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 POSTGERSQL»

по дисциплине «Проектирование и реализация баз данных»

Обучающийся Христофоров Владислав Николаевич
Факультет прикладной информатики
Группа K3240
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

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

Практическое задание:

1. Создать 3 процедуры для индивидуальной БД согласно варианту (часть 4 ЛР 2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры. (3 балла)
2. Создать триггеры для индивидуальной БД согласно варианту:

Вариант 2.1. 3 триггера - 3 балла (min). Допустимо использовать триггеры логирования из практического занятия по функциям и триггерам.

Вариант 2.2. 7 оригинальных триггеров - 7 баллов (max).

Индивидуальное задание: 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.

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.

1 СОЗДАНИЕ ХРАНИМЫХ ПРОЦЕДУР

1.1 Проверка наличия экземпляров

Создать хранимую процедуру для проверки наличия экземпляров заданной книги в библиотеке (процедура должна возвращать количество экземпляров книги).

Скрипт кода процедуры:

```
CREATE OR REPLACE PROCEDURE check_book_copy(book_id integer, OUT
copy_count integer)
LANGUAGE plpgsql
AS $$
BEGIN
     SELECT COUNT(*) INTO copy_count
     FROM library_schema.publication p
          JOIN library_schema.receipt rec ON p.publication_id =
rec.publication_id
          JOIN library_schema.copy c ON rec.receipt_id =
c.receipt_id
     WHERE p.publication_id = book_id;
END;
$$;
```

Создание процедуры и пример ее выполнения

```
library=# CREATE OR REPLACE PROCEDURE check_book_copy(book_id integer, OUT copy_count integer)
library-# LANGUAGE plpgsql
library-# AS $$
library$# BEGIN
library$# SELECT COUNT(*) INTO copy_count
library$# FROM library_schema.publication p
library$# JOIN library_schema.receipt rec ON p.publication_id = rec.publication_id
library$# JOIN library_schema.copy c ON rec.receipt_id = c.receipt_id
library$# WHERE p.publication_id = book_id;
library$# END;
library$# $$;
CREATE PROCEDURE
library=# CALL check_book_copy(2722, NULL);
 copy_count
------------
         20
(1 строка)
```

1.2 Ввод в БД новой книги

Создать хранимую процедуру для ввода в базу данных новой книги.

Скрипт кода процедуры:

```
CREATE OR REPLACE PROCEDURE insert_book(
     p_title             VARCHAR(255),
     p_volume_num        INT,
     p_original_lang     VARCHAR(15),
     p_publication_type  VARCHAR(50),
     p_knowledge_field   VARCHAR(50),
     p_publication_year  INT,
     p_page_num          INT,
     p_isbn              CHAR(13),
     p_publisher_id      INT,
     OUT new_book_id     INT,
     OUT out_status      TEXT
)
LANGUAGE plpgsql
AS $$
BEGIN
     INSERT INTO library_schema.publication
          (title, volume_num, original_lang, publication_type,
               knowledge_field, publication_year, page_num, isbn,
publisher_id)
     VALUES
          (p_title, p_volume_num, p_original_lang,
p_publication_type,
               p_knowledge_field, p_publication_year, p_page_num,
p_isbn, p_publisher_id)
     RETURNING publication_id INTO new_book_id;

     out_status := 'Книга добавлена';
END;
$$;
```

Создание процедуры

```
library=# CREATE OR REPLACE PROCEDURE insert_book(
library(# p_title             VARCHAR(255),
library(# p_volume_num        INT,
library(# p_original_lang     VARCHAR(15),
library(# p_publication_type  VARCHAR(50),
library(# p_knowledge_field   VARCHAR(50),
library(# p_publication_year  INT,
library(# p_page_num          INT,
library(# p_isbn              CHAR(13),
library(# p_publisher_id      INT,
library(# OUT new_book_id     INT,
library(# OUT out_status      TEXT
library(# )
library-# LANGUAGE plpgsql
library-# AS $$
library$# BEGIN
library$# INSERT INTO library_schema.publication
library$# (title, volume_num, original_lang, publication_type,
library$# knowledge_field, publication_year, page_num, isbn, publisher_id)
library$# VALUES
library$# (p_title, p_volume_num, p_original_lang, p_publication_type,
library$# p_knowledge_field, p_publication_year, p_page_num, p_isbn, p_publisher_id)
library$# RETURNING publication_id INTO new_book_id;
library$#
library$#     out_status := 'Книга добавлена';
library$# END;
library$# $$;
CREATE PROCEDURE
```

