|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 - ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА**

**Отчет**

**по лабораторной работе № 4**

**Вариант №1**

**Название:** Массивы, курсоры, триггеры, роли

**Дисциплина:** Базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-31Б |  |  | Г.А. Артемьев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | А.П. Кудрявцев |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2024

**Цель:**

Данная лабораторная работа призвана сформировать у студента понимание назначения массивов, курсоров, триггеров и ролей, их написание и использование.

**Задачи:**

* Ознакомиться с использованием массивов.
* Научиться (изменять\добавлять\удалять) данные в массиве с помощью встроенных операций.
* Получить знания о курсорах и научиться использовать курсоры.
* Узнать о ролях и пользователях.
* Научиться использоваться командами GRANT и REVOKE для того, чтобы (предоставлять\отзывать) доступ к данным.

**Задание:**

Написать и протестировать триггеры, выполняющие следующие действия для своей предметной области. Задания формулируются на примере БД «Библиотека» вариант №1.

* Контроль соответствия дат выдачи и возврата книги.
* Контроль дублирования читателя.
* Запрет на удаления читателя, если на руках у читателя имеется книга.
* Создать таблицу, состоящую из двух целочисленных полей и содержащую одну запись, для хранения количества книг и читателей. Написать триггеры для таблиц *книги* и *читатели,* подсчитывающие при добавлении и удалении общее количество книг и читателей и, сохраняющие итоги в созданной таблице.
* Создать пользователя test и выдать ему доступ к базе данных.
* Составить и выполнить скрипты присвоения новому пользователю прав доступа к таблицам, созданным в практическом задании 1. При этом права доступа к различным таблицам должны быть различными, а именно:
  + По крайней мере, для одной таблицы новому пользователю присваиваются права SELECT, INSERT, UPDATE в полном объеме.
  + По крайней мере, для одной таблицы новому пользователю присваиваются права SELECT и UPDATE только избранных столбцов.
  + По крайней мере, для одной таблицы новому пользователю присваивается только право SELECT.

