

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №5 «ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В PostgreSQL»

по дисциплине «Проектирование и реализация баз данных»

Автор: Морозов Артём

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Цель работы

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Практическое задание:

Вариант 1 (max - 6 баллов)

1. Создать процедуры/функции 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 (max - 8 баллов)

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

Указание. Работа выполняется в консоли SQL Shell (psql).

Вариант 3. БД «Библиотека»

Описание предметной области: Каждая книга может храниться в нескольких экземплярах. Для каждого экземпляра известно место его хранения (комната, стеллаж, полка). Читателю не может быть выдано более 3-х книг одновременно. Книги выдаются читателям на срок не более 10 дней. В случае просрочки читателю назначается денежный штраф.

Все издания, поступающие в библиотеку ставятся на библиотечный учет, согласно существующим требованиям. Необходимо хранить информацию, кто из сотрудников поставил экземпляр на учет.

Книги принимаются к учету на основании первичных учетных документов (накладной от поставщика, акта о приеме документов). Если документы поступают на безвозмездной основе (в результате передачи обязательных экземпляров и т. п.), оформляется акт о приеме документов. Документы, поступающие от читателей взамен утерянных и признанные равноценными утраченным, оформляются актом о приеме документов взамен утерянных.

Выбытие документов из библиотеки отражается в учете в связи с физической утратой либо утратой потребительских свойств (по причине ветхости, дефектности, устарелости по содержанию, непрофильности). Непрофильность издания определяется на основании профиля комплектования фонда или иного документа, утверждаемого руководителем библиотеки. При выбытии документов из библиотеки оформляется акт о списании исключенных объектов библиотечного фонда (далее – акт о списании), к которому прилагается список исключаемых объектов библиотечного фонда. В акте о списании отражаются сведения о количестве и общей стоимости исключаемых документов, а также причина списания и направление изданий после выбытия с учета. В прилагаемом к акту списке указываются:

- регистрационный номер и шифр хранения издания;
- краткое библиографическое описание;
- стоимость, зафиксированная в регистре индивидуального учета издания;

- коэффициент переоценки, стоимость после переоценки;
- общая стоимость исключаемых документов.

БД должна содержать следующий минимальный набор сведений: · Автор (фамилия и имя (инициалы) или псевдоним автора издания). · Название (заглавие) издания. · Номер тома (части, книги, выпуска). · Составитель (фамилия и имена (инициалы) каждого из составителей издания). · Язык, с которого выполнен перевод издания. · Вид издания (сборник, справочник, монография ...). · Область знания. · Переводчик (фамилия и инициалы переводчика). · Место издания (город). · Издательство (название издательства). · Год выпуска издания. · Библиотечный шифр (например, ББК 32.973). · Номер (инвентарный номер) экземпляра. · Номер комнаты (помещения для хранения экземпляров). · Номер стеллажа в комнате. · Номер полки на стеллаже. · Цена конкретного экземпляра. · Дата изъятия экземпляра с установленного места. · Номер читательского билета (формуляра). · Фамилия читателя. · Имя читателя. · Отчество читателя. · Паспортные данные. Адрес читателя (фактический). Телефон читателя. Электронная почта читателя.

Дополнить исходные данные информацией о читательском абонементе (выдаче книг).

Задание 1.1 (ЛР 1 БД). Выполните инфологическое моделирование базы данных системы. (Ограничения задать самостоятельно.)

Задание 1.2. Создайте логическую модель БД, используя ИЛМ (задание 1.1). Используйте необходимые средства поддержки целостности данных в СУБД.

Задание 2. Создайте запросы:

- Вывести список читателей, имеющих на руках книги, переведенные с английского языка, изданные позднее 2000 года.
- Вывести список читателей, не вернувших в срок книги и имеющих на руках более десяти книг.
- Найти количество читателей, не вернувших в срок книги и имеющих на руках более десяти книг.
- Вывести список книг, которые находятся в библиотеке в единственном экземпляре.
- Подсчитать количество читателей, которые не обращались в библиотеку в течение года.
- Подсчитать количество читателей библиотеки по уровню образования.
- Вывести список книг по программированию на C#, экземпляры которых отсутствуют в библиотеке, и которые должны быть возвращены не позднее, чем через 3 дня.

