Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»
Институт компьютерных наук и кибербезопасности
Высшая школа технологий искусственного интеллекта

КУРСОВАЯ РАБОТА

УПРАВЛЕНИЕ МЕТАДАННЫМИ РЕЛЯЦИОННОЙ СИСТЕМЫ УПРАВЛЕНИЯ БАЗЫ ДАННЫХ POSTGRESQL

Выполнил:
студент гр. 5140201/50301 А.С. Тимофеев

Преподаватель:
доцент ВШТИИ ИКНК С.Г. Попов

Санкт-Петербург
2025

СОДЕРЖАНИЕ

ПОСТАНОВКА ЗАДАЧИ

В рамках работы необходимо реализовать приложение для работы с метаданными реляционной системы управления базой данных PostgreSQL и реализовать генератор подмножества `SELECT` запросов для заданных баз данных, а также хранение истории сгенерированных запросов. Для решения данной задачи выделены следующие подзадачи:

1. Описание заданного окружения.
2. Создание хранилища метаданных:
   - проектирование схемы базы данных метаданных;
   - составление запросов к `information schema`;
   - реализация сохранения метаданных по DSN-строке подключения к СУБД;
3. Реализация графического интерфейса.
4. Разработка генератора подмножества `SELECT`.
5. Организация хранения истории запросов и возможности их повторного выполнения.

1 ОПИСАНИЕ ЗАДАННОГО ОКРУЖЕНИЯ

Общая схема сетевого взаимодействия сущностей

В текущей работы рассматриваются три схемы и три СУБД расположенных удаленном сервере в отдельных Docker-контейнерах:

– База данных метаданных;
– База данных по баскетболу;
– База данных по игре Dota 2;

На Рисунке 1 представлена схема взаимодействия приложения управления метаданными и заданными базами данных.

