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 }
```

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.

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 второгосуществительного Сущ2, prep – строка – предлог, относящийся к Сущ2 (возможно, пустой предлог nil).

Выход: semrel – строка – обозначение семантического отношения, реализующегося в сочетании (Сущ1, prep, Сущ2); conc-noun1 – строка - обозначение семантической единицы, ассоциированной с Сущ1 в рассматриваемом запросе; conc-noun2 – строка - обозначение семантической единицы, ассоциированной с Сущ2 в рассматриваемом запросе.

Алгоритм

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

{base1 - базовая форма (лексема) первого существительного}

base2:= Dictionary-form(Rm, Rc, position2)

{base2 - базовая форма (лексема) второго существительного}

В цикле по k от 1 до lentext построить одномерный массив wordnouns, состоящий изсуществительных 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
Кесли
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

3.2. Описание реализованной схемы лингвистической базы данных

3.2.1. Морфологическая база данных

Для морфологической базы данных схема представлена на рисунке 2.

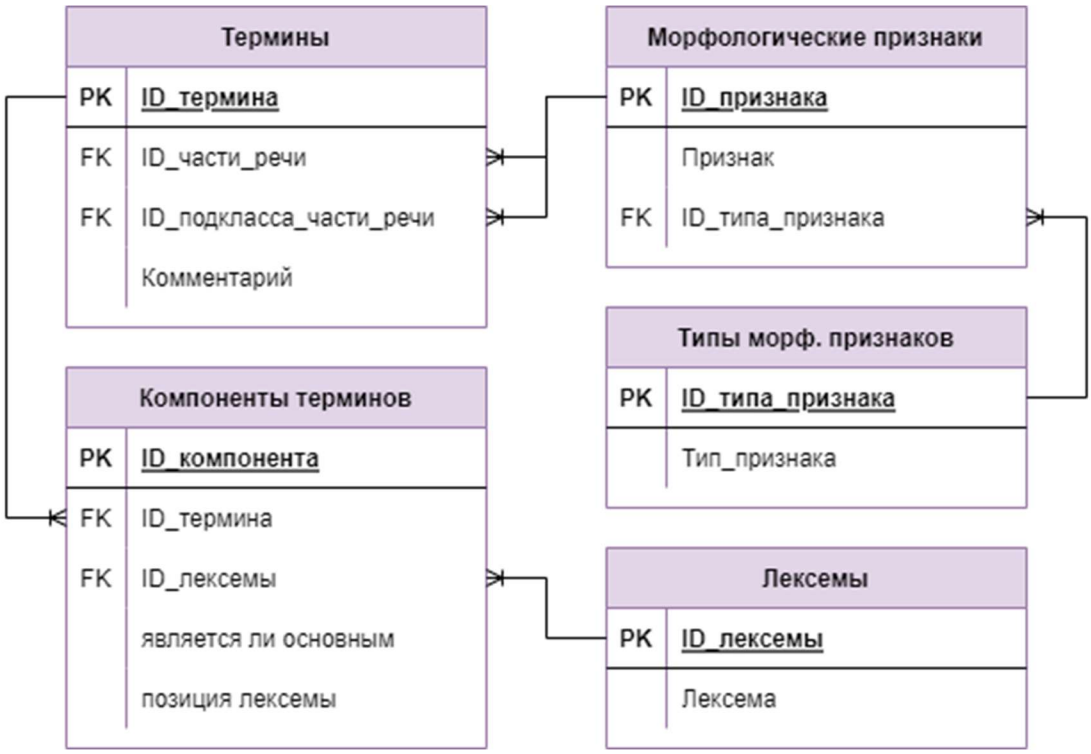

Рисунок 2 – Схема морфологической базы данных

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 «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).

3.2.2. Лексико-семантический словарь

Схема базы данных для лексико-семантического словаря представлена на рисунке 3.

