**Предметная область**

В интернет-магазине продаются книги. Каждая книга имеет название, написана одним автором, относится к одному жанру, имеет определенную цену. В магазине в наличии есть несколько экземпляров каждой книги.

Покупатель регистрируется на сайте интернет-магазина, задает свое имя и фамилию, электронную почтуи город проживания. Он может сформировать один или несколько заказов*,*для каждого заказа написать какие-то пожелания. Каждый заказ включает одну или несколько книг, каждую книгу можно заказать в нескольких экземплярах. Затем заказ проходит ряд последовательных этапов (операций): оплачивается, упаковывается, передается курьеру или транспортной компании для транспортировки и, наконец, доставляется покупателю. Фиксируется дата каждой операции. Для каждого города известно среднее время доставки книг.

При этом в магазине ведется учет книг, при покупке их количество уменьшается, при поступлении товара увеличивается, при исчерпании количества – оформляется заказ и пр.

В данном уроке сначала будет построена [концептуальная модель](https://stepik.org/lesson/308891/step/2?unit=291017) базы данных, затем ее [логическая модель](https://stepik.org/lesson/308891/step/3?unit=291017). Также будут определены[структура и содержание таблиц](https://stepik.org/lesson/308891/step/4?unit=291017) базы данных «Интернет-магазин книг».

Затем для разработанной базы данных рассматриваются следующие запросы:

1. [Вывести фамилии всех клиентов, которые заказали определенную книгу](https://stepik.org/lesson/308891/step/5?unit=291017).
2. [Посчитать, сколько раз была заказана каждая книга](https://stepik.org/lesson/308891/step/6?unit=291017).
3. [Вывести города, в которых живут клиенты магазина](https://stepik.org/lesson/308891/step/7?unit=291017).
4. [Вывести информацию об оплате каждого заказа](https://stepik.org/lesson/308891/step/8?unit=291017).
5. [Вывести подробную информацию о каждом заказе](https://stepik.org/lesson/308891/step/9?unit=291017).
6. [Вывести информацию о движении каждого заказа](https://stepik.org/lesson/308891/step/10?unit=291017).
7. [Вывести заказы, доставленные с опозданием](https://stepik.org/lesson/308891/step/11?unit=291017).
8. [Вывести клиентов, которые заказывали книги определенного автора](https://stepik.org/lesson/308891/step/12?unit=291017).
9. [Вывести самый популярный жанр](https://stepik.org/lesson/308891/step/13?unit=291017).
10. [Сравнить ежемесячную выручку за текущий и прошлый год](https://stepik.org/lesson/308891/step/14?unit=291017).

**Проектирование концептуальной модели базы данных**

**Шаг 1.**Детально проанализировать предметную область и выделить те информационные объекты, которые будут храниться в базе данных (выделены зеленым):

*В интернет-магазине продаются****книги****. Каждая книга имеет название, написана одним****автором****, относится к одному****жанру****, имеет определенную цену. В магазине в наличии есть несколько экземпляров каждой книги.*

***Покупатель****регистрируется на сайте интернет-магазина, задает свое имя и фамилию,  электронную почту и****город****проживания (для упрощения считаем, что человек не может поменять свой город проживания). Он может сформировать один или несколько****заказов****,  для каждого заказа написать какие-то пожелания. Каждый заказ включает одну или несколько книг, каждую книгу можно заказать в нескольких экземплярах. Затем заказ проходит ряд последовательных****этапов****(операций): оплачивается, упаковывается, передается курьеру или транспортной компании для транспортировки и, наконец, доставляется покупателю. Фиксируется дата каждой операции. Для каждого города известно среднее время доставки книг.*

*При этом в магазине ведется учет книг, при покупке их количество уменьшается, при поступлении товара увеличивается, при исчерпании количества – оформляется заказ и пр.*

**Шаг 2.** Для каждого выделенного информационного объекта указать его характеристики, для этого:

а) сначала выделить их в описании предметной области (синий цвет):

*В интернет-магазине продаются****книги****. Каждая книга имеет****название****, написана одним****автором****, относится к одному****жанру****, имеет определенную****цену****. В магазине в наличии есть****несколько экземпляров****каждой книги.*

***Покупатель****регистрируется на сайте интернет-магазина, задает свое****имя и фамилию****,****электронную почту****и****город****проживания. Он может сформировать один или несколько****заказов****, для каждого заказа написать какие-то****пожелания****. Каждый заказ включает****одну или несколько книг****, каждую книгу можно заказать****в нескольких экземплярах****. Затем заказ проходит ряд последовательных****этапов****(операций): оплачивается, упаковывается, передается курьеру или транспортной компании для транспортировки и, наконец, доставляется покупателю. Фиксируется****дата каждой операции****. Для каждого города известно****среднее время доставки книг****.*

*При этом в магазине ведется учет книг, при покупке их количество уменьшается, при поступлении товара увеличивается, при исчерпании количества – оформляется заказ и пр.*

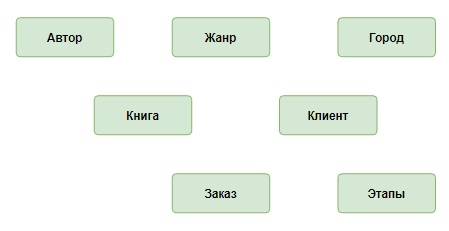
б) затем связать их с информационным объектом:

* **Книга** – название, количество, цена;
* **Автор** – фамилия и инициалы;
* **Жанр** – название;
* **Покупатель (клиент)**– фамилия и имя, электронная почта;
* **Город** – название, среднее время доставки;
* **Заказ** – код заказа, пожелания;
* **Этап** – название этапов.

