Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ № 3

по теме:

«Процедуры, функции, триггеры в PostgreSQL»

по дисциплине: Проектирование и реализация баз данных

Специальность:
45.03.04 Интеллектуальные системы в гуманитарной сфере

Проверила:
Говорова М.М.
Дата: «29» мая 2023 г.
Оценка _____

Выполнила:
студентка группы К32422
Королева Е. М.

Санкт-Петербург 2022/2023

Цель работы: овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Практическое задание:

Вариант 1

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).

2. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

Вариант 2

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
2.
    2.1. Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника (см. Практическое задание 1 Лабораторного практикума (Приложение)) с максимальным учетом «узких» мест некорректных данных по входу и выходу.
    2.2. Создать авторский триггер по варианту индивидуального задания.

Выполнение работы:

Предметная область – Вариант 3. БД «Библиотека»

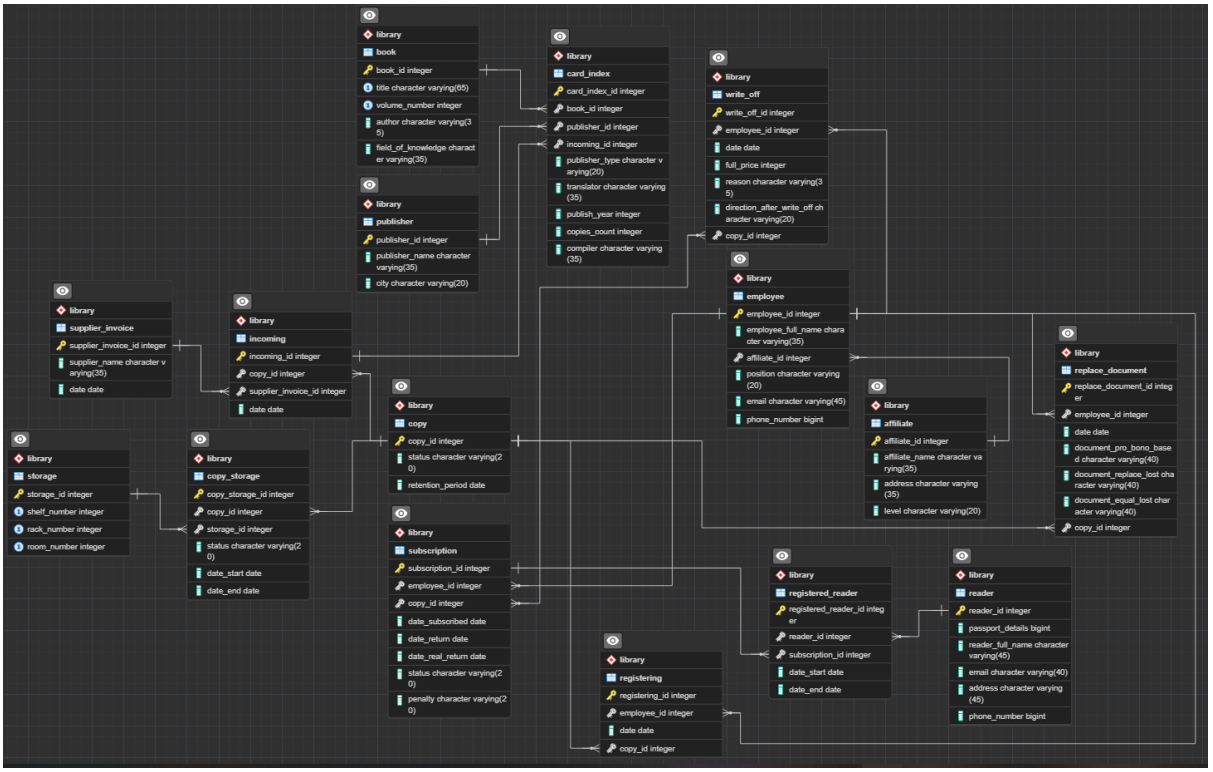

Рисунок 1 – ERD базы данных

Выполнение работы:

Индивидуальное задание:

Задание 4. Создать хранимые процедуры:

1. Для проверки наличия экземпляров заданной книги в библиотеке (процедура должна возвращать количество экземпляров книги).
2. Для ввода в базу данных новой книги.
3. Для ввода нового читателя (необходимо проверить наличие читателя в картотеке, чтобы не назначить ему номер вторично).