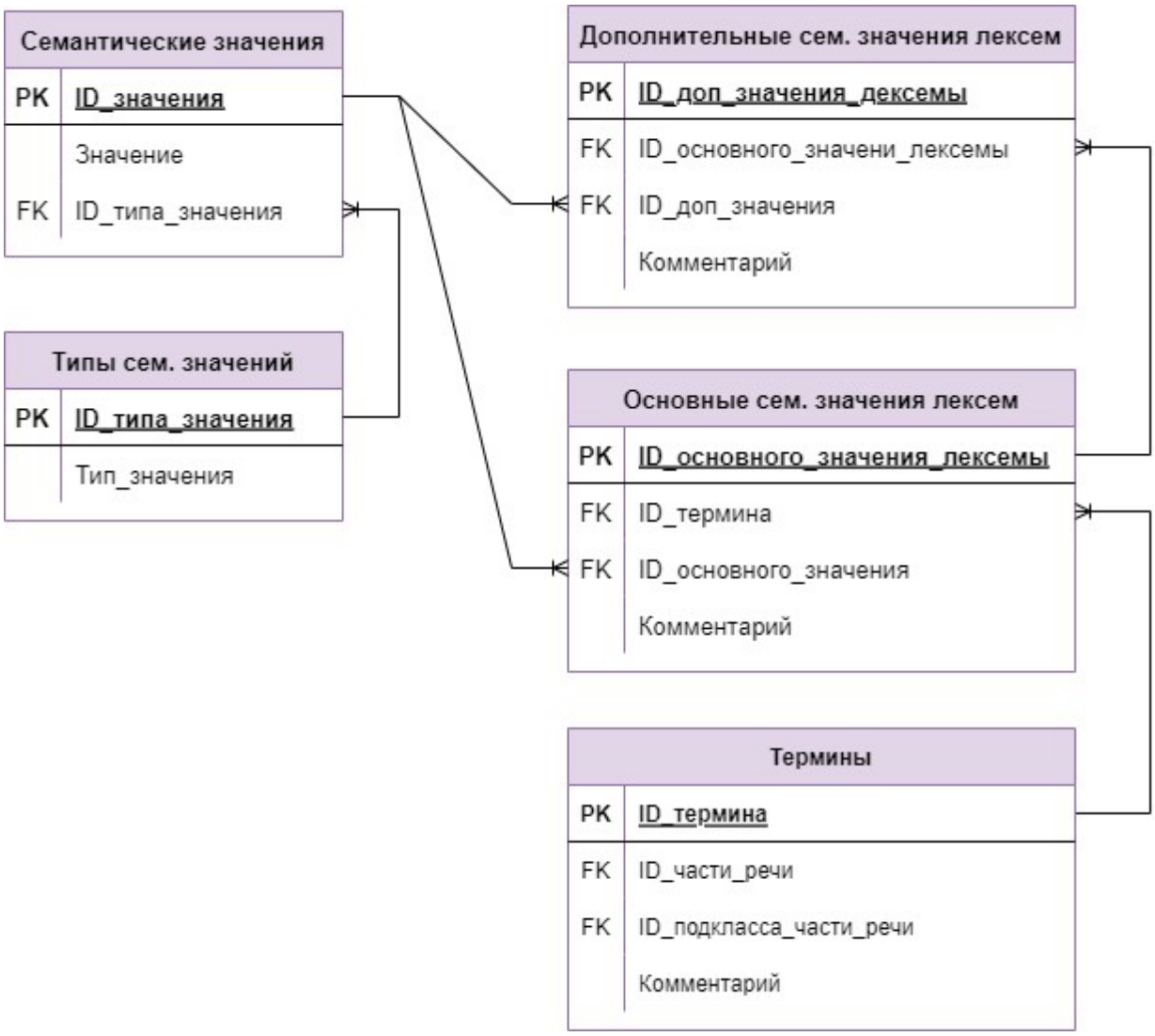

Рисунок 3 – Схема лексико-семантического словаря

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).

- 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.

3.2.3. Словарь предложных фреймов

Схема базы данных для словаря предложных фреймов представлена на рисунке 4.

