|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Министерство науки и высшего образования РФ | | | | | |
| Федеральное государственное автономное | | | | | |
| образовательное учреждение высшего образования | | | | | |
| **«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»** | | | | | |
|  | | | | | |
| Институт космических и информационных технологий | | | | | |
| институт | | | | | |
| Программная инженерия | | | | | |
| кафедра | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №10** | | | | | |
| Проектирование базы данных и ее реализация в среде СУБД PostgreSQL | | | | | |
| тема | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| Преподаватель | |  |  |  | А. Д. Вожжов |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ23-17/1б, 032322546 |  |  |  | Е. А. Гуртякин |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |
|  | | | | | |
|  | | | | | |
| Красноярск 2025 | | | | | |

# ВВЕДЕНИЕ

## Цель работы

Изучить теоретический материал по теме «Проектирование базы данных и ее реализация в среде СУБД PostgreSQL». Выполнить задания.

## Задачи

В рамках данной практической работы необходимо выполнить следующие задачи:

1. изучить теоретический материал по предложенной теме;
2. выполнить задание;
3. предоставить отчёт преподавателю.

## Задание

Задание данной практической работы состоит из следующих частей:

1. Изучить материал лекций 6-8;
2. Выбрать предметную область, которая вам интересна и в которой вы разбираетесь;
3. Спроектировать базу данных для выбранной предметной области с учётом требований;
4. Ввести небольшое количество записей в таблицы базы данных, чтобы можно было продемонстрировать типичные запросы к базе данных;
5. Подготовить несколько типичных запросов к базе данных и сохранить их в отдельных текстовых файлах. Для демонстрации этих запросов их можно вызывать как извне утилиты «psql», так и изнутри нее.

Для каждого отношения (таблицы) необходимо указать номер нормальной формы, в которой это отношение находится, и кратко обосновать, из чего это следует. Если какое-либо отношение не находится хотя бы в 3НФ, необходимо обосновать, почему принято такое проектное решение.

# ХОД РАБОТЫ

## Предметная область

В качестве предметной области была выбрана система выдачи и хранения книг.

## Концептуальная модель

Концептуальная модель представлена на рисунке 1.

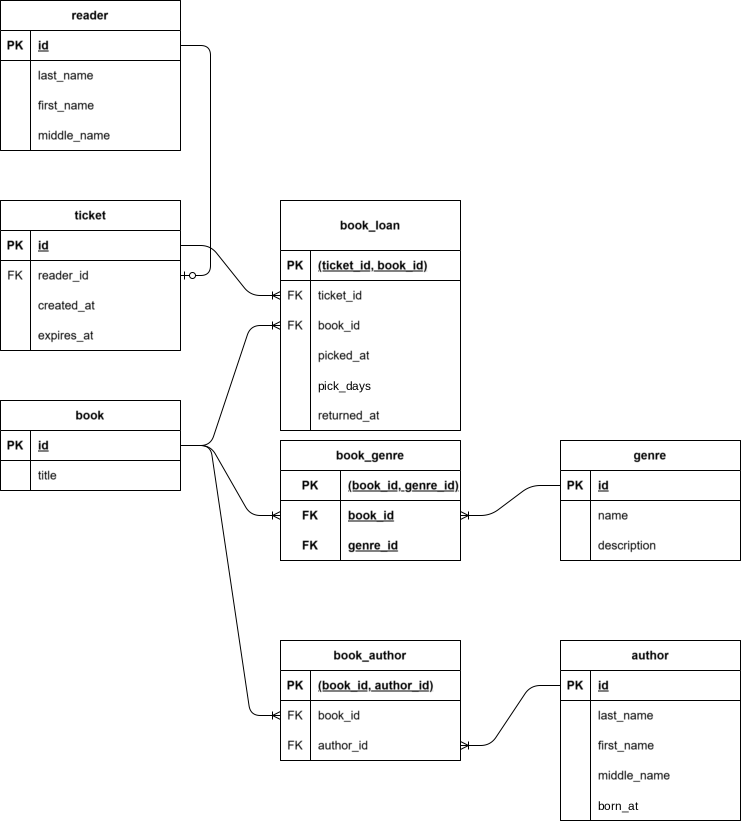


Рисунок 1 – Концептуальная модель

## Логическая модель

Логическая модель представлена на рисунке 2.

