|  |  |
| --- | --- |
| 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 № 2**

**Название:** Создание БД

**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

**Задание №1**

Второе практическое задание связано с проектированием схемы базы данных для работы приложения (WEB/Mobile/Desktop). Каждый индивидуальный вариант (см. приложение «Схемы предметных областей») содержит предметную область, для которой должна быть разработана база данных. К первоначальной модели предметной области необходимо добавить не менее 2-х дополнительных сущностей (таблиц), необходимых для более полного решения поставленной задачи. Студент должен решить, для чего будет использоваться создаваемая база данных, построить бизнес-процессы её применения и, исходя из анализа бизнес-процессов, создать концептуальную схему БД. Результатом данной части лабораторной работы является схема базы данных (в виде ER-диаграммы, содержащей таблицы и связи между ними, с уточнением типов полей, с описанием внешних и первичных ключей). При сдаче задания студент должен обосновать соответствие созданной схемы поставленной задаче.

Необходимо определить первичные и внешние ключи (с атрибутами NO ACTION, CASCADE и др.).

На основе разработанной ER–модели, создать БД на сервере PostgreSQL.

Для проектирования схемы и построения диаграммы можно использовать любые предназначенные для этой цели средства.

# Вариант №1

Предметная область для практических заданий: **Библиотека**

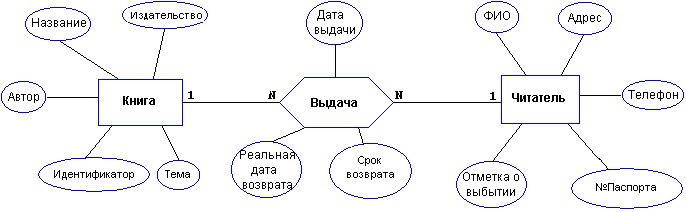


Рисунок 1 - Схема предметной области

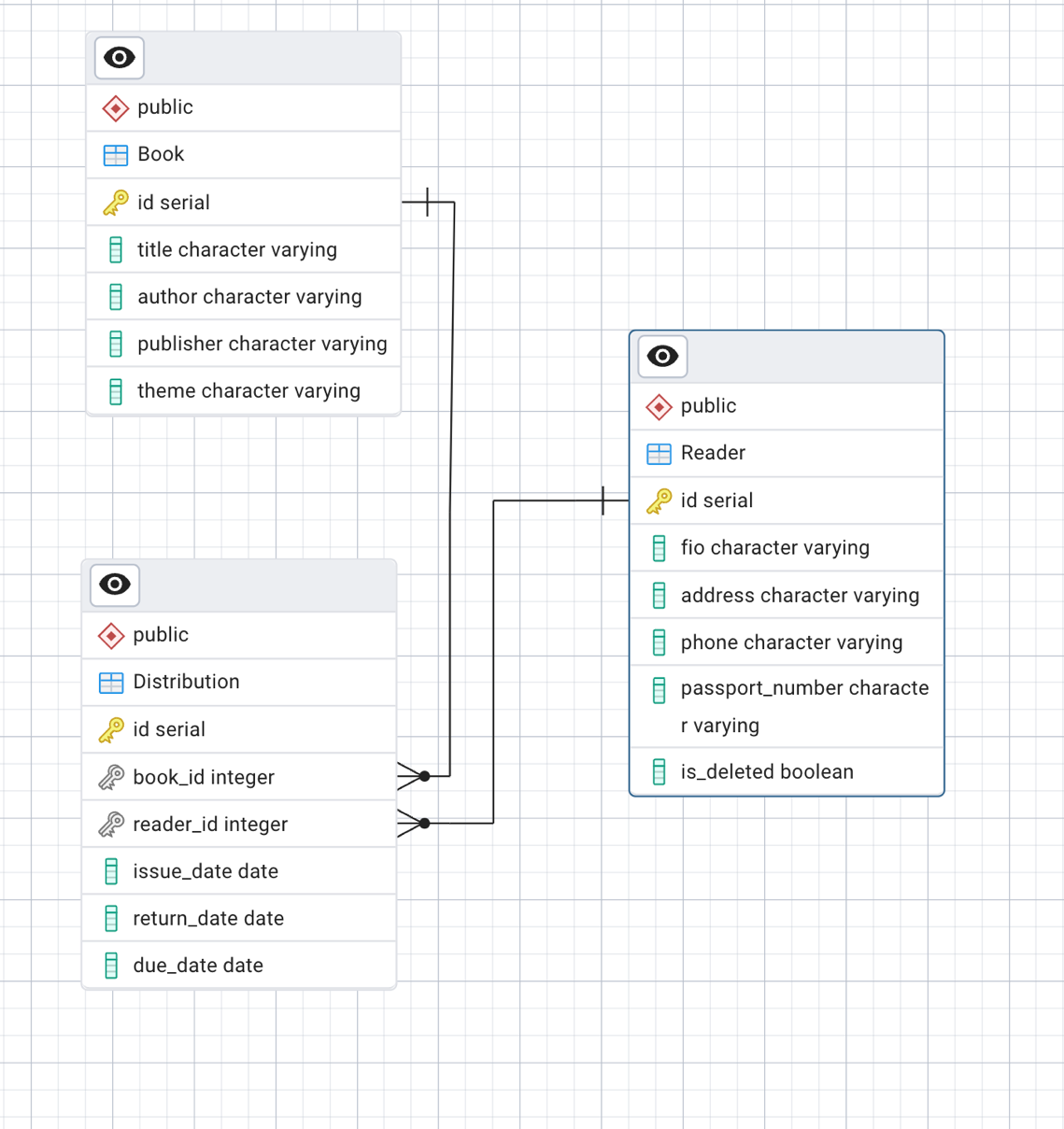


Рисунок 2 - ER-диаграмма