Пример выполнения

```
library=# CALL insert_book('Олонхо', '1', 'русский', 'сборник', 'искусство', 2025, 1000, '1122334455667', 157, NULL, NULL);
 new_book_id |   out_status
-------------+-----------------
        2746 | Книга добавлена
(1 строка)


library=# SELECT * FROM publication WHERE publication_id=2746;
 publication_id | title  | volume_num | original_lang | publication_type | knowledge_field | publication_year | page_num |     isbn      | publisher_id
----------------+--------+------------+---------------+------------------+-----------------+------------------+----------+---------------+--------------
           2746 | Олонхо |          1 | русский       | сборник          | искусство       |             2025 |     1000 | 1122334455667 |          157
(1 строка)
```

1.3 Ввод в БД нового читателя

Создать хранимую процедуру для ввода нового читателя (необходимо проверить наличие читателя в картотеке, чтобы не назначить ему номер вторично).

Скрипт кода процедуры:

```
CREATE OR REPLACE PROCEDURE insert_reader(
     p_form_num        VARCHAR(20),
     p_full_name       VARCHAR(80),
     p_passport        CHAR(10),
     p_address         VARCHAR(255),
     p_phone           CHAR(16),
     p_email           VARCHAR(50),
     OUT out_reader_id INT,
     OUT out_status    TEXT
)
LANGUAGE plpgsql
AS $$
BEGIN
     IF EXISTS (
          SELECT 1
          FROM library_schema.reader
          WHERE passport_details = p_passport
     ) THEN
          out_reader_id := NULL;
          out_status    := 'Читатель с таким паспортом уже есть';
          RETURN;
     END IF;

     IF EXISTS (
          SELECT 1
          FROM library_schema.reader
          WHERE full_name = p_full_name
               AND address   = p_address
     ) THEN
          out_reader_id := NULL;
```

```
          out_status    := 'Читатель с таким именем и адресом уже
есть';
          RETURN;
     END IF;

     INSERT INTO library_schema.reader
          (form_num, full_name, passport_details, address, phone,
email, registration_date, status)
     VALUES
          (p_form_num, p_full_name, p_passport, p_address, p_phone,
p_email, CURRENT_DATE, 'активен')
     RETURNING reader_id INTO out_reader_id;

     out_status := 'Читатель добавлен';
END;
$$;
```

Создание процедуры

```
library=# CREATE OR REPLACE PROCEDURE insert_reader(
library(# p_form_num        VARCHAR(20),
library(# p_full_name       VARCHAR(80),
library(# p_passport        CHAR(10),
library(# p_address         VARCHAR(255),
library(# p_phone           CHAR(16),
library(# p_email           VARCHAR(50),
library(# OUT out_reader_id INT,
library(# OUT out_status    TEXT
library(# )
library-# LANGUAGE plpgsql
library-# AS $$
library$# BEGIN
library$# IF EXISTS (
library$# SELECT 1
library$# FROM library_schema.reader
library$# WHERE passport_details = p_passport
library$# ) THEN
library$# out_reader_id := NULL;
library$# out_status    := 'Читатель с таким паспортом уже есть';
library$# RETURN;
library$# END IF;
library$#
library$# IF EXISTS (
library$# SELECT 1
library$# FROM library_schema.reader
library$# WHERE full_name = p_full_name
library$# AND address   = p_address
library$# ) THEN
library$# out_reader_id := NULL;
library$# out_status    := 'Читатель с таким именем и адресом уже есть';
library$# RETURN;
library$# END IF;
library$#
library$# INSERT INTO library_schema.reader
library$# (form_num, full_name, passport_details, address, phone, email, registration_date, status)
library$# VALUES
library$# (p_form_num, p_full_name, p_passport, p_address, p_phone, p_email, CURRENT_DATE, 'активен')
library$# RETURNING reader_id INTO out_reader_id;
library$#
library$# out_status := 'Читатель добавлен';
library$# END;
library$# $$;
CREATE PROCEDURE
```

