****МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ

РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

**Институт № 3 «Системы управления, информатика и электроэнергетика»**

**Кафедра 319 Группа М3О-435Б-18**

**Направление подготовки Информатика и вычислительная техника**

**Профиль Программное обеспечение средств вычислительной техники и**

**автоматизированных систем**

**Квалификация Бакалавр**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**БАКАЛАВРА**

На тему: «Семантически-ориентированный естественно-языковой интерфейс

для взаимодействия с Системой взаимосвязанных открытых данных

(Linked Open Data)»

Автор ВКРБ Урубков Владислав Станиславович (\_\_\_\_\_\_\_\_\_\_\_\_\_)

(фамилия, имя, отчество полностью)

Руководитель Фомичев Владимир Александрович (\_\_\_\_\_\_\_\_\_\_\_\_\_)

(фамилия, имя, отчество полностью)

**К защите допустить**

Заведующий кафедрой 319 Нагибин Сергей Яковлевич (\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(№ каф) (фамилия, имя, отчество полностью)

“\_\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

Москва 2022г.

****МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ

РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

**Институт № 3 «Системы управления, информатика и электроэнергетика»**

**Кафедра 319 Группа М3О-435Б-18**

**Направление подготовки Информатика и вычислительная техника**

**Профиль Программное обеспечение средств вычислительной техники и**

**автоматизированных систем**

**Квалификация Бакалавр**

**УТВЕРЖДАЮ**

Заведующий кафедрой 319 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Нагибин С.Я.

(№ каф.) (подпись) (инициалы, фамилия)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

***З А Д А Н И Е***

**на выпускную квалификационную работу бакалавра**

**Обучающийся** Урубков Владислав Станиславович

(фамилия, имя, отчество полностью)

**Руководитель** Фомичев Владимир Александрович

(фамилия, имя, отчество полностью

доктор технических наук, профессор, профессор кафедры 319

(ученая степень, ученое звание, должность и место работы)

**1. Наименование темы** Семантически-ориентированный естественно-языковой

интерфейс для взаимодействия с Системой взаимосвязанных открытых данных

(Linked Open Data)

**2. Срок сдачи обучающимся законченной работы** 24 мая 2022

**3. Задание и исходные данные к работе**

Разработка семантически-ориентированного естественно-языкового интерфейса для

взаимодействия с Системой взаимосвязанных открытых данных (LOD)

Входные запросы от пользователя должны быть на русском языке. Входной запрос должен

преобразовываться в SPARQL-запрос, результат выполнения которого должен быть выведен на

экран

**Перечень иллюстративно-графических материалов: \*при наличии**

|  |  |  |
| --- | --- | --- |
| №  п/п | Наименование | Количество листов |
| 1 | 2 | 3 |
| 1 | Титульный слайд | 1 |
| 2 | Цели и задачи | 1 |
| 3 | Сведения о системе LOD | 3 |
| 4 | Подходы к описанию семантической структуры текстов | 2 |
| 5 | Лингвистическая база данных | 4 |
| 6 | Структура входного запроса | 1 |
| 7 | Примеры запросов | 1 |
| 8 | Структура семантического представления | 1 |
| 9 | Проблемы онтологий | 2 |
| 10 | Принципы преобразования параметров запросов к LOD | 2 |
| 11 | Преобразование семантического представления в SPARQL-запрос | 4 |
| 12 | Средства разработки | 1 |
| 13 | Интерфейс приложения | 1 |
| 14 | Выводы | 1 |

**4. Перечень подлежащих разработке разделов и этапы выполнения работы**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п/п | Наименование раздела или этапа | Трудоёмкость в % от  полной трудоёмкости  дипломной работы | Срок выполнения | Примечание |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Подготовка введения | 5 | 15.02.2022 |  |
| 2 | Анализ существующих решений | 20 | 07.03.2022 |  |
| 3 | Разработка алгоритмов для реализации преобразования «ЕЯ-запрос → SPARQL-запрос» | 25 | 31.03.2022 |  |
| 4 | Программная реализация | 25 | 03.05.2022 |  |
| 5 | Проверка работоспособности | 10 | 11.05.2022 |  |
| 6 | Оформление документации | 15 | 23.05.2022 |  |

**5. Исходные материалы и пособия**

1. Sébastien Ferré. SQUALL: A Controlled Natural Language as Expressive as SPARQL 1.1 // Proc. 18th Intern. Conf. on Applications of Natural Language to Information Systems, NLDB 2013, Salford, UK, June 2013. Springer, 2013, LNCS 7934, pp. 114-125.
2. Fomichov V. A. Semantics-Oriented Natural Language Processing: Mathematical Models and Algorithms. IFSR International Series on Systems Science and Engineering, vol. 27. New York, Dordrecht, Heidelberg, London: Springer, 2010. – 352 p.
3. Bauer F., Kaltenböck M. Linked Open Data: The Essentials. A Quick Start Guide for Decision Makers. – 62 p.

**6. Дата выдачи задания** 09 февраля 2022

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Задание принял к исполнению\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**АННОТАЦИЯ**

Тема дипломной работы: «Семантически-ориентированный естественно-языковой интерфейс для взаимодействия с Системой взаимосвязанных открытых данных Linked Open Data».

Данная работа посвящена разработке семантически-ориентированного естественно-языкового интерфейса для взаимодействия c Системой взаимосвязанных открытых данных. В ходе работы рассматриваются существующие применения системы Linked Open Data (LOD) и язык запросов к системе LOD SPARQL. Разработан алгоритм для реализации преобразования «ЕЯ-запрос → SPARQL-запрос», включающий алгоритмы для реализации преобразований «ЕЯ-запрос→Семантическое представление» и «Семантическое представление → SPARQL-запрос».

Объём дипломной работы – 98 страниц, она содержит 17 рисунков, 12 таблиц, 43 источника литературы, 1 приложение.

Результатом выполнения данной работы является разработка интеллектуального естественно-языкового интерфейса для обращения к Системе взаимосвязанных открытых данных.

Дипломная работа состоит из введения, трех основных разделов и заключения.

Во введении обосновывается актуальность исследования по данной тематике, описывается проблема, решение которой разрабатывается в данной работе.

Глава «Обзор существующих решений» содержит сведения о системе LOD и ее применениях, примеры синтаксиса и основных конструкций языка SPARQL, рассмотрение основных подходов к формальному семантическому описанию семантической структуры текстов на естественном языке (ЕЯ), основных подходов к разработке семантически-ориентированных ЕЯ-интерфейсов для взаимодействия с прикладными интеллектуальными системами и основных подходов к разработке интеллектуальных интерфейсов для преобразования запроса к LOD на ЕЯ в запросы на языке SPARQL.

