ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Журнал практики

Институт № 8 «Компьютерные науки и прикладная математика»

Кафедра 806 Учебная группа М8О-406Б-21

ФИО обучающегося Мезенин Олег Александрович

Направление подготовки/ специальность 01.03.02 Прикладная математика и информатика

шифр, наименование направления подготовки/специальности

Вид практики преддипломная

учебная, производственная, преддипломная или другой вид практики

Оценка за практику _____ Крылов С.С.

Москва

2025

1. **Место и сроки проведения практики:**

Наименование организации: __Кафедра 806_____

Сроки проведения практики

дата начала практики: __10.02.2025_____

дата окончания практики: __10.05.2025_____

2. **Инструктаж по технике безопасности:**

_____ /_____/ ___10 _февраля___ 2025г.

подпись проводившего *расшифровка подписи* *дата проведения*

3. **Индивидуальное задание обучающегося:**

Необходимо спроектировать систему, обеспечивающую перенос взаимосвязанных данных между из производственной среды в тестовую с реализацией следующих свойств: анонимизация данных при переносе, а также генерацию тестовых данных. В рамках работы предполагается разработка требований и реализация прототипа данной системы с последующим проведением анализа её жизнеспособности.

4. **План выполнения индивидуального задания обучающегося:**

№ п/п	Место проведения	Тема	Период выполнения
1	Кафедра 806	Инструктаж.	10.02.2025-10.02.2025
2	Кафедра 806	Определение требований к проектируемой системе.	11.02.2025-15.02.2025
3	Кафедра 806	Анализ и исследование существующих аналогов.	16.02.2025-25.02.2025
4	Кафедра 806	Проектирование архитектуры системы.	26.02.2025-11.03.2025
5	Кафедра 806	Разработка языка описания данных и алгоритма переноса взаимосвязанных данных.	12.03.2025-15.04.2025
6	Кафедра 806	Реализация прототипа системы.	16.04.2025-05.05.2025
7	Кафедра 806	Анализ полученных результатов.	06.05.2025-07.05.2025
8	Кафедра 806	Оформление отчета. Подведение итогов.	08.05.2025-10.05.2025

Утверждаю

_____ /_____/ ___10 _февраля___ 2025г.

подпись руководителя от МАИ *расшифровка подписи* *дата утверждения*

_____ /____Миронов Е. С.____/ ___10 _февраля___ 2025г.

подпись руководителя от организации/предприятия *расшифровка подписи* *дата утверждения*

Ознакомлен

_____ /____Мезенин О. А.___/ ___10 _февраля___ 2025г.

подпись обучающегося *расшифровка подписи* *дата ознакомления*

5. **Отзыв руководителя практики от организации/предприятия:**

Практика Мезенина Олега Александровича, студента группы М8О-406Б-21, проходила на кафедре 806. В течение преддипломной практики студент выполнял проектирование системы для переноса взаимосвязанных данных из производственной среды в тестовую. В его задачи входило обеспечение анонимизации данных и генерация тестовых данных. Студент провел тщательный анализ требований, разработал архитектуру системы, а также создал прототип, соответствующий поставленным задачам. В ходе работы была проведена оценка жизнеспособности системы, которая подтвердила её эффективность. Студент проявил отличные личные и деловые качества. Он ответственно подходил к поставленным задачам, демонстрировал инициативу и внимательность к деталям. В его работе чувствовалась организованность и умение эффективно управлять временем. Во время практики студент продемонстрировал глубокие знания в области системного анализа и проектирования программного обеспечения. Он уверенно использовал современные инструменты и технологии для разработки программных систем. Материалы, изложенные в отчёте обучающегося, полностью соответствуют индивидуальному заданию.

_____ /___Миронов Е. С.___/ _____ _____ 2025г.

подпись руководителя от организации/предприятии | *расшифровка подписи* | *дата*

6. Отчет обучающего по практике:

Требования к системе можно классифицировать на две основные категории: функциональные и архитектурные. Функциональные требования описывают конкретный набор функций и возможностей, которые система должна обеспечивать для удовлетворения потребностей пользователей. Требования к свойствам архитектуры системы фокусируются на качественных характеристиках системы, которые обеспечивают её устойчивую и эффективную работу.

