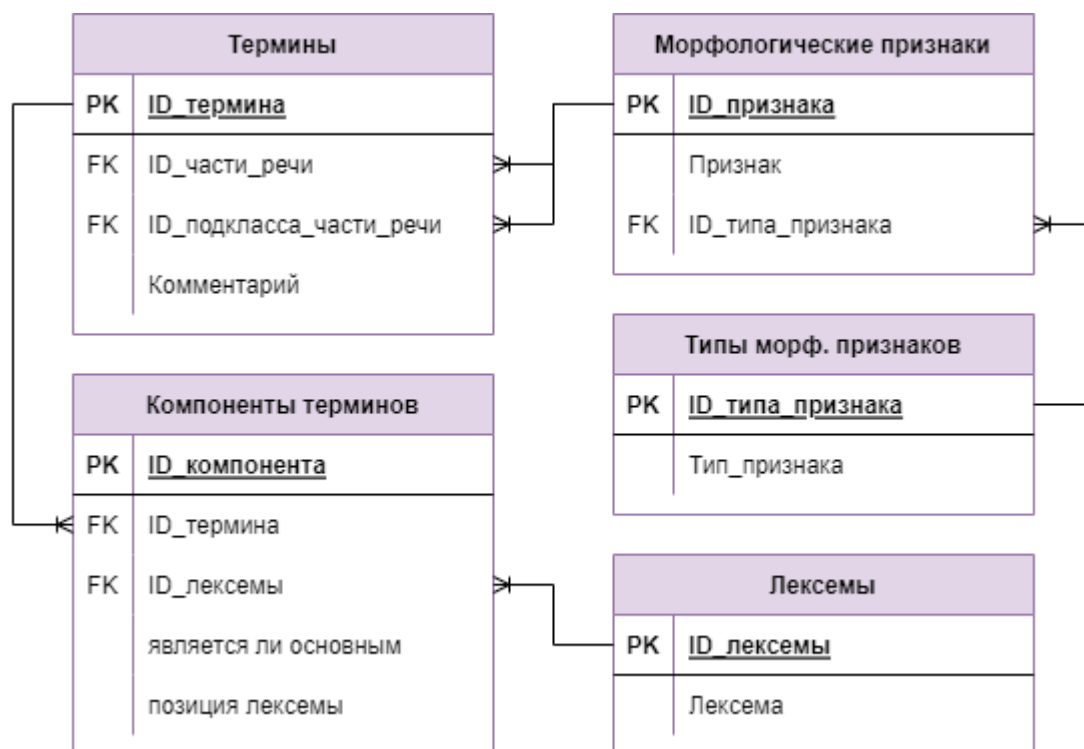


Рисунок 1 – Схема морфологической базы данных

### Описание таблиц МБД

#### Таблица «Лексемы»

Содержит список лексем (уникальных).

Поля таблицы «Лексемы»:

- ID\_лексемы – идентификатор конкретной части речи (Тип данных: целое число);
- Лексема – конкретная уникальная лексема (Тип данных: строка переменного размера).

### **Таблица «Компоненты терминов»**

Таблица содержит компоненты терминов и информацию о них (является ли главным, порядковый номер компонента в термине).

*Поля таблицы «Компоненты терминов»:*

- ID\_компонента – идентификатор конкретного компонента (Тип данных: целое число);
- ID\_термина – идентификатор термина, в который входит данный компонент (Тип данных: целое число);
- ID\_лексемы – идентификатор конкретной лексемы, являющейся компонентом термина (Тип данных: целое число);
- является\_ли\_главным – определяет является ли данный компонент главным в термине (Тип данных: логический тип);
- позиция\_компонента – порядковый номер компонента в термине (Тип данных: целое число).

### **Таблица «Термины»**

Таблица содержит список терминов и информацию о них (часть речи и подкласс части речи).

*Поля таблицы «Термины»:*

- ID\_термина – идентификатор термина, в который входит данный компонент (Тип данных: целое число);
- ID\_лексемы – идентификатор конкретной лексемы, являющейся компонентом термина (Тип данных: целое число);
- ID\_части\_речи – идентификатор конкретной части речи (Тип данных: целое число);
- ID\_подкласса\_части\_речи – идентификатор конкретного подкласса части речи (Тип данных: целое число);
- Комментарий – содержит пример использования данного термина.



### **Таблица «Типы морфологических признаков»**

Содержит список типов морфологических признаков (часть речи, подкласс части речи, падеж, склонение и т.п.).

*Поля таблицы «Типы морфологических признаков»:*

- ID\_типа\_признака – идентификатор конкретной словоформы (Тип данных: целое число);
- Тип\_признака – название типа признака (Тип данных: строка переменного размера).

### **Таблица «Морфологические признаки»**

Содержит список возможных значений морфологических признаков (родительный, 1 склонение и т.п.).

*Поля таблицы «Морфологические признаки»:*

- ID\_признака – идентификатор значения признака (Тип данных: целое число);
- Признак – значение признака (Тип данных: строка переменной длины);
- ID\_типа\_признака – идентификатор типа признака (Тип данных: целое число).

#### Лексико-семантический словарь

Схема базы данных для лексико-семантического словаря представлена на рисунке 2.