В главе «Разработка алгоритмов для реализации преобразования «ЕЯ-запрос → SPARQL-запрос» описываются логическая структура лингвистической базы данных и алгоритмы для осуществления преобразований «ЕЯ-запрос → Семантическое представление» и «Семантическое представление → SPARQL-запрос». Также описывается проблема неоднозначности именования в онтологиях, обнаруженная во время разработки алгоритмов, и предлагается ее решение.

Глава «Сведения о программной реализации» содержит сведения об использованных средствах разработки, о реализованной структуре лингвистической базы данных, описание интерфейса приложения и демонстрацию работоспособности приложения.

В заключении представлены выводы по выполненной работе.

В приложении представлен презентационный материал.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 7](#_Toc104663174)

[1. АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ 9](#_Toc104663175)

[1.1. Краткие сведения о семантической информационной системе LOD и ее применениях 9](#_Toc104663176)

[1.2. Основные конструкции языка запросов SPARQL 13](#_Toc104663177)

[1.3. Основные современные подходы к формальному описанию семантической структуры текстов на естественном языке (ЕЯ) 20](#_Toc104663178)

[1.4. Основные подходы к разработке семантически-ориентированных ЕЯ-интерфейсов для взаимодействия с прикладными интеллектуальными системами 23](#_Toc104663179)

[1.5. Основные подходы к разработке интеллектуальных интерфейсов для преобразования запроса к LOD на ЕЯ в запросы на языке SPARQL 26](#_Toc104663180)

[1.6. Выводы по главе 28](#_Toc104663181)

[2. РАЗРАБОТКА АЛГОРИТМОВ ДЛЯ РЕАЛИЗАЦИИ ПРЕОБРАЗОВАНИЯ ВИДА «ЕЯ-ЗАПРОС SPARQL-ЗАПРОС» 29](#_Toc104663182)

[2.1. Разработка логической структуры лингвистической базы данных 29](#_Toc104663183)

[2.2. Алгоритм построения семантического представления входного запроса на естественном языке 31](#_Toc104663184)

[2.2.1. Постановка задачи 31](#_Toc104663185)

[2.2.2. Описания вспомогательных алгоритмов 32](#_Toc104663186)

[2.2.3. Описание головного модуля целевого алгоритма 37](#_Toc104663187)

[2.3. Алгоритм построения SPARQL-запроса по К-представлению входного запроса 40](#_Toc104663188)

[2.3.1. Постановка задачи 40](#_Toc104663189)

[2.3.2. Проблема неоднозначности имен предикатов 42](#_Toc104663190)

[2.3.3. Принципы преобразования параметров запросов к LOD 43](#_Toc104663191)

[2.3.4. Описание вспомогательных алгоритмов для построения SPARQL-запроса 44](#_Toc104663192)

[2.3.5. Описание основного алгоритма построения SPARQL-запроса 47](#_Toc104663193)

[2.4. Выводы по главе 49](#_Toc104663194)

[3. СВЕДЕНИЯ О ПРОГРАММНОЙ РЕАЛИЗАЦИИ 50](#_Toc104663195)

[3.1. Используемые средства разработки 50](#_Toc104663196)

[3.2. Описание реализованной схемы лингвистической базы данных 51](#_Toc104663197)

[3.2.1. Морфологическая база данных 51](#_Toc104663200)

[3.2.2. Лексико-семантический словарь 53](#_Toc104663201)

[3.2.3. Словарь предложных фреймов 56](#_Toc104663202)

[3.2.4. Компонент разрешения имен 57](#_Toc104663203)

[3.2.5. Общая схема лингвистической базы данных 59](#_Toc104663204)

[3.3. Интерфейс приложения 60](#_Toc104663205)

[3.4. Работоспособность приложения 61](#_Toc104663206)

[3.5. Выводы по главе 66](#_Toc104663207)

[ЗАКЛЮЧЕНИЕ 67](#_Toc104663208)

[СПИСОК ЛИТЕРАТУРЫ 68](#_Toc104663209)

[ПРИЛОЖЕНИЕ А. ПРЕЗЕНТАЦИЯ 74](#_Toc104663210)

# ВВЕДЕНИЕ

В современном мире Интернет продолжает развиваться, и всё большее распространение получают связные данные в связи с внедрением в Интернет соответствующих принципов [1]. Связность данных обеспечивается в Системе взаимосвязанных открытых данных (Linked Open Data, LOD). Данная система выступает технологической основой для создания важной ветви Семантического Веба (Semantic Web), делая данные в сети не только машиночитаемыми, но и «машинопонимаемыми».

Для обращения к системе LOD используется язык SPARQL [2]. Но для людей, не знакомых с ним, это делает использование LOD абсолютно невозможным (SPARQL имеет высокий образовательный порог входа). Для решения этой проблемы исследователями, работающими в этом направлении, создаются естественно-языковые интерфейсы (ЕЯ-интерфейсы). Они позволяют обращаться к связным данным с помощью запросов на естественном языке (ЕЯ). Но все работы, сделанные в данной области [3-5], в качестве естественного языка предполагают использование английского или его подъязыков. Поэтому целью данного исследования является разработка интеллектуального естественно-языкового интерфейса для обращения к LOD на русском языке.

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

Будет дано краткое введение в язык запросов к сиcтеме LOD SPARQL, его синтаксис и основные конструкций.

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 [6-25].

Будут рассмотрены подходы к разработке алгоритмов получения семантического представления (СП) текста на ЕЯ. Обычно исследователи, использующие в своих работах СП, предусматривают последовательные лексический, синтаксический, семантический анализы с последующим построением семантического представления. В свою очередь, в теории К-представлений В.А. Фомичев [6] предлагает новый метод построения СП текста на ЕЯ. 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.

Будут рассмотрены подходы, 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 С. Ферре [3;4] и американскими исследователями Н. Златаревой и Д. Амином в работе [5].

Основной научный результат исследования заключается в разработке алгоритма для реализации преобразования «ЕЯ-запрос → SPARQL-запрос». Для этого разработаны алгоритмы для реализации преобразований «ЕЯ-запрос → Семантическое представление» и «Семантическое представление → SPARQL-запрос».

# 1. АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

## 1.1. Краткие сведения о семантической информационной системе LOD и ее применениях

LOD (Linked Open Data) – Система взаимосвязанных открытых данных – система, подразумевающая использование методов публикации и связывания структурированных данных в Интернете. Основывается на четырех принципах, введенных Тимом Бернерс-Ли [1]:

1. Необходимо использование URI (Universal Resource Identifier) 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 использовать языковую систему 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).