**Задание №2**

1. Подготовить SQL-скрипта для создания БД и таблиц согласно схеме, полученной в предыдущем задании. БД должна быть создана с локалью ru\_RU.utf8.
2. Необходимо определить первичные и внешние ключи.
3. Необходимо определить ограничения полей (возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т.д.).
4. Необходимо определить индексы, позволяющие ускорить запросы.
5. Создать БД и таблицы. Проверить локаль БД, вызвав команду \l в psql или SHOW LC\_COLLATE, SHOW LC\_CTYPE.

SQL-скрипт:   
-- Создание базы данных с локалью ru\_RU.utf8

CREATE DATABASE LibraryDB

WITH

OWNER = postgres

ENCODING = 'UTF8'

LC\_COLLATE = 'ru\_RU.utf8'

LC\_CTYPE = 'ru\_RU.utf8'

TEMPLATE = template0;

-- Таблица Book

CREATE TABLE Book (

id SERIAL PRIMARY KEY, -- Первичный ключ

title VARCHAR(255) NOT NULL, -- Название книги (обязательно)

author VARCHAR(255) NOT NULL, -- Автор книги (обязательно)

publisher VARCHAR(255), -- Издательство

theme VARCHAR(100) -- Тема книги

);

-- Таблица Reader

CREATE TABLE Reader (

id SERIAL PRIMARY KEY, -- Первичный ключ

fio VARCHAR(255) NOT NULL, -- ФИО (обязательно)

address VARCHAR(255), -- Адрес

phone VARCHAR(20), -- Телефон

passport\_number VARCHAR(50) UNIQUE, -- Уникальный номер паспорта

is\_deleted BOOLEAN NOT NULL DEFAULT FALSE -- Отметка о выбытии

);

-- Таблица Distribution

CREATE TABLE Distribution (

id SERIAL PRIMARY KEY, -- Первичный ключ

book\_id INT NOT NULL, -- Ссылка на книгу

reader\_id INT NOT NULL, -- Ссылка на читателя

issue\_date DATE NOT NULL, -- Дата выдачи (обязательно)

return\_date DATE, -- Реальная дата возврата (может быть NULL)

due\_date DATE NOT NULL, -- Срок возврата (обязательно)