Задание 3. Создать представления для администрации библиотеки, содержащие:

- сведения о должниках;
- сведения о наиболее популярных книгах (все экземпляры находятся на руках у читателей).

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.

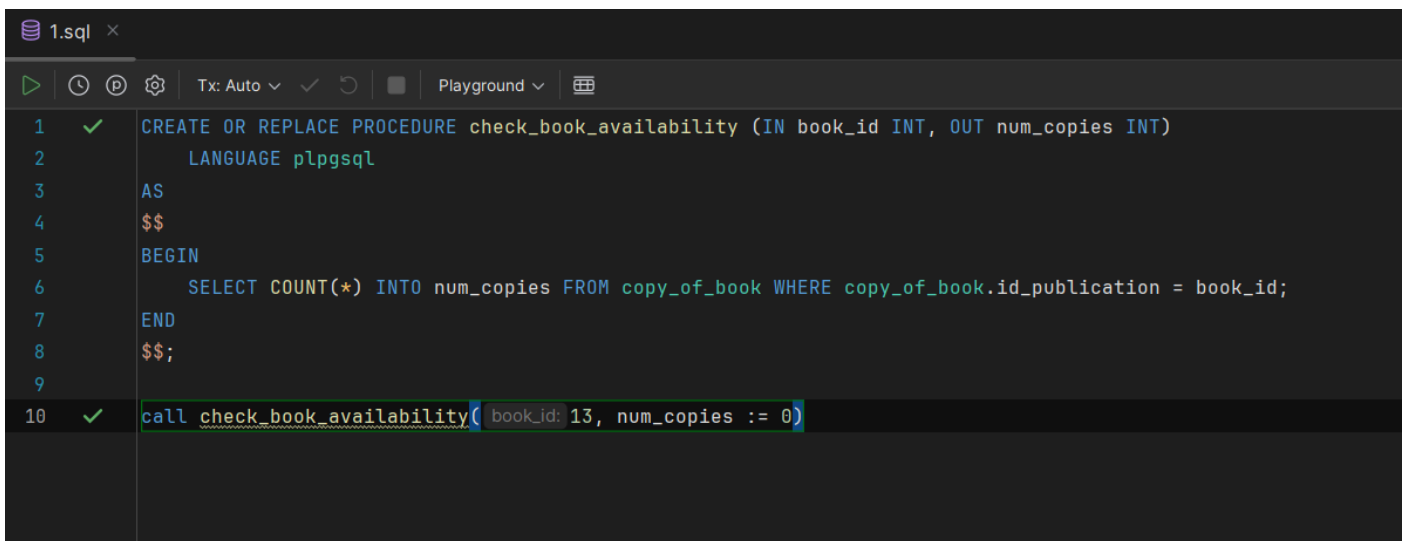
Выполнение практического задания

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

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

```
CREATE OR REPLACE PROCEDURE check_book_availability (IN book_id INT, OUT num_copies
INT)
    LANGUAGE plpgsql
AS
$$
BEGIN
    SELECT COUNT(*) INTO num_copies FROM copy_of_book WHERE
copy_of_book.id_publication = book_id;
END
$$;

call check_book_availability(13, num_copies := 0)
```