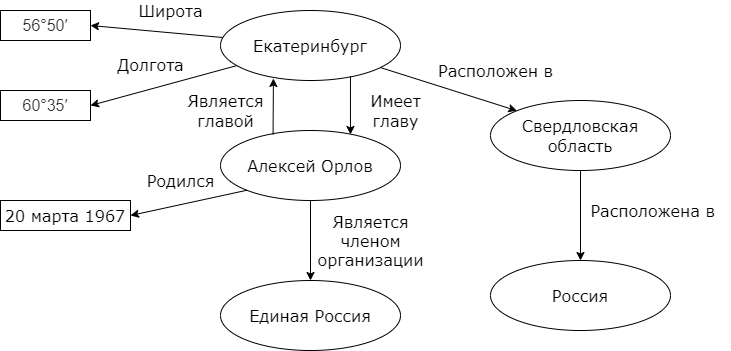


Рисунок 1 – Пример представления LOD

RDF (Resource Description Framework) [26] – языковая система (или фреймворк) для 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 данных и знаний. RDF представляет собой совокупность утверждений о ресурсах в виде, удобном для машинной обработки.

Формальная модель RDF рассматривает следующие множества:

* 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 определенных ресурсов, а множество «Свойства», соответственно – содержащих описания соответствующих предикатов. Множество «Литералы» состоит из всевозможных литералов, применение которых допустимо в описании тройки-утверждения.

Также существует языковая система RDFS [27], 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) [28]. 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, рефлективность, 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 [29].

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) [30]. 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 [31].

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 [32]. Классификационная система обеспечивает семантическую интеграцию библиотечных данных посредством публикации последних в виде LOD. Классификационная система работает на основе данных Российской государственной библиотеки.

## 1.2. 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» [2].

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 вариантов: с пометкой (тегом) языка, литерал со стандартным типом данных и литерал с произвольным типом данных (возможен еще четвертый тип литерала – простой литерал без тегов языка и типов, но в следующих трёх примерах запросов SPARQL он рассматриваться не будет). Пример данных с использованием 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" }

Таблица 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 }

Таблица 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.

Таблица 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.

Таблица 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 |

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

Одним из основных современных подходов к формальному описанию семантической структуры текстов на ЕЯ является Абстрактное представление смысла (АПС) [33;34]. В рамках данного подхода смысл каждого предложения 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 [35]. 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 → there is NP { np λx.true } “there is [NP 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.

Указанным выше подходам свойственна изначальная ориентация на описание высказываний (в Грамматике Монтегю еще есть возможность описывать вопросы, но этого все равно недостаточно). Однако еще существуют выражения, соответствующие целям, действиям, определениям, назначениям вещей, поэтому должен быть и формальный аппарат, позволяющий описывать смысловую структуру таких выражений.

Применяемым в данной работе подходом к формальному описанию семантической структуры текстов на ЕЯ является теория   
К-представлений (концептуальных представлений) В.А. Фомичева [6-25]. Данная теория, во-первых, предлагает 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 предложений на подъязыках английского. С помощью теории К-представлений (ТКП) возможно описать СП текстов в первую очередь на русском, а также на английском, немецком и французском языках. Кроме того, 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 (КП) можно привести семантическое представление следующего предложения: «Сколько раз в этом году запрашивался учебник Коробова?». Для данного предложения будет построено следующее КП:

Вопрос (x1, (x1 ≡ Колич-элем(все запрос1 \*(Время, текущий-год) (Предмет-запроса, нек учебник \* (Автор, нек человек \*(Фамилия, “Коробов”) : x2))))) .

## 1.4. 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) [36-39] 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 В.А. Фомичев в своих научных монографиях в рамках теории К-представлений формулирует новый метод преобразования ЕЯ-текста в СП текста [6;14]. Данный метод предназначается 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 значения морфологических признаков для лексических единиц входного 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 недоопределенность снимется шаг за шагом. Для снятия недоопределенности 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 долгого времени как отечественными, так и зарубежными исследователями.

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

Аналогичной работой по созданию интеллектуальных интерфейсов для преобразования запроса к LOD на ЕЯ в запросы на языке SPARQL занимается французский исследователь Себастьен 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» [3] и «SQUALL: A Controlled Natural Language as Expressive as SPARQL 1.1.» [4].

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» [5] (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.

## 1.6. Выводы по главе

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 к формальному представлению 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 произвольно сложных предложений и дискурсов на русском, английском, немецком, французском и многих других языках.

Проведен анализ подходов к разработке семантически-ориентированных ЕЯ-интерфейсов для взаимодействия с прикладными интеллектуальными системами. Для использования в данной работе в качестве отправной точки 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.

# 2. РАЗРАБОТКА АЛГОРИТМОВ ДЛЯ РЕАЛИЗАЦИИ ПРЕОБРАЗОВАНИЯ ВИДА «ЕЯ-ЗАПРОС → SPARQL-ЗАПРОС»

## 2.1. Разработка логической структуры лингвистической базы данных

Лингвистическая база данных (ЛБД) ЕЯ-интерфейса должна содержать информацию, позволяющую устанавливать возможные семантические отношения следующих сочетаний: «Существительное1 + Предлог + Существительное2», «Число + Существительное», «Прилагательное + Существительное», «Существительное1 + Существительное2».

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

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, Сема, 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.

Сема – соответствующая лексеме семантическая единица. Для глаголов, причастий, деепричастий это 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 сема представляется в виде выражения СК-языка. Например, для лексемы «зеленый» семой будет выражение Цвет (z1, зеленый).

Семантические координаты сущности, характеризуемой семой, определяют различную смысловую нагрузку, которая может быть у нее, в разных контекстах. Например, если сема – 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, алюминевый, 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 содержаться 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, 30-33].

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.

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

Таблица 9 – Структура морфологического представления запроса

|  |  |
| --- | --- |
| **base** | **morph** |
| 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

|  |  |  |  |
| --- | --- | --- | --- |
| **unit** | **tclass** | **subclass** | **mcoord** |
| в | предлог | - | 1 |
| самолет | существительное | сущ-нарицательное | 2 |

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

#### Описание функции Dictionary-form

**Вход:** Rm – морфологическое представление входного запроса Req, Rc - классифицирующее представление (КлП) запроса Req, p – целое – позиция   
какой-то словоформы в КлП Rc, т.е. 1≤ p ≤ lentext, где lentext = Длина(Т) – количество заполненных строк двумерного массива Rc.

**Значение:** лексема (базовая форма) слова в позиции p массива Rc, т.е. слова Rc[p, unit].

**Пример.** Пусть Req = Запрос6 = "недорогое вращающееся кресло для школьника", p = 5. Тогда Rc[p, unit] = «школьника», Dictionary-form(Rm, Rc, p) = «школьник».