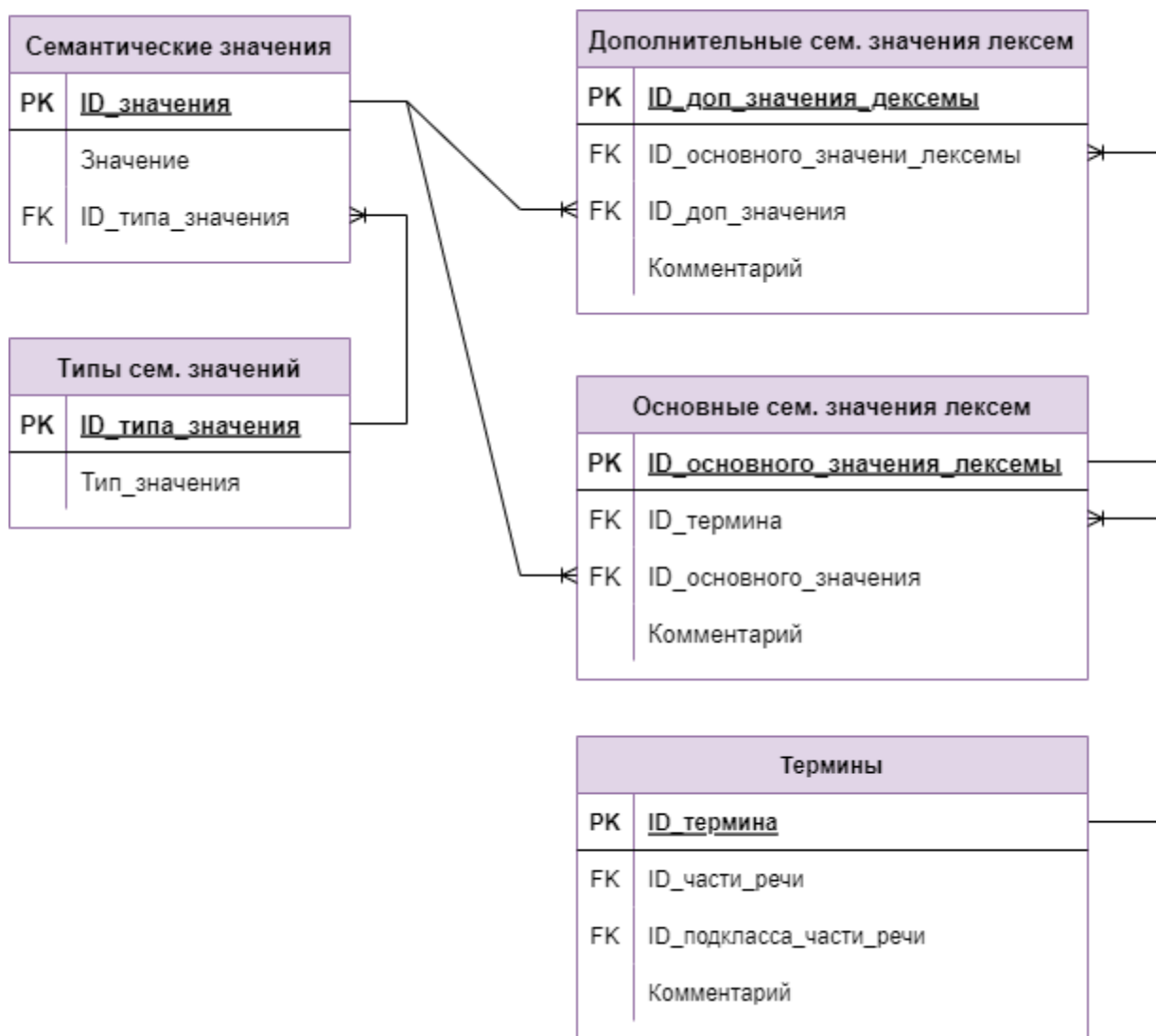


Рисунок 2 – Схема лексико-семантического словаря

Описание таблиц лексико-семантического словаря

Таблица «Термины» является частью МБД и описана в предыдущем разделе.

**Таблица «Типы семантических значений»**

Содержит набор возможных типов семантических значений.

*Поля таблицы «Типы семантических значений»:*

- ID\_типа\_значения – идентификатор типа семантического значения (Тип данных: целое число);
- Тип\_значения – название типа семантического значения (Тип данных: строка переменной длины).

### **Таблица «Семантические значения»**

Содержит набор возможных семантических значений разных типов (основное значение, дополнительное значение, тематическая роль и т.п.).

*Поля таблицы «Семантические значения»:*

- ID\_значения – идентификатор семантического значения (Тип данных: целое число);
- ID\_типа\_значения – идентификатор типа семантического значения (Тип данных: целое число).

### **Таблица «Основные семантические значения лексем»**

Содержит связь лексем с их основными семантическими значениями.

*Поля таблицы «Основные семантические значения лексем»:*

- ID\_основного\_значения\_лексем – идентификатор значения конкретной лексем (Тип данных: целое число);
- ID\_термина – идентификатор конкретного термина (Тип данных: целое число);
- ID\_основного\_значения – идентификатор основного семантического значения (Тип данных: целое число);
- Комментарий – необязательное поле, в котором может содержаться пример использования данной лексем с заданным семантическим значением.

### **Таблица «Дополнительные семантические значения лексем»**

Содержит связь лексем с их дополнительными семантическими значениями.

*Поля таблицы «Дополнительные семантические значения лексем»:*

- ID\_доп\_значения\_лексем – идентификатор дополнительного значения конкретной лексем (Тип данных: целое число);

- ID\_основного\_значения\_лексемы – идентификатор лексемы с определенным основным семантическим значением (Тип данных: целое число);
- ID\_дополнительного\_значения – идентификатор дополнительного семантического значения (Тип данных: целое число);
- Комментарий – необязательное поле, в котором может содержаться пример использования данной лексемы с заданным семантическим значением.

### Словарь предложных фреймов

Схема базы данных для словаря предложных фреймов представлена на рисунке 3.



Рисунок 3 – схема словаря предложных фреймов

### Описание таблиц словаря предложных фреймов

Таблицы «Термины» и «Морфологические признаки» описаны в рамках морфологической базы данных. Таблицы «Семантические значения» и

«Типы семантических значений» описаны в рамках лексико-семантического словаря.

### **Таблица «Предложные фреймы»**

Содержит описания предложных фреймов.

*Поля таблицы «Предложные фреймы»:*

- ID\_предложного\_фрейма – идентификатор предложного фрейма (Тип данных: целое число);
- ID\_лексема\_предлога – идентификатор термина предлога (Тип данных: целое число);
- ID\_сем\_значения\_до\_предлога – идентификатор дополнительного значения, которое может принимать слово расположенное перед предлогом (Тип данных: целое число);
- ID\_сем\_значения\_после\_предлога – идентификатор дополнительного значения, которое может принимать слово расположенное после предлога (Тип данных: целое число);
- ID\_падеж\_слова – идентификатор падежа, в котором должно находиться слово, расположенное после предлога (Тип данных: целое число);
- ID\_сем\_значение\_фрейма – идентификатор семантического значения фрейма (Тип данных: целое число);
- Комментарий – необязательное поле, в котором может содержаться пример использования данного фрейма.

#### *Компонент разрешения имен*

В следующих разделах будет описана проблема неоднозначности именования параметров в онтологиях, обнаруженная при разработке алгоритма, и данный компонент содержит информацию необходимую для преодоления указанной проблемы при построении запроса. Схема этого компонента представлена на рисунке 4.



Рисунок 4 – Схема компонента разрешения имен параметров  
Описание таблиц компонента разрешения имен

### Таблица «Параметры в К-представлении»

Содержит список параметров, используемых в К-представлении.

*Поля таблицы «Параметры в К-представлении»:*

- ID\_параметра\_в\_представлении – идентификатор параметра, используемого в К-представлении (Тип данных: целое число);
- Параметр\_в\_представлении – параметр, используемый в К-представлении (Тип данных: строка переменной длины).

### Таблица «Параметры в онтологии»

Содержит список параметров, используемых в онтологии.

*Поля таблицы «Параметры в онтологии»:*

- ID\_параметра\_в\_онтологии – идентификатор параметра, используемого в онтологии (Тип данных: целое число);

- Параметр\_в\_онтологии – параметр, используемый в онтологии (Тип данных: строка переменной длины).

### Таблица «Связь параметров»

Содержит информации о связи между параметрами К-представления и онтологии.

Поля таблицы «Связи параметров»:

- ID\_связи – идентификатор конкретной связи параметров (Тип данных: целое число);
- ID\_параметра\_в\_представлении – идентификатор параметра, используемого в К-представлении (Тип данных: целое число);
- ID\_параметра\_в\_онтологии – идентификатор параметра, используемого в онтологии (Тип данных: целое число).

## Общая схема лингвистической базы данных

Общая схема всей ЛБД представлена на рисунке 5.

