МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский государственный университет аэрокосмического приборостроения»

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

КУРСОВОЙ ПРОЕКТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Преподаватель |  | 05.07.2023 |  | И.Л. Рохманько |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЁТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ  В СОСТАВЕ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ  ПМ.04 Разработка, администрирование и защита баз данных  *код и наименование профессионального модуля* |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | С021к |  | 05.07.2023 |  | В.Д. Панков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на прохождение производственной практики обучающегося по специальности 09.02.07 «Информационные системы и программирование»

1. Фамилия, имя, отчество обучающегося: Панков Василий Дмитриевич
2. Группа: С021К Сроки проведения практики: с « 15 » июня 2023г. по « 05 » июля 2023г.
3. Тема задания: приобретение профессиональных умений и опыта практической деятельности, закрепление сформированных компетенций по профессиональному модулю

ПМ.04 РАЗРАБОТКА, АДМИНИСТРИРОВАНИЕ И ЗАЩИТА БАЗ ДАННЫХ

1. Вопросы, подлежащие изучению:

* Обследование предметной области . Проектирование логической схемы базы данных с использованием современных case-средств;
* Обоснование выбора СУБД. Построение схемы базы данных в конкретной СУБД. Создание хранимых процедур и триггеров;
* Импорт данных в базу данных;
* Администрирование БД;
* Обеспечение информационной безопасности на уровне базы данных;
* Разработка пользовательского приложения для работы с базой данных;
* Установка базы данных на сервер.

1. Выполнение комплексных работ: Разработка, администрирование и защита базы данных учёта коллекционных записей экспедиционных сборов млекопитающих
2. Содержание отчетной документации:
   1. Отчёт, включающий в себя:

* титульный лист;
* индивидуальное задание;
* материалы о выполнении индивидуального задания (содержание определяется цикловой комиссией);
* список использованных источников.
  1. Аттестационный лист.

1. Срок представления отчета заместителю декана по учебно–производственной работе: «\_05\_\_»\_\_\_июля\_\_\_\_\_2022\_\_ г.

Руководитель практики от факультета СПО:

преподаватель 15.06.23 И.Л. Рохманько

должность, уч. степень, звание подпись, дата инициалы, фамилия

СОГЛАСОВАНО

Руководитель практики от организации:

заведующей лабораторией, ведущий научный сотрудник 15.05.23 Н.И. Абрамсон

должность подпись, дата инициалы, фамилия

Задание принял к исполнению:

Обучающийся

15.06.23 В.Д. Панков

дата подпись инициалы, фамилия

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc138071182)

[1 Анализ предметной области поставленной задачи 5](#_Toc138071183)

[1.1 Общая характеристика объекта исследования 5](#_Toc138071184)

[1.2 Постановка и развернутое описание задачи 6](#_Toc138071185)

[1.3 Средства решения поставленной задачи 7](#_Toc138071186)

[2 Разработка и реализация проекта базы данных 12](#_Toc138071187)

[2.1 Определение логической (даталогической) структуры реляционной базы данных 12](#_Toc138071188)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 16](#_Toc138071189)

[ПРИЛОЖЕНИЕ А Созданные таблицы БД 18](#_Toc138071190)

# ВВЕДЕНИЕ

Информационная система — это совокупность программных и аппаратных средств, предназначенных для сбора, хранения, обработки, анализа и распространения информации в организации или предприятии.

Существует множество лабораторией во всём мире, которые занимаются различными исследованиями с животными. Данную производственную практику я проходил в «Лаборатории геномики и палеогеномики» при «Государственном зоологическом институте Российской Федерации». Данная лаборатория занимается изучением геномов существующих и исчезнувших видов млекопитающих.

Целью моей работы является проектирование информационной системы, которая облегчит процесс учета коллекционных записей экспедиционных сборов млекопитающих, а также обеспечит возможность анализа полученной информации для научных исследований в данной области.

Для достижения поставленной цели необходимо решить следующие задачи:

* изучить методы и технологии учета и хранения информации о коллекционных записях экспедиционных сборов млекопитающих;
* определить функциональные требования к информационной системе;
* разработать структуру базы данных для хранения информации о коллекционных записях;
* разработать интерфейс для ввода и обработки данных;
* реализовать информационную систему на основе выбранных методов и технологий;

# Анализ предметной области

## Общая характеристика объекта исследования

В данной задаче практики предметной областью является учёт коллекционных записей экспедиционных сборов млекопитающих, а в качестве внешних пользователей могут выступать:

* заведующий лабораторией – лицо, которое может редактировать информацию о сотрудниках, редактировать топологию, а также выдавать доступ новым сотрудникам и менять им пароли;
* научный сотрудник – лицо, будет иметь доступ к просмотру всех коллекционных записей, имеет возможности редактирование и добавление записей и отправку запросов на их удаление, также пользователь может сменить свой пароль.
* гость – группа, пользователей, которая может только просматривать данные.

Научные экспедиции по изучению млекопитающих проходят в различных уголках мира, и собранные в ходе них образцы требуют качественного учёта и обработки. В информационной системе будут храниться данные о месте, времени сбора, полного названия млекопитающего, которое будет включать отряд, семейство, род и вид, его генетические характеристики.

Также некоторые экземпляры могут передавать в другие лаборатории или музеи, для их показа или исследования и иметь там свой уникальный номер.

В самой лаборатории используются собственная идентификация элементов, имеются комментарии к месту сбора и к самой записи.

## Постановка и развернутое описание задачи

Информационная система должна обеспечивать следующие функции:

* ввод, просмотр редактирование информацию о записи, авторе сборного материала, дате сбора, месте сбора, отряде, семействе, роде виде, полу, возрасту и выделена ли RNA;
* выполнять запросы для автоматизации (просмотр по месту сбора, фильтрация по множеству параметров: по автору сборного материала, промежутке сбора, месту сбора, по таксономии, полу, возрасту и выделена ли RNA;);
* возможность хранение ссылок на файлы c полногеномными данными;
* обеспечение быстрого добавления записей (указание точки на карте, выбор сборщика из списка, генерация идентификатора);
* авторизация пользователей и сохранение информации об внесённых ими изменений;
* возможность редактирования древовидной структуры таксономии.

Входные данные для проектирования информационной системы:

* информация о сборе: место сбора, сборщик материала, дата сбора, RNA, внутренний номер в лаборатории, отряд, семейство, род, вид и пол млекопитающего данных сбора.

Выходные данные: в качестве выходных данных будут результаты выполнения запросов.

Требования, предъявляемые к разрабатываемой программе:

Программа должна быть надёжна, простота в использовании и обладать достаточно высоким быстродействием.

Пользовательский интерфейс должен быть интуитивно понятен любому пользователю. В программе должна быть предусмотрена возможность корректировки базы данных.

## Средства решения

Для решения поставленной задачи были выбраны следующие средства:

Dart – объектно-ориентированный язык программированный, разработанный Google. Используется для создания быстрых и эффективных приложений на разных платформах. Одной из важных особенностей данного языка является его способность к компиляции в эффективный нативный код. Это позволяет создавать быстрые и производительные приложения, которые могут выполняться без промежуточной интерпретации. [3]

Material You – это эволюция концепции Material Design, разработанная Google. Он представляет собой дизайн-язык, который ставит на передний план персонализацию и индивидуальность пользователей.