#### Описание функции Input-line

**Вход:** base - лексема (базовая форма) некоторого слова, Arls – двумерный массив – проекция лексико-смантического словаря Lsdic на входной запрос. **Значение:** целое – наименьший номер k такой строки масива Arls, что Arls[k, lec] = base, т.е. лексема base расположена в массиве Arls на пересечении строки с номером k и столбца с индексом lec.

#### Описание функции Modif-form

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

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

**Пример.** *Modif-form(Вес(z, 3/тонна)) = (Вес, 3/тонна).*

#### Описание алгоритма Construct-sem-image

**Вход:** Rm – морфологическое представление входного запроса Req, Rc - классифицирующее представление (КлП) запроса Req, j – целое – позиция какого-то прилагательного в КлП Rc, m – целое – позиция некоторого прилагательного в КлП Rc, причем 1≤ j ≤ 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(Rm, Rc, 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:** Rm – морфологическое представление входного запроса Req, Rc - 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 отнощения, 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 cемантико-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 Валентино’).*

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 (проекции лексико- семантического словаря *Lsdic* на входной запрос), состоящего из записей упорядоченных наборов вида

*(i, lec, pt, sem, st1,…,stk, commelentext)* ,

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

Затем проверяется следующее Условие 1: среди сортов*,* ***st1,…, stk*** 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 c номером line2):

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

Если ДА, то в выходную строку semrel рассматриваемого адгоритма 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

#### Алгоритм SemParsing

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

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

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 Т соответственно лексико-семантического словаря Lsdic и словаря предложных семантико-синтаксических фреймов Frp.

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 NounOneNounTwoConnection

Кесл и

Конец

#### Описание основной подсистемы головного модуля

#### Алгоритм NounOneNounTwoConnection

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

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

**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} То 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}

То 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.)

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

Кесли

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

То 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:

А (B1, R1, C1) (B2, R2, C2) … (Bn, Rn, Cn),

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

B1, B2, …, Bn – имена смысловых параметров представления на русском языке,

R1, R2, …, Rn – имена бинарных отношений на русском языке,

С1, С2, …, Сn – 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 должными 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 именований одного и того же предиката, что не позволяет делать унифицированные по структуре запросы на языке SPARQL для запросов на естественном языке даже c одинаковой структурой. Например, в онтологии DBpedia в информации о городе предикат, связывающий данный город с количеством жителей, может иметь одно из следующих именований: «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 [40-43], 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 организуются таблица связи, которая будет содержать информацию о связи параметров и отношений К-представления и онтологии (пример таблицы связи представлен в таблице 12). 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 12 – Связь параметров в К-представлении с соответствующими параметрами в онтологии

|  |  |
| --- | --- |
| **Параметр в К-представлении** | **Параметр в онтологии** |
| 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», «p» 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 A, 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:** creatуEqTriple([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 А (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

То translatingValues = triple[2]

Кесли

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

То 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

То 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

То request = request + sorting + “limit 1”

Кесли

Вернуть request

Конец

## Выводы по главе

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

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

Разработан 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.

# 3. СВЕДЕНИЯ О ПРОГРАММНОЙ РЕАЛИЗАЦИИ

## 3.1. Используемые средства разработки

В качестве средства разработки для приложения была выбрана платформа .NET (использовалась версия .NET6), 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 используются компоненты .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 DeepMorphy. 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 нескольким причинам: есть ее реализации от отечественных разработчиков, также она предоставляет возможность создавать базы данных неограниченного размера и обеспечивает целостность данных на высоком уровне.

## 3.2. Описание реализованной схемы лингвистической базы данных



### Морфологическая база данных

Для морфологической базы данных схема представлена на рисунке 2.

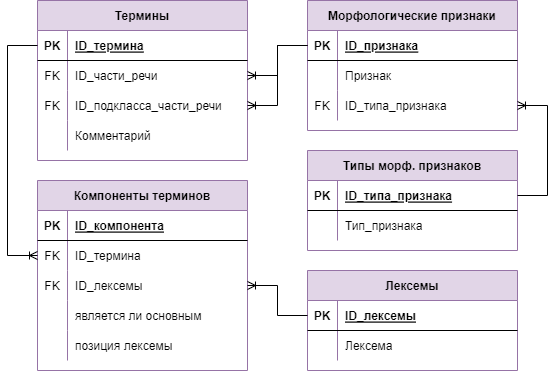


Рисунок 2 – Схема морфологической базы данных

*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 3.

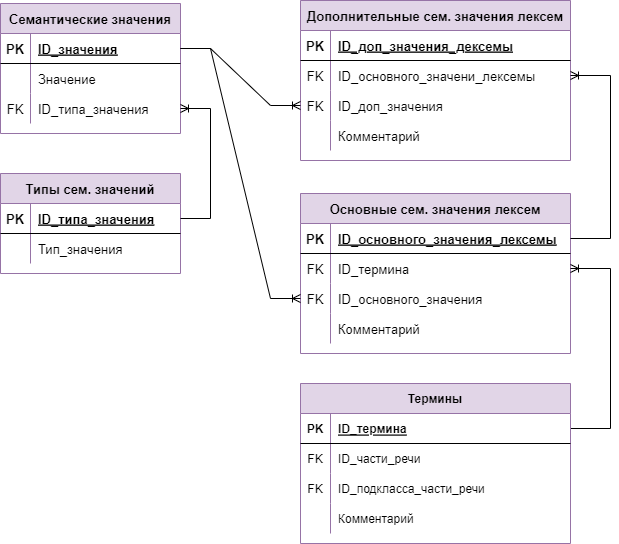


Рисунок 3 – Схема лексико-семантического словаря

#### 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\_лексемы – идентификатор значения конкретной лексемы (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 лексемы 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\_лексемы – идентификатор дополнительного значения конкретной лексемы (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 4.

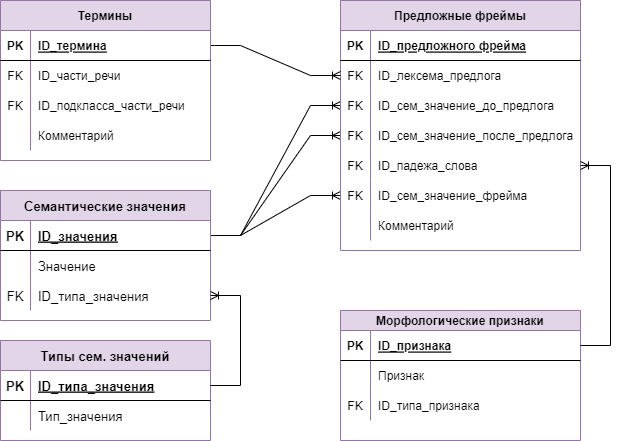


Рисунок 4 – схема словаря предложных фреймов

#### 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 находится 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 5.

