Практика по базам данных
ОТЧЕТ

Чирков Дмитрий Виктрович
Группа 23.Б15-мм

Предметная область: «Библиотека»

Реализация: PostgreSQL

Содержание

ОПИСАНИЕ СИСТЕМЫ

Требования

Система предназначена для учета выдачи и возврата книг в библиотеке редких книг.

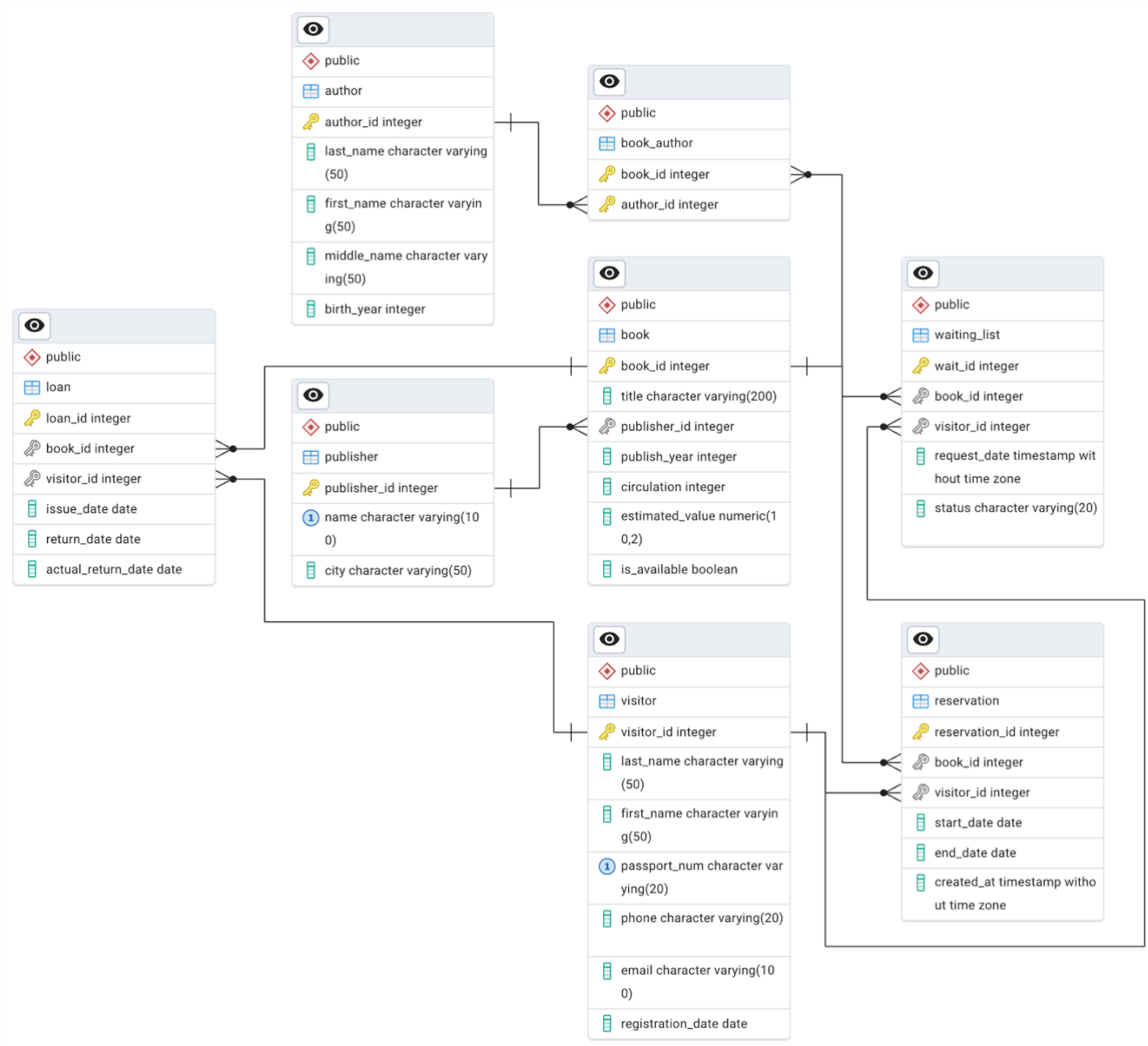
Сотрудник библиотеки регистрирует выданные посетителю и возвращенные им книги. Данные о выданных книгах сохраняются в истории выдач. Если нужная посетителю книга в данный момент находится на руках, сотрудник библиотеки регистрирует запрос на нее и ставит его в очередь ожидания.