Система должна поддерживать следующую функциональность:

1) перенос и анонимизация данных. Система должна обеспечивать возможность переноса взаимосвязанных данных между базами данных, а также их анонимизацию. Правила выбора взаимосвязанных данных и правила анонимизации задаются метаданными;
2) генерация данных. Система должна предоставлять функционал для генерации данных. Правила генерации задаются метаданными;
3) работа с метаданными. Система должна осуществлять проверку корректности метаданных, а также обеспечивать функциональность для загрузки предварительно подготовленных метаданных или их интеграции на этапе ввода данных;
4) гибкость в выборе базы данных. Необходимо внедрить функциональность, позволяющую пользователю выбрать базу данных, указав как уникальный идентификатор базы, так и строку подключения;

Архитектура системы должна соответствовать следующим критериям:

1) безопасность. Архитектура должна обеспечивать защиту от несанкционированного доступа к базе данных, а также обеспечивать защиту конфиденциальной информации;
2) надёжность. Система должна функционировать без сбоев в течение продолжительных периодов времени. Это предполагает наличие мер по обеспечению отказоустойчивости и возможности восстановления после сбоев;
3) высокая производительность. Архитектура должна обеспечивать эффективную обработку больших объёмов данных;
4) асинхронная обработка запросов. Поскольку перенос или генерация данных может занимать значительное время, система должна поддерживать выполнение запросов в асинхронном режиме. Это означает, что клиентские приложения не должны блокироваться в ожидании завершения операции, но должны иметь возможность проверять статус выполнения запроса по мере необходимости.

На уровне контекста можно наблюдать взаимодействие различных систем, а также взаимодействие этих систем с пользователями и специалистами. Схема взаимодействия представлена на рисунке 1.