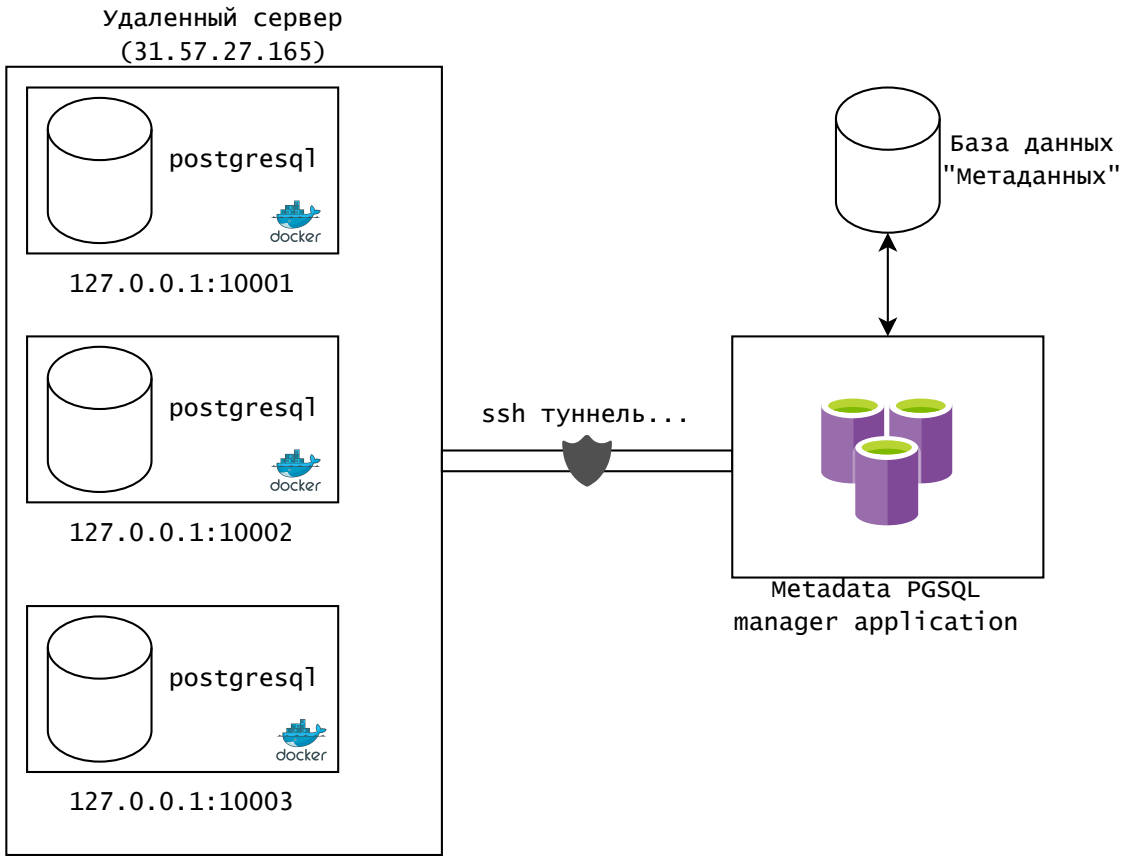

Рис. 1 — Схема сетевого взаимодействия приложения и баз данных

В рамках данной работы соединение с удаленным сервером с развернутыми СУБД выполнено через SSH-туннель.

Развертывание и установка соединения с БД

Для развертывания Docker-контейнеров были использованы Docker-compose скрипты представленные в Листинге 1.1

```
services:
metadata-db:
    image: postgres:16
    container_name: metadata-db
    restart: unless-stopped
    env_file: .env
    ports:
        - "127.0.0.1:${VPS_PORT}:5432"
    volumes:
        - metadata_db_data:/var/lib/postgresql/data
volumes:
    metadata_db_data:
```

Листинг 1.1 — docker-compose.yml файл для создания контейнера с СУБД

Параметр VPS-PORT задается в файле переменных окружения .env, пример файла представлен в Листинге 1.2

```
POSTGRES_DB=app
POSTGRES_USER=appuser
POSTGRES_PASSWORD=ChangeMe_12345
VPS_PORT=10001
```

Листинг 1.2 — .env файл для создания контейнера с СУБД

В .env файле дополнительно настраиваются переменные окружения для подключения к базе данных: POSTGRES_DB, POSTGRES_USER, POSTGRES_PASSWORD.

Для подключения к базе данных используется DSN (Data soruce name) по следующему шаблону:

postgresql://[user[:password]@][host][:port]/[database_name].

Для установки соединения необходимо настройку SSH-туннеля для портов, соответствующих контейнеров на удаленном сервере.

Описание взаимодействия внутренних компонент

В рамках курсовой работы были выделены следующие основные компоненты:

1. `Metadata Manager` - приложение (web-сервер).
2. `Metadata Database` - СУБД Postgresql с базой данных для хранения метаданных.
3. Функциональные компоненты взаимодействия с СУБД:
   - `Metadata Extractor Service` - извлечение метаданных из базы данных;
   - `Metadata Writer Service` - запись в texttttMetadata Database;
   - `Metadata Reader Service` - чтение из texttttMetadata Database;
4. `Web-UI` - пользовательский графический веб-интерфейс.

На Рисунке 2 представлена схема взаимодействия взаимодействия основных компонент приложения.