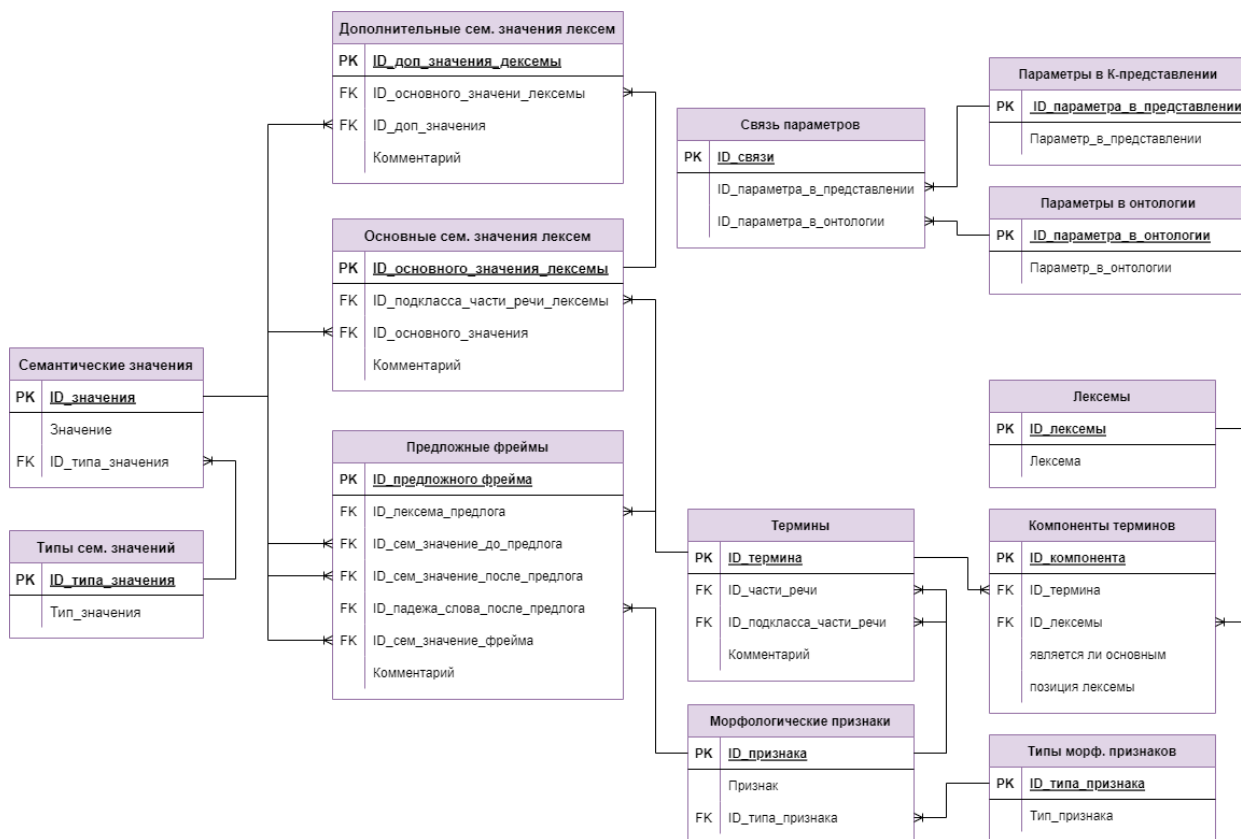


Рисунок 5 – Общая схема лингвистической базы данных

## Интерфейс приложения

Внешний вид разработанного приложения представлен на рисунке 6.

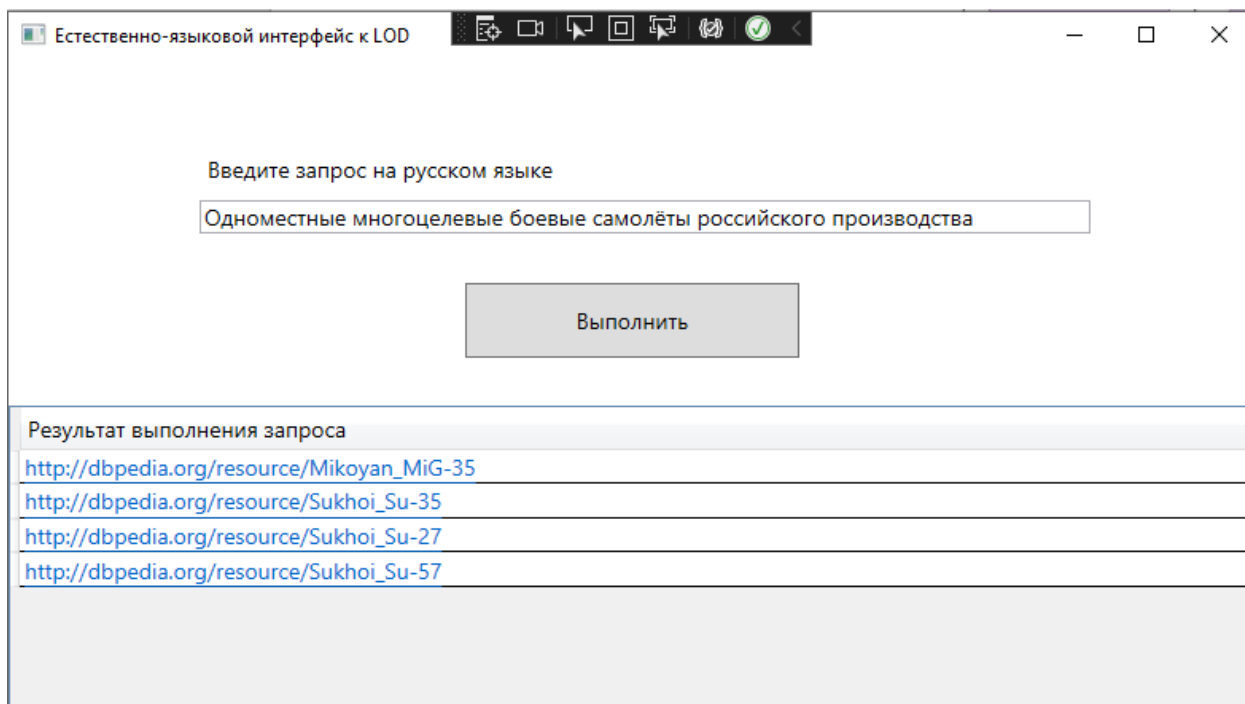


Рисунок 6 – Интерфейс приложения

Интерфейс приложения содержит поле для ввода запроса («Введите запрос на русском языке»), таблицу для вывода результатов выполнения запроса («Результат выполнения запроса») и кнопку «Выполнить», после нажатия пользователем которой будет выполнено преобразование «Запрос на ЕЯ» → «SPARQL-запрос».