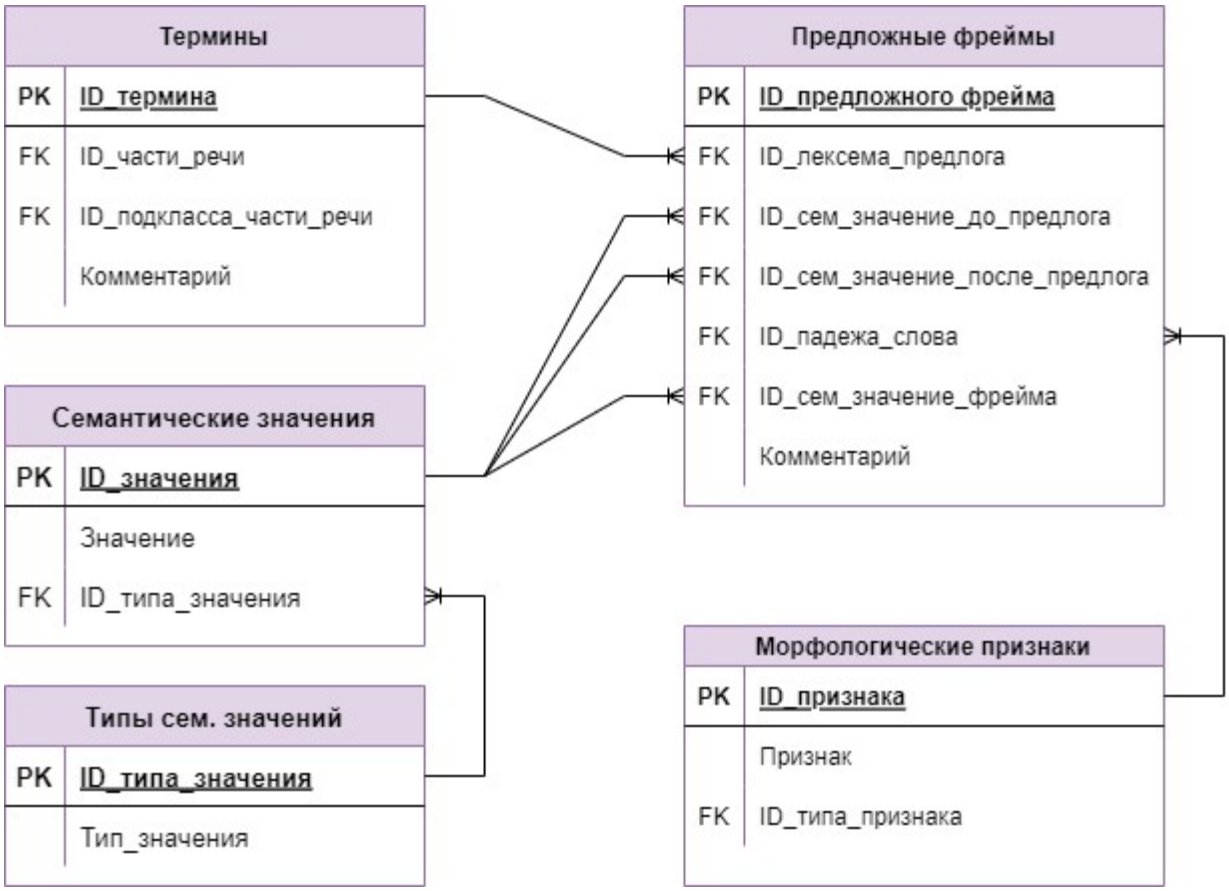

Рисунок 4 – схема словаря предложных фреймов

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.

3.2.4. 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

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).

3.2.5. Общая схема лингвистической базы данных

Общая схема всей ЛБД представлена на рисунке 6.