Выполнение:

1. Для проверки наличия экземпляров заданной книги в библиотеке (процедура должна возвращать количество экземпляров книги).

Запрос:

```
create function check_book_availability(_title text, _volume integer, _author text) returns integer
    language plpgsql
as
$$
BEGIN
    RETURN (SELECT COUNT(c.copy_id)
            FROM book b
                     JOIN card_index ci on b.book_id = ci.book_id
                     JOIN incoming i on ci.card_index_id = i.card_index_id
                     JOIN copy c on i.incoming_id = c.incoming_id
            WHERE b.title = _title
              AND b.volume_number = _volume
              AND b.author = _author
              AND c.status = 'возвращенный');
END;
$$;
```

Вызов функции:

```
SELECT check_book_availability( _title: 'Война и мир', _volume: 2, _author: 'Лев Толстой')
```

Вывод:

	chech_book_availability
1	4

2. Для ввода в базу данных новой книги.

Запрос:

```
CREATE OR REPLACE PROCEDURE insert_new_book(_title TEXT, _volume INT, _author TEXT, _field TEXT, _language TEXT)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO book(title, volume_number, author, field_of_knowledge, language)
    VALUES (_title, _volume, _field, _author, _language);
END;
$$
```

Вызов процедуры:

```
CALL insert_new_book(
     _title: 'Цветы для элджернона',
     _volume: 1,
     _author: 'Дэниел Киз',
     _field: 'Литература',
     _language: 'Английский'
);

SELECT * FROM book
WHERE book_id = (SELECT MAX(book_id) FROM book);
```

Вывод:

	book_id	title	volume_number	author	field_of_knowledge	language
1	31	Цветы для элджернона	1	Литература	Дэниел Киз	Английский

3. Для ввода нового читателя (необходимо проверить наличие читателя в картотеке, чтобы не назначить ему номер вторично).

```
CREATE OR REPLACE PROCEDURE insert_new_reader(
    _passport BIGINT,
    _name TEXT,
    _email TEXT,
    _address TEXT,
    _phone BIGINT
)
LANGUAGE plpgsql
AS $$
DECLARE
    reader_exists INT;
BEGIN
    SELECT COUNT(reader_id) INTO reader_exists
    FROM reader
    WHERE _passport = passport_details;

    IF reader_exists = 0 THEN
        INSERT INTO reader(passport_details, reader_full_name, email, address, phone_number)
        VALUES (_passport, _name, _email, _address, _phone);
    ELSE
        RAISE NOTICE 'Читатель уже есть в базе данных';
    END IF;
END;
$$
```

Вызов процедуры:

```
CALL insert_new_reader(
      _passport: 1234567890,
      _name: 'Рив Илон Маск',
      _email: 'ElonMaskTop2003@mail.ru',
      _address: 'Воронеж, ул. Спэйсиксная, д. 14к1',
      _phone: 89130371337
);

SELECT * FROM reader
WHERE reader_id = (SELECT MAX(reader_id) FROM reader)
```

Вывод:

```
CALL insert_new_reader(
     _passport: 1234567899,
     _name: 'Рив Илон Маск',
     _email: 'ElonMaskTop2003@mail.ru',
     _address: 'Воронеж, ул. Спэйсиксная, д. 14к1',
     _phone: 89130371337
);

SELECT * FROM reader
WHERE reader_id = (SELECT MAX(reader_id) FROM reader);
```

Вызов процедуры повторно, чтобы проверить исключение:

```
Output   library.library.reader
library.library> CALL insert_new_reader(
                     1234567899,
                     'Рив Илон Маск',
                     'ElonMaskTop2003@mail.ru',
                     'Воронеж, ул. Спэйсиксная, д. 14к1',
                     89130371337
                 )
Читатель уже есть в базе данных
[2023-05-24 20:40:55] completed in 2 ms
```

4. Создать триггер для логирования событий вставки, удаления, редактирования данных.

   Создадим триггер над таблицей subscription для регистрации изменений. Все события будем записывать в таблицу log_audit.