Основные преимущества Material You:

1. персонализация: Material You позволяет пользователям настраивать визуальный стиль своих устройств и приложений, включая цвета, шрифты и стили. Это создает более индивидуальный и удовлетворяющий опыт использования.
2. адаптивность: Material You обеспечивает адаптивный интерфейс, который подстраивается под различные размеры экранов и контексты использования. Приложения выглядят и работают хорошо на разных устройствах.
3. системные возможности: Material You интегрируется с функциями операционной системы и платформы, позволяя использовать новые возможности и API для создания более глубокого пользовательского опыта.
4. совместимость с Material Design: Material You соответствует принципам Material Design, обеспечивая согласованный визуальный стиль и поведение приложений на разных устройствах и платформах.
5. поддержка разработчиков: Material You получил широкую поддержку разработчиков и сообщества. Google предоставляет ресурсы, инструменты и руководства для помощи разработчикам внедрить Material You в свои приложения. [7]

Flutter – это открытый кросс-платформенный фреймворк разработки мобильных приложений, созданный Google. Он позволяет создавать высокопроизводительные приложения для iOS, Android, веб и настольных компьютеров с использованием общего кода на языке программирования Dart. Flutter отличается своим собственным движком рендеринга, обеспечивающим быструю и плавную визуализацию. Он предлагает готовые виджеты и инструменты для удобной разработки пользовательского интерфейса. Благодаря функции горячей перезагрузки, изменения в коде отображаются мгновенно, что ускоряет разработку и упрощает тестирование приложений. Flutter также поддерживает пользовательские анимации, интеграцию с нативными функциями устройства и управление состоянием приложения.

Основные принципы Flutter:

* кроссплатформенность: Flutter позволяет разрабатывать приложения для разных платформ, таких как iOS, Android, веб и настольные компьютеры, с использованием общего кода. Это позволяет сократить время разработки и упростить поддержку приложений на разных платформах;
* горячая перезагрузка: Горячая перезагрузка (Hot Reload) в Flutter позволяет мгновенно видеть изменения в коде и мгновенно обновлять интерфейс приложения без перезапуска. Это значительно ускоряет процесс разработки и позволяет разработчикам быстро итерироваться над интерфейсом и функциональностью приложения;
* однородность интерфейса: Все элементы интерфейса в Flutter являются виджетами, включая даже текст, отступы и цвета. Это создает единообразие во всем интерфейсе приложения и облегчает его масштабирование, адаптацию и изменение визуального стиля;
* производительность: Flutter обеспечивает высокую производительность и плавность работы приложений благодаря использованию собственного движка рендеринга. Он позволяет создавать быстрые и отзывчивые пользовательские интерфейсы, а также оптимизирует использование ресурсов устройства;
* открытость и активное сообщество: Flutter является открытым проектом, и у него активное сообщество разработчиков. Это обеспечивает доступ к большому количеству инструментов, библиотек и пакетов, которые расширяют возможности разработки и помогают решать разнообразные задачи;
* Material You дизайн: Flutter тесно интегрирован с Material You дизайном, стандартным дизайн-языком Google. Это позволяет разработчикам создавать приложения с современным и стильным внешним видом, соответствующим рекомендациям Google по пользовательскому опыту. [4]

Syncfusion DataGrid for Flutter – гибкий компонент пользовательского интерфейса для отображения и редактирования табличных данных в приложениях Flutter. [5]

PostgreSQL– это мощная, открытая реляционная система управления базами данных. PostgreSQL предлагает расширенный набор функций и возможностей, обеспечивая надежное хранение и эффективное управление данными.

Основные характеристики PostgreSQL:

* реляционная модель данных: PostgreSQL основан на реляционной модели данных, что позволяет организовывать данные в таблицы с определенными структурами и связями. Это обеспечивает структурированное хранение информации и возможность выполнения сложных запросов и аналитических операций;
* масштабируемость и производительность: PostgreSQL способен обрабатывать большие объемы данных и обеспечивать высокую производительность при выполнении запросов. Он поддерживает параллельную обработку, индексы, оптимизацию запросов и другие механизмы для оптимизации производительности;
* поддержка стандартов SQL: PostgreSQL полностью совместим с ANSI SQL и поддерживает широкий набор стандартных операторов, функций и запросов. Это облегчает разработку и переносимость приложений, использующих PostgreSQL;
* расширяемость и гибкость: PostgreSQL предлагает механизмы для создания пользовательских типов данных, функций и расширений. Это позволяет разработчикам расширять функциональность СУБД и адаптировать ее под уникальные требования проекта;
* транзакционная безопасность: PostgreSQL обеспечивает транзакционную безопасность, поддерживая ACID-свойства (атомарность, согласованность, изолированность, долговечность). Это гарантирует целостность данных и защиту от потери информации в случае сбоев или ошибок;
* многоязыковая поддержка: PostgreSQL поддерживает множество языков программирования, включая Python, Java, C/C++, Ruby и многие другие. Это облегчает интеграцию СУБД с различными приложениями и разработку на разных платформах; [2]

RESTful API (Representational State Transfer) – это архитектурный стиль для проектирования и взаимодействия сетевых приложений, основанный на принципах REST. Он обеспечивает стандартизированный способ обмена данными между клиентом и сервером с использованием HTTP-протокола.

PostgREST – это программное обеспечение с открытым исходным кодом, которое предоставляет простой и удобный RESTful интерфейс для работы с базой данных PostgreSQL. Он позволяет вам взаимодействовать с базой данных через стандартные HTTP-методы, такие как GET, POST, PUT и DELETE, и автоматически генерирует API-эндпоинты для каждой таблицы в базе данных. PostgREST обеспечивает функции фильтрации, сортировки, пагинации и аутентификации данных, а также поддерживает вложенные запросы и пользовательские функции. Использование PostgREST значительно упрощает разработку и создание API на основе PostgreSQL, обеспечивая гибкость и масштабируемость веб-приложений и микросервисов. [2]

Nominatim (от латинского «by name») – это инструмент для поиска данных OSM по имени и адресу и для создания синтетических адресов точек OSM (обратное геокодирование). [9]

# Разработка и реализация проекта базы данных

## Определение логической (даталогической) структуры реляционной базы данных

Даталогическая структура реляционной базы данных определяется совокупностью логически связанных реляционных таблиц.

Логические связи соответствуют структурным связям между объектами в концептуальной модели, каждый объект в логической модели отображается соответствующей реляционной таблицей.

Связи между таблицами осуществляются посредством общих атрибутов. Реляционная модель проектируемой базы данных.

Таблица 1 содержит информацию о возрастах.

Таблица 1 – Описание логической структуры таблицы «age»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| - | name | Символьный (переменный) | 20 | - |

Таблица 2 содержит информацию о возрастах.

Таблица 2 – Описание логической структуры таблицы «collection»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| PK | id | Числовой | - | NOT NULL |
| - | CatalogueNumber | Текстовый | - | - |
| - | collect\_id | Текстовый | - | - |
| FK | kind\_id | Числовой | - | NOT NULL |
| FK | subregion\_id | Числовой | - | NOT NULL |
| - | gen\_bank\_id | Символьный (переменный) | 20 | - |
| - | point | География (точка) | - | - |
| FK | vouch\_inst\_id | Числовой | - | - |
| - | vouch\_id | Символьный (переменный) | 20 | - |
| - | rna | Логический | - | DEFAULT false NOT NULL |
| FK | sex\_id | Числовой | - | - |
| FK | age\_id | Числовой | - | - |
| Продолжение таблица 2 | | | | |
| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| - | day | Числовой | - | - |
| - | month | Числовой | - | - |
| - | year | Числовой | - | NOT NULL |
| - | comment | Текстовый | - | - |
| - | geo\_comment | Текстовый | - | - |
| - | file\_url | Текстовый | - | - |

Таблица 3 содержит информацию о коллекторах.