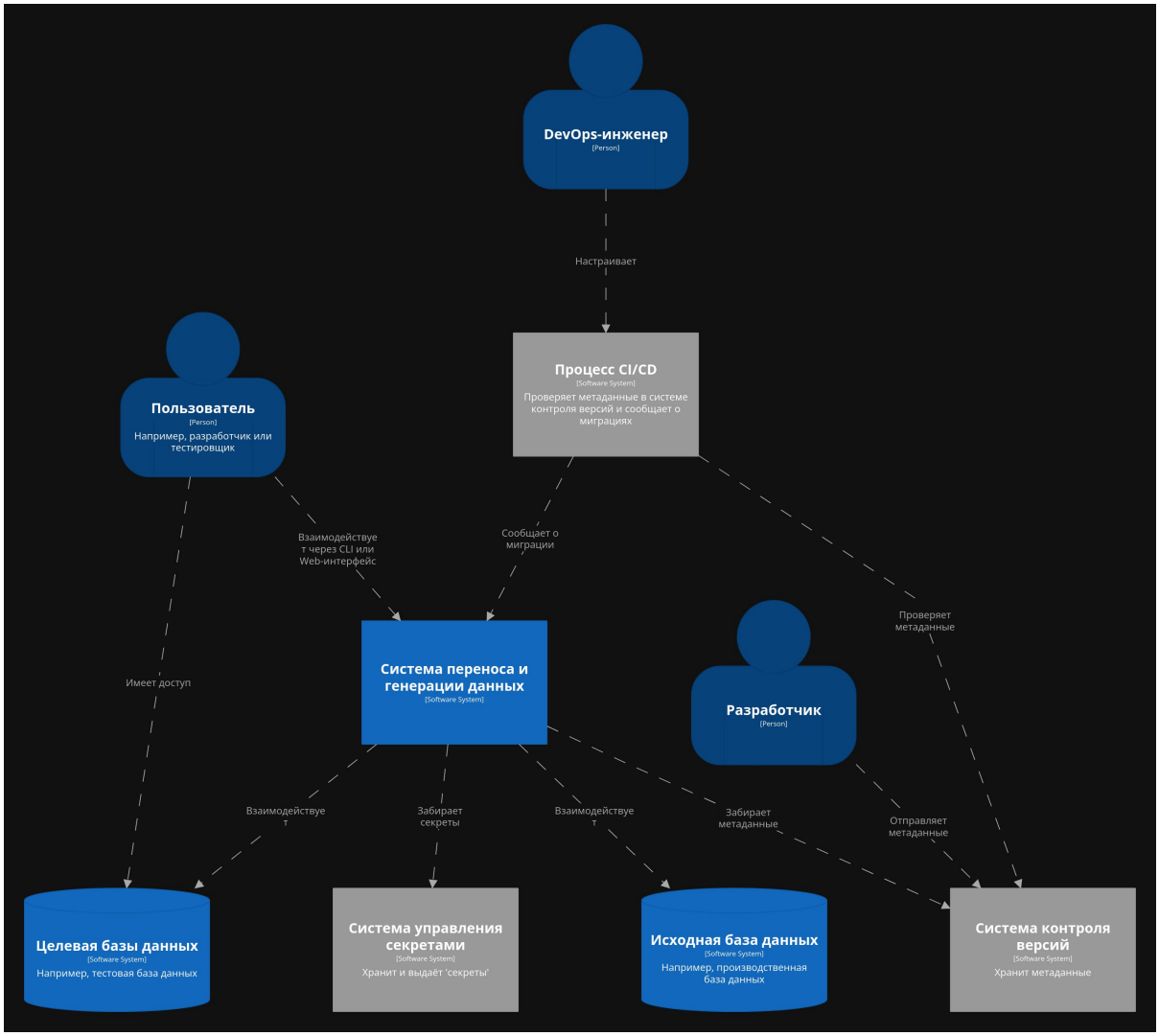

Рисунок 1 — Взаимодействие систем

Рассмотрим систему переноса и генерации данных на уровне контейнеров. Схема контейнеров представлена на рисунке 2.