Если посетитель хочет зарезервировать за собой книгу на определенный период времени, сотрудник библиотеки проверяет, нет ли заказов на резервирование этой книги на этот период и, если их нет, резервирует книгу для данного посетителя на данный период.

Информация, которую нужно хранить о посетителях – минимальна, она включает ФИО, паспортные и контактные данные.

Информация о книгах включает не только название и авторов, но и год выпуска, издательство, тираж, оценочная стоимость. Особенностью библиотеки редких книг является то, что каждое произведение здесь в единственном экземпляре.

Модель данных



Функциональность

Серверная часть

Хранимые процедуры / функции	Реализация (имя в скрипте)	Комментарии
Проверка доступности книги	check_book_availability	Функция. Проверяет статус is_available и наличие пересечений по датам в бронировании.
Регистрация выдачи	register_loan	Процедура. Создает запись в loan, меняет статус книги. Если книга недоступна — вызывает исключение.
Генерация отчета о штрафах	calculate_fines	Процедура с курсором. Проходит по всем просроченным книгам и рассчитывает сумму штрафа.
Обновление статуса при возврате	tf_on_return	Триггерная функция. Автоматически делает книгу доступной при простановке даты возврата.

Триггеры	Реализация	Комментарии
Авто-возврат книги в фонд	tr_on_return	AFTER UPDATE на таблице loan. Меняет статус книги на TRUE.
Валидация дат бронирования	tr_check_res_dates	BEFORE INSERT на reservation. Запрещает бронирование "задним числом".
Обработка очереди ожидания	tr_book_became_free	AFTER UPDATE на book. Если книга освободилась, проверяет waiting_list и уведомляет первого в очереди.

Представления	Реализация	Комментарии
Полный каталог	v_full_catalog	Сборная таблица: Книга + Авторы (строкой) + Издательство + Категория ценности.
Активные выдачи	v_active_loans	Список книг, которые сейчас на руках (кто взял, сколько дней держит).
Рейтинг читателей	v_visitor_stats	Статистика по количеству взятых книг каждым посетителем.

Клиентская часть

Экранные формы / Задачи	Запрос (из queries.sql)	Описание
Поиск и фильтрация	Запрос №1	Поиск книг определенного издательства за определенный период.
Карточка автора	Запрос №2	(Left Join) Вывод всех авторов и их книг (даже если книг нет).
Статистика фонда	Запрос №3, №4	(Group By + Having) Подсчет книг по издательствам и поиск самых дорогих коллекций.
Аналитика стоимости	Запрос №5	(Подзапрос в Select) Сравнение стоимости книги со средней по библиотеке.
Исторический поиск	Запрос №6	(Подзапрос в Where) Книги авторов XIX века.
Аудит посетителей	Запрос №7	(Exists) Поиск посетителей, которые зарегистрировались, но не брали книг.

Экранные формы / Задачи	Запрос (из queries.sql)	Описание
Рассылка	Запрос №8	(Union) Общий список персон (Авторы + Посетители) для уведомлений.
Администрирование	Запрос №9, №10	Переоценка фондов и очистка старых заявок в очереди.
Отчеты	Запрос №11, №12	Отчет по особо ценным книгам (через View) и самая популярная книга.