в) перечислить характеристики, которые остались не привязанными к информационным объектам (к ним необходимо вернуться при реализации связей между таблицами):

* книги в заказе;
* количество книг в заказе;
* дата каждой операции.

**Шаг 3.** Нарисовать схему, на которой изобразить информационные объекты в виде прямоугольников:

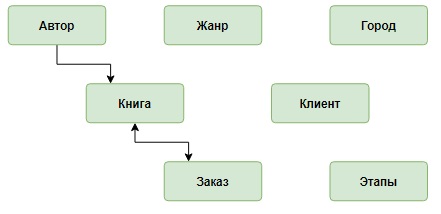


**Шаг 4.**Установить связи между информационными объектами. Связь «один ко многим» обозначить в виде →, «многие ко многим» –  ↔.

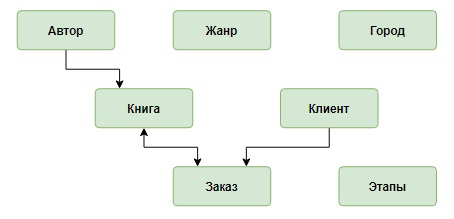
* Каждая **книга** написана одним **автором**, каждый **автор** написал несколько **книг**, следовательно между этими таблицами связь «один ко многим»:



* Каждая **книга** может включаться в несколько**заказов**, один **заказ** может содержать несколько **книг**, между этими таблицами связь «многие ко многим»:

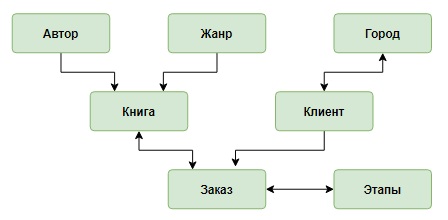
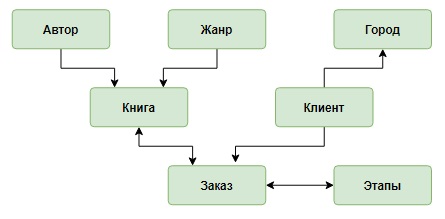


* Каждый **клиент** может сформировать несколько **заказов**, каждый**заказ** формируется только одним **клиентом**:



**Задание**

Установите связи между информационными объектами **Жанр** и **Книга**, **Город** и **Клиент**, **Заказ** и **Этапы**. Выберите верную концептуальную схему.



**Построение логической схемы базы данных**

На предыдущем шаге получена концептуальная модель базы данных:



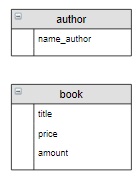
На основе этой модели создается логическая модель, в которой информационные объекты описываются в виде реляционных таблиц.

Для каждой пары таблиц необходимо выполнить следующие шаги:

**Шаг 1**. Выбрать пару таблиц из схемы, например:



**Шаг 2**. Разработать структуру таблиц для каждого информационного объекта. Таблица в качестве столбцов должна включать все характеристики информационного объекта, полученные на этапе концептуального проектирования, кроме тех, которые соответствуют названиям других информационных объектов. Они будут включены в таблицы при создании связей.  В нашем случае это:



**Шаг 3**. Реализовать связь между таблицами, в нашем случае это связь «[один ко многим](https://stepik.org/lesson/308885/step/2?unit=291011)»:



**Шаг 4.** Вернуться к описанию концептуальной модели и проверить, нужно ли включить какие-то характеристики, непривязанные к информационным объектам? В нашем случае ничего добавлять не надо. Чаще всего характеристики добавляются при реализации связи «многие ко многим».

Продолжим построение логической схемы:

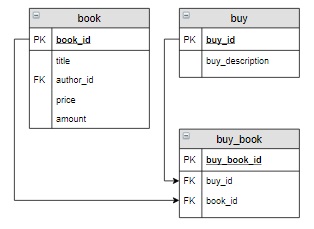
**Шаг 1.**Выберем следующую пару таблиц:



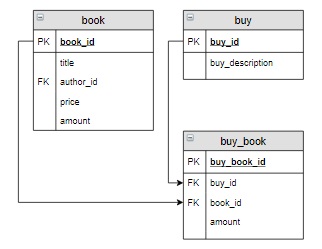
**Шаг 2**. Структура таблиц каждого информационного объекта (сохраняем уже полученные ранее структуры таблиц):



**Шаг 3**. Реализуем связь «[многие ко многим](https://stepik.org/lesson/308885/step/3?unit=291011)»:



**Шаг 4**. В описании предметной области указывается, что нужно хранить количество книг, которые включены в заказ. Добавим эту характеристику в таблицу-связку **buy\_book**:



Аналогично создаются реляционные таблицы для остальных пар информационных объектов.

**Задание**

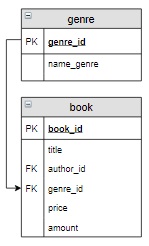
 Сопоставьте фрагменты концептуальной модели с фрагментами логической модели.