Пример выполнения

```
library=# CALL insert_reader('F-1234', 'Лунтик', '1234567890', 'Луна', '+1111111111', 'luntik@moon.com', NULL, NULL);
 out_reader_id |    out_status
---------------+------------------
          3468 | Читатель добавлен
(1 строка)


library=# SELECT * FROM reader WHERE reader_id=3468;
 reader_id | form_num | registration_date | status  | full_name | passport_details | address |    phone    |      email
-----------+----------+-------------------+---------+-----------+------------------+---------+-------------+-----------------
      3468 | F-1234   | 2025-05-27        | активен | Лунтик    | 1234567890       | Луна    | +1111111111 | luntik@moon.com
(1 строка)
```

```
library=# CALL insert_reader('F-4321', 'Злой Лунтик', '1234567890', 'Темная сторона Луны', '+9999999999', 'zloy_luntik@moon.com', NULL, NULL);
 out_reader_id |            out_status
---------------+----------------------------------
               | Читатель с таким паспортом уже есть
(1 строка)
```

```
library=# CALL insert_reader('F-4321', 'Лунтик', '0987654321', 'Луна', '+1111111111', 'luntik@moon.com', NULL, NULL);
 out_reader_id |                out_status
---------------+-------------------------------------------
               | Читатель с таким именем и адресом уже есть
(1 строка)
```

2 СОЗДАНИЕ ТРИГГЕРОВ

2.1 Поддержка статуса экземпляра

2.1.1 Статус «выдан» при выдаче

Скрипт кода триггерной функции

```
CREATE OR REPLACE FUNCTION fn_set_copy_loaned()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE library_schema.copy
   SET status = 'выдан'
   WHERE copy_id = NEW.copy_id;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Скрипт кода создания триггера

```
CREATE TRIGGER trg_set_copy_loaned
   AFTER INSERT ON library_schema.loan
   FOR EACH ROW
   EXECUTE FUNCTION fn_set_copy_loaned();
```

```
library=# SELECT * FROM copy WHERE copy_id=27357;
 copy_id | inventory_num |   copy_condition    |  status  | receipt_id
---------+---------------+---------------------+----------+------------
   27357 | INV-CLONE-13  | удовлетворительный | доступен |      13127
(1 строка)
```

```
library=# INSERT INTO loan (loan_date, return_date, copy_id, reader_id, employee_id)
library-# VALUES (CURRENT_DATE, CURRENT_DATE + INTERVAL '10 day', 27357, 3193, 367);
INSERT 0 1
library=# SELECT * FROM copy WHERE copy_id=27357;
 copy_id | inventory_num |   copy_condition    | status | receipt_id
---------+---------------+---------------------+--------+------------
   27357 | INV-CLONE-13  | удовлетворительный | выдан  |      13127
(1 строка)
```

2.1.2 Статус «доступен» при возврате

Скрипт кода триггерной функции

```
CREATE OR REPLACE FUNCTION fn_set_copy_returned()
RETURNS TRIGGER AS $$
BEGIN
   IF OLD.actual_return_date IS NULL
     AND NEW.actual_return_date IS NOT NULL THEN
          UPDATE library_schema.copy
          SET status = 'доступен'
          WHERE copy_id = NEW.copy_id;
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Скрипт кода создания триггера

```
CREATE TRIGGER trg_set_copy_returned
   AFTER UPDATE OF actual_return_date ON library_schema.loan
   FOR EACH ROW
   EXECUTE FUNCTION fn_set_copy_returned();
```

```
library=# SELECT * FROM loan WHERE copy_id=27357;
 loan_id | loan_date  | return_date | actual_return_date | fine | copy_id | reader_id | employee_id
---------+------------+-------------+--------------------+------+---------+-----------+-------------
    5131 | 2025-05-28 | 2025-06-07  |                    | f    |   27357 |      3193 |         367
(1 строка)


library=# UPDATE loan SET actual_return_date=CURRENT_DATE
library-# WHERE loan_id=5131;
UPDATE 1
library=# SELECT * FROM copy WHERE copy_id=27357;
 copy_id | inventory_num |   copy_condition    |  status  | receipt_id
---------+---------------+---------------------+----------+------------
   27357 | INV-CLONE-13  | удовлетворительный | доступен |      13127
(1 строка)
```