СКРИПТЫ

Серверная часть

```
-- ================================================================
-- Представления
-- ================================================================

-- Полный каталог книг (Книги + Авторы одной строкой + Издательство)
CREATE OR REPLACE VIEW v_full_catalog AS
SELECT
    b.book_id,
    b.title,
    p.name AS publisher,
    b.publish_year,
    STRING_AGG(a.last_name || ' ' || LEFT(a.first_name, 1) || '.', ', ') AS authors,
    b.is_available,
    CASE WHEN b.estimated_value > 50000 THEN 'Особо ценная' ELSE 'Обычная' END AS category
FROM book b
JOIN publisher p ON b.publisher_id = p.publisher_id
JOIN book_author ba ON b.book_id = ba.book_id
JOIN author a ON ba.author_id = a.author_id
GROUP BY b.book_id, b.title, p.name, b.publish_year, b.is_available, b.estimated_value;

-- Активные выдачи (Кто что сейчас читает)
CREATE OR REPLACE VIEW v_active_loans AS
SELECT
    l.loan_id,
    b.title,
    v.last_name || ' ' || v.first_name AS visitor_name,
    l.issue_date,
    l.return_date,
```

```
    (CURRENT_DATE - l.issue_date) AS days_held
FROM loan l
JOIN book b ON l.book_id = b.book_id
JOIN visitor v ON l.visitor_id = v.visitor_id
WHERE l.actual_return_date IS NULL;

-- Рейтинг самых читающих посетителей
CREATE OR REPLACE VIEW v_visitor_stats AS
SELECT
    v.visitor_id,
    v.last_name,
    COUNT(l.loan_id) as total_loans
FROM visitor v
LEFT JOIN loan l ON v.visitor_id = l.visitor_id
GROUP BY v.visitor_id, v.last_name;

-- ================================================================
-- Функции и Процедуры
-- ================================================================

-- Проверка доступности книги (учитывается статус и бронь)
CREATE OR REPLACE FUNCTION check_book_availability(p_book_id INTEGER)
RETURNS BOOLEAN AS $$
DECLARE
    v_is_avail BOOLEAN;
    v_reserved_count INTEGER;
BEGIN
    SELECT is_available INTO v_is_avail FROM book WHERE book_id = p_book_id;

    SELECT COUNT(*) INTO v_reserved_count
    FROM reservation
    WHERE book_id = p_book_id
      AND CURRENT_DATE BETWEEN start_date AND end_date;

    IF v_is_avail = TRUE AND v_reserved_count = 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END;
$$ LANGUAGE plpgsql;

-- Регистрация выдачи книги
```

```
CREATE OR REPLACE PROCEDURE register_loan(
    p_book_id INTEGER,
    p_visitor_id INTEGER,
    p_days INTEGER
) LANGUAGE plpgsql AS $$
BEGIN
    IF NOT check_book_availability(p_book_id) THEN
        RAISE EXCEPTION 'Книга % недоступна для выдачи', p_book_id;
    END IF;

    INSERT INTO loan (loan_id, book_id, visitor_id, issue_date, return_date)
    VALUES (
        (SELECT COALESCE(MAX(loan_id), 0) + 1 FROM loan),
        p_book_id,
        p_visitor_id,
        CURRENT_DATE,
        CURRENT_DATE + p_days
    );

    UPDATE book SET is_available = FALSE WHERE book_id = p_book_id;

    COMMIT;
END;
$$;

-- Процедура с курсором: генерация отчета о просрочках (проходим по должникам и начисляет штраф)
CREATE OR REPLACE PROCEDURE calculate_fines()
LANGUAGE plpgsql AS $$
DECLARE
    cur_loans CURSOR FOR
        SELECT loan_id, (CURRENT_DATE - return_date) as overdue_days
        FROM loan
        WHERE actual_return_date IS NULL AND return_date < CURRENT_DATE;

    rec RECORD;
    fine_amount DECIMAL;
BEGIN
    OPEN cur_loans;

    LOOP
        FETCH cur_loans INTO rec;
        EXIT WHEN NOT FOUND;
```

```
        fine_amount := rec.overdue_days * 100.00;

        RAISE NOTICE 'Запись выдачи #%: просрочка % дн., штраф % руб.',
                     rec.loan_id, rec.overdue_days, fine_amount;
    END LOOP;

    CLOSE cur_loans;
END;
$$;

-- ================================================================
-- Триггеры
-- ================================================================

-- Автоматическое обновление статуса книги при возврате
CREATE OR REPLACE FUNCTION tf_on_return() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.actual_return_date IS NOT NULL AND OLD.actual_return_date IS NULL THEN
        UPDATE book SET is_available = TRUE WHERE book_id = NEW.book_id;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_on_return
AFTER UPDATE ON loan
FOR EACH ROW
EXECUTE FUNCTION tf_on_return();

-- Запрет бронирования, если даты в прошлом
CREATE OR REPLACE FUNCTION tf_check_res_dates() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.start_date < CURRENT_DATE THEN
        RAISE EXCEPTION 'Нельзя бронировать книгу в прошлом!';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_check_res_dates
BEFORE INSERT ON reservation
FOR EACH ROW
EXECUTE FUNCTION tf_check_res_dates();
```

```sql
-- Логирование попадания в очередь ожидания. Если книга освободилась, проверяем очередь ожидания
CREATE OR REPLACE FUNCTION tf_check_waiting_list() RETURNS TRIGGER AS $$
DECLARE
   v_wait_id INTEGER;
BEGIN
   SELECT wait_id INTO v_wait_id
   FROM waiting_list
   WHERE book_id = NEW.book_id AND status = 'active'
   ORDER BY request_date ASC
   LIMIT 1;

   IF v_wait_id IS NOT NULL THEN
      UPDATE waiting_list SET status = 'fulfilled' WHERE wait_id = v_wait_id;
      RAISE NOTICE 'Книга % освободилась! Заявка % в очереди обработана.', NEW.book_id, v_wait_id;
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_book_became_free
AFTER UPDATE OF is_available ON book
FOR EACH ROW
WHEN (NEW.is_available = TRUE)
EXECUTE FUNCTION tf_check_waiting_list();
```