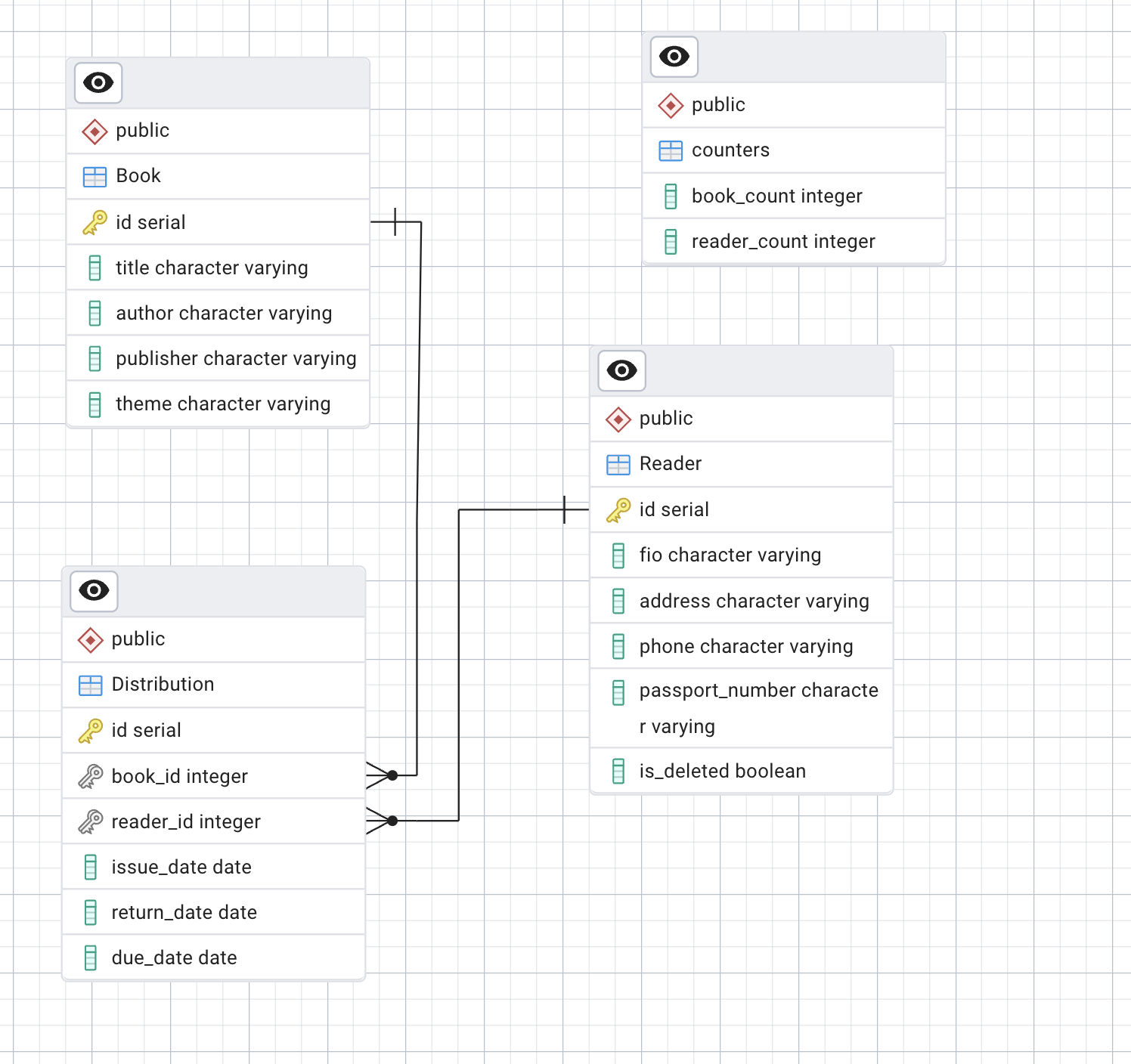


Рисунок 1 – ER диаграмма

**Код выполнения задания**

**1. Создание триггеров**

**Контроль соответствия дат выдачи и возврата книги**

CREATE OR REPLACE FUNCTION check\_distribution\_dates() RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN

IF NEW.due\_date < NEW.issue\_date THEN

RAISE EXCEPTION 'Дата возврата не может быть раньше даты выдачи.';

END IF;

RETURN NEW;

END;

$$;

CREATE TRIGGER check\_distribution\_dates\_trigger

BEFORE INSERT OR UPDATE ON Distribution

FOR EACH ROW

EXECUTE PROCEDURE check\_distribution\_dates();

**Контроль дублирования читателя**

CREATE OR REPLACE FUNCTION check\_reader\_duplicate() RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

DECLARE

existing\_count INTEGER;

BEGIN

SELECT COUNT(\*) INTO existing\_count

FROM "Reader"

WHERE passport\_number = NEW.passport\_number;

IF existing\_count > 0 THEN

RAISE EXCEPTION 'Читатель с таким номером паспорта уже существует.';

END IF;

RETURN NEW;

END;

$$;

CREATE TRIGGER check\_reader\_duplicate\_trigger

BEFORE INSERT ON Reader

FOR EACH ROW

EXECUTE PROCEDURE check\_reader\_duplicate();

**Запрет на удаление читателя, если на руках у него имеется книга**

CREATE OR REPLACE FUNCTION prevent\_reader\_deletion() RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

DECLARE

book\_count INTEGER;

BEGIN

SELECT COUNT(\*) INTO book\_count FROM Distribution WHERE reader\_id = OLD.id AND return\_date IS NULL;

IF book\_count > 0 THEN

RAISE EXCEPTION 'Нельзя удалить читателя, у которого на руках имеются книги.';

END IF;

RETURN OLD;

END;

$$;

CREATE TRIGGER prevent\_reader\_deletion\_trigger

BEFORE DELETE ON Reader

FOR EACH ROW

EXECUTE PROCEDURE prevent\_reader\_deletion();

**Подсчёт количества книг и читателей**

-- Создание таблицы для хранения подсчётов:

CREATE TABLE IF NOT EXISTS counters (

book\_count INTEGER DEFAULT 0,

reader\_count INTEGER DEFAULT 0

);

-- Инициализация начальных значений

INSERT INTO counters (book\_count, reader\_count) VALUES (0, 0);

-- Триггер для обновления количества книг:

CREATE OR REPLACE FUNCTION update\_book\_count() RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN

UPDATE counters SET book\_count = (SELECT COUNT(\*) FROM Book);

RETURN NEW;

END;

$$;

CREATE TRIGGER update\_book\_count\_trigger

AFTER INSERT OR DELETE ON Book

FOR EACH ROW

EXECUTE PROCEDURE update\_book\_count();

-- Триггер для обновления количества читателей:

CREATE OR REPLACE FUNCTION update\_reader\_count() RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN

