

**ФГАОУ ВО «МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ»**

**Лабораторная работа №7**  
**Разработка приложений СУБД**

По дисциплине:

Базы данных

Выполнил студент 1-го курса группы 243-323

Онищенко А. А.

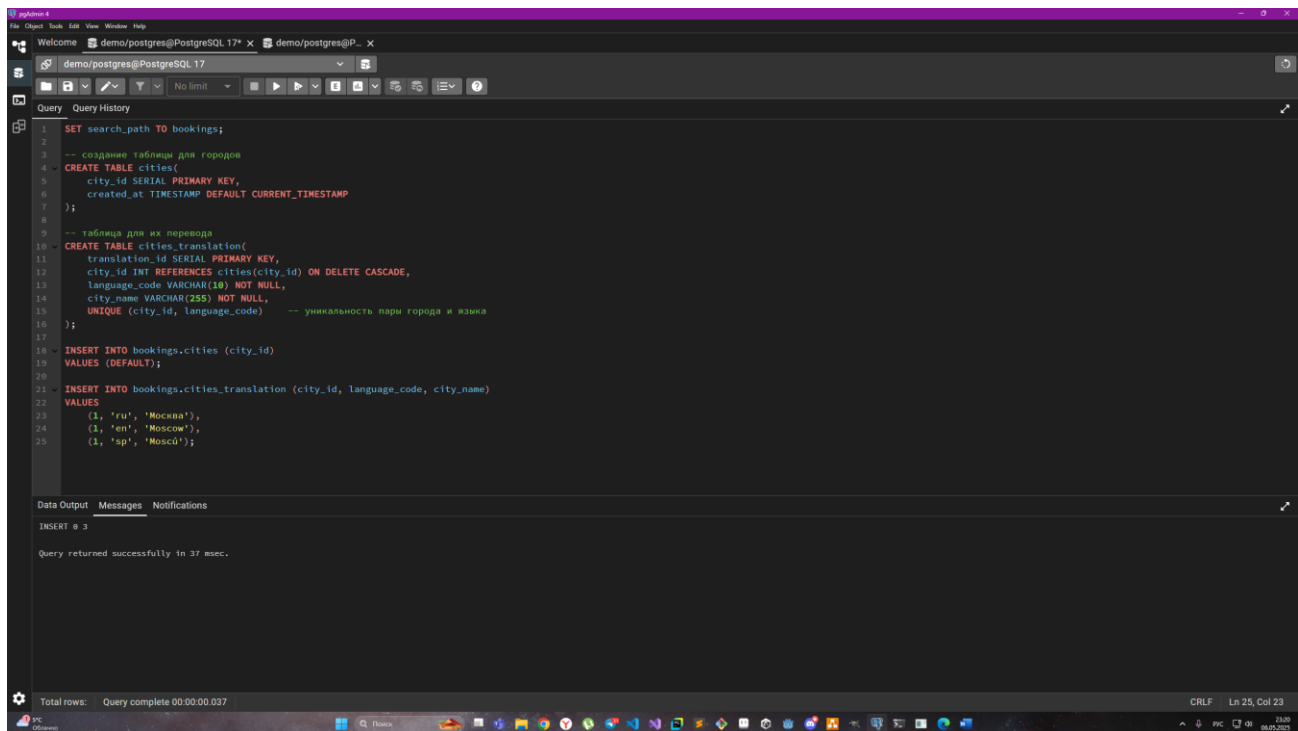
Проверил

\_\_\_\_\_ Красникова И.Н.

**Москва, 2024**

# 1. Упражнение 7.1

Создайте новую схему демонстрационной базы таким образом, чтобы названия населенных пунктов хранились в соответствии с выбранным вами способом организации многоязычности. Данные должны быть представлены как минимум на русском и английском языках. Предусмотрите возможность расширения списка используемых языков.