Таблица 3 – Описание логической структуры таблицы «collector»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| - | last\_name | Символьный (переменный) | 100 | - |
| - | first\_name | Символьный (переменный) | 100 | - |
| - | second\_name | Символьный (переменный) | 100 | - |

Таблица 4, является смежной таблицей, которая содержит информацию о соотношении коллекторов к коллекции.

Таблица 4 – Описание логической структуры таблицы «collector\_to\_collection»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK, FK | collector\_id | Числовой | - | NOT NULL |
| PK, FK | collection\_id | Числовой | - | NOT NULL |

Таблица 5 содержит информацию о странах.

Таблица 5 – Описание логической структуры таблицы «country»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| - | name | Текстовый | - | NOT NULL |

Таблица 6 содержит информацию о регионах.

Таблица 6 – Описание логической структуры таблицы «region»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| FK | country\_id | Числовой | - | NOT NULL |
| - | name | Текстовый | - |  |

Таблица 7 содержит информацию о субрегионах.

Таблица 7 – Описание логической структуры таблицы «subregion»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| PK | id | Числовой | - | NOT NULL |
| FK | region\_id | Числовой | - | NOT NULL |
| - | name | Текстовый | - |  |

Таблица 8 содержит информацию об отрядах.

Таблица 8 – Описание логической структуры таблицы «order»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| - | name | Текстовый | - |  |

Таблица 9 содержит информацию о семействах.

Таблица 9 – Описание логической структуры таблицы «family»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| FK | order\_id | Числовой | - | NOT NULL |
| - | name | Текстовый | - |  |

Таблица 10 содержит информацию о родах.

Таблица 10 – Описание логической структуры таблицы «genus»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| FK | family\_id | Числовой | - | NOT NULL |
| - | name | Текстовый | - |  |

Таблица 11 содержит информацию о видах.

Таблица 11 – Описание логической структуры таблицы «kind»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| FK | genus\_id | Числовой | - | NOT NULL |
| - | name | Текстовый | - |  |

Таблица 12 содержит информацию о полах.

Таблица 12 – Описание логической структуры таблицы «sex»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| - | name | Символьный (переменный) | 40 | NOT NULL |

Таблица 13 содержит информацию о ваучерных институтах.

Таблица 13 – Описание логической структуры таблицы «voucher\_institute»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| - | name | Текстовый | - | NOT NULL |

На рисунке 1 показана схема связей данных.

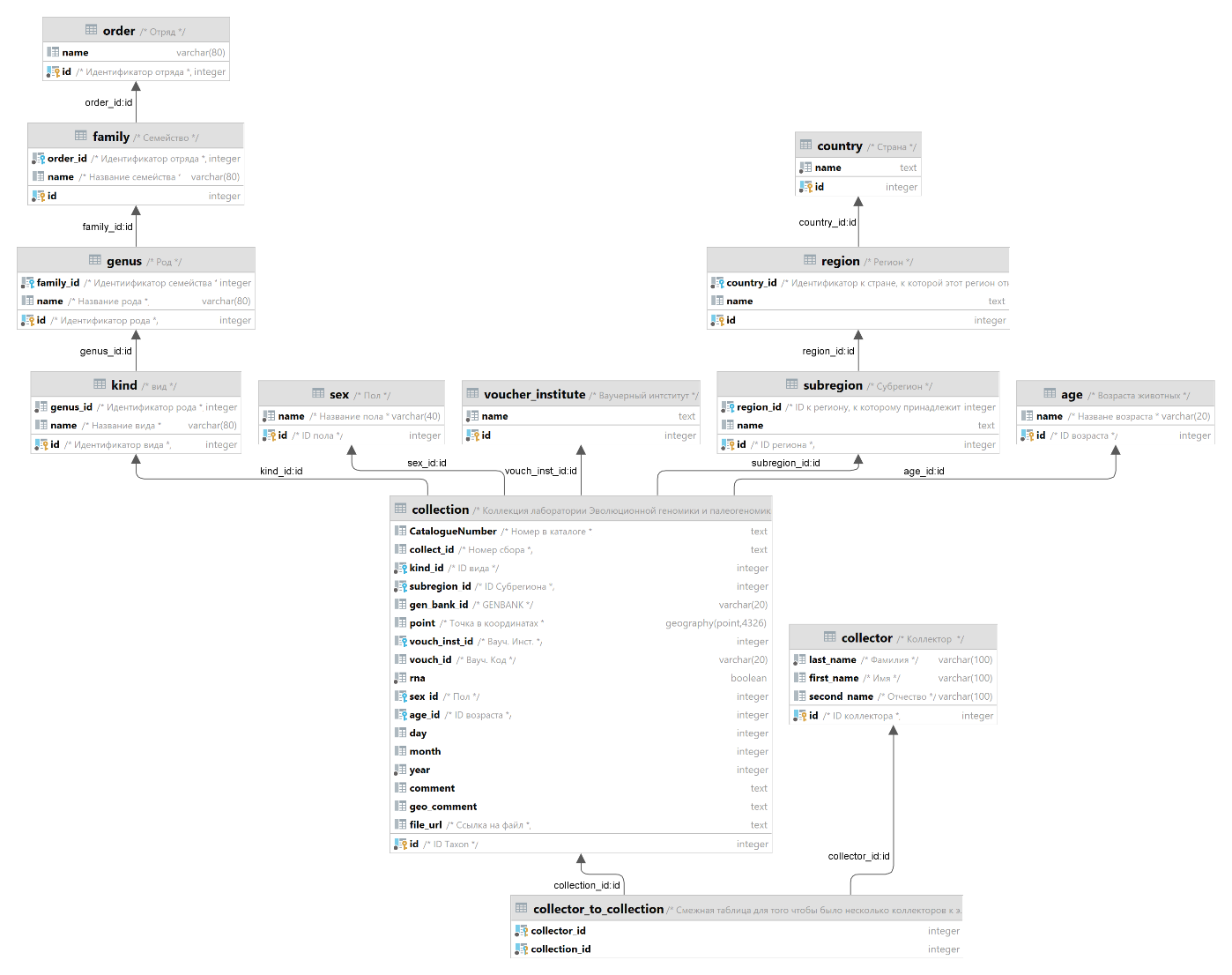


Рисунок 1 – Схема связей данных в СУБД PostgreSQL

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Документация расширения pgcrypto для шифрования PostgreSQL – URL: https://www.postgresql.org/docs/current/pgcrypto.html (дата обращения 01.06.2023).

2 Документация программы для создания restful API на основе базы данных PostgreSQL – URL: https://postgrest.org/en/stable/ (дата обращения 01.06.2023).

3 Документация языка программирования Dart – URL: https://dart.dev/guides (дата обращения 01.06.2023).

4 Документация библиотеки для разработки кроссплатформенных приложений Flutter – URL: https://docs.flutter.dev (дата обращения 01.06.2023).

5 Документация библиотеки для Flutter для отображения таблиц Syncfusion DataGrid – URL: https://help.syncfusion.com/flutter/datagrid/getting-started (дата обращения 01.06.2023).

6 Расширение для создание jwt токенов в PostgrSQL – URL: https://github.com/michelp/pgjwt (дата обращения 01.06.2023).

7 Дизайн система от Google Material 3 – URL: https://m3.material.io (дата обращения 01.06.2023).

8 Библиотека для организации http запросов в языке программировании Dart – URL: https://pub.dev/packages/http (дата обращения 01.06.2023).