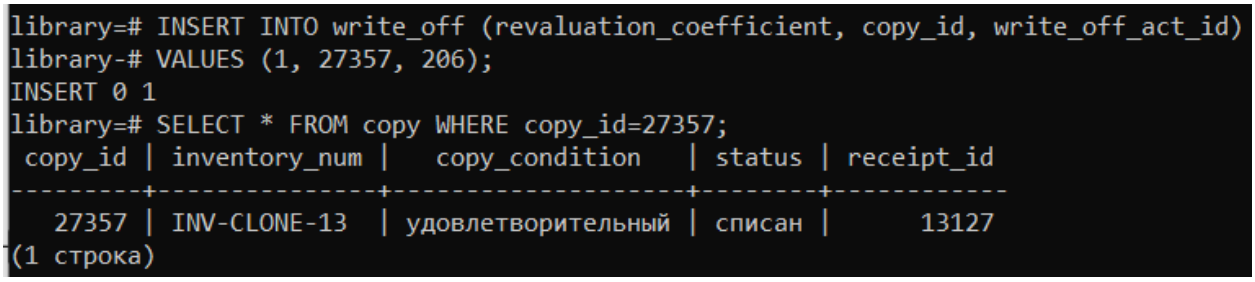
2.1.3 Статус «списан» при списании

Скрипт кода триггерной функции

```
CREATE OR REPLACE FUNCTION fn_set_copy_written_off()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE library_schema.copy
   SET status = 'списан'
   WHERE copy_id = NEW.copy_id;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Скрипт кода создания триггера

```
CREATE TRIGGER trg_set_copy_written_off
   AFTER INSERT ON library_schema.write_off
   FOR EACH ROW
   EXECUTE FUNCTION fn_set_copy_written_off();
```



2.2 Контроль выдач и штрафов

2.2.1 Ограничение на количество одновременных выдач читателю (не более 3 книг)

Скрипт кода триггерной функции

```
CREATE OR REPLACE FUNCTION fn_loan_limit()
RETURNS TRIGGER AS $$
DECLARE cnt INT;
BEGIN
   SELECT COUNT(*) INTO cnt
   FROM library_schema.loan
   WHERE reader_id = NEW.reader_id
     AND actual_return_date IS NULL;
   IF cnt >= 3 THEN
     RAISE EXCEPTION 'Читателю % уже выдано 3 книги одновременно', NEW.reader_id;
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Скрипт кода создания триггера

```
CREATE TRIGGER trg_loan_limit
   BEFORE INSERT ON library_schema.loan
   FOR EACH ROW
   EXECUTE FUNCTION fn_loan_limit();
```

```
library=# INSERT INTO loan (return_date, copy_id, reader_id, employee_id)
library-# VALUES (CURRENT_DATE + INTERVAL '10 day', 27353, 3193, 367),
library-# (CURRENT_DATE + INTERVAL '10 day', 27362, 3193, 367),
library-# (CURRENT_DATE + INTERVAL '10 day', 27375, 3193, 367);
INSERT 0 3
```

```
library=# INSERT INTO loan (return_date, copy_id, reader_id, employee_id)
library-# VALUES (CURRENT_DATE + INTERVAL '10 day', 27400, 3193, 367);
ОШИБКА:  Читателю 3193 уже выдано 3 книги одновременно
КОНТЕКСТ:  функция PL/pgSQL fn_loan_limit(), строка 10, оператор RAISE
```

```
library=# SELECT * FROM loan WHERE reader_id=3193 ORDER BY loan_id DESC;
 loan_id | loan_date  | return_date | actual_return_date | fine | copy_id | reader_id | employee_id
---------+------------+-------------+--------------------+------+---------+-----------+-------------
    5134 | 2025-05-28 | 2025-06-07  |                    | f    |   27375 |      3193 |         367
    5133 | 2025-05-28 | 2025-06-07  |                    | f    |   27362 |      3193 |         367
    5132 | 2025-05-28 | 2025-06-07  |                    | f    |   27353 |      3193 |         367
    5131 | 2025-05-28 | 2025-06-07  | 2025-05-28         | f    |   27357 |      3193 |         367
    4539 | 2025-05-01 | 2025-05-11  | 2025-05-02         | f    |   27090 |      3193 |         379
    4538 | 2025-05-01 | 2025-05-11  | 2025-05-02         | f    |   27091 |      3193 |         367
    4537 | 2025-05-01 | 2025-05-11  | 2025-05-02         | f    |   27093 |      3193 |         385
(7 строк)
```

2.2.2 Выставление штрафа при просрочке возврата

Скрипт кода триггерной функции

```
CREATE OR REPLACE FUNCTION fn_set_fine()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW.actual_return_date IS NOT NULL
     AND NEW.actual_return_date > NEW.return_date THEN
     NEW.fine := TRUE;
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Скрипт кода создания триггера