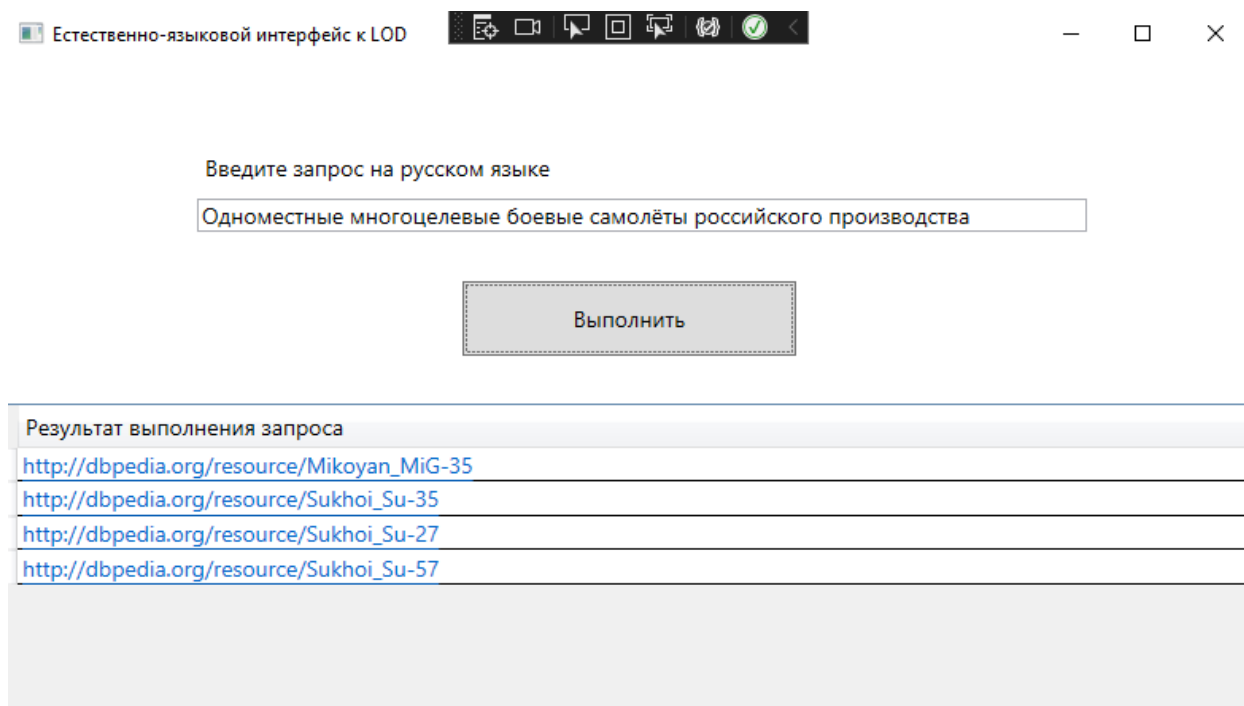
## Работоспособность приложения

Разработанное приложение рассчитано на выполнение следующих запросов (рисунки 7 - 12):

1. "Одноместные многоцелевые боевые самолёты российского производства",
2. "Экспериментальные летательные аппараты Китая",
3. "Широкофюзеляжные самолёты компании Airbus",
4. "Планета с самым большим радиусом",
5. "Частные аэропорты Германии",



## 6. “Канадские города с населением меньше 50000”.



Естественно-языковой интерфейс к LOD

Введите запрос на русском языке

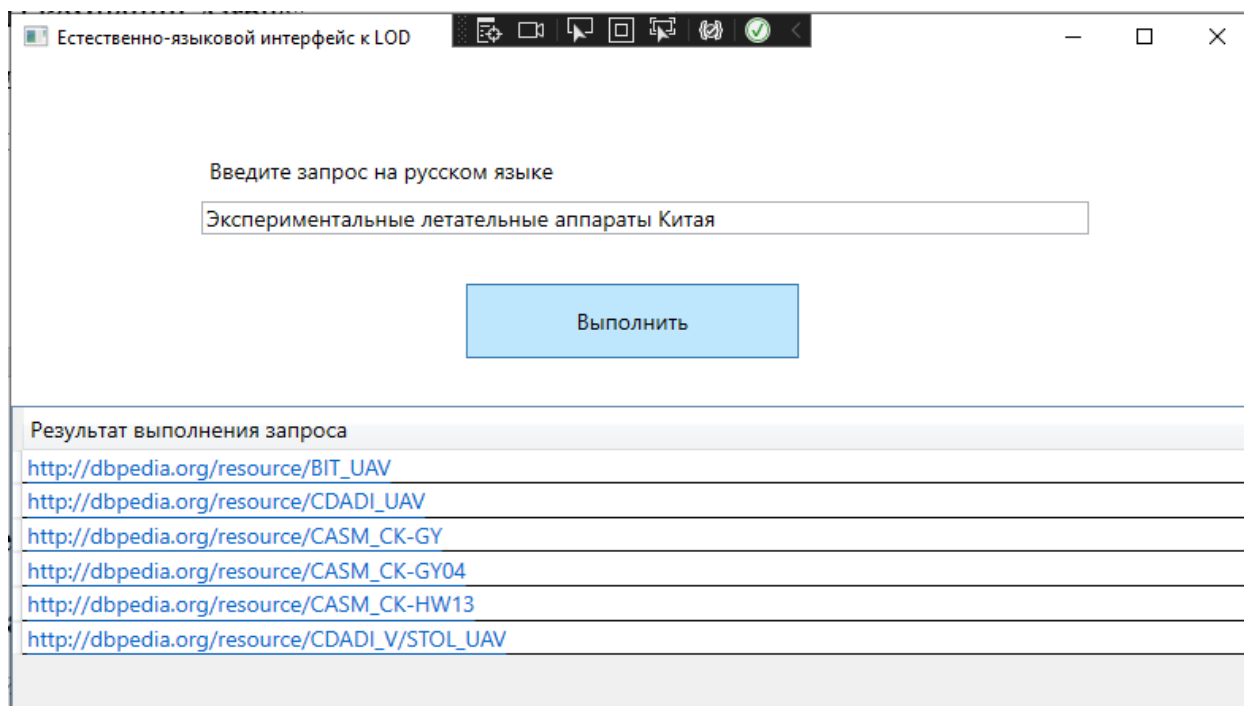
Одноместные многоцелевые боевые самолёты российского производства

Выполнить

Результат выполнения запроса

<a href="http://dbpedia.org/resource/Mikoyan_MiG-35">http://dbpedia.org/resource/Mikoyan_MiG-35</a>
<a href="http://dbpedia.org/resource/Sukhoi_Su-35">http://dbpedia.org/resource/Sukhoi_Su-35</a>
<a href="http://dbpedia.org/resource/Sukhoi_Su-27">http://dbpedia.org/resource/Sukhoi_Su-27</a>
<a href="http://dbpedia.org/resource/Sukhoi_Su-57">http://dbpedia.org/resource/Sukhoi_Su-57</a>

Рисунок 7 – Демонстрация выполнения первого запроса



Естественно-языковой интерфейс к LOD

Введите запрос на русском языке

Экспериментальные летательные аппараты Китая

Выполнить

Результат выполнения запроса

<a href="http://dbpedia.org/resource/BIT_UAV">http://dbpedia.org/resource/BIT_UAV</a>
<a href="http://dbpedia.org/resource/CDADI_UAV">http://dbpedia.org/resource/CDADI_UAV</a>
<a href="http://dbpedia.org/resource/CASM_CK-GY">http://dbpedia.org/resource/CASM_CK-GY</a>
<a href="http://dbpedia.org/resource/CASM_CK-GY04">http://dbpedia.org/resource/CASM_CK-GY04</a>
<a href="http://dbpedia.org/resource/CASM_CK-HW13">http://dbpedia.org/resource/CASM_CK-HW13</a>
<a href="http://dbpedia.org/resource/CDADI_V/STOL_UAV">http://dbpedia.org/resource/CDADI_V/STOL_UAV</a>