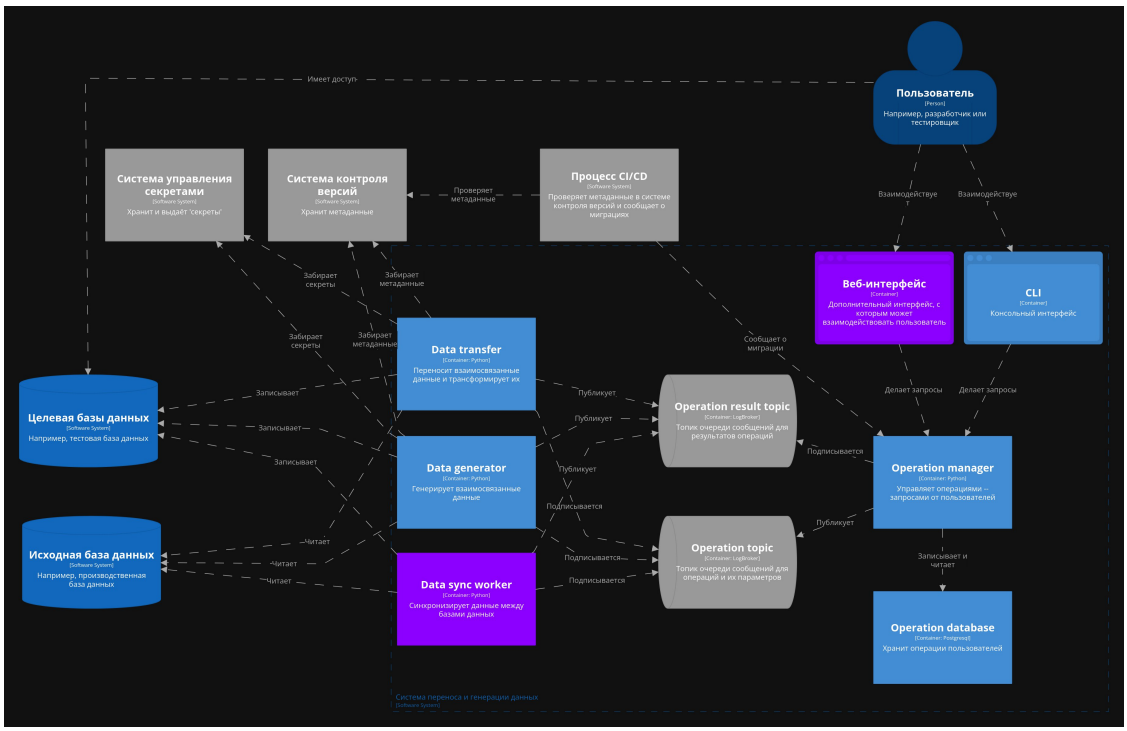

Рисунок 2 – Система переноса и генерации данных

Взаимодействие пользователя с системой осуществляется через командную строку (CLI), хотя может быть реализован и альтернативный интерфейс взаимодействия, например, веб-интерфейс (выделен фиолетовым как контейнер, который может быть добавлен в перспективе).

Пользователь посылает запросы в контейнер Operation Manager. Возможны два типа запросов: запуск операции по переносу или генерации данных, а также проверка статуса выполняемой операции. В случае получения запроса на запуск операции, информация об операции сохраняется в базе данных, а уникальный идентификатор операции возвращается пользователю, что позволяет ему отслеживать статус выполнения.

Далее Operation Manager отправляет запросы на выполнение операции в очередь сообщений Logbroker. Контейнер, обрабатывающий такой запрос, определяется типом операции: если речь идет о переносе данных, операция обрабатывается контейнером Data Transfer; если о генерации данных — контейнером Data Generator.

На схеме также представлен контейнер Data Sync Worker, являющийся гипотетическим контейнером, предназначенным для обработки операций по синхронизации данных в базах данных.

Контейнеры Data Transfer и Data Generator осуществляют перенос и генерацию данных, взаимодействуя с базами данных, системой контроля версий и системой управления секретами. В процессе выполнения операции, а также после её выполнения, информация о статусе возвращается в очередь сообщений, из которой Operation Manager извлекает данные и обновляет статус операции в базе данных.

Опишем алгоритм, который принимает на вход метаграф базы данных,

множество начальных вершин, а также два множества правил метаграфа: одно для метарёбер, другое для рёбер.

На выходе алгоритм возвращает множество вершин, связанных с начальными вершинами, включая сами начальные вершины. В процессе работы алгоритм применяет правила метаграфа. Алгоритм представлен на рисунке 3.

```
1  Metagraph_Traversal_With_Rules(DB, SV, RME, RE)
2    ▷ Инициализация собственного метаграфа MG: копирование
     метавершин и метарёбер из метаграфа базы данных
3    MG ← <∅, ∅, ∅, ∅>
4    MG.MV ← Copy(DB.MV)
5    MV.ME ← Copy(DB.ME)
6    ▷ Применение правил RME к метарёбрам
7    MG.ME ← Apply_Rules(MG.ME, RME)
8    queue ← ∅
9    visited ← ∅
10   for each v in SV
11     do Enqueue(queue, v)
12   while queue ≠ ∅
13     do cur_v ← Dequeue(queue)
14        Add(visited, cur_v)
15        ▷ Дополнение метаграфа MG текущей вершиной
16        Add(MG.V, cur_v)
17        ▷ Получение инцидентных к вершине cur_v рёбер
18        incident_edges ← Incident(DB, cur_v)
19        ▷ Применение правил RE к полученным рёбрам
20        new_incident_edges ← Apply_Rules(incident_edges, RE)
21        ▷ Дополнение метаграфа MG новым множеством рёбер
22        Extend(MG.E, new_incedent_edges)
23        ▷ Проход по всем вершинам, смежными с вершиной cur_v,
     обновлённого метаграфа MG
24        for each u in Adjacent(MG, cur_v)
25          do if u ∉ visited
26             then Enqueue(queue, u)
27   return visited
28
```

Рисунок 3 — Алгоритм поиска взаимосвязанных данных

Рассмотрим разработку грамматики языка и реализацию синтаксического анализатора с использованием инструмента ANTLR4. В качестве основы для построения грамматики была выбрана грамматика языка SQLite.