9 Nominatim API – API для поиска мест по адресу или по координатам – URL: https://nominatim.org/release-docs/latest/api/Overview/ (дата обращения 01.06.2023).

10 Библиотека для работы с картами – Flutter Map – URL: https://pub.dev/packages/flutter\_map (дата обращения 01.06.2023).

11 Классификация организмов 9 класс – URL: https://www.yaklass.ru/p/biologia/9-klass/osnovy-evoliutcionnogo-ucheniia-246743/poniatie-vida-printcipy-sistematiki-246744/re-32dcb240-dd92-4865-bee7-96de4ee12880#:~:text=%D0%91%D0%BB%D0%B8%D0%B7%D0%BA%D0%B8%D0%B5%20(%D1%80%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5)%20%D0%B2%D0%B8%D0%B4%D1%8B%20%D0%B6%D0%B8%D0%B2%D0%BE%D1%82%D0%BD%D1%8B%D1%85%20%D0%BE%D0%B1%D1%8A%D0%B5%D0%B4%D0%B8%D0%BD%D1%8F%D1%8E%D1%82%D1%81%D1%8F,%2C%20%D0%B0%20%D1%82%D0%B8%D0%BF%D1%8B%20%E2%80%94%20%D0%B2%20%D1%86%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%BE (дата обращения 01.06.2023).

# ПРИЛОЖЕНИЕ А Созданные таблицы БД

Код созданных таблиц:

CREATE TABLE auth.users (

login text NOT NULL,

pass text NOT NULL,

role name NOT NULL,

avatar text,

CONSTRAINT users\_pass\_check CHECK ((length(pass) < 512)),

CONSTRAINT users\_role\_check CHECK ((length((role)::text) < 512))

);

CREATE TABLE public.age (

id integer NOT NULL,

name character varying(20)

);

CREATE SEQUENCE public.age\_id\_seq

START WITH 0

INCREMENT BY 1

MINVALUE 0

NO MAXVALUE

CACHE 1;

CREATE TABLE public.collection (

id integer NOT NULL,

"CatalogueNumber" text,

collect\_id text,

kind\_id integer NOT NULL,

subregion\_id integer NOT NULL,

gen\_bank\_id character varying(20),

point public.geography(Point,4326),

vouch\_inst\_id integer,

vouch\_id character varying(20),

rna boolean DEFAULT false NOT NULL,

sex\_id integer,

age\_id integer,

day integer,

month integer,

year integer NOT NULL,

comment text,

geo\_comment text,

file\_url text,

CONSTRAINT genbank\_check CHECK ((((gen\_bank\_id)::text ~ 'OP\d+'::text) OR (gen\_bank\_id IS NULL)))

);

CREATE SEQUENCE public.collection\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.collector (

id integer NOT NULL,

last\_name character varying(100) NOT NULL,

first\_name character varying(100),

second\_name character varying(100)

);

CREATE SEQUENCE public.collector\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.collector\_to\_collection (

collector\_id integer NOT NULL,

collection\_id integer NOT NULL

);

CREATE TABLE public.country (

id integer NOT NULL,

name text NOT NULL

);

CREATE SEQUENCE public.country\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.family (

id integer NOT NULL,

order\_id integer NOT NULL,

name character varying(80)

);

CREATE SEQUENCE public.family\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.genus (

id integer NOT NULL,

family\_id integer NOT NULL,

name character varying(80)

);

CREATE SEQUENCE public.genus\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.kind (

id integer NOT NULL,

genus\_id integer NOT NULL,

name character varying(80)

);

CREATE SEQUENCE public.kind\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public."order" (

id integer NOT NULL,

name character varying(80)

);

CREATE SEQUENCE public.order\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.region (

id integer NOT NULL,

country\_id integer NOT NULL,

name text

);

CREATE SEQUENCE public.region\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.sex (

id integer NOT NULL,

name character varying(40) NOT NULL

);

CREATE SEQUENCE public.sex\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.subregion (

id integer NOT NULL,

region\_id integer NOT NULL,

name text

);

CREATE SEQUENCE public.subregion\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.voucher\_institute (

id integer NOT NULL,

name text NOT NULL

);

CREATE SEQUENCE public.voucher\_institute\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE ONLY public.age ALTER COLUMN id SET DEFAULT nextval('public.age\_id\_seq'::regclass);

ALTER TABLE ONLY public.collection ALTER COLUMN id SET DEFAULT nextval('public.collection\_id\_seq'::regclass);

ALTER TABLE ONLY public.collector ALTER COLUMN id SET DEFAULT nextval('public.collector\_id\_seq'::regclass);

--

-- Name: family id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.family ALTER COLUMN id SET DEFAULT nextval('public.family\_id\_seq'::regclass);

--

-- Name: genus id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.genus ALTER COLUMN id SET DEFAULT nextval('public.genus\_id\_seq'::regclass);

--

-- Name: kind id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.kind ALTER COLUMN id SET DEFAULT nextval('public.kind\_id\_seq'::regclass);

--

-- Name: order id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."order" ALTER COLUMN id SET DEFAULT nextval('public.order\_id\_seq'::regclass);

--

-- Name: region id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.region ALTER COLUMN id SET DEFAULT nextval('public.region\_id\_seq'::regclass);

--

-- Name: sex id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.sex ALTER COLUMN id SET DEFAULT nextval('public.sex\_id\_seq'::regclass);

--

-- Name: subregion id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.subregion ALTER COLUMN id SET DEFAULT nextval('public.subregion\_id\_seq'::regclass);

--

-- Name: voucher\_institute id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.voucher\_institute ALTER COLUMN id SET DEFAULT nextval('public.voucher\_institute\_id\_seq'::regclass);

--

-- Name: users users\_pkey; Type: CONSTRAINT; Schema: auth; Owner: postgres

--

ALTER TABLE ONLY auth.users

ADD CONSTRAINT users\_pkey PRIMARY KEY (login);

--

-- Name: age age\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.age

ADD CONSTRAINT age\_pk PRIMARY KEY (id);

--

-- Name: collection collection\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collection

ADD CONSTRAINT collection\_pk PRIMARY KEY (id);

--

-- Name: collector collector\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collector

ADD CONSTRAINT collector\_pk PRIMARY KEY (id);

--

-- Name: collector\_to\_collection collector\_to\_collection\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collector\_to\_collection

ADD CONSTRAINT collector\_to\_collection\_pk PRIMARY KEY (collector\_id, collection\_id);

--

-- Name: country country\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.country

ADD CONSTRAINT country\_pk PRIMARY KEY (id);

--

-- Name: family family\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.family

ADD CONSTRAINT family\_pk PRIMARY KEY (id);

--

-- Name: genus genus\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.genus

ADD CONSTRAINT genus\_pk PRIMARY KEY (id);

--

-- Name: kind kind\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.kind

ADD CONSTRAINT kind\_pk PRIMARY KEY (id);

--

-- Name: order order\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."order"

ADD CONSTRAINT order\_pk PRIMARY KEY (id);

--

-- Name: region region\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.region

ADD CONSTRAINT region\_pk PRIMARY KEY (id);

--

-- Name: sex sex\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.sex

ADD CONSTRAINT sex\_pk PRIMARY KEY (id);

--

-- Name: subregion subregion\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.subregion

ADD CONSTRAINT subregion\_pk PRIMARY KEY (id);

--

-- Name: voucher\_institute voucher\_institute\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.voucher\_institute

ADD CONSTRAINT voucher\_institute\_pk PRIMARY KEY (id);