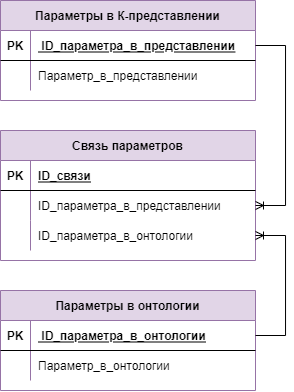


Рисунок 5 – Схема компонента разрешения имен параметров

#### 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 6.

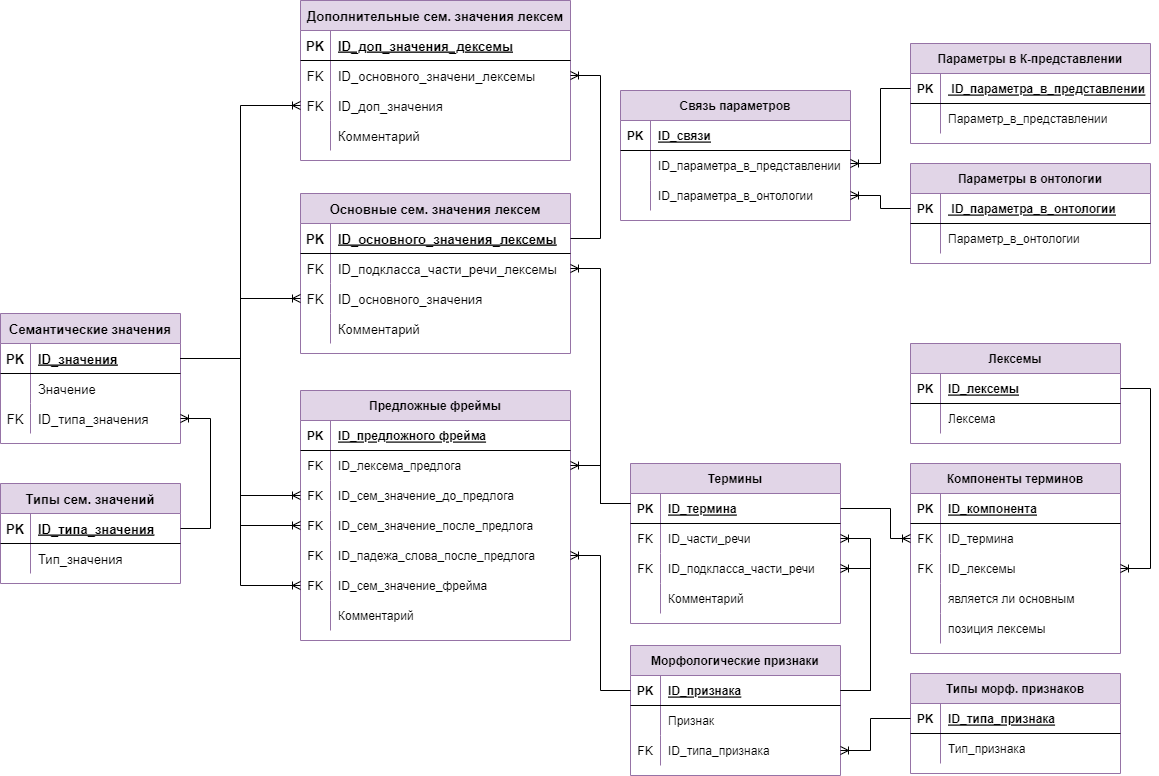


Рисунок 6 – Общая схема лингвистической базы данных

## Интерфейс приложения

Внешний вид разработанного приложения представлен на рисунке 7.

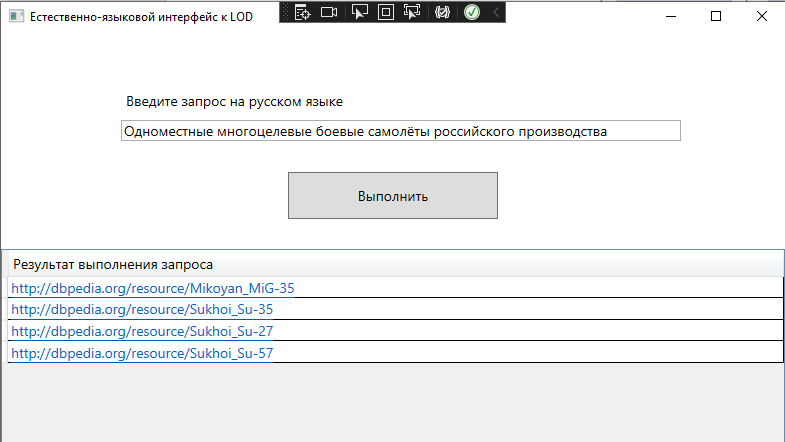


Рисунок 7 – Интерфейс приложения

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 8 - 13):

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”.