CONSTRAINT fk\_book FOREIGN KEY (book\_id) REFERENCES Book (id) ON DELETE CASCADE, -- Внешний ключ с каскадным удалением

CONSTRAINT fk\_reader FOREIGN KEY (reader\_id) REFERENCES Reader (id) ON DELETE CASCADE -- Внешний ключ с каскадным удалением

);

-- Определение ограничений

-- Проверочные ограничения для поля due\_date (срок возврата не может быть раньше даты выдачи)

ALTER TABLE Distribution ADD CONSTRAINT check\_due\_date CHECK (due\_date >= issue\_date);

-- Создание индексов

-- Индекс для ускорения поиска по названию книги

CREATE INDEX idx\_book\_title ON Book (title);

-- Индекс для ускорения поиска по автору книги

CREATE INDEX idx\_book\_author ON Book (author);

-- Индекс для ускорения поиска по ФИО читателя

CREATE INDEX idx\_reader\_fio ON Reader (fio);

-- Индекс для ускорения поиска по номеру паспорта

CREATE INDEX idx\_reader\_passport ON Reader (passport\_number);

-- Индекс для ускорения поиска по дате выдачи

CREATE INDEX idx\_distribution\_issue\_date ON Distribution (issue\_date);

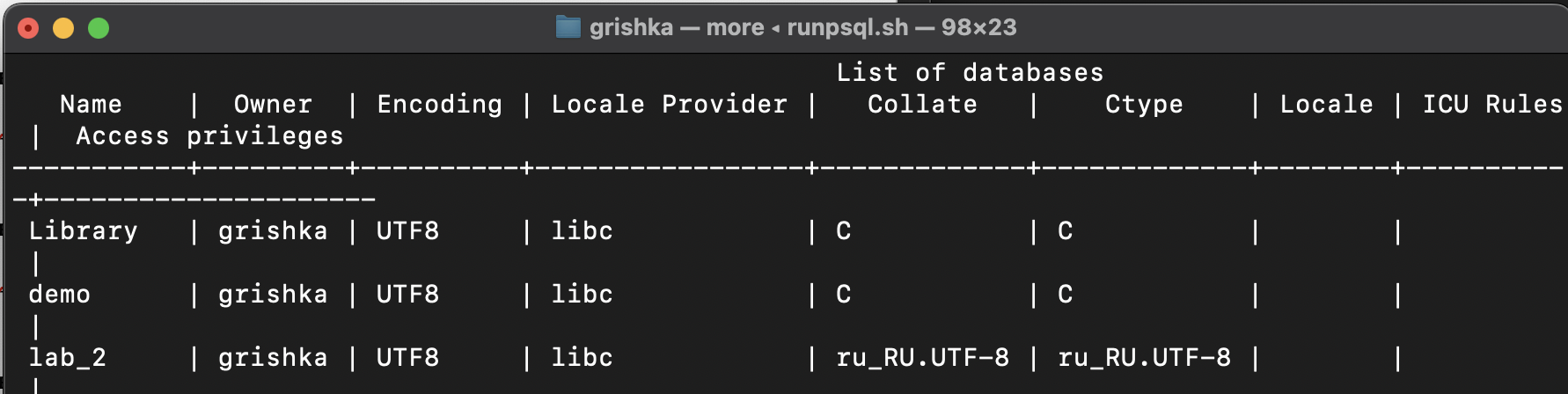


Рисунок 3 – Результат