Рисунок 8 – Демонстрация выполнения второго запроса

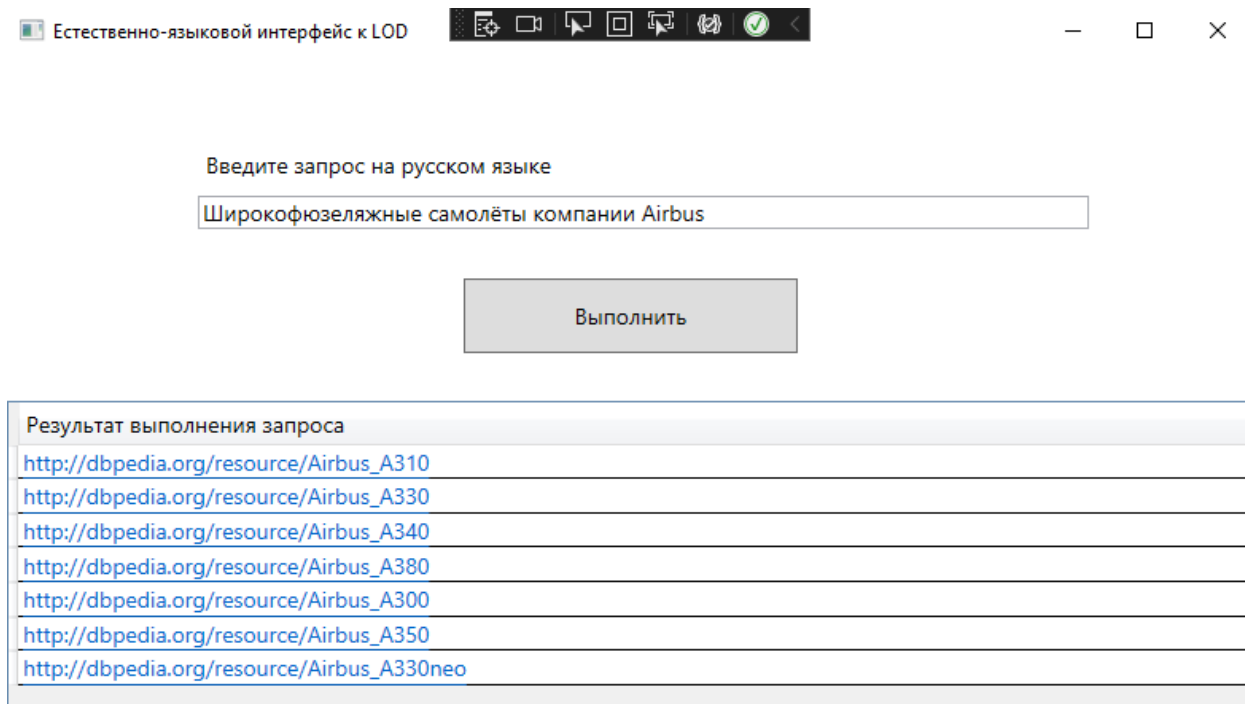


Рисунок 9 – Демонстрация выполнения третьего запроса

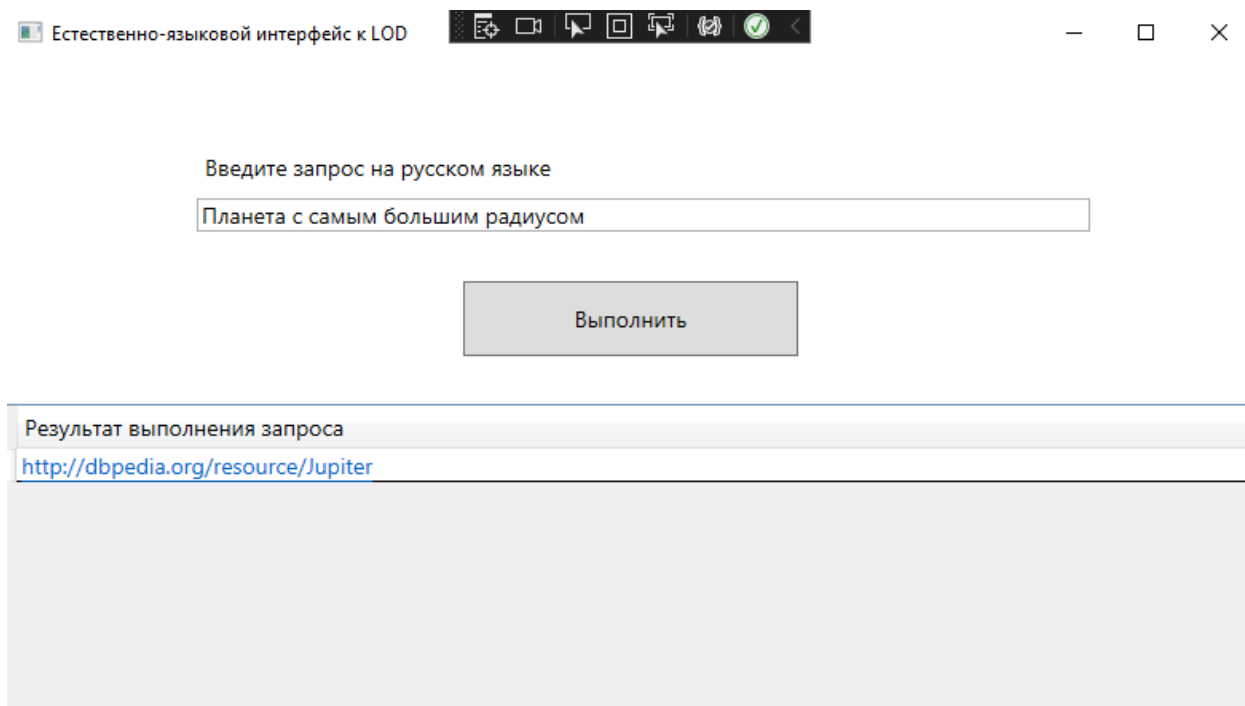


Рисунок 10 – Демонстрация выполнения четвертого запроса

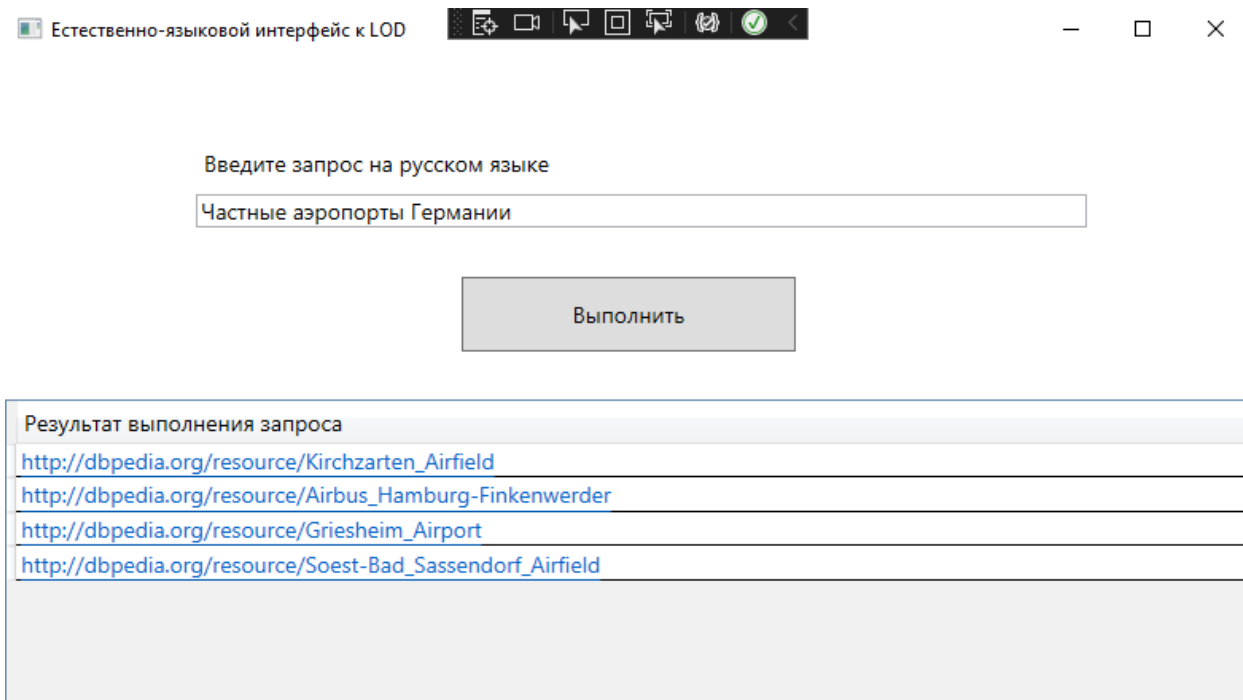