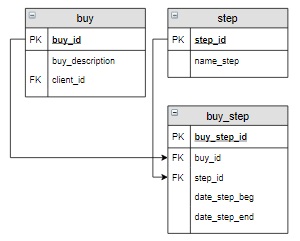


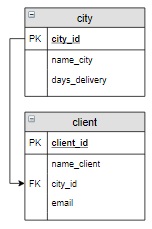






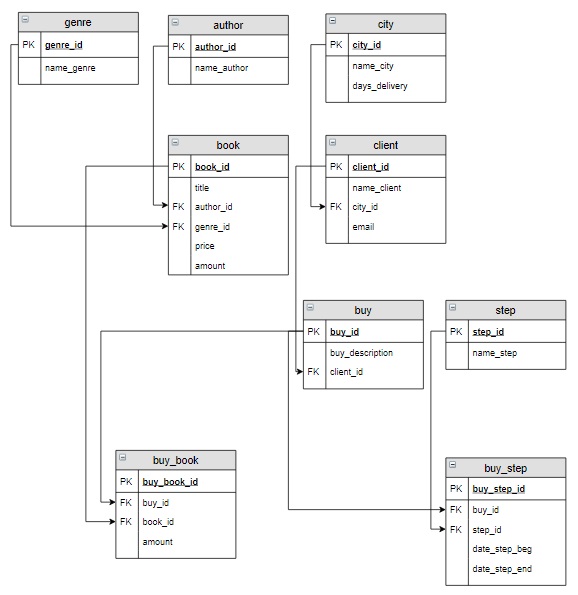






**Создание базы данных**

**​Логическая модель базы данных**

​

Логическая модель базы данных служит основой для физической модели, в которой определяются характеристики каждого столбца (тип и другие опции). После создания структуры, таблицы наполняются информацией.

**Структура и наполнение таблиц базы данных «Интернет-магазин книг»**

Таблица**author**([создание](https://stepik.org/lesson/308885/step/6?unit=291011), [заполнение](https://stepik.org/lesson/308885/step/7?unit=291011)):

|  |  |
| --- | --- |
| **author\_id** | **name\_author** |
| 1 | Булгаков М.А. |
| 2 | Достоевский Ф.М. |
| 3 | Есенин С.А. |
| 4 | Пастернак Б.Л. |
| 5 | Лермонтов М.Ю. |

Таблица**genre**([создание](https://stepik.org/lesson/297508/step/6?unit=279268), [заполнение](https://stepik.org/lesson/297508/step/7?unit=279268), рассмотрено в качестве примеров):

|  |  |
| --- | --- |
| **genre\_id** | **name\_genre** |
| 1 | Роман |
| 2 | Поэзия |
| 3 | Приключения |

Таблица **book** ([создание](https://stepik.org/lesson/308885/step/9?unit=291011), [заполнение](https://stepik.org/lesson/308885/step/11?unit=291011)):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **book\_id** | **title** | **author\_id** | **genre\_id** | **price** | **amount** |
| 1 | Мастер и Маргарита | 1 | 1 | 670.99 | 3 |
| 2 | Белая гвардия | 1 | 1 | 540.50 | 5 |
| 3 | Идиот | 2 | 1 | 460.00 | 10 |
| 4 | Братья Карамазовы | 2 | 1 | 799.01 | 2 |
| 5 | Игрок | 2 | 1 | 480.50 | 10 |
| 6 | Стихотворения и поэмы | 3 | 2 | 650.00 | 15 |
| 7 | Черный человек | 3 | 2 | 570.20 | 6 |
| 8 | Лирика | 4 | 2 | 518.99 | 2 |

Таблица **city**(в последнем столбце указано примерное количество дней, необходимое для доставки товара в каждый город):

|  |  |  |
| --- | --- | --- |
| **city\_id** | **name\_city** | **days\_delivery** |
| **INT PRIMARY KEY AUTO\_INCREMENT** | **VARCHAR(30)** | **INT** |
| 1 | Москва | 5 |
| 2 | Санкт-Петербург | 3 |
| 3 | Владивосток | 12 |

Таблица **client**:

|  |  |  |  |
| --- | --- | --- | --- |
| **client\_id** | **name\_client** | **city\_id** | **email** |
| **INT PRIMARY KEY AUTO\_INCREMENT** | **VARCHAR(50)** | **INT** | **VARCHAR(30)** |
| 1 | Баранов Павел | 3 | baranov@test |
| 2 | Абрамова Катя | 1 | abramova@test |
| 3 | Семенонов Иван | 2 | semenov@test |
| 4 | Яковлева Галина | 1 | yakovleva@test |

Таблица **buy**(столбец **buy\_description** предназначен для пожеланий покупателя, которые он хочет добавить в свой заказ, если пожеланий нет - поле остается пустым):

|  |  |  |
| --- | --- | --- |
| **buy\_id** | **buy\_description** | **client\_id** |
| **INT PRIMARY KEY AUTO\_INCREMENT** | **VARCHAR(100)** | **INT** |
| 1 | Доставка только вечером | 1 |
| 2 |  | 3 |
| 3 | Упаковать каждую книгу по отдельности | 2 |
| 4 |  | 1 |

Таблица **buy\_book**:

|  |  |  |  |
| --- | --- | --- | --- |
| **buy\_book\_id** | **buy\_id** | **book\_id** | **amount** |
| **INT PRIMARY KEY AUTO\_INCREMENT** | **INT** | **INT** | **INT** |
| 1 | 1 | 1 | 1 |
| 2 | 1 | 7 | 2 |
| 3 | 1 | 3 | 1 |
| 4 | 2 | 8 | 2 |
| 5 | 3 | 3 | 2 |
| 6 | 3 | 2 | 1 |
| 7 | 3 | 1 | 1 |
| 8 | 4 | 5 | 1 |

Таблица **step**:

|  |  |
| --- | --- |
| **step\_id** | **name\_step** |
| **INT PRIMARY KEY AUTO\_INCREMENT** | **VARCHAR(30)** |
| 1 | Оплата |
| 2 | Упаковка |
| 3 | Транспортировка |
| 4 | Доставка |