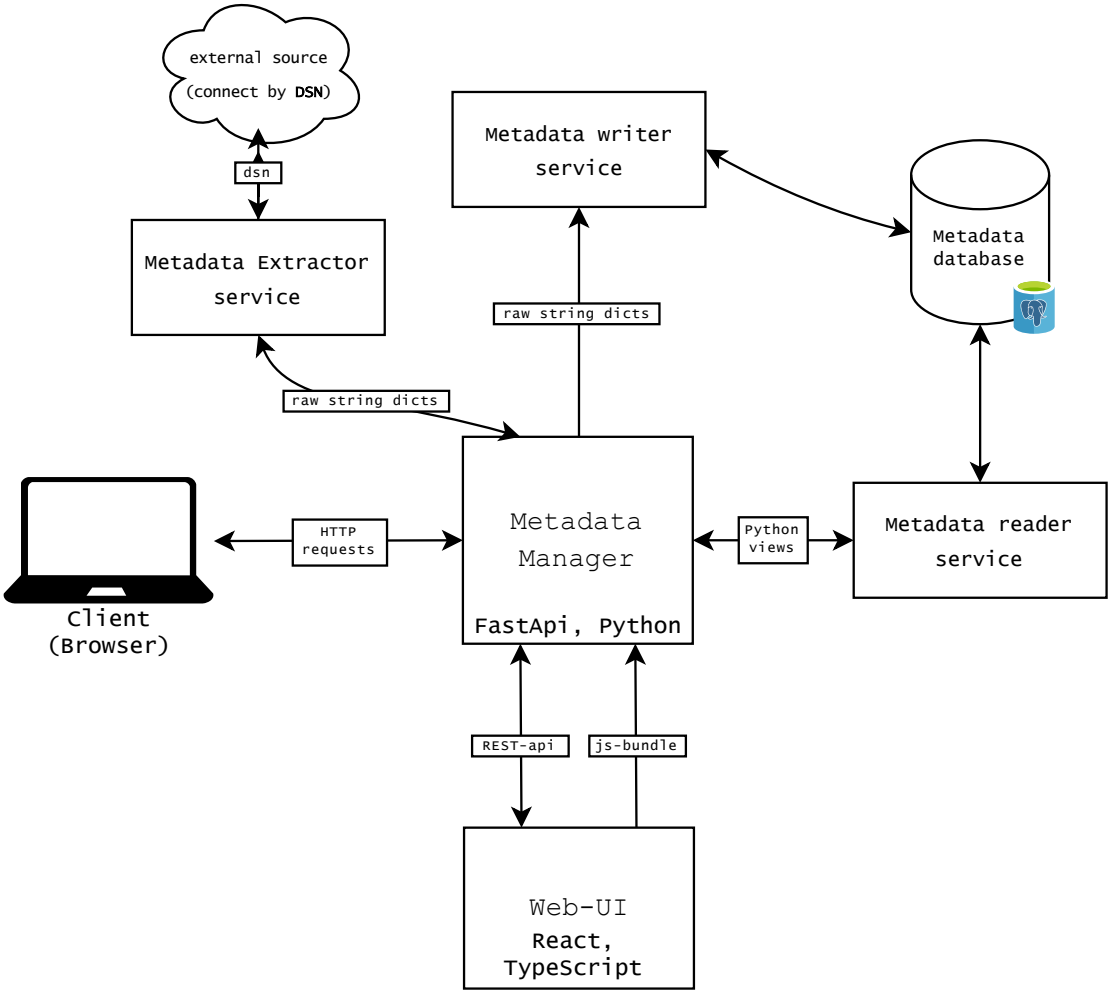

Рис. 2 — Схема взаимодействия основных компонент приложения

Для реализации приложения использованы следующие языки программирования и технологии:

- Python 3.12;
- TypeScript;
- React
- FastApi, uvicorn web server;
- PostgreSQL;

2 ХРАНИЛИЩЕ МЕТАДАННЫХ

Метаданные - информация о другой информации, или данные, относящиеся к дополнительной информации о содержимом или объекте. Метаданные раскрывают сведения о признаках и свойствах, характеризующих какие-либо сущности, позволяющие автоматически искать и управлять ими в больших информационных потоках. [1]

2.1 Схема базы данных

Для реляционной базы данных, рассматриваемой в курсовой работе, выбран следующий набор метаданных, характеризующих базу данных:

1. Имя базы данных.
2. Секреты для подключения к базе данных:
   - сетевой адрес;
   - порт;
   - имя пользователя;
   - пароль;
3. Набор таблиц.
4. Набор колонок.
5. Набор первичных ключей.
6. Набор внешних ключей.

Данный набор сущностей позволяет реализовать генератор подмножества `SELECT`-запросов.

На Рисунке 4 представлена реляционная схема базы данных метаданных, хранящая вышеописанные сущности.

`SQL`-скрипт создания представленной схемы приведен в Приложении А.