Рисунок 11 – Демонстрация выполнения пятого запроса

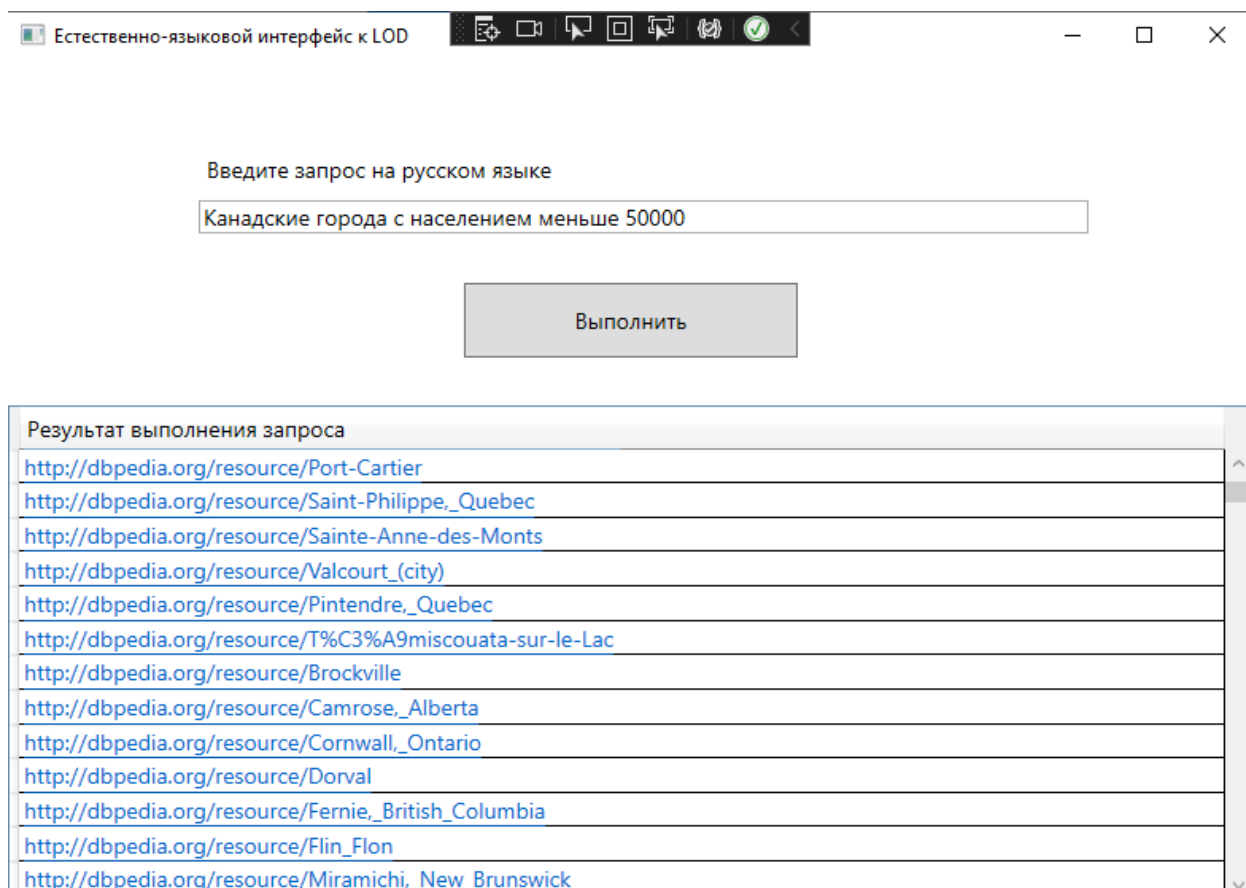


Рисунок 12 – Демонстрация выполнения шестого запроса

Для запросов, не входящих в указанный выше перечень, предусмотрены исключения в приложении и соответствующие сообщения об ошибках пользователю (рисунки 13 - 14).