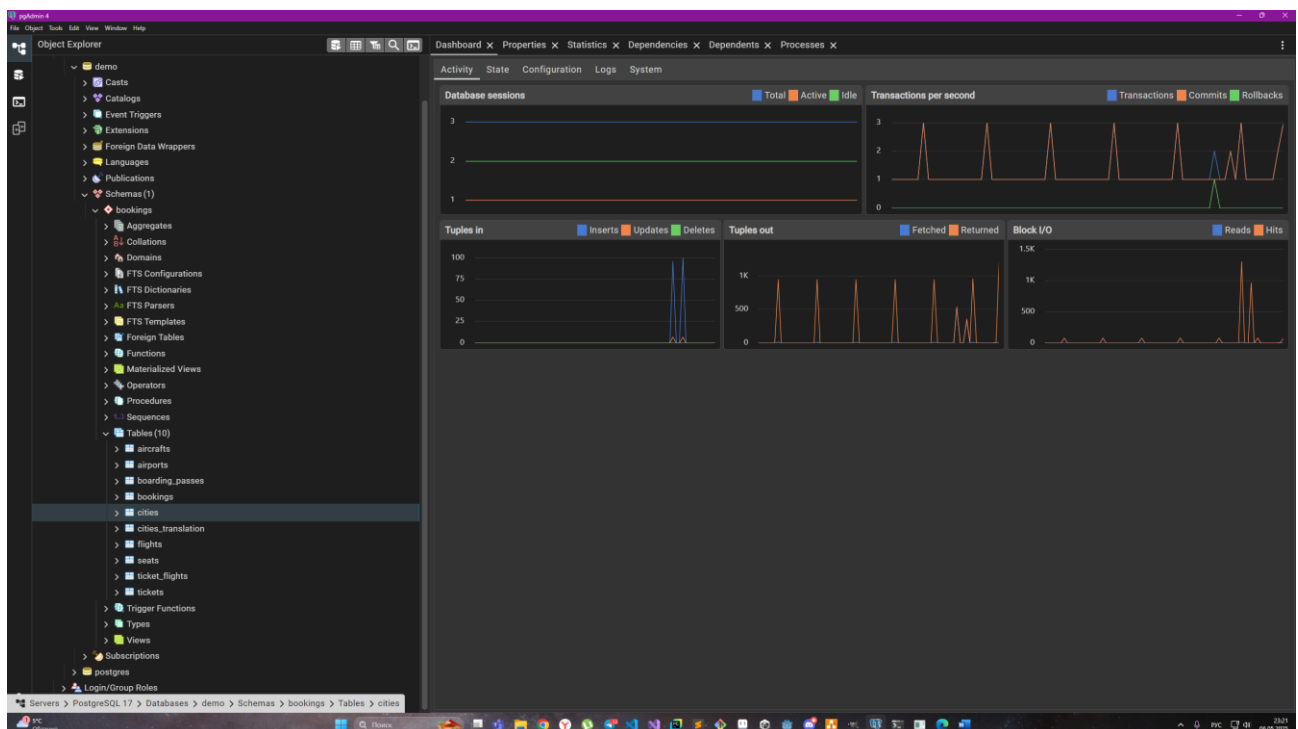
```
1 SET search_path TO bookings;
2
3 -- создание таблицы для городов
4 CREATE TABLE cities(
5     city_id SERIAL PRIMARY KEY,
6     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
7 );
8
9 -- таблица для их перевода
10 CREATE TABLE cities_translation(
11     translation_id SERIAL PRIMARY KEY,
12     city_id INT REFERENCES cities(city_id) ON DELETE CASCADE,
13     language_code VARCHAR(10) NOT NULL,
14     city_name VARCHAR(255) NOT NULL,
15     UNIQUE (city_id, language_code) -- уникальность пары города и языка
16 );
17
18 -- вставка данных
19 INSERT INTO bookings.cities (city_id)
20 VALUES (DEFAULT);
21
22 INSERT INTO bookings.cities_translation (city_id, language_code, city_name)
23 VALUES
24 (1, 'ru', 'Москва'),
25 (1, 'en', 'Moscow');
```