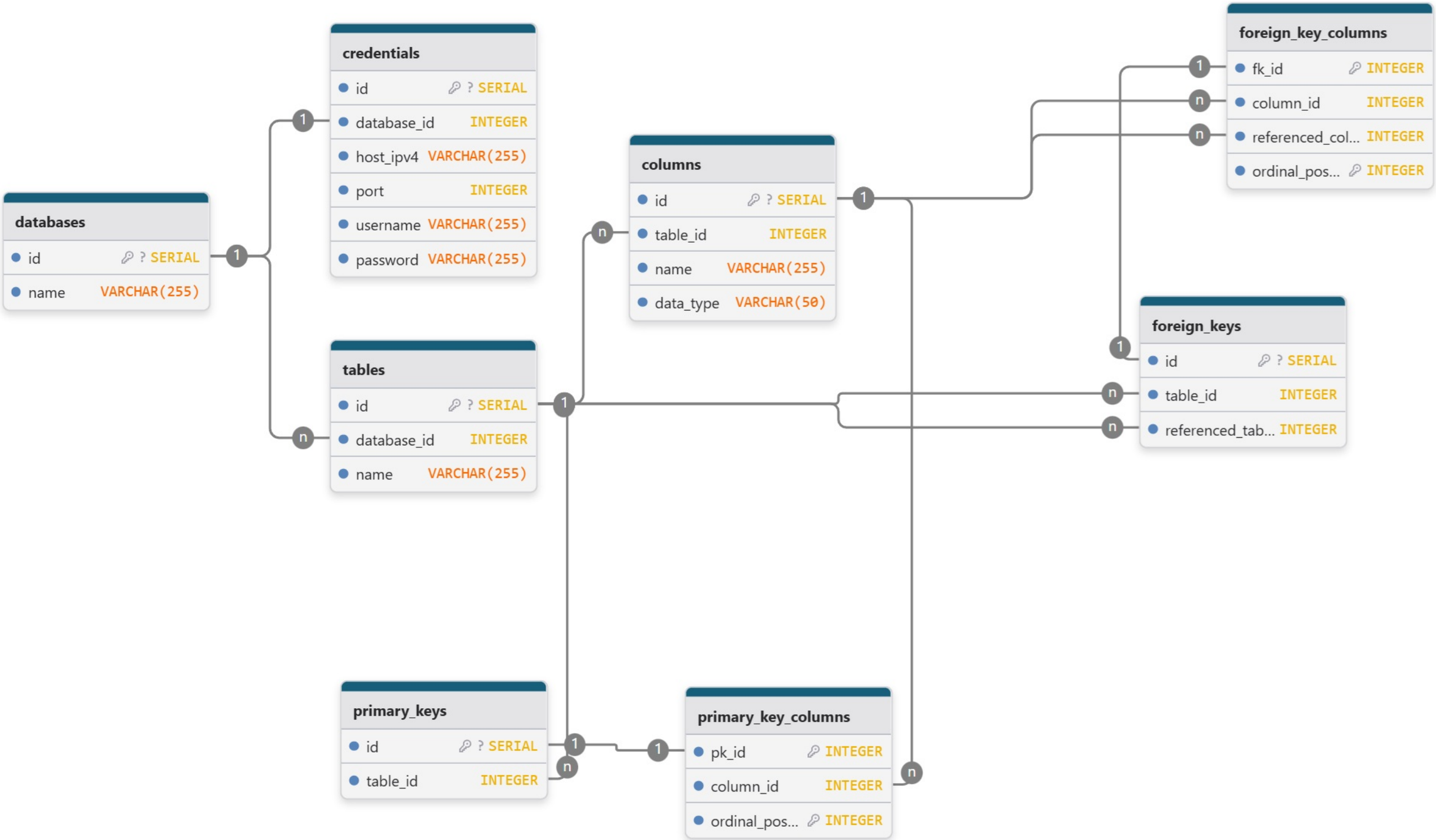

Рисунок 4 — Рассмотрение заявки на турнир

2.2 Запросы к `information schema`

Каждая созданная в СУБД `PostgreSQL` база данных содержит отдельную информационную схему, называющуюся `inforamtion schmmea`. Данная схема содержит таблицы, колонки и связи созданные пользователем. [2]

Получение списка таблиц

Представление `information_schema.tables` содержит в себе таблицы в текущей базе данных. При обращении к ней есть только те таблицы, которые доступны текущему пользователю. [3]

В Листинге 2.1 приведен пример выполнения запроса к `information schema` для получения имен таблиц.

```
SELECT
table_name
FROM information_schema.tables
WHERE table_schema = 'public' AND table_type='BASE␣TABLE'
ORDER BY table_name;

     table_name
----------------------
columns
credentials
databases
foreign_key_columns
foreign_keys
primary_key_columns
primary_keys
saved_queries
tables
```

Листинг 2.1 — Пример выполнения запроса для получения списка таблиц

Получение списка колонок

Представление `information_schema.columns` содержит в себе колонки в текущей базе данных. При обращении к ней есть только те таблицы, которые доступны текущему пользователю.

В Листинге 2.2 приведен пример выполнения запроса к `information schema` для получения имен колонок, типов данных и порядка колонки.

```
SELECT
ordinal_position,
column_name,
data_type
FROM information_schema.columns
WHERE table_schema = 'public'
AND table_name = 'credentials'
ORDER BY ordinal_position;

 ordinal_position | column_name |     data_type
------------------+-------------+--------------------
                1 | id          | integer
                2 | database_id | integer
                3 | host_ipv4   | character varying
                4 | port        | integer
                5 | username    | character varying
                6 | password    | character varying
```

Листинг 2.2 — Пример выполнения запроса для получения списка колонок для таблицы credentials

Получение списка первичных ключей

Представление `information_schema.table_constraints` содержит в себе необходимую информацию о первичных и внешних ключах таблицы. В представлении `information_schema.key_column_usage` содержится информация о используемых ключах.