Таблица **buy\_step** ( если столбец **date\_step\_end**не заполнен (имеет значение Null), это означает что операция еще не выполнена, например для заказа с **id** 2, книги переданы для доставки 2020-03-02, но еще не доставлены):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **buy\_step\_id** | **buy\_id** | **step\_id** | **date\_step\_beg** | **date\_step\_end** |
| **INT PRIMARY KEY AUTO\_INCREMENT** | **INT** | **INT** | **DATE** | **DATE** |
| 1 | 1 | 1 | 2020-02-20 | 2020-02-20 |
| 2 | 1 | 2 | 2020-02-20 | 2020-02-21 |
| 3 | 1 | 3 | 2020-02-22 | 2020-03-07 |
| 4 | 1 | 4 | 2020-03-08 | 2020-03-08 |
| 5 | 2 | 1 | 2020-02-28 | 2020-02-28 |
| 6 | 2 | 2 | 2020-02-29 | 2020-03-01 |
| 7 | 2 | 3 | 2020-03-02 |  |
| 8 | 2 | 4 |  |  |
| 9 | 3 | 1 | 2020-03-05 | 2020-03-05 |
| 10 | 3 | 2 | 2020-03-05 | 2020-03-06 |
| 11 | 3 | 3 | 2020-03-06 | 2020-03-10 |
| 12 | 3 | 4 | 2020-03-11 |  |
| 13 | 4 | 1 | 2020-03-20 |  |
| 14 | 4 | 2 |  |  |
| 15 | 4 | 3 |  |  |
| 16 | 4 | 4 |  |  |

**Запросы на основе трех и более связанных таблиц**

**Пример**

Вывести фамилии всех клиентов, которые заказали книгу Булгакова «Мастер и Маргарита».

*Запрос:*

Этот запрос строится на основе нескольких таблиц, для удобства нужно определить фрагмент логической схемы базы данных, на основе которой строится запрос. В нашем случае выбираются название книги из таблицы **book** и фамилия клиента из таблицы **client**. Эти таблицы между собой непосредственно не связаны, поэтому нужно добавить «связующие» таблицы  **buy** и **buy\_book**:

Для соединения этих таблиц используется INNER JOIN. Для удобства рекомендуется связи описывать последовательно: **client**→ **buy**→ **buy\_book**→ **book**.  А для соединения использовать пару **первичный ключ** и **внешний ключ** соответствующих таблиц. Например, соединение таблиц **client**и **buy** осуществляется по условию **client.client\_id = buy.client\_id**.

Чтобы не усложнять схему, будем считать, что нам известен **id** Булгакова (это 1)

SELECT DISTINCT name\_client

FROM

client

INNER JOIN buy ON client.client\_id = buy.client\_id

INNER JOIN buy\_book ON buy\_book.buy\_id = buy.buy\_id

INNER JOIN book ON buy\_book.book\_id=book.book\_id

WHERE title ='Мастер и Маргарита' and author\_id = 1;

В запросе отбираются уникальные клиенты (DISTINCT) так как один и тот же клиент мог заказать одну и ту же книгу несколько раз.

*Результат:*

+---------------+

| name\_client |

+---------------+

| Баранов Павел |

| Абрамова Катя |

+---------------+

**Задание**

Вывести все заказы Баранова Павла (**id** заказа, какие книги, по какой цене и в каком количестве он заказал) в отсортированном по номеру заказа и названиям книг виде.

**Пояснение**

Если в нескольких таблицах столбцы называются одинаково – необходимо явно указывать из какой таблицы берется столбец. Например, столбец **amount** есть и в таблице **book**, и в таблице **buy\_book**. В запросе нужно указать количество заказанных книг, то есть **buy\_book.amount**.

**Связанные шаги**

* [выборка столбцов](https://stepik.org/lesson/297509/step/3?unit=279269);
* [соединение таблиц](https://stepik.org/lesson/308886/step/2?unit=291012);
* условие отбора ([шаг](https://stepik.org/lesson/297509/step/8?unit=279269), [шаг](https://stepik.org/lesson/297509/step/12?unit=279269));
* [сортировка](https://stepik.org/lesson/297509/step/11?unit=279269).

**Результат**

**Наполнение таблиц**

1

SELECT buy.buy\_id, book.title, book.price, buy\_book.amount

2

FROM

3

  client

4

   INNER JOIN buy ON client.client\_id = buy.client\_id

5

   INNER JOIN buy\_book ON buy.buy\_id = buy\_book.buy\_id

6

   INNER JOIN book ON book.book\_id = buy\_book.book\_id

7

WHERE client.client\_id = 1

8

ORDER BY 1, 2;

9

​

10

​

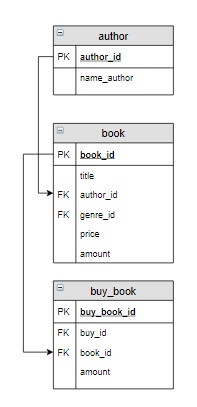
11

​

**Задание**

Посчитать, сколько раз была заказана каждая книга, для книги вывести ее автора (нужно посчитать, в каком количестве заказов фигурирует каждая книга).  Вывести фамилию и инициалы автора, название книги, последний столбец назвать **Количество**. Результат отсортировать сначала  по фамилиям авторов, а потом по названиям книг.

