он-лайн курс на STEPIK

Интерактивный тренажер по SQL

Дальневосточный федеральный университет

Преподаватель курса

Озерова Галина Павловна

Текстовый материал подготовила

модератор курса

Арсентьева Анастасия

Описание курса

В курсе большинство шагов — это практические задания на создание SQL-запросов. Каждый шаг включает минимальные теоретические аспекты по базам данных или языку SQL, примеры похожих запросов и пояснение к реализации.

Для создания, выполнения и отладки SQL-запросов используется платформа Stepik, на свой компьютер ничего дополнительно устанавливать не надо.

Сложность запросов возрастает по мере прохождения курса. Сначала они формулируются для отдельных таблиц, а затем для баз данных, разработанных для предметных областей, таких как "Интернет-магазин", "Тестирование", "Абитуриент", "Учебная аналитика по курсу". Причем в процессе выполнения шагов курса решаются практические задачи из выбранной предметной области.

Каждый учащийся может придумать свои задания на создание SQL-запросов. В курсе есть модуль, в котором будут размещаться лучшие из них.

Данный курс направлен на то, чтобы научить слушателя создавать базы данных и реализовывать запросы к ним на языке SQL для различных предметных областей.

Преподаватель курса

Озерова Галина Павловна

Оглавление

МОДУЛЬ 1: Основы реляционной модели и SQL

В данном модуле рассматриваются основные понятия реляционной модели, а также различные виды SQL запросов к одной таблице базы данных.

УРОК 1.1: Отношение (таблица)

Шаг 1: Структура уроков курса

Ссылка в интернете

Первый шаг каждого урока – краткий перечень рассматриваемых вопросов или типов запросов SQL, а также, при необходимости, описание структур данных, используемых в запросах.

Все **остальные шаги** – это задания, в большинстве из которых нужно написать запрос на языке SQL. На локальный компьютер ничего ставить не нужно, запросы пишутся и проверяются на платформе. Перед выполнением задания необходимо прочитать краткий теоретический материал, затем разобрать пример и посмотреть, как он выполняется. Для каждого задания приводится результат, который должен получиться после выполнения запроса, а также необходимые пояснения.

На **последнем шаге** большинства уроков нужно придумать задание для рассматриваемой структуры данных, реализовать соответствующий запрос, проверить его и при желании разместить в комментариях. В последнем модуле мы разместим лучшие запросы, и все смогут их выполнить и проголосовать за понравившиеся.

Важно! На платформе Stepik используется **MySQL**, версия 5.7.12, запросы описываются для этой системы.

Шаг 2: Содержание урока

Ссылка в интернете

- основные понятия реляционных баз данных;
- отношение (таблица);
- используемые типы данных;
- создание таблицы SQL запросом;
- добавление данных SQL запросом.

Шаг 3: Основные понятия реляционных баз данных

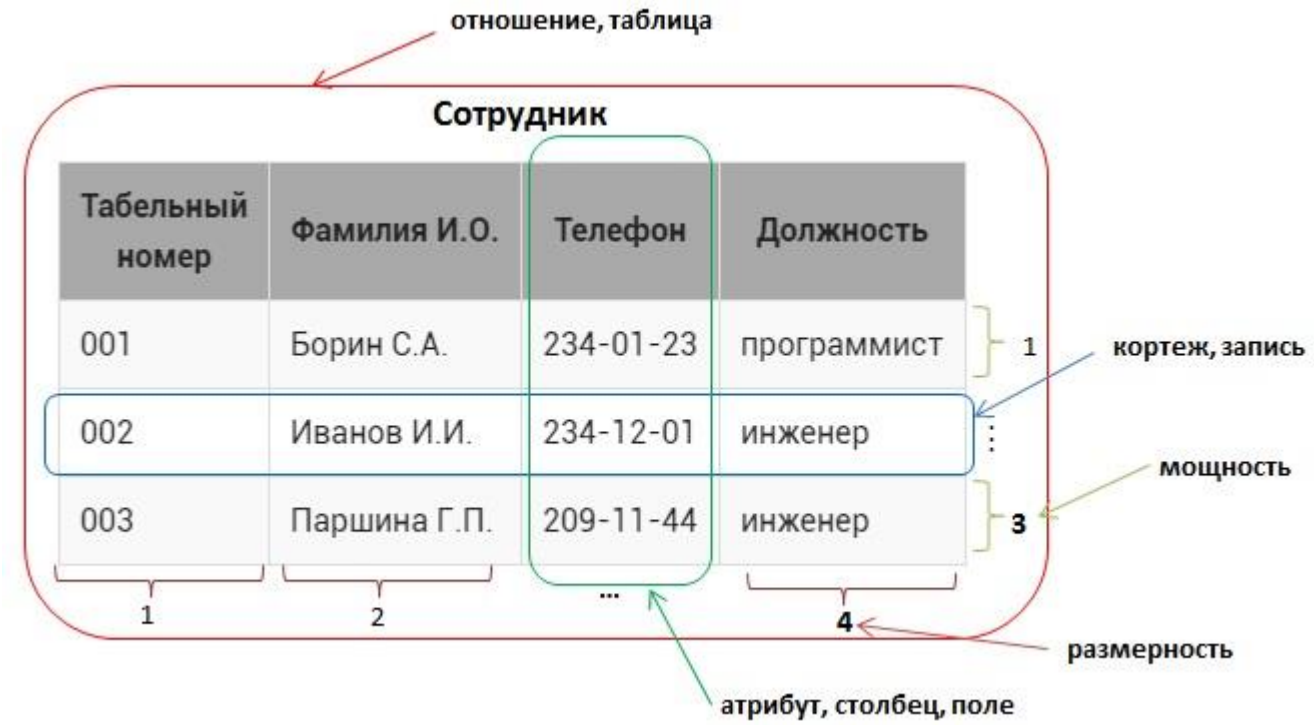
Ссылка в интернете

Реляционная модель была разработана в конце 1960-х годов Е.Ф.Коддом. Она определяет способ представления данных (структуру данных), методы защиты данных (целостность данных), и операции, которые можно выполнять с данными (манипулирование данными). Эта модель лежит в основе всех реляционных баз данных до настоящего времени.

Основные принципы реляционных баз данных:

- все данные на концептуальном уровне представляются в виде объектов, заданных в виде строк и столбцов, называемых отношением, более распространенное название – таблица;
- в пересечение строки и столбца таблицы можно занести только одно значение;
- все операции выполняются над целыми отношениями и результатом этих операций является отношение.

Пример отношения:



На примере таблицы **Сотрудник** рассмотрим **терминологию реляционных баз данных:**

- ***отношение*** – это структура данных целиком, набор записей (в обычном понимании – таблица), в примере –это **Сотрудник** ;
- ***кортеж*** – это каждая строка, содержащая данные (более распространенный термин – запись), например, <00*1, Борин С.А, 234-01-23, программист*>, все кортежи в отношении должны быть различны;
- ***мощность*** – число кортежей в таблице (проще говоря, число записей), в данном случае 3, мощность отношения может быть любой (от 0 до бесконечности), порядок следования кортежей - неважен;
- ***атрибут*** – это столбец в таблице (более распространенный термин – поле), в примере – **Табельный номер, Фамилия И.О., Телефон, Должность**)
- ***размерность*** – это число атрибутов в таблице, в данном случае – 4;
- размерность отношения должна быть больше 0, порядок следования атрибутов существенен;
- ***домен атрибута*** – это допустимые значения (неповторяющиеся), которые можно занести в поле, например, для атрибута **Должность** домен – {инженер, программист}.

Задание - Для таблицы Сотрудник отметьте верные ячейки

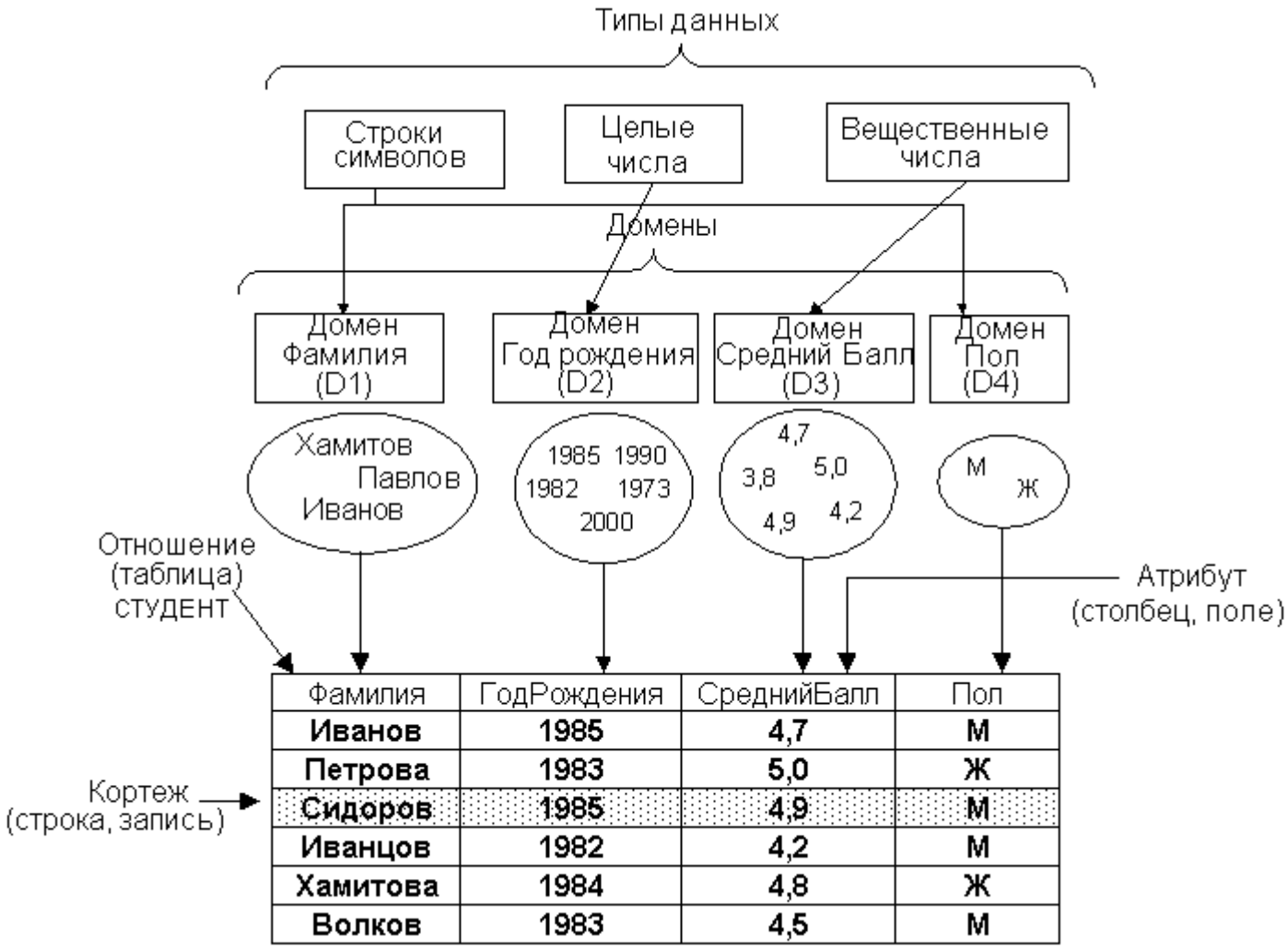
Задание - Для таблицы Сотрудник отметьте верные ячейки (В каждом столбце и строке - один правильный ответ).

Комментарии учащихся:

1.

Ilia Stepanov

Ещё такой вариант нашёл



Andrey Buynichenko

@Ilia_Stepanov, непонятно почему домен указан не для значений в кружочках

Mihail Tabakaev

@Ilia_Stepanov, как я понял из материала. Чем меньше атрибутов, тем выше вероятность наличия повторяющихся кортежей (не из-за ошибки). Следовательно, чем меньше атрибутов, тем менее мощное отношение. И обратное. В приведенном примере мощность явно оставляет желать лучшего.

2.

Игорь Смирнов

Я вот не уверен на счет домена - это ведь ограничение на допустимые значения, а здесь просто перечислены те значения, которые действительно были приняты. Это не одно и то же.

Галина Озерова

@Игорь_Смирнов, Согласна, с общим замечанием. Но для конкретного примера - это верно, как мне кажется... А как по-другому можно описать домен для этого столбца, может быть Вы предложите вариант более корректный?

Лариса Фернандес

@Галина_Озерова, может быть, так "ааа-аа-аа", где а - цифры от 0 до 9?

Галина Озерова

@Лариса_Фернандес, Спасибо большое, это отличный вариант, как мне кажется, только вот как то, что номера различны указать....

Лариса Фернандес

@Галина_Озерова, точно, где-то в лекциях CSC на ютубе слышала, что это, вроде, задается при создании поля с помощью UNIQUE

Игорь Смирнов

@Галина_Озерова, ^\d{3}-\d{2}-\d{2}$

Галина Озерова

@Игорь_Смирнов, Спасибо, это хороший вариант, тогда нужно в текст включить, что значит каждый символ в шаблоне.

Andrey Buynichenko

@Галина_Озерова, думаю, органичнее и более знакомо для сокрытия случайных символов воспринимается "•" (как в окошке ввода пароля):
{•••-••-••}, где "•" - цифра от 0 до 9

Шаг 4: Отношение, реляционная модель

Ссылка в интернете

База данных, в том числе и реляционная, используется для формального описания некоторой предметной области реального мира, например, склада, учебного процесса и пр. Обязательным этапом перед созданием базы данных является ее проектирование (этот процесс разбирается в следующих модулях).

В первом модуле будем рассматривать простейшие предметные области, информацию о которых можно описать в виде одной таблицы. Каждая такая таблица ассоциируется с неким информационным объектом или событием реального мира – человеком, документом, посещением и т.д.

Пример.

Рассмотрим некоторый склад, на котором хранятся книги. Известно название книги, ее автор, количество экземпляров на складе и ее цена.

Всю эту информацию можно представить в виде таблицы, состоящей из 4 столбцов (приведено только 4 записи, на самом деле их значительно больше):

Название	Автор	Цена, руб	Количество
Мастер и Маргарита	Булгаков М.А.	670.99	3
Белая гвардия	Булгаков М.А.	540.50	5
Идиот	Достоевский Ф.М.	460	10
Братья Карамазовы	Достоевский Ф.М.	799.01	2

Перед созданием таблицы в базе данных необходимо описать ее структуру. Для этого выполняется следующая последовательность шагов:

1. Дать таблице имя, пусть она будет называться **book**, вот некоторые ***правила для выбора имен таблиц***:

- может включать английские буквы, цифры и знак подчеркивания, должно начинаться с буквы;
- имя должно быть уникальным в пределах базы данных.

Также **рекомендуется**:

- чтобы имя было существительным в единственном числе;
- имя должно быть понятным и соответствовать тому объекту, который оно описывает;
- имя должно быть как можно короче, максимум до 10 символов.

Важно. Имена таблиц являются регистрозависимыми из-за операционной системы на которой работает **stepik**, то есть имя **book** и **Book** – разные имена. Рекомендуется для записи имен таблиц использовать только строчные (маленькие) буквы.

2. Определить структуру таблицы, из каких атрибутов (столбцов, полей) она будет состоять в нашем случае это:

- **title** – поле для хранения названия книги;
- **author** – поле с фамилией автора книги;
- **price** – цена книги;
- **amount** – количество книг.

Правила по выбору имени поля информационного объекта:

- может включать английские буквы, цифры и знак подчеркивания, должно начинаться с буквы;
- имя поля должно быть уникальным в пределах таблицы.

Рекомендации по выбору имени поля информационного объекта:

- имя должно быть понятным и соответствовать тем данным, которые хранятся в поле;
- имя может состоять из нескольких слов, тогда слова разделяются подчеркиванием, после подчеркивания слово пишется с маленькой буквы.

3. Включить ключевое поле **book_id**, которое является ОБЯЗАТЕЛЬНЫМ ЭЛЕМЕНТОМ каждой реляционной таблицы. Ключевое поле является уникальным для каждой записи, однозначно определяет запись и в дальнейшем будет использоваться для связей с другими таблицами.

Рекомендации по именованию ключевых полей:

- имя должно состоять из двух частей: начинаться с названия таблицы, которой поле принадлежит, затем через подчеркивание необходимо указать **id.**

Таким образом, наша таблица **book** будет выглядеть следующим образом:

book_id	Title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2

Задание - Отметьте ПРАВИЛЬНЫЕ имена.