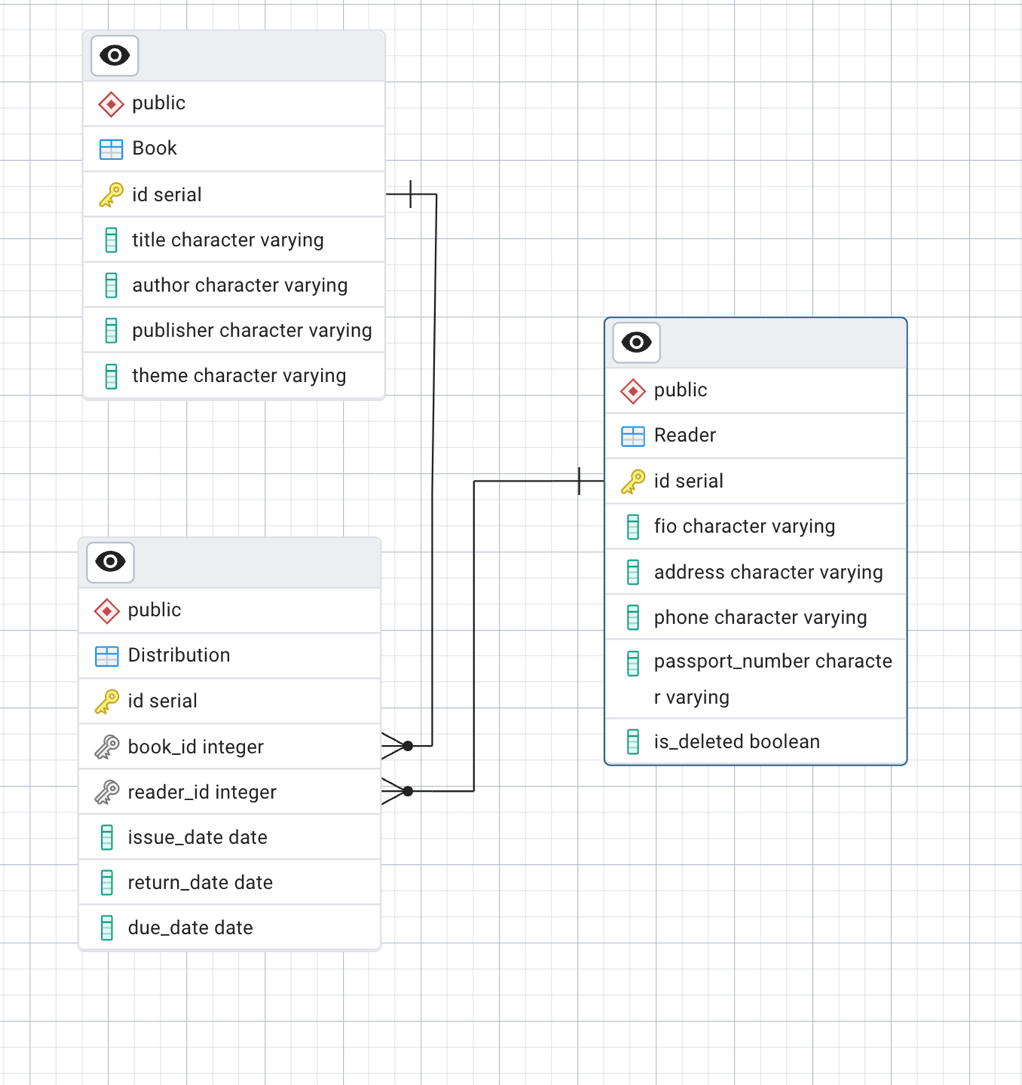


Рисунок 4 - ER-диаграмма

**Задание №3**

1. Заполнить данными созданные таблицы с использованием SQL-скриптов для вставки соответствующих строк в таблицы БД. Объем данных должен составлять не менее 10 экземпляров для каждой из стержневых сущностей и 1000 экземпляров для целевой сущности.
2. Необходимо подготовить два запроса:

* Запрос к одной таблице, содержащий фильтрацию по нескольким полям.
* Запрос к нескольким связанным таблицам (JOIN), содержащий фильтрацию по нескольким полям.

1. Для каждого из этих запросов необходимо провести следующие шаги:

* Получить время выполнения запроса без использования индексов.
* Создать нужные индексы, позволяющие ускорить запрос. Получить время выполнения запроса с использованием индексов и сравнить с первоначальной статистикой.

1. Сформулировать 3 запроса на изменение и удаление из базы данных. Запросы должны быть сформулированы в терминах предметной области.
2. Среди запросов обязательно должны быть такие, которые будут вызывать срабатывание ограничений целостности. Составить SQL-скрипты для выполнения этих запросов.