--

-- Name: users encrypt\_pass; Type: TRIGGER; Schema: auth; Owner: postgres

--

CREATE TRIGGER encrypt\_pass BEFORE INSERT OR UPDATE ON auth.users FOR EACH ROW EXECUTE FUNCTION auth.encrypt\_pass();

--

-- Name: collection collection\_age\_id\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collection

ADD CONSTRAINT collection\_age\_id\_fk FOREIGN KEY (age\_id) REFERENCES public.age(id) ON UPDATE CASCADE ON DELETE RESTRICT;

--

-- Name: collection collection\_kind\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collection

ADD CONSTRAINT collection\_kind\_null\_fk FOREIGN KEY (kind\_id) REFERENCES public.kind(id);

--

-- Name: collection collection\_sex\_id\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collection

ADD CONSTRAINT collection\_sex\_id\_fk FOREIGN KEY (sex\_id) REFERENCES public.sex(id) ON UPDATE CASCADE ON DELETE RESTRICT;

--

-- Name: collection collection\_subregion\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collection

ADD CONSTRAINT collection\_subregion\_null\_fk FOREIGN KEY (subregion\_id) REFERENCES public.subregion(id);

--

-- Name: collection collection\_voucher\_institute\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collection

ADD CONSTRAINT collection\_voucher\_institute\_null\_fk FOREIGN KEY (vouch\_inst\_id) REFERENCES public.voucher\_institute(id);

--

-- Name: collector\_to\_collection collector\_to\_collection\_collection\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collector\_to\_collection

ADD CONSTRAINT collector\_to\_collection\_collection\_null\_fk FOREIGN KEY (collection\_id) REFERENCES public.collection(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- Name: collector\_to\_collection collector\_to\_collection\_collector\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collector\_to\_collection

ADD CONSTRAINT collector\_to\_collection\_collector\_null\_fk FOREIGN KEY (collector\_id) REFERENCES public.collector(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- Name: family family\_order\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.family

ADD CONSTRAINT family\_order\_null\_fk FOREIGN KEY (order\_id) REFERENCES public."order"(id);

--

-- Name: genus genus\_family\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.genus

ADD CONSTRAINT genus\_family\_null\_fk FOREIGN KEY (family\_id) REFERENCES public.family(id);

--

-- Name: region region\_country\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.region

ADD CONSTRAINT region\_country\_null\_fk FOREIGN KEY (country\_id) REFERENCES public.country(id);

--

-- Name: subregion subregion\_region\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.subregion

ADD CONSTRAINT subregion\_region\_null\_fk FOREIGN KEY (region\_id) REFERENCES public.region(id);

Данные в таблицах:

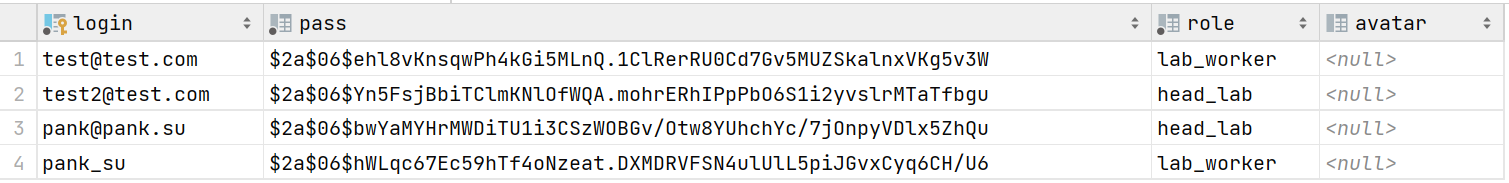


Рисунок A.1 – Таблица users

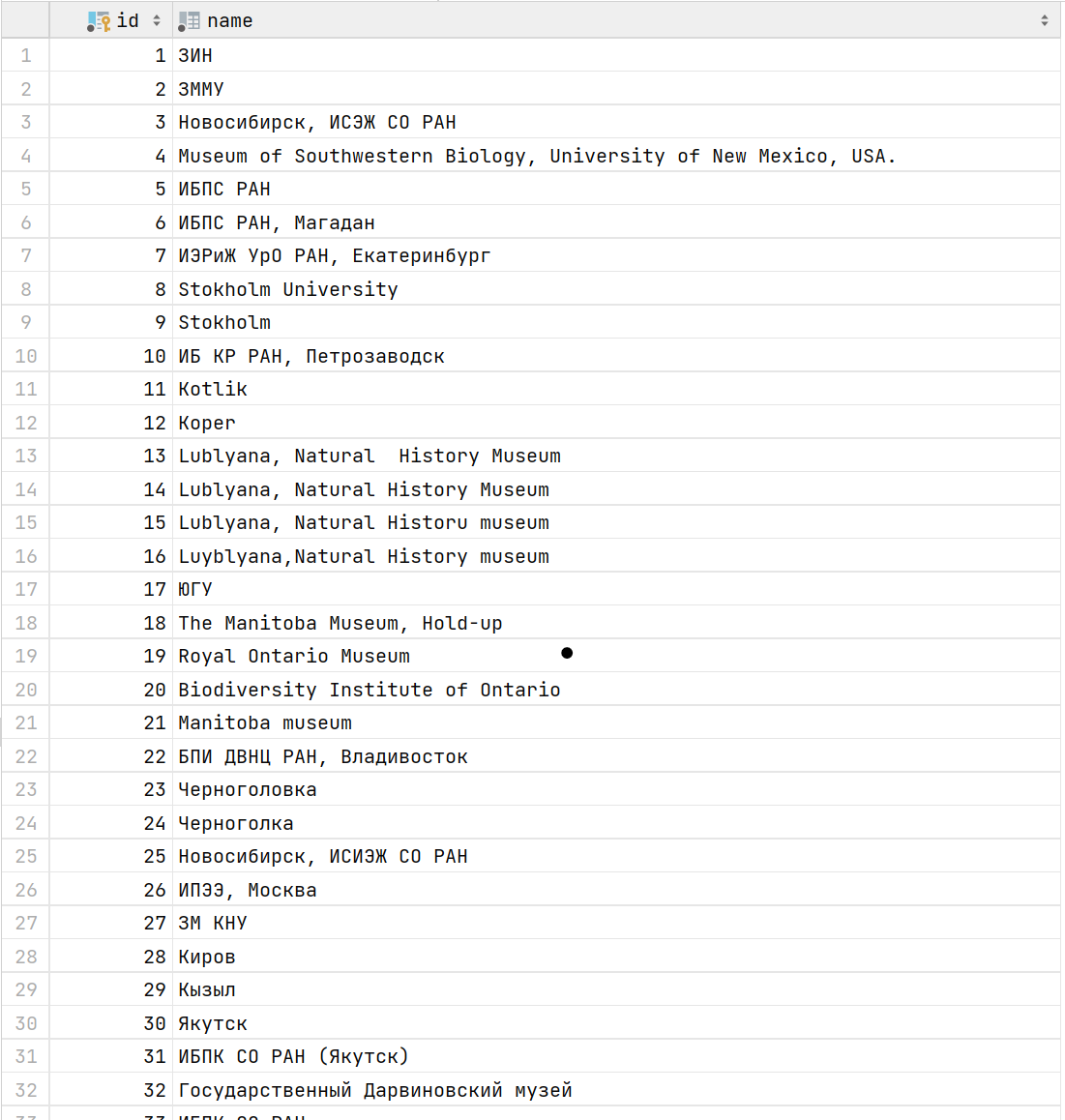


Рисунок A.2 – Таблица voucher\_institute

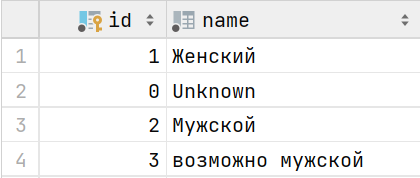


Рисунок A.3 – Таблица sex

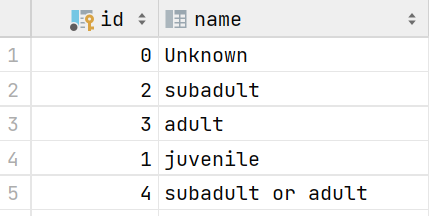


Рисунок A.4 – Таблица age

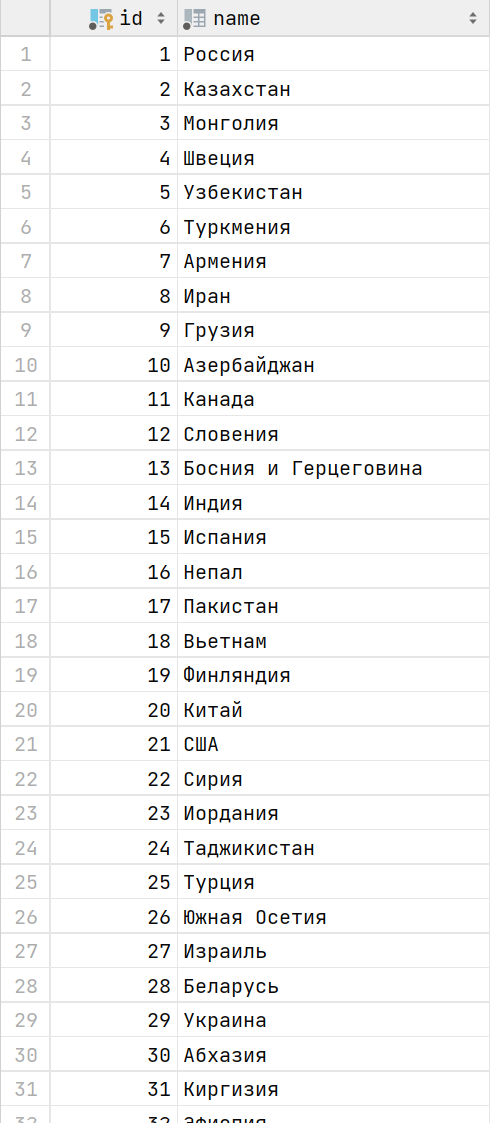


