Разработка серверной части приложений PostgreSQL 16 **(dev-1)**

Приложение «Книжный магазин»

Схема данных и интерфейс

Темы

Обзор приложения «Книжный магазин»

Проектирование схемы данных, нормализация

Итоговая схема данных приложения

Организация интерфейса между клиентом и сервером

Приложение

Книжный магазин

Магазин | Авторы | Книги | Каталог

Магазин

Имя автора | Название книги | Есть на складе

Поиск

Название	Наличие	
Война и мир. Толстой Л. Н.	0	Купить
Муму. Тургенев И. С.	0	Купить
Путешествия в некоторые удаленные страны.... Свифт Д.	0	Купить
Сказка о царе Салтане. Пушкин А. С.	25	Купить
Трудно быть богом. Стругацкий А. Н., Стругацкий Б. Н.	0	Купить
Хрестоматия. Пушкин А. С., Толстой Л. Н. и др.	0	Купить

Вы купили книгу

Найдено книг: 6

Роль БД

postgres

```
select buy_book (
  book_id=>$1
) result
```

```
select * from get_catalog ($1, $2, $3)
```

PostgreSQL API (REST API PostgreSQL)

- PostgreSQL REST API — это веб-служба, которая использует HTTP-запросы, чтобы позволить приложению взаимодействовать с база данных PostgreSQL.
- Он использует стандартные методы HTTP, такие как GET, POST, PUT и DELETE, вместо традиционных языков запросов к базе данных, таких как SQL.
- API REST возвращают данные в формате JSON.
- PostgREST — это автономный веб-сервер, который может напрямую превратить базу данных PostgreSQL в RESTful API.

ПЛЮСЫ И МИНУСЫ APIPOSTGREST

У реализации **API PostgREST** есть как плюсы, так и минусы. Их понимание поможет вам максимально эффективно использовать эти **API**.

Плюсы:

- Автоматическое создание **API** сводит к минимуму усилия по написанию кода, поэтому разработчики могут вместо этого сосредоточиться на логике приложения.
- Поскольку **PostgREST** использует схему базы данных для создания конечных точек, API немедленно отразит любые изменения в базе данных.
- Инструмент оптимизирован для повышения производительности, поэтому он эффективно обрабатывает подключения к базе данных и генерацию запросов.
- Он предлагает различные меры безопасности, такие как управление доступом на основе ролей и разрешения на операции с базой данных, а также интегрируется с JWT для безопасной передачи данных.
- Быстрое прототипированию приложений, в которых структура API напрямую зависит от схемы базы данных (т. е. конечные точки, параметры и ответы API создаются на основе таблиц, столбцов и связей базы данных).

Минусы:

- PostgREST очень эффективен для базовых операций CRUD, но работа с более **сложной бизнес-логикой** может быть сложной и требовать внешней специальной разработки.
- Разработка пользовательских рабочих процессов или интеграция **PostgREST** со сторонними системами требует дополнительных инструментов или ручного написания кода.
- Для полного использования PostgREST требуется понимание его более продвинутых функций, таких как внешние ключи, транзакции, наследование и оконные функции.