Data Output Messages Notifications

INSERT 0 3

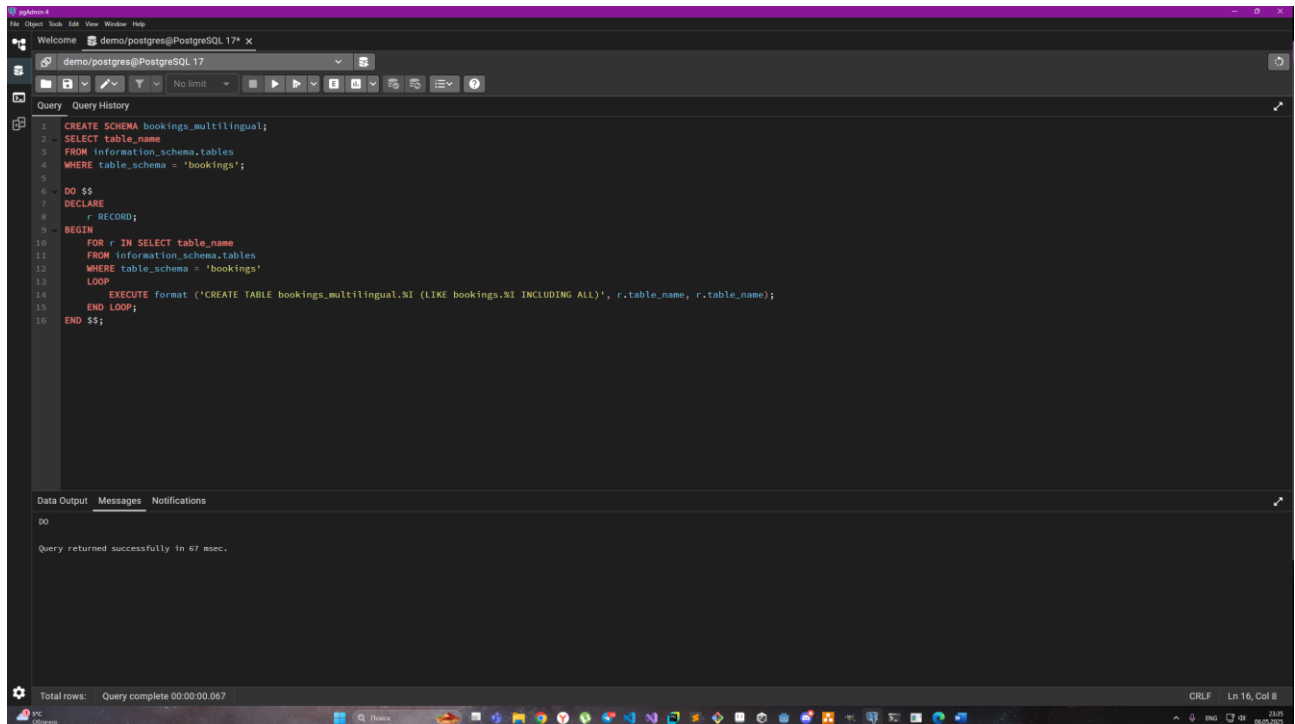
Query returned successfully in 37 msec.

Total rows: Query complete 00:00:00.037



## 2. Упражнение 7.2

Мигрируйте все данные из старой схемы демонстрационной базы в новую из предыдущего упражнения. Добавьте названия населенных пунктов на каких-либо других языках.