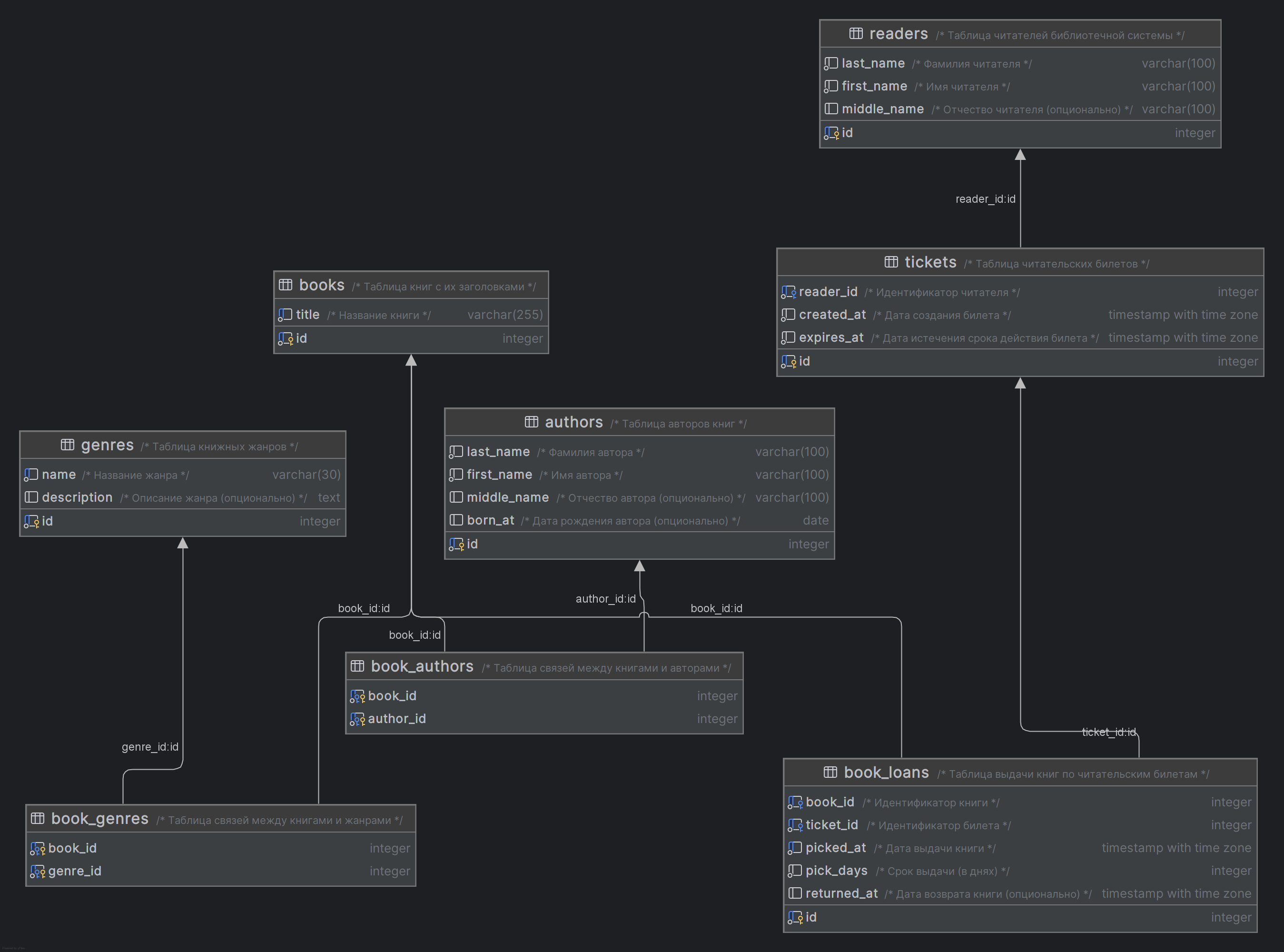


Рисунок 2 – Логическая модель

## Физическая модель

Физическая модель представлена ниже.