Книги

ER-модель для высокоуровневого проектирования

сущности — понятия предметной области

связи — отношения между сущностями

атрибуты — свойства сущностей и связей

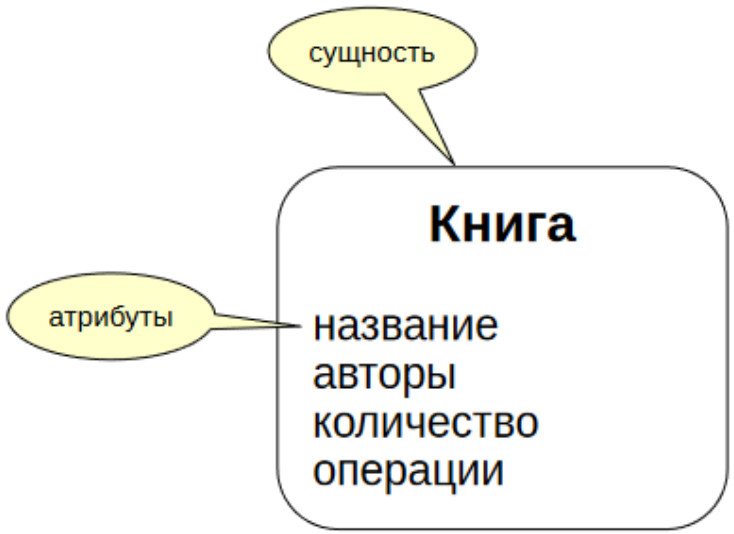
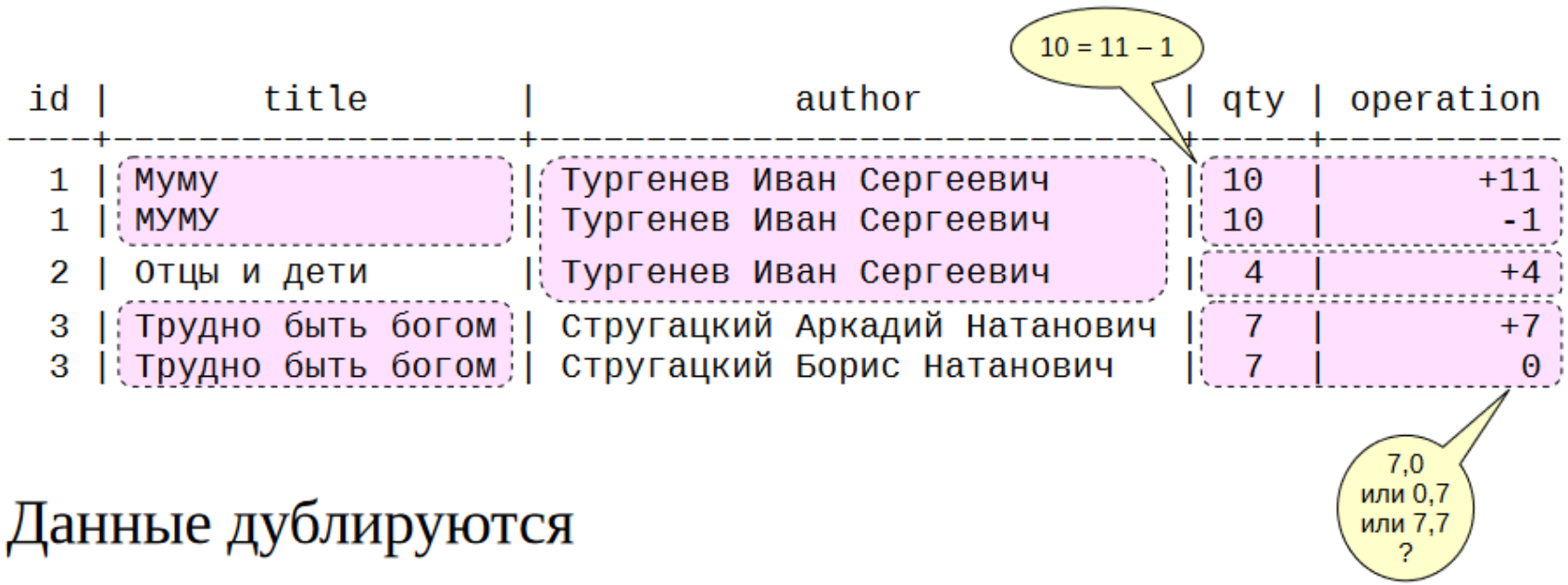


Схема данных



Данные дублируются

- сложно поддерживать согласованность
- сложно обновлять
- сложно писать запросы

Схема данных (вариант)

entity	attribute	value
1	title	Муму
1	author	Тургенев Иван Сергеевич
1	qty	10
1	operation	+11
1	operation	-1
2	title	Отцы и дети
2	author	Тургенев Иван Сергеевич
2	qty	4
2	operation	+4
...	...	...

Данные без схемы

- поддержка согласованности на стороне приложения
- сложно писать запросы
- низкая производительность (множественные соединения)

Схема данных (вариант)

```
 book_id |                          description
---------+-------------------------------------------------------
       1 | {"title": "Муму",
         |  "authors": [ "Тургенев Иван Сергеевич" ],
         |  "qty": 10,
         |  "operations": [ +11, -1 ]}
       3 | {"title": "Трудно быть богом",
         |  "authors": [ "Стругацкий Аркадий Натанович",
         |               "Стругацкий Борис Натанович" ],
         |  "qty": 7,
         |  "operations": [ +7 ]}
     ... | ...
```