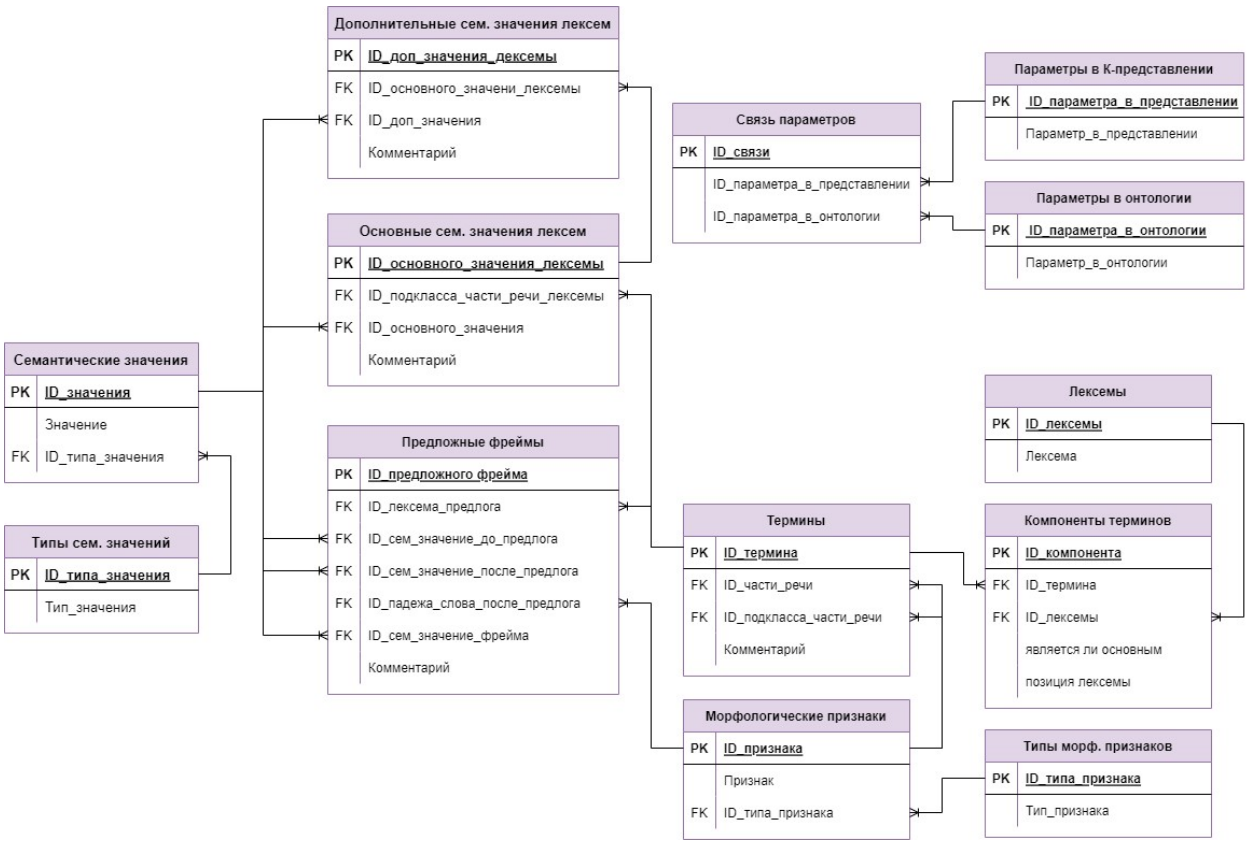

Рисунок 6 – Общая схема лингвистической базы данных

3.3. Интерфейс приложения

Внешний вид разработанного приложения представлен на рисунке 7.