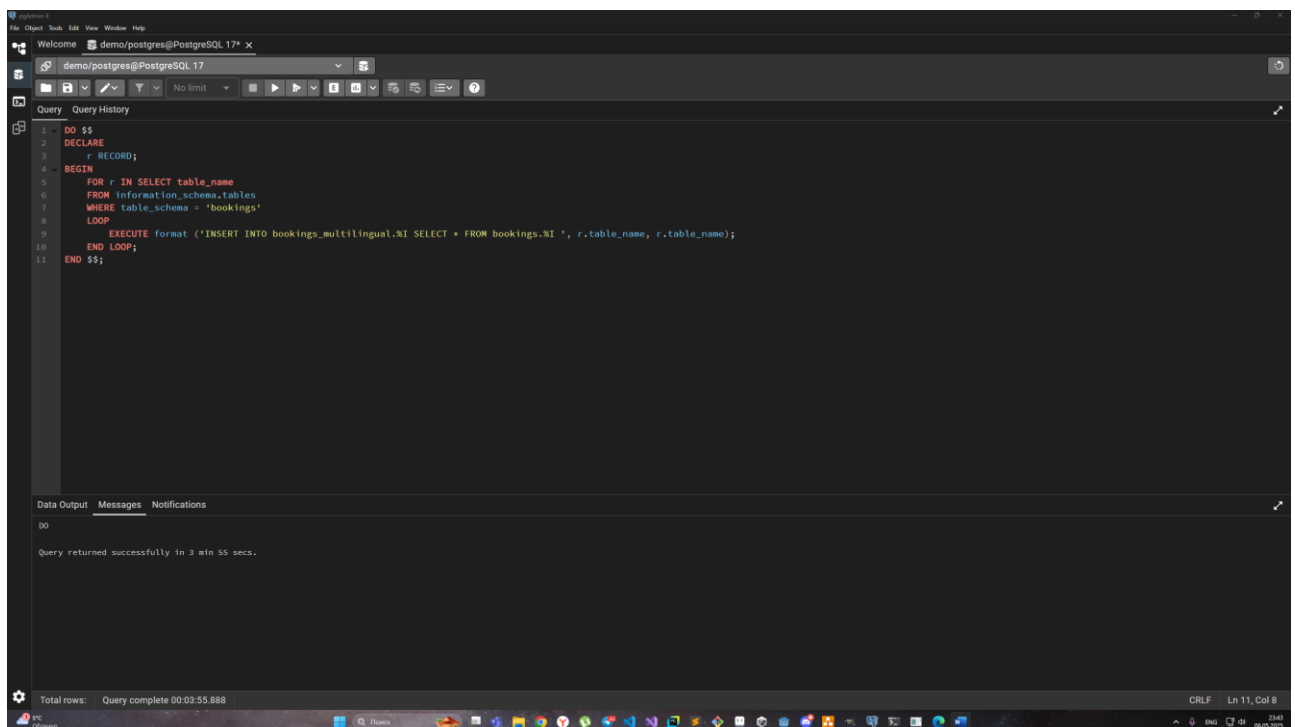


The screenshot shows the pgAdmin 4 interface with a SQL query editor. The query is as follows:

```
1 CREATE SCHEMA bookings_multilingual;
2 SELECT table_name
3 FROM information_schema.tables
4 WHERE table_schema = 'bookings';
5
6 DO $$
7 DECLARE
8     r RECORD;
9 BEGIN
10     FOR r IN SELECT table_name
11               FROM information_schema.tables
12               WHERE table_schema = 'bookings'
13     LOOP
14         EXECUTE format ('CREATE TABLE bookings_multilingual.%i (LIKE bookings.%i INCLUDING ALL)', r.table_name, r.table_name);
15     END LOOP;
16 END $$;
```

The Data Output pane shows the message: "Query returned successfully in 67 msec."