```
CREATE TRIGGER trg_set_fine
   BEFORE UPDATE OF actual_return_date ON library_schema.loan
   FOR EACH ROW
   EXECUTE FUNCTION fn_set_fine();
```

```
library=# SELECT * FROM loan WHERE actual_return_date IS NULL AND return_date < CURRENT_DATE ORDER BY loan_id LIMIT 1;
 loan_id | loan_date  | return_date | actual_return_date | fine | copy_id | reader_id | employee_id
---------+------------+-------------+--------------------+------+---------+-----------+-------------
    4771 | 2025-04-17 | 2025-04-27  |                    | f    |   26624 |      3255 |         376
(1 строка)


library=# UPDATE loan SET actual_return_date=CURRENT_DATE WHERE loan_id=4771;
UPDATE 1
library=# SELECT * FROM loan WHERE loan_id=4771;
 loan_id | loan_date  | return_date | actual_return_date | fine | copy_id | reader_id | employee_id
---------+------------+-------------+--------------------+------+---------+-----------+-------------
    4771 | 2025-04-17 | 2025-04-27  | 2025-05-28         | t    |   26624 |      3255 |         376
(1 строка)
```

2.3 Управление статусом читателя

2.3.1 Проверка статуса читателя перед выдачей и реактивация

Скрипт кода триггерной функции

```
CREATE OR REPLACE FUNCTION fn_loan_reader_status()
RETURNS TRIGGER AS $$
DECLARE cur_status TEXT;
BEGIN
   SELECT status INTO cur_status
   FROM library_schema.reader
   WHERE reader_id = NEW.reader_id;

   IF cur_status IN ('заблокирован','приостановлен') THEN
     RAISE EXCEPTION 'Читателю % выдача запрещена (status=%)',
NEW.reader_id, cur_status;
   ELSIF cur_status = 'неактивен' THEN
     UPDATE library_schema.reader
     SET status = 'активен'
     WHERE reader_id = NEW.reader_id;
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Скрипт кода создания триггера

```
CREATE TRIGGER trg_loan_reader_status
   BEFORE INSERT ON library_schema.loan
   FOR EACH ROW
   EXECUTE FUNCTION fn_loan_reader_status();
```

```
library=# SELECT * FROM reader WHERE status='неактивен' LIMIT 1;
 reader_id | form_num | registration_date |  status   |         full_name          | pas
-----------+----------+-------------------+-----------+----------------------------+----

      3166 | F-0001   | 2024-10-06        | неактивен | Щербаков Ладимир Андреевна | 068
(1 строка)


library=# INSERT INTO loan (return_date, copy_id, reader_id, employee_id)
library-# VALUES (CURRENT_DATE + INTERVAL '10 day', 27400, 3166, 367);
INSERT 0 1
library=# SELECT * FROM reader WHERE reader_id=3166;
 reader_id | form_num | registration_date | status  |         full_name          | passp
-----------+----------+-------------------+---------+----------------------------+------
      3166 | F-0001   | 2024-10-06        | активен | Щербаков Ладимир Андреевна | 06826
(1 строка)
```

```
library=# SELECT * FROM reader WHERE status='заблокирован' ORDER BY reader_id DESC LIMIT 1;
 reader_id | form_num | registration_date |    status    |         full_name          | passpo
hone     |           email
-----------+----------+-------------------+--------------+----------------------------+-------
---------+--------------------------
      3463 | F-0298   | 2021-11-23        | заблокирован | Кулакова Аникита Аксёнович | 456821
05121221 | viktorin_1971@example.com
(1 строка)


library=# INSERT INTO loan (return_date, copy_id, reader_id, employee_id)
library-# VALUES (CURRENT_DATE + INTERVAL '10 day', 27378, 3463, 367);
ОШИБКА:  Читателю 3463 выдача запрещена (status=заблокирован)
КОНТЕКСТ:  функция PL/pgSQL fn_loan_reader_status(), строка 9, оператор RAISE
```

2.4 Закрытие предыдущего хранения

2.4.1 Закрытие старого хранения при перемещении экземпляра на новое место хранения.