-- Пример вставки данных в таблицу Book

INSERT INTO Book (id, title, author, publisher, theme) values (1, 'Dr', 'Dorie Eakin', 'Voonte', 'Drama');

INSERT INTO Book (id, title, author, publisher, theme) values (2, 'Mr', 'Roobbie Sivewright', 'Plajo', 'Animation|Children|Comedy|Western');

INSERT INTO Book (id, title, author, publisher, theme) values (3, 'Mrs', 'Celestine Muffitt', 'Meejo', 'Comedy|Musical');

-- Пример вставки данных в таблицу Reader

INSERT INTO Reader (id, fio, address, phone, passport\_number, is\_deleted) values (1, 'Rudyard Pethybridge', '6 Elka Pass', '763 635 3613', '321930 322987 345877 321931 342999 311564 312652 314783 342675 312431', false);

INSERT INTO Reader (id, fio, address, phone, passport\_number, is\_deleted) values (2, 'Antonella Skilbeck', '6486 Golf Course Parkway', '257 725 5735', '321930 322987 345877 321931 342999 311564 312652 314783 342675 312431', false);

INSERT INTO Reader (id, fio, address, phone, passport\_number, is\_deleted) values (3, 'Lavinia Chessell', '357 Corry Hill', '910 162 4734', '321930 322987 345877 321931 342999 311564 312652 314783 342675 312431', true);

-- Пример вставки данных в таблицу Distribution

INSERT INTO Distribution (id, book\_id, reader\_id, issue\_date, return\_date, due\_date) values (1, 18, 37, '15.8.2023', '2.6.2022', '21.9.2022');

INSERT INTO Distribution (id, book\_id, reader\_id, issue\_date, return\_date, due\_date) values (2, 10, 32, '6.7.2024', '4.12.2021', '11.4.2023');

INSERT INTO Distribution (id, book\_id, reader\_id, issue\_date, return\_date, due\_date) values (3, 50, 6, '28.11.2023', '10.3.2024', '3.2.2024');