   Запрос:

   1. Создание таблицы логов

```sql
CREATE TABLE subcription_audit(
    id SERIAL PRIMARY KEY,
    operation VARCHAR(6) NOT NULL,
    operation_timestamp TIMESTAMP NOT NULL DEFAULT NOW(),
    subscription_id INT NOT NULL,
    employee_id INT NOT NULL,
    copy_id INT NOT NULL,
    date_subscribed DATE NOT NULL,
    date_return DATE NOT NULL,
    date_real_return DATE,
    status VARCHAR(20) NOT NULL,
    penalty VARCHAR(20)
);
```

   2. Создание триггерной функции

```sql
CREATE OR REPLACE FUNCTION log_subcription() RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF TG_OP = 'DELETE' THEN
        INSERT INTO subcription_audit(operation, subscription_id, employee_id, copy_id, date_subscribed, date_return, date_real_return, status, penalty)
        SELECT TG_OP,
            OLD.subscription_id,
            OLD.employee_id,
            OLD.copy_id,
            OLD.date_subscribed,
            OLD.date_return,
            OLD.date_real_return,
            OLD.status,
            OLD.penalty;
        RETURN OLD;
    ELSE
        INSERT INTO subcription_audit(operation, subscription_id, employee_id, copy_id, date_subscribed, date_return, date_real_return, status, penalty)
        SELECT TG_OP,
            NEW.subscription_id,
            NEW.employee_id,
            NEW.copy_id,
            NEW.date_subscribed,
            NEW.date_return,
            NEW.date_real_return,
            NEW.status,
            NEW.penalty;
        RETURN NEW;
    END IF;
END;
```

3. Создание триггера над таблицей

```
CREATE OR REPLACE TRIGGER trigger_log_subscription
AFTER DELETE OR UPDATE OR INSERT ON subscription
FOR EACH ROW
EXECUTE FUNCTION log_subcription();
```

4. Проверка работы триггера

```
INSERT INTO subscription(employee_id, copy_id, date_subscribed, date_return, status)
SELECT
    1,
    1,
    NOW()::date,
    NOW()::date + INTERVAL '1 month',
    'активный';

UPDATE subscription
SET
    status = 'неактивный',
    date_real_return = NOW()::date
WHERE subscription_id = (SELECT MAX(subscription_id) from subscription);

DELETE FROM subscription
WHERE subscription_id = (SELECT MAX(subscription_id) from subscription);
```

Данные в таблице логов:

	id	operation	operation_timestamp	subscription_id	employee_id	copy_id	date_subscribed	date_return	date_real_return	status
1	1	INSERT	2023-05-24 18:32:30.311298	36	1	1	2023-05-24	2023-06-24	<null>	активный
2	2	UPDATE	2023-05-24 18:32:30.311298	36	1	1	2023-05-24	2023-06-24	2023-05-25	неактивный
3	3	DELETE	2023-05-24 18:32:30.311298	36	1	1	2023-05-24	2023-06-24	2023-05-25	неактивный

Заключение:

В процессе выполнения данной лабораторной работы мы подробно изучили несколько ключевых аспектов работы с PostgreSQL, а именно: создание функций, процедур и триггерных функций. Эти элементы играют важную роль в обработке данных, и их важность не может быть недооценена.

Прежде всего, были разработаны функции для выполнения определенных задач, таких как проверка наличия книги в библиотеке и добавление нового читателя. Эти функции демонстрируют, как можно использовать SQL для создания повторно используемых процессов, обеспечивающих точное выполнение задач и облегчающих поддержку кода.

Затем, мы создали процедуру для ввода новой книги в базу данных. Процедуры в PostgreSQL позволяют объединить несколько операций SQL в единый логический блок, увеличивая эффективность и снижая вероятность ошибок.

Наконец, мы занялись созданием триггерной функции и использовали ее для создания триггера, который отслеживает события модификации в определенной таблице и регистрирует их в логе. Этот процесс демонстрирует способность PostgreSQL автоматически реагировать на изменения в данных, что является ценным инструментом для обеспечения целостности данных и учета их истории.

В целом, эта лабораторная работа предоставила нам возможность применить теоретические знания о PostgreSQL на практике и улучшить наше понимание сложных аспектов работы с базами данных.