Скрипт кода триггерной функции

```
CREATE OR REPLACE FUNCTION fn_close_prev_storage()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE library_schema.storage
   SET end_date = NEW.start_date
   WHERE copy_id = NEW.copy_id
     AND end_date = '9999-12-31'
     AND storage_id <> NEW.storage_id;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Скрипт кода создания триггера

```
CREATE TRIGGER trg_close_prev_storage
   BEFORE INSERT ON library_schema.storage
   FOR EACH ROW
   EXECUTE FUNCTION fn_close_prev_storage();
```

```
library=# SELECT * FROM storage WHERE end_date='9999-12-31' LIMIT 1;
 storage_id | start_date |  end_date  | storage_location_id | copy_id
------------+------------+------------+---------------------+---------
       9980 | 2024-05-16 | 9999-12-31 |                 213 |   25109
(1 строка)


library=# INSERT INTO storage (start_date, storage_location_id, copy_id)
library-# VALUES (CURRENT_DATE, 214, 25109);
INSERT 0 1
library=# SELECT * FROM storage WHERE copy_id=25109 ORDER BY start_date DESC LIMIT 2;
 storage_id | start_date |  end_date  | storage_location_id | copy_id
------------+------------+------------+---------------------+---------
     112002 | 2025-05-28 | 9999-12-31 |                 214 |   25109
       9980 | 2024-05-16 | 2025-05-28 |                 213 |   25109
(2 строки)
```

2.5 Поддержка итоговых сумм в документах

2.5.1 Пересчет сумм в документе поступления при изменениях в поступлении.

Скрипт кода триггерной функции

```
CREATE OR REPLACE FUNCTION fn_update_receipt_doc_totals()
RETURNS TRIGGER AS $$
DECLARE doc_id INT;
BEGIN
    doc_id := COALESCE(NEW.receipt_document_id,
OLD.receipt_document_id);
    UPDATE library_schema.receipt_document rd
    SET quantity_total = sub.qty,
      cost_total = sub.cost
    FROM (
      SELECT
           receipt_document_id,
           SUM(quantity) AS qty,
           SUM(quantity * unit_price) AS cost
      FROM library_schema.receipt
      WHERE receipt_document_id = doc_id
      GROUP BY receipt_document_id
    ) AS sub
    WHERE rd.receipt_document_id = sub.receipt_document_id;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

Скрипт кода создания триггера

```
CREATE TRIGGER trg_update_receipt_doc_totals
    AFTER INSERT OR UPDATE OR DELETE ON library_schema.receipt
    FOR EACH ROW
    EXECUTE FUNCTION fn_update_receipt_doc_totals();
```

INSERT

```
library=# SELECT * FROM receipt_document ORDER BY receipt_document_id DESC LIMIT 1;
 receipt_document_id | document_num | document_date | document_type | receipt_source | quantity_total | cost_total | receipt_date | employee_id | supplier_id
---------------------+--------------+---------------+---------------+----------------+----------------+------------+--------------+-------------+-------------
                2423 | DOC-00200    | 2025-05-28    | накладная     | ИП «Федотова»  |             30 |    5885.00 | 2025-05-28   |         367 |         301
(1 строка)


library=# INSERT INTO receipt (quantity, unit_price, publication_id, receipt_document_id)
library-# VALUES (10, 100.00, 2721, 2423);
INSERT 0 1
library=# SELECT * FROM receipt_document WHERE receipt_document_id=2423;
 receipt_document_id | document_num | document_date | document_type | receipt_source | quantity_total | cost_total | receipt_date | employee_id | supplier_id
---------------------+--------------+---------------+---------------+----------------+----------------+------------+--------------+-------------+-------------
                2423 | DOC-00200    | 2025-05-28    | накладная     | ИП «Федотова»  |             40 |    6885.00 | 2025-05-28   |         367 |         301
(1 строка)
```

UPDATE

```
library=# UPDATE receipt SET quantity=20 WHERE receipt_id=13801;
UPDATE 1
library=# SELECT * FROM receipt_document WHERE receipt_document_id=2423;
 receipt_document_id | document_num | document_date | document_type | receipt_source | quantity_total | cost_total | receipt_date | employee_id | supplier_id
---------------------+--------------+---------------+---------------+----------------+----------------+------------+--------------+-------------+-------------
                2423 | DOC-00200    | 2025-05-28    | накладная     | ИП «Федотова»  |             50 |    7885.00 | 2025-05-28   |         367 |         301
(1 строка)
```

DELETE

```
library=# DELETE FROM receipt WHERE receipt_id=13801;
DELETE 1
library=# SELECT * FROM receipt_document WHERE receipt_document_id=2423;
 receipt_document_id | document_num | document_date | document_type | receipt_source | quantity_total | cost_total | receipt_date | employee_id | supplier_id
---------------------+--------------+---------------+---------------+----------------+----------------+------------+--------------+-------------+-------------
                2423 | DOC-00200    | 2025-05-28    | накладная     | ИП «Федотова»  |             30 |    5885.00 | 2025-05-28   |         367 |         301
(1 строка)
```

2.5.2 Пересчет сумм в акте списания при изменениях в списании.