Клиентская часть

```sql
-- ================================================================
-- Запросы клиента
-- ================================================================

-- 1. Фильтры и сортировка
-- "Найти все книги издательства 'Наука', изданные до 1900 года"
SELECT title, publish_year
FROM book
WHERE publisher_id = (SELECT publisher_id FROM publisher WHERE name = 'Наука')
  AND publish_year < 1900
ORDER BY publish_year;

-- 2. Left join
-- "Показать всех авторов и книги, которые они написали (если есть)"
SELECT a.last_name, b.title
```

```sql
FROM author a
LEFT JOIN book_author ba ON a.author_id = ba.author_id
LEFT JOIN book b ON ba.book_id = b.book_id;

-- 3. Group by
-- "Сколько книг каждого издательства есть в фонде?"
SELECT p.name, COUNT(b.book_id) as book_count
FROM publisher p
JOIN book b ON p.publisher_id = b.publisher_id
GROUP BY p.name;

-- 4. Фильтрация групп через having
-- "Издательства, у которых общая стоимость книг превышает 200000"
SELECT p.name, SUM(b.estimated_value) as total_value
FROM publisher p
JOIN book b ON p.publisher_id = b.publisher_id
GROUP BY p.name
HAVING SUM(b.estimated_value) > 200000;

-- 5. Подзапрос в SELECT
-- "Список книг с указанием, насколько их стоимость выше средней по библиотеке"
SELECT title, estimated_value,
       (estimated_value - (SELECT AVG(estimated_value) FROM book)) as diff_from_avg
FROM book;

-- 6. Подзапрос в WHERE
-- "Книги авторов, родившихся в 19 веке (1801-1900)"
SELECT title
FROM book
WHERE book_id IN (
    SELECT book_id
    FROM book_author ba
    JOIN author a ON ba.author_id = a.author_id
    WHERE a.birth_year BETWEEN 1801 AND 1900
);

-- 7. Подзапрос с EXISTS
-- "Посетители, которые никогда не брали книги (нет записей в loan)"
SELECT last_name, first_name
FROM visitor v
WHERE NOT EXISTS (SELECT 1 FROM loan l WHERE l.visitor_id = v.visitor_id);

-- 8. Соединение запросов через Union
```

```sql
-- "Общий список всех людей в базе (Авторы + Посетители) для рассылки поздравлений"
SELECT first_name, last_name, 'Author' as role FROM author
UNION
SELECT first_name, last_name, 'Visitor' as role FROM visitor
ORDER BY last_name;

-- 9. Update с подзапросом
-- "Поднять оценочную стоимость на 10% для книг самого популярного издательства"
UPDATE book
SET estimated_value = estimated_value * 1.10
WHERE publisher_id = (
    SELECT publisher_id
    FROM book
    GROUP BY publisher_id
    ORDER BY COUNT(*) DESC
    LIMIT 1
);

-- 10. Удаление с подзапросом
-- "Удалить из очереди ожидания заявки, которым больше года"
DELETE FROM waiting_list
WHERE request_date < (CURRENT_TIMESTAMP - INTERVAL '1 year');

-- 11. Использование созданного view
-- "Получить список особо ценных книг из представления каталога"
SELECT title, authors, publisher
FROM v_full_catalog
WHERE category = 'Особо ценная';

-- 12. Получение самой популярная книги по "количеству выдач"
SELECT b.title, COUNT(l.loan_id) as loans_cnt
FROM book b
JOIN loan l ON b.book_id = l.book_id
GROUP BY b.title
ORDER BY loans_cnt DESC
LIMIT 1;
```