****

Таблица 1 – Книги

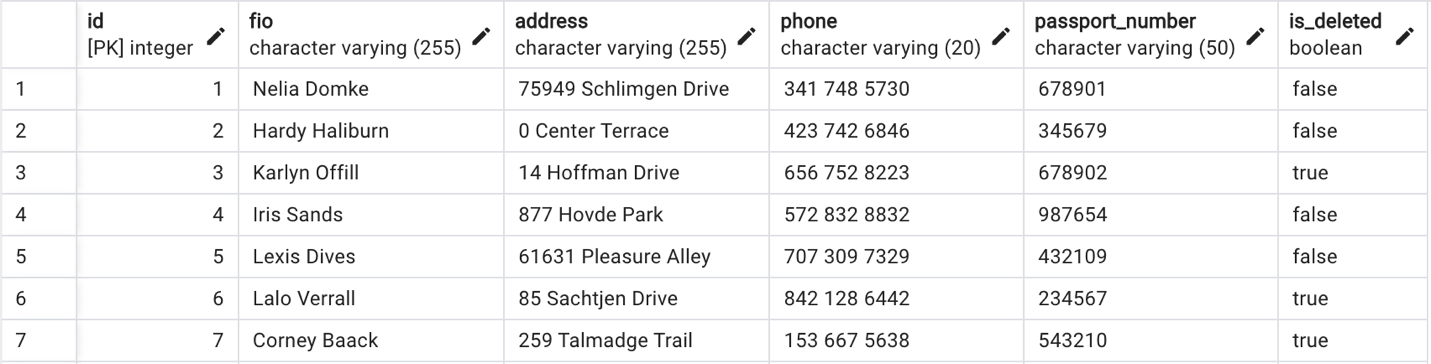


Таблица 2 – Читатели

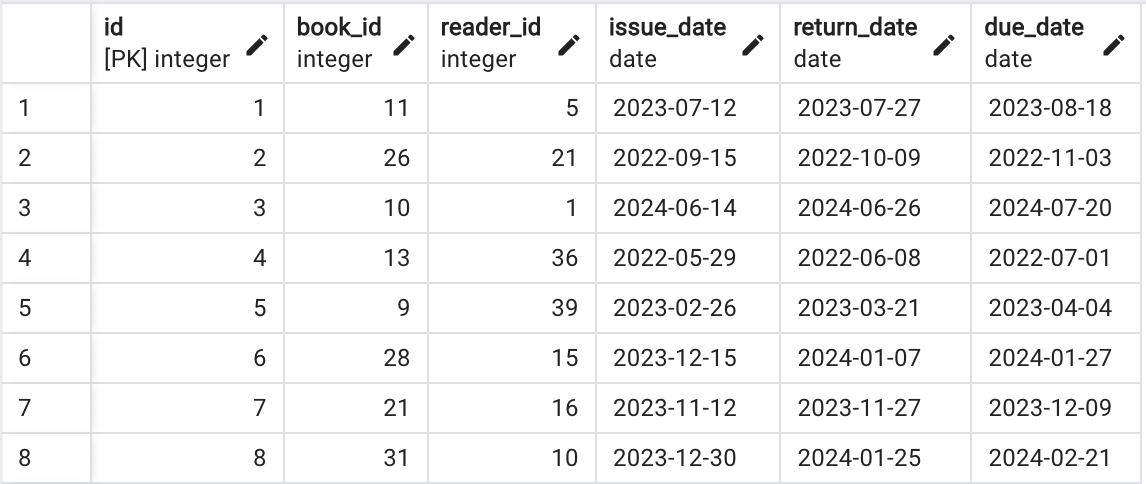


Таблица 3 – Выдача

-- Запрос 1: к одной таблице (Reader)

-- Без индексов

SET enable\_seqscan TO ON;

EXPLAIN ANALYZE SELECT \* FROM "Reader" WHERE address LIKE '9%' AND is\_deleted = FALSE;

SET enable\_seqscan TO ON;

SELECT \* FROM "Reader" WHERE address LIKE '9%' AND is\_deleted = FALSE;

-- C индексами

SET enable\_seqscan TO OFF;

CREATE INDEX IF NOT EXISTS idx\_reader\_address ON "Reader"(address);

CREATE INDEX IF NOT EXISTS idx\_reader\_is\_deleted ON "Reader"(is\_deleted);

EXPLAIN ANALYZE SELECT \* FROM "Reader" WHERE address LIKE '9%' AND is\_deleted = FALSE;

SET enable\_seqscan TO OFF;

SELECT \* FROM "Reader" WHERE address LIKE '9%' AND is\_deleted = FALSE;

-- Запрос 2: к нескольким связанным таблицам (JOIN)

-- Без индексов

SET enable\_seqscan TO ON;

EXPLAIN ANALYZE SELECT r.fio, b.title, d.issue\_date, d.due\_date

FROM "Reader" r

JOIN "Distribution" d ON r.id = d.reader\_id

JOIN "Book" b ON d.book\_id = b.id

WHERE d.return\_date IS NULL AND d.due\_date > CURRENT\_DATE;

SET enable\_seqscan TO ON;

SELECT r.fio, b.title, d.issue\_date, d.due\_date

FROM "Reader" r

JOIN "Distribution" d ON r.id = d.reader\_id

JOIN "Book" b ON d.book\_id = b.id

WHERE d.return\_date IS NULL AND d.due\_date > CURRENT\_DATE;

-- С индексами (idx\_distribution\_due\_date \_id, idx\_distribution\_return\_date, idx\_distribution\_reader\_id, idx\_distribution\_book\_id)

SET enable\_seqscan TO OFF;

CREATE INDEX IF NOT EXISTS idx\_distribution\_due\_date ON "Distribution"(due\_date);

CREATE INDEX IF NOT EXISTS idx\_distribution\_return\_date ON "Distribution"(return\_date);

CREATE INDEX IF NOT EXISTS idx\_distribution\_reader\_id ON "Distribution"(reader\_id);

CREATE INDEX IF NOT EXISTS idx\_distribution\_book\_id ON "Distribution"(book\_id);