CREATE SCHEMA library;

CREATE TABLE library.readers (

id SERIAL PRIMARY KEY,

last\_name VARCHAR(100) NOT NULL,

first\_name VARCHAR(100) NOT NULL,

middle\_name VARCHAR(100),

CONSTRAINT check\_last\_name CHECK (TRIM(last\_name) <> ''),

CONSTRAINT check\_first\_name CHECK (TRIM(first\_name) <> ''),

CONSTRAINT check\_middle\_name CHECK (middle\_name IS NULL OR TRIM(middle\_name) <> '')

);

COMMENT ON TABLE library.readers IS 'Таблица читателей библиотечной системы';

COMMENT ON COLUMN library.readers.last\_name IS 'Фамилия читателя';

COMMENT ON COLUMN library.readers.first\_name IS 'Имя читателя';

COMMENT ON COLUMN library.readers.middle\_name IS 'Отчество читателя (опционально)';

CREATE TABLE library.tickets (

id SERIAL PRIMARY KEY,

reader\_id INTEGER NOT NULL,

created\_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT\_TIMESTAMP,

expires\_at TIMESTAMPTZ NOT NULL,

FOREIGN KEY (reader\_id) REFERENCES library.readers(id) ON DELETE CASCADE,

CONSTRAINT check\_created\_at CHECK (created\_at <= expires\_at)

);

COMMENT ON TABLE library.tickets IS 'Таблица читательских билетов';

COMMENT ON COLUMN library.tickets.reader\_id IS 'Идентификатор читателя';

COMMENT ON COLUMN library.tickets.created\_at IS 'Дата создания билета';

COMMENT ON COLUMN library.tickets.expires\_at IS 'Дата истечения срока действия билета';

CREATE TABLE library.books (

id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL UNIQUE,

CONSTRAINT check\_title CHECK (TRIM(title) <> '')

);

COMMENT ON TABLE library.books IS 'Таблица книг с их заголовками';

COMMENT ON COLUMN library.books.title IS 'Название книги';

CREATE TABLE library.genres (

id SERIAL PRIMARY KEY,

name VARCHAR(30) NOT NULL UNIQUE,

description TEXT,

CONSTRAINT check\_name CHECK (TRIM(name) <> ''),

CONSTRAINT check\_description CHECK (description IS NULL OR TRIM(description) <> '')

);

COMMENT ON TABLE library.genres IS 'Таблица книжных жанров';

COMMENT ON COLUMN library.genres.name IS 'Название жанра';

COMMENT ON COLUMN library.genres.description IS 'Описание жанра (опционально)';

CREATE TABLE library.book\_genres (

book\_id INTEGER NOT NULL,

genre\_id INTEGER NOT NULL,

PRIMARY KEY (book\_id, genre\_id),

FOREIGN KEY (book\_id) REFERENCES library.books(id) ON DELETE CASCADE,

FOREIGN KEY (genre\_id) REFERENCES library.genres(id) ON DELETE CASCADE

);

COMMENT ON TABLE library.book\_genres IS 'Таблица связей между книгами и жанрами';

CREATE TABLE library.authors (

id SERIAL PRIMARY KEY,

last\_name VARCHAR(100) NOT NULL,

first\_name VARCHAR(100) NOT NULL,

middle\_name VARCHAR(100),

born\_at DATE,

CONSTRAINT check\_last\_name CHECK (TRIM(last\_name) <> ''),

CONSTRAINT check\_first\_name CHECK (TRIM(first\_name) <> ''),

CONSTRAINT check\_middle\_name CHECK (middle\_name IS NULL OR TRIM(middle\_name) <> ''),

CONSTRAINT check\_born\_at CHECK (born\_at IS NULL OR born\_at <= CURRENT\_DATE)

);

COMMENT ON TABLE library.authors IS 'Таблица авторов книг';

COMMENT ON COLUMN library.authors.last\_name IS 'Фамилия автора';

COMMENT ON COLUMN library.authors.first\_name IS 'Имя автора';

COMMENT ON COLUMN library.authors.middle\_name IS 'Отчество автора (опционально)';