ПРИЛОЖЕНИЕ: Создание и заполнение базы данных

```sql
-- ============================================================
-- Создание таблиц
-- ============================================================
```

```sql
-- Издательства
CREATE TABLE IF NOT EXISTS publisher (
    publisher_id    INTEGER         NOT NULL,
    name            VARCHAR(100)    NOT NULL UNIQUE,
    city            VARCHAR(50),
    CONSTRAINT publisher_pk PRIMARY KEY (publisher_id)
);

-- Авторы
CREATE TABLE IF NOT EXISTS author (
    author_id       INTEGER         NOT NULL,
    last_name       VARCHAR(50)     NOT NULL,
    first_name      VARCHAR(50)     NOT NULL,
    middle_name     VARCHAR(50),
    birth_year      INTEGER         CHECK (birth_year > 0 AND birth_year < 2025),
    CONSTRAINT author_pk PRIMARY KEY (author_id)
);

-- Посетители
CREATE TABLE IF NOT EXISTS visitor (
    visitor_id      INTEGER         NOT NULL,
    last_name       VARCHAR(50)     NOT NULL,
    first_name      VARCHAR(50)     NOT NULL,
    passport_num    VARCHAR(20)     NOT NULL UNIQUE,
    phone           VARCHAR(20)     NOT NULL,
    email           VARCHAR(100),
    registration_date DATE          NOT NULL DEFAULT CURRENT_DATE,
    CONSTRAINT visitor_pk PRIMARY KEY (visitor_id)
);

-- Книги
CREATE TABLE IF NOT EXISTS book (
    book_id         INTEGER         NOT NULL,
    title           VARCHAR(200)    NOT NULL,
    publisher_id    INTEGER,
    publish_year    INTEGER         CHECK (publish_year <= EXTRACT(YEAR FROM CURRENT_DATE)),
    circulation     INTEGER         CHECK (circulation > 0),
    estimated_value DECIMAL(10, 2)  CHECK (estimated_value > 0),
    is_available    BOOLEAN         NOT NULL DEFAULT TRUE,
    CONSTRAINT book_pk PRIMARY KEY (book_id)
);
```

```sql
-- Книги-Авторы
CREATE TABLE IF NOT EXISTS book_author (
    book_id         INTEGER         NOT NULL,
    author_id       INTEGER         NOT NULL,
    CONSTRAINT book_author_pk PRIMARY KEY (book_id, author_id)
);

-- Выдача книг
CREATE TABLE IF NOT EXISTS loan (
    loan_id             INTEGER     NOT NULL,
    book_id             INTEGER     NOT NULL,
    visitor_id          INTEGER     NOT NULL,
    issue_date          DATE        NOT NULL DEFAULT CURRENT_DATE,
    return_date         DATE,       -- Планируемая дата возврата
    actual_return_date  DATE,       -- Фактическая дата
    CONSTRAINT loan_pk PRIMARY KEY (loan_id),
    CONSTRAINT chk_loan_dates CHECK (actual_return_date >= issue_date OR actual_return_date IS NULL)
);

-- Бронирование
CREATE TABLE IF NOT EXISTS reservation (
    reservation_id  INTEGER         NOT NULL,
    book_id         INTEGER         NOT NULL,
    visitor_id      INTEGER         NOT NULL,
    start_date      DATE            NOT NULL,
    end_date        DATE            NOT NULL,
    created_at      TIMESTAMP       DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT reservation_pk PRIMARY KEY (reservation_id),