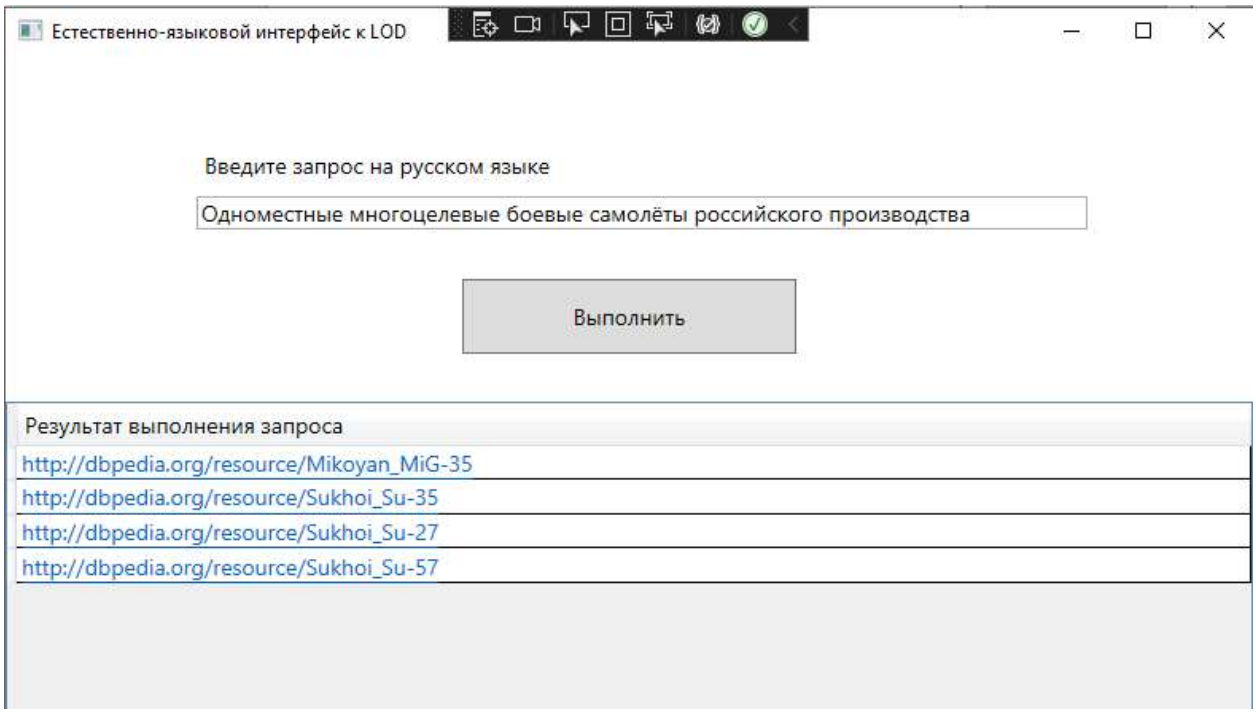

Рисунок 7 – Интерфейс приложения

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».

3.4. 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”.

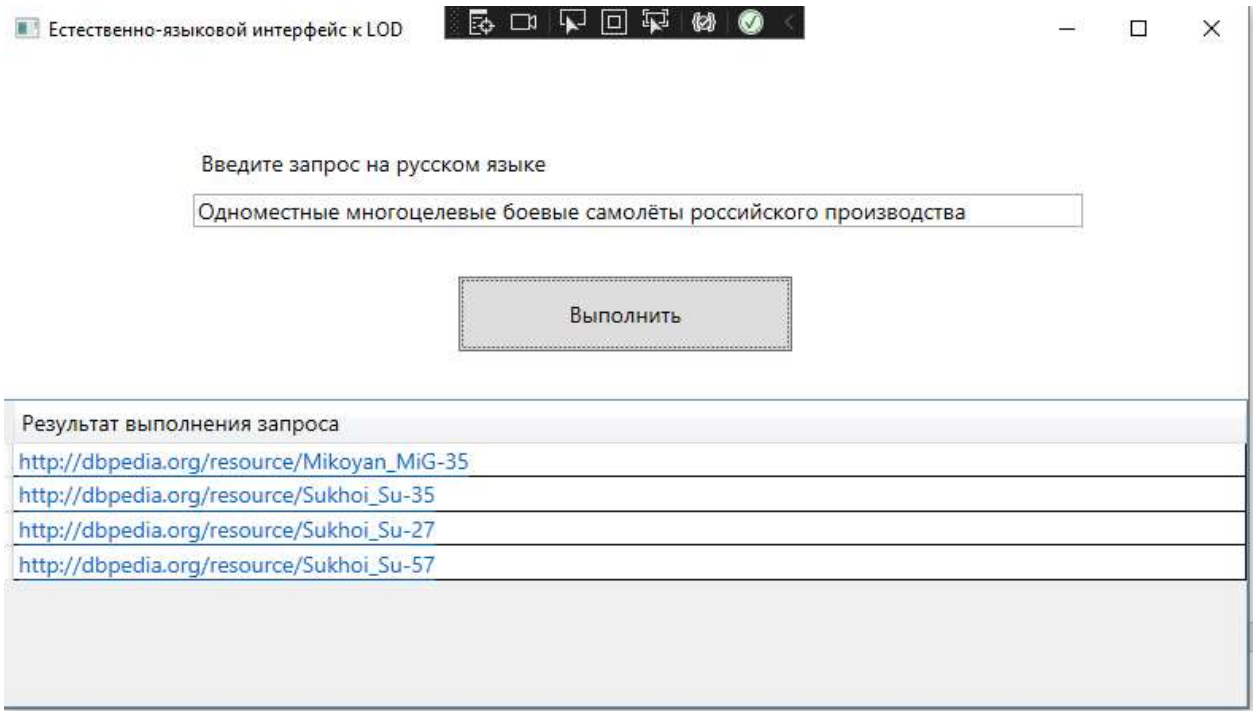

