



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления (ИУ)»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии (ИУ7)»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

*«Разработка приложения для выполнения
распределенного запроса, использующего данные из
PostgreSQL и MySQL СУБД.»*

Студент ИУ7-62Б
(Группа)

(Подпись, дата)

И.В. Козлова
(И. О. Фамилия)

Руководитель курсовой работы

(Подпись, дата)

Ю.М. Гаврилова
(И. О. Фамилия)

2022 г.

РЕФЕРАТ

Курсовая работа представляет собой реализацию приложения, выполняющего запрос в две различные СУБД (PostgreSQL и MySQL), и базы данных, таблицы которой расположены с различных СУБД. Готовое приложение предоставляет возможность ввода соответствующего запроса, а также просмотра результата.

Приложение реализовано на языке программирования Python 3 с использованием PyQt5 в качестве модуля для создания GUI приложения.

Ключевые слова: распределенный запрос, PostgreSQL, MySQL, Python, SQL.

Расчетно-пояснительная записка к курсовой работе содержит 51 страниц, 21 иллюстраций, 8 таблиц, 14 источников, 1 приложение.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	7
1 Аналитический раздел	9
1.1 Обзор языка SQL	9
1.2 Различия в диалектах SQL СУБД MySQL и PostgreSQL	10
1.3 Тернарная логика	11
1.4 Оператор SELECT и табличные выражения	13
1.5 Оператор WHERE	14
1.6 Лексический анализатор	15
1.7 Синтаксический анализатор	16
1.8 Семантический анализатор	17
1.9 Вывод	18
2 Конструкторский раздел	19
2.1 Общий алгоритм работы программы	19
2.2 Реализация лексического анализатора	19
2.3 Реализация синтаксического анализатора	23
2.4 Реализация семантического анализатора	24
2.5 Формирование и хранение результата	25
2.6 Вывод	27
3 Технологический раздел	28
3.1 Выбор языка программирования и среды разработки	28
3.2 Подключение к базам данных	28
3.3 Используемая база данных	30
3.3.1 Создание таблиц	31
3.3.2 Наполнение таблиц	31
3.4 Структура программы	33
3.5 Интерфейс	34
3.6 Результаты работы программного обеспечения	35

3.7	Тестирование	38
3.8	Вывод	41
4	Исследовательский раздел	42
4.1	Технические характеристики	42
4.2	Постановка эксперимента	42
4.2.1	Цель эксперимента	42
4.2.2	Время работы приложения	42
4.3	Вывод	44
	ЗАКЛЮЧЕНИЕ	46
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	47
	ПРИЛОЖЕНИЕ А	49

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие термины и сокращения с соответствующими определениями.

СУБД – Система управления базами данных.

РБНФ – Расширенная форма Бэкуса-Наура.

РСУБД – Реляционная система управления базами данных.

SQL – Structured Query Language — язык структурированных запросов.

NoSQL – Not only Structed Query Language — термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL.

PostgreSQL – свободная объектно-реляционная система управления базами данных.

MySQL – свободная реляционная система управления базами данных.

InnoDB – это механизм хранения данных для систем управления базами данных MySQL и MariaDB.

IDEF0 (Integration Definition for Function Modeling) – нотация графического моделирования, используемая для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции.

ВВЕДЕНИЕ

В современном мире при решении многих прикладных задач используют различные базы данных. Иногда в больших проектах может использоваться сразу несколько различных баз данных, которые имеют отличную друг от друга архитектуру. Причин для этого может быть много, например, когда два отдельных друг от друга проекта сливаются в один, или, когда для решения различных задач требуются различные по архитектуре базы данных.

В больших проектах различные базы данных обрабатывают информацию, которую иногда следует обрабатывать совместно друг с другом. Например, в одной базе данных хранится информация о готовности товара, а в другой – о выпуске товара и в обеих базах данных обрабатывается информация о каких-то одинаковых товарах. На сегодняшний день существует очень мало приложений, которые могут работать с данными из различных баз данных, связано это может быть с уникальностью архитектур различных СУБД [1].

Реляционная модель данных нужна для организации данных в виде таблиц (отношений) с помощью связей - полей таблиц, ссылающихся на другие таблицы, которые называются внешними ключами. Сегодня существуют другие модели данных, включая NoSQL, но системы управления реляционными базами данных (СУБД) остаются доминирующими для хранения и управления данными во всем мире [2].

Одними из самых популярных СУБД являются MySQL и PostgreSQL [3].

MySQL — свободная реляционная система управления базами данных.

PostgreSQL - свободная объектно-реляционная система управления базами данных. Postgres более функциональная СУБД, нежели MySQL хотя и сложнее для использования.

PostgreSQL является популярным выбором для функций NoSQL. Она изначально поддерживает большое разнообразие типов данных, например таких как JSON, hstore и XML [4].

Целью данной курсовой работы является разработка приложения для выполнения распределенного запроса, который использует данные из PostgreSQL и MySQL СУБД.

Для достижения данной цели требуется решить следующие задачи:

- проанализировать язык SQL, а также группу операторов DML;
- проанализировать совместимость и различия PostgreSQL и MySQL;
- проанализировать работу оператора SELECT и табличные выражения;
- проанализировать и сформулировать ограничения на операторы JOIN и WHERE;
- разработать программное обеспечение, включающие в себя лексический, семантический и синтаксический анализаторы;
- сравнить время выполнения запроса для баз данных, находящихся в различных СУБД и в одной СУБД.

1 Аналитический раздел

В данном разделе представлено описание языка SQL, различий диалектов PostgreSQL и MySQL, описание реализуемых анализаторов и логики, используемой в SQL.

1.1 Обзор языка SQL

SQL – это декларативный язык программирования, применяемый для создания, модификации и управления данными в РСУБД. Декларативные выражения представляют собой запросы, описание того, что хочет получить пользователь.

Можно сказать, что язык SQL – это инструмент, с помощью которого человек управляет базой данных. Сам язык SQL состоит из операторов, инструкций и вычисляемых функций. Операторы можно разделить на следующие четыре группы.

1. Операторы определения данных (Data Definition Language, DDL).
2. Операторы манипуляции данными (Data Manipulation Language, DML).
3. Операторы определения доступа (Data Control Language, DCL).
4. Операторы управления транзакциями (Transaction Control Language, TCL).

Группы операторов DDL, DCL, TCL напрямую не работают с данными, поэтому такие операторы не будут рассматриваться подробно в данной курсовой работе.

Группа операторов манипуляции данными включает в себя четыре основных оператора:

1. SELECT (будет рассмотрен более подробно) выбирает данные, удовлетворяющие заданным условиям.
2. INSERT добавляет новые данные.

3. UPDATE изменяет существующие данные.
4. DELETE удаляет данные.

1.2 Различия в диалектах SQL СУБД MySQL и PostgreSQL

Чтобы лучше понимать исследуемую область, следует выявить особенности, которые есть в MySQL [5], но нет в PostgreSQL [6].

Функциональные различия приведены в таблице 1.1.

Таблица 1.1 – Функциональные различия СУБД

Особенности	PostgreSQL	MySQL
ANSI SQL совместимость	Близка к стандарту ANSI SQL	Следует некоторым стандартам ANSI SQL
Вложенные SELECT	Да	Нет
Транзакции	Да	Да, однако должен использоваться тип таблицы InnoDB
Поддержка внешних ключей	Да	Нет
Представления	Да	Нет
Хранимые процедуры	Да	Нет
Триггеры	Да	Нет
Ограничители целостности	Да	Нет
Различные типы таблиц	Нет	Да

Стоит отметить, некоторые синтаксические отличия данных СУБД.

- PostgreSQL чувствителен к регистру при сравнении строк, MySQL – нет.
- Имена баз данных, таблиц, полей и столбцов в PostgreSQL не зависят от регистра, если их не создали с двойными кавычками вокруг имени, в

этом случае они чувствительны к регистру, в MySQL имена таблиц могут быть чувствительны к регистру или нет, в зависимости от используемой операционной системы.

- MySQL принимает операторы языка C для логики (`|` – ИЛИ, `&` – И), SQL требует AND, OR; решение использовать стандартные ключевые слова SQL, обе базы данных могут с ними работать.

1.3 Тернарная логика

В СУБД вместо стандартной булевой логики чаще всего используется тернарная логика, которая является расширением булевой логики.

С точки зрения SQL, результат логического выражения может быть `true` (ПРАВДА), `false` (ЛОЖЬ) или `unknown` (НЕИЗВЕСТНО). С другой стороны есть тип `boolean`, и у него есть 3 значения: `true`, `false` и `null`. То есть формально `unknown` – формат вычисления, а `null` – конкретное значение, которое может быть записано в базе данных.

Рассмотрим основные операции в тернарной логике при помощи таблиц истинностей.

Конъюнкция («and»).

Таблица 1.2 – Конъюнкция («and»)

and	true	unknown	false
true	true	unknown	false
unknown	unknown	unknown	false
false	false	false	false

Дизъюнкция («or»).

Таблица 1.3 – Дизъюнкция («or»)

or	true	unknown	false
true	true	true	true
unknown	true	unknown	unknown
false	true	unknown	false

Отрицание («not»).

Таблица 1.4 – Отрицание («not»)

	true	unknown	false
not	false	unknown	true

Сравнение с помощью операнда «равенство» («=»).

Таблица 1.5 – Сравнение с помощью операнда «равенство» («=»)

=	true	unknown	false
true	true	unknown	false
unknown	unknown	unknown	unknown
false	false	unknown	false

Сравнение с помощью операнда «is».

Таблица 1.6 – Сравнение с помощью операнда «is»

	true	unknown	false
is true	true	false	false
is unknown	false	true	false
is false	false	false	true
is not true	false	true	true
is not unknown	true	false	true
is not false	true	true	false

1.4 Оператор SELECT и табличные выражения

Оператор SELECT – оператор языка SQL, возвращающий набор данных (выборку) из базы данных, удовлетворяющих заданному условию. Он позволяет производить выборки данных из одной или нескольких таблиц базы данных и преобразовывать к нужному виду полученные результаты. Этот оператор способен выполнять действия, эквивалентные операторам реляционной алгебры.

Оператор SELECT состоит из нескольких разделов.

1. Список возвращаемых столбцов (существующих в таблицах или вычисляемых на их основе с помощью различных функций).
2. Табличное выражение определяющие базовый набор данных.
3. Ограничение на строки из табличного выражения.
4. Объединение (группировка) строк с помощью правил.
5. Ограничения на сгруппированные строки.
6. Сортировка результирующей выборки.

В качестве табличного представления может выступать таблица из базы данных – тривиальные табличные выражения, но в более сложных выражениях такие таблицы можно преобразовывать и комбинировать самыми разными способами, например с помощью оператора JOIN можно соединить две и более таблицы (реальные или производные от них) в соответствии с правилами соединения.