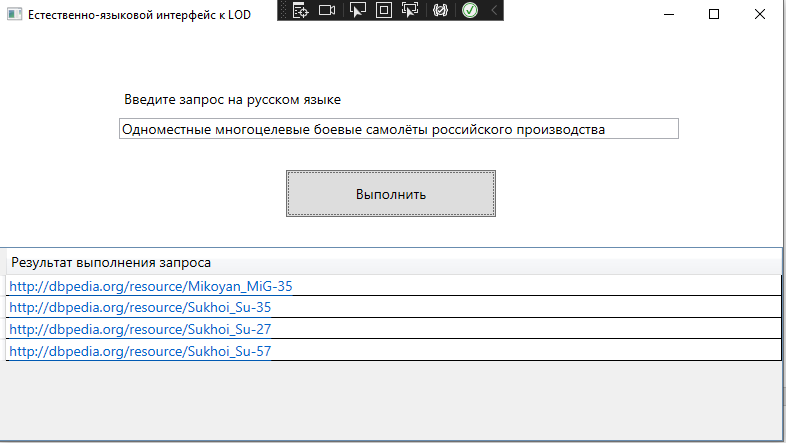


Рисунок 8 – Демонстрация выполнения первого запроса

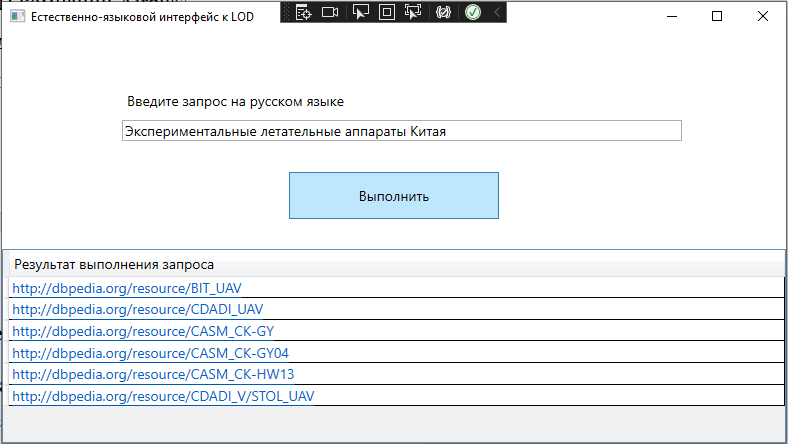


Рисунок 9 – Демонстрация выполнения второго запроса

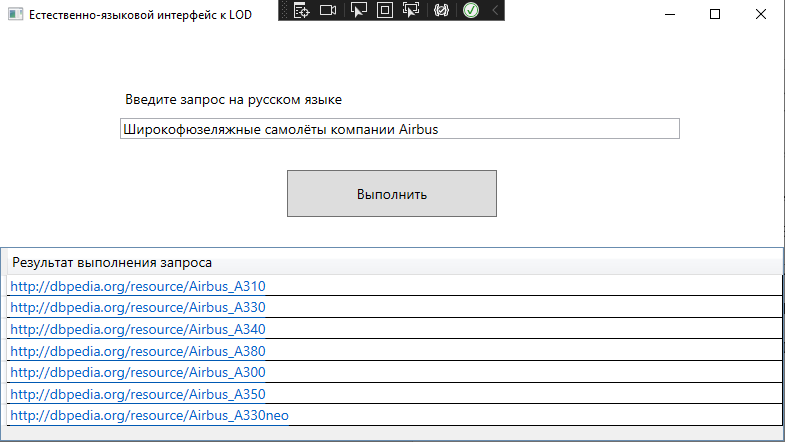


Рисунок 10 – Демонстрация выполнения третьего запроса

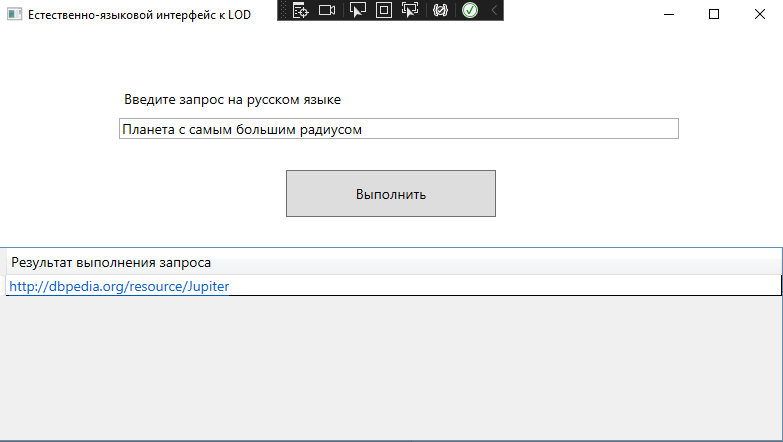


Рисунок 11 – Демонстрация выполнения четвертого запроса

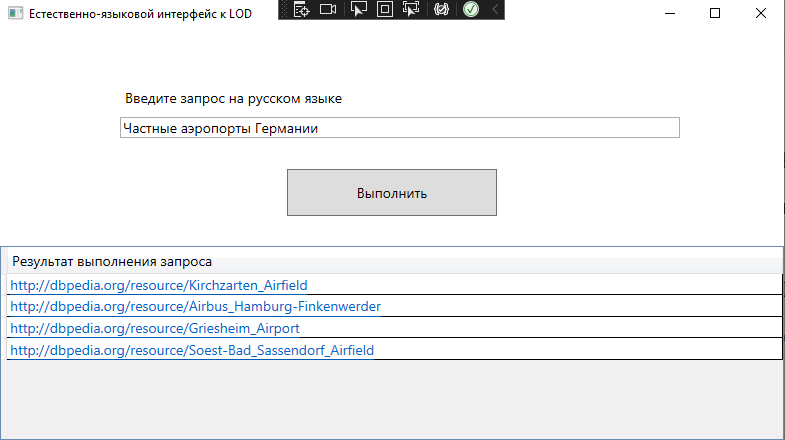


Рисунок 12 – Демонстрация выполнения пятого запроса

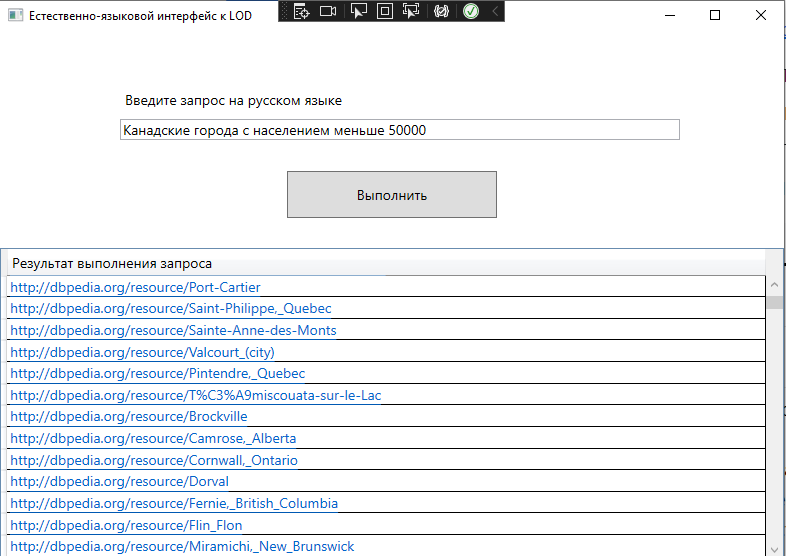


Рисунок 13 – Демонстрация выполнения шестого запроса

Для запросов, не входящих в указанный выше перечень, предусмотрены исключения в приложении и соответствующие сообщения об ошибках пользователю (рисунки 14 - 15).