EXPLAIN ANALYZE SELECT r.fio, b.title, d.issue\_date, d.due\_date

FROM "Reader" r

JOIN "Distribution" d ON r.id = d.reader\_id

JOIN "Book" b ON d.book\_id = b.id

WHERE d.return\_date IS NULL AND d.due\_date > CURRENT\_DATE;

SET enable\_seqscan TO OFF;

SELECT r.fio, b.title, d.issue\_date, d.due\_date

FROM "Reader" r

JOIN "Distribution" d ON r.id = d.reader\_id

JOIN "Book" b ON d.book\_id = b.id

WHERE d.return\_date IS NULL AND d.due\_date > CURRENT\_DATE;

-- 4. Запросы на изменение и удаление

-- Изменение срока возврата:

BEGIN;

UPDATE "Distribution"

SET due\_date = due\_date + INTERVAL '7 days'

WHERE book\_id = 1 AND reader\_id = 2;

COMMIT;

SELECT \* FROM "Distribution" WHERE book\_id = 1 AND reader\_id = 2;

-- Удаление выбывших читателей:

BEGIN;

DELETE FROM "Reader" WHERE is\_deleted = TRUE;

COMMIT;

SELECT \* FROM "Reader";

-- Добавление новой выдачи книги:

BEGIN;

INSERT INTO "Distribution" (book\_id, reader\_id, issue\_date, due\_date)

VALUES (3, 4, '2024-01-15', '2024-01-25')

ON CONFLICT (book\_id, reader\_id) DO NOTHING;

SELECT \* FROM "Distribution" WHERE book\_id = 3 AND reader\_id = 4;

COMMIT;

-- 5. Скрипты, вызывающие срабатывание ограничений целостности

-- Попытка удалить книгу, на которую есть ссылки в Distribution:

BEGIN;

DELETE FROM "Book" WHERE id = 1;

COMMIT;

-- Попытка вставить запись с несуществующим reader\_id:

BEGIN;

INSERT INTO "Distribution" (book\_id, reader\_id, issue\_date, due\_date)

VALUES (1, 999, '2024-01-01', '2024-01-10');

COMMIT;

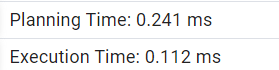
-- Попытка установить due\_date раньше issue\_date:

BEGIN;

UPDATE "Distribution"

SET due\_date = issue\_date - INTERVAL '1 day'

WHERE id = 1;

COMMIT;

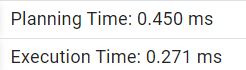
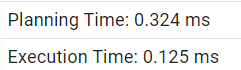
Рисунок 5 – Время работы без индексов Рисунок 6 – Время работы с индексом

Рисунок 7 – Время работы без индексов Рисунок 8 – Время работы с индексом

**Задание №4**

1. Создайте три представления, которые позволяли бы упростить манипуляции с данными или позволяли бы ограничить доступ к данным, предоставляя только необходимую информацию.
2. Продемонстрировать изменение и вставку данных через представления.
3. Продемонстрировать невозможность изменения данных через представление.

-- 1. Создание представлений

-- Предоставляет информацию только о читателях, которые не выбыли:

CREATE VIEW ActiveReaders AS

SELECT id, fio, address, phone

FROM "Reader"

WHERE is\_deleted = FALSE;

-- Отображает книги, которые находятся у читателей, с указанием срока возврата:

CREATE VIEW BooksDistribution AS

SELECT b.title AS book\_title, r.fio AS reader\_name, d.issue\_date, d.due\_date

FROM "Distribution" d

JOIN "Book" b ON d.book\_id = b.id

JOIN "Reader" r ON d.reader\_id = r.id

WHERE d.return\_date IS NULL;

-- Показывает книги с просроченным сроком возврата:

CREATE VIEW OverdueBooks AS

SELECT b.title AS book\_title, r.fio AS reader\_name, d.due\_date

FROM "Distribution" d

JOIN "Book" b ON d.book\_id = b.id

JOIN "Reader" r ON d.reader\_id = r.id

WHERE d.due\_date < CURRENT\_DATE AND d.return\_date IS NULL;