В рамках данной работы будет рассматриваться соединение только двух таблиц (оператор JOIN, в рамках данной работы рассматривается INNER JOIN⁴⁴).

Оператор JOIN – оператор языка SQL, который формирует результирующую таблицу по заданным условиям, из одной или нескольких уже существующих таблиц SQL.

Существуют несколько следующих типов JOIN.

1. INNER JOIN (или просто JOIN) – каждая строка из первой (левой) таблицы, сопоставляется с каждой строкой из второй (правой) таблицы, после чего, происходит проверка условия.
2. LEFT JOIN – важен порядок следования таблиц. Сначала происходит формирование таблицы соединением INNER JOIN. Затем, в результат добавляются записи левой таблицы, не вошедшие в результат после INNER JOIN.
3. RIGHT JOIN – важен порядок следования таблиц. Аналогично LEFT JOIN, но во главе вторая таблица. Сначала происходит формирование таблицы соединением INNER JOIN.
4. FULL JOIN – оператор FULL JOIN можно воспринимать как сочетание операторов INNER JOIN + LEFT JOIN + RIGHT JOIN. Сначала происходит формирование таблицы соединением INNER JOIN. Затем, в результат добавляются записи левой и правой таблиц, не вошедшие в результат INNER JOIN.
5. CROSS JOIN (декартово произведение) – каждая строка левой таблицы сопоставляется с каждой строкой правой таблицы.

1.5 Оператор WHERE

Для эффективного использования ресурсов компьютера необходимо запрашивать из внешних источников только те данные, которые непосредственно участвуют в построении выборки. Другими словами, нет необходимости запрашивать колонки, которые не задействуются в выводе, также нет необходимости запрашивать данные, которые в последствие будут отфильтрованы.

Оператор WHERE задает ограничения на входную выборку. Все логические условия, которые участвуют в операторе WHERE можно разделить на три группы:

- условия, не использующие информацию из таблицы, то есть константы;
- условия, использующие данные из одной таблицы;
- условия, использующие данные из нескольких таблиц.

Наибольший интерес в операторе WHERE представляет группа условий, использующие только одну таблицу, так как такие условия можно применить на этапе запроса данных, таким образом уменьшается объем передаваемых данных.

1.6 Лексический анализатор

В грамматике SQL можно выделить шесть лексических доменов:

- целое число – ненулевая последовательность цифр, которая не начинается с нуля или ноль;
- число с плавающей точкой – два числа разделенных символом «.»;
- идентификатор – непустая последовательность латинских букв, цифр и символов «_», если идентификатор совпадает с ключевым словом, то его необходимо заключить в одинарные кавычки;
- ключевые слова – множество зарезервированных слов, обычно указываются в верхнем регистре;
- специальные символы представлены в таблице 1.7.
- пробельные символы – символы, с помощью которых разделяются компоненты (пробел, табуляция, перевод строки).

Таблица 1.7 – Специальные символы грамматики SQL.

Символ	Название (англ.)
>=	Greater than or equals operator
<=	Less than or equals operator
<>	Not equals operator
*	Asterisk
,	Comma
=	Equals operator
>	Greater than operator
<	Less than operator
(Left paren
)	Right paren
-	Minus sign
+	Plus sign
.	Period
;	Semicolon
/	Solidus

Также стоит отметить, что грамматика SQL не учитывает регистр символов для ключевых слов, поэтому «*SELECT*» == «*select*» (далее ключевые слова будут указываться в верхнем регистре).

Можно выделить основных 22 ключевых слова, представленных на листинге 1.1.

Листинг 1.1 – Основные ключевые слова языка SQL

AND; AS; CROSS; DISTINCT; FALSE; FROM; GROUP BY; HAVING; INNER; IS; JOIN; LEFT; NOT; NULL; ON; OR; ORDER BY; OUTER; RIGHT; SELECT; TRUE; WHERE

1.7 Синтаксический анализатор

В данной работе будет использоваться только оператор выборки данных – SELECT, поэтому нет необходимости реализовывать полную поддержку грамматики SQL (в данной работе рассматривался стандарт SQL:1999).

В приложении А представлен диалект грамматики SQL, в формате расширенной формы Бэкуса-Наура¹(РФБН).

Один из отличий грамматики SQL:1999 от той, которую необходимо будет реализовать и использовать, это возможность использования расширенного имени для таблицы.

В стандарте SQL:1999 полное имя таблицы формируется из трех частей: имя базы данных, имя схемы и имя таблицы внутри заданной схемы. Так как данное приложение позволяет работать с множеством различных СУБД, то в полном имени таблицы должен упоминаться уникальный идентификатор СУБД, поэтому в используемой грамматике полное имя таблицы формируется из четырех частей: имя СУБД, имя базы данных, имя схемы и имя таблицы внутри заданной схемы.

1.8 Семантический анализатор

Работа семантического анализатора начинается в том случае, если не было ошибок в работе лексического и синтаксического анализатора.

На стадии семантического анализа запроса производится проверка всех введенных значений.

1. Производится проверка существования таблиц, а также собирается информация о существующих колонках в данной таблице.
2. Производится проверка условия для JOIN, а именно проверка существования колонок, используемых в логических условиях.
3. Проверка списка запрашиваемых колонок, как существующих, так и генерируемых на основе существующих.
4. Проверка логического условия WHERE (на этом этапе так же, как и на этапе проверки условий для JOIN, производится проверка на существование колонок).

¹Формы Бэкуса-Наура – подход к описанию синтаксических конструкций, при котором одни фрагменты выражения определяются через другие [7].

1.9 Вывод

В данном разделе был проведен обзор языка SQL, анализ различий диалектов PostgreSQL и MySQL, представлено описание лексического, синтаксического и семантического анализаторов, а также разобрана тернарная логика, используемая в SQL.

2 Конструкторский раздел

В данном разделе будут рассмотрены алгоритмы работы анализаторов.

2.1 Общий алгоритм работы программы

Работу данного предложения можно разделить на следующие шаги.

1. Лексический, синтаксический, семантический анализы запроса, полученного на входе.
2. Генерация запросов к различным СУБД.
3. Выборка необходимых данных из различных СУБД.
4. Формирование результата с условием запроса пользователя.

На рисунке 2.1 представлена схема общего алгоритма работы приложения.

На рисунках 2.2-2.3 представлена диаграмма в нотации IDEF0 [8].

2.2 Реализация лексического анализатора

Основные лексические домены грамматики SQL можно описать при помощи регулярных выражений. На листинге 2.1 представлены регулярные выражения для доменов идентификаторов, ключевых слов, целых чисел и чисел и плавающей точкой.

Листинг 2.1 – Регулярные выражения для доменов идентификаторов

```
<IDENTIFIER> ::= r '[a-zA-Z_][a-zA-Z_0-9]*|'[a-zA-Z_][a-zA-Z_0-9]*'
```

```
<KEYWORD> ::= r '[a-zA-Z_][a-zA-Z_0-9]*'
```

```
<INTEGER> ::= r '([1-9]\d*|0)(?![0-9A-Za-z_])'
```

```
<FLOAT> ::= r '([1-9]\d*|0)?\.\d+|([1-9]\d*|0)\.)(?![0-9A-Za-z_])'
```

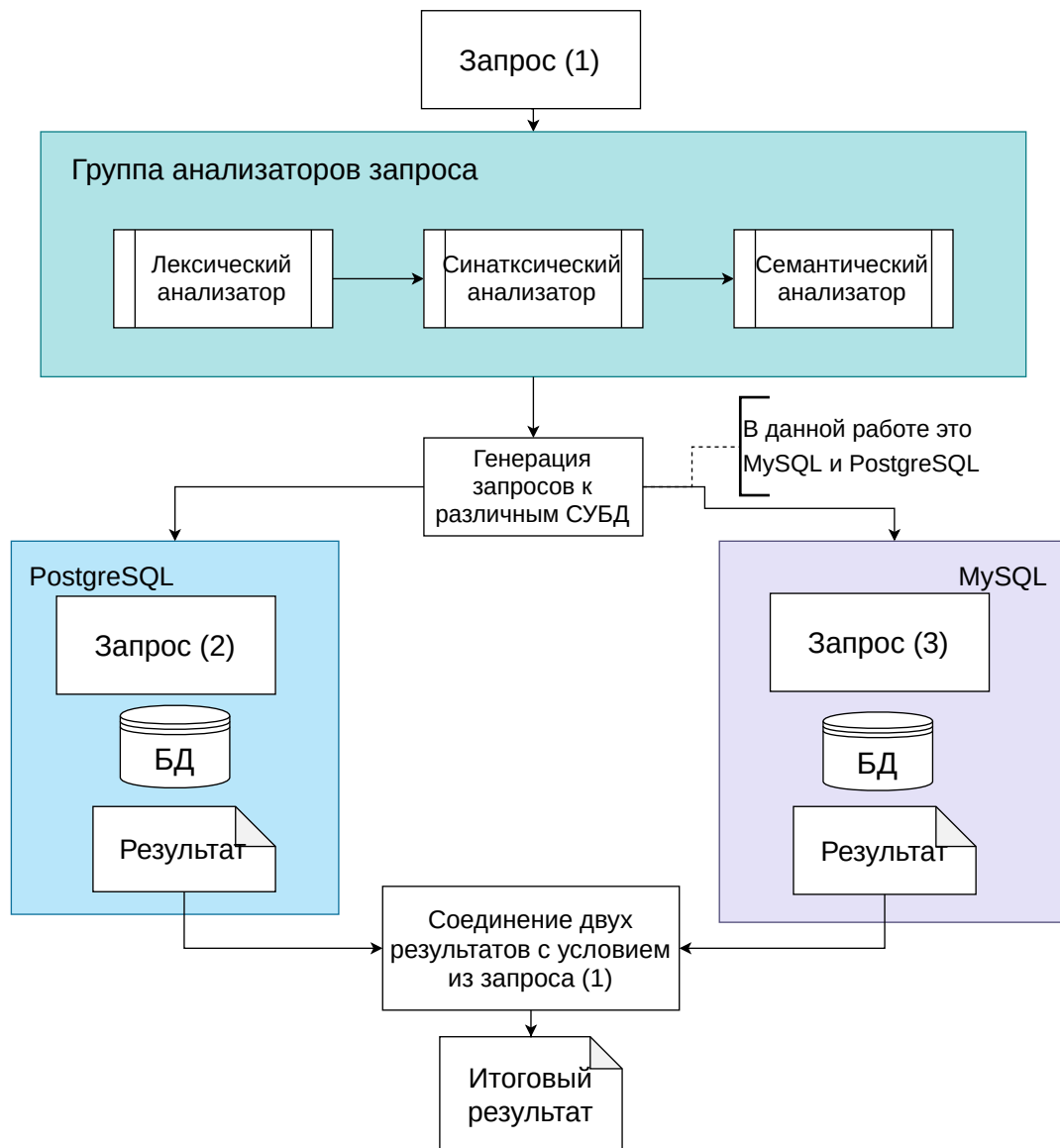


Рисунок 2.1 – Общий алгоритм работы программы

Следует отметить, что лексема числа должна быть разделена от лексемы идентификатора пробельным символом, то есть можно сказать, что не допустима следующая подстрока: «1234qwerty», именно по этой причине в регулярных выражениях есть проверка на символ, следующий за числом.

