

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:

09.03.03 Мобильная и сетевая разработка

Проверила:

Говорова М.М.

Дата: «..» ... 2023 г.

Оценка _____

Выполнила:

студент группы К32392

Жигалова А. Е.

Санкт-Петербург 2022/2023

Практическое задание

Задание 4. Создать хранимые процедуры:

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

Задание 5. Создать необходимые триггеры.

Выполнение

База Данных: library_bd


Создание хранимых процедур

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

```
CREATE OR REPLACE FUNCTION count_available_books(  
    IN book_title VARCHAR(100),  
    IN book_author VARCHAR(80)  
) RETURNS INTEGER  
AS $$  
DECLARE  
    all_books INTEGER;  
    unavailable_books INTEGER;  
    available_books INTEGER;  
BEGIN  
    SELECT COUNT(*) INTO all_books  
    FROM instance i  
        INNER JOIN book b ON b.id_book = i.id_book  
    WHERE b.title = book_title AND b.author = book_author;  
  
    SELECT COUNT(*) INTO unavailable_books  
    FROM instance i  
        INNER JOIN book b ON b.id_book = i.id_book  
        INNER JOIN issuance_books ib ON ib.id_instance = i.id_instance  
    WHERE b.title = book_title AND b.author = book_author AND  
    ib.date_of_return IS NULL;  
  
    available_books := all_books - unavailable_books;  
    RETURN available_books;  
END  
$$ LANGUAGE plpgsql;
```

Вывод таблицы:

```
SELECT * From count_available_books('C# 7 и .NET Core. Кросс-платформенная  
разработка для профессионалов', 'Прайс Марк Дж.');
```

	count_available_books 
	integer
1	1

```
SELECT * FROM count_available_books('Собачье Сердце', 'Михаил Булгаков');
```

	count_available_books
	integer
1	0

Для ввода в базу данных новой книги.

```
CREATE OR REPLACE FUNCTION add_book(
    IN book_title VARCHAR(100),
    IN book_author VARCHAR(80),
    IN book_year_of_publication INT,
    IN book_language VARCHAR(15),
    IN book_id_publishing INT,
    IN book_area_of_knowledge VARCHAR(80),
    IN book_type_of_publication VARCHAR(20),
    IN book_number_of_instances INT,
    IN instance_location INT,
    IN instance_cost NUMERIC(10,2),
    IN book_description VARCHAR(250) = NULL
) RETURNS INTEGER
AS $$
DECLARE
    book_id INTEGER;
BEGIN
    INSERT INTO book (title, author, year_of_publication, language,
id_publishing, area_of_knowledge, type_of_publication, number_of_instances,
description)
        VALUES (book_title, book_author, book_year_of_publication, book_language,
book_id_publishing, book_area_of_knowledge, book_type_of_publication,
book_number_of_instances, book_description)
        RETURNING id_book INTO book_id;

    IF book_number_of_instances > 0 THEN
        INSERT INTO instance (id_book, id_location, cost_of_book)
        SELECT book_id, instance_location, instance_cost
        FROM generate_series(1, book_number_of_instances);
    END IF;

    RETURN book_id;
END
$$ LANGUAGE plpgsql;
```

Вывод таблицы:

```
SELECT add_book(
    'Мартин Иден',
    'Джек Лондон',
    1892,
    'English',
    1,
    'Mystery and Detective Fiction',
    'Fiction',
    3,
    1,
    700
);
```

add_book ▾
85383

```
SELECT * FROM book WHERE id_book = 85383;
```

	id_book [PK] integer	title character varying (100)	author character varying (80)	year_of_publication integer	language character varying (15)	description character varying (250)	id_publishing integer	area_of_knowledge character varying (80)
1	85383	Мартин Иден	Джек Лондон	1892	English	[null]	1	Mystery and Detective Fic

SELECT * FROM instance WHERE id_book = 85383;

	id_instance [PK] integer	id_book integer	id_location integer	cost_of_book numeric (6,2)
1	25	85383	1	700.00
2	26	85383	1	700.00
3	27	85383	1	700.00