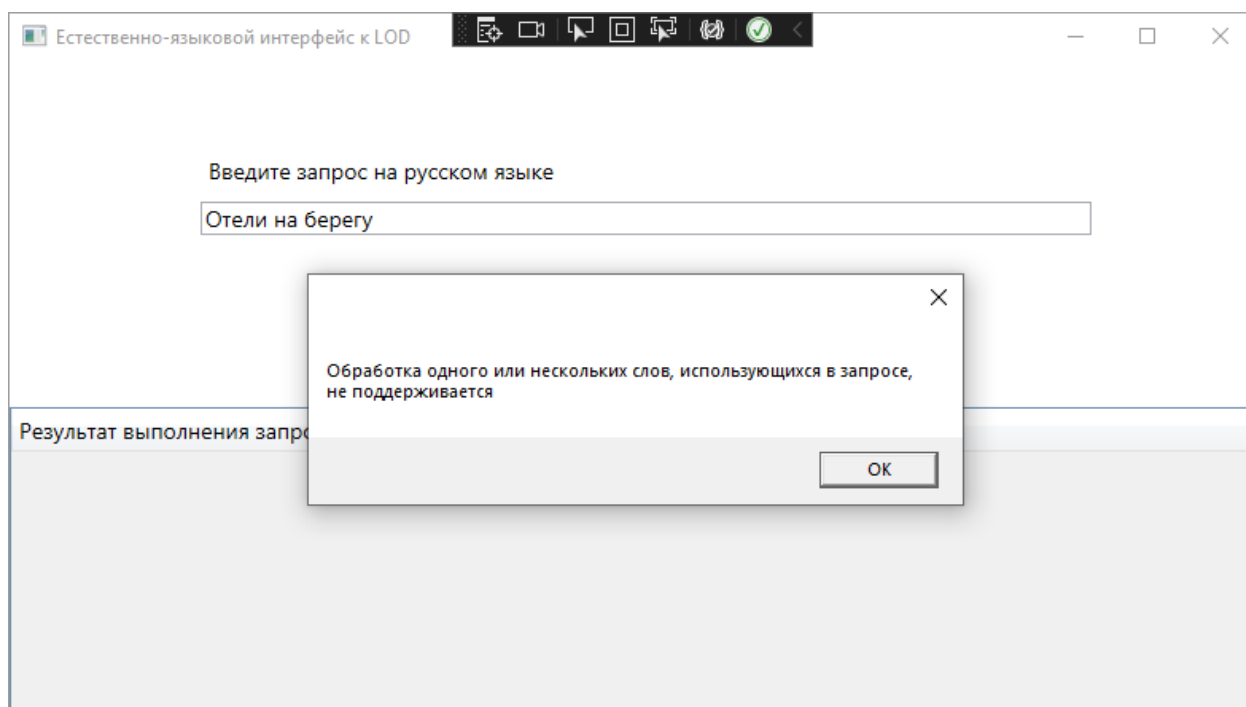


Рисунок 13 – Реакция приложения на ввод запроса со словом, отсутствующим в ЛБД

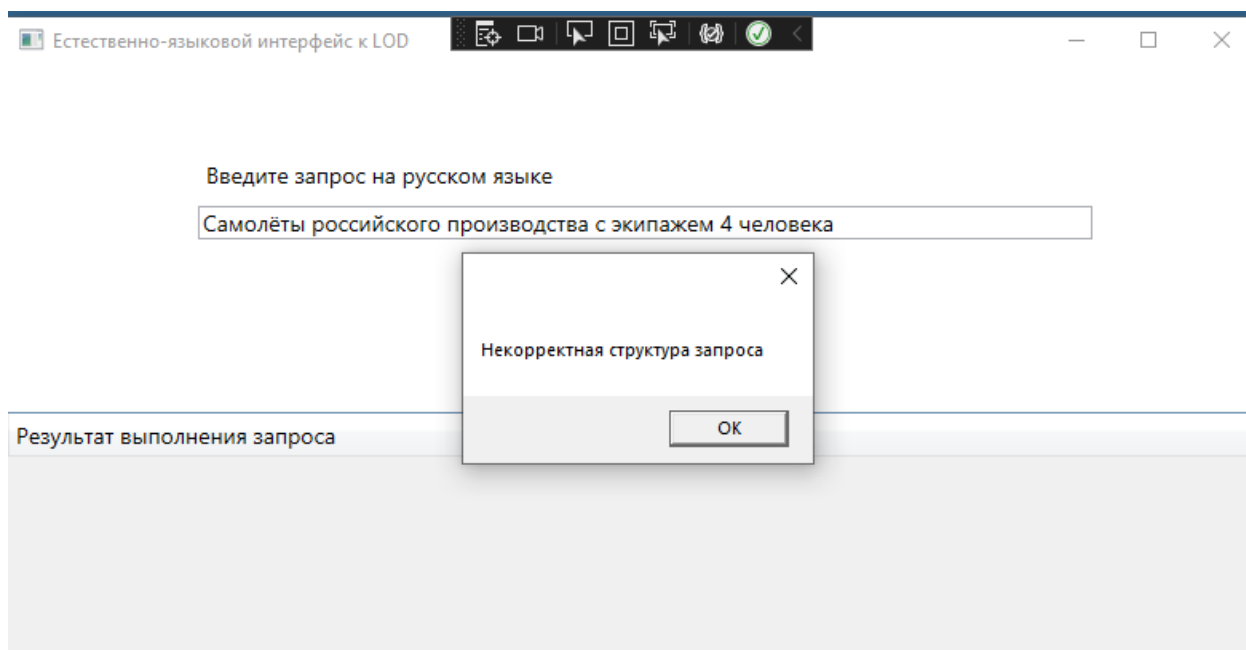


Рисунок 14 – Реакция приложения на ввод запроса со некорректной структурой

Также предусмотрены сообщения для пользователя об ошибках в работе с базой данных (рисунок 15) и о появлении неизвестных ошибок (рисунок 16).

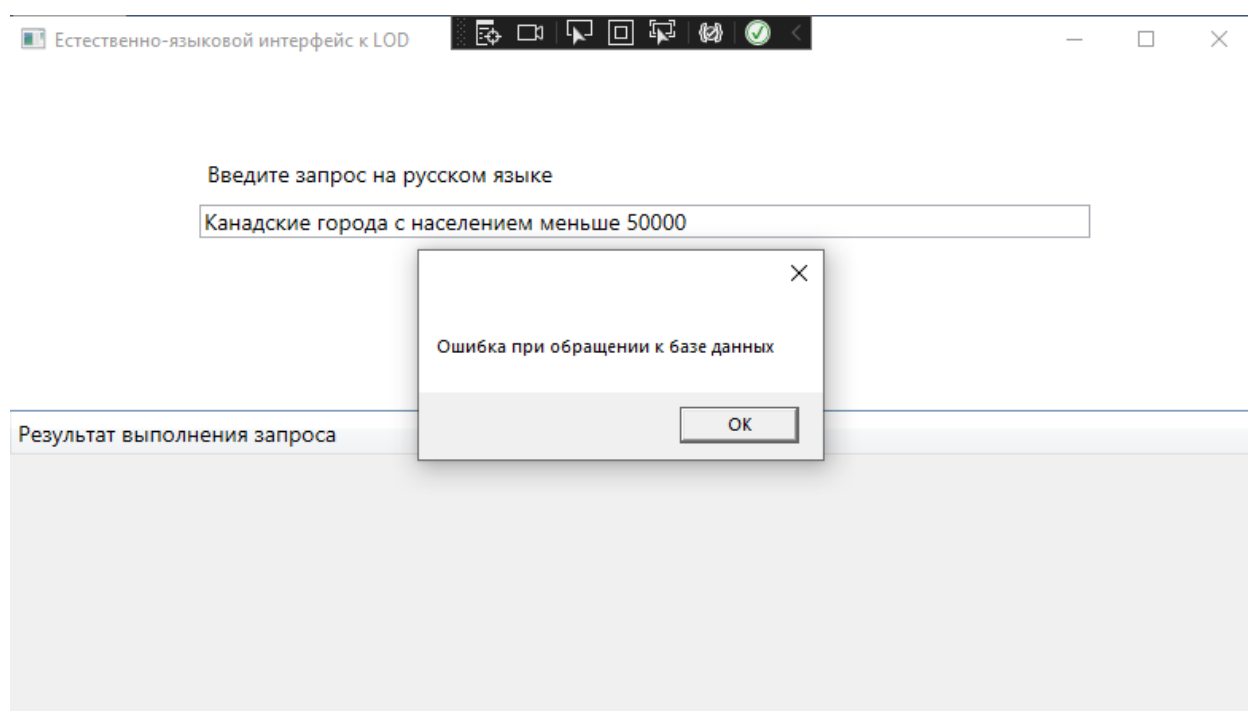


Рисунок 15 – Демонстрация реакции программы на появление ошибки при работе с базой данных

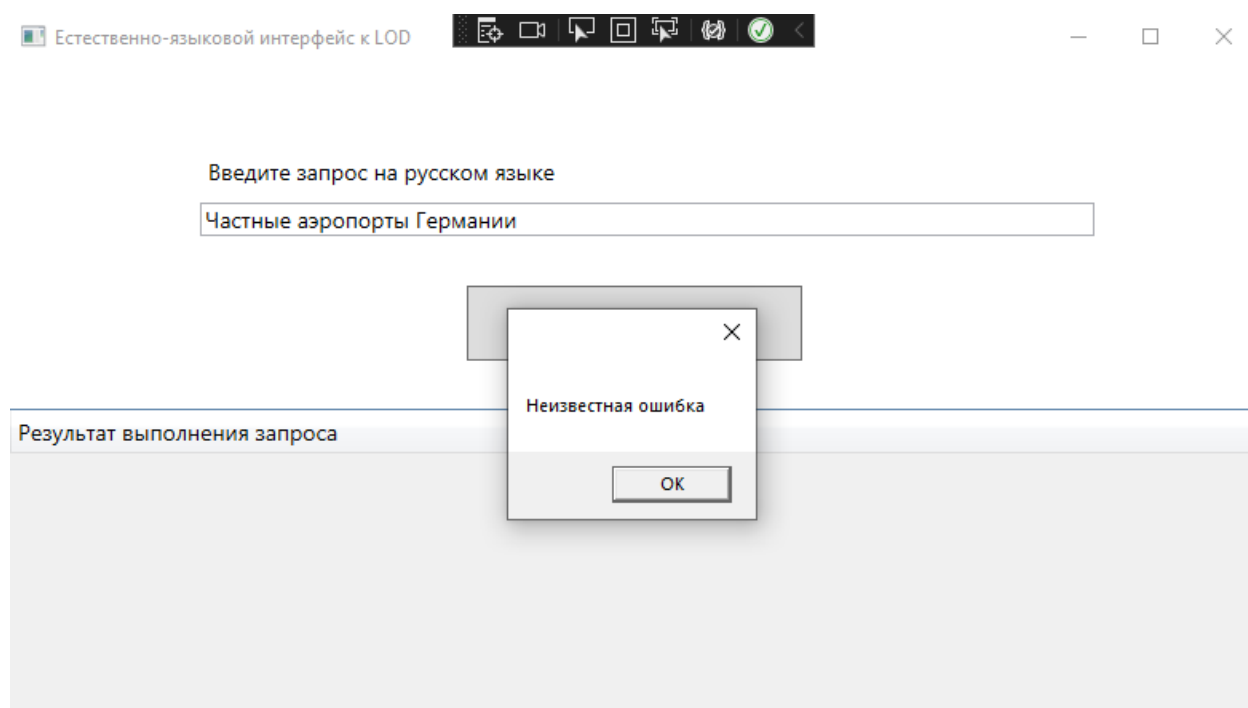


Рисунок 16 – Демонстрация реакции программы на появление неизвестной ошибки

## **Выводы**

В данной главе были выбраны средства разработки.