-- 2-3. Изменение и вставка данных через представления (неработоспособно)

-- Пример вставки (нельзя через представление)

UPDATE OverdueBooks

SET due\_date = CURRENT\_DATE + INTERVAL '7 days'

WHERE book\_title = 'Some Book';

-- Пример изменения (нельзя через представление)

INSERT INTO BooksDistribution (book\_title, reader\_name, issue\_date, due\_date) VALUES ('New Book', 'Jane Doe', '2024-01-01', '2024-01-10');

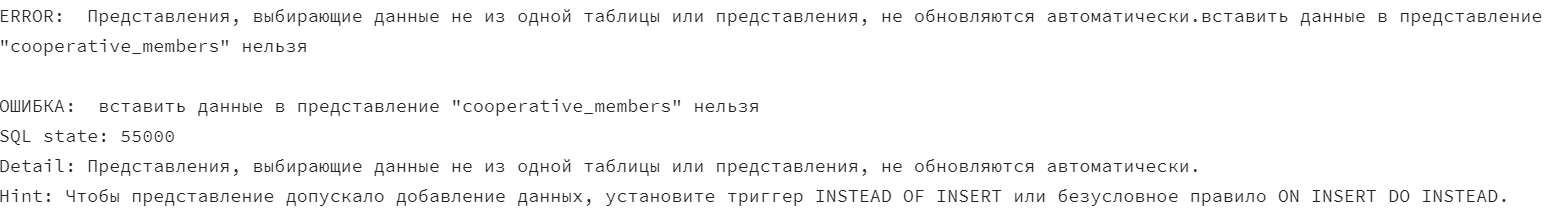


Рисунок 9 – Ошибка при вставке через представление

**Задание №5**

Скрипт SQL для 1 сессии

1. READ COMMITTED

-- Грязное чтение

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

UPDATE "Reader" SET address = 'Temporary Address' WHERE id = 1;

SELECT address FROM "Reader" WHERE id = 1;

ROLLBACK;

-- Неповторяющееся чтение

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

SELECT address FROM "Reader" WHERE id = 1;

SELECT address FROM "Reader" WHERE id = 1;

COMMIT;

2. REPEATABLE READ

-- Неповторяющееся чтение

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

START TRANSACTION;

SELECT address FROM "Reader" WHERE id = 1;

SELECT address FROM "Reader" WHERE id = 1;

COMMIT;

-- Фантомное чтение

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

START TRANSACTION;

SELECT COUNT(\*) FROM "Reader" WHERE address LIKE 'Test%';

SELECT COUNT(\*) FROM "Reader" WHERE address LIKE 'Test%';

COMMIT;

3. SERIALIZABLE

-- Фантомное чтение

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

SELECT COUNT(\*) FROM "Reader" WHERE address LIKE 'Test%';

SELECT COUNT(\*) FROM "Reader" WHERE address LIKE 'Test%';

COMMIT;

-- Скрипт SQL для 2 сессии

1. READ COMMITTED

-- Грязное чтение

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

SELECT address FROM "Reader" WHERE id = 1;

COMMIT;

-- Неповторяющееся чтение

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

UPDATE "Reader" SET address = 'Updated Address' WHERE id = 1;

COMMIT;

-- 2. REPEATABLE READ

-- Неповторяющееся чтение

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

START TRANSACTION;

UPDATE "Reader" SET address = 'Updated Address' WHERE id = 1;

COMMIT;

-- Фантомное чтение

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

START TRANSACTION;

INSERT INTO "Reader" (fio, address, is\_deleted) VALUES ('New Reader', 'Test Address', FALSE);

COMMIT;

-- 3. SERIALIZABLE

-- Фантомное чтение

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

INSERT INTO "Reader" (fio, address, is\_deleted) VALUES ('New Reader', 'Test Address', FALSE);

COMMIT;



Рисунки 11-12 – Резульат выполнение грязного чтения

**Вывод:** В ходе лабораторной работы была создана БД, а также написаны запросы и скрипты в сответсвии с заданиями.