Чтобы проверить домен на принадлежность к ключевым словам, найденное слово приводится к верхнему регистру и проверяется на принадлежность ключевых слов.

Чтобы сократить затраты ресурсов памяти для компьютера, работа лек-

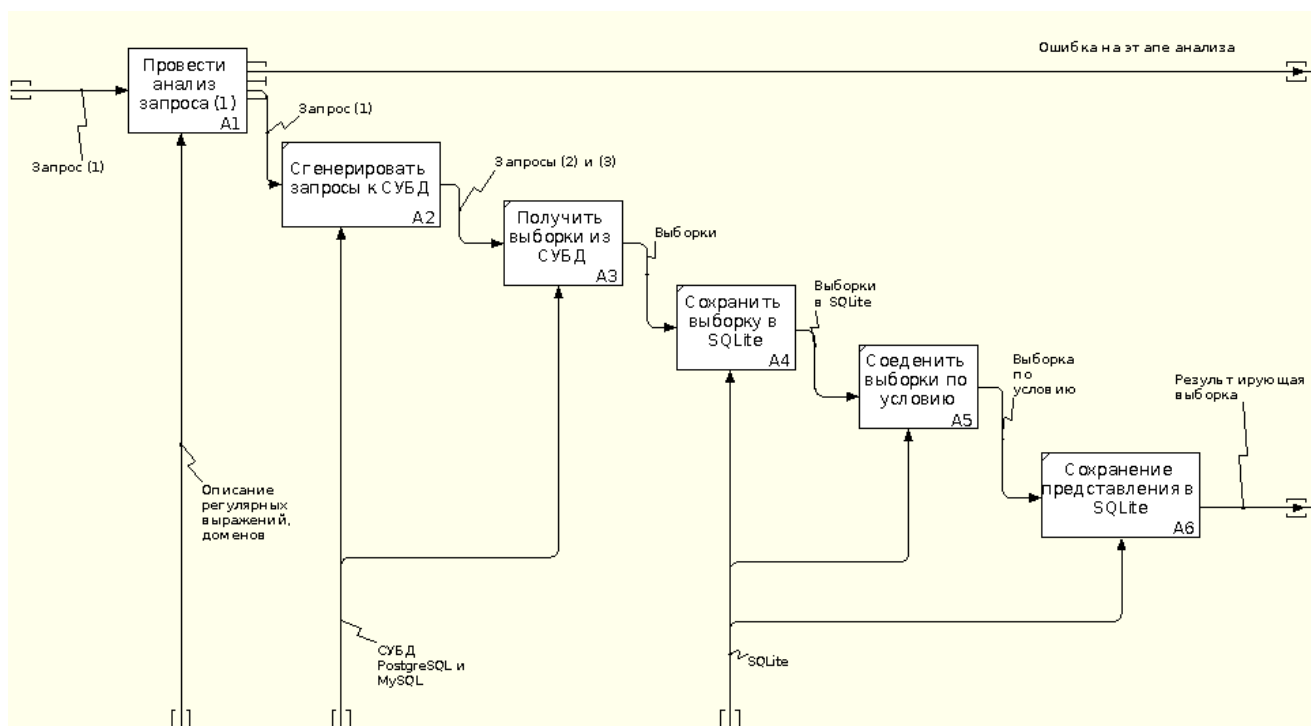


Рисунок 2.2 – Диаграмма в нотации IDEF0 (1)

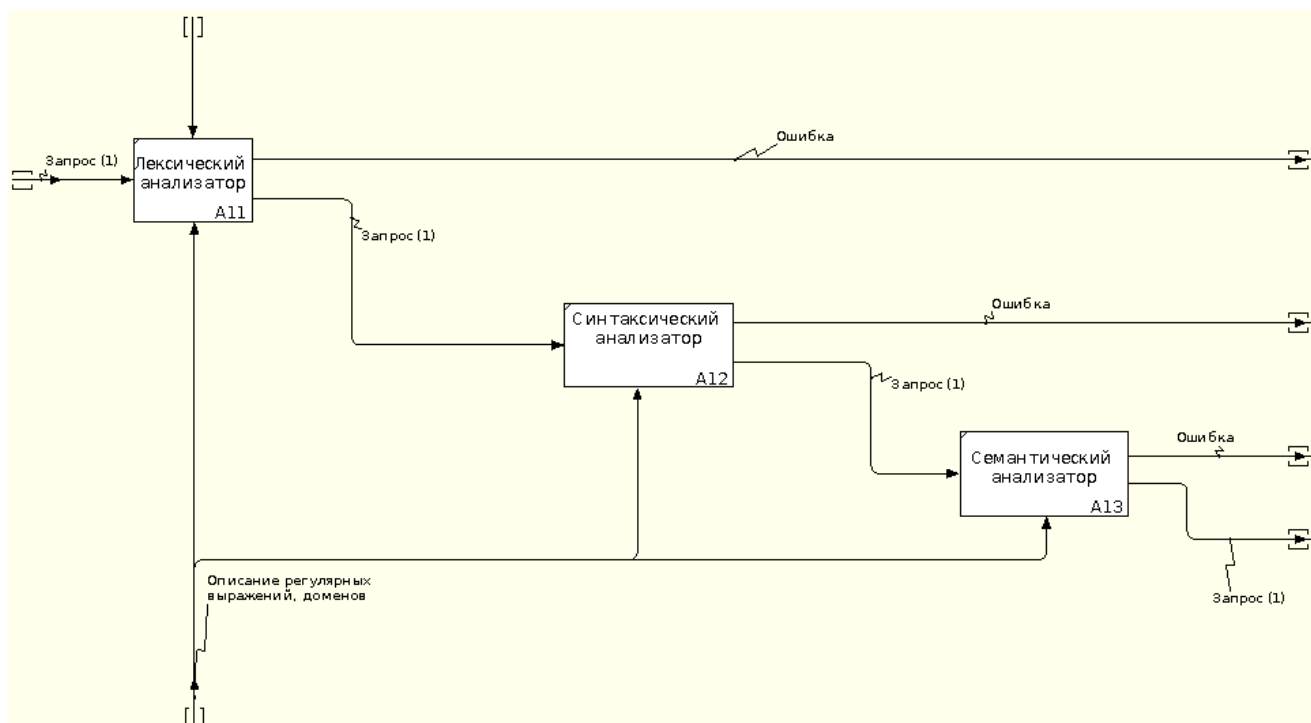


Рисунок 2.3 – Диаграмма в нотации IDEF0 (2)

сического анализатора совмещена с работой синтаксического анализатора.

В каждый момент работы программы в памяти хранится информация только об одной лексеме, это позволяет сократить затраты памяти.

На рисунке 2.4 представлена схема работы лексического анализатора.

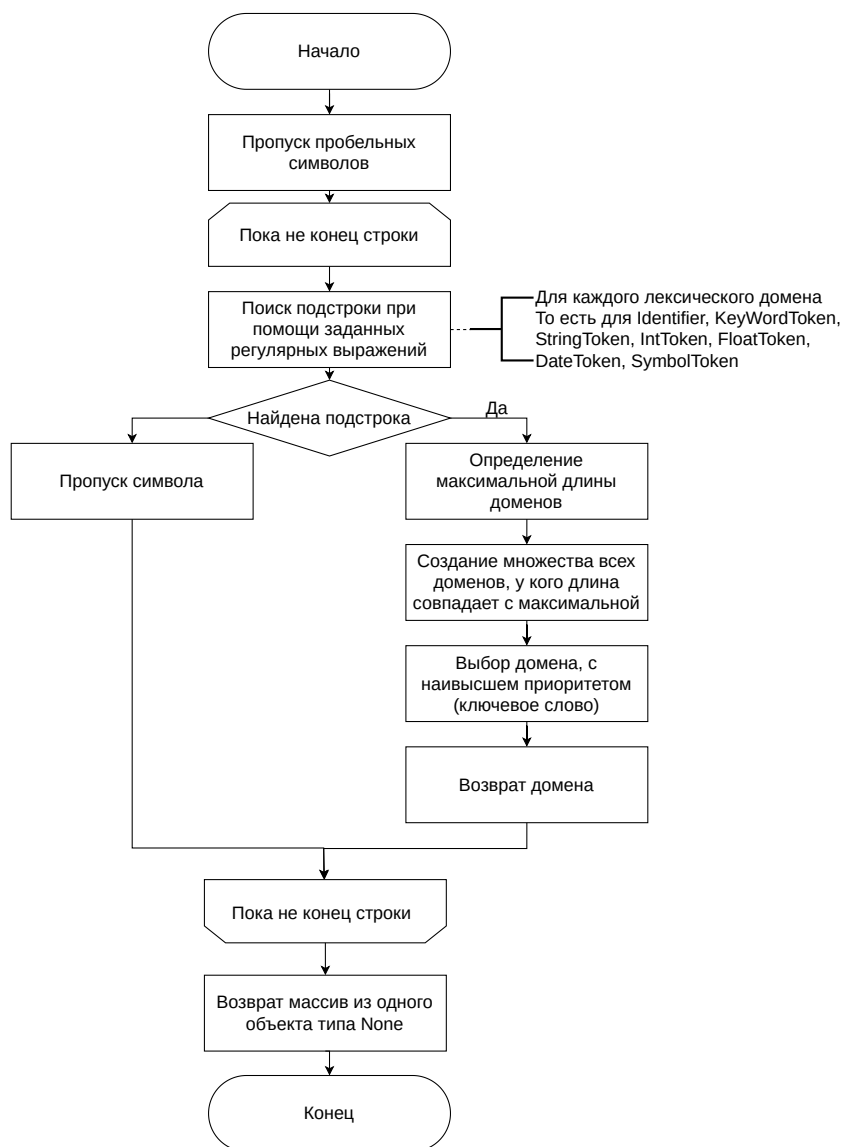


Рисунок 2.4 – Схема алгоритма работы лексического анализатора

2.3 Реализация синтаксического анализатора

Синтаксический анализатор реализован при использовании метода рекурсивного спуска с возможностью отката. Данный метод просто реализовать, а также легко расширять путем добавления новых инструкций.

На рисунке 2.5 представлена схема работы синтаксического анализатора.