Данные без схемы

- поддержка согласованности на стороне приложения
- сложно писать запросы (специальный язык)
- индексная поддержка есть

Книги и операции

Нормализация — уменьшение избыточности данных

разбиение крупных сущностей на более мелкие

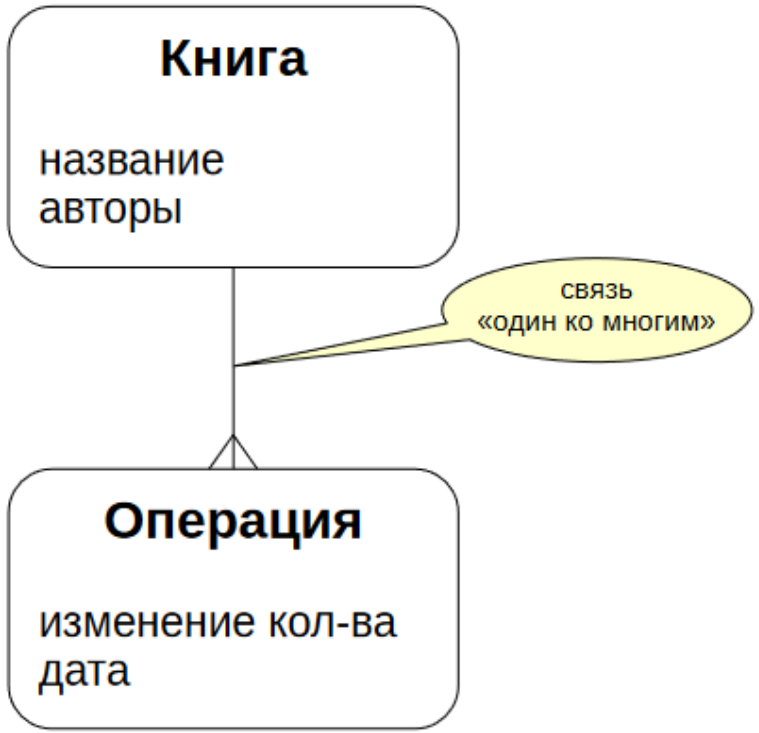


Схема данных

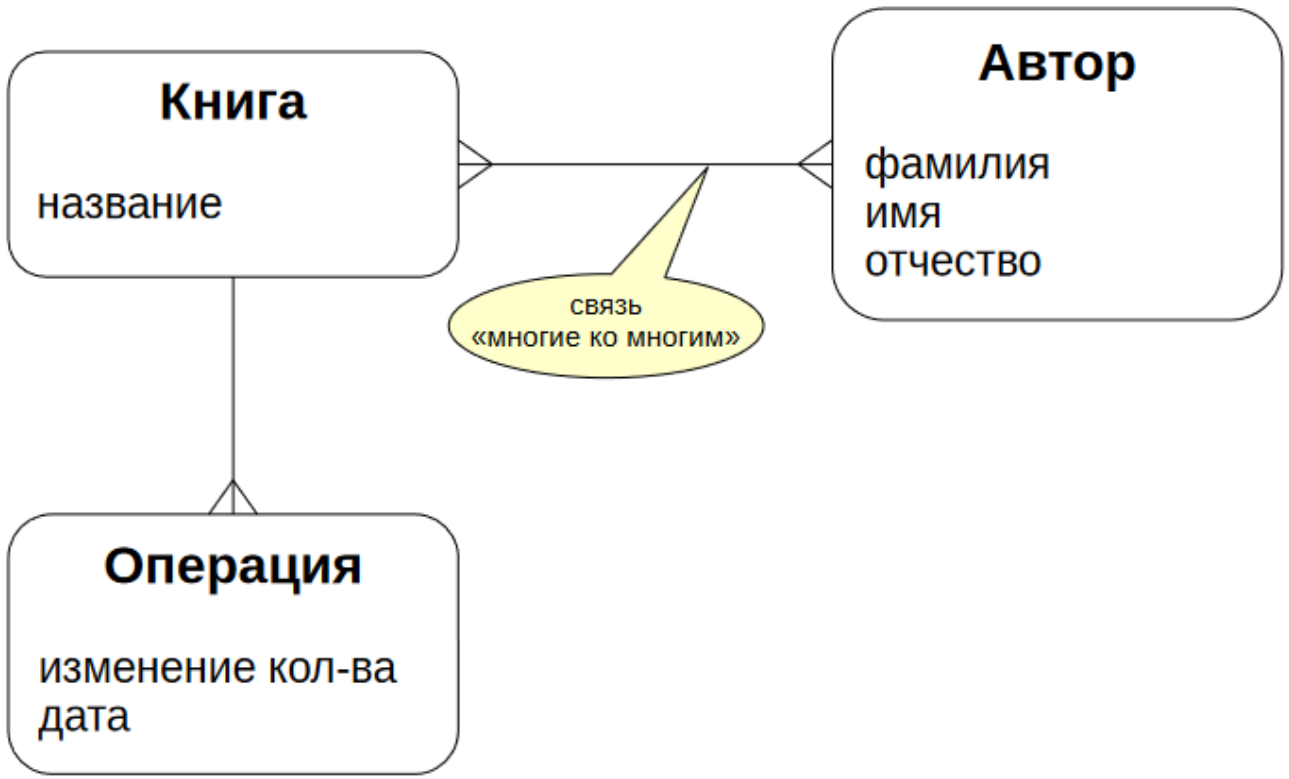
books

book_id	title	author
1	Муму	Тургенев Иван Сергеевич
2	Отцы и дети	Тургенев Иван Сергеевич
3	Трудно быть богом	Стругацкий Аркадий Натанович
3	Трудно быть богом	Стругацкий Борис Натанович

operations

operation_id	book_id	qty_change	date_created
1	1	+10	2020-07-13
2	1	-1	2020-08-25
3	3	+7	2020-07-13
4	2	+4	2020-07-13

Книги, авторы и операции



Практика

1. В базе данных bookstore создайте схему bookstore. Настройте путь поиска к этой схеме на уровне подключения к БД.
2. В схеме bookstore создайте таблицы books, authors, authorship и operations с необходимыми ограничениями целостности так, чтобы они соответствовали показанным в демонстрации.
3. Вставьте в таблицы данные о нескольких книгах. Проверьте себя с помощью запросов.
4. В схеме bookstore создайте представления authors_v, catalog_v и operations_v так, чтобы они соответствовали показанным в демонстрации.
   Проверьте, что приложение стало показывать данные на вкладках «Книги», «Авторы» и «Каталог».

Схема данных приложения состоит из четырех таблиц

```
                List of relations
  Schema   |     Name     | Type  |  Owner
-----------+--------------+-------+---------
 bookstore | authors      | table | student
 bookstore | authorship   | table | student
 bookstore | books        | table | student
 bookstore | operations   | table | student
(4 rows)
```

Книги:

```
                        Table "bookstore.books"
 Column  |  Type   | Collation | Nullable |           Default
---------+---------+-----------+----------+-------------------------------
 book_id | integer |           | not null | generated always as identity
 title   | text    |           | not null |
Indexes:
    "books_pkey" PRIMARY KEY, btree (book_id)
Referenced by:
    TABLE "authorship" CONSTRAINT "authorship_book_id_fkey" FOREIGN KEY (book_id)
REFERENCES books(book_id)
    TABLE "operations" CONSTRAINT "operations_book_id_fkey" FOREIGN KEY (book_id)
REFERENCES books(book_id)
```

Организация интерфейса

Таблицы и триггеры

чтение данных напрямую из таблицы (представления);
запись данных напрямую в таблицу (представление),
плюс триггеры для изменения связанных таблиц