**Фрагмент логической схемы базы данных:**



**Пояснение**

**Связанные шаги**

**Текст задания** (чтобы не прокручивать страницу)

*Посчитать, сколько раз была заказана каждая книга, для книги вывести ее автора (нужно посчитать, в каком количестве заказов фигурирует каждая книга).  Вывести фамилию и инициалы автора, название книги, последний столбец назвать****Количество****. Результат отсортировать сначала  по фамилиям авторов, а потом по названиям книг.*

**Результат**

**Наполнение таблиц**

1

SELECT author.name\_author, book.title, COUNT(buy\_book.amount) AS 'Количество'

2

FROM author

3

   INNER JOIN book ON author.author\_id = book.author\_id

4

   LEFT JOIN buy\_book ON book.book\_id = buy\_book.book\_id

5

GROUP BY author.name\_author, book.title

6

ORDER BY 1, 2;

7

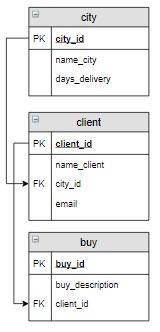
8

​

**Задание**

Вывести города, в которых живут клиенты, оформлявшие заказы в интернет-магазине. Указать количество заказов в каждый город, этот столбец назвать **Количество**. Информацию вывести по убыванию количества заказов, а затем в алфавитном порядке по названию городов.

**Фрагмент логической схемы базы данных:**



**Связанные шаги**

**Текст задания** (чтобы не прокручивать страницу) :

*Вывести города, в которых живут клиенты, оформлявшие заказы в интернет-магазине. Указать количество заказов в каждый город, этот столбец назвать****Количество****. Информацию вывести по убыванию количества заказов, а затем в алфавитном порядке по названию городов.*

**Результат**

**Наполнение таблиц**

1

SELECT name\_city, COUNT(buy.client\_id) AS 'Количество'

2

FROM city

3

   INNER JOIN client ON city.city\_id = client.city\_id

4

   INNER JOIN buy ON client.client\_id = buy.client\_id

5

GROUP BY name\_city

6

ORDER BY 2 DESC, 1 ;

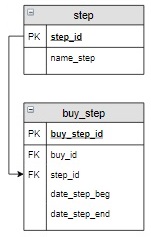
7

​

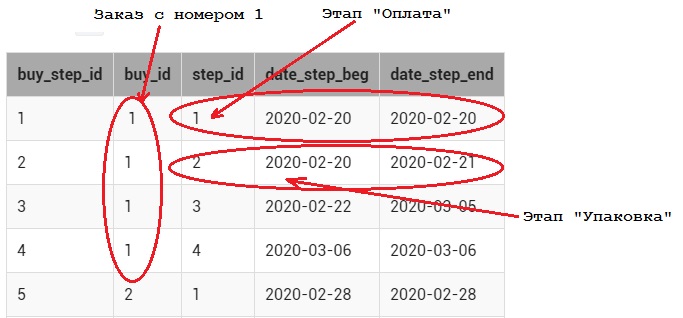
**Задание**

Вывести номера всех оплаченных заказов и даты, когда они были оплачены.

**Фрагмент логической схемы базы данных:**

​​

**Пояснение**



**Связанные шаги**

**Текст задания** (чтобы не прокручивать страницу) :

*Вывести номера всех оплаченных заказов и даты, когда они были оплачены.*

**Результат**

**Наполнение таблиц**

1

SELECT buy\_id, date\_step\_end

2

FROM step

3

   INNER JOIN buy\_step ON step.step\_id = buy\_step.step\_id

4

WHERE name\_step = 'Оплата' AND

5

    date\_step\_end IS NOT NULL

6

;

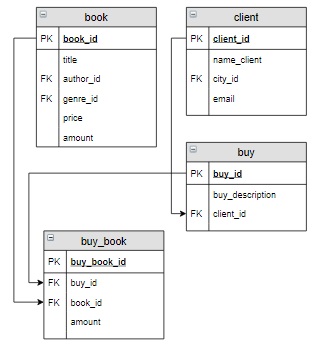
7

​

**Задание**

Вывести информацию о каждом заказе: его номер, кто его сформировал (фамилия пользователя) и его стоимость (сумма произведений количества заказанных книг и их цены), в отсортированном по номеру заказа виде. Последний столбец назвать **Стоимость**.

**Фрагмент логической схемы базы данных:**

​

**Связанные шаги**

**Текст задания** (чтобы не прокручивать страницу) :

*Вывести информацию о каждом заказе: его номер, кто его сформировал (фамилия пользователя) и его стоимость (сумма произведений количества заказанных книг и их цены), в отсортированном по номеру заказа виде. Последний столбец назвать****Стоимость****.*

**Результат**

**Наполнение таблиц**

1

SELECT buy.buy\_id, name\_client, SUM(buy\_book.amount \* price) AS 'Стоимость'

2

FROM client

3

   JOIN buy ON client.client\_id = buy.client\_id

4

   JOIN buy\_book ON buy.buy\_id = buy\_book.buy\_id

5

   JOIN book ON book.book\_id = buy\_book.book\_id

6

GROUP BY buy.buy\_id

7

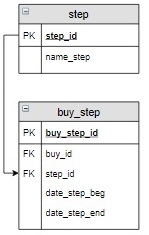
ORDER BY 1;

**Задание**

Вывести номера заказов (**buy\_id**) и названия этапов,  на которых они в данный момент находятся. Если заказ доставлен –  информацию о нем не выводить. Информацию отсортировать по возрастанию **buy\_id**.