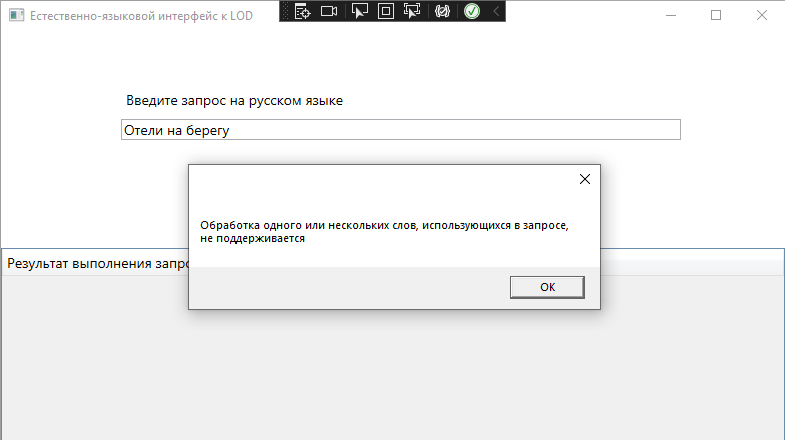


Рисунок 14 – Реакция приложения на ввод запроса со   
словом, отсутствующим в ЛБД

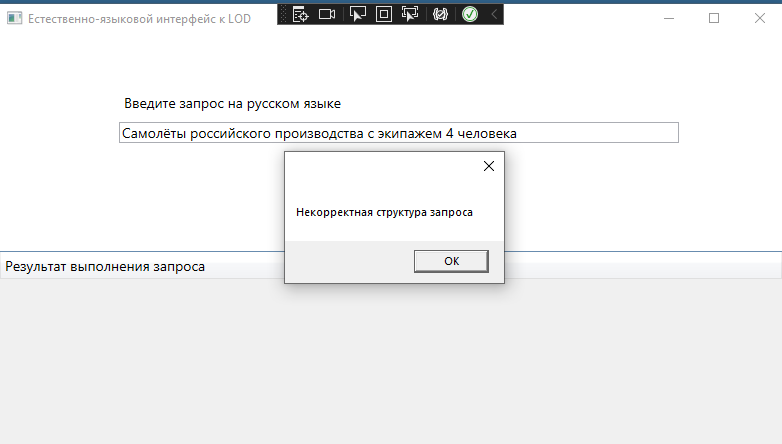


Рисунок 15 – Реакция приложения на ввод запроса с  
 некорректной структурой

Также предусмотрены сообщения для пользователя об ошибках в работе с базой данных (рисунок 16) и о появлении неизвестных ошибок (рисунок 17).

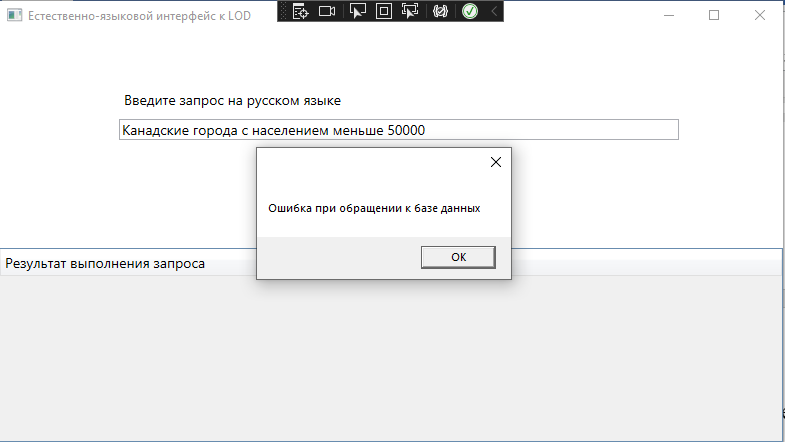


Рисунок 16 – Демонстрация реакции программы на появление ошибки при работе с базой данных