    CONSTRAINT chk_res_dates CHECK (end_date >= start_date)
);

-- Очередь ожидания
CREATE TABLE IF NOT EXISTS waiting_list (
    wait_id         INTEGER         NOT NULL,
    book_id         INTEGER         NOT NULL,
    visitor_id      INTEGER         NOT NULL,
    request_date    TIMESTAMP       NOT NULL DEFAULT CURRENT_TIMESTAMP,
    status          VARCHAR(20)     DEFAULT 'active' CHECK (status IN ('active', 'fulfilled', 'cancelled')),
    CONSTRAINT waiting_list_pk PRIMARY KEY (wait_id)
);

-- ============================================================
-- Добавление FK
```

```
-- ============================================================

ALTER TABLE book
ADD CONSTRAINT FK_Book_Publisher
FOREIGN KEY (publisher_id) REFERENCES publisher(publisher_id) ON DELETE SET NULL;

ALTER TABLE book_author
ADD CONSTRAINT FK_BookAuthor_Book
FOREIGN KEY (book_id) REFERENCES book(book_id) ON DELETE CASCADE;

ALTER TABLE book_author
ADD CONSTRAINT FK_BookAuthor_Author
FOREIGN KEY (author_id) REFERENCES author(author_id) ON DELETE CASCADE;

ALTER TABLE loan
ADD CONSTRAINT FK_Loan_Book
FOREIGN KEY (book_id) REFERENCES book(book_id);

ALTER TABLE loan
ADD CONSTRAINT FK_Loan_Visitor
FOREIGN KEY (visitor_id) REFERENCES visitor(visitor_id);

ALTER TABLE reservation
ADD CONSTRAINT FK_Reservation_Book
FOREIGN KEY (book_id) REFERENCES book(book_id) ON DELETE CASCADE;

ALTER TABLE reservation
ADD CONSTRAINT FK_Reservation_Visitor
FOREIGN KEY (visitor_id) REFERENCES visitor(visitor_id) ON DELETE CASCADE;

ALTER TABLE waiting_list
ADD CONSTRAINT FK_Waiting_Book
FOREIGN KEY (book_id) REFERENCES book(book_id) ON DELETE CASCADE;

ALTER TABLE waiting_list
ADD CONSTRAINT FK_Waiting_Visitor
FOREIGN KEY (visitor_id) REFERENCES visitor(visitor_id) ON DELETE CASCADE;

-- ============================================================
-- Индексы
-- ============================================================
CREATE INDEX idx_book_title ON book(title);
CREATE INDEX idx_loan_active ON loan(visitor_id) WHERE actual_return_date IS NULL;
```

```sql
CREATE INDEX idx_reservation_dates ON reservation(book_id, start_date, end_date);
CREATE INDEX idx_visitor_lastname ON visitor(last_name);

-- ================================================================
-- Заполнение данными
-- ================================================================

INSERT INTO publisher(publisher_id, name, city) VALUES
(1, 'Наука', 'Москва'),
(2, 'Просвещение', 'Москва'),
(3, 'Азбука-Аттикус', 'Санкт-Петербург'),
(4, 'Academia', 'Москва'),
(5, 'Oxford University Press', 'Oxford');

INSERT INTO author(author_id, last_name, first_name, middle_name, birth_year) VALUES
(1, 'Пушкин', 'Александр', 'Сергеевич', 1799),
(2, 'Достоевский', 'Федор', 'Михайлович', 1821),
(3, 'Толстой', 'Лев', 'Николаевич', 1828),
(4, 'Булгаков', 'Михаил', 'Афанасьевич', 1891),