Total rows: Query complete 00:00:00.067

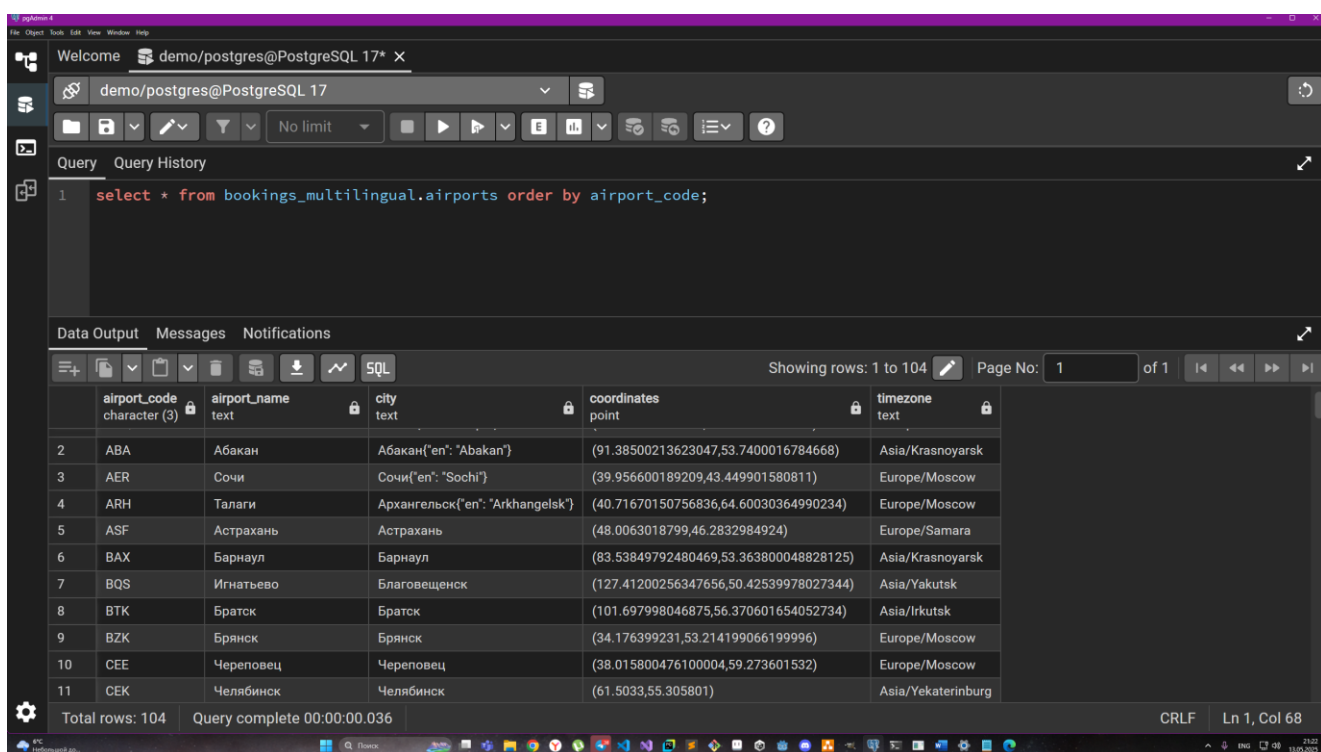
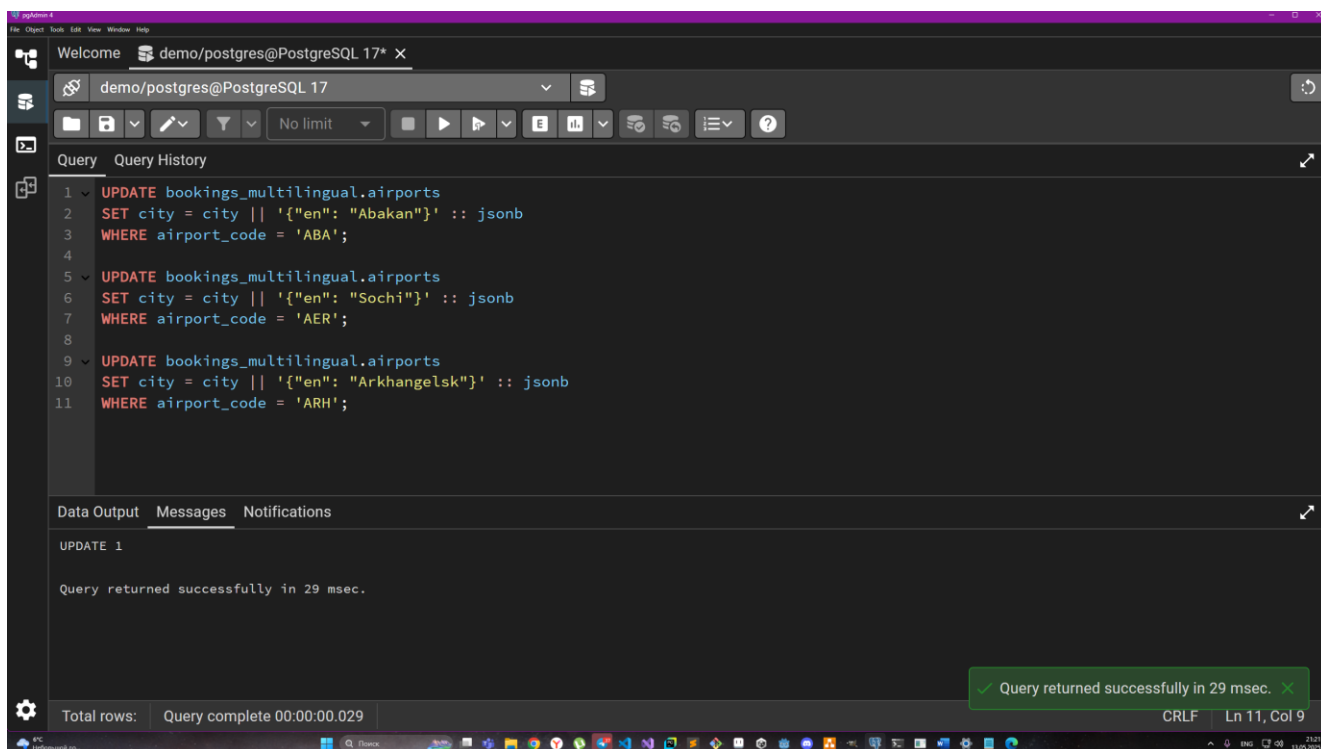