**Пояснение**

**Фрагмент логической схемы базы данных:**



**Связанные шаги**

**Текст задания** (чтобы не прокручивать страницу) :

*Вывести номера заказов (****buy\_id****) и названия этапов, на которых они в данный момент находятся. Если заказ доставлен –  информацию о нем не выводить. Информацию отсортировать по возрастанию****buy\_id****.*

**Результат**

**Наполнение таблиц**

1

SELECT buy\_id, name\_step

2

FROM step

3

   JOIN buy\_step ON step.step\_id = buy\_step.step\_id

4

WHERE date\_step\_beg IS NOT NULL AND date\_step\_end IS NULL

5

ORDER BY 1;

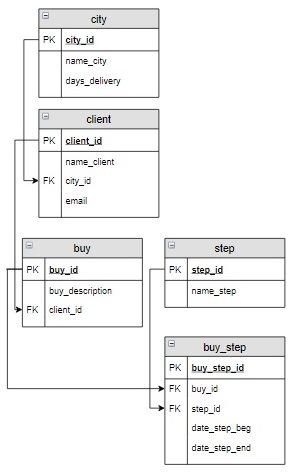
6

​

**Задание**

В таблице **city** для каждого города указано количество дней, за которые заказ может быть доставлен в этот город (рассматривается только этап **Транспортировка**). Для тех заказов, которые прошли этап транспортировки, вывести количество дней за которое заказ реально доставлен в город. А также, если заказ доставлен с опозданием, указать количество дней задержки, в противном случае вывести 0. В результат включить номер заказа (**buy\_id**), а также вычисляемые столбцы **Количество\_дней** и **Опоздание**. Информацию вывести в отсортированном по номеру заказа виде.

**Фрагмент логической схемы базы данных:**



**Пояснение**

**Связанные шаги**

**Текст задания** (чтобы не прокручивать страницу) :

*В таблице****city****для каждого города указано количество дней, за которые заказ может быть доставлен в этот город (рассматривается только этап "Транспортировка"). Для тех заказов, которые прошли этап транспортировки, вывести количество дней за которое заказ реально доставлен в город. А также, если заказ доставлен с опозданием, указать количество дней задержки, в противном случае вывести 0. В результат включить номер заказа (****buy\_id****), а также вычисляемые столбцы****Количество\_дней****и****Опоздание****. Информацию вывести в отсортированном по номеру заказа виде.*

**Результат**

**Наполнение таблиц**

1

SELECT buy.buy\_id,

2

      DATEDIFF(date\_step\_end, date\_step\_beg) AS 'Количество\_дней',

3

      IF(DATEDIFF(date\_step\_end, date\_step\_beg) > days\_delivery, DATEDIFF(date\_step\_end,         date\_step\_beg) - days\_delivery, 0) AS 'Опоздание'

4

FROM city

5

   JOIN client ON city.city\_id = client.city\_id

6

   JOIN buy ON client.client\_id = buy.client\_id

7

   JOIN buy\_step ON buy.buy\_id = buy\_step.buy\_id

8

   JOIN step ON step.step\_id = buy\_step.step\_id

9

WHERE step.step\_id = 3 AND date\_step\_beg IS NOT NULL AND date\_step\_end IS NOT NULL

10

ORDER BY 1;

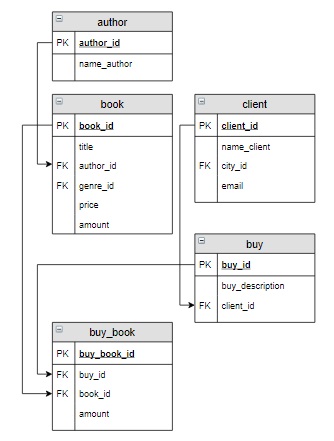
11

​

**Задание**

Выбрать всех клиентов, которые заказывали книги Достоевского, информацию вывести в отсортированном по алфавиту виде. В решении используйте фамилию автора, а не его **id**.

**Фрагмент логической схемы:**



**Связанные шаги**

**Текст задания** (чтобы не прокручивать страницу) :

*Выбрать всех клиентов, которые заказывали книги Достоевского, информацию вывести в отсортированном по алфавиту виде. В решении используйте фамилию автора, а не его****id****.*

**Результат**

**Наполнение таблиц**

1

SELECT name\_client

2

FROM author

3

   JOIN book ON author.author\_id = book.author\_id

4

   JOIN buy\_book ON book.book\_id = buy\_book.book\_id

5

   JOIN buy ON buy.buy\_id = buy\_book.buy\_id

6

   JOIN client ON client.client\_id = buy.client\_id

7

GROUP BY name\_client, name\_author

8

HAVING name\_author LIKE 'Достоевск%'

9

ORDER BY 1;

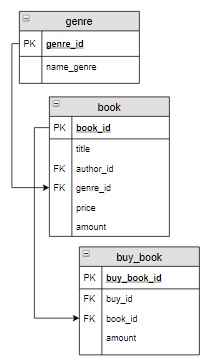
10

​

**Задание**

Вывести жанр (или жанры), в котором было заказано больше всего экземпляров книг, указать это количество. Последний столбец назвать **Количество**.

**Фрагмент логической схемы:**



**Пояснение**

**Связанные шаги**

**Текст задания** (чтобы не прокручивать страницу) :

*Вывести жанр (или жанры), в котором было заказано больше всего экземпляров книг, указать это количество . Последний столбец назвать****Количество****.*

**Результат**

**Наполнение таблиц**

1

SELECT name\_genre, SUM(buy\_book.amount) AS 'Количество'

2

FROM genre

3

   JOIN book ON genre.genre\_id = book.genre\_id

4

   JOIN buy\_book ON book.book\_id = buy\_book.book\_id

5

GROUP BY name\_genre

6

HAVING SUM(buy\_book.amount) =

7

(

8

   SELECT MAX(Количество) AS max\_sum\_amount

9

   FROM

10

   (

11

       SELECT name\_genre, SUM(buy\_book.amount) AS 'Количество'

12

       FROM genre

13

       JOIN book ON genre.genre\_id = book.genre\_id

14

       JOIN buy\_book ON book.book\_id = buy\_book.book\_id

15

       GROUP BY name\_genre

16

   ) люба\_назва

17

);

18

​

19

/\*OR SO\*/

20

​

21

SELECT name\_genre, SUM(buy\_book.amount) AS 'Количество'