В Листинге 2.3 приведен пример выполнения запроса к `information schema` для получения списка первичных ключей для таблицы `databases`.

```
SELECT
kcu.column_name AS column_name,
kcu.ordinal_position AS position_in_pk
FROM information_schema.table_constraints AS tc
JOIN information_schema.key_column_usage AS kcu`
ON tc.constraint_name = kcu.constraint_name
AND tc.table_schema = kcu.table_schema
WHERE tc.constraint_type = 'PRIMARY␣KEY'
```

```
AND tc.table_schema = 'public'
AND tc.table_name = 'databases'
ORDER BY kcu.ordinal_position;

 column_name | position_in_pk
-------------+---------------
 id          |              1
```

Листинг 2.3 — Пример выполнения запроса для получения списка первичных ключей для таблицы databases

Получение списка внешних ключей

В Листинге 2.4 приведен пример выполнения запроса к `information schema` для получения списка внешних ключей для таблицы `databases`.

```
SELECT
kcu.table_name            AS src_table,
kcu.column_name           AS src_column,
ccu.table_name            AS tgt_table,
ccu.column_name           AS tgt_column,
kcu.ordinal_position      AS position
FROM information_schema.table_constraints AS tc
JOIN information_schema.key_column_usage AS kcu
ON tc.constraint_name = kcu.constraint_name
AND tc.table_schema = kcu.table_schema
JOIN information_schema.constraint_column_usage AS ccu
ON ccu.constraint_name = tc.constraint_name
AND ccu.constraint_schema = tc.table_schema
WHERE tc.constraint_type = 'FOREIGN␣KEY'
AND kcu.table_schema = 'public'
AND kcu.table_name = 'credentials'
ORDER BY position;
  src_table  | src_column  | tgt_table | tgt_column | position
-------------+-------------+-----------+------------+----------
credentials  | database_id | databases | id         |        1
```

Листинг 2.4 — Пример выполнения запроса для получения списка внешних ключей для таблицы credentials

2.3 Использование построенных запросов

Чтение метаданных

Полученные запросы используются в `core/extractor-service` для получения метаданных из внешней базы данных. Для подключения к внешней базе данных используется DSN-строка, например: `postgresql://appuser:password123@31.124.33.3:5432/databasename`. Полученные данные представляются как есть в виде словарей.

Реализованный модуль для извлечения метаданных содержит следующие основные методы:

- `list_tables(database)` - получение списка таблиц;
- `list_columns(database, table)` - получение списка колонок;
- `list_primary_keys(database, table, column)` - получение списка первичных ключей;
- `list_foreign_keys(database, table, column)` - получение списка внешних ключей;

В Листинге 2.5 приведен пример ”псевдовызова”метода и результат вызова.

```
> extractor('postgresql://appuser:password123@31.124.33.3:5432/
   databasename')
     .list_columns('users')
>
 [
     {'name': 'id', 'data_type': 'integer'},
     {'name': 'name', 'data_type': 'text'},
     {'name': 'age', 'data_type': 'integer'},
     {'name': 'is_active', 'data_type': 'boolean'}
 ]
```

Листинг 2.5 — Пример выполнения запроса из core/extractor-service

Запись метаданных

Модуль `core/writer-service` используются для записи метаданных из внешней базы данных в локальную базу данных метаданных. Входные данные ожидаются как есть в виде словарей.

Реализованный модуль для извлечения метаданных содержит следующие основные методы:

- `ensure_database(database_name)` - запись информации о базе данных;
- `ensure_credentials(database_name, dsn)` - запись секретов о базе данных;
- `ensure_tables(database, tables)` - запись таблиц;
- `ensure_columns(database, table, columns)` - запись колонок;
- `ensure_primary_keys(database, table, pkeys)` - запись первичных ключей;
- `ensure_foreign_keys(database, table, fkeys)` - запись внешних ключей;

Чтение метаданных

Модуль `core/reader-service` используются для чтения метаданных из локальной базы данных метаданных. Выходные данные ожидаются представление ”один ко дному”данных на уровне программы.

Реализованный модуль для чтения метаданных содержит следующие основные методы:

- `list_database(database_name)` - чтение информации о базе данных;
- `list_credentials(database_name)` - чтение секретов о базе данных;
- `list_tables(database, tables)` - чтение таблиц;
- `list_columns(database, table)` - чтение колонок;
- `list_primary_keys(database, table)` - чтение первичных ключей;
- `list_foreign_keys(database, table)` - чтение внешних ключей;

Таким образом, реализованные модули чётко разделяют зоны ответственности, а их взаимодействие через определённые контракты обеспечивает соблюдение уровней абстракции внутри приложения Manager Metadata.