The screenshot shows the pgAdmin 4 interface with a SQL query editor. The query is as follows:

```
1 DO $$
2 DECLARE
3     r RECORD;
4 BEGIN
5     FOR r IN SELECT table_name
6               FROM information_schema.tables
7               WHERE table_schema = 'bookings'
8     LOOP
9         EXECUTE format ('INSERT INTO bookings_multilingual.%i SELECT * FROM bookings.%i ', r.table_name, r.table_name);
10    END LOOP;
11 END $$;
```

The Data Output pane shows the message: "Query returned successfully in 3 min 55 secs."

Total rows: Query complete 00:03:55.888



### 3. Упражнение 7.3

Напишите запрос, для каждого города показывающий количество пассажиров, прилетающих в него из Москвы в какой-нибудь определенный день. Запрос должен выдавать названия населенных пунктов на английском языке.

The screenshot shows the pgAdmin 4 interface with a SQL query executed against a PostgreSQL database. The query is as follows:

```

1 SELECT
2     f.arrival_city,
3     COUNT(*) AS total
4 FROM
5     flights_v f
6 WHERE
7     f.departure_city = 'Moscow'
8     AND DATE(f.scheduled_arrival) = '2017-08-10'
9 GROUP BY
10    f.arrival_city
11 ORDER BY
12    total DESC;

```

The results are displayed in a table with the following data:

	arrival_city	total
1	St. Petersburg	12
2	Bryansk	9
3	Ulyanovsk	5
4	Yoshkar-Ola	4
5	Petrozavodsk	4
6	Dudinka	2

The status bar indicates "Total rows: 70" and "Query complete 00:00:00.041".

## 4. Упражнение 7.4

Реализуйте функцию покупки билета. С точки зрения пассажира покупка состоит из двух этапов. На первом этапе система выдает список возможных перелетов для указанных параметров: пункты отправления и назначения, количество мест, дата вылета. На втором — выбранный пассажиром вариант оформляется в виде бронирования, и пассажиру возвращается номер бронирования