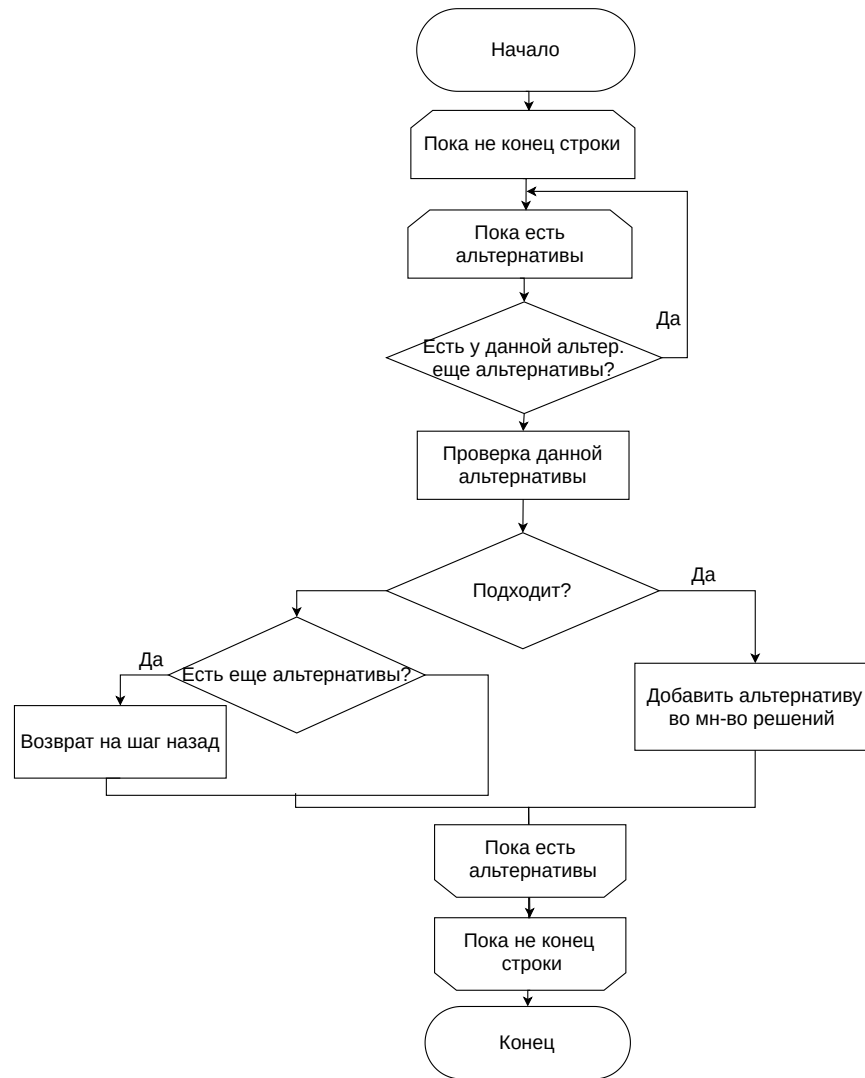


Рисунок 2.5 – Схема алгоритма работы синтаксического анализатора

2.4 Реализация семантического анализатора

Как было сказано ранее, работа семантического анализа начинается только в том случае, если не было ошибок в работе лексического и синтаксического анализатора.

На рисунке 2.6 представлена схема работы семантического анализатора с комментариями.



Рисунок 2.6 – Схема алгоритма работы семантического анализатора

2.5 Формирование и хранение результата

После получения результатов из PostgreSQL и MySQL, как показано на рисунке 2.1, данные переносятся во временную базу данных SQLite [9].

По умолчанию база данных создается в оперативной памяти, так как это позволяет сократить время, которое тратится на запись в файл и чтение из файла.

Для формирования результирующей выборки в базе данных SQLite определялось представление (VIEW¹). Использование представлений позволяет не сохранять результирующую выборку в оперативной памяти, так как ее объем может быть в разы больше чем данные, которые используются при ее построении.

На рисунке 2.7 представлена диаграмма в нотации IDEF0, описывающая процесс формирования и хранения результата в SQLite.

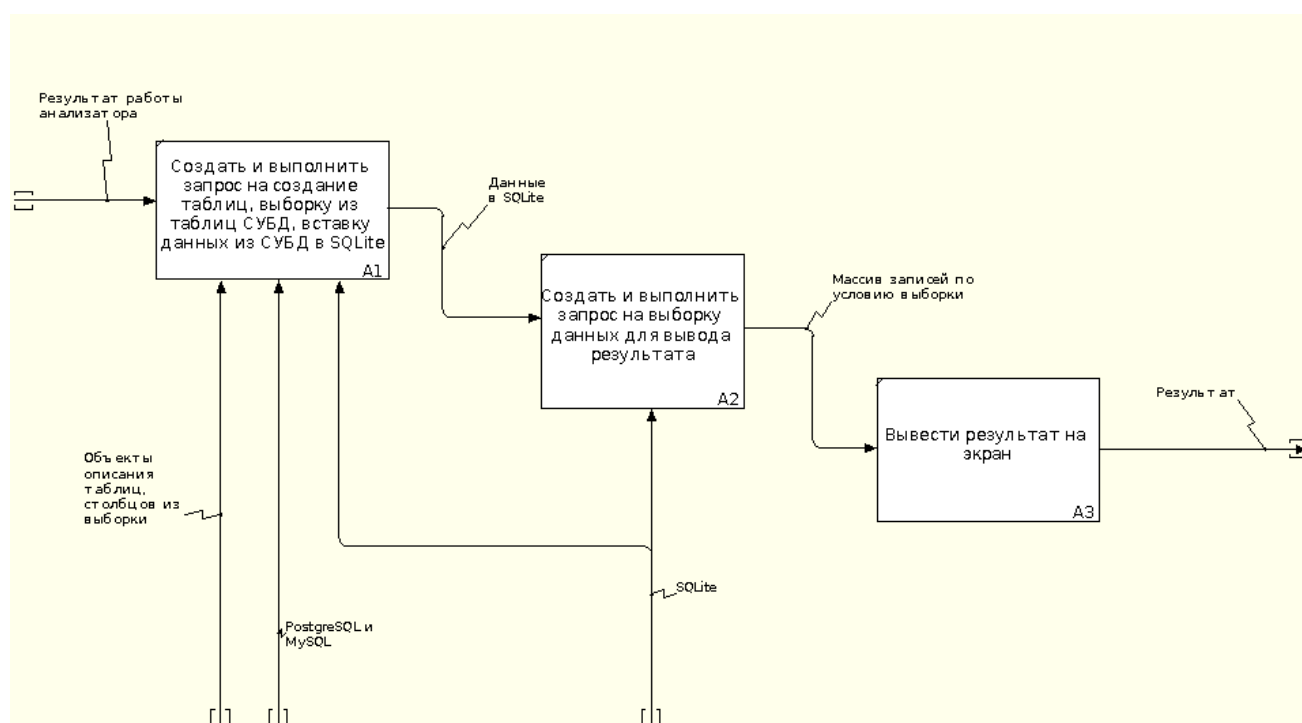


Рисунок 2.7 – Описание процесса формирования и хранения результата в SQLite

В данной работе предусматривается, что данных, хранящихся в MySQL и PostgreSQL, гарантированно меньше, чем объем оперативной памяти.

¹VIEW – это хранимый запрос в базе данных.

2.6 Вывод

В данном разделе были подробно рассмотрены алгоритмы работы лексического, синтаксического и семантического анализаторов, а также общий алгоритм работы программы.

3 Технологический раздел

В этом разделе будет обоснован выбор языка программирования и среды разработки, рассмотрена диаграмма основных классов и разобран интерфейс, предлагаемый пользователю.

3.1 Выбор языка программирования и среды разработки

При разработке программного продукта был использован язык программирования Python (версия 3.8.10) [10].

Данный выбор был сделан по следующим причинам.

1. Опыт работы с рассматриваемым языком.
2. Поддержка объектно-ориентированного подхода к программированию.
3. Большое количество литературы, связанной с ЯП Python.

В качестве среды разработки были использованы PyCharm [11] и Visual Studio Code [12], поскольку они бесплатны для студентов, удобны в процессе разработки и ранее активно использовались.

Время работы приложения измерялось при помощи функции *process_time()* из библиотеки *time* [13]. Для создания GUI (Graphical User Interface) был использован модуль *PyQt5* [14].

3.2 Подключение к базам данных

Для подключения к той или иной базе данных нужно иметь информацию о ней: адрес подключения, используемый драйвер, тип СУБД, логин и пароль пользователя.

Для ввода данной информации в интерфейсе предоставлены специальные поля. На рисунке 3.1 показаны данные поля.

БАЗА db1
PostgreSQL Driver
'{PostgreSQL Unicode}'
PostgreSQL Server
127.0.0.1
PostgreSQL Port
5432
PostgreSQL Логин
postgres
PostgreSQL Пароль
postgres
PostgreSQL Type
psql

БАЗА db2
MySQL Driver
'{MySQL}'
MySQL Server
127.0.0.1
MySQL Port
3306
MySQL Логин
admin
MySQL Пароль
admin
MySQL Type
mysql

OK

Рисунок 3.1 – Поля для подключения к СУБД

В данном примере *db1* и *db2* уникальное имя СУБД, которое будет использовать пользователь, чтобы указать какая именно БД используется в данном запросе. *PostgreSQLDriver* и *MySQLDriver* – названия драйверов, данные поля уже заполнены по умолчанию. *PostgreSQLServer* и *MySQLServer* – адреса баз данных, поля заполнены для подключению к локальным базам данных. *PostgreSQLType* и *MySQLType* – типы СУБД.

3.3 Используемая база данных

На рисунке 3.2 представлена ER-диаграмма базы данных, которая использовалась для проверки работы приложения.

Данная база данных была разделена по различным СУБД, как показано на рисунке 3.2, а также для проверки работы данная база данных была расположена в одной СУБД (PostgreSQL).