Приведено описание структуры реализованной базы данных.

Описан интерфейс разработанного приложения и продемонстрирована его работоспособность. Фактическое поведение приложения совпадает с ожидаемым.

## **Заключение**

В ходе данной работы был разработан и реализован семантически-ориентированный естественно-языковой интерфейс для взаимодействия с Системой взаимосвязанных открытых данных (LOD).

Для этого были проанализированы сведения о системе LOD и ее применениях.

Рассмотрены синтаксис и основные конструкции языка SPARQL.

Были проведены анализ и сравнение основных современных подходов к формальному описанию семантической структуры текстов на ЕЯ. Для использования в работе был выбран подход теории К-представлений В.А. Фомичева.

Были рассмотрены основные подходы к разработке семантически-ориентированных ЕЯ-интерфейсов для взаимодействия с прикладными интеллектуальными системами.

Проанализированы основные подходы к разработке интеллектуальных интерфейсов для преобразования запроса к LOD на ЕЯ в запросы на языке SPARQL.

Была разработана структура лингвистической базы данных.

Разработаны алгоритмы для реализации преобразования «ЕЯ-запрос → SPARQL-запрос», в частности разработаны алгоритмы для реализации преобразований «ЕЯ-запрос → Семантическое представление» и «Семантическое представление → SPARQL-запрос».

Обнаружена проблема неоднозначности имен в онтологии, потребовавшая дополнительного исследования, и предложен принцип преобразования параметров запроса, позволяющий преодолевать данную проблему.

## Список литературы

1. SPARQL 1.1 Query Language // W3C URL: <https://www.w3.org/TR/sparql11-query/> (дата обращения: 10.02.2020).
2. Sébastien Ferré. SQUALL: a Controlled Natural Language for Querying and Updating RDF Graphs (2012).
3. Sébastien Ferré. SQUALL: A Controlled Natural Language as Expressive as SPARQL 1.1. (2013).
4. Neli Zlatareva, Devansh Amin. Natural Language to SPARQL Query Builder for Semantic Web Applications // Journal of Machine Intelligence and Data Science. - 2021. - Volume 2. - С. 44-53.
5. Фомичев В.А. Формализация проектирования лингвистических процессоров. - М.: МАКС Пресс, 2005. - 367 с.
6. Linked Data // W3C URL: <https://www.w3.org/DesignIssues/LinkedData.html> (дата обращения: 10.02.2022).
7. RDF 1.1 Primer // W3C URL: <https://www.w3.org/TR/rdf11-primer/> (дата обращения: 10.02.2022).
8. RDF Schema 1.1 // W3C URL: <https://www.w3.org/TR/2014/REC-rdf-schema-20140225/> (дата обращения: 10.02.2022).
9. OWL 2 Web Ontology Language Primer (Second Edition) // W3C URL: <https://www.w3.org/TR/2012/REC-owl2-primer-20121211/> (дата обращения: 11.02.2022).
10. Stadler C., Lehmann J., Höffner K., Auer S. LinkedGeoData: A Core for a Web of Spatial Open Data. Semantic Web 3(4):333-354. IOS Press, 2012. – 23 p. doi: 10.3233/SW-2011-0052 URL: [https://www.researchgate.net/publication/240615213\\_LinkedGeoData\\_A\\_Core\\_for\\_a\\_Web\\_of\\_Spatial\\_Open\\_Data](https://www.researchgate.net/publication/240615213_LinkedGeoData_A_Core_for_a_Web_of_Spatial_Open_Data) (дата обращения: 05.03.2022).
11. Tirad M. Almalahmeh, Sameem Abdul Kareem, Mansoor A. Abdulgabber Semantic recommender system with natural language interface: malaysian



- tourism industry // International Journal of Scientific & Engineering Research. - 2013. - Volume 4, Issue 9. - C. 1059-1065.
12. Bauer F., Kaltenböck M. Linked Open Data: The Essentials. A Quick Start Guide for Decision Makers. – 62 p. URL: <https://www.recep.org/LOD-the-Essentials.pdf> (дата обращения: 05.03.2022).
  13. BDK Linked Open Data // Официальный сайт Российской государственной библиотеки. URL: <https://lod.rsl.ru/> (дата обращения: 15.03.2020).
  14. Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., Schneider, N. (2013). Abstract Meaning Representation for Sembanking. *In: Proceedings of the 7th ACL Linguistic Annotation Workshop and Interoperability with Discourse*, Sofia, Bulgaria, August 8-9, 2013 (2013)([www.aclweb.org/anthology/W13-2322](http://www.aclweb.org/anthology/W13-2322))
  15. Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., Schneider, N. (2019). *Abstract Meaning Representation (AMR) 1.2.6 Specification*; [github.com/amrisi/amr-guidelines/blob/master/amr.md](https://github.com/amrisi/amr-guidelines/blob/master/amr.md).
  16. Montague Semantics // Stanford Encyclopedia of Philosophy URL: <https://plato.stanford.edu/entries/montague-semantics/> (дата обращения: 16.03.2022).
  17. B. Nethravathi, G. Amitha, Anusha Saruka, T. P. Bharath, Setu Suyagya. Structuring Natural Language to Query Language: A Review // Engineering, Technology and Applied Science Research. - 2020. - №10(6). - C. 6521-6525.
  18. Lee M. Christensen, Henk Harkema, Peter J. Haug, Jeannie Y. Irwin, Wendy W. Chapman (2009). ONYX: A System for the Semantic Analysis of Clinical Text
  19. Chuan Wang, Nianwen Xue, Sameer Pradhan (2015). A Transition-based Algorithm for AMR Parsing.

20. Sylvain Pogodalla (2004). Computing Semantic Representation: Towards ACG Abstract Terms as Derivation Trees.
21. Alisa Kongthon, Sarawoot Kongyoung, Choochart Haruechaiyasak and Pornpimon Palingoon A Semantic Based Question Answering System for Thailand Tourism Information // Proceedings of the KRAQ11 Workshop. - Chiang Mai: Asian Federation of Natural Language Processing, 2011. - C. 38-42.
22. Steven Neale, Joao Silva and Antonio Branco Small in Size, Big in Precision: A Case for Using Language-Specific Lexical Resources for Word Sense Disambiguation // Proceedings of the Second Workshop on Natural Language Processing and Linked Open Data. - Hissar: INCOMA Ltd. Shoumen, 2015. - C. 6-15.
23. Thierry Declerck, Piroska Lendvai Towards the Representation of Hashtags in Linguistic Linked Open Data Format // Proceedings of the Second Workshop on Natural Language Processing and Linked Open Data. - Hissar: INCOMA Ltd. Shoumen, 2015. - C. 16-22.
24. Kiril Simov, Atanas Kiryakov Accessing Linked Open Data via A Common Ontology // Proceedings of the Second Workshop on Natural Language Processing and Linked Open Data. - Hissar: INCOMA Ltd. Shoumen, 2015. - C. 33-41.