The screenshot shows the pgAdmin 4 interface with a PL/pgSQL function defined and executed. The function is as follows:

```

1 -- Удаляем функцию, если она есть
2 DROP FUNCTION IF EXISTS find_available_flights(character, character, date);
3
4 -- Создаем функцию
5 CREATE OR REPLACE FUNCTION find_available_flights(
6     p_departure_airport character,
7     p_arrival_airport character,
8     p_flight_date DATE
9 )
10 RETURNS TABLE (flight_id INT, departure_airport character, arrival_airport character, scheduled_departure TIMESTAMP WITH TIME ZONE) AS $$
11
12 BEGIN
13     RETURN QUERY
14     SELECT
15         f.flight_id,
16         f.departure_airport,
17         f.arrival_airport,
18         f.scheduled_departure
19     FROM
20         flights f
21     WHERE
22         f.departure_airport = p_departure_airport
23         AND f.arrival_airport = p_arrival_airport
24         AND f.scheduled_departure :: date = p_flight_date;
25 END;
26
27 $$ LANGUAGE plpgsql;
28
29 --Пример
30 SELECT * FROM find_available_flights('AER', 'VKO', '2017-09-10');
31

```

The results of the function execution are displayed in a table with the following data:

flight_id	departure_airport	arrival_airport	scheduled_departure
30533	AER	VKO	2017-09-10 17:00:00+03

The status bar indicates "Total rows: 1" and "Query complete 00:00:00.049".

The screenshot shows the pgAdmin 4 interface with a query window titled 'demo/postgres@PostgreSQL 17\* X'. The query is as follows:

```
1 DROP FUNCTION IF EXISTS create_booking(character varying, character varying, character varying, numeric);
2
3 CREATE OR REPLACE FUNCTION create_booking(
4     p_passenger_name VARCHAR,
5     p_seat_no VARCHAR,
6     p_flight_code VARCHAR,
7     p_total_amount NUMERIC
8 )
9 RETURNS VARCHAR AS $$
10 DECLARE
11     book_ref VARCHAR;
12 BEGIN
13     -- генерация уникального номера
14     book_ref := upper(
15         substring(
16             md5(random()::text || clock_timestamp()::text)
17             from 1 for 6
18         )
19     );
20
21     -- Вставляем новую запись в таблицу бронирования
22     INSERT INTO bookings_multilingual.bookings (book_ref, book_date, total_amount)
23     VALUES (book_ref, NOW(), p_total_amount);
24
25     RETURN book_ref;
26 END;
27 $$ LANGUAGE plpgsql;
28
29 -- Пример использования функции
30 SELECT create_booking('Alex Udarov', '9A', '21B', 1000.00);
```

The 'Data Output' tab shows the result of the query:

create_booking character varying
1 28B282

The status bar at the bottom indicates 'Total rows: 1' and 'Query complete 00:00:00.036'.

The screenshot shows the pgAdmin 4 interface with a query window titled 'demo/postgres@PostgreSQL 17\* X'. The query is as follows:

```
1 DROP FUNCTION IF EXISTS create_booking(character varying, character varying, character varying, numeric);
2
3 CREATE OR REPLACE FUNCTION create_booking(
4     p_passenger_name VARCHAR,
5     p_seat_no VARCHAR,
6     p_flight_code VARCHAR,
7     p_total_amount NUMERIC
8 )
9 RETURNS VARCHAR AS $$
10 DECLARE
11     book_ref VARCHAR;
12 BEGIN
13     -- генерация уникального номера
14     book_ref := upper(
15         substring(
16             md5(random()::text || clock_timestamp()::text)
17             from 1 for 6
18         )
19     );
20
21     -- Вставляем новую запись в таблицу бронирования
22     INSERT INTO bookings_multilingual.bookings (book_ref, book_date, total_amount)
23     VALUES (book_ref, NOW(), p_total_amount);
24
25     RETURN book_ref;
26 END;
27 $$ LANGUAGE plpgsql;
28
29 -- Пример использования функции
30 SELECT create_booking('Valeria Udarova', '18A', '21B', 1001.00);
```

The 'Data Output' tab shows the result of the query:

create_booking character varying
1 EAD415

The status bar at the bottom indicates 'Total rows: 1' and 'Query complete 00:00:00.034'.