Рисунок A.5 – Таблица country

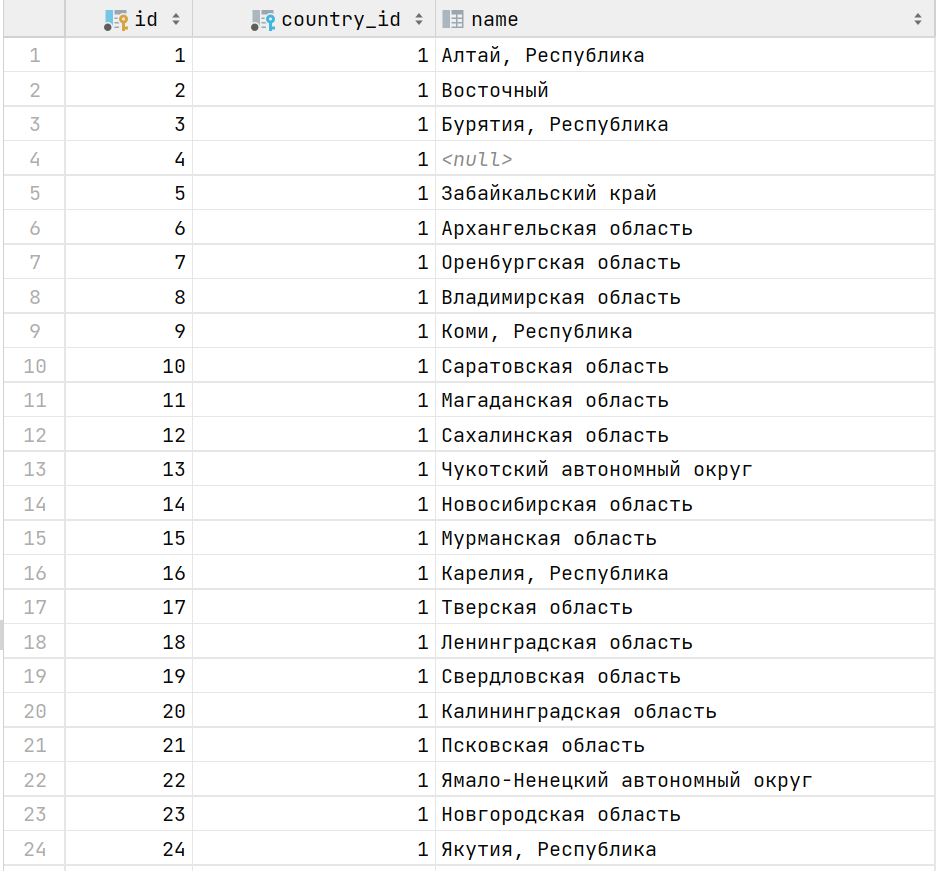


Рисунок A.6 – Таблица region

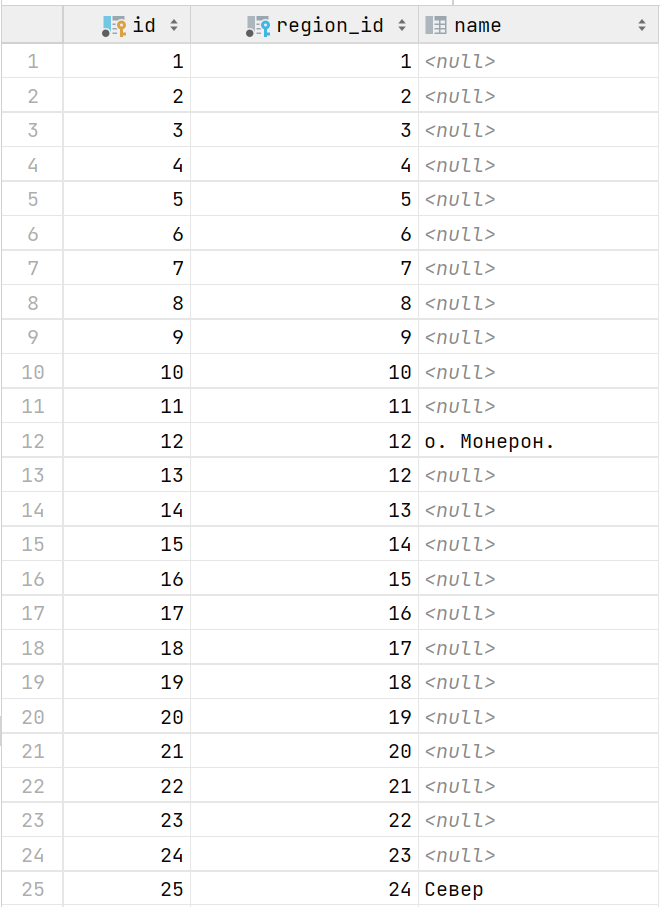


Рисунок A.7 – Таблица subregion



Рисунок A.8 – Таблица order

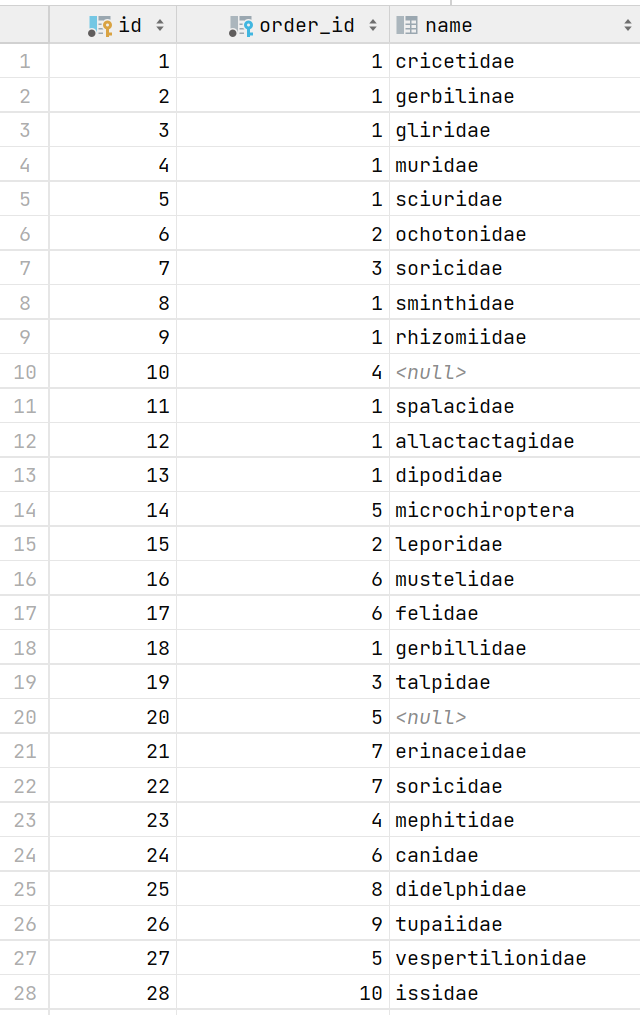


Рисунок A.9 – Таблица family

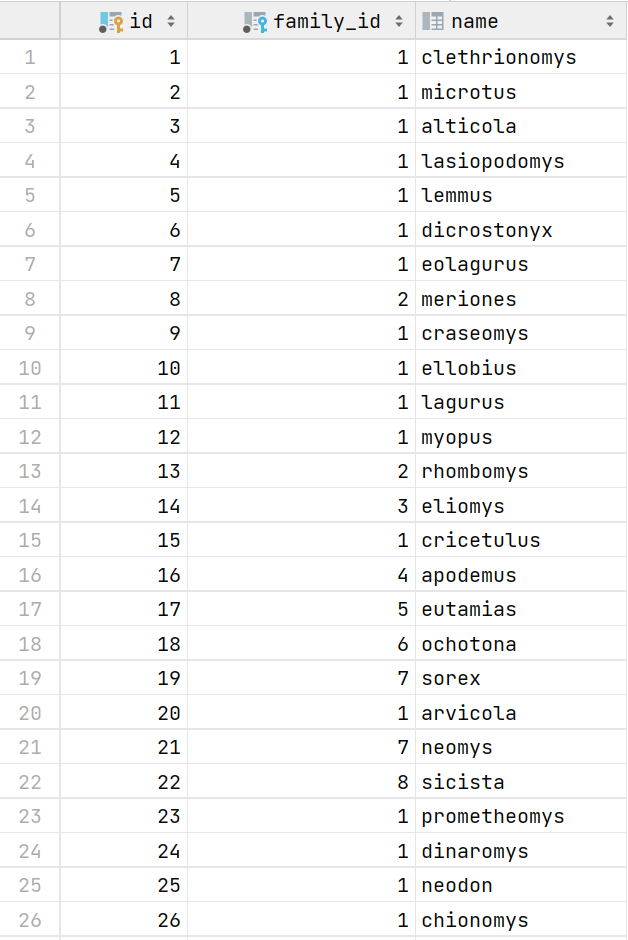