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


```
CREATE OR REPLACE FUNCTION add_reader(
    IN reader_number_ticket VARCHAR(10),
    IN reader_name VARCHAR(60),
    IN reader_date_registration DATE,
    IN reader_passport_data VARCHAR(20),
    IN reader_address VARCHAR(100),
    IN reader_phone_number VARCHAR(20) = NULL,
    IN reader_email VARCHAR(50) = NULL
) RETURNS INTEGER
AS $$
DECLARE
    ticket INTEGER;
BEGIN
    SELECT number_ticket INTO ticket
        FROM form
        WHERE number_ticket = reader_number_ticket;
    IF ticket IS NOT NULL THEN
        RAISE EXCEPTION 'Reader with number ticket % already exists!',
reader_number_ticket;
    END IF;

    INSERT INTO form (number_ticket, reader, date_registration,
passport_data, address, phone_number, email)
    VALUES (reader_number_ticket, reader_name, reader_date_registration,
reader_passport_data, reader_address, reader_phone_number, reader_email)
    RETURNING number_ticket INTO ticket;

    RETURN ticket;
END
$$ LANGUAGE plpgsql;
```

Вывод таблицы:

```
SELECT add_reader(
    '1234567890',
    'John Smith',
    '2021-11-01',
    'AB123456',
    '123 Main St, Anytown, USA',
    '+1-555-555-5555'
);
```

	add_reader integer 
1	1234567890

Создание триггеров

Триггер, проверяющий, что в таблицу instance нельзя будет добавить большее количество экземпляров, чем указано для книги в таблице book

```
CREATE OR REPLACE FUNCTION check_number_of_instances() RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM book AS b
            INNER JOIN instance AS i ON i.id_book = b.id_book
        WHERE i.id_book = NEW.id_book
        GROUP BY b.id_book, b.number_of_instances
        HAVING COUNT(*) >= b.number_of_instances
    ) THEN
        RAISE EXCEPTION 'Cannot add instance, the maximum number of instances
for the book has already been reached';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER check_number_of_instances_trigger
BEFORE INSERT ON instance
FOR EACH ROW
EXECUTE FUNCTION check_number_of_instances();
```

Вывод таблицы:

При повторном добавлении книги мы получим ошибку.

```
INSERT INTO instance(id_book, id_location, cost_of_book) VALUES (1, 1, 1111);
```

```
ERROR: ОШИБКА: Cannot add instance, the maximum number of instances for the book has already been reached
CONTEXT: функция PL/pgSQL check_number_of_instances(), строка 11, оператор RAISE
```

Триггер, запрещающий добавление книги уже находящейся в таблице book (совпадают названия и автор книги)

```
CREATE OR REPLACE FUNCTION check_existing_book() RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM book
        WHERE title = NEW.title AND author = NEW.author
    ) THEN
        RAISE EXCEPTION 'The book is already in the library';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE TRIGGER check_existing_book_trigger
    BEFORE INSERT ON book
    FOR EACH ROW
EXECUTE FUNCTION check_existing_book();
```

Вывод таблицы:

```
INSERT INTO book(title, author, year_of_publication, language, id_publishing,
area_of_knowledge, type_of_publication, number_of_instances)
VALUES ('книга1', 'автор1', 123, '123', 1, '123', '123', 3);
```

```
ERROR: ОШИБКА: The book is already in the library
CONTEXT: функция PL/pgSQL check_existing_book(), строка 8, оператор RAISE
```

Триггер, запрещающий выдавать экземпляр книги, который еще не был возвращен

```
CREATE OR REPLACE FUNCTION check_returned_instance() RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM issuance_books
        WHERE id_instance = NEW.id_instance AND
issuance_books.date_of_return IS NULL
    ) THEN
        RAISE EXCEPTION 'This instance cannot be issued, it has not been
returned to the library';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER check_returned_instance_trigger
    BEFORE INSERT ON issuance_books
    FOR EACH ROW
EXECUTE FUNCTION check_returned_instance();
```

Вывод таблицы:

```
INSERT INTO issuance_books(number_ticket, id_instance, id_employee,
date_of_issue)
VALUES ('1-10', 1, 1, '12-12-1212');
```

```
ERROR: ОШИБКА: This instance cannot be issued, it has not been returned to the library
CONTEXT: функция PL/pgSQL check_returned_instance(), строка 8, оператор RAISE
```

Вывод

В ходе выполнения данной работы были приобретены практические навыки создания хранимых процедур и триггеров в базе данных PostgreSQL. Были созданы триггеры и процедуры согласно заданию для базы данных библиотека, упрощающие работу с ней.