приложение должно быть в курсе модели данных,
максимальная гибкость

сложно поддерживать согласованность

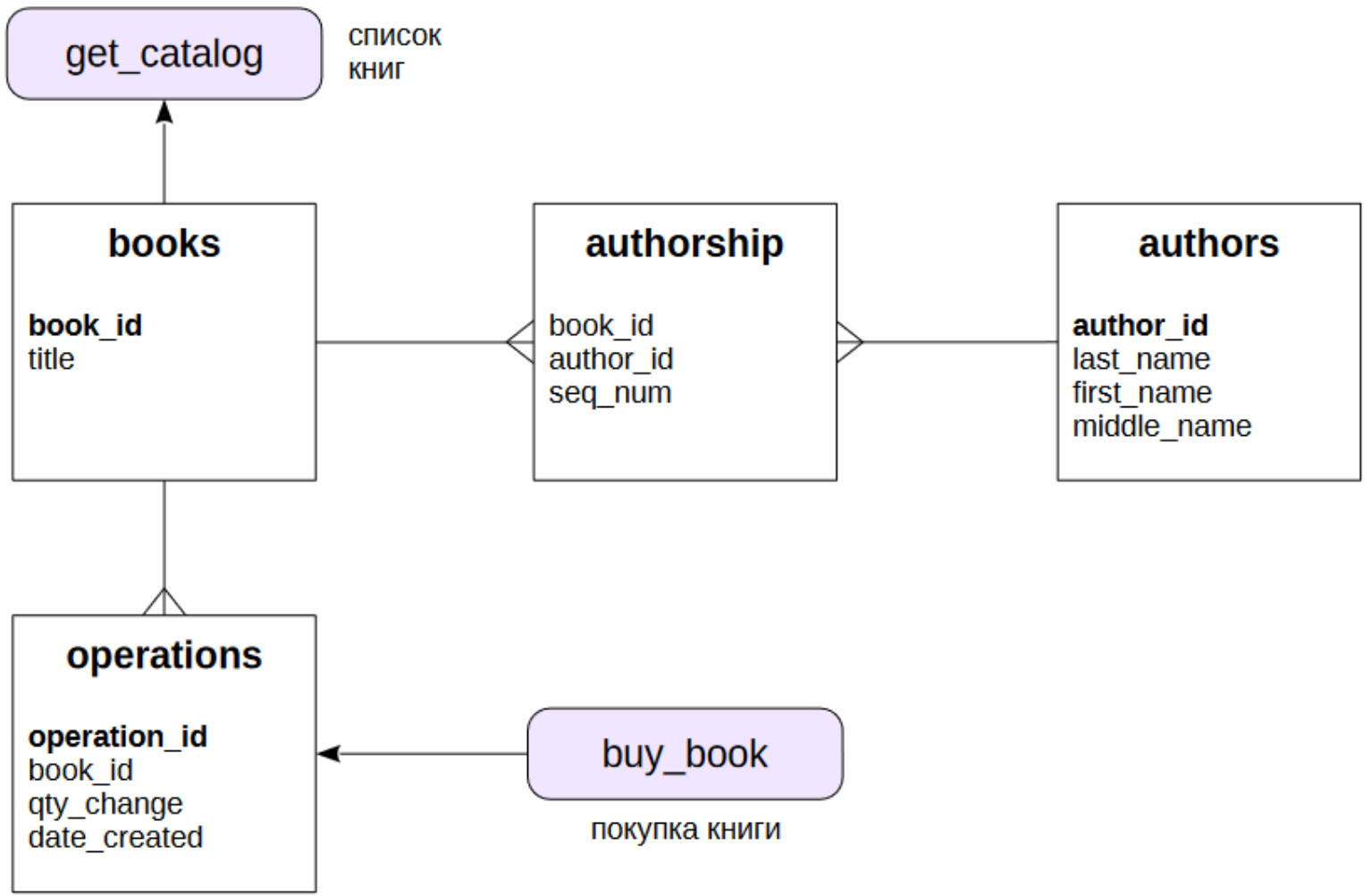
Функции

чтение данных из табличных функций;
запись данных через вызов функций

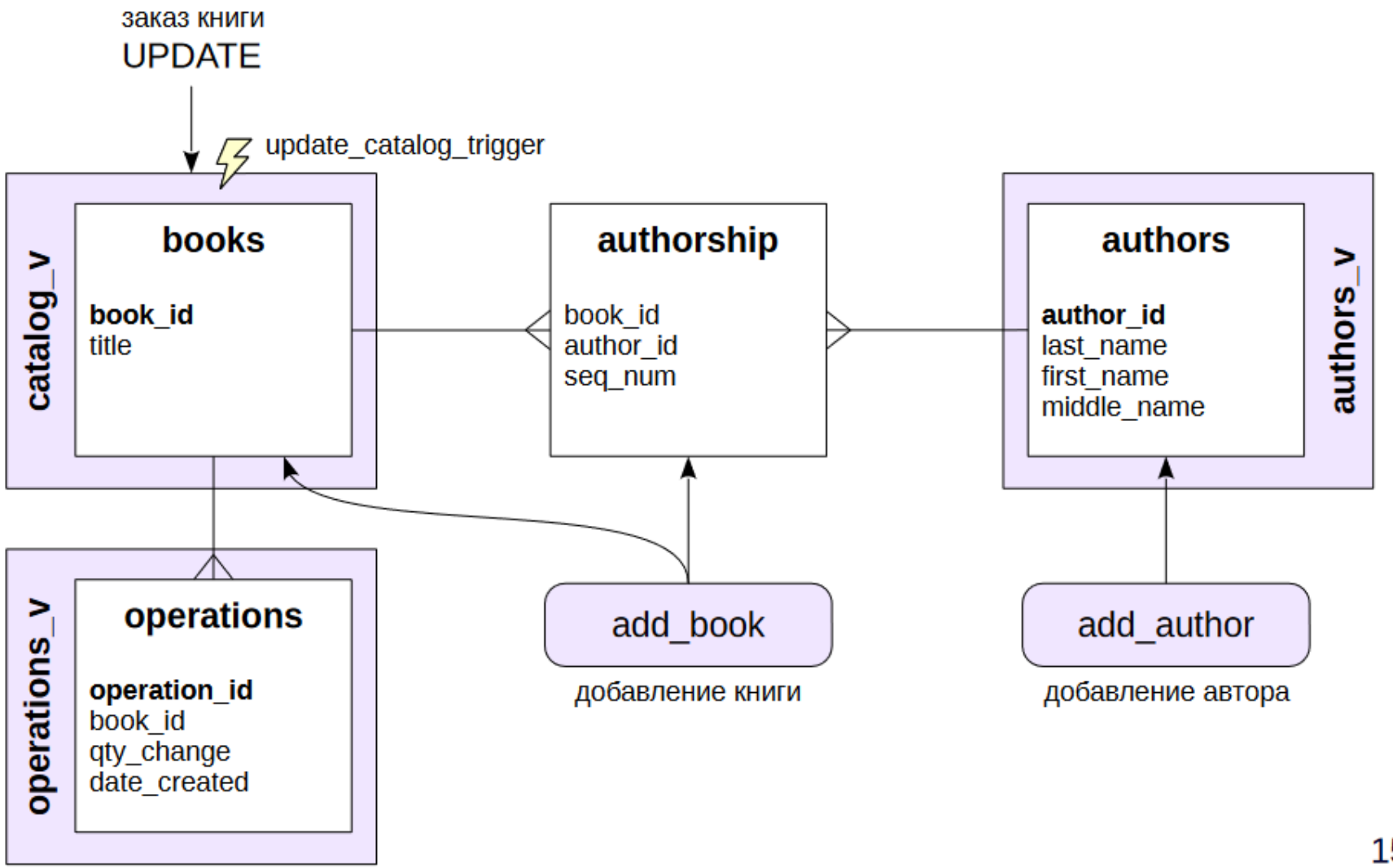
приложение отделено от модели данных и ограничено API

большой объем работы по изготовлению функций-оберток,
потенциальные проблемы с производительностью

Интерфейс покупателя



Интерфейс сотрудника



Итоги

Проектирование баз данных — отдельная тема

- теория важна, но не заменяет здравый смысл

Отсутствие избыточности в данных делает работу удобнее и упрощает поддержку согласованности

Для клиент-серверного интерфейса можно использовать таблицы, представления, функции, триггеры

Практика

1. В базе данных bookstore создайте схему bookstore. Настройте путь поиска к этой схеме на уровне подключения к БД.
2. В схеме bookstore создайте таблицы books, authors, authorship и operations с необходимыми ограничениями целостности так, чтобы они соответствовали показанным в демонстрации.
3. Вставьте в таблицы данные о нескольких книгах. Проверьте себя с помощью запросов.
4. В схеме bookstore создайте представления authors_v, catalog_v и operations_v так, чтобы они соответствовали показанным в демонстрации.
   Проверьте, что приложение стало показывать данные на вкладках «Книги», «Авторы» и «Каталог».

Практика+

1. Какие дополнительные атрибуты могут появиться у выделенных сущностей при развитии приложения?
2. Допустим, требуется хранить информацию об издательстве. Дополните ER-диаграмму и отобразите ее в таблицы.
3. Некоторые книги могут входить в серии (например, «Библиотека приключений»). Как изменится схема данных?
4. Пусть наш магазин стал торговать компьютерными комплектующими (материнскими платами, процессорами, памятью, жесткими дисками, мониторами и т. п.). Какие сущности и какие атрибуты вы бы выделили? Учтите, что на рынке постоянно появляются новые типы оборудования со своими характеристиками.