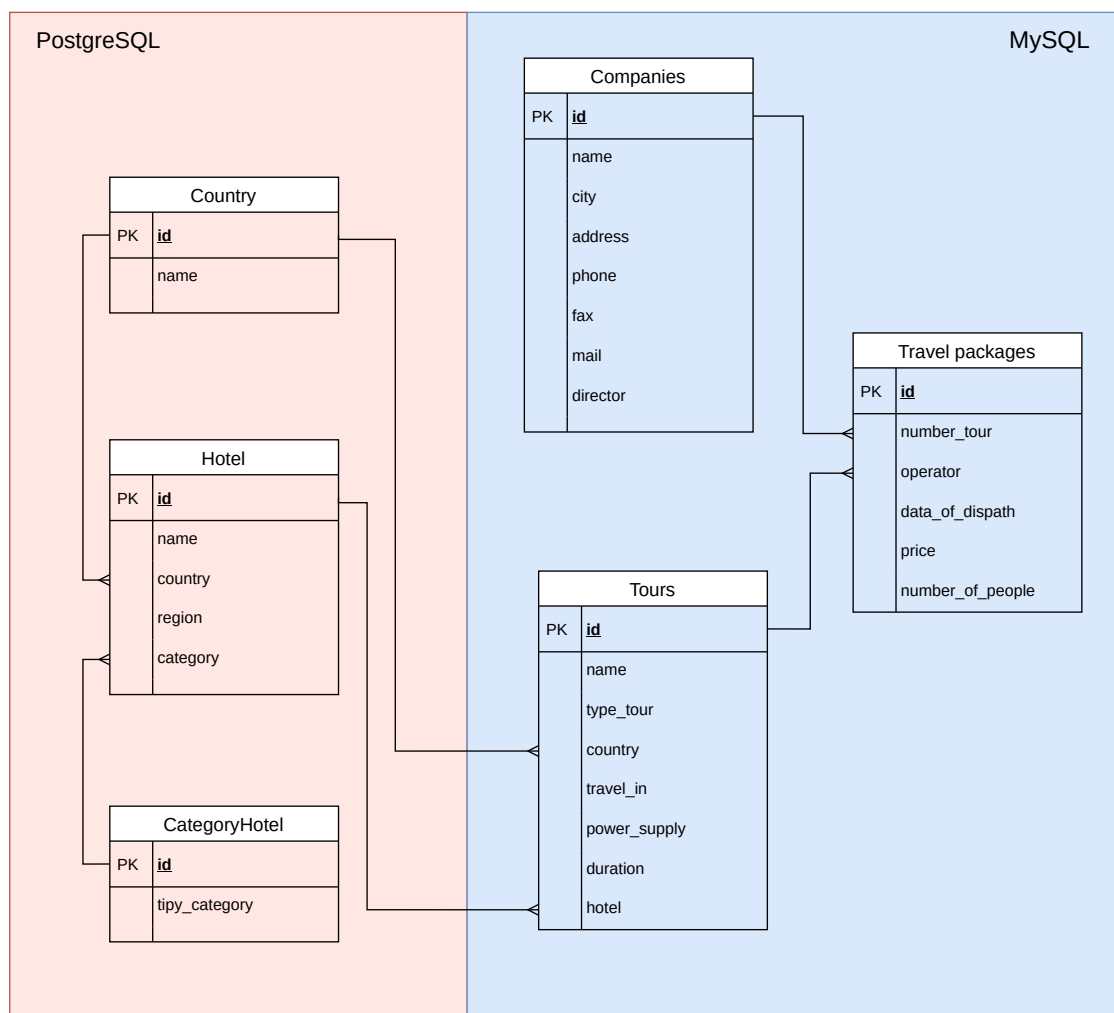


Рисунок 3.2 – Диаграмма БД

3.3.1 Создание таблиц

В соответствии со схемой выше было создано 3 таблицы в PostgreSQL СУБД и 3 таблицы в СУБД MySQL.

Создание некоторых из них приведено, а именно hotels в PostgreSQL и tours в MySQL представлено в листинге 3.1.

Листинг 3.1 – Создание некоторых таблиц

```
— PostgreSQL
CREATE TABLE IF NOT EXISTS Hotels(
    HotelId INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    NameHotel VARCHAR(100) NOT NULL, — имя
    CountryId INT, — номер страны
    Region VARCHAR(100) NOT NULL, — название региона
    CategoryId — номер категории
);

— MySQL
CREATE TABLE sys.Tours(
    TourId INT PRIMARY KEY,
    NameTour VARCHAR(100) NOT NULL, — имя тура
    TypeTour VARCHAR(100) NOT NULL, — тип тура
    CountryId INT, — номер страны
    TravelOnOff INT, — дорога входит или нет
    PowerSupply INT, — питание сколько раз
    Duration INT, — длительность
    HotelId INT — номер отеля
);
```

Следует отметить, что в PostgreSQL используется автогенерация первичного ключа (в таблице hotels это hotelid).

3.3.2 Наполнение таблиц

Таблицы заполняются сгенерированными данными с помощью библиотеки Faker [15].

Полученные данные записываются в файл, из которого далее будут подгружаться данные. На листинге 3.2 приведены функции генерации данных

для таблицы отелей (hotels) и туров (tours). Другие таблицы заполняются аналогично.

Листинг 3.2 – Генерация данных для таблиц

```
# Функция генерации данных для таблицы Отелей
cat_hotel = [1, 2, 3, 4, 5]
def generate_hotel():
    faker = Faker()
    f = open('hotel.csv', 'w')
    for i in range(MAX_N):
        country = randint(1, 239)
        line = "{0};{1};{2};{3}\n".format(
            faker.name(),
            country,
            faker.country() + "Region" + str(choice(cat_hotel)),
            choice(cat_hotel)
        )
        f.write(line)
    f.close()

# Функция генерации данных для таблицы Туров
on_off = [0, 1]
eat = [1, 2, 3]
t_tour = ["beach", "excursion", "sports", "quiet", "mobile"]
def generate_tours():
    faker = Faker()
    f = open('tour.csv', 'w')
    for i in range(MAX_N):
        week = randint(1, 6)
        hotel = randint(1, 999)
        country = randint(1, 239)
        line = "{0};{1};{2};{3};{4};{5};{6}\n".format(
            faker.name(),
            choice(t_tour),
            country,
            choice(on_off),
            choice(eat),
            week,
            hotel,
        )
        f.write(line)
    f.close()
```

3.4 Структура программы

При написании программы используется язык Python, что дает возможность использовать его со стороны объектно-ориентированного подхода.

Условно классы в программе можно разделить на следующие группы.

1. `Semantic_analyzer` – класс, описывающий работу семантического анализатора.
2. `BaseDialect`, `MySQL`, `PostgreSQL` – группа классов, описывающих диалекты `MySQL` и `PostgreSQL`.
3. `BaseToken`, `EndToken`, `IntToken`, `IdentifierToken`, `StringToken`, `FloatToken`, `SymbolToken`, `KeywordToken`, `DateToken` – группа классов, описывающих все возможные токены.
4. `Sintactic_analyzer`, `SQLParser` – группа классов, описывающих синтаксический анализатор.
5. `Lexer_analyzer` – группа классов, описывающих лексический анализатор.
6. `BaseJoin`, `InnerJoin` – группа классов, описывающих работу `Inner Join`.

На рисунке 3.3 представлена структура и состав классов.

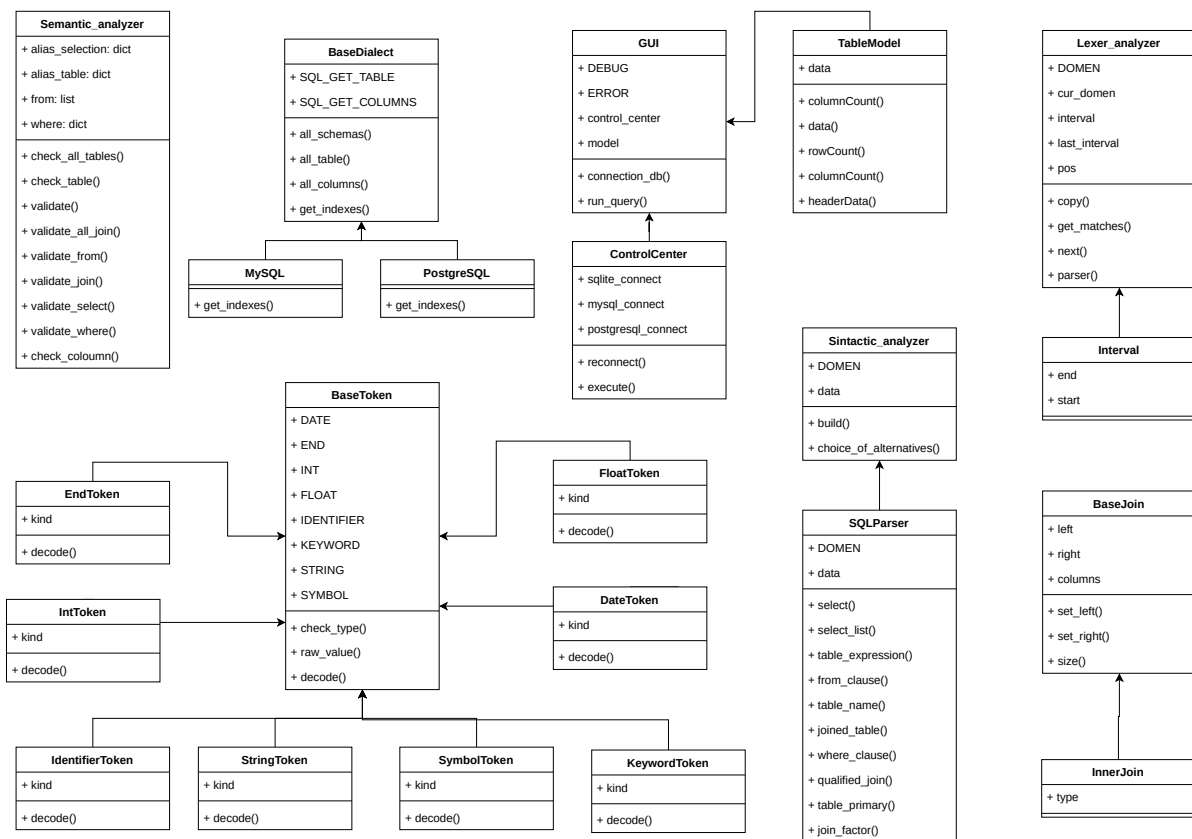


Рисунок 3.3 – Структура программы

3.5 Интерфейс

На рисунке 3.4 представлен интерфейс программы.

Функции предоставленных кнопок следующие:

- поле «ЗАПРОС» – поле для ввода SQL запроса;
- поле «ОШИБКИ» – если в результате выполнения запроса возникают ошибки, то эти ошибки пишутся в данное поле на экране;
- поле «РЕЗУЛЬТАТ» – поля для вывода результата;
- кнопка «ОК» – выполняется подключение к базам данных;
- кнопка «Выполнить запрос» – выполняется запрос, записанный в поле «ЗАПРОС»

The screenshot shows a software interface for connecting to and querying databases. It is divided into two main parts: one for PostgreSQL (БАЗА db1) and one for MySQL (БАЗА db2). Each part contains configuration fields for driver, server, port, login, and password. The PostgreSQL part also has a query input field and an error display area. The MySQL part has a result display area. A central button allows executing the query, and an OK button is at the bottom left.