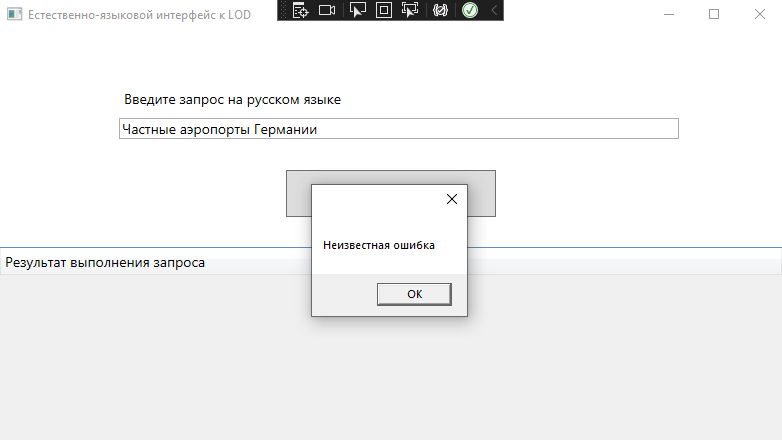


Рисунок 17 – Демонстрация реакции программы на появление неизвестной ошибки

## Выводы по главе

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 данного исследования 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 естественном языке (ЕЯ). Для использования в работе был выбран подход теории К-представлений (концептуальных 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 естественно-языкового запроса к LOD 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. Linked Data // W3C URL: https://www.w3.org/DesignIssues/LinkedData.html (дата обращения: 10.02.2022).
2. SPARQL 1.1 Query Language // W3C URL: https://www.w3.org/TR/sparql11-query/ (дата обращения: 10.02.2020).
3. Sébastien Ferré. SQUALL: a Controlled Natural Language for Querying and Updating RDF Graphs // Proc. Intern. Workshop on Controlled Natural Language (CNL 2012). Springer, LNAI, vol. 7427, pp. 11-25.
4. Sébastien Ferré. SQUALL: A Controlled Natural Language as Expressive as SPARQL 1.1 // Proc. 18th Intern. Conf. on Applications of Natural Language to Information Systems, NLDB 2013, Salford, UK, June 2013. Springer, 2013, LNCS 7934, pp. 114-125.
5. 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.
6. Fomichov V. A. Semantics-Oriented Natural Language Processing: Mathematical Models and Algorithms. // IFSR International Series on Systems Science and Engineering, vol. 27. New York, Dordrecht, Heidelberg, London: Springer, 2010. – 352 p.
7. Fomichov V. A. Integral Formal Semantics and the Design of Legal Full-Text Databases // Cybernetica. Quarterly Review of the International Association for Cybernetics (Belgium, Namur), 1994, vol. 37, no. 2, pp. 145–177.
8. Fomichov V. A. A Mathematical Model for Describing Structured Items of Conceptual Level // Informatica. An Intern. Journal of Computing and Informatics (Slovenia), 1996, vol. 20, no. 1, pp. 5–32.
9. Fomichov V. A. Theory of Restricted K-calculuses as a Comprehensive Framework for Constructing Agent Communication Languages // Fomichov V.A., Zeleznikar A.P. (eds.). Special Issue on NLP and Multi-Agent Systems. Informatica. An Intern. Journal of Computing and Informatics (Slovenia), 1998, vol. 22, no. 4, pp. 451-463.
10. Fomichov V. A. An ontological mathematical framework for electronic commerce and semantically-structured Web // Zhang, Y., Fomichov, V.A., Zeleznikar, A.P. (Eds.) Special Issue on Database, Web, and Cooperative Systems. Informatica. An Intern. Journal of Computing and Informatics (Slovenia), 2000. vol. 24, no. 1, pp. 39-49.
11. Fomichov V. A. Theory of K-calculuses as a powerful and flexible mathematical framework for building ontologies and designing natural language-processing systems // Andreasen, T., Motro, A., Christiansen, H., Larsen, H.L. (Eds.), Flexible Query Answering Systems, 5th Intern. Conference, FQAS 2002, Proceedings, Lecture Notes in Artificial Intelligence. 2002. Berlin, Heidelberg, New York: Springer. vol. 2522, pp. 183-196.
12. Фомичев В. А. Математические основы представления смысла текстов для разработки лингвистических информационных технологий. Часть I // Информационные технологии, 2002, № 10, С. 16–25.
13. Фомичев В. А. Математические основы представления смысла текстов для разработки лингвистических информационных технологий. Часть II // Информационные технологии, 2002, № 11, С. 34–45.
14. **Фомичев В. А.** Формализация проектирования лингвистических процессоров. М: МАКС Пресс, 2005. – 368 с.
15. Фомичев В. А. Математические основы представления содержания посланий компьютерных интеллектуальных агентов. М.: Издательство ТЕИС, 2007. – 176 с.
16. Fomichov V. A. A Comprehensive Mathematical Framework for Bridging a Gap between Two Approaches to Creating a Meaning-Understanding Web // Intern. Journal of Intelligent Computing and Cybernetics, 2008, vol. 1, no. 1, pp. 143-163.
17. Fomichov V. A. Theory of K-representations as a Comprehensive Formal Framework for Developing a Multilingual Semantic Web // Informatica. An Intern. Journal of Computing and Informatics (Slovenia). 2010. vol. 34, no. 3, pp. 387-396.
18. Fomichov V. A. SK-languages as a Powerful and Flexible Semantic Formalism for the Systems of Cross-Lingual Intelligent Information Access // Informatica. An Intern. Journal of Computing and Informatics (Slovenia), 2017, vol. 41, no. 2, pp. 221-232.
19. Фомичев В.А., Разоренов А. А. Значение теории К-представлений для исследований по автоматическому выявлению семантических ролей // Информационные технологии. 2015. Т. 21. № 6. С. 403-411.
20. Razorenov A. A., Fomichov V. A. A new formal approach to semantic parsing of instructions and to file manager design // Database and Expert Systems Applications. Proceedings of the 27th International Conference, DEXA 2016, Porto, Portugal, September 5-8, 2016, vol. 9827. Part. I. Cham: Springer, 2016. P. 416-430.
21. Разоренов А. А., Фомичев В. А. Компактная формализация входных и промежуточных данных алгоритмов семантического анализа предписаний // Информационные технологии. 2016. Т. 22. № 12. С. 883-891
22. Разоренов А. А., Фомичев В. А. Новый подход к формализации семантической обработки предписаний на основе теории К-представлений // Информационные технологии. 2017. Т. 23. № 1. С. 3-14.
23. Fomichov V. A., Razorenov A.A. [Theory of K-representations as a Tool for Designing File Managers with a Natural Language Interface](https://publications.hse.ru/view/222734368), in: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), May 21 – 25, 2018, Opatija, Croatia. Proceedings*.* Rijeka : Croatian Society for Information and Communication Technology, Electronics and Microelectronics, 2018. P. 1185-1190.
24. Fomichov V.A. Intelligent Monitoring of News on Economics and Finance Based on Formal Semantics of the Movement Verbs // 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO), September 27 – October 1, 2021, Opatija, Croatia. Proceedings. Rijeka: Croatian Society for Information, Communication and Electronic Technology - MIPRO, 2021.
25. Fomichov V.A.: Semantic Mapping of Definitions for Constructing Ontologies of Business Processes // 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), May 23 - 27, 2022, Opatija, Croatia. Proceedings. Rijeka: Croatian Society for Information, Communication and Electronic Technology - MIPRO, 2022.
26. RDF 1.1 Primer // W3C URL: https://www.w3.org/TR/rdf11-primer/ (дата обращения: 10.02.2022).
27. RDF Schema 1.1 // W3C URL: https://www.w3.org/TR/2014/REC-rdf-schema-20140225/ (дата обращения: 10.02.2022).
28. 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).
29. 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 (дата обращения: 05.03.2022).
30. 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. - P. 1059-1065.
31. Bauer F., Kaltenböck M. Linked Open Data: The Essentials. A Quick Start Guide for Decision Makers. – 62 p. URL: https://www.reeep.org/LOD-the-Essentials.pdf (дата обращения: 05.03.2022).
32. BBK Linked Open Data // Официальный сайт Российской государственной библиотеки. URL: https://lod.rsl.ru/ (дата обращения: 15.03.2020).
33. 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/ URL: www.aclweb.org/anthology/W13-2322 (дата обращения: 16.03.2022).
34. 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. URL: github.com/amrisi/amr-guidelines/blob/master/amr.md. (дата обращения: 16.03.2022).
35. Montague Semantics // Stanford Encyclopedia of Philosophy URL: https://plato.stanford.edu/entries/montague-semantics/ (дата обращения: 16.03.2022).
36. 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). - С. 6521-6525.
37. 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 // Proc. of the Workshop on bioNLP, Boulder, Colorado, June 2009. ACL, 2019, pp. 19-27.
38. Chuan Wang, Nianwen Xue, Sameer Pradhan (2015). A Transition-based Algorithm for AMR Parsing // Proc. of the 2015 Conf. of the North American Chapter of the ACL: Human Language Technologies. Denver, Colorado, ACL, 2015, pp. 366-375.
39. Sylvain Pogodalla Computing Semantic Representation: Towards ACG Abstract Terms as Derivation Trees // TAG+7: Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms. - Vancouver, BC, CA: 2004. - С. 64-71.
40. 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. - С. 38-42.
41. 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. - С. 6-15.
42. 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. - С. 16-22.
43. 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. - С. 33-41.

# ПРИЛОЖЕНИЕ А. ПРЕЗЕНТАЦИЯ