The screenshot shows a SQL playground window titled '1.sql'. The interface includes a toolbar with icons for running, undo, redo, and settings, along with a 'Tx: Auto' dropdown and a 'Playground' dropdown. The SQL code is displayed in a dark-themed editor with line numbers 1 through 10 on the left. Lines 1-9 define the 'check_book_availability' procedure, and line 10 calls it with 'book_id: 13, num_copies := 0'. Green checkmarks are visible next to lines 1 and 10, indicating successful execution. The code is as follows:

```
1 CREATE OR REPLACE PROCEDURE check_book_availability (IN book_id INT, OUT num_copies INT)
2     LANGUAGE plpgsql
3 AS
4 $$
5 BEGIN
6     SELECT COUNT(*) INTO num_copies FROM copy_of_book WHERE copy_of_book.id_publication = book_id;
7 END
8 $$;
9
10 call check_book_availability( book_id: 13, num_copies := 0)
```

2. Для ввода в базу данных новой книги.

```
CREATE OR REPLACE PROCEDURE add_new_book (IN library_code INT, IN
name_of_publishing_house VARCHAR, IN year_of_release INT, IN id_book INT, IN language
VARCHAR)
    LANGUAGE plpgsql
AS
$$
DECLARE
    id INT;
BEGIN
    SELECT id_publishing_house INTO id FROM publishing_house WHERE
publishing_house.name = name_of_publishing_house;
    INSERT INTO publication VALUES (library_code, name_of_publishing_house,
year_of_release, id_book, language);
END;
$$;
```

The screenshot shows a SQL IDE with two tabs: '1.sql' and '2.sql'. The '2.sql' tab is active and contains the following PL/SQL code:

```
1 CREATE OR REPLACE PROCEDURE add_new_book (IN library_code INT, IN name_of_publishing_house VARCHAR, IN year_of_release INT, IN id_book INT, IN language VARCHAR)
2 LANGUAGE plpgsql
3
4 AS
5 $$
6 DECLARE
7     id INT;
8 BEGIN
9     SELECT id_publishing_house INTO id FROM publishing_house WHERE publishing_house.name = name_of_publishing_house;
10    INSERT INTO publication VALUES (library_code, name_of_publishing_house, year_of_release, id_book, language);
11 END;
12 $$;
```

Below the code editor, there is a console window showing the execution results:

```
AS
$$
DECLARE
    id INT;
BEGIN
    SELECT id_publishing_house INTO id FROM publishing_house WHERE publishing_house.name = name_of_publishing_house;
    INSERT INTO publication VALUES (library_code, name_of_publishing_house, year_of_release, id_book, language);
END;
$$
[2024-01-18 12:23:18] completed in 14 ms
```

The status bar at the bottom indicates '11:4 CRLF UTF-8 4 spa' and '12:23'.

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

```
4. CREATE OR REPLACE PROCEDURE add_new_reader
(
    IN reader_name VARCHAR,
    IN reader_surname VARCHAR,
    IN reader_patronymic VARCHAR,
    IN reader_phone_number VARCHAR,
    IN reader_address VARCHAR,
    IN reader_passport VARCHAR,
    IN reader_email VARCHAR
)
LANGUAGE plpgsql
AS
$$
DECLARE
    reader_exists BOOLEAN = false;
BEGIN
    SELECT true FROM reader WHERE reader.name = reader_name AND
reader.phone_number = reader_phone_number INTO reader_exists;

    IF NOT reader_exists THEN
        INSERT INTO reader (id_reader, count_of_books, name, surname, patronymic,
phone_number, address, passport, email)
VALUES (id_reader := 9999, count_of_books := 0, reader_name,
reader_surname, reader_patronymic, reader_phone_number, reader_address,
reader_passport, reader_email);
    ELSE
        RAISE NOTICE 'Читатель уже есть в БД';
    END IF;
END;
$$;
```

```
1 CREATE OR REPLACE PROCEDURE add_new_reader
2 (
3     IN reader_name VARCHAR,
4     IN reader_surname VARCHAR,
5     IN reader_patronymic VARCHAR,
6     IN reader_phone_number VARCHAR,
7     IN reader_address VARCHAR,
8     IN reader_passport VARCHAR,
9     IN reader_email VARCHAR
10 )
11 LANGUAGE plpgsql
12 AS
13 $$
14 DECLARE
15     reader_exists BOOLEAN;
16 BEGIN
17     SELECT true FROM reader WHERE reader.name = reader_name AND reader.phone_number = reader_phone_number INTO reader_exists;
18
19     IF NOT reader_exists THEN
20         INSERT INTO reader (id_reader, count_of_books, name, surname, patronymic, phone_number, address, passport, email)
21         VALUES (id_reader := 9999, count_of_books := 0, reader_name, reader_surname, reader_patronymic, reader_phone_number, reader_address, reader_passport, reader_email);
22     ELSE
23         RAISE NOTICE 'Читатель уже есть в БД';
24     END IF;
25 END;
26 $$;
```