Рисунок 3.4 – Интерфейс приложения

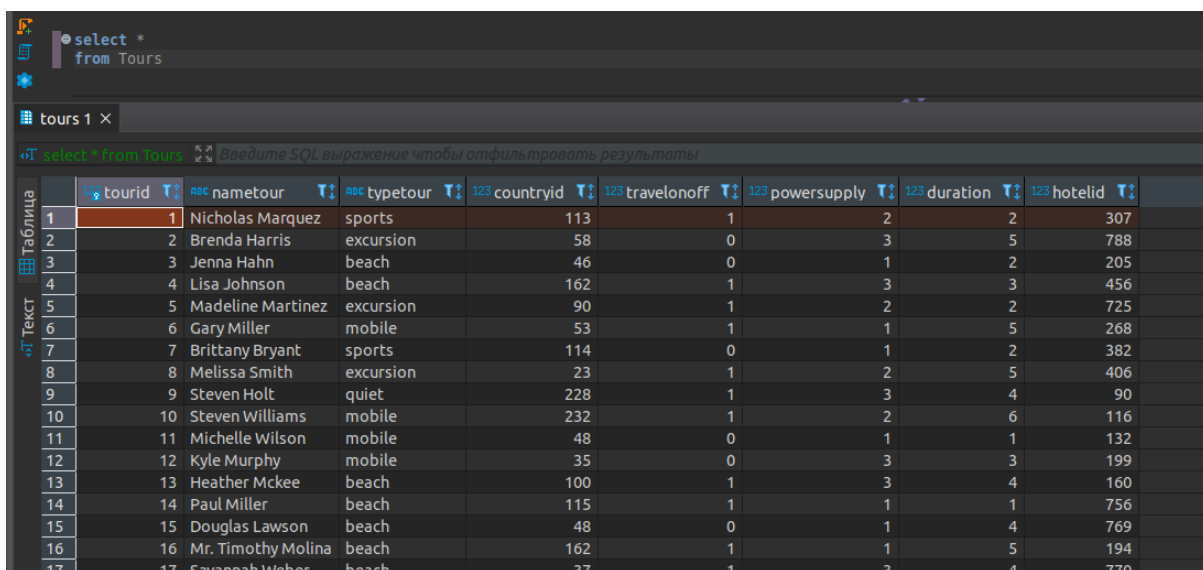
3.6 Результаты работы программного обеспечения

На рисунке 3.5 представлен результат запроса 3.3, выполненного непосредственно напрямую в СУБД MySQL.

На рисунке 3.6 представлен результат этого же запроса (запрос 3.3), выполненного с помощью приложения.

Листинг 3.3 – Запрос в MySQL

```
select *
from public.sys.Tours
```



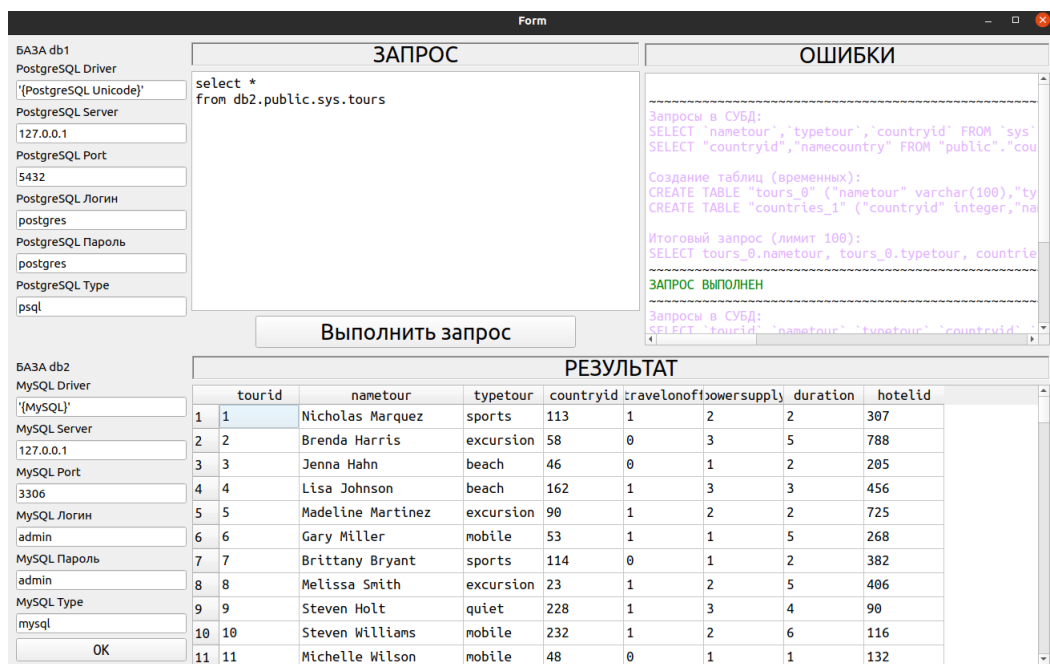
select *
from Tours

tours 1 x

select * from Tours

	tourid	nametour	typetour	countryid	travelonoff	powersupply	duration	hotelid
1	1	Nicholas Marquez	sports	113	1	2	2	307
2	2	Brenda Harris	excursion	58	0	3	5	788
3	3	Jenna Hahn	beach	46	0	1	2	205
4	4	Lisa Johnson	beach	162	1	3	3	456
5	5	Madeline Martinez	excursion	90	1	2	2	725
6	6	Gary Miller	mobile	53	1	1	5	268
7	7	Brittany Bryant	sports	114	0	1	2	382
8	8	Melissa Smith	excursion	23	1	2	5	406
9	9	Steven Holt	quiet	228	1	3	4	90
10	10	Steven Williams	mobile	232	1	2	6	116
11	11	Michelle Wilson	mobile	48	0	1	1	132
12	12	Kyle Murphy	mobile	35	0	3	3	199
13	13	Heather McKee	beach	100	1	3	4	160
14	14	Paul Miller	beach	115	1	1	1	756
15	15	Douglas Lawson	beach	48	0	1	4	769
16	16	Mr. Timothy Molina	beach	162	1	1	5	194
17	17	Savannah Weber	beach	37	1	3	4	770

Рисунок 3.5 – Результат запроса в MySQL



Form

БАЗА db1
PostgreSQL Driver
'(PostgreSQL Unicode)'
PostgreSQL Server
127.0.0.1
PostgreSQL Port
5432
PostgreSQL Логин
postgres
PostgreSQL Пароль
postgres
PostgreSQL Type
psql

ЗАПРОС

```
select *  
from db2.public.sys.tours
```

ОШИБКИ

Запросы в СУБД:
SELECT 'nametour', 'typetour', 'countryid' FROM 'sys'
SELECT 'countryid', 'namecountry' FROM 'public', 'cou

Создание таблиц (временных):
CREATE TABLE "tours_0" ("nametour" varchar(100), "ty
CREATE TABLE "countries_1" ("countryid" integer, "na

Игровой запрос (лимит 100):
SELECT tours_0.nametour, tours_0.typetour, countrie

ЗАПРОС ВЫПОЛНЕН

Запросы в СУБД:
SELECT 'tourid', 'nametour', 'typetour', 'countryid', '

Выполнить запрос

БАЗА db2
MySQL Driver
'(MySQL)'
MySQL Server
127.0.0.1
MySQL Port
3306
MySQL Логин
admin
MySQL Пароль
admin
MySQL Type
mysql

РЕЗУЛЬТАТ

	tourid	nametour	typetour	countryid	travelonoff	powersupply	duration	hotelid
1	1	Nicholas Marquez	sports	113	1	2	2	307
2	2	Brenda Harris	excursion	58	0	3	5	788
3	3	Jenna Hahn	beach	46	0	1	2	205
4	4	Lisa Johnson	beach	162	1	3	3	456
5	5	Madeline Martinez	excursion	90	1	2	2	725
6	6	Gary Miller	mobile	53	1	1	5	268
7	7	Brittany Bryant	sports	114	0	1	2	382
8	8	Melissa Smith	excursion	23	1	2	5	406
9	9	Steven Holt	quiet	228	1	3	4	90
10	10	Steven Williams	mobile	232	1	2	6	116
11	11	Michelle Wilson	mobile	48	0	1	1	132

OK

Рисунок 3.6 – Результат запроса, выполненного через приложение

На рисунке 3.7 представлен результат запроса 3.4, выполненного непосредственно напрямую в СУБД PostgreSQL.

На рисунке 3.8 представлен результат этого же запроса (запрос 3.4), выполненного с помощью приложения.

Листинг 3.4 – Запрос в PostgreSQL

```
select *  
from postgres.public.hotels
```

The screenshot shows a PostgreSQL client window with the query `select * from postgres.public.hotels` entered. Below the query editor, a table titled "hotels 1" displays the results. The table has five columns: `hotelid`, `namehotel`, `countryid`, `region`, and `categoryid`. The results are listed in 15 rows.

	hotelid	namehotel	countryid	region	categoryid
1	1	Robert Lawson	188	El SalvadorRegion1	5
2	2	Karen Richardson	188	PanamaRegion1	4
3	3	Angelica Solomon	97	Cape VerdeRegion4	2
4	4	Jared Griffith	47	BelgiumRegion5	2
5	5	Paul Johnson	192	NigerRegion2	2
6	6	Samantha Deleon	114	JapanRegion5	2
7	7	Richard Coleman	18	AustraliaRegion5	2
8	8	Cody Davis	141	FranceRegion4	4
9	9	Wayne Rogers	42	TurkeyRegion2	4
10	10	Christina Gonzalez	89	CroatiaRegion5	5
11	11	Christine Simmons	75	PalauRegion1	4
12	12	Jessica Lewis	237	CongoRegion4	2
13	13	Mark Rodriguez	36	TanzaniaRegion1	2
14	14	Teresa Pope	2	GermanyRegion5	1
15	15	Matthew Smith	14	SpainRegion3	4

Рисунок 3.7 – Результат запроса в PostgreSQL

The screenshot shows a database client application window titled "Form". It has two main panes: "ЗАПРОС" (Query) and "ОШИБКИ" (Errors). The "ЗАПРОС" pane contains the query `select * from db1.postgres.public.hotels`. The "ОШИБКИ" pane shows the execution log, including the query, the creation of a temporary table `hotels_0`, and the execution of the query. Below the panes is a button labeled "Выполнить запрос" (Execute query). Below the button is a table titled "РЕЗУЛЬТАТ" (Result) showing the results of the query. The table has five columns: `hotelid`, `namehotel`, `countryid`, `region`, and `categoryid`. The results are listed in 11 rows.

	hotelid	namehotel	countryid	region	categoryid
1	1	Robert Lawson	188	El SalvadorRegion1	5
2	2	Karen Richardson	188	PanamaRegion1	4
3	3	Angelica Solomon	97	Cape VerdeRegion4	2
4	4	Jared Griffith	47	BelgiumRegion5	2
5	5	Paul Johnson	192	NigerRegion2	2
6	6	Samantha Deleon	114	JapanRegion5	2
7	7	Richard Coleman	18	AustraliaRegion5	2
8	8	Cody Davis	141	FranceRegion4	4
9	9	Wayne Rogers	42	TurkeyRegion2	4
10	10	Christina Gonzalez	89	CroatiaRegion5	5
11	11	Christine Simmons	75	PalauRegion1	4

Рисунок 3.8 – Результат запроса в PostgreSQL

3.7 Тестирование

На рисунке 3.9 представлен результат выполнения запроса 3.5.