ЗАКЛЮЧЕНИЕ

TBD

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Шеймович А.В.* Электронные базы метаданных и построение диалектологического атласа // Родной язык. — 2022. — № 2. — URL: %5Csmall%7Bhttps://cyberleninka.ru/article/n/elektronnye-bazy-metadannyh-i-postroenie-dialektologicheskogo-atlasa%7D ; (дата обращения: 01.11.2025).
2. *PostgreSQL Global Development Group*. PostgreSQL Documentation: Glossary. — 2025. — URL: https://www.postgresql.org/docs/18/glossary.html ; (дата обращения: 01.11.2025).
3. *PostgreSQL Global Development Group*. PostgreSQL Documentation: 35.54. tables. — 2025. — URL: https://www.postgresql.org/docs/18/infoschema-tables.html ; (дата обращения: 01.11.2025).

ПРИЛОЖЕНИЕ А

```
-- ========================
-- DROP EXISTING TABLES
-- ========================
DROP TABLE IF EXISTS
credentials,
foreign_key_columns,
foreign_keys,
primary_key_columns,
primary_keys,
columns,
tables,
databases
CASCADE;

-- ========================
-- DATABASES
-- ========================
CREATE TABLE databases (
id SERIAL PRIMARY KEY,
name VARCHAR(255) NOT NULL
);

-- ========================
-- TABLES
-- ========================
CREATE TABLE tables (
id SERIAL PRIMARY KEY,
database_id INT NOT NULL REFERENCES databases(id) ON DELETE
   CASCADE,
name VARCHAR(255) NOT NULL
);

-- ========================
-- COLUMNS
-- ========================
CREATE TABLE columns (
id SERIAL PRIMARY KEY,
table_id INT NOT NULL REFERENCES tables(id) ON DELETE CASCADE,
```

```
name VARCHAR(255) NOT NULL,
data_type VARCHAR(50) NOT NULL
);

-- ========================
-- PRIMARY KEYS
-- ========================
CREATE TABLE primary_keys (
id SERIAL PRIMARY KEY,
table_id INT NOT NULL REFERENCES tables(id) ON DELETE CASCADE
);

CREATE TABLE primary_key_columns (
pk_id INT NOT NULL REFERENCES primary_keys(id) ON DELETE CASCADE,
column_id INT NOT NULL REFERENCES columns(id) ON DELETE CASCADE,
ordinal_position INT NOT NULL,
PRIMARY KEY (pk_id, ordinal_position),
UNIQUE (pk_id, column_id)
);

-- ========================
-- FOREIGN KEYS
-- ========================
CREATE TABLE foreign_keys (
id SERIAL PRIMARY KEY,
table_id INT NOT NULL REFERENCES tables(id) ON DELETE CASCADE,
referenced_table_id INT NOT NULL REFERENCES tables(id) ON DELETE
   CASCADE
);

CREATE TABLE foreign_key_columns (
fk_id INT NOT NULL REFERENCES foreign_keys(id) ON DELETE CASCADE,
column_id INT NOT NULL REFERENCES columns(id) ON DELETE CASCADE,
referenced_column_id INT NOT NULL REFERENCES columns(id) ON
   DELETE CASCADE,
ordinal_position INT NOT NULL,
PRIMARY KEY (fk_id, ordinal_position),
UNIQUE (fk_id, column_id, referenced_column_id)
);

-- ========================
-- CREDENTIALS
```

```
-- ======================
CREATE TABLE credentials (
id SERIAL PRIMARY KEY,
database_id INT NOT NULL REFERENCES databases(id) ON DELETE
   CASCADE,
host_ipv4 VARCHAR(255) NOT NULL,
port INT NOT NULL CHECK (port > 0 AND port <= 65535),
username VARCHAR(255) NOT NULL,
password VARCHAR(255) NOT NULL
);

-- ========================================
-- SAVED QUERIES
-- ========================================
CREATE TABLE saved_queries (
id SERIAL PRIMARY KEY,
database_id INTEGER NOT NULL REFERENCES databases(id) ON DELETE
   CASCADE,
sql_query TEXT NOT NULL,
created_at TIMESTAMP DEFAULT NOW()
);
```

Листинг 2.6 — SQL-скрипт создания схемы базы данных