UPDATE counters SET reader\_count = (SELECT COUNT(\*) FROM Reader);

RETURN NEW;

END;

$$;

CREATE TRIGGER update\_reader\_count\_trigger

AFTER INSERT OR DELETE ON Reader

FOR EACH ROW

EXECUTE PROCEDURE update\_reader\_count();

**2. Тестирование триггеров**

**Добавление книг и читателей:**

INSERT INTO Book (title, author, publisher, theme) VALUES ('Книга 1', 'Автор 1', 'Издательство 1', 'Тема 1');

INSERT INTO Reader (fio, address, phone, passport\_number) VALUES ('Иван Иванов', 'ул. Ленина, 1', '1234567890', '9876543210');

**Попытка дублирования читателя:**

INSERT INTO Reader (fio, address, phone, passport\_number) VALUES ('Иван Иванов', 'ул. Ленина, 1', '1234567890', '9876543210');

-- Будет выдана ошибка: 'Читатель с таким номером паспорта уже существует.'

**Удаление читателя с выданной книгой:**

DELETE FROM Reader WHERE id = 1;

-- Будет выдана ошибка: 'Нельзя удалить читателя, у которого на руках имеются книги.'

**Проверка таблицы counters:**

SELECT \* FROM counters;

**3. Создание пользователя и назначение прав**

**Создание пользователя:**

CREATE USER test WITH PASSWORD '123';

**Назначение прав:**

**Права доступа к базе данных:**

GRANT ALL PRIVILEGES ON DATABASE lab\_2 TO test;

**Полные права на таблицу Book:**

GRANT SELECT, INSERT, UPDATE ON TABLE Book TO test;

**Частичные права на таблицу Reader:**

GRANT SELECT (fio, address), UPDATE (fio, address) ON TABLE Reader TO test;

**Только право SELECT на таблицу Distribution:**

GRANT SELECT ON TABLE Distribution TO test;

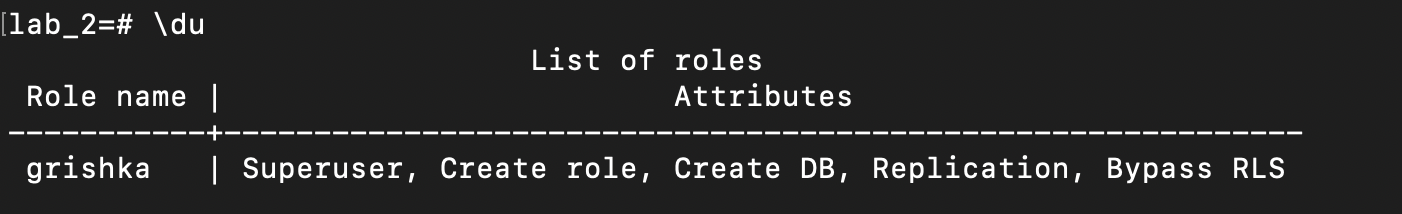


Рисунок 2 – Список ролей



Рисунок 3 – Привилегии выданные с условиями задания

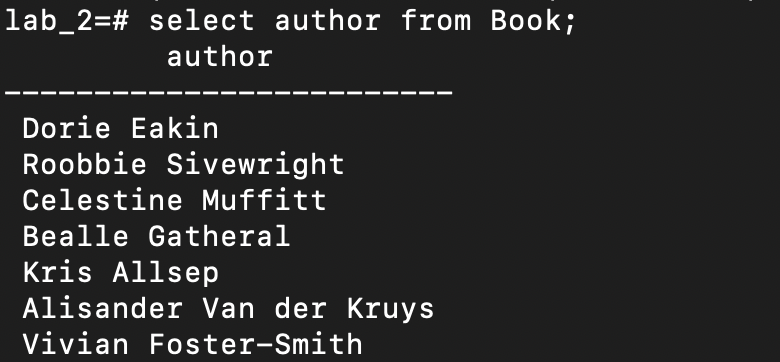


Рисунок 4 – Успешное получение данных из столбца name

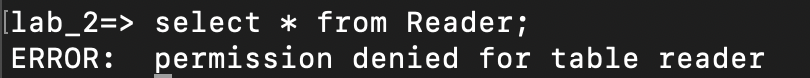


Рисунок 5 – Ошибка доступа ко всем столбцам

**Вывод:** в ходе лабораторной работы были написаны и протестированы триггеры, а также создан пользователь, которому были присвоены частичные права к таблицам с условиями задания