Здесь и далее: в правой части рисунка результат, полученный приложением, в левой – результат, в случае, когда все таблицы расположены в одной СУБД (в данном случае PostgreSQL).

Данный запрос направлен на тестирование оператора JOIN двух таблиц из разных СУБД.

	nametour	typetour	namecountry
1	Nicholas Marquez	sports	Lebanon
2	Brenda Harris	excursion	Honduras
3	Jenna Hahn	beach	Gabon
4	Lisa Johnson	beach	Paraguay
5	Madeline Martinez	excursion	Cameroon
6	Gary Miller	mobile	Guiana
7	Brittany Bryant	sports	Libya
8	Melissa Smith	excursion	Belgium
9	Steven Holt	quiet	Sri Lanka
10	Steven Williams	mobile	Ethiopia
11	Michelle Wilson	mobile	Guyana

	nametour	typetour	namecountry
1	Nicholas Marquez	sports	Lebanon
2	Brenda Harris	excursion	Honduras
3	Jenna Hahn	beach	Gabon
4	Lisa Johnson	beach	Paraguay
5	Madeline Martinez	excursion	Cameroon
6	Gary Miller	mobile	Guiana
7	Brittany Bryant	sports	Libya
8	Melissa Smith	excursion	Belgium
9	Steven Holt	quiet	Sri Lanka
10	Steven Williams	mobile	Ethiopia
11	Michelle Wilson	mobile	Guyana
12	Kyle Murphy	mobile	Bhutan
13	Heather Mckee	beach	Congo
14	Paul Miller	beach	Liechtenstein
15	Douglas Lawson	beach	Guyana
16	Mr. Timothy Molina	beach	Paraguay
17	Savannah Weber	beach	Vatican

Рисунок 3.9 – Результат запроса 3.5

Листинг 3.5 – Запрос в PostgreSQL и приложения

— Запрос в СУБД PostgreSQL

```
select t.NameTour, t.TypeTour, c.NameCountry
from Tours as t join Countries as c on t.CountryId = c.CountryId
```

— Запрос в приложении

```
select t.nametour, t.typetour, c.namecountry
from db2.public.sys.tours as t join db1.postgres.public.countries as c on
t.countryid = c.countryid
```

На рисунке 3.10 представлен результат выполнения запроса 3.6.

Данный запрос направлен на тестирование оператора WHERE для выборки из таблицы, расположенной в MySQL.

РЕЗУЛЬТАТ				
companyId	namecompany	city	addresscom	phone
1 1	Desir	Morrisshire	156 Hall ...	96452546
2 2	Marvi	South Jason	USCGC ...	28072671
3 3	Teres	Walkerland	Unit 1010 Box ...	22484201
4 4	April	Paynetown	815 James ...	30526474
5 5	Bryan	Rebeccafort	4296 Grace ...	40217052
6 6	Steph	Lake Brandistad	633 Johnson ...	56058746
7 7	Jodi	Woodfurt	5834 Harris ...	18028344
8 8	Susan	West Rickyfort	99050 Amy Ways...	40756557
9 9	Danie	Lake Alan	15850 Mark ...	22027006

Рисунок 3.10 – Результат запроса 3.6

Листинг 3.6 – Запрос в MySQL и приложение

— Запрос в СУБД MySQL

```
select *
from sys.companies as c
where c.companyId < 10
```

— Запрос в приложении

```
select *
from db2.public.sys.companies as c
where c.companyId < 10
```

На рисунке 3.11 представлен результат выполнения запроса 3.7.

Данный запрос направлен на тестирование оператора WHERE для выборки из таблицы, расположенной в PostgreSQL.

countryId	namecountry
1 1	Australia
2 2	Austria
3 3	Azerbaijan
4 4	Albania
5 5	Algeria
6 6	Anguilla
7 7	Angola
8 8	Andorra
9 9	Antarctic

Рисунок 3.11 – Результат запроса 3.7

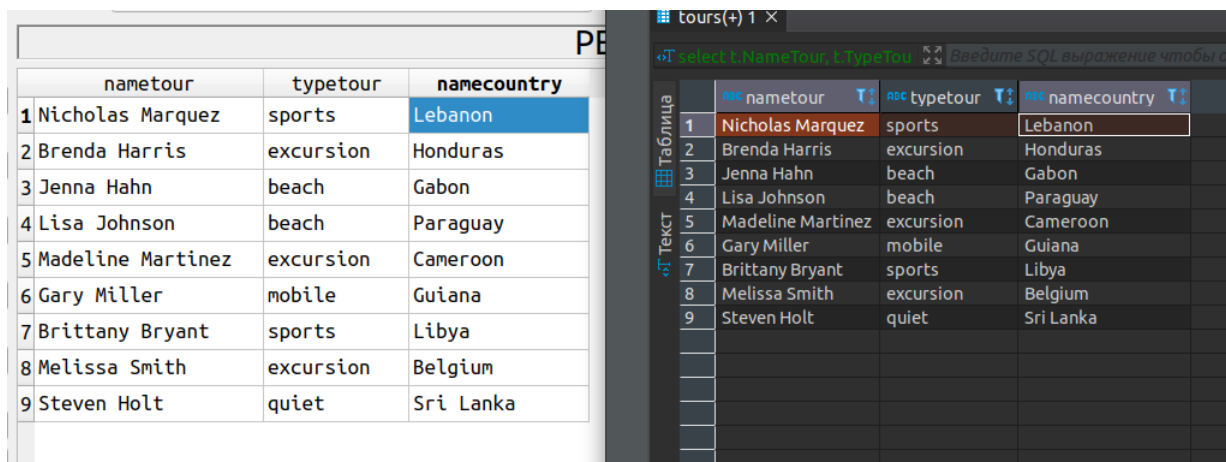
Листинг 3.7 – Запрос в PostgreSQL и приложение

```
— Запрос в СУБД PostgreSQL
select *
from postgres.public.countries c
where c.countryid < 10

— Запрос в приложении
select *
from db1.postgres.public.countries as c
where c.countryid < 10
```

На рисунке 3.12 представлен результат выполнения запроса 3.8.

Данный запрос направлен на тестирование операторов JOIN и WHERE одновременно.



	nametour	typetour	namecountry
1	Nicholas Marquez	sports	Lebanon
2	Brenda Harris	excursion	Honduras
3	Jenna Hahn	beach	Gabon
4	Lisa Johnson	beach	Paraguay
5	Madeline Martinez	excursion	Cameroon
6	Gary Miller	mobile	Guiana
7	Brittany Bryant	sports	Libya
8	Melissa Smith	excursion	Belgium
9	Steven Holt	quiet	Sri Lanka

Рисунок 3.12 – Результат запроса 3.8

Листинг 3.8 – Запрос в PostgreSQL и приложение

```
— Запрос в СУБД PostgreSQL
select t.NameTour, t.TypeTour, c.NameCountry
from Tours as t join Countries as c on t.CountryId = c.CountryId
where t.tourid < 10

— Запрос в приложении
select t.nametour, t.typetour, c.namecountry
from db2.public.sys.tours as t join db1.postgres.public.countries as c on
t.countryid = c.countryid
where t.tourid < 10
```

3.8 Вывод

В этом разделе был выбран язык программирования и среда разработки, рассмотрена UML-диаграмма основных классов, подробно разобран интерфейс приложения, приведены результаты тестирования, а также протестированное разработанное приложение.

В курсовой работе рассматривались лишь два оператора WHERE и JOIN. Исследование и разработка в этом направлении могут быть продолжены до реализации полного функционала работы СУБД.

4 Исследовательский раздел

В текущем разделе будет поставлена цель эксперимента, проведён сам эксперимент и сделаны соответствующие выводы.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- Операционная система: Ubuntu 20.04.3 [16] Linux [17] x86_64.
- Память: 8 GiB.
- Процессор: 11th Gen Intel® Core™ i5-1135G7 @ 2.40GHz [18];
- 4 физических ядра и 8 логических ядра.

Тестирование проводилось на ноутбуке, включенном в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения, а также непосредственно системой тестирования.

4.2 Постановка эксперимента

4.2.1 Цель эксперимента

Цель эксперимента – оценить зависимость времени работы от сложности запроса, поступающего на вход приложению.

4.2.2 Время работы приложения

Результаты замеров времени (в секундах) работы приложения приведены в таблице 4.1. Время посчитано как среднее арифметическое 100 замеров

времени выполнения запросов. Перед каждым запросом в СУБД MySQL и PostgreSQL выполняется очистка кеша.

Запросы отдельно в СУБД MySQL и PostgreSQL выполнялись с выборкой из 1-5 столбцов. В листинге 4.1 представлены запросы сразу с выборкой 5 столбцов.

Также в листинге 4.1 представлен запрос сразу в две СУБД одновременно, аналогично предыдущему, замеры выполнялись с выборкой 2-8 столбцов, запрос представлен сразу с выборкой 8 столбцов.

Листинг 4.1 – Запросы для проведения эксперимента

```
— Запрос в MySQL
select t.tourid , t.nametour , t.typetour , t.countryid , t.duration
from sys.tours as t

— Запрос в PostgreSQL
select h.hotelid , h.namehotel , h.countryid , h.region , h.categoryid
from hotels as h

— Запрос сразу в две СУБД
select t.tourid , t.nametour , t.typetour , t.duration , t.hotelid ,
      t.travelonoff , t.powersupply , c.namecountry
from db2.public.sys.tours as t join db1.postgres.public.countries as c on
      t.countryid = c.countryid
```

Результаты замеров времени представлены в таблице 4.1.

На рисунке 4.1 представлены графики, основанные на результатах замеров времени выполнения запросов отдельно в СУБД и через приложения.

На рисунке 4.2 представлен график замеров времени выполнения запроса одновременно в две СУБД через приложение.