Tx: Auto ✓ | Playground | Library: public

[2024-01-18 12:32:50] Connected
[2024-01-18 12:32:50] completed in 87 ms

Авторские триггеры

1. Триггер, который выставляет читателю штраф, если он просрочивает сдачу книг.

```
• CREATE OR REPLACE PROCEDURE fine_check(IN id INT)
  LANGUAGE plpgsql
AS
$$
BEGIN
    UPDATE book_circulation
    SET fine = 100
    WHERE book_circulation.number_of_operation = id
      AND book_circulation.return_date > book_circulation.return_date_plan;
END;
$$;

CREATE TRIGGER fine_checker
  AFTER UPDATE ON book_circulation
  FOR EACH ROW
EXECUTE PROCEDURE fine_check(NEW.number_of_operation);
```

```
1.sql 2.sql 3.sql Trigger1.sql x Trigger2.sql
Tx: Auto ✓ ↺ Playground
1 CREATE OR REPLACE PROCEDURE fine_check(IN id INT)
2   LANGUAGE plpgsql
3 AS
4 $$
5 BEGIN
6   UPDATE book_circulation
7   SET fine = 100
8   WHERE book_circulation.number_of_operation = id
9   AND book_circulation.return_date > book_circulation.return_date_plan;
10 END;
11 $$;
12
13
14 CREATE TRIGGER fine_checker
15   AFTER UPDATE ON book_circulation
16   FOR EACH ROW
17 EXECUTE PROCEDURE fine_check(id: NEW.number_of_operation);
```

Tx, ✓ ↺

```
[2024-01-18 14:05:34] Connected
[2024-01-18 14:05:34] completed in 1 s 194 ms
[2024-01-18 14:10:10] completed in 20 ms
```

2. Триггер, который рассчитывает планируемую дату возврата книги

```
3. CREATE OR REPLACE PROCEDURE plan_date(IN id INT)
  LANGUAGE plpgsql
AS
$$
BEGIN
  UPDATE book_circulation
  SET return_date_plan = issue_date + INTERVAL '20 days'
  WHERE book_circulation.number_of_operation = id;
END;
$$;

CREATE TRIGGER fine_checker
  AFTER UPDATE ON book_circulation
```

```
FOR EACH ROW  
EXECUTE PROCEDURE plan_date(NEW.number_of_operation);
```

1.sql2.sql3.sqlTrigger1.sqlTrigger2.sql ×

Tx: Auto ✓ ↺ Playground ▾

```
1 CREATE OR REPLACE PROCEDURE plan_date(IN id INT)
2   LANGUAGE plpgsql
3 AS
4 $$
5 BEGIN
6   UPDATE book_circulation
7   SET return_date_plan = issue_date + INTERVAL '20 days'
8   WHERE book_circulation.number_of_operation = id;
9 END;
10 $$;
11
12 CREATE TRIGGER fine_checker
13   AFTER UPDATE ON book_circulation
14   FOR EACH ROW
15   EXECUTE PROCEDURE plan_date(id: NEW.number_of_operation);
```

fine_checker

Tx ✓ ↺

[2024-01-18 14:05:34] Connected

[2024-01-18 14:05:34] completed in 1 s 194 ms

[2024-01-18 14:10:10] completed in 20 ms