Скрипт кода триггерной функции

```
CREATE OR REPLACE FUNCTION fn_update_writeoff_act_totals()
RETURNS TRIGGER AS $$
DECLARE act_id INT;
BEGIN
    act_id := COALESCE(NEW.write_off_act_id, OLD.write_off_act_id);
    UPDATE library_schema.write_off_act wa
    SET quantity_total = sub.qty,
      cost_total = sub.cost
    FROM (
      SELECT
           wo.write_off_act_id,
           COUNT(*) AS qty,
           SUM(rec.unit_price * wo.revaluation_coefficient) AS cost
      FROM library_schema.write_off AS wo
           JOIN library_schema.copy AS c ON wo.copy_id = c.copy_id
           JOIN library_schema.receipt AS rec ON c.receipt_id =
rec.receipt_id
      WHERE wo.write_off_act_id = act_id
      GROUP BY wo.write_off_act_id
```

```
    ) AS sub
    WHERE wa.write_off_act_id = sub.write_off_act_id;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

Скрипт кода создания триггера

```
CREATE TRIGGER trg_update_writeoff_act_totals
    AFTER INSERT OR UPDATE OR DELETE ON library_schema.write_off
    FOR EACH ROW
    EXECUTE FUNCTION fn_update_writeoff_act_totals();
```

INSERT

```
library=# SELECT * FROM write_off_act ORDER BY write_off_act_id DESC LIMIT 1;
 write_off_act_id | act_num |  act_date  | quantity_total | cost_total |  reason  | post_direction  | employee_id
------------------+---------+------------+----------------+------------+----------+-----------------+-------------
              256 | WO-0050 | 2025-05-28 |              3 |     709.47 | ветхость | Городской архив |         380
(1 строка)


library=# SELECT c.copy_id, rec.unit_price FROM copy c
library-# JOIN receipt rec ON c.receipt_id=rec.receipt_id
library-# WHERE c.copy_condition='ветхий' AND c.status='доступен'
library-# ORDER BY c.copy_id DESC LIMIT 1;
 copy_id | unit_price
---------+------------
  127323 |     472.98
(1 строка)


library=# INSERT INTO write_off (revaluation_coefficient, copy_id, write_off_act_id)
library-# VALUES (0.1, 127323, 256);
INSERT 0 1
library=# SELECT * FROM write_off_act WHERE write_off_act_id=256;
 write_off_act_id | act_num |  act_date  | quantity_total | cost_total |  reason  | post_direction  | employee_id
------------------+---------+------------+----------------+------------+----------+-----------------+-------------
              256 | WO-0050 | 2025-05-28 |              4 |     756.77 | ветхость | Городской архив |         380
(1 строка)
```

UPDATE

```
library=# UPDATE write_off SET revaluation_coefficient=1 WHERE copy_id=127323 AND write_off_act_id=256;
UPDATE 1
library=# SELECT * FROM write_off_act WHERE write_off_act_id=256;
 write_off_act_id | act_num |  act_date  | quantity_total | cost_total |  reason  | post_direction  | employee_id
------------------+---------+------------+----------------+------------+----------+-----------------+-------------
              256 | WO-0050 | 2025-05-28 |              4 |    1182.45 | ветхость | Городской архив |         380
(1 строка)
```

DELETE

```
library=# DELETE FROM write_off WHERE copy_id=127323 AND write_off_act_id=256;
DELETE 1
library=# SELECT * FROM write_off_act WHERE write_off_act_id=256;
 write_off_act_id | act_num |  act_date  | quantity_total | cost_total |  reason  | post_direction  | employee_id
------------------+---------+------------+----------------+------------+----------+-----------------+-------------
              256 | WO-0050 | 2025-05-28 |              3 |     709.47 | ветхость | Городской архив |         380
(1 строка)
```

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы созданы три хранимые процедуры в PostgreSQL для базы данных библиотеки по индивидуальному заданию. Также созданы триггерные функции и триггеры, обеспечивающие автоматическое соблюдение основных бизнес-правил.