Рисунок A.10 – Таблица genus

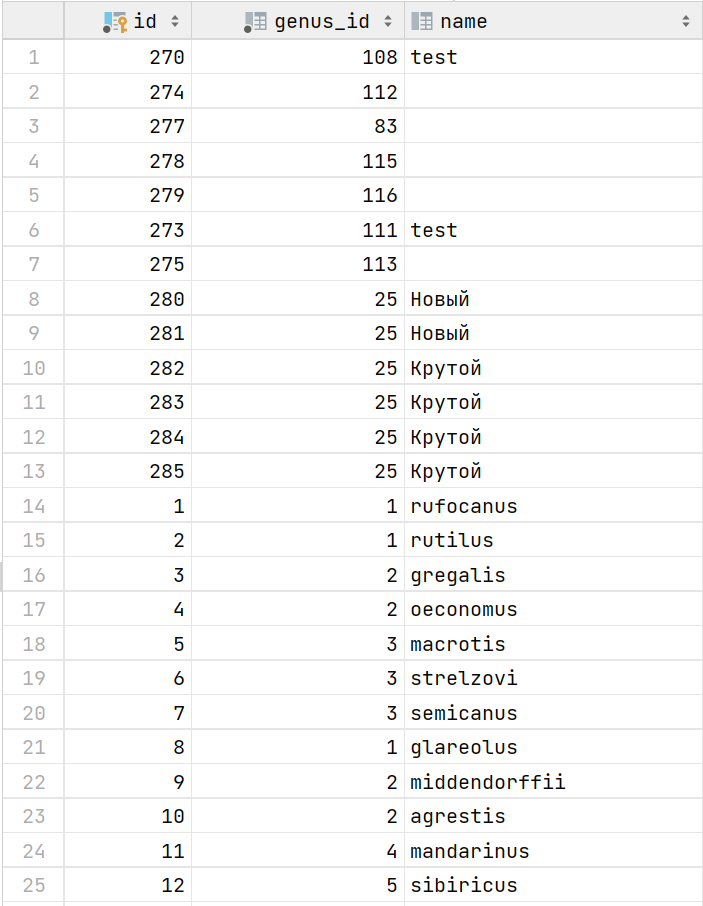


Рисунок A.11 – Таблица kind

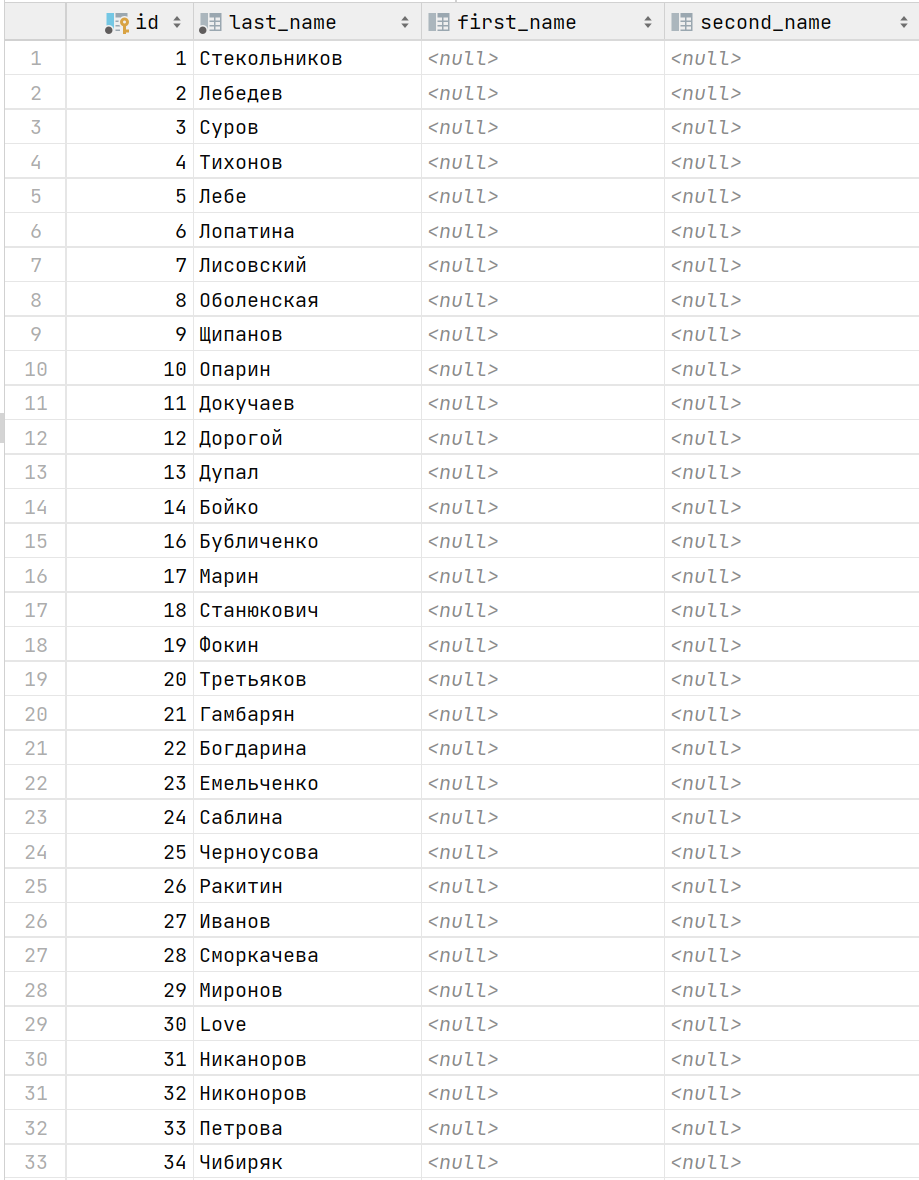


Рисунок A.12 – Таблица collector

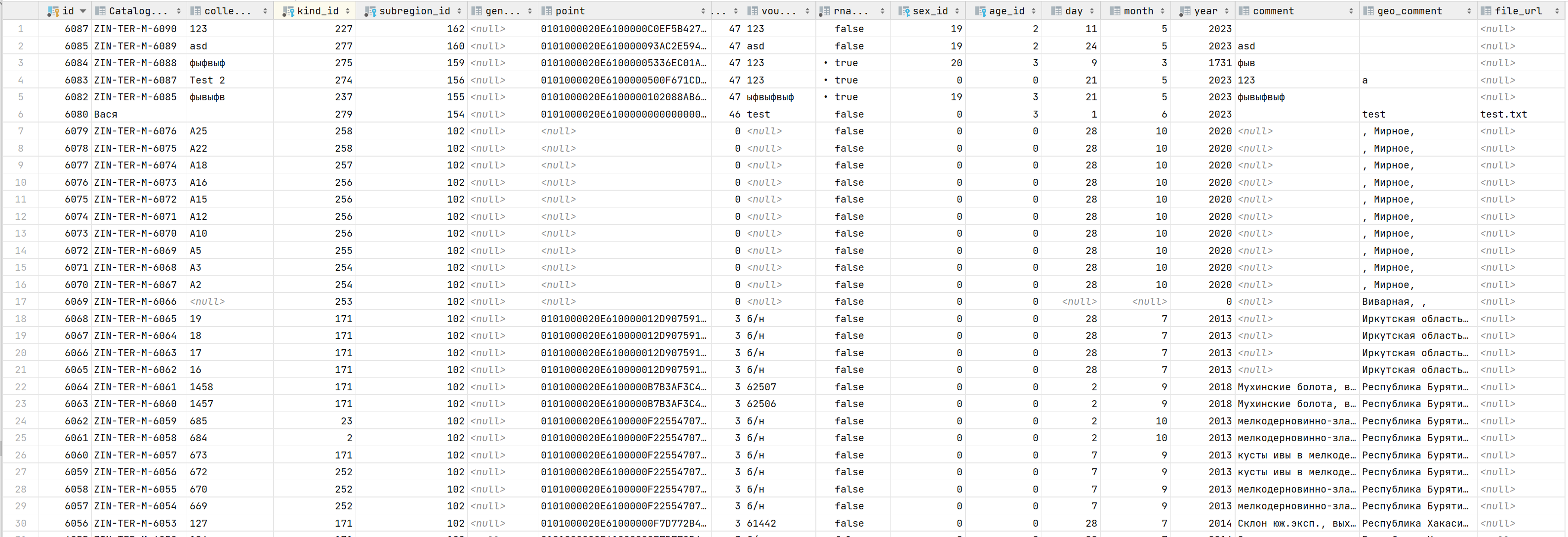


Рисунок A.13 – Таблица collection

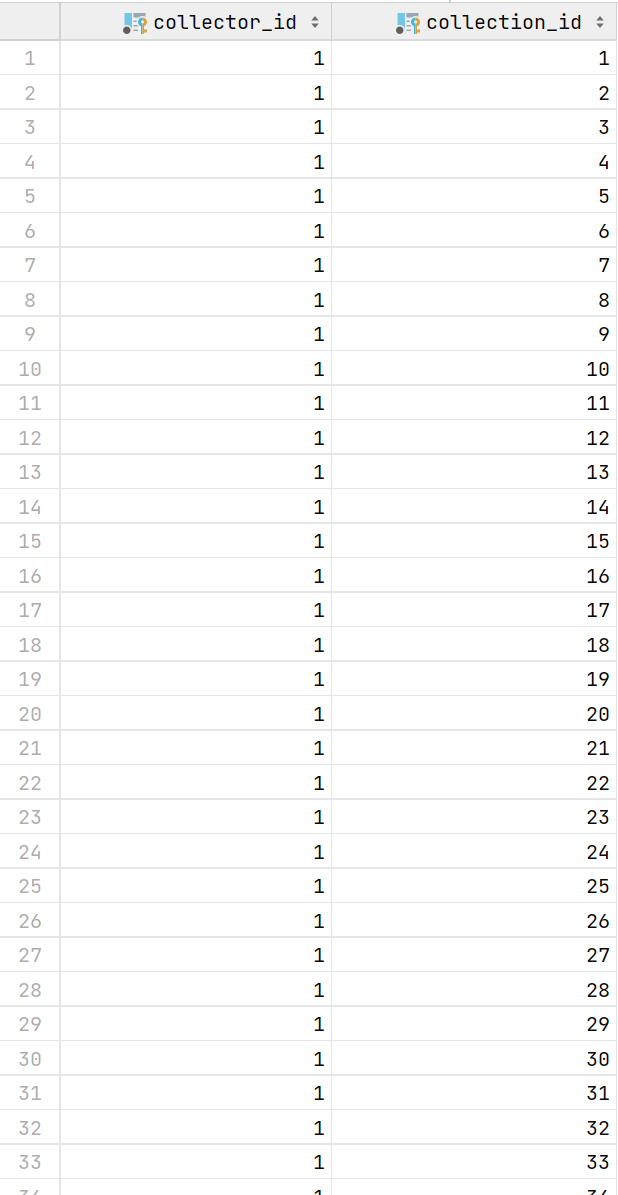


Рисунок A.14 – Таблица collector\_to\_collection