22

FROM genre

23

    JOIN book ON genre.genre\_id = book.genre\_id

24

    JOIN buy\_book ON book.book\_id = buy\_book.book\_id

25

GROUP BY name\_genre

26

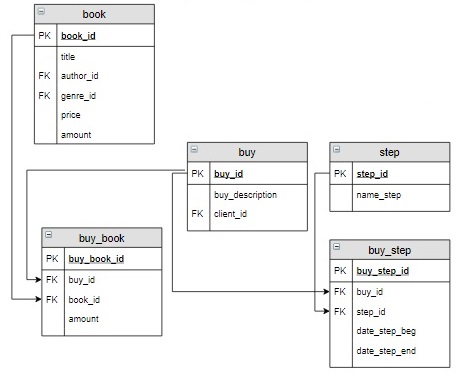
LIMIT 1;

**Задание**

*Этот шаг добавлен по рекомендациям пользователей:*[*Тимур Timmmyyy*](https://stepik.org/users/37374841)*,*[*Todor Illia*](https://stepik.org/users/33149099)*,*[*Лёха Last name*](https://stepik.org/users/123836326)*,*[*Игорь Владимирович Лапшин*](https://stepik.org/users/58247085)*и др.*

Сравнить ежемесячную выручку от продажи книг за текущий и предыдущий годы. Для этого вывести год, месяц, сумму выручки в отсортированном сначала по возрастанию месяцев, затем по возрастанию лет виде. Название столбцов: **Год**, **Месяц**, **Сумма**.

**Фрагмент логической схемы базы данных (в запросе НЕ ОБЯЗАТЕЛЬНО использовать все таблицы):**



Информация о продажах предыдущего года хранится в архивной таблице **buy\_archive**, которая создается в конце года на основе информации из таблиц базы данных и имеет следующую структуру:

|  |  |
| --- | --- |
| **Название столбца** | **Описание** |
| **buy\_archive\_id** | ключевой столбец |
| **buy\_id** | **id**заказов, выбирается из таблицы **buy** |
| **client\_id** | **id**клиентов, выбирается из из таблицы **client** |
| **book\_id** | **id**книги, выбирается из таблицы**book** |
| **date\_payment** | дата оплаты заказа, выбирается из столбца **date\_step\_end** таблицы **buy\_step** этапа «Оплата» соответствующего заказа |
| **price** | цена книги в текущем заказе из таблицы **book** (хранится, так как цена может измениться ) |
| **amount** | количество купленных книг в текущем заказе, из таблицы **buy\_book** |

**Оператор UNION**

Оператор **UNION** используется для объединения двух и более SQL запросов, его синтаксис:

SELECT столбец\_1\_1, столбец\_1\_2, ...

FROM

...

UNION

SELECT столбец\_2\_1, столбец\_2\_2, ...

FROM

...

или

SELECT столбец\_1\_1, столбец\_1\_2, ...

FROM

  ...

UNION ALL

SELECT столбец\_2\_1, столбец\_2\_2, ...

FROM

  ...

Важно отметить, что каждый из **SELECT** должен иметь в своем запросе одинаковое количество столбцов и  совместимые типы возвращаемых данных. Каждый запрос может включать разделы **WHERE**, **GROUP BY** и пр.

В результате выполнения этой конструкции будет выведена таблица, имена столбцов которой соответствуют именам столбцов в первом запросе. А в таблице результата сначала отображаются записи-результаты первого запроса, а затем второго. Если указано ключевое слово **ALL**, то в результат включаются все записи запросов, в противном случае - различные.

**Пример**

Вывести всех клиентов, которые делали заказы или в этом, или в предыдущем году.

На этом примере рассмотрим разницу между **UNION** и **UNION ALL**.

С **UNION**клиенты будут выведены без повторений:

SELECT name\_client

FROM

buy\_archive

INNER JOIN client USING(client\_id)

UNION

SELECT name\_client

FROM

buy

INNER JOIN client USING(client\_id)

+-----------------+

| name\_client |

+-----------------+

| Баранов Павел |

| Абрамова Катя |

| Яковлева Галина |

| Семенонов Иван |

+-----------------+

Affected rows: 4

C **UNION ALL** будут выведены клиенты с повторением (для тех, кто заказывал книги в обоих годах, а также несколько раз в одном году)

SELECT name\_client

FROM

buy\_archive

INNER JOIN client USING(client\_id)

UNION ALL

SELECT name\_client

FROM

buy

INNER JOIN client USING(client\_id)

+-----------------+

| name\_client |

+-----------------+

| Баранов Павел |

| Баранов Павел |

| Абрамова Катя |

| Абрамова Катя |

| Абрамова Катя |

| Яковлева Галина |

| Яковлева Галина |

| Баранов Павел |

| Абрамова Катя |

| Абрамова Катя |

| Баранов Павел |

| Баранов Павел |

| Абрамова Катя |

| Семенонов Иван |

+-----------------+

Affected rows: 14

**Пример**

Вывести информацию об оплаченных заказах за предыдущий и текущий год, информацию отсортировать по  **client\_id**.