Таблица 4.1 – Результаты замеров времени (сек) работы приложения

MySQL			
Запрос (кол-во столбцов)	Приложение	СУБД	Разница
1	0.000652	0.000395	0.000257
2	0.000784	0.000485	0.000298
3	0.000815	0.000536	0.000278
4	0.000909	0.000658	0.000351
5	0.000998	0.000789	0.000209
PostgreSQL			
Запрос (кол-во столбцов)	Приложение	СУБД	Разница
1	0.000721	0.000495	0.000225
2	0.000801	0.000575	0.000225
3	0.000858	0.000621	0.000237
4	0.000903	0.000721	0.000182
5	0.000989	0.000857	0.000132

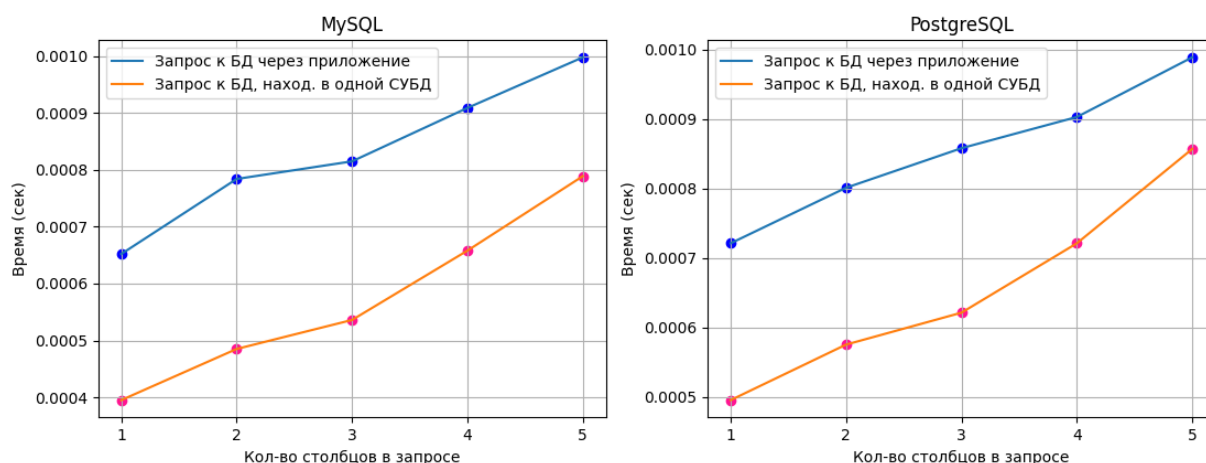


Рисунок 4.1 – Графики, основанные на результатах замеров времени выполнения запросов отдельно в СУБД и через приложения

4.3 Вывод

В результате эксперимента было выявлено, что запросы через приложения работают медленнее, чем запросы сразу в СУБД в случае, если в запросах используются таблицы, которые расположены в одной конкретной СУБД, что

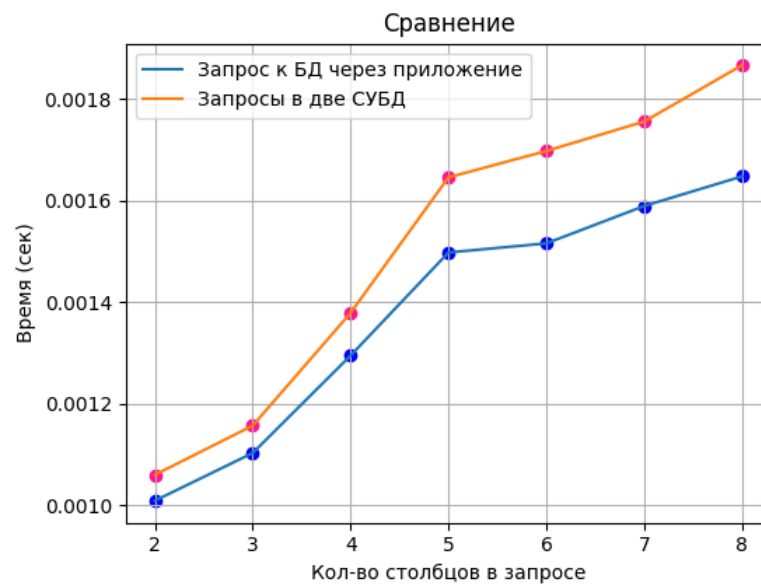


Рисунок 4.2 – График замеров времени выполнения запроса одновременно в две СУБД через приложение

видно на рисунке 4.1.

В случае, когда запрашиваются данные, находящиеся в двух различных СУБД, запрос, выполненный через приложение работает быстрее, чем запросы, выполняющиеся отдельно в каждую СУБД, что видно на рисунке 4.2.

ЗАКЛЮЧЕНИЕ

В ходе написания курсового проекта была достигнута поставленная цель, а именно разработано приложение для выполнения распределенного запроса, который использует данные из PostgreSQL и MySQL СУБД.

В процессе ее выполнения были решены следующие задачи:

- проанализирован язык SQL, а также группу операторов DML;
- проанализированы совместимость и различия PostgreSQL и MySQL;
- проанализирована работа оператора SELECT и табличные выражения;
- проанализированы и сформулированы ограничения на операторы JOIN и WHERE;
- разработано программное обеспечение, включающие в себя лексический, семантический и синтаксический анализаторы;
- было произведено сравнение времени выполнения запроса для баз данных, находящихся в различных СУБД и в одной СУБД.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Firebird RDBMS. URL: <https://firebirdsql.org> (дата обращения: 30.04.2022).
- [2] DBMS popularity broken down by database model. URL: https://db-engines.com/en/ranking_categories (дата обращения: 28.04.2022).
- [3] DB-Engines Ranking of Relational DBMS. URL: <https://db-engines.com/en/ranking/relational+dbms> (дата обращения: 30.04.2022).
- [4] Типы данных. URL: <https://www.postgresql.org/docs/14/datatype.html> (дата обращения: 01.05.2022).
- [5] MySQL Standards Compliance. URL: <https://dev.mysql.com/doc/refman/8.0/en/compatibility.html> (дата обращения: 01.05.2022).
- [6] PostgreSQL Conformance. URL: <https://www.postgresql.org/docs/current/features.html> (дата обращения: 29.04.2022).
- [7] Расширенная Бэкуса — Наура форма (РБНФ). URL: https://studme.org/380486/informatika/rasshirennaya_bekusa_naura_forma_rbnf (дата обращения: 05.05.2022).
- [8] Описание IDEF0. URL: <https://bpmn.pro/process/idef0> (дата обращения: 30.04.2022).
- [9] SQLite. URL: <https://www.sqlite.org/index.html> (дата обращения: 30.04.2022).
- [10] Welcome to Python. URL: <https://www.python.org> (дата обращения: 30.04.2022).
- [11] PyCharm. URL: <https://www.jetbrains.com/ru-ru/pycharm/> (дата обращения: 30.04.2022).
- [12] Visual Studio Code. URL: <https://code.visualstudio.com/> (дата обращения: 30.04.2022).

- [13] time — Time access and conversions. URL: <https://docs.python.org/3/library/time.html#functions> (дата обращения: 04.09.2021).
- [14] PyQt5. URL: <https://pypi.org/project/PyQt5/> (дата обращения: 05.05.2022).
- [15] Документация Faker. URL: <https://faker.readthedocs.io/en/master/> (дата обращения: 30.04.2022).
- [16] Ubuntu 20.04.3 LTS (Focal Fossa). URL: <https://releases.ubuntu.com/20.04/> (дата обращения: 30.04.2022).
- [17] Linux – Википедия. URL: <https://ru.wikipedia.org/wiki/Linux> (дата обращения: 30.04.2022).
- [18] Процессор Intel Core i5-1135G7. URL: <https://www.intel.ru/content/www/ru/ru/products/sku/208658/intel-core-i51135g7-processor-8m-cache-up-to-4-20-ghz/specifications.html> (дата обращения: 30.04.2022).

ПРИЛОЖЕНИЕ А

Диалект грамматики SQL в формате РБНФ.

Листинг 4.1 – диалект грамматики SQL в формате РБНФ

```
<select> ::= SELECT <select list> <table expression> [ <semicolon> ]

<select list> ::= <asterisk> | <select sublist> [ { <comma> <select sublist> } ]

<select sublist> ::= <qualified asterisk> | <derived column>

<qualified asterisk> ::= <asterisked identifier chain> <asterisk>

<asterisked identifier chain> ::= <IDENTIFIER> <period> [ { <IDENTIFIER>
    <period> } ]

<derived column> ::= <value expression> [ [ AS ] <IDENTIFIER> ]

<value expression> ::= <numeric value expression> | <boolean value expression>

<numeric value expression> ::= <term> | <term> <plus sign> <numeric value
    expression> | <term> <minus sign> <numeric value expression>

<term> ::= <factor> | <factor> <asterisk> <term> | <factor> <solidus> <term>

<factor> ::= [ <sign> ] <numeric primary>

<sign> ::= <minus sign> | <plus sign>

<numeric primary> ::= <value expression primary>

<boolean value expression> ::= <boolean term> | <<boolean term> OR <boolean
    value expression>

<boolean term> ::= <boolean factor> | <boolean factor> AND <boolean term>

<boolean factor> ::= [ NOT ] <boolean test>

<boolean test> ::= <boolean primary> [ IS [ NOT ] <truth value> ]

<truth value> ::= TRUE | FALSE | NULL

<boolean primary> ::= <predicate> | <value expression primary>
```


`<predicate> ::= <comparison predicate>`
`<comparison predicate> ::= <operand comparison> <comp op> <operand comparison>`
`<operand comparison> ::= <not boolean value expression> | <parenthesized value expression>`
`<not boolean value expression> ::= <numeric value expression>`
`<comp op> ::= <equals operator> | <not equals operator> | <less than operator> | <greater than operator> | <less than or equals operator>`
`<value expression primary> ::= <parenthesized value expression> | <nonparenthesized value expression primary>`
`<parenthesized value expression> ::= <left paren> <value expression> <right paren>`
`<nonparenthesized value expression primary> ::= <unsigned value specification> | <column reference>`
`<unsigned value specification> ::= <unsigned literal>`
`<unsigned literal> ::= <unsigned numeric literal> | <general literal>`
`<unsigned numeric literal> ::= <INTEGER> | <FLOAT>`
`<general literal> ::= TRUE | FALSE | NULL`
`<column reference> ::= <basic identifier chain>`
`<basic identifier chain> ::= <identifier chain>`
`<identifier chain> ::= <IDENTIFIER> [{ <period> <IDENTIFIER> }]`
`<table expression> ::= <from clause> [<where clause>]`
`<from clause> ::= FROM <table reference list>`
`<table reference list> ::= <table reference> [{ <comma> <table reference> }]`
`<table reference> ::= <join factor> [{ <join type> }]`
`<join factor> ::= <table primary> | <left paren> <table reference> <right paren>`

```
<table primary> ::= <table or query name> [ [ AS ] <IDENTIFIER> ]  
  
<table or query name> ::= <table name>  
  
<table name> ::= <basic identifier chain>  
  
<joined table> ::= <cross join> | <qualified join>  
  
<cross join> ::= CROSS JOIN <join factor>  
  
<qualified join> ::= [ <join type> ] JOIN <join factor> <join specification>  
  
<join type> ::= INNER | <outer join type> [ OUTER ]  
  
<outer join type> ::= LEFT | RIGHT  
  
<join specification> ::= <join condition>  
  
<join condition> ::= ON <search condition>  
  
<search condition> ::= <boolean value expression>  
  
<where clause> ::= WHERE <search condition>
```