COMMENT ON COLUMN library.authors.born\_at IS 'Дата рождения автора (опционально)';

CREATE TABLE library.book\_authors (

book\_id INTEGER NOT NULL,

author\_id INTEGER NOT NULL,

PRIMARY KEY (book\_id, author\_id),

FOREIGN KEY (book\_id) REFERENCES library.books(id) ON DELETE CASCADE,

FOREIGN KEY (author\_id) REFERENCES library.authors(id) ON DELETE CASCADE

);

COMMENT ON TABLE library.book\_authors IS 'Таблица связей между книгами и авторами';

CREATE TABLE library.book\_loans (

id SERIAL PRIMARY KEY,

book\_id INTEGER NOT NULL,

ticket\_id INTEGER NOT NULL,

picked\_at TIMESTAMPTZ NOT NULL,

pick\_days INT NOT NULL,

returned\_at TIMESTAMPTZ,

UNIQUE (book\_id, ticket\_id, picked\_at),

FOREIGN KEY (book\_id) REFERENCES library.books(id) ON DELETE CASCADE,

FOREIGN KEY (ticket\_id) REFERENCES library.tickets(id) ON DELETE CASCADE,

CONSTRAINT check\_picked\_at CHECK (picked\_at <= CURRENT\_TIMESTAMP),

CONSTRAINT check\_pick\_days CHECK (pick\_days > 0),

CONSTRAINT check\_returned\_at CHECK (returned\_at IS NULL OR returned\_at >= picked\_at)

);

COMMENT ON TABLE library.book\_loans IS 'Таблица выдачи книг по читательским билетам';

COMMENT ON COLUMN library.book\_loans.book\_id IS 'Идентификатор книги';

COMMENT ON COLUMN library.book\_loans.ticket\_id IS 'Идентификатор билета';

COMMENT ON COLUMN library.book\_loans.picked\_at IS 'Дата выдачи книги';

COMMENT ON COLUMN library.book\_loans.pick\_days IS 'Срок выдачи (в днях)';

COMMENT ON COLUMN library.book\_loans.returned\_at IS 'Дата возврата книги (опционально)';

-- Индексы для оптимизации

CREATE INDEX idx\_tickets\_reader\_id ON library.tickets(reader\_id);

CREATE INDEX idx\_book\_genres\_book\_id ON library.book\_genres(book\_id);

CREATE INDEX idx\_book\_authors\_book\_id ON library.book\_authors(book\_id);

CREATE INDEX idx\_book\_loans\_ticket\_id ON library.book\_loans(ticket\_id);

CREATE INDEX idx\_book\_loans\_book\_id ON library.book\_loans(book\_id);

-- Триггер для проверки выдачи книги на срок, превышающий срок действия читательского билета

CREATE FUNCTION check\_loan\_expiry() RETURNS TRIGGER AS $$

BEGIN

IF NEW.picked\_at + NEW.pick\_days \* INTERVAL '1 day' > (

SELECT expires\_at FROM library.tickets WHERE id = NEW.ticket\_id

) THEN

RAISE EXCEPTION 'Период выдачи книги превышает срок действия читательского билета';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger\_check\_loan\_expiry

BEFORE INSERT OR UPDATE ON library.book\_loans

FOR EACH ROW EXECUTE FUNCTION check\_loan\_expiry();

## Нормальные формы

Нормальные формы для таблиц представлены в таблице 1.

Таблица 1 – нормальные формы таблиц базы данных

|  |  |
| --- | --- |
| **Таблица** | **Нормальные формы таблицы** |
| readers | 1, 2, 3 |
| tickets | 1, 2, 3 |
| books | 1, 2, 3 |
| genres | 1, 2, 3 |
| book\_genres | 1, 2, 3 |
| authors | 1, 2, 3 |
| book\_authors | 1, 2, 3 |
| book\_loans | 1, 2, 3 |

# ЗАКЛЮЧЕНИЕ

По результатам работы был изучен теоретический материал по теме «Проектирование базы данных и ее реализация в среде СУБД PostgreSQL». Все поставленные цели и задачи были выполнены.