*Запрос:*

SELECT buy\_id, client\_id, book\_id, date\_payment, amount, price

FROM

buy\_archive

UNION ALL

SELECT buy.buy\_id, client\_id, book\_id, date\_step\_end, buy\_book.amount, price

FROM

book

INNER JOIN buy\_book USING(book\_id)

INNER JOIN buy USING(buy\_id)

INNER JOIN buy\_step USING(buy\_id)

INNER JOIN step USING(step\_id)

WHERE date\_step\_end IS NOT Null and name\_step = "Оплата"

*Результат:*

+--------+-----------+---------+--------------+--------+--------+

| buy\_id | client\_id | book\_id | date\_payment | amount | price |

+--------+-----------+---------+--------------+--------+--------+

| 2 | 1 | 1 | 2019-02-21 | 2 | 670.60 |

| 2 | 1 | 3 | 2019-02-21 | 1 | 450.90 |

| 1 | 2 | 2 | 2019-02-10 | 2 | 520.30 |

| 1 | 2 | 4 | 2019-02-10 | 3 | 780.90 |

| 1 | 2 | 3 | 2019-02-10 | 1 | 450.90 |

| 3 | 4 | 4 | 2019-03-05 | 4 | 780.90 |

| 3 | 4 | 5 | 2019-03-05 | 2 | 480.90 |

| 4 | 1 | 6 | 2019-03-12 | 1 | 650.00 |

| 5 | 2 | 1 | 2019-03-18 | 2 | 670.60 |

| 5 | 2 | 4 | 2019-03-18 | 1 | 780.90 |

| 1 | 1 | 3 | 2020-02-20 | 1 | 460.00 |

| 1 | 1 | 7 | 2020-02-20 | 2 | 570.20 |

| 1 | 1 | 1 | 2020-02-20 | 1 | 670.99 |

| 2 | 3 | 8 | 2020-02-28 | 2 | 518.99 |

| 3 | 2 | 1 | 2020-03-05 | 1 | 670.99 |

| 3 | 2 | 2 | 2020-03-05 | 1 | 540.50 |

| 3 | 2 | 3 | 2020-03-05 | 2 | 460.00 |

+--------+-----------+---------+--------------+--------+--------+

В результат включены сначала записи архивной таблицы, а затем информация об оплаченных заказах  текущего года. Для того, чтобы изменить порядок следования записей в объединенном запросе, можно использовать **сортировку** по всем объединенным записям. В этом случае ключевые слова **ORDER BY** указываются после последнего запроса:

SELECT buy\_id, client\_id, book\_id, date\_payment, amount, price

FROM

buy\_archive

UNION ALL

SELECT buy.buy\_id, client\_id, book\_id, date\_step\_end, buy\_book.amount, price

FROM

book

INNER JOIN buy\_book USING(book\_id)

    INNER JOIN buy USING(buy\_id)

    INNER JOIN buy\_step USING(buy\_id)

    INNER JOIN step USING(step\_id)

WHERE  date\_step\_end IS NOT Null and name\_step = "Оплата"

ORDER BY client\_id

*Результат:*

+--------+-----------+---------+--------------+--------+--------+

| buy\_id | client\_id | book\_id | date\_payment | amount | price |

+--------+-----------+---------+--------------+--------+--------+

| 2 | 1 | 3 | 2019-02-21 | 1 | 450.90 |

| 2 | 1 | 1 | 2019-02-21 | 2 | 670.60 |

| 1 | 1 | 3 | 2020-02-20 | 1 | 460.00 |

| 1 | 1 | 7 | 2020-02-20 | 2 | 570.20 |

| 4 | 1 | 6 | 2019-03-12 | 1 | 650.00 |

| 1 | 1 | 1 | 2020-02-20 | 1 | 670.99 |

| 3 | 2 | 1 | 2020-03-05 | 1 | 670.99 |

| 3 | 2 | 2 | 2020-03-05 | 1 | 540.50 |

| 3 | 2 | 3 | 2020-03-05 | 2 | 460.00 |

| 5 | 2 | 4 | 2019-03-18 | 1 | 780.90 |

| 5 | 2 | 1 | 2019-03-18 | 2 | 670.60 |

| 1 | 2 | 3 | 2019-02-10 | 1 | 450.90 |

| 1 | 2 | 4 | 2019-02-10 | 3 | 780.90 |

| 1 | 2 | 2 | 2019-02-10 | 2 | 520.30 |

| 2 | 3 | 8 | 2020-02-28 | 2 | 518.99 |

| 3 | 4 | 5 | 2019-03-05 | 2 | 480.90 |

| 3 | 4 | 4 | 2019-03-05 | 4 | 780.90 |

+--------+-----------+---------+--------------+--------+--------+

**Связанные шаги**

**Текст задания** (чтобы не прокручивать страницу) :

*Сравнить ежемесячную выручку от продажи книг за текущий и предыдущий годы. Для этого вывести год, месяц, сумму выручки в отсортированном сначала по возрастанию месяцев, затем по возрастанию лет виде. Название столбцов:****Год****,****Месяц****,****Сумма****.*

**Пояснение**

**Результат**

**Наполнение таблиц**

1

SELECT YEAR(date\_payment) AS 'Год',

2

      MONTHNAME(date\_payment) AS 'Месяц',

3

      SUM(price \* amount) AS 'Сумма'

4

FROM

5

  buy\_archive

6

GROUP BY Год, Месяц

7

UNION

8

SELECT YEAR(date\_step\_end) AS 'Год',

9

      MONTHNAME(date\_step\_end) AS 'Месяц',

10

      SUM(book.price \* buy\_book.amount) AS 'Сумма'

11

FROM

12

  book

13

   JOIN buy\_book ON book.book\_id = buy\_book.book\_id

14

   JOIN buy ON buy.buy\_id = buy\_book.buy\_id

15

   JOIN buy\_step ON buy.buy\_id = buy\_step.buy\_id

16

   JOIN step ON step.step\_id = buy\_step.step\_id

17

/\*WHERE name\_step = 'Оплата' AND date\_step\_end IS NOT NULL\*/

18

GROUP BY Год, Месяц, name\_step

19

HAVING name\_step = 'Оплата' AND Год IS NOT NULL

20

ORDER BY 2,1;