Рисунок 8 – Демонстрация выполнения первого запроса

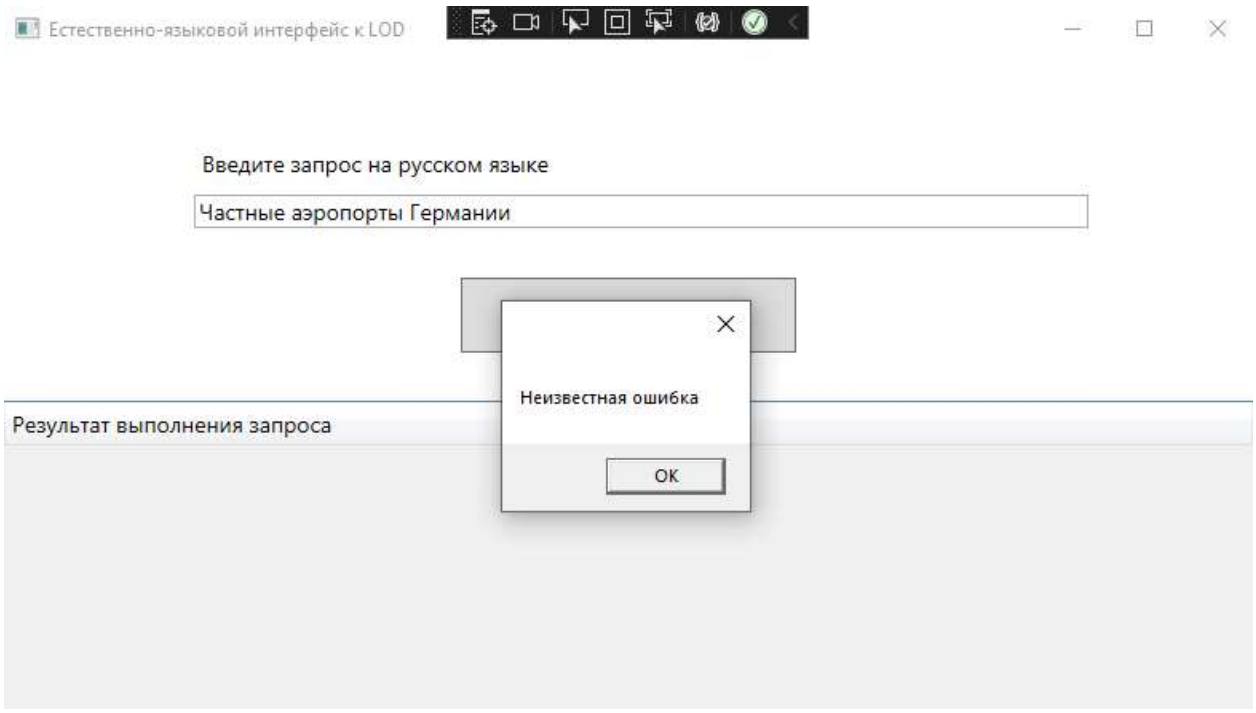

Рисунок 17 – Демонстрация реакции программы на появление неизвестной ошибки

3.5. 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.