На рисунке 4 представлены основные конструкции в формате грамматики ANTLR4.

```
1  graph_source_stmt: GRAPH_ SOURCE_ table_name (WHERE_ expr)?;
2  include_edge_stmt: INCLUDE_ EDGE_ table_name DOT column_name
     table_name DOT column_name;
3  exclude_edge_stmt: EXCLUDE_ EDGE_ table_name DOT column_name
     table_name DOT column_name;
4  no_enter_stmt: NO_ ENTER_ table_name (WHERE_ expr)?;
5  no_exit_stmt: NO_ EXIT_ table_name (WHERE_ expr)?;
6  limit_visits_stmt: LIMIT_ VISITS_ INTEGER_LITERAL FOR_
     table_name;
7  limit_distance_stmt: LIMIT_ DISTANCE_ INTEGER_LITERAL FOR_
     table_name;
8  transformer_stmt: TRANSFORMER_ function_call FOR_ table_name
     DOT column_name (COMMA table_name DOT column_name)*;
9  set_generation_values_stmt: SET_ GENERATION_ VALUES_ (
     set_of_values | range_of_values | function_call) FOR_
     table_name DOT column_name (COMMA table_name DOT
     column_name)*;
10 set_generation_amount_stmt: SET_ GENERATION_ AMOUNT_
     table_name ASSIGN INTEGER_LITERAL (COMMA table_name ASSIGN
      INTEGER_LITERAL)*;
11
```

Рисунок 4 — Описание основных конструкций в формате грамматики ANTLR4

В рамках работы был разработан прототип системы в виде инструмента с интерфейсом командной строки. В прототипе предусмотрены две версии взаимодействия с базой данных и между компонентами Graph Walker и Data Writer: синхронная и асинхронная. В случае синхронного взаимодействия компонент Graph Walker ожидает завершения записи данных компонентом Data Writer, в то время как при асинхронном взаимодействии компоненты функционируют независимо друг от друга.

Проведём оценку производительности. В тестах рассматриваются прототипы синхронной и асинхронной версий, а также программа, осуществляющая экспорт всех данных из исходной базы с помощью утилиты pg_dump и их вставку в целевую базу посредством psql.

На рисунке 5 и продемонстрировано время выполнения разных программ для разных тестовых данных. Можно отметить, что время выполнения программы, использующей pg_dump, увеличивается незначительно. Также, начиная с теста, содержащего 770 строк данных, асинхронная версия прототипа демонстрирует более высокую скорость выполнения по сравнению с синхронной версией.