(5, 'Набоков', 'Владимир', 'Владимирович', 1899);

INSERT INTO book(book_id, title, publisher_id, publish_year, circulation, estimated_value, is_available) VALUES
(1, 'Евгений Онегин (Прижизненное)', 1, 1833, 1200, 500000.00, TRUE),
(2, 'Преступление и наказание', 2, 1866, 3000, 150000.00, TRUE),
(3, 'Война и мир. Том 1', 2, 1869, 4800, 200000.00, FALSE), -- На руках
(4, 'Мастер и Маргарита (Рукопись)', 4, 1966, 1, 1000000.00, TRUE),
(5, 'Lolita (First Edition)', 5, 1955, 5000, 12000.50, TRUE);

INSERT INTO book_author(book_id, author_id) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);

INSERT INTO visitor(visitor_id, last_name, first_name, passport_num, phone, email) VALUES
(1, 'Иванов', 'Иван', '4020 123456', '+79001112233', 'ivanov@mail.ru'),
(2, 'Петров', 'Петр', '4020 654321', '+79004445566', 'petrov@yandex.ru'),
(3, 'Сидорова', 'Анна', '4505 987654', '+79115556677', 'anna.sid@gmail.com');

INSERT INTO loan(loan_id, book_id, visitor_id, issue_date, return_date, actual_return_date) VALUES
(1, 1, 1, '2023-01-10', '2023-01-20', '2023-01-19'),
(2, 2, 2, '2023-02-01', '2023-02-15', '2023-02-20'),
```

```
(3, 3, 3, '2023-03-05', '2023-03-15', NULL); -- Книга еще у Сидоровой

INSERT INTO reservation(reservation_id, book_id, visitor_id, start_date, end_date) VALUES
(1, 1, 2, '2023-12-01', '2023-12-10');

-- Иванов ждет книгу 3, которая у Сидоровой
INSERT INTO waiting_list(wait_id, book_id, visitor_id, request_date, status) VALUES
(1, 3, 1, '2023-03-10 10:00:00', 'active');

-- ================================================================
-- Удаление таблиц
-- ================================================================
DROP TABLE IF EXISTS waiting_list CASCADE;
DROP TABLE IF EXISTS reservation CASCADE;
DROP TABLE IF EXISTS loan CASCADE;
DROP TABLE IF EXISTS book_author CASCADE;
DROP TABLE IF EXISTS book CASCADE;
DROP TABLE IF EXISTS visitor CASCADE;
DROP TABLE IF EXISTS author CASCADE;
DROP TABLE IF EXISTS publisher CASCADE;

-- ================================================================
-- Удаление логики: тригеры, функции, процедуры
-- ================================================================

DROP TRIGGER IF EXISTS tr_book_became_free ON book;
DROP FUNCTION IF EXISTS tf_check_waiting_list;

DROP TRIGGER IF EXISTS tr_check_res_dates ON reservation;
DROP FUNCTION IF EXISTS tf_check_res_dates;

DROP TRIGGER IF EXISTS tr_on_return ON loan;
DROP FUNCTION IF EXISTS tf_on_return;

DROP PROCEDURE IF EXISTS calculate_fines;
DROP PROCEDURE IF EXISTS register_loan;
DROP FUNCTION IF EXISTS check_book_availability;

-- ================================================================
-- Удаление представлений
-- ================================================================

DROP VIEW IF EXISTS v_visitor_stats;
```

```
DROP VIEW IF EXISTS v_active_loans;
DROP VIEW IF EXISTS v_full_catalog;
```