# Ответы на контрольные вопросы

## 1. Что такое каркасы объектно-реляционных отображений?

Каркасы объектно-реляционных отображений (ORM, Object-Relational Mapping) — это технологии и инструменты, которые позволяют разработчикам работать с реляционными базами данных, используя объектно-ориентированные подходы. ORM автоматически преобразует данные между объектами в приложении и реляционными таблицами в базе данных, что упрощает взаимодействие с базой данных и уменьшает количество необходимого кода для выполнения операций с данными

## 2. Какие стратегии используются для отображения наследования объектной модели?

- Стратегия таблицы на класс (Table per Class) - каждому классу соответствует отдельная таблица, и все поля класса хранятся в этой таблице.
- Стратегия таблицы на иерархию (Table per Hierarchy) - все классы в иерархии наследования хранятся в одной таблице, и различия между классами определяются с помощью специального столбца (например, типа).
- Стратегия таблицы на подкласс (Table per Subclass) - каждый подкласс имеет свою таблицу, которая содержит только поля, специфичные для этого подкласса, а общие поля хранятся в родительском классе.

## 3. Когда применять каркасы?

- Когда необходимо упростить взаимодействие с базой данных, особенно в приложениях с сложной бизнес-логикой.
- Когда требуется уменьшить количество кода для выполнения операций CRUD (создание, чтение, обновление, удаление).
- Когда необходимо обеспечить переносимость между различными СУБД, так как ORM может абстрагировать детали конкретной базы данных.
- Когда проект требует использования объектно-ориентированных подходов и паттернов проектирования.

## 4. Перечислите различные стратегии параллельного доступа к кешированным данным?

- *Оптимистичная блокировка* предполагает, что конфликты редки, и проверяет их только при сохранении изменений.
- *Пессимистичная блокировка* блокирует данные на время их использования, предотвращая доступ к ним другим пользователям.
- *Кеширование с версионированием*. Каждое изменение данных увеличивает версию, и кеш обновляется только при изменении версии.
- *Кеширование с тайм-аутом*. Данные в кеше имеют срок действия, после которого они считаются устаревшими и должны быть обновлены.

## 5. Перечислите параметры запросов?

- Фильтры - условия для ограничения выборки данных (WHERE).
- Сортировка - указание порядка сортировки результатов (ORDER BY).
- Пагинация - ограничение количества возвращаемых записей и указание смещения (LIMIT и OFFSET).
- Выбор полей - указание конкретных полей, которые нужно вернуть (SELECT).
- Группировка - объединение данных по определенным критериям (GROUP BY).

## 6. Как реализовать многоязычность на уровне базы данных?

- *Отдельные таблицы для каждого языка*. Создание отдельных таблиц для хранения данных на разных языках.
- *Столбцы для каждого языка*. Добавление отдельных столбцов для каждого языка в одну таблицу (например, title\_en, title\_fr).
- *Таблица локализации*. Создание таблицы, которая связывает идентификаторы записей с переводами на разные языки (например, translations с полями record\_id, language, translated\_text).

## 7. Что называется настройкой?

Настройка — это процесс конфигурации системы или приложения для достижения определенных целей или соответствия требованиям. В контексте баз данных это может включать настройку параметров производительности, безопасности, резервного копирования и восстановления, а также конфигурацию соединений и других аспектов, влияющих на работу базы данных.

## 8. Перечислите преимущества пользовательских функций, возвращающих отношения, по сравнению с представлениями.

- *Гибкость.* Пользовательские функции могут принимать параметры, что позволяет динамически изменять поведение функции в зависимости от входных данных.
- *Логика обработки.* Функции могут содержать сложную бизнес-логику и выполнять вычисления, которые невозможно реализовать в представлениях.
- *Управление состоянием.* Функции могут сохранять состояние между вызовами, что позволяет реализовать более сложные сценарии.
- *Оптимизация.* Функции могут быть оптимизированы для конкретных задач, что может привести к улучшению производительности по сравнению с представлениями.