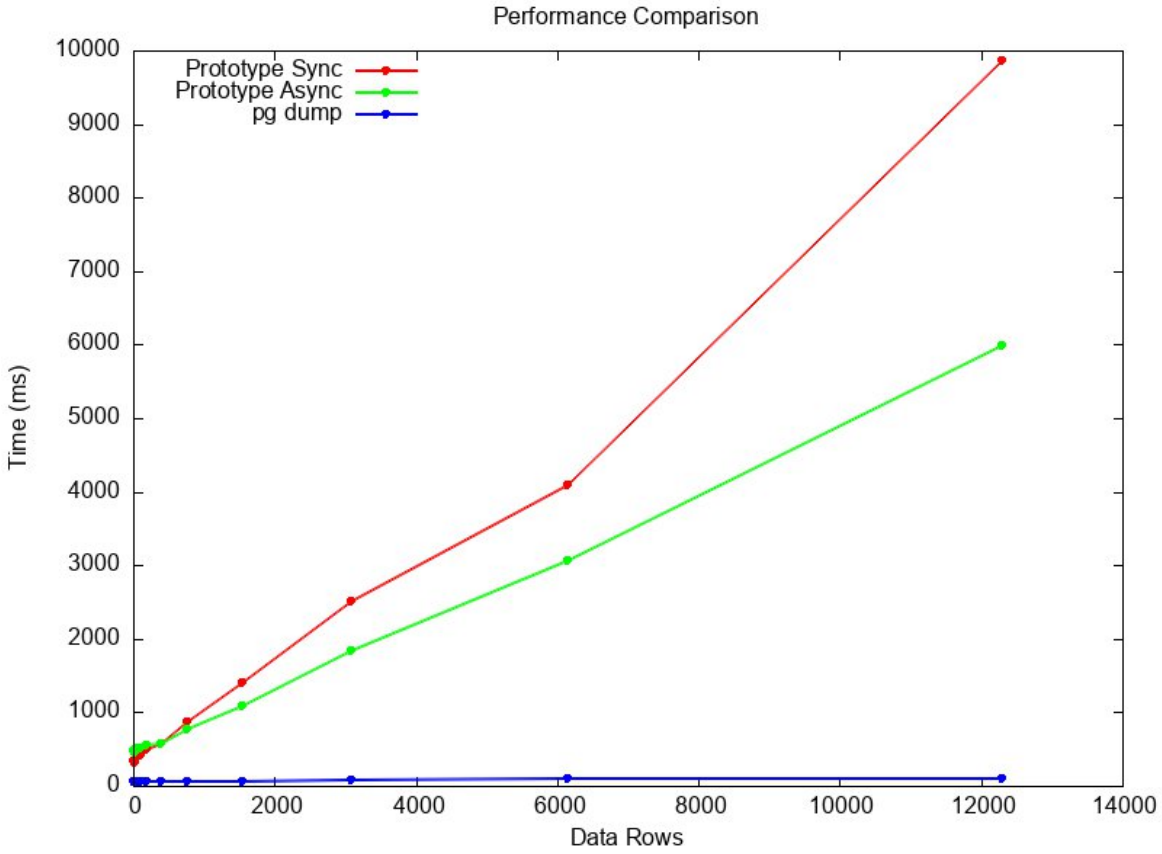

Рисунок 5 — График производительности

Проведём анализ жизнеспособности системы на основе спроектированной системы, разработанных алгоритма, языка и прототипа, а также результатов производительности.

Стоит отметить, что готового решения, удовлетворяющего всем описанным ранее требованиям, в виде единного программного продукта найдено не было. Несмотря на это, задачи переноса взаимосвязанных данных, генерации данных и обфускации данных можно решить с помощью отдельных готовых инструментов.

Например, с помощью инструмента Jailer можно получить взаимосвязанные данные. Тем не менее, он обладает только графическим интерфейсом, что усложняет его интеграцию с системами автоматизации. Кроме того, непосредственное подключение к производственной базе конечным пользователем представляет собой значительную угрозу безопасности данных, требующую устранения.

Для анонимизации данных можно использовать инструмент Triki. Но его нельзя использовать в процессе переноса данных: сначала данные нужно перенести, а только потом анонимизировать. В связи с этим, необходимо обеспечение непрерывности процесса от переноса до анонимизации данных, чтобы пользователи целевой базы не смогли получить доступ к чувствительным данным.

На основании изложенного можно заключить, что сборка рассматриваемой системы из готовых компонентов возможна, но потребует

значительных доработок, соизмеримых по трудозатратам с разработкой новой системы.

Разработанным алгоритму выбора взаимосвязанных данных и языку можно сопоставить sql-запрос с применением CTE, который будет выбирать взаимосвязанные данные. Но такой запрос будет большим и трудноподдерживаемым.

Особенность разработанных алгоритма и языка состоит в том, что человек, описывающий метаданные, должен хорошо знать структуру базы данных, либо пользоваться инструментами для анализа связей данных, такими как Jailer. В противном случае существует риск неэффективности процесса переноса данных: либо данных будет недостаточно, либо будет их избыток.

Результаты тестов производительности свидетельствуют о том, что при необходимости переноса всех или большинства данных использование рассматриваемой системы нецелесообразно: в таких случаях предпочтительнее применение pg_dump.

Отметим также, что асинхронная версия прототипа демонстрирует лучшие результаты, что делает использование синхронной версии необязательным.

Подводя итоги, можно сделать вывод о жизнеспособности системы, однако её применение требует осторожности: необходимо понимание структуры базы данных и объёма данных, требуемых для тестирования.

_____	/___Мезенин О. А.___/	10 мая 2025 г.
подпись обучающегося	*расшифровка подписи*	*дата*