Задание - Отметьте ПРАВИЛЬНЫЕ имена, которые можно выбрать в качестве названий таблиц или полей.

Комментарии учащихся:

1.
Сергей Чернятин
В описании правил именования таблиц и полей не всегда понятно, что является правилом, а что рекомендацией.
В вопросе теста тоже не совсем понятно, что означает "ВЕРНЫЕ имена" - это правильные имена или рекомендованные имена или что-то другое.
Я это понимаю так:

- **Правила** именования определяются типом базы данных и операционной системы. Правила соблюдать строго необходимо, иначе база данных будет работать неправильно или не будет работать вообще.
- **Рекомендации** по именованию это не строгие правила, а просто советы, которые нужно соблюдать для того, чтобы база данных была понятна и удобна для пользователей, программистов и других людей. Сама по себе база будет работать и при нарушении рекомендаций, но при этом она может быть совершенно неудобной и непонятной для других

2.
Siarhei Konanau
Имена таблиц являются регистрозависимыми на платформе **stepik**
Уточню, это из-за операционной системы на которой работает stepik.
Галина Озерова
@Siarhei_Konanau, Спасибо! Добавила уточнение.
Антон Сивидов
а в консоли они регистрозависимы?
Siarhei Konanau
@Антон_Сивидов, https://dev.mysql.com/doc/refman/5.7/en/identifier-case-sensitivity.html
Для mysql базы данных - папки, таблицы - файлы, поэтому чувствительность имен к регистру зависит от операционной системы (файловой системы, которая эта ОС использует) на которой запущена mysql.
На Windows у вас не получится создать два файла, например, с именами FILE и file в одной папке, для системы это одинаковые имена.
Ну а консоль тут ни при чем, так как это просто программа для взаимодействия с ОС (и другими программами) посредством текстового интерфейса.
Антон Сивидов
у меня linux)
Siarhei Konanau
@Антон_Сивидов, на нем сможете, и соответственно таблицы Books и books будут разными.

Шаг 5: Выбор типов данных для полей

Ссылка в интернете

После описания структуры таблицы необходимо выбрать типы данных для каждого поля.

Основные типы данных SQL:

Тип данных	Описание	Пример

INT INTEGER	Целое число, могут принимать значения от -2 147 483 648 до 2 147 483 647	-567 1205
DECIMAL NUMERIC	Вещественное число, в скобках указывается максимальная длина числа (включает символы слева и справа от десятичной запятой) и количество знаков после запятой. Можно использовать оба этих типа, они эквивалентны, принимают значения в диапазоне $-10^{38}+1$ до $10^{38}-1$. DECIMAL(4,1) NUMERIC(6,3)	34.6 -3.294
DATE	Дата в формате ГГГГ-ММ-ДД *26 июля 2020 года* *3 января 2021 года*	2020-07-26 2021-01-03
VARCHAR	Строка длиной не более 255 символов, в скобках указывается максимальная длина строки, которая может храниться в поле VARCHAR(10)(рассматриваются однобайтовые кодировки, для которых число в скобках соответствует максимальному количеству символов в строке)	пример описание

Рекомендации по выбору типов данных для полей таблицы.

- Выбирайте минимальный тип данных исходя из максимального значения поля. Например, если максимальный текст, который может быть записан в поле, имеет длину 25 символов, значит нужно использовать тип VARCHAR(25).
- Для описания ключевого поля используйте описание `INT PRIMARY KEY AUTO_INCREMENT`. Это значит, что в поле будут заноситься различные целые числа, при этом они будут автоматически генерироваться (каждая следующая строка будет иметь значение ключа на 1 больше предыдущего).

Определим тип данных для каждого поля таблицы **book**:

book_id	Title	Author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2

- **book_id** - ключевой столбец, целое число, которое должно генерироваться автоматически - `INT PRIMARY KEY AUTO_INCREMENT`;
- **title** - строка текста, ее длина выбирается в зависимости от данных, которые предполагается хранить в поле, предположим, что название книги не превышает 50 символов - `VARCHAR(50)`;
- **author** - строка текста - `VARCHAR(30)`;

- **price** - для описание денежного значения используется числовой тип данных с двумя знаками после запятой - DECIMAL(8,2) ;
- **amount** - целое число - INT .

Задание - Сопоставьте значения и типы данных

Задание - Сопоставьте значения и типы данных, с помощью которых их можно описать

Комментарии учащихся:

1.

Александра Корнилова
очень сложно описано DECIMAL NUMERIC
Проще говоря речь идет о десятичных дробях = в скобках указано (a, b), где:
a = сколько максимум цифр в целом числе этой дроби (2 - максимум двухзначное число, 3 - максимум трехзначное и т.д.)
b = кол-во разрядов / или оно же = кол-во цифр после запятой / или оно же = как округляем число
пример DECIMAL (3,1) = > 897, 1
пример DECIMAL (3,1) => 34,0
пример NUMERIC (9,10) = > 12345688,0000000001
корректно?
Галина Озерова
@Александра_Корнилова, **а** - это общая длина числа, без учета точки. Остальное верно.
DECIMAL (4,1) = > 897, 1
Александра Корнилова
@Галина_Озерова, о, понятно. спасибо большое !
Тарас Новохатько
@Галина_Озерова, подскажите как в таком случае будет выглядеть 5678,234 в DECIMAL? (7,3)?
Дияр Зульяров
Да, именно так

2.

Anonymous 44099323
Такой вопрос - говоря про длину поля VARCHAR, нет ни слова про кодировку текста! Правильно ли понимаю, что речь идет про однобайтовые текстовые кодировки? Поймите следующее, **если, кодировать текст в уникоде, размер поля - невозможно точно опредилить, кроме как, использовать фиксированное кодирование - 4 байта на символ!** Получается, очень накладно. В случае динамической utf-8 длина поля не фиксирована(по той причине, может быть любой символ юникода, длина символа зависит от используемого символа). Поясните пожалуйста..
Галина Озерова
@Anonymous_44099323, Спасибо, исправила указала ссылку на Ваше объяснение.
Anonymous 44099323
@Галина_Озерова, извиняюсь, мои мысли - не стандарт, поэтому, лучше внимательно почитать тут:
https://dev.mysql.com/doc/refman/5.7/en/char.html
https://mariadb.com/kb/en/varchar/
Точно не опредилился, поскольку, мне ещё, не всё понятно, нужно почитать.
Галина Озерова
@Anonymous_44099323, убрала ссылку на Ваш комментарий - но по сути Вы верно написали.
Блинов Виталий
@Галина_Озерова, здравствуйте! Ссылка на объяснение кодировки не работает(
Anonymous 44099323
@Блинов_Виталий, её не было. Ниже ссылки, где подробно описано:
http://www.mysql.ru/docs/man/CHAR.html
https://docs.microsoft.com/ru-ru/sql/t-sql/data-types/char-and-varchar-transact-sql?view=sql-server-ver15
Блинов Виталий

@Anonymous_44099323, а зачем в таблице в 3й колонке, 4й строки, синими буквами, написаны слова: ПРИМЕР, ОПИСАНИЕ?! Синие буквы обычно значат ссылки...
Галина Озерова
@Блинов_Виталий, Это текстовая константа так обозначена, изменила цвет, спасибо!

3.
Anonymous 52302811
Я думаю, что было бы полезно дать еще градацию целых чисел, раз вы говорите о поиске максимально подходящего типа данных

Шаг 6: Создание таблицы (**CREATE TABLE**)

Ссылка в интернете

На этом шаге нужно написать и проверить SQL запрос. Сначала кратко описывается структура и особенности запроса, приводится пример. А затем формулируется задание, для которого нужно реализовать запрос.

Для создания таблицы используется SQL-запрос. В нем указывается какая таблица создается, из каких атрибутов(полей) она состоит и какой тип данных имеет каждое поле, при необходимости указывается описание полей (ключевое поле и т.д.). Его структура:

- ключевые слова: `CREATE TABLE`
- имя создаваемой таблицы;
- открывающая круглая скобка «(»;
- название поля и его описание, которое включает тип поля и другие необязательные характеристики;
- запятая;
- название поля и его описание;
- ...
- закрывающая скобка «)».

Пример. Создадим таблицу **`genre`** следующей структуры:

Поле	Тип, описание
genre_id	`INT PRIMARY KEY AUTO_INCREMENT`
name_genre	`VARCHAR(30)`

Запрос:

```
CREATE TABLE genre(genre_id INT PRIMARY KEY AUTO_INCREMENT, name_genre VARCHAR(30));
```

Созданная таблица - пустая.

Рекомендации по записи SQL запроса

- Ключевые слова: SQL не является регистрозависимым языком (CREATE и `create` - одно и тоже ключевое слово).

- Ключевые слова SQL и типы данных рекомендуется записывать прописными (большими) буквами.
- Имена таблиц и полей - строчными (маленькими) буквами.
- SQL-запрос можно писать на нескольких строках.
- В конце SQL-запроса ставится точка с запятой (хотя если Вы пишете один запрос, это необязательно).

Задание (Сформулируйте SQL запрос для создания таблицы book)

Сформулируйте SQL запрос для создания таблицы book, занесите его в окно кода (расположено ниже) и отправьте на проверку (кнопка **Отправить**). Структура таблицы **book**:

Поле	Тип, описание
book_id	INT PRIMARY KEY AUTO_INCREMENT
title	VARCHAR(50)
author	VARCHAR(30)
price	DECIMAL(8, 2)
amount	INT

Пояснение. При записи сохраняйте порядок следования полей.

Комментарии учащихся:

1.
Александр Ибраимов
если кто-то использует SQL Server. То вместо AUTO_INCREMENT используйте IDENTITY
2.
Oleg Shitov
Хотелось бы увидеть в Рекомендациях или где-нибудь дополнительно про пробелы: нужны, не нужны, влияют ли на что-нибудь?
Галина Озерова
@Oleg_Shitov, Пробелы не влияют. Но есть особенности этой платформы. Например, при использовании групповых функций, типа SUM(), пробел перед скобкой вызывает ошибку, хотя никаких ограничений по этому поводу нет.
3.
Святослав Орешин
А разве для фиксации таблицы в базе, после ; не нужно указывать commit; ? Или для mysql это не принципиально?
Галина Озерова
@Святослав_Орешин, Для MYSQL это не принципиально, в отличие от ORACLE
4.
Вадим Котеленец
Параллельно делаю запускаю запросы из урока в SQL SMS. Чтобы так сказать сохранить для себя. Но он ругается : Incorrect syntax near 'AUTO_INCREMENT'. Конфликт версий?
Галина Озерова
@Вадим_Котеленец, Наверное, посмотрите в описании, может для ключа нужно указать, что это число без знака? Или обязательно указать NOT NULL? Мне кажется, здесь есть ответ

https://stackoverflow.com/questions/10991894/auto-increment-primary-key-in-sql-server-management-studio-2012
Siarhei Konanau
@Вадим_Котеленец, Цитата ниже взята из http://www.bseu.by/it/tohod/lekcii7_4.htm
Несмотря на наличие международного стандарта ANSI SQL, многие компании, занимающиеся разработкой СУБД, вносят изменения в язык SQL, применяемый в разрабатываемой СУБД, тем самым отступая от стандарта.
Пример для MS SQL Server с автоикрементом взял здесь - https://www.w3schools.com/sql/sql_autoincrement.asp
The MS SQL Server uses the IDENTITY keyword to perform an auto-increment feature.

```
CREATE TABLE Persons (
    Personid int IDENTITY(1,1) PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

Шаг 7: Вставка записи в таблицу (**INSERT INTO**)

Ссылка в интернете

На каждом шаге можно посмотреть, как работает запрос примера. Для этого нужно скопировать его код в окно для ввода и нажать на черную кнопку **Запустить код** (не отправляя на проверку). Те запросы, которые уже проверены, можно не удалять, а просто закомментировать, используя /* и */.

Для занесения новой записи в таблицу используется SQL запрос, в котором указывается в какую таблицу, в какие поля заносить новые значения. Структура запроса:

- ключевые слова `INSERT INTO;`
- имя таблицы, в которую добавляется запись;
- открывающая круглая скобка «(»;
- список полей через запятую, в которые следует занести новые данные;
- закрывающая скобка «)»;
- ключевое слово `VALUES` ;
- открывающая круглая скобка «(»;
- список значений через запятую, которые заносятся в соответствующие поля, при этом текстовые значения заключаются в кавычки, числовые значения записываются без кавычек, в качестве разделителя целой и дробной части используется точка;
- закрывающая скобка «)».

Пример. В **таблицу**, состоящую из двух столбцов добавим новую строку, при этом в **поле1** заносится **значение1**, в **поле2** - **значение2**.

```
INSERT INTO таблица(поле1, поле2) VALUES (значение1, значение2)
```

В результате выполнения запроса новая запись заносится в конец обновляемой таблицы.

При составлении списка полей и списка значений необходимо учитывать следующее:

1. количество полей и количество значений в списках должны совпадать;
2. должно существовать прямое соответствие между позицией одного и того же элемента в обоих списках, поэтому первый элемент списка значений должен относиться к первому столбцу в списке столбцов, второй – ко второму столбцу и т.д.;

3. типы данных элементов в списке значений должны быть совместимы с типами данных соответствующих столбцов таблицы (целое число можно занести в поле типа DECIMAL, обратная операция - недопустима);
4. новые значения нельзя добавлять в поля, описанные как `PRIMARY KEY AUTO_INCREMENT`;
5. рекомендуется заполнять все поля записи, если же поле пропущено, значение этого поля зависит от установленных по умолчанию значений, если значения не установлены - на данной платформе вставляется пустое значение (**NULL**).

Пример

Вставим новую запись в таблицу **genre**, созданную на предыдущем шаге (в первых двух строках показана структура таблицы, далее - ее содержимое):

genre_id	name_genre
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(30)
1	Роман

Запрос:

```
INSERT INTO genre (name_genre) VALUES ('Роман');
```

Заносится только значение поля **name_genre**, значение ключевого поля формируется автоматически.

Результат: в таблицу будет вставлена новая строка, после запуска запроса на платформе **stepik**, имеем:

Чтобы увидеть, как именно выглядит таблица **genre**, можно добавить SQL запрос, который выберет все записи из таблицы:

```
SELECT * FROM genre;
```

Результат:

```
Affected rows: 1

Query result:
+----------+------------+
| genre_id | name_genre |
+----------+------------+
| 1        | Роман      |
+----------+------------+
Affected rows: 1
Свернуть
```

```
1  INSERT INTO genre (name_genre) VALUES ('Роман');
2  SELECT * FROM genre;
3
4
5
6
7
8
```

Задание (Занесите новую строку в таблицу book)

Занесите новую строку в таблицу book (текстовые значения (тип VARCHAR) заключать либо в двойные, либо в одинарные кавычки):

book_id	title	author	price	amount
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(50)	VARCHAR(30)	DECIMAL(8,2)	INT
1	Мастер и Маргарита	Булгаков М.А.	670.99	3

Рекомендация: текстовые поля копируйте из таблицы, представленной в задании, и вставляйте в запрос во избежание ошибок...

Комментарии учащихся:

1.

Виктор Дзюба

Подскажите, пожалуйста, есть ли какая-нибудь среда разработки для SQL, чтобы установить себе и выполнять задания там

Галина Озерова

@Виктор_Дзюба, Подходит любая среда, MYSQL. На платформе установлена сейчас версия SQL 8.0.21. Можно посмотреть здесь https://dev.mysql.com/downloads/

Виктор Дзюба

@Галина_Озерова, Microsoft SQL Server Management Studio из этой же области или это что то другое?

Галина Озерова

@Виктор_Дзюба, Мне кажется, это подойдет.

Шаг 8: Задание (Занесите три последние записи в таблицу **book**)

Ссылка в интернете

Занесите три последние записи в таблицу **book**, первая запись уже добавлена на предыдущем шаге:

book_id	title	Author	price	amount

INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(50)	VARCHAR(30)	DECIMAL(8,2)	INT
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2

Пояснение. Каждая строка вставляется отдельным SQL запросом, запросы обязательно разделять точкой с запятой. Для просмотра полученной таблицы после запросов на добавление записей вставить:

```
SELECT * FROM book;
```

УРОК 1.2: Выборка данных

Шаг 1: Содержание урока

Ссылка в интернете

В этом уроке будет рассмотрен синтаксис и семантика следующих SQL запросов:

- выборка всех данных из таблицы;
- выборка данных из отдельных столбцов;
- выборка отдельных столбцов и присвоение им новых имен;
- создание вычисляемых столбцов;
- вычисляемые столбцы, математические функции;
- вычисляемые столбцы, логические функции;
- выборка данных по простому условию;
- выборка данных с использованием логических выражений и операций;
- выборка данных, операторы BETWEEN, IN;
- выборка текстовых данных по шаблону, оператор LIKE;
- выборка данных с сортировкой.

Структура и наполнение таблицы

Все запросы будут формулироваться для таблицы **book** (создание, заполнение):

book_id	Title	author	Price	amount
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(50)	VARCHAR(30)	DECIMAL(8,2)	INT
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Шаг 2: Выборка всех данных из таблицы

Ссылка в интернете

Для того чтобы отобрать все данные из таблицы используется SQL запрос следующей структуры:

- ключевое слово `SELECT;`
- символ «`*`»;

- ключевое слово `FROM;`
- имя таблицы.

Результатом является таблица, в которую включены все строки и столбцы указанной в запросе таблицы.

Задание: Выбрать все записи таблицы `book`.

Запрос:

```
SELECT * FROM book;
```

Задание (Вывести информацию о всех книгах, хранящихся на складе).

Для этого:

- Напишите SQL запрос в окне кода;
- Отправьте на проверку (кнопка отправить) **Отправить;**
- Если запрос работает неверно, исправьте его и снова отправьте на проверку.

Важно! В окне кода можно использовать комментарии для сохранения разных вариантов запросов или пояснений. Комментарии заключаются в `/*` и `*/`:

Шаг 3: Выборка отдельных столбцов

Ссылка в интернете

Для того чтобы отобрать данные из определенных столбцов таблицы используется SQL запрос следующей структуры:

- ключевое слово `SELECT`;
- список столбцов таблицы через запятую;
- ключевое слово `FROM`;
- имя таблицы.

Результатом является таблица, в которую включены все данные из указанных после `SELECT` столбцов исходной таблицы.

Пример

Выбрать названия книг и их количества из таблицы **`book`**.

Запрос:

```
SELECT title, amount FROM book;
```

Результат:

```
+----------------------+--------+
| title                | amount |
+----------------------+--------+
| Мастер и Маргарита   | 3      |
| Белая гвардия        | 5      |
| Идиот                | 10     |
| Братья Карамазовы    | 2      |
| Стихотворения и поэмы | 15     |
+----------------------+--------+
```

Пояснение. Чтобы посмотреть, как работает запрос примера, скопируйте его код в окно для ввода и нажмите на черную кнопку **Запустить код**. После запуска выведется результат запроса, который можно сравнить с приведенным образцом.

Задание (Выбрать авторов, название книг и их цену из таблицы book).

Шаг 4: Выборка отдельных столбцов и присвоение им новых имен

Ссылка в интернете

Для того чтобы отобрать данные из определенных столбцов таблицы и одновременно задать столбцам новые имена используется SQL запрос следующей структуры:

- ключевое слово `SELECT`;
- имя столбца;
- ключевое слово `AS`;
- новое название столбца (можно русскими буквами), но это должно быть одно слово, если название состоит из двух слов – соединяйте их подчеркиванием, например, **`Количество_книг`**;
- запятая;
- имя столбца;
- ....
- ключевое слово `FROM`;
- имя таблицы.

В одном запросе можно использовать и имена столбцов из таблицы, и новые названия.

Результатом является таблица, в которую включены все данные из указанных после `SELECT` столбцов исходной таблицы. Каждому столбцу присваивается новое имя, заданное после `AS`, или столбец получает имя столбца исходной таблицы, если `AS` отсутствует.

Пример

Выбрать все названия книг и их количества из таблицы **`book`**, для столбца **`title`** задать новое имя **Название**.

Запрос:

```
SELECT title AS Название, amount FROM book;
```

Результат:

```
+----------------------+--------+
| Название             | amount |
+----------------------+--------+
| Мастер и Маргарита   | 3      |
| Белая гвардия        | 5      |
| Идиот                | 10     |
| Братья Карамазовы    | 2      |
| Стихотворения и поэмы | 15     |
+----------------------+--------+
```

Задание (Выбрать названия книг и авторов)

Выбрать названия книг и авторов из таблицы **book**, для поля **title** задать новое имя **Название**, для поля **author** – **Автор**

Шаг 5: Выборка данных с созданием вычисляемого столбца

Ссылка в интернете

С помощью SQL запросов можно осуществлять вычисления по каждой строке таблицы с помощью вычисляемого столбца. Для него в списке полей после оператора SELECT указывается выражение и задается имя.

Выражение может включать имена столбцов, константы, знаки операций, встроенные функции.

Результатом является таблица, в которую включены все данные из указанных после SELECT столбцов, а также новый столбец, в каждой строке которого вычисляется заданное выражение.

Пример

Вывести всю информацию о книгах, а также для каждой позиции посчитать ее стоимость (произведение цены на количество). Вычисляемому столбцу дать имя **total**.

Запрос:

```
SELECT title, author, price, amount, price * amount AS total FROM book;
```

Результат:

```
+----------------------+------------------+--------+--------+---------+
| title                | author           | price  | amount | total   |
+----------------------+------------------+--------+--------+---------+
| Мастер и Маргарита   | Булгаков М.А.    | 670.99 | 3      | 2012.97 |
| Белая гвардия        | Булгаков М.А.    | 540.50 | 5      | 2702.50 |
| Идиот                | Достоевский Ф.М. | 460.00 | 10     | 4600.00 |
| Братья Карамазовы    | Достоевский Ф.М. | 799.01 | 2      | 1598.02 |
| Стихотворения и поэмы | Есенин С.А.      | 650.00 | 15     | 9750.00 |
+----------------------+------------------+--------+--------+---------+
```

Задание (посчитать стоимость упаковки для каждой книги)

Для упаковки каждой книги требуется 1 лист бумаги, цена которого 1 рубль 65 копеек. Посчитать стоимость упаковки для каждой книги (сколько денег потребуется, чтобы упаковать все экземпляры книги). В запросе вывести название книги, ее количество и стоимость упаковки, последний столбец назвать **pack**.

Шаг 6: Выборка данных, вычисляемые столбцы, математические функции

Ссылка в интернете

В SQL реализовано множество математических функций для работы с числовыми данными. В таблице приведены некоторые из них.

Функция	**Описание**	**Пример**
`ceiling(x)`	возвращает наименьшее целое число, большее или равное **x**	`ceiling(4.2)=5` `ceiling(-5.8)=-5`
`round(x, k)`	округляет значение **x** до **k** знаков после запятой, если **k** не указано – **x** округляется до целого	`round(4.361)=4` `round(5.86592,1)=5.9`
`floor(x)`	возвращает наибольшее целое число, меньшее или равное **x**	`floor(4.2)=4` `floor(-5.8)=-6`
`power(x, y)`	возведение **x** в степень **y**	`power(3,4)=81.0`
`sqrt(x)`	квадратный корень из **x**	`sqrt(4)=2.0` `sqrt(2)=1.41...`
`degrees(x)`	конвертирует значение **x** из радиан в градусы	`degrees(3) = 171.8...`
`radians(x)`	конвертирует значение **x** из градусов в радианы	`radians(180)=3.14...`
`abs(x)`	модуль числа **x**	`abs(-1) = 1` `abs(1) = 1`
`Pi`	pi = 3.1415926...	

Пояснение. Существуют разные способы округления чисел. В SQL реализовано **математическое округление**. Для округления вещественного числа нужно в записи числа выбрать разряд в дробной части, до которого производится округление. Цифра, записанная в выбранном разряде: не меняется, если следующая за ней справа цифра - 0, 1, 2, 3 или 4; увеличивается на единицу, если следующая за ней справа цифра - 5,6,7,8 или 9.

Пример

Для каждой книги из таблицы **book** вычислим налог на добавленную стоимость (имя столбца **tax**) , который включен в цену и составляет k = 18%, а также цену книги (**price_tax**) без него. Формулы для вычисления:

$$tax = \frac{price * \frac{k}{100}}{1 + \frac{k}{100}},$$

$$price\_tax = \frac{price}{1 + \frac{k}{100}}$$

Запрос:

```
SELECT title, price, (price*18/100)/(1+18/100) AS tax, price/(1+18/100) AS price_tax
FROM book
```

Результат:

```
+----------------------+--------+---------------+------------+
| title                | price  | tax           | price_tax  |
+----------------------+--------+---------------+------------+
| Мастер и Маргарита   | 670.99 | 102.3544067797 | 568.635593 |
| Белая гвардия        | 540.50 | 82.4491525424  | 458.050847 |
| Идиот                | 460.00 | 70.1694915254  | 389.830508 |
| Братья Карамазовы    | 799.01 | 121.8828813559 | 677.127119 |
| Стихотворения и поэмы | 650.00 | 99.1525423729  | 550.847458 |
+----------------------+--------+---------------+------------+
```

Сумма налога и цена книги без налога – это деньги, поэтому количество знаков после запятой у этих чисел должно быть 2. Следовательно необходимо округлить полученные значения.

Запрос (код записан на нескольких строках, чтобы его удобно было читать, такая запись тоже допустима):

```
SELECT title,
       price,
       round((price*18/100)/(1+18/100),2) AS tax,
       round(price/(1+18/100),2) AS price_tax
FROM book
```

Результат:

```
+----------------------+--------+--------+-----------+
| title                | price  | tax    | price_tax |
+----------------------+--------+--------+-----------+
| Мастер и Маргарита   | 670.99 | 102.35 | 568.64    |
| Белая гвардия        | 540.50 | 82.45  | 458.05    |
| Идиот                | 460.00 | 70.17  | 389.83    |
| Братья Карамазовы    | 799.01 | 121.88 | 677.13    |
| Стихотворения и поэмы | 650.00 | 99.15  | 550.85    |
+----------------------+--------+--------+-----------+
```

Задание (пересчитать стоимость книг на – 30%)

В конце года цену всех книг на складе пересчитывают – снижают ее на 30%. Написать SQL запрос, который из таблицы **book** выбирает названия, авторов, количества и вычисляет новые цены книг. Столбец с новой ценой назвать **new_price**, цену округлить до 2-х знаков после запятой.

Результат:

```
+----------------------+-----------------+--------+-----------+
| title                | author          | amount | new_price |
+----------------------+-----------------+--------+-----------+
| Мастер и Маргарита   | Булгаков М.А.   | 3      | 469.69    |
| Белая гвардия        | Булгаков М.А.   | 5      | 378.35    |
| Идиот                | Достоевский Ф.М.| 10     | 322.00    |
| Братья Карамазовы    | Достоевский Ф.М.| 2      | 559.31    |
| Стихотворения и поэмы| Есенин С.А.     | 15     | 455.00    |
+----------------------+-----------------+--------+-----------+
```

Комментарии учащихся:

1.

Arpine Vardanyan
ceiling(4.2)=5 Ошибка в объяснении ceiling(4.2) равно 4
@Arpine_Vardanyan, нет, ошибки нет, сравните с определением floor(). Другими словами, ceiling() - это округление вверх, в то время как floor() - округление вниз, round() - округление к ближайшему.
Arpine Vardanyan
@Наталья_Соколова, floor(-5.8)=-6 почему тогда здесь не -5?
Александра Хен
@Arpine_Vardanyan, добрый день,
floor(x) - Округляет число вниз, при этом floor(1.5) = 1, floor(-1.5) = -2
ceil(x) - Округляет число вверх, при этом ceil(1.5) = 2, ceil(-1.5) = -1
Насчет floor(-5.8)=-6, чтобы было понятнее: **floor(x)** - возвращает *наибольшее целое число, меньшее или равное* **x**, тогда floor(5.8)=5 (меньшее x и наибольшее целое из чисел вниз), floor(-5.8)=-6 (из математики - меньшее x и наибольшее целое из чисел вниз)
Arpine Vardanyan
@Александра_Хен, поняла, большое спасибо за объяснение!

Шаг 7: Выборка данных, вычисляемые столбцы, логические функции (IF)

Ссылка в интернете

В SQL реализована возможность заносить в поле значение в зависимости от условия. Для этого используется функция `IF`:

```
IF(логическое_выражение, выражение_1, выражение_2)
```

Функция вычисляет `логическое_выражение`, если оно истина – в поле заносится значение `выражения_1`, в противном случае – значение `выражения_2`. Все три параметра `IF()` являются обязательными.

Допускается использование вложенных функций, вместо `выражения_1` или `выражения_2` может стоять новая функция **`IF`**.

Пример

Для каждой книги из таблицы **`book`** установим скидку следующим образом: если количество книг меньше 4, то скидка будет составлять 50% от цены, в противном случае 30%.

Запрос:

```
SELECT title, amount, price,
    IF(amount<4, price*0.5, price*0.7) AS sale
```

```
FROM book;
```

Результат:

```
+----------------------+--------+--------+---------+
| title                | amount | price  | sale    |
+----------------------+--------+--------+---------+
| Мастер и Маргарита   | 3      | 670.99 | 335.495 |
| Белая гвардия        | 5      | 540.50 | 378.350 |
| Идиот                | 10     | 460.00 | 322.000 |
| Братья Карамазовы    | 2      | 799.01 | 399.505 |
| Стихотворения и поэмы | 15     | 650.00 | 455.000 |
+----------------------+--------+--------+---------+
```

Цена по скидке должна отображаться с двумя знаками после запятой, добавим в запрос округление:

```
SELECT title, amount, price,
    round(IF(amount<4, price*0.5, price*0.7),2) AS sale
FROM book;
```

Результат:

```
+----------------------+--------+--------+--------+
| title                | amount | price  | sale   |
+----------------------+--------+--------+--------+
| Мастер и Маргарита   | 3      | 670.99 | 335.50 |
| Белая гвардия        | 5      | 540.50 | 378.35 |
| Идиот                | 10     | 460.00 | 322.00 |
| Братья Карамазовы    | 2      | 799.01 | 399.51 |
| Стихотворения и поэмы | 15     | 650.00 | 455.00 |
+----------------------+--------+--------+--------+
```

Пример

Усложним вычисление цены книги по скидке. Если количество книг меньше 4 – то скидка 50%, меньше 11 – 30%, в остальных случаях – 10%.

Запрос:

```
SELECT title, amount, price,
    round(IF(amount<4, price*0.5, IF(amount<11, price*0.7, price*0.9)),2) AS sale
FROM book;
```

Результат:

```
+----------------------+--------+--------+--------+
| title                | amount | price  | sale   |
+----------------------+--------+--------+--------+
| Мастер и Маргарита   | 3      | 670.99 | 335.50 |
| Белая гвардия        | 5      | 540.50 | 378.35 |
```

```
| Идиот                  | 10      | 460.00 | 322.00 |
| Братья Карамазовы      | 2       | 799.01 | 399.51 |
| Стихотворения и поэмы  | 15      | 650.00 | 585.00 |
+------------------------+---------+--------+--------+
```

Задание (Вывести информацию с учётом нового прайса)

При анализе продаж книг выяснилось, что наибольшей популярностью пользуются книги Михаила Булгакова, на втором месте книги Сергея Есенина. Исходя из этого решили поднять цену книг Булгакова на 10%, а цену книг Есенина - на 5%. Написать запрос, куда включить автора, название книги и новую цену, последний столбец назвать **new_price**. Значение округлить до двух знаков после запятой.

Пояснение:

- фамилию автора задавать с инициалами (как занесено в таблице), заключая в одинарные или двойные кавычки;
- для сравнения на равенство использовать знак =, например, **author="Булгаков М.А."**.

Результат:

```
+------------------+------------------------+-----------+
| author           | title                  | new_price |
+------------------+------------------------+-----------+
| Булгаков М.А.    | Мастер и Маргарита     | 738.09    |
| Булгаков М.А.    | Белая гвардия          | 594.55    |
| Достоевский Ф.М. | Идиот                  | 460.00    |
| Достоевский Ф.М. | Братья Карамазовы      | 799.01    |
| Есенин С.А.      | Стихотворения и поэмы  | 682.50    |
+------------------+------------------------+-----------+
```

Комментарии учащихся:

1.
Даниил Александров
Не забывайте о выражениях, в IF из два.
IF(условие_1, Выражение_1, Выражение_2 **это** >>>IF(условие_2, Выражение_1, Выражение_2))
Пока во втором IF не дописал Выражение_2 код не срабатывал.

Шаг 8: Выборка данных по условию (**WHERE**)

Ссылка в интернете

С помощью запросов можно включать в итоговую выборку не все строки исходной таблицы, а только те, которые отвечают некоторому условию. Для этого после указания таблицы, откуда выбираются данные, задается ключевое слово WHERE и логическое выражение, от результата которого зависит будет ли включена строка в выборку или нет. Если условие – истина, то строка(запись) включается в выборку, если ложь – нет.

Логическое выражение может включать **операторы сравнения** (равно «=», не равно «<>», больше «>», меньше «<», больше или равно «>=», меньше или равно «<=») и выражения, допустимые в SQL.

Пример

Вывести название и цену тех книг, цены которых меньше 600 рублей.

Запрос:

```
SELECT title, price
FROM book
WHERE price < 600;
```

Результат:

```
+---------------+--------+
| title         | price  |
+---------------+--------+
| Белая гвардия | 540.50 |
| Идиот         | 460.00 |
+---------------+--------+
```

Пример

Вывести название, автора и стоимость (цена умножить на количество) тех книг, стоимость которых больше 4000 рублей

Запрос:

```
SELECT title, author, price * amount AS total
FROM book
WHERE price * amount > 4000;
```

Результат:

```
+----------------------+------------------+---------+
| title                | author           | total   |
+----------------------+------------------+---------+
| Идиот                | Достоевский Ф.М. | 4600.00 |
| Стихотворения и поэмы | Есенин С.А.      | 9750.00 |
+----------------------+------------------+---------+
```

Пояснение. В логическом выражении после **WHERE** нельзя использовать названия столбцов, присвоенные им с помощью **AS**, так как при выполнении запроса сначала вычисляется логическое выражение для каждой строки исходной таблицы, выбираются строки, для которых оно истинно. А только после этого формируется "шапка запроса" – столбцы, включаемые в запрос.

Задание (Вывести книги, количество которых меньше 10).

Вывести автора, название и цены книг, количество которых меньше 10).

Результат:

```
+------------------+--------------------+--------+
| author           | title              | price  |
+------------------+--------------------+--------+
| Булгаков М.А.    | Мастер и Маргарита | 670.99 |
| Булгаков М.А.    | Белая гвардия      | 540.50 |
| Достоевский Ф.М. | Братья Карамазовы  | 799.01 |
+------------------+--------------------+--------+
```

Шаг 9: Выборка данных, логические операции (**AND, OR, NOT**)

Ссылка в интернете

Логическое выражение после ключевого слова `WHERE` кроме операторов сравнения и выражений может включать **логические операции** (И «**and**», ИЛИ «**or**», НЕ «**not**») и круглые скобки, изменяющие приоритеты выполнения операций.

Приоритеты операций:

1. круглые скобки
2. умножение (*), деление (/)
3. сложение (+), вычитание (-)
4. операторы сравнения (=, >, <, >=, <=, <>)
5. NOT
6. AND
7. OR

Пример

Вывести название, автора и цену тех книг, которые написал Булгаков, ценой больше 600 рублей

Запрос:

```
SELECT title, author, price
FROM book
WHERE price > 600 AND author = 'Булгаков М.А.';
```

Результат:

```
+--------------------+---------------+--------+
| title              | author        | price  |
+--------------------+---------------+--------+
| Мастер и Маргарита | Булгаков М.А. | 670.99 |
+--------------------+---------------+--------+
```

Пример

Вывести название, цену тех книг, которые написал Булгаков или Есенин, ценой больше 600 рублей

Запрос:

```
SELECT title, author, price

FROM book

WHERE (author = 'Булгаков М.А.' OR author = 'Есенин С.А.') AND price > 600;
```

Результат:

```
+----------------------+--------------+--------+
| title                | author       | price  |
+----------------------+--------------+--------+
| Мастер и Маргарита   | Булгаков М.А. | 670.99 |
| Стихотворения и поэмы | Есенин С.А.  | 650.00 |
+----------------------+--------------+--------+
```

Пояснение. В данном запросе обязательно нужно поставить скобки, так как без них сначала вычисляется `author = 'Есенин С.А.' and price > 600,` а потом уже выражение через `or`. Без скобок были бы отобраны все книги Булгакова и те книги Есенина, цена которых больше 600.

Запрос:

```
SELECT title, author, price

FROM book

WHERE author = 'Булгаков М.А.' OR author = 'Есенин С.А.' AND price > 600;
```

Результат (сравните с предыдущим):

```
+----------------------+--------------+--------+
| title                | author       | price  |
+----------------------+--------------+--------+
| Мастер и Маргарита   | Булгаков М.А. | 670.99 |
| Белая гвардия        | Булгаков М.А. | 540.50 |
| Стихотворения и поэмы | Есенин С.А.  | 650.00 |
+----------------------+--------------+--------+
```

Задание (Вывести книги, цена меньше 500 или больше 600, а стоимость 5000)

Вывести название, автора, цену и количество всех книг, цена которых меньше 500 или больше 600, а стоимость всех экземпляров этих книг больше или равна 5000.

Результат:

```
+----------------------+------------+--------+--------+
| title                | author     | price  | amount |
+----------------------+------------+--------+--------+
| Стихотворения и поэмы | Есенин С.А. | 650.00 | 15     |
+----------------------+------------+--------+--------+
```

Шаг 10: Выборка данных, операторы **BETWEEN, IN**

Ссылка в интернете

Логическое выражение после ключевого слова `WHERE` может включать операторы `BETWEEN` и `IN`. Эти операторы имеют самый низкий приоритет (как `OR`).

Оператор `BETWEEN` позволяет отобрать данные, относящиеся к некоторому интервалу, включая его границы.

Пример

Выбрать названия и количества тех книг, количество которых от 5 до 14 включительно.

Запрос:

```
SELECT title, amount
FROM book
WHERE amount BETWEEN 5 AND 14;
```

Результат:

```
+---------------+--------+
| title         | amount |
+---------------+--------+
| Белая гвардия | 5      |
| Идиот         | 10     |
+---------------+--------+
```

Этот запрос можно реализовать по-другому, результат будет точно такой же.

```
SELECT title, amount
FROM book
WHERE amount >= 5 AND amount <=14;
```

Оператор `IN` позволяет выбрать данные, соответствующие значениям из списка.

Пример

Выбрать названия и цены книг, написанных Булгаковым или Достоевским.

Запрос:

```
SELECT title, price
FROM book
WHERE author IN ('Булгаков М.А.', 'Достоевский Ф.М.');
```

Результат:

```
+-------------------+--------+
| title             | price  |
+-------------------+--------+
| Мастер и Маргарита | 670.99 |
| Белая гвардия     | 540.50 |
| Идиот             | 460.00 |
| Братья Карамазовы | 799.01 |
+-------------------+--------+
```

Этот запрос можно реализовать по-другому, результат будет точно такой же.

```
SELECT title, price

FROM book

WHERE author = 'Булгаков М.А.' OR author = 'Достоевский Ф.М.';
```

Задание (Вывести книги в интервале по стоимости)

Вывести название и авторов тех книг, цены которых принадлежат интервалу от 540.50 до 800 (включая границы), а количество или 2, или 3, или 5, или 7 .

Результат:

```
+--------------------+------------------+
| title              | author           |
+--------------------+------------------+
| Мастер и Маргарита | Булгаков М.А.    |
| Белая гвардия      | Булгаков М.А.    |
| Братья Карамазовы  | Достоевский Ф.М. |
+--------------------+------------------+
```

Комментарии учащихся:

1.
Anna
Скажите, что работает быстрее, BETWEEN IN или запрос вида "
WHERE author = 'Булгаков М.А.' OR author = 'Достоевский Ф.М.';"
Павел Москвин
@Anna, Вот здесь кое-какие мнения по этому поводу:
https://techarks.ru/qa/sql/in-vs-or-v-predlozhenii-sql-w-J0/
https://stackru.com/questions/16673563/est-li-raznitsa-v-proizvoditelnosti-mezhdu-between-i-in-s-mysql-ili-v-sql-v-tselom

Шаг 11: Выборка данных, оператор **LIKE**

Ссылка в интернете

Оператор `LIKE` используется для сравнения строк. В отличие от операторов отношения равно (=) и не равно (<>), `LIKE` позволяет сравнивать строки не на полное совпадение (не совпадение), а в соответствии с шаблоном. Шаблон может включать **обычные символы** и **символы-шаблоны**. При сравнении с шаблоном, его обычные символы должны в точности совпадать с символами, указанными в строке. Символы-шаблоны могут совпадать с произвольными элементами символьной строки.

Символ-шаблон	Описание	Пример
%	Любая строка, содержащая ноль или более символов	`SELECT * FROM book WHERE author LIKE '%M.%'` выполняет поиск и выдает все книги, инициалы авторов которых содержат «*M.*»
_ (подчеркивание)	Любой одиночный символ	`SELECT * FROM book WHERE title LIKE 'Поэм_'`

Символ-шаблон	Описание	Пример
		выполняет поиск и выдает все книги, названия которых либо «*Поэма*», либо «*Поэмы*» и пр.

Пример

Вывести названия книг, начинающихся с буквы «*Б*».

Запрос:

```
SELECT title

FROM book

WHERE title LIKE 'Б%';
```

Результат:

```
+-------------------+
| title             |
+-------------------+
| Белая гвардия     |
| Братья Карамазовы |
+-------------------+
```

Строчные и прописные буквы в строках эквивалентны, следующий запрос выдаст тот же результат.

```
SELECT title

FROM book

WHERE title LIKE 'б%';
```

Задание (вывести книги, автор содержит букву С.)

Вывести название и автора тех книг, название которых состоит из двух и более слов, а инициалы автора содержат букву «С».

Результат:

```
+----------------------+-------------+
| title                | author      |
+----------------------+-------------+
| Стихотворения и поэмы | Есенин С.А. |
+----------------------+-------------+
```

Пояснение. При записи условия, необходимо учесть, что слово в названии обязательно должно содержать хотя бы один символ.

Дополнительно можно прочесть здесь: https://www.w3schools.com/sql/sql_wildcards.asp

Комментарии учащихся:

1.

Дмитрий Ионов
Решил, но не совсем понимаю одного момента. Будет ли верен мой запрос, если название имеет следующий вид "Мастер и Маргарита"(много пробелов до и после "и"). Если да, то объясните, пожалуйста, смысл конструкций "_%" и "_%".
Галина Озерова
@Дмитрий_Ионов,
"_%" - обирает текстовые значения, которые состоят из последовательности любых символов, при чем должен быть хотя бы один символ. _ - означает наличие одного символа, % - любое количество символов в том числе пустое. То есть под этот шаблон подходят строки
1
в
аапаап
цу 2 4 34
И так далее. В этом шаблоне не указано, что название должно состоять из двух и более слов, то есть обязательно иметь пробел между словами.
"_%" - вторая часть у Вас такая же как первая
То решение, которое прошло, отбирает записи, которые включают как минимум один пробел между двумя словами. Если пробелов больше - эти записи тоже отберутся
Дмитрий Ионов
Спасибо за своевременной ответ! А есть разница между записью "_%" и "%_"? Как я понимаю, запись '_%абвг' говорит о том, что слева от "абвг" может быть сколько угодно символов, но затем обязательно символ отличный от пробела или я что-то путаю?
Галина Озерова
@Дмитрий_Ионов, Разницы нет, и дальше - Ваше объяснение верно

Шаг 12: Выборка данных с сортировкой (**ORDER BY**)

Ссылка в интернете

При выборке можно указывать столбец или несколько столбцов, по которым необходимо отсортировать отобранные строки. Для этого используются ключевые слова `ORDER BY`, после которых задаются имена столбцов. При этом строки сортируются по первому столбцу, если указан второй столбец, сортировка осуществляется только для тех строк, у которых значения первого столбца одинаковы. По умолчанию `ORDER BY` выполняет сортировку по возрастанию. Чтобы управлять направлением сортировки вручную, после имени столбца указывается ключевое слово `ASC` (по возрастанию) или `DESC` (по убыванию).

Логический порядок операций для запроса SQL следующий:

1. FROM
2. WHERE
3. SELECT
4. ORDER BY

Поскольку сортировка выполняется позже `SELECT`, для указания столбцов, по которым выполняется сортировка, можно использовать имена, присвоенные им после `SELECT`, а также порядковый номер столбца в перечислении.

Пример

Вывести название, автора и цены книг. Информацию отсортировать по названиям книг в алфавитном порядке.

Запрос:

```
SELECT title, author, price

FROM book

ORDER BY title;
```

Результат:

```
+----------------------+------------------+--------+
| title                | author           | price  |
+----------------------+------------------+--------+
| Белая гвардия        | Булгаков М.А.    | 540.50 |
| Братья Карамазовы    | Достоевский Ф.М. | 799.01 |
| Идиот                | Достоевский Ф.М. | 460.00 |
| Мастер и Маргарита   | Булгаков М.А.    | 670.99 |
| Стихотворения и поэмы | Есенин С.А.     | 650.00 |
+----------------------+------------------+--------+
```

Аналогичный результат получится при использовании запроса:

```
SELECT title, author, price

FROM book

ORDER BY 1;
```

Пример

Вывести автора, название и количество книг, в отсортированном в алфавитном порядке по автору и по убыванию количества, для тех книг, цены которых меньше 750 рублей.

Запрос:

```
SELECT author, title, amount AS Количество

FROM book

WHERE price < 750

ORDER BY author, amount DESC;
```

Результат:

```
+------------------+----------------------+------------+
| author           | title                | Количество |
+------------------+----------------------+------------+
| Булгаков М.А.    | Белая гвардия        | 5          |
| Булгаков М.А.    | Мастер и Маргарита   | 3          |
| Достоевский Ф.М. | Идиот                | 10         |
| Есенин С.А.      | Стихотворения и поэмы | 15         |
+------------------+----------------------+------------+
```

Можно использовать другие варианты записи запроса:

```
SELECT author, title, amount AS Количество
FROM book
WHERE price < 750
ORDER BY author, Количество DESC;
```

```
SELECT author, title, amount AS Количество
FROM book
WHERE price < 750
ORDER BY 1, 3 DESC;
```

Задание (вывести книги, количество в интервале)

Вывести автора и название книг, количество которых принадлежит интервалу от 2 до 14 (включая границу). Информацию отсортировать по авторам (в обратном алфавитном порядке) и названиям (по алфавиту).

Результат:

```
+------------------+--------------------+
| author           | title              |
+------------------+--------------------+
| Достоевский Ф.М. | Братья Карамазовы  |
| Достоевский Ф.М. | Идиот              |
| Булгаков М.А.    | Белая гвардия      |
| Булгаков М.А.    | Мастер и Маргарита |
+------------------+--------------------+
```

Шаг 13: Задание (Придумать запрос к таблице book)

Ссылка в интернете

Придумайте один или несколько запросов к нашей таблице **book**. Проверьте, правильно ли они работают.

При желании можно формулировку запросов разместить в комментариях. Размещенные задания можно использовать для закрепления материала урока. Оценивайте понравившиеся Вам запросы.

В последнем модуле создан отдельный урок, в котором мы разместим запросы, набравшие наибольшее количество лайков.

УРОК 1.3: Запросы, групповые операции

Шаг 1: Содержание урока

Ссылка в интернете

SQL запросы позволяют производить вычисления не только для каждой строки таблицы, но и для группы элементов, расположенных в одном столбце. Для этого используются групповые (агрегатные) функции.

На этом уроке рассмотрим запросы, которые реализуют:

- выборку различных элементов столбца;
- выборку данных (групповые функции **sum** и **count**);
- выборку данных (групповые функции **min**, **max** и **avg**);
- выборку данных с вычислением (групповые функции);
- вычисления по таблице целиком;
- выборку данных по условию (групповые функции)
- выборка данных по условию (групповые функции WHERE и HAVING).

Структура и наполнение таблицы

Все запросы будут формулироваться для таблицы **book** (создание, заполнение):

book_id	Title	author	price	amount
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(50)	VARCHAR(30)	DECIMAL(8,2)	INT
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	3
5	Игрок	Достоевский Ф.М.	480.50	10
6	Стихотворения и поэмы	Есенин С.А.	650.00	15

Шаг 2: Выбор различных элементов столбца (**DISTINCT, GROUP BY**)

Ссылка в интернете

Чтобы отобрать уникальные элементы некоторого столбца используется ключевое слово `DISTINCT`, которое размещается сразу после `SELECT`.

Пример

Выбрать различных авторов, книги которых хранятся в таблице **book**.

Запрос:

```
SELECT DISTINCT author
FROM book;
```

Результат:

```
+------------------+
| author           |
+------------------+
| Булгаков М.А.    |
| Достоевский Ф.М. |
| Есенин С.А.      |
+------------------+
```

Другой способ – использование оператора GROUP BY, который группирует данные при выборке, имеющие одинаковые значения в некотором столбце. Столбец, по которому осуществляется группировка, указывается после GROUP BY.

С помощью GROUP BY можно выбрать уникальные элементы столбца, по которому осуществляется группировка. Результат будет точно такой же как при использовании DISTINCT.

Запрос:

```
SELECT author
FROM book
GROUP BY author;
```

Задание (Отобрать различные элементы столбца **amount** таблицы **book**).

Комментарии учащихся:

1.

Нурмагомед Омаров

Добрый день! А в чем различия между этими двумя операторами? В каких случаях будут другие результаты?:)

Галина Озерова

@Нурмагомед_Омаров, Здравствуйте. Результаты будут всегда одинаковыми. Но если просто нужно получить неповторяющиеся значения - лучше использовать DISTINCT. Возможности GROUP BY значительно шире, в частности он позволяет производить вычисления над записями, входящими в группу с одинаковыми значениями в указанном после GROUP BY столбце.

Шаг 3: Выборка данных, групповые функции **SUM и COUNT**

Ссылка в интернете

При группировке над элементами столбца, входящих в группу можно выполнить различные действия, например, просуммировать их или найти количество элементов в группе.

При группировке данных таблицы **book** по столбцу **author**, получается три группы. В первую группу входят две строки, в поле **author** которых стоит значение «Булгаков М.А.». В столбце **amount** к этой группе относится значения 3 и 5. Просуммировав эти значения, можно узнать общее количество книг Булгакова на складе. Для этого используется групповая функция SUM(). В скобках указывается столбец, по которому осуществляется суммирование.

Пример

Посчитать, сколько экземпляров книг каждого автора хранится на складе.

Запрос:

```
SELECT author, SUM(amount)
FROM book
GROUP BY author;
```

Результат:

```
+------------------+-------------+
| author           | SUM(amount) |
+------------------+-------------+
| Булгаков М.А.    | 8           |
| Достоевский Ф.М. | 23          |
| Есенин С.А.      | 15          |
+------------------+-------------+
```

Примечание. Обратите внимание, что в качестве названия вычисляемого столбца в результирующей таблице используется выражение. Рекомендуется всем вычисляемым столбцам давать имя.

Также групповые функции позволяют посчитать сколько записей относится к каждой группе, для этого используется функция count().

Пример

Посчитать, сколько различных книг каждого автора хранится на складе.

Запрос:

```
SELECT author, COUNT(author), COUNT(amount), COUNT(*)
FROM book
GROUP BY author;
```

Результат:

```
+------------------+---------------+---------------+----------+
| author           | COUNT(author) | COUNT(amount) | COUNT(*) |
+------------------+---------------+---------------+----------+
| Булгаков М.А.    | 2             | 2             | 2        |
| Достоевский Ф.М. | 3             | 3             | 3        |
| Есенин С.А.      | 1             | 1             | 1        |
+------------------+---------------+---------------+----------+
```

Примечание. Из таблицы с результатами запроса видно, что функцию **COUNT()** можно применять к любому столбцу, в том числе можно использовать и *, если таблица не содержит пустых значений. Если же в столбцах есть значения **Null**, то

COUNT(*) — подсчитывает все записи, относящиеся к группе, в том числе и со значением **NULL**;

COUNT(имя_столбца) — возвращает количество записей конкретного столбца (только **NOT NULL**), относящихся к группе.

ВАЖНО. После оператора GROUP BY должны перечисляться ВСЕ неагрегированные столбцы (то есть столбцы, к которым не применены групповые функции), указанные после SELECT.

Задание (посчитать количество экземпляров книг каждого автора)

Посчитать, количество различных книг и количество экземпляров книг каждого автора, хранящихся на складе. Вычисляемые столбцы назвать **Различных_книг** и **Количество_экземпляров** соответственно, столбец с фамилиями авторов назвать **Автор**.

Результат:

```
+------------------+----------------+------------------------+
| Автор            | Различных_книг | Количество_экземпляров |
+------------------+----------------+------------------------+
| Булгаков М.А.    | 2              | 8                      |
| Достоевский Ф.М. | 3              | 23                     |
| Есенин С.А.      | 1              | 15                     |
+------------------+----------------+------------------------+
```

Пояснение. Название столбцов может состоять из нескольких слов, тогда их нужно заключать в кавычки. Но если слова написать через подчеркивание, тогда получится, что название состоит из одного слова, и кавычки можно не ставить.

Комментарии учащихся:

1.
Татьяна Барбуца
У меня вопрос, под вторым примером
ВАЖНО. После оператора GROUP BY должны перечисляться ВСЕ неагрегированные столбцы (то есть столбцы, к которым не применены групповые функции), указанные после SELECT.
не поняла, как это связано со вторым примером и вообще смысл не очень понятен
Никита Лень
@Татьяна_Барбуца, SELECT author, COUNT(author), COUNT(amount) FROM book GROUP BY author;
Как Вы можете видеть, после SELECT перечислены три столбца: author, COUNT(author), COUNT(amount). COUNT это групповая функция, она применена к author и amount, НО в SELECT также есть и просто author, уже без групповой функции. Получается, что это единственный неагрегированный столбец, по которому и должна происходить группировка с помощью GROUP BY author.
Если у Вас остались вопросы в части теории, можете поискать дополнительные материалы в интернете.
2.
Игорь Стороженко
В тексте:

ВАЖНО. После оператора GROUP BY должны перечисляться ВСЕ неагрегированные столбцы (то есть столбцы к которым не применены групповые функции), указанные после SELECT.
лишний пробел после скобки и нет запятой, после "то есть столбцы"
И я так и не понял смысл этого абзаца... Не могли бы вы пояснить?
Курс отличный, спасибо!
Никита Лень
@Игорь_Стороженко, давайте объясню на примере.
SELECT author, title, SUM(amount)
FROM book
GROUP BY author, title;
В данном запросе после GROUP BY были перечислены все неагрегированные столбцы (author, title), те столбцы, к которым групповые функции не были применены, а агрегированный столбец amount (к которому применена групповая функция SUM) не был перечислен.
Игорь Стороженко
@Никита_Лень, в этом примере группировка будет и по автору, и по названию? а если мне надо группировать только по автору?
Никита Лень
@Игорь_Стороженко, да, если у Вас SELECT author, COUNT(title) написано, то группировка только по автору, а если в SELECT есть ещё какие-то неагрегированные столбцы, то надо добавить и их в GROUP BY.
Анастасия Дурыманова
@Никита_Лень, но при таком запросе SUM не работает, я имею в виду ваш пример
SELECT author, title, SUM(amount)
FROM book
GROUP BY author, title;
Дмитрий Чупров
@Анастасия_Дурыманова, SUM() работает, он посчитал сумму всех книг для каждой из уникальных комбинаций author + title.
То есть для :
Булгаков М.А. | Белая гвардия получется 5
Булгаков М.А. | Мастер и Маргарита получется 3
Достоевский Ф.М. | Братья Карамазовы получется 3
и тд
Анастасия Дурыманова
@Дмитрий_Чупров, Извините, но ничего не понятно. Новая таблица не отличается от исходной.
Дмитрий Чупров
@Анастасия_Дурыманова, Исходная - это book, созданная в начале? Если да, то она не должна изменятся. Нужно напечатать "новую" (всё что в ней напечатанно/изменено не сохраняется, т.е. в book не появляется новых столбцаов)
Надеюсь, я вас правильно понял.
Дмитрий Чупров
@Анастасия_Дурыманова, Возможно, я не подробно описал первое объяснение (https://stepik.org/lesson/297515/step/3?discussion=1742791&reply=1836859&unit=279275). Я имел ввиду, что, когда в GROUP BY пишут author. title, то SQL принимает в SUM(amount) каждую строку, где встречаются данные комбинации .
Например GROUP BY author, title
Создаёт
```
+-----------------+----------------------+.....
| author          | title                |
+-----------------+----------------------+.....
| Булгаков М.А.   | Белая гвардия        |
| Булгаков М.А.   | Мастер и Маргарита   |.....
| Достоевский Ф.М. | Братья Карамазовы   |
| Достоевский Ф.М. | Игрок               |.....
| Достоевский Ф.М. | Идиот               |
| Есенин С.А.     | Стихотворения и поэмы |.....
+-----------------+----------------------+
```
Каждая строка уникальна
ИЛИ

GROUP BY author
Создаёт
+------------------+.....
| author |
+------------------+.....
| Булгаков М.А. |
| Достоевский Ф.М. |.....
| Есенин С.А. |
+------------------+.....
Каждая строка уникальна
И далее заполняются остальные/нужные
Анастасия Дурыманова
@Дмитрий_Чупров, спасибо, теперь понятно) Если бы в таблице book было бы несколько записей, например, Достоевский Ф.М. Игрок, то в новой таблице, сгруппированной по автору и названию, sum(amount) бы не совпадало с количеством в исходной таблице.
3.
Даур Абишев
Я правильно понимаю, что агрегатные функции нельзя использовать в условиях WHERE?
Никита Лень
@Даур_Абишев, да, необходимо использовать вложенный запрос. Например,
SELECT author
FROM book
WHERE (SELECT COUNT(amount) FROM book) = 1

Шаг 4: Выборка данных, групповые функции **MIN, MAX и AVG**

Ссылка в интернете

К групповым функциям SQL относятся: `MIN()`, `MAX()` и `AVG()`, которые вычисляют минимальное, максимальное и среднее значение элементов столбца, относящихся к группе.

Пример

Вывести минимальную цену книги каждого автора

Запрос:

```
SELECT author, MIN(price) AS min_price

FROM book

GROUP BY author;
```

Результат:

```
+------------------+-----------+
| author           | min_price |
+------------------+-----------+
| Булгаков М.А.    | 540.50    |
| Достоевский Ф.М. | 460.00    |
| Есенин С.А.      | 650.00    |
+------------------+-----------+
```

Задание (Вывести минимальную, максимальную и среднюю цены для каждого автора)

Вывести минимальную, максимальную и среднюю цену книг каждого автора. Вычисляемые столбцы назвать **Минимальная_цена, Максимальная_цена** и **Средняя_цена** соответственно.

Результат:

```
+-----------------+------------------+--------------------+--------------+
| author          | Минимальная_цена | Максимальная_цена  | Средняя_цена |
+-----------------+------------------+--------------------+--------------+
| Булгаков М.А.   | 540.50           | 670.99             | 605.745000   |
| Достоевский Ф.М.| 460.00           | 799.01             | 579.836667   |
| Есенин С.А.     | 650.00           | 650.00             | 650.000000   |
+-----------------+------------------+--------------------+--------------+
```

Комментарии учащихся:

1.

Denys_Petryk

Почему у меня получилось все без AS в поле AVG(price)'Средняя_цена' ?
SELECT...

Галина Озерова

@Denys_Petryk, Это зависит от того, как реализован интерпретатор. В документации указано, что AS писать нужно. Почему здесь не выдается ошибка - не знаю.

Siarhei Konanau

@Галина_Озерова, из документации.

The following list provides additional information about other `SELECT` clauses:

- A *`select_expr`* can be given an alias using `AS` *`alias_name`*. The alias is used as the expression's column name and can be used in `GROUP BY`, `ORDER BY`, or `HAVING` clauses. For example:

```
SELECT CONCAT(last_name,', ',first_name) AS full_name
  FROM mytable ORDER BY full_name;
```

The `AS` keyword is optional when aliasing a *`select_expr`* with an identifier. The preceding example could have been written like this:

```
SELECT CONCAT(last_name,', ',first_name) full_name
  FROM mytable ORDER BY full_name;
```

However, because the `AS` is optional, a subtle problem can occur if you forget the comma between two *`select_expr`* expressions: MySQL interprets the second as an alias name. For example, in the following statement, `columnb` is treated as an alias name:

```
SELECT columna columnb FROM mytable;
```

For this reason, it is good practice to be in the habit of using `AS` explicitly when specifying column aliases.

Шаг 5: Выборка данных с вычислением, групповые функции

Ссылка в интернете

В качестве аргумента групповых функций SQL может использоваться не только столбец, но и любое допустимое в SQL арифметическое выражение.

Пример

Вывести суммарную стоимость книг каждого автора.

Запрос:

```
SELECT author, SUM(price * amount) AS Стоимость
FROM book
GROUP BY author;
```

Результат:

```
+------------------+-----------+
| author           | Стоимость |
+------------------+-----------+
| Булгаков М.А.    | 4715.47   |
| Достоевский Ф.М. | 11802.03  |
| Есенин С.А.      | 9750.00   |
+------------------+-----------+
```

Групповые функции могут быть элементами выражений. Например, при вычислении средней стоимости книг каждого автора на предыдущем шаге получились значения с шестью знаками после запятой. А поскольку это деньги, значения нужно округлить до 2 знаков после запятой.

Пример

Найти среднюю цену книг каждого автора.

Запрос:

```
SELECT author, round(AVG(price),2) AS Средняя_цена
FROM book
GROUP BY author;
```

Результат:

```
+------------------+--------------+
| author           | Средняя_цена |
+------------------+--------------+
| Булгаков М.А.    | 605.75       |
| Достоевский Ф.М. | 579.84       |
| Есенин С.А.      | 650.00       |
+------------------+--------------+
```

Задание (Вычислить стоимость книг, НДС и стоимость без НДС)

Для каждого автора вычислить суммарную стоимость книг **S** (имя столбца **Стоимость**), а также вычислить налог на добавленную стоимость для полученных сумм (имя столбца **НДС**), который включен в стоимость и составляет k = 18%, а также стоимость книг (**Стоимость_без_НДС**) без него. Значения округлить до двух знаков после запятой. Формулы для вычисления:

$$tax = \frac{S * \frac{k}{100}}{1 + \frac{k}{100}},$$

$$S\_without\_tax = \frac{S}{1 + \frac{k}{100}}$$

Результат:

```
+-----------------+-----------+---------+-------------------+
| author          | Стоимость | НДС     | Стоимость_без_НДС |
+-----------------+-----------+---------+-------------------+
| Булгаков М.А.   | 4715.47   | 719.31  | 3996.16           |
| Достоевский Ф.М. | 11802.03  | 1800.31 | 10001.72          |
| Есенин С.А.     | 9750.00   | 1487.29 | 8262.71           |
+-----------------+-----------+---------+-------------------+
```

Пояснение. Имена столбцов, присвоенные им с помощью `AS`, нельзя использовать в выражениях, используйте названия столбцов исходной таблицы.

Шаг 6: Вычисления по таблице целиком

Ссылка в интернете

Групповые функции позволяют вычислять итоговые значения по всей таблице. Например, можно посчитать общее количество книг на складе, вычислить суммарную стоимость и пр. Для этого после ключевого слова `SELECT` указывается групповая функция для выражения или имени столбца, а ключевые слова `GROUP BY` опускаются.

Пример

Посчитать количество книг на складе.

Запрос:

```
SELECT SUM(amount) AS Количество
FROM book;
```

Результат:

```
+------------+
| Количество |
+------------+
| 46         |
+------------+
```

Результатом таких запросов является единственная строка с вычисленными по таблице значениями.

Пример

Посчитать общее количество книг на складе и их стоимость.

Запрос:

```
SELECT SUM(amount) AS Количество,
    SUM(price * amount) AS Стоимость
```

```
FROM book;
```

Результат:

```
+------------+-----------+
| Количество | Стоимость |
+------------+-----------+
| 46         | 26267.50  |
+------------+-----------+
```

Задание (Вычислить максимальную, минимальную и среднюю цены)

Вывести цену самой дешевой книги, цену самой дорогой и среднюю цену книг на складе. Названия столбцов **Минимальная_цена, Максимальная_цена, Средняя_цена** соответственно. Среднюю цену округлить до двух знаков после запятой.

Результат:

```
+-----------------+-------------------+--------------+
| Минимальная_цена | Максимальная_цена | Средняя_цена |
+-----------------+-------------------+--------------+
| 460.00           | 799.01            | 600.17       |
+-----------------+-------------------+--------------+
```

Шаг 7: Выборка данных по условию, групповые функции (**WHERE, HAVING**)

Ссылка в интернете

В запросы с групповыми функциями можно включать условие отбора строк, которое в обычных запросах записывается после `WHERE`. В запросах с групповыми функциями вместо `WHERE` используется ключевое слово `HAVING`, которое размещается после оператора `GROUP BY`.

Пример

Найти минимальную и максимальную цену книг всех авторов, общая стоимость книг которых больше 5000.

Запрос:

```
SELECT author,
    MIN(price) AS Минимальная_цена,
    MAX(price) AS Максимальная_цена
FROM book
GROUP BY author
HAVING SUM(price*amount) > 5000;
```

Результат:

```
+-----------------+-----------------+-----------------+
| author          | Минимальная_цена | Максимальная_цена |
+-----------------+-----------------+-----------------+
| Достоевский Ф.М. | 460.00          | 799.01          |
```

```
| Есенин С.А.       | 650.00           | 650.00            |
+-------------------+------------------+-------------------+
```

Также в запросах с группировкой можно сортировать данные.

Пример

Найти минимальную и максимальную цену книг всех авторов, общая стоимость книг которых больше 5000. Результат вывести по убыванию минимальной цены.

Запрос:

```
SELECT author,
    MIN(price) AS Минимальная_цена,
    MAX(price) AS Максимальная_цена
FROM book
GROUP BY author
HAVING SUM(price*amount) > 5000
ORDER BY Минимальная_цена DESC;
```

Результат:

```
+-------------------+------------------+-------------------+
| author            | Минимальная_цена | Максимальная_цена |
+-------------------+------------------+-------------------+
| Есенин С.А.       | 650.00           | 650.00            |
| Достоевский Ф.М.  | 460.00           | 799.01            |
+-------------------+------------------+-------------------+
```

Пояснение. При указании столбца, по которому выполняется сортировка, если столбцу присвоено имя с помощью `AS`, можно использовать это имя.

Задание (Вычислить среднюю цену и суммарную стоимость)

Вычислить среднюю цену и суммарную стоимость тех книг, количество которых принадлежит интервалу от 5 до 14, включительно. Столбцы переименовать как показано в образце, значения округлить до 2-х знаков после запятой.

Результат:

```
+--------------+-----------+
| Средняя_цена | Стоимость |
+--------------+-----------+
| 493.67       | 12107.50  |
+--------------+-----------+
```

Пояснение. Если в запросе с групповыми функциями отсутствует `GROUP BY`, то для отбора записей используется ключевое слово `WHERE`.

Шаг 8: Выборка данных по условию, групповые функции, **WHERE и HAVING**

Ссылка в интернете

Для этого урока теоретическая часть подготовлена Alexandra Klinnikova, спасибо большое!

`WHERE` и `HAVING` могут использоваться в одном запросе. При этом необходимо учитывать порядок выполнения SQL запроса на выборку на сервере:

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

Сначала определяется таблица, из которой выбираются данные (`FROM`), затем из этой таблицы отбираются записи в соответствии с условием `WHERE`, выбранные данные агрегируются (`GROUP BY`), из агрегированных записей выбираются те, которые удовлетворяют условию после `HAVING`. Потом формируются данные результирующей выборки, как это указано после `SELECT` (вычисляются выражения, присваиваются имена и пр.). Результирующая выборка сортируется, как указано после `ORDER BY`.

Пример

Вывести максимальную и минимальную цену книг всех авторов, кроме Есенина, количество экземпляров книг которых больше 10.

```
SELECT author,
    MIN(price) AS Минимальная_цена,
    MAX(price) AS Максимальная_цена
FROM book
WHERE author <>'Есенин С.А.'
GROUP BY author
HAVING SUM(amount)>10;
```

Результат:

```
+------------------+------------------+-------------------+
| author           | Минимальная_цена | Максимальная_цена |
+------------------+------------------+-------------------+
| Достоевский Ф.М. | 460.00           | 799.01            |
+------------------+------------------+-------------------+
```

Это запрос будет работать если его переписать следующим образом:

```
SELECT author,
    MIN(price) AS Минимальная_цена,
    MAX(price) AS Максимальная_цена
FROM book
GROUP BY author
```

```
HAVING SUM(amount)>10 AND author <>'Есенин С.А.';
```

Не смотря на то что результат будет одинаковым, так делать **не рекомендуется**. «Потому что как написано - запрос сначала выбирает всех авторов, потом выводит данные, рассчитывая минимальное и максимальное значение цены для каждого, и только после всего убирает Есенина. Можно убрать Есенина в данном случае раньше и не использовать ресурсы базы для расчета его минимального и максимального значения, как это сделано в первом варианте. На небольшой базе быстродействия не ощутить, но если выполнять такое на продуктивной, то второй вариант значительно проигрывает...»[Alexandra Klinnikova].

Задание (Посчитать стоимость всех экземпляров с условием)

Посчитать стоимость всех экземпляров каждого автора без учета книг «Идиот» и «Белая гвардия». В результат включить только тех авторов, у которых суммарная стоимость книг более 5000 руб. Результат отсортировать по убыванию стоимости.

```
+------------------+-----------+
| author           | Стоимость |
+------------------+-----------+
| Есенин С.А.      | 9750.00   |
| Достоевский Ф.М. | 7202.03   |
+------------------+-----------+
```

Комментарии учащихся:

1.

Фируз Саидов

Господа преподаватели, во первых спасибо вам огромное за такой замечательный курс.
ВОПРОС(код из примера):

```
SELECT author,
        MIN(price) AS Минимальная_цена,
        MAX(price) AS Максимальная_цена
FROM book
WHERE author <>'Есенин С.А.'
GROUP BY author
HAVING SUM(amount)>10;
```

Почему мы по такой же логике как в примере (для уменьшения количества вычислений), не можем запихнуть SUM(amount)>10 в WHERE?
так:

```
WHERE author <>'Есенин С.А.'  AND SUM(amount)>10
```

Лариса Фернандес

@Фируз_Саидов, в предыдущем комментарии я как раз это объясняла)
необходимо учитывать порядок выполнения SQL запроса на выборку на сервере (https://stepik.org/lesson/297515/step/8?unit=279275) :

1. FROM - сначала выбираем таблицы и, возможно, джойним несколько в одну
2. WHERE - проверяем какие-то условия в столбцах
3. GROUP BY - **группируем**
4. HAVING - отбираем результат группировки
5. SELECT - выводим
6. ORDER BY

SUM - групповая функция, пока мы что-то там не сгруппируем, она не посчитается. WHERE проверяет условие в исходной таблице, HAVING - в результате действий над таблицей

Фируз Саидов

@Лариса_Фернандес, Теперь понятно. Спасибо

Шаг 9: Задание (Придумать запрос к таблице book)

Ссылка в интернете

Придумайте один или несколько запросов к нашей таблице **book**, используя групповые функции. Проверьте, правильно ли они работают.

При желании можно формулировку запросов разместить в комментариях. Размещенные задания можно использовать для закрепления материала урока. Оценивайте понравившиеся Вам запросы.

В последнем модуле создан отдельный урок, в котором мы разместим запросы, набравшие наибольшее количество лайков.

УРОК 1.4: Вложенные запросы

Шаг 1: Содержание урока

Ссылка в интернете

SQL позволяет создавать вложенные запросы. Вложенный запрос (подзапрос, внутренний запрос) – это запрос внутри другого запроса SQL.

Вложенный запрос используется для выборки данных, которые будут использоваться в условии отбора записей основного запроса. Его применяют для:

- сравнения выражения с результатом вложенного запроса;
- определения того, включено ли выражение в результаты вложенного запроса;
- проверки того, выбирает ли запрос определенные строки.
- Вложенный запрос, имеет следующие компоненты:
- ключевое слово `SELECT` после которого указываются имена столбцов или выражения (чаще всего список содержит один элемент) ;
- ключевое слово `FROM` и имя таблицы, из которой выбираются данные;
- необязательное предложение `WHERE` ;
- необязательное предложение `GROUP BY:`
- необязательное предложение `HAVING` .

Вложенные запросы могут включаться в WHERE так (в квадратных скобках указаны необязательные элементы, через | – один из элементов):

`WHERE` выражение *оператор_сравнения* (вложенный запрос) ;

`WHERE` выражение, включающее вложенный запрос**;**

`WHERE` выражение [NOT] IN (вложенный запрос) ;

`WHERE` выражение *оператор_сравнения* ANY | ALL (вложенный запрос) .

Также вложенные запросы могут вставляться в основной запрос после ключевого слова `SELECT` .

Структура и наполнение таблицы

Все запросы в данном уроке будут формулироваться для таблицы **book** (создание, заполнение):

book_id	Title	author	price	amount
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(50)	VARCHAR(30)	DECIMAL(8,2)	INT
1	Мастер и Маргарита	Булгаков М.А.	670.99	3

2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	3
5	Игрок	Достоевский Ф.М.	480.50	10
6	Стихотворения и поэмы	Есенин С.А.	650.00	15

Шаг 2: Вложенный запрос, возвращающий одно значение

Ссылка в интернете

Вложенный запрос, возвращающий одно значение, может использоваться в условии отбора записей `WHERE` как обычное значение совместно с операциями =, <>, >=, <=, >, <.

Пример

Вывести информацию о самых дешевых книгах, хранящихся на складе.

Для реализации этого запроса нам необходимо получить минимальную цену из столбца **price** таблицы **book**, а затем вывести информацию о тех книгах, цена которых равна минимальной. Первая часть – поиск минимума – реализуется вложенным запросом.

Запрос:

```
SELECT title, author, price, amount
FROM book
WHERE price = (SELECT MIN(price) FROM book);
```

Результат:

```
+-------+------------------+--------+--------+
| title | author           | price  | amount |
+-------+------------------+--------+--------+
| Идиот | Достоевский Ф.М. | 460.00 | 10     |
+-------+------------------+--------+--------+
```

Вложенный запрос определяет минимальную цену книг во всей таблице (это 460.00), а затем в основном запросе для каждой записи проверяется, равна ли цена минимальному значению, если равна, информация о книге включается в результирующую таблицу запроса.

Рекомендация. При использовании вложенного запроса рекомендуется сначала проверить, правильно ли он работает (занести текст запроса в окно кода и нажать черную кнопку **Запустить**), если выдается верный результат – использовать код в качестве вложенного запроса.

Задание (Вывести информацию о книгах, цены ниже или равны средней цене)

Вывести информацию (автора, название и цену) о тех книгах, цены которых меньше или равны средней цене книг на складе. Информацию вывести в отсортированном по убыванию цены виде. Среднее вычислить как среднее по цене книги.

Результат:

```
+------------------+---------------+--------+
| author           | title         | price  |
+------------------+---------------+--------+
| Булгаков М.А.    | Белая гвардия | 540.50 |
| Достоевский Ф.М. | Игрок         | 480.50 |
| Достоевский Ф.М. | Идиот         | 460.00 |
+------------------+---------------+--------+
```

Шаг 3: Использование вложенного запроса в выражении

Ссылка в интернете

Вложенный запрос, возвращающий одно значение, может использоваться в выражениях как обычный операнд, например, к нему можно что-то прибавить, отнять и пр.

Пример

Вывести информацию о книгах, количество которых отличается от среднего количества книг на складе более чем на 3. То есть нужно вывести и те книги, количество которых меньше среднего на 3, и больше среднего на 3.

Запрос:

```
SELECT title, author, amount
FROM book
WHERE ABS(amount - (SELECT AVG(amount) FROM book)) >3;
```

Результат:

```
+----------------------+------------------+--------+
| title                | author           | amount |
+----------------------+------------------+--------+
| Мастер и Маргарита   | Булгаков М.А.    | 3      |
| Братья Карамазовы    | Достоевский Ф.М. | 3      |
| Стихотворения и поэмы | Есенин С.А.     | 15     |
+----------------------+------------------+--------+
```

Задание (Вывести информацию цены больше минимальной, не более чем на 150 рублей)

Вывести информацию (автора, название и цену) о тех книгах, цены которых превышают минимальную цену книги на складе не более чем на 150 рублей в отсортированном по возрастанию цены виде.

Результат:

```
+-----------------+--------------+--------+
```

```
| author            | title          | price  |
+-------------------+----------------+--------+
| Достоевский Ф.М.  | Идиот          | 460.00 |
| Достоевский Ф.М.  | Игрок          | 480.50 |
| Булгаков М.А.     | Белая гвардия  | 540.50 |
+-------------------+----------------+--------+
```

Шаг 4: Вложенный запрос, оператор **IN**

Ссылка в интернете

Вложенный запрос может возвращать несколько значений одного столбца. Оператор `IN` определяет, совпадает ли указанное в логическом выражении значение с одним из значений, содержащихся во вложенном запросе, при этом логическое выражение получает значение истина. Оператор `NOT IN` выполняет обратное действие – выражение истинно, если значение не содержится во вложенном запросе.

Пример

Вывести информацию о книгах тех авторов, общее количество экземпляров книг которых не менее 12.

Запрос:

```
SELECT title, author, amount, price
FROM book
WHERE author IN (SELECT author FROM book GROUP BY author having SUM(amount) >= 12);
```

Результат:

```
+----------------------+------------------+--------+--------+
| title                | author           | amount | price  |
+----------------------+------------------+--------+--------+
| Идиот                | Достоевский Ф.М. | 10     | 460.00 |
| Братья Карамазовы    | Достоевский Ф.М. | 3      | 799.01 |
| Игрок                | Достоевский Ф.М. | 10     | 480.50 |
| Стихотворения и поэмы | Есенин С.А.     | 15     | 650.00 |
+----------------------+------------------+--------+--------+
```

Вложенный запрос отбирает двух авторов (Достоевского и Есенина). А в основном запросе для каждой записи таблицы **book** проверяется, входит ли автор книги в отобранный список, если входит - информация о книге включается в запрос.

Задание (Вывести информацию о книгах, количество которых не повторяется)

Вывести информацию (автора, книгу и количество) о тех книгах, количество которых в таблице **book** не повторяется.

Результат:

```
+--------------+---------------------+--------+
```

```
| author         | title                  | amount |
+----------------+------------------------+--------+
| Булгаков М.А.  | Белая гвардия          | 5      |
| Есенин С.А.    | Стихотворения и поэмы  | 15     |
+----------------+------------------------+--------+
```

Пояснение. Во вложенном запросе отберите те значения столбца **amount**, количество которых, вычисленное с помощью функции **count()**, равно 1.

Шаг 5: Вложенный запрос, операторы **ANY и ALL**

Ссылка в интернете

Вложенный запрос, возвращающий несколько значений одного столбца, можно использовать для отбора записей с помощью операторов ANY и ALL совместно с операциями отношения (=, <>, <=, >=, <, >).

Операторы ANY и ALL используются в SQL для сравнения некоторого значения с результирующим набором вложенного запроса, состоящим из одного столбца. При этом тип данных столбца, возвращаемого вложенным запросом, должен совпадать с типом данных столбца (или выражения), с которым происходит сравнение.

При использовании оператора ANY в результирующую таблицу будут включены все записи, для которых выражение со знаком отношения верно хотя бы для одного элемента результирующего запроса. Как работает оператор ANY():

amount > ANY (10, 12) эквивалентно amount > 10

amount < ANY (10, 12) эквивалентно amount < 12

amount = ANY (10, 12) эквивалентно (amount = 10) OR (amount = 12), а также amount IN (10,12)

amount <> ANY (10, 12) вернет все записи с любым значением amount, включая 10 и 12

При использовании оператора ALL в результирующую таблицу будут включены все записи, для которых выражение со знаком отношения верно для всех элементов результирующего запроса. Как работает оператор ALL:

amount > ALL (10, 12) эквивалентно amount > 12

amount < ALL (10, 12) эквивалентно amount < 10

amount = ALL (10, 12) не вернет ни одной записи, так как эквивалентно (amount = 10) AND (amount = 12)

amount <> ALL (10, 12) вернет все записи кроме тех, в которых amount равно 10 или 12

Важно! Операторы `ALL` и `ANY` можно использовать т**олько с вложенными запросами**. В примерах выше (10, 12) приводится как результат вложенного запроса просто для того, чтобы показать как эти операторы работают. В запросах так записывать нельзя.

Пример

Вывести информацию о книгах тех авторов, общее количество экземпляров книг которых не меньше 12.

Запрос:

```
SELECT title, author, amount, price
FROM book
WHERE author = ANY (SELECT author FROM book GROUP BY author having SUM(amount) >= 12);
```

Результат:

```
+----------------------+------------------+--------+--------+
| title                | author           | amount | price  |
+----------------------+------------------+--------+--------+
| Идиот                | Достоевский Ф.М. | 10     | 460.00 |
| Братья Карамазовы    | Достоевский Ф.М. | 3      | 799.01 |
| Игрок                | Достоевский Ф.М. | 10     | 480.50 |
| Стихотворения и поэмы | Есенин С.А.     | 15     | 650.00 |
+----------------------+------------------+--------+--------+
```

Пояснение:

Вложенный запрос `SELECT author FROM book GROUP BY author having SUM(amount) >= 12` отбирает 2 записи, с фамилиями двух авторов (Достоевский и Есенин), так как общее количество экземпляров книг у них 23 и 15 соответственно.

В условии отбора основного запроса фамилия автора с помощью `= ANY` сравнивается с результатом вложенного запроса (Достоевский и Есенин). Если фамилия автора из основного запроса совпадет с какой-нибудь фамилией результата, то соответствующая запись включается в итоговую таблицу запроса.

Таким образом, наш запрос отобрал все книги Достоевского и Есенина, так как их общее количество превышает 12. (Книг Булгакова всего 8).

Если в наш запрос вместо `ANY` вставить `ALL`, то в результирующую таблицу ничего включено не будет, так как фамилия автора одновременно не может быть равна и Есенину, и Достоевскому.

Вывести информацию о книгах тех авторов, общее количество экземпляров книг которых больше или равно 12, также можно, используя вместо `=ANY` оператор `IN`.

Запрос:

```
SELECT title, author, amount, price
FROM book
WHERE author IN (SELECT author FROM book GROUP BY author having SUM(amount) >= 12);
```

Задание (Вывести информацию о книгах, средняя цена выше, чем средняя цена на складе)

Вывести информацию о книгах (автор, название, цена) только тех авторов, средняя цена книг которых выше, чем средняя цена книг на складе в целом.

Результат:

author	title	price
Булгаков М.А.	Мастер и Маргарита	670.99
Булгаков М.А.	Белая гвардия	540.50
Есенин С.А.	Стихотворения и поэмы	650.00

Пояснение. В запросе использовать два вложенных запроса:

-сначала посчитать среднюю цену книг на складе (`SELECT ___`);

-затем отобрать авторов, средняя цена книг которых выше средней (средняя цена вычислена в первом запросе, этот запрос использовать в условии отбора) (`SELECT ... (SELECT __ )`).

В основном запросе отобрать книги тех авторов, которые отбираются вторым вложенным запросом, при этом каждый вложенный запрос должен заключаться в круглые скобки:

```
SELECT  ...
FROM ...
WHERE author = ... (SELECT ...
                    (SELECT ___ ))
```

Шаг 6: Вложенный запрос после **SELECT**

Ссылка в интернете

Вложенный запрос может располагаться после ключевого слова `SELECT`. В этом случае результат выполнения запроса выводится в отдельном столбце результирующей таблицы. При этом результатом запроса может быть либо одно значение, тогда оно будет повторяться во всех строках, либо несколько значений, количество которых равно количеству отобранных записей в основном запросе.

Пример

Вывести информацию о книгах, количество которых отличается от среднего количества книг на складе более чем на 3, а также указать среднее значение количества книг.

Запрос:

```
SELECT title, author, amount,
    (SELECT AVG(amount) FROM book) AS Среднее_количество
FROM book
```

```
WHERE abs(amount - (SELECT AVG(amount) FROM book)) >3;
```

Результат:

```
+----------------------+------------------+--------+--------------------+
| title                | author           | amount | Среднее_количество |
+----------------------+------------------+--------+--------------------+
| Мастер и Маргарита   | Булгаков М.А.    | 3      | 7.6667             |
| Братья Карамазовы    | Достоевский Ф.М. | 3      | 7.6667             |
| Стихотворения и поэмы | Есенин С.А.     | 15     | 7.6667             |
+----------------------+------------------+--------+--------------------+
```

Во вложенном запросе вычисляется среднее количество книг на складе. Этот запрос используется и в условии отбора, и для создания столбца **Среднее_количество** в результирующей таблице запроса. Значения столбца одинаковы во всех строках, поскольку вложенный запрос возвращает одно значение.

Среднее количество в виде дробного числа выглядит не очень правильно. Полученное значение можно округлить "вниз" - до ближайшего меньшего целого.

Запрос:

```
SELECT title, author, amount,
    FLOOR((SELECT AVG(amount) FROM book)) AS Среднее_количество
FROM book
WHERE abs(amount - (SELECT AVG(amount) FROM book)) >3;
```

Результат:

```
+----------------------+------------------+--------+--------------------+
| title                | author           | amount | Среднее_количество |
+----------------------+------------------+--------+--------------------+
| Мастер и Маргарита   | Булгаков М.А.    | 3      | 7                  |
| Братья Карамазовы    | Достоевский Ф.М. | 3      | 7                  |
| Стихотворения и поэмы | Есенин С.А.     | 15     | 7                  |
+----------------------+------------------+--------+--------------------+
```

Задание (Посчитать сколько и каких книг заказать у поставщика)

Посчитать сколько и каких книг нужно заказать поставщикам, чтобы на складе было одинаковое количество каждой книги, равное максимальному значению из всех количеств книг, хранящихся на складе. Столбцу с количеством заказываемых книг присвоить имя **Заказ**.

Результат:

```
+--------------------+------------------+--------+--------+
| title              | author           | amount | Заказ  |
+--------------------+------------------+--------+--------+
| Мастер и Маргарита | Булгаков М.А.    | 3      | 12     |
| Белая гвардия      | Булгаков М.А.    | 5      | 10     |
| Идиот              | Достоевский Ф.М. | 10     | 5      |
| Братья Карамазовы  | Достоевский Ф.М. | 3      | 12     |
| Игрок              | Достоевский Ф.М. | 10     | 5      |
+--------------------+------------------+--------+--------+
```

Пояснение. Поскольку книгу с максимальным количеством экземпляров заказывать не нужно, в условии отбора запроса укажите, что книгу с максимальным значением количества в результирующую таблицу не включать.

Комментарии учащихся:

1.

Ксения Бордукова
почему нельзя написать так
(SELECT (max(amount)-amount) FROM book) ?

Галина Озерова
@Ксения_Бордукова, Вы хотите во вложенном запросе отнять значение из основного запроса. А это нельзя сделать . **(SELECT (max(amount)-amount) FROM book) --> (SELECT max(amount) FROM book) -amount**, так будет верно, так как amount вычитается на уровне основного запроса.

Evgeny Skotarenko
@Галина_Озерова,
большое спасибо

Дмитрий Захаров
Большое спасибо, это очень помогло привести в порядок понимание структуры запроса. А ведь кажется так неочевидно. Я бы вообще для таких замечаний оформлял отдельный блок..

Галина Озерова
@Дмитрий_Захаров, Я его закреплю просто

Шаг 7: Задание (Придумать запрос к таблице book)

Ссылка в интернете

Придумайте один или несколько запросов к нашей таблице **book**, используя вложенные запросы. Проверьте, правильно ли они работают.

При желании можно формулировку запросов разместить в комментариях.

Размещенные задания можно использовать для закрепления материала урока.

Оценивайте понравившиеся Вам запросы.

В последнем модуле создан отдельный урок, в котором мы разместим запросы, набравшие наибольшее количество лайков.

УРОК 1.5: Запросы корректировки данных

Шаг 1: Содержание урока

Ссылка в интернете

SQL позволяет не только выбирать данные из таблиц базы данных, но и корректировать информацию в них. Для этого используются запросы корректировки данных, с помощью которых можно:

- создать пустую таблицу;
- добавить в таблицу записи как совокупность значений;
- добавить записи из другой таблицы;
- добавить записи из другой таблицы, используя вложенный запрос;
- изменить значения в одном столбце;
- изменить значения в нескольких столбцах;
- изменить данные, используя несколько таблиц;
- удалить записи из таблицы;
- создать таблицу на основе данных других таблиц.

На данном уроке будут рассматриваться запросы для реализации типичных для склада действий:

- получение нового товара (добавление, обновление, удаление данных);
- продажа товара (обновление данных);
- формирование заказа на новый товар (создание таблицы).

Структура и наполнение таблицы

В запросах будет участвовать таблица **book** (создание, заполнение):

book_id	Title	Author	price	amount
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(50)	VARCHAR(30)	DECIMAL(8,2)	INT
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Шаг 2: Создание пустой таблицы

Ссылка в интернете

Создание таблицы осуществляется с помощью запроса `CREATE`, подробно рассмотренного в первом уроке модуля.

Задание (Создать таблицу supply)

Создать таблицу поставка (**supply**), которая имеет ту же структуру, что и таблица **book**.

Поле	Тип, описание
supply_id	INT PRIMARY KEY AUTO_INCREMENT
title	VARCHAR(50)
author	VARCHAR(30)
price	DECIMAL(8, 2)
amount	INT

Комментарии учащихся:

1.
Sergey Agalakov
Вдруг кто не знает, после VALUES можно перечислить все значения, без создания кучи INSERT'ов
INSERT INTO MyTable (Column1, Column2) VALUES
(Value1, Value2), (Value1, Value2)
2.
Leo Matyushkin
Обратите внимание, что Stepik поддерживает множественную установку курсоров – зажав [Ctrl], можно кликнуть в нескольких местах и вставить на всех позициях нужный символ.

Шаг 3: Добавление записей в таблицу

Ссылка в интернете

Добавление записей в таблицу осуществляется с помощью запроса `INSERT`, подробно рассмотренного в первом уроке.

Задание (Занести в таблицу supply четыре записи)

Занесите в таблицу **supply** четыре записи, чтобы получилась следующая таблица:

supply_id	title	Author	price	amount
1	Лирика	Пастернак Б.Л.	518.99	2
2	Черный человек	Есенин С.А.	570.20	6
3	Белая гвардия	Булгаков М.А.	540.50	7

4	Идиот	Достоевский Ф.М.	360.80	3

Пояснение. Каждая строка вставляется отдельным SQL запросом, запросы обязательно разделять точкой с запятой. Для просмотра полученной таблицы после запросов на добавление вставить запрос на выборку всех данных из таблицы **supply**.

Результат :

Affected rows: 1

Affected rows: 1

Affected rows: 1

Affected rows: 1

Query result:

```
+-----------+-----------------+-------------------+--------+--------+
| supply_id | title           | author            | price  | amount |
+-----------+-----------------+-------------------+--------+--------+
| 1         | Лирика          | Пастернак Б.Л.    | 518.99 | 2      |
| 2         | Черный человек  | Есенин С.А.       | 570.20 | 6      |
| 3         | Белая гвардия   | Булгаков М.А.     | 540.50 | 7      |
| 4         | Идиот           | Достоевский Ф.М.  | 360.80 | 3      |
+-----------+-----------------+-------------------+--------+--------+
```

Шаг 4: Добавление записей из другой таблицы (INSERT INTO … SELECT…)

Ссылка в интернете

С помощью запроса на добавление можно не только добавить в таблицу конкретные значения (список VALUES), но и записи из другой таблицы, отобранные с помощью запроса на выборку. В этом случае вместо раздела VALUES записывается запрос на выборку, начинающийся с SELECT. В нем можно использовать WHERE, GROUP BY, ORDER BY.

Правила соответствия между полями таблицы и вставляемыми значениями из запроса:

- количество полей в таблице и количество полей в запросе должны совпадать;
- должно существовать прямое соответствие между позицией одного и того же элемента в обоих списках, поэтому первый столбец запроса должен относиться к первому столбцу в списке столбцов таблицы, второй – ко второму столбцу и т.д.
- типы столбцов запроса должны быть совместимы с типами данных соответствующих столбцов таблицы (целое число можно занести в поле типа DECIMAL, обратная операция – недопустима).

Пример

Занести все книги из таблицы **supply** в таблицу **book**.

Запрос:

```
INSERT INTO book (title, author, price, amount)
    SELECT title, author, price, amount
    FROM supply;
SELECT * FROM book;
```

Результат:

Affected rows: 4

Query result:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15
6	Лирика	Пастернак Б.Л.	518.99	2
7	Черный человек	Есенин С.А.	570.20	6
8	Белая гвардия	Булгаков М.А.	540.50	7
9	Идиот	Достоевский Ф.М.	360.80	3

Affected rows: 9

С помощью этого запроса в таблицу **book** включены все книги из **supply**, даже те, которые в **book** уже есть («Белая гвардия» и «Идиот»). В результате в таблице одна и та же книга, например «Белая гвардия», имеет код 2 и 8. Для реляционной модели это нежелательная ситуация. Устранить эту проблему можно с помощью вложенных запросов.

Задание (Добавить книги авторов из таблицы supply в таблицу book)

Добавить из таблицы supply в таблицу book, все книги, кроме книг, написанных Булгаковым и Достоевским.

Результат:

Affected rows: 2

Query result:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15
6	Лирика	Пастернак Б.Л.	518.99	2
7	Черный человек	Есенин С.А.	570.20	6

```
+---------+----------------------+------------------+--------+--------+
```

Шаг 5: Добавление записей, вложенные запросы

Ссылка в интернете

В запросах на добавление можно использовать вложенные запросы.

Пример

Занести из таблицы **supply** в таблицу **book** только те книги, названия которых отсутствуют в таблице **book.**

Запрос:

```
INSERT INTO book (title, author, price, amount)
    SELECT title, author, price, amount
    FROM supply
    WHERE title NOT IN (SELECT title from book);
SELECT * FROM book;
```

Результат:

Affected rows: 2

Query result:

```
+---------+----------------------+------------------+--------+--------+
| book_id | title                | author           | price  | amount |
+---------+----------------------+------------------+--------+--------+
| 1       | Мастер и Маргарита   | Булгаков М.А.    | 670.99 | 3      |
| 2       | Белая гвардия        | Булгаков М.А.    | 540.50 | 5      |
| 3       | Идиот                | Достоевский Ф.М. | 460.00 | 10     |
| 4       | Братья Карамазовы    | Достоевский Ф.М. | 799.01 | 2      |
| 5       | Стихотворения и поэмы | Есенин С.А.      | 650.00 | 15     |
| 6       | Лирика               | Пастернак Б.Л.   | 518.99 | 2      |
| 7       | Черный человек       | Есенин С.А.      | 570.20 | 6      |
+---------+----------------------+------------------+--------+--------+
```

Вложенным запросом отбираются все названия книг, которые есть в таблице **book**. Основным запросом SELECT из таблицы **supply** выбираются книги, названия которых нет в результате вложенного запроса. Отобранные записи добавляются в конец таблицы **book** запросом на добавление INSERT.

Задание (Занести те, книги, авторов которых нет в таблице)

Занести из таблицы **supply** в таблицу **book** только те книги, авторов которых нет в **book.**

Результат:

Affected rows: 1

Query result:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15
6	Лирика	Пастернак Б.Л.	518.99	2

Шаг 6: Запросы на обновление **UPDATE**

Ссылка в интернете

Под обновлением данных подразумевается изменение значений в существующих записях таблицы. При этом возможно как изменение значений полей в группе строк (даже всех строк таблицы), так и правка значения поля отдельной строки.

Изменение записей в таблице реализуется с помощью запроса `UPDATE`. Простейший запрос на обновление выглядит так:

```
UPDATE таблица SET поле = выражение
```

где
таблица – имя таблицы, в которой будут проводиться изменения;
поле – поле таблицы, в которое будет внесено изменение;
выражение – выражение, значение которого будет занесено в поле.

Пример

Уменьшить на 30% цену книг в таблице **book**.

Запрос:

```
UPDATE book SET price = 0.7 * price;

SELECT * FROM book;
```

Результат:

```
Affected rows: 5

Query result:
```

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	469.69	3
2	Белая гвардия	Булгаков М.А.	378.35	5
3	Идиот	Достоевский Ф.М.	322.00	10
4	Братья Карамазовы	Достоевский Ф.М.	559.31	2
5	Стихотворения и поэмы	Есенин С.А.	455.00	15

С помощью запросов на обновление можно изменять не все записи в таблице (как в предыдущем запросе), а только часть из них. Для этого в запрос включается ключевое слово `WHERE`, после которого указывается условие отбора строк для изменения.

Пример

Уменьшить на 30% цену тех книг в таблице **book**, количество которых меньше 5.

Запрос:

```
UPDATE book SET price = 0.7 * price
WHERE amount < 5;
SELECT * FROM book;
```

Результат:

Affected rows: 2

Query result:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	469.69	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	559.31	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

В этом запросе обновляется только 2 записи (цена книг «Мастер и Маргарита» и «Братья Карамазовы»).

Задание (Уменьшить на 10% цену книг из интервала)

Уменьшить на 10% цену тех книг в таблице **book**, количество которых принадлежит интервалу от 5 до 10 включительно.

Результат:

Affected rows: 2

Query result:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	486.45	5
3	Идиот	Достоевский Ф.М.	414.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Комментарии учащихся:

1.
Denis Pobelkin
Г-да у кого не получилось, прошу сюда
https://dev.mysql.com/doc/refman/8.0/en/comparison-operators.html#operator_between

Шаг 7: Запросы на обновление нескольких столбцов **UPDATE**

Ссылка в интернете

Запросом UPDATE можно обновлять значения нескольких столбцов одновременно. В этом случае простейший запрос будет выглядеть так:

UPDATE таблица SET поле1 = выражение1, поле2 = выражение2

На складе, кроме хранения и получения книг, выполняется их оптовая продажа. Для реализации этого действия включим дополнительный столбец **buy** в таблицу **book**:

book_id	Title	Author	Price	amount	buy
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(50)	VARCHAR(30)	DECIMAL(8,2)	INT	int
1	Мастер и Маргарита	Булгаков М.А.	670.99	3	0
2	Белая гвардия	Булгаков М.А.	540.50	5	3
3	Идиот	Достоевский Ф.М.	460.00	10	8
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2	0
5	Стихотворения и поэмы	Есенин С.А.	650.00	15	18

Пример

В столбце **buy** покупатель указывает количество книг, которые он хочет приобрести. Для каждой книги, выбранной покупателем, необходимо уменьшить ее количество на складе на указанное в столбце **buy** количество, а в столбец **buy** занести 0.

Запрос:

```
UPDATE book SET amount = amount - buy,
               buy = 0;
SELECT * FROM book;
```

Результат:

Affected rows: 3

Query result:

book_id	title	author	price	amount	buy
1	Мастер и Маргарита	Булгаков М.А.	670.99	3	0
2	Белая гвардия	Булгаков М.А.	540.50	2	0
3	Идиот	Достоевский Ф.М.	460.00	2	0
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2	0
5	Стихотворения и поэмы	Есенин С.А.	650.00	-3	0

Как видно из таблицы, без проверки данных, которые занесены в столбец, нельзя запускать запрос на обновление (может получиться отрицательное значение количества).

Задание (Скорректировать значения для покупателей)

В таблице **book** необходимо скорректировать значение для покупателя в столбце **buy** таким образом, что бы оно не превышало допустимый остаток в столбце **amount**. А цену тех книг, которые покупатель не заказывал, снизить на 10%, округлив полученную цену до двух знаков после запятой.

Результат:

Affected rows: 1

Query result:

book_id	title	author	price	amount	buy
1	Мастер и Маргарита	Булгаков М.А.	603.89	3	0
2	Белая гвардия	Булгаков М.А.	540.50	5	3
3	Идиот	Достоевский Ф.М.	460.00	10	8
4	Братья Карамазовы	Достоевский Ф.М.	719.11	2	0
5	Стихотворения и поэмы	Есенин С.А.	650.00	15	15

Пояснение. Запрос на обновление количества книг должен корректировать значения в столбце **buy** таблицы **book** следующим образом:

- если покупатель заказал количество книг больше, чем есть на складе, то заменить значение **buy** на имеющееся на складе количество **amount**;
- если покупатель хочет купить количество книг меньшее или равное количеству книг на складе, то значение **buy** изменять не надо.

Шаг 8: Запросы на обновление, несколько таблиц **(UPDATE)**

Ссылка в интернете

В запросах на обновление можно использовать несколько таблиц, но тогда

- для столбцов, имеющих одинаковые имена, необходимо указывать имя таблицы, к которой они относятся, например, **book.price** – столбец **price** из таблицы **book**, **supply.price** – столбец **price** из таблицы **supply**;
- все таблицы, используемые в запросе, нужно перечислить после ключевого слова UPDATE;
- в запросе обязательно условие WHERE, в котором указывается условие при котором обновляются данные.

Пример

Если в таблице **supply** есть те же книги, что и в таблице **book**, добавлять эти книги в таблицу **book** не имеет смысла. Необходимо увеличить их количество на значение столбца **amount** таблицы **supply**.

Запрос:

```
UPDATE book, supply SET book.amount = book.amount + supply.amount
WHERE book.title = supply.title AND book.author = supply.author;
SELECT * FROM book;
```

Результат:

Affected rows: 2

Query result:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	12
3	Идиот	Достоевский Ф.М.	460.00	13
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

В этом запросе увеличилось количество двух книг: «Белая гвардия», которая в **supply** имеет ту же цену, и «Идиот», но цена этой книги в таблицах **book** и **supply** отличается. Для этой книги нужно пересчитать цену.

Задание (Для одинаковых книг увеличить количество)

Для одинаковых книг в таблицах **book** и **supply** не только увеличить их количество в таблице **book** (увеличить их количество на значение столбца **amount** таблицы **supply**), но и пересчитать их цену (для каждой книги найти сумму цен из таблиц **book** и **supply** и разделить на 2).

Результат:

Affected rows: 2

Query result:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	12
3	Идиот	Достоевский Ф.М.	410.40	13
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

Пояснение. Пересчет для книг с одинаковым названием и ценой не повлияет на результат, поэтому в запросе не обязательно рассматривать два случая: когда цена у одинаковых книг равна и когда нет.

Комментарии учащихся:

1.

Кирилл салтовский

Почему после Update мы пишем обе таблицы, если нам надо обновить только таблицу book? для того чтоб sql понимал с какими таблицами ему работать?

Галина Озерова

@кирилл_салтовский, Да, если таблицу не указать, то нельзя будет обратиться к ее столбцам.

Шаг 9: Запросы на удаление (**DELETE**)

Ссылка в интернете

Запросы корректировки данных позволяют удалить одну или несколько записей из таблицы. Простейший запрос на удаление имеет вид:

```
DELETE FROM таблица;
```

Этот запрос удаляет все записи из указанной после `FROM` таблицы.

Пример

После того, как информация о книгах из таблицы **supply** перенесена в **book** , необходимо очистить таблицу **supply**.

Запрос:

```
DELETE FROM supply;
```

```
SELECT * FROM supply;
```

Результат:

```
Affected rows: 4
```

```
Affected rows: 0
```

Из таблицы удалены все записи. Запрос на выборку отобрал 0 записей.

Запрос на удаления позволяет удалить не все записи таблицы, а только те, которые удовлетворяют условию, указанному после ключевого слова `WHERE`:

DELETE FROM таблица

WHERE условие;

Пример

Удалить из таблицы **supply** все книги, названия которых есть в таблице **book**.

Запрос:

```
DELETE FROM supply
WHERE title IN (SELECT title FROM book);
SELECT * FROM supply;
```

Результат:

Affected rows: 2

Query result:

```
+-----------+--------------------------+------------------+--------+--------+
| supply_id | title                    | author           | price  | amount |
+-----------+--------------------------+------------------+--------+--------+
| 1         | Лирика                   | Пастернак Б.Л.   | 518.99 | 2      |
| 2         | Черный человек           | Есенин С.А.      | 570.20 | 6      |
| 5         | Преступление и наказание | Достоевский Ф.М. | 670.99 | 5      |
+-----------+--------------------------+------------------+--------+--------+
```

Из таблицы **supply** удалены две записи о книгах «Белая гвардия» и «Идиот».

Задание (Удалить из таблицы книги при условии)

Удалить из таблицы **supply** книги тех авторов, общее количество экземпляров книг которых в таблице **book** превышает 10.

Результат:

Affected rows: 2

Query result:

```
+-----------+---------------+----------------+--------+--------+
| supply_id | title         | author         | price  | amount |
+-----------+---------------+----------------+--------+--------+
| 1         | Лирика        | Пастернак Б.Л. | 518.99 | 2      |
| 3         | Белая гвардия | Булгаков М.А.  | 540.50 | 7      |
+-----------+---------------+----------------+--------+--------+
```

Шаг 10: Запросы на создание таблицы (**CREATE TABLE… SELECT…**)

Ссылка в интернете

Новая таблица может быть создана на основе данных из другой таблицы. Для этого используется запрос SELECT, результирующая таблица которого и будет новой таблицей базы данных. При

этом имена столбцов запроса становятся именами столбцов новой таблицы. Запрос на создание новой таблицы имеет вид:

```
CREATE TABLE имя_таблицы AS
SELECT ...
```

Пример

Создать таблицу заказ (**ordering**), куда включить авторов и названия тех книг, количество которых в таблице **book** меньше 4. Для всех книг указать одинаковое количество 5.

Запрос:

```
CREATE TABLE ordering AS
SELECT author, title, 5 AS amount
FROM book
WHERE amount < 4;
SELECT * FROM ordering;
```

Результат:

Affected rows: 2

Query result:

```
+------------------+--------------------+--------+
| author           | title              | amount |
+------------------+--------------------+--------+
| Булгаков М.А.    | Мастер и Маргарита | 5      |
| Достоевский Ф.М. | Братья Карамазовы  | 5      |
+------------------+--------------------+--------+
```

При создании таблицы можно использовать вложенные запросы как после SELECT, так и после WHERE.

Пример

Создать таблицу заказ (**ordering**), куда включить авторов и названия тех книг, количество которых в таблице **book** меньше 4. Для всех книг указать одинаковое значение - среднее количество книг в таблице **book**.

Запрос:

```
CREATE TABLE ordering AS
SELECT author, title,
    (SELECT round(AVG(amount)) FROM book) AS amount
FROM book
WHERE amount < 4;
```

```
SELECT * FROM ordering;
```

Результат:

Affected rows: 2

Query result:

```
+------------------+--------------------+--------+
| author           | title              | amount |
+------------------+--------------------+--------+
| Булгаков М.А.    | Мастер и Маргарита | 7      |
| Достоевский Ф.М. | Братья Карамазовы  | 7      |
+------------------+--------------------+--------+
```

Задание (Создать таблицу Заказ – ordering)

Создать таблицу заказ (**ordering**), куда включить авторов и названия тех книг, количество которых в таблице **book** меньше среднего количества книг в таблице **book**. Для всех книг указать одинаковое значение - среднее количество книг в таблице **book**.

Результат:

Affected rows: 3

Query result:

```
+------------------+--------------------+--------+
| author           | title              | amount |
+------------------+--------------------+--------+
| Булгаков М.А.    | Мастер и Маргарита | 7      |
| Булгаков М.А.    | Белая гвардия      | 7      |
| Достоевский Ф.М. | Братья Карамазовы  | 7      |
+------------------+--------------------+--------+
```

Шаг 11: Задание (Придумать запрос корректировки данных)

Ссылка в интернете

Придумайте один или несколько запросов корректировки данных к таблицам **book** и **supply**. Проверьте, правильно ли они работают.

При желании можно формулировку запросов разместить в комментариях.

Размещенные задания можно использовать для закрепления материала урока.

Оценивайте понравившиеся Вам запросы.

В последнем модуле создан отдельный урок, в котором мы разместим запросы, набравшие наибольшее количество лайков.

УРОК 1.6: Таблица "Командировки", запросы на выборку

Шаг 1: Содержание урока

Ссылка в интернете

В этом уроке на каждом шаге нужно реализовать запросы на выборку для таблицы, в которой представлена информация о командировках сотрудников некоторой организации:

- Вывести информацию о командировках тех сотрудников, фамилия которых заканчивается на букву «а».
- Вывести в алфавитном порядке фамилии, имена и отчества тех сотрудников, которые были в командировке в Москве.
- Для каждого города посчитать, сколько раз сотрудники в нем были.
- Вывести два города, в которых чаще всего были в командировках сотрудники.
- Вывести информацию о длительности командировок сотрудников.
- Вывести информацию о командировках сотрудника(ов), которые были самыми короткими по времени.
- Вывести информацию о командировках, начало и конец которых относятся к одному месяцу.
- Вывести номер месяца и количество командировок, первый день которых приходился на этот месяц.
- Вывести сумму суточных для командировок сотрудников.
- Вывести фамилию с инициалами и общую сумму суточных, полученных за все командировки для тех сотрудников, которые были в командировках больше чем 3 раза.

Структура и наполнение таблицы

Таблица **trip**, в которой представлена информация о командировках сотрудников некоторой организации (фамилия сотрудника, город, куда он ездил, размер суточных, даты первого и последнего дня командировки):

trip_id	Name	city	per_diem	date_first	date_last
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(30)	VARCHAR(25)	DECIMAL(8,2)	DATE	DATE
1	Баранов П.Е.	Москва	700	2020-01-12	2020-01-17
2	Абрамова К.А.	Владивосток	450	2020-01-14	2020-01-27
3	Семенов И.В.	Москва	700	2020-01-23	2020-01-31
4	Ильиных Г.Р.	Владивосток	450	2020-01-12	2020-02-02
5	Колесов С.П.	Москва	700	2020-02-01	2020-02-06
6	Баранов П.Е.	Москва	700	2020-02-14	2020-02-22
7	Абрамова К.А.	Москва	700	2020-02-23	2020-03-01
8	Лебедев Т.К.	Москва	700	2020-03-03	2020-03-06

9	Колесов С.П.	Новосибирск	450	2020-02-27	2020-03-12
10	Семенов И.В.	Санкт-Петербург	700	2020-03-29	2020-04-05
11	Абрамова К.А.	Москва	700	2020-04-06	2020-04-14
12	Баранов П.Е.	Новосибирск	450	2020-04-18	2020-05-04
13	Лебедев Т.К.	Томск	450	2020-05-20	2020-05-31
14	Семенов И.В.	Санкт-Петербург	700	2020-06-01	2020-06-03
15	Абрамова К.А.	Санкт-Петербург	700	2020-05-28	2020-06-04
16	Федорова А.Ю.	Новосибирск	450	2020-05-25	2020-06-04
17	Колесов С.П.	Новосибирск	450	2020-06-03	2020-06-12
18	Федорова А.Ю.	Томск	450	2020-06-20	2020-06-26
19	Абрамова К.А.	Владивосток	450	2020-07-02	2020-07-13
20	Баранов П.Е.	Воронеж	450	2020-07-19	2020-07-25

Пояснение

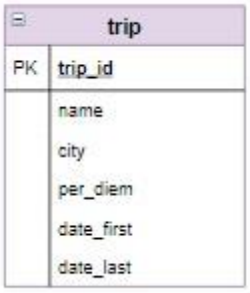
Тип **DATE** – позволяет описать дату в формате ГГГГ-ММ-ДД, например, 2020-02-02. При вставке данных в таблицу с помощью INSERT INTO ... VALUES значение даты заключается в кавычки.

Шаг 2: Задание (Вывести сотрудников, у которых фамилия заканчивается на букву «а»)

Ссылка в интернете

Вывести из таблицы **trip** информацию о командировках тех сотрудников, фамилия которых заканчивается на букву «а», в отсортированном по убыванию даты последнего дня командировки виде. В результат включить столбцы **name, city, per_diem, date_first, date_last.**

Структура таблицы:



Sql запрос, создающий таблицу trip :

```
CREATE TABLE trip (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(30),
    city VARCHAR(25),
    per_diem DECIMAL (8,2),
    date_first DATE,
    date_last DATE
);
```

Результат:

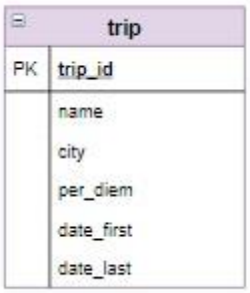
```
+---------------+-----------------+----------+------------+------------+
| name          | city            | per_diem | date_first | date_last  |
+---------------+-----------------+----------+------------+------------+
| Абрамова К.А. | Владивосток     | 450.00   | 2020-07-02 | 2020-07-13 |
| Федорова А.Ю. | Томск           | 450.00   | 2020-06-20 | 2020-06-26 |
| Абрамова К.А. | Санкт-Петербург | 700.00   | 2020-05-28 | 2020-06-04 |
| Федорова А.Ю. | Новосибирск     | 450.00   | 2020-05-25 | 2020-06-04 |
| Абрамова К.А. | Москва          | 700.00   | 2020-04-06 | 2020-04-14 |
| Абрамова К.А. | Москва          | 700.00   | 2020-02-23 | 2020-03-01 |
| Абрамова К.А. | Владивосток     | 450.00   | 2020-01-14 | 2020-01-27 |
+---------------+-----------------+----------+------------+------------+
```

Шаг 3: Задание (Вывести сотрудников, которые были в командировке)

Ссылка в интернете

Вывести в алфавитном порядке фамилии и инициалы тех сотрудников, которые были в командировке в Москве.

Структура таблицы:



Sql запрос, создающий таблицу trip :

```
CREATE TABLE trip (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(30),
    city VARCHAR(25),
    per_diem DECIMAL (8,2),
    date_first DATE,
    date_last DATE
);
```

Результат:

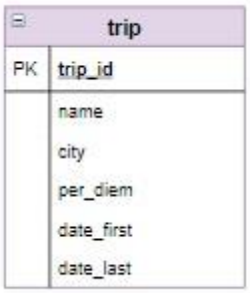
```
+---------------+
| name          |
+---------------+
| Абрамова К.А. |
| Баранов П.Е.  |
| Колесов С.П.  |
| Лебедев Т.К.  |
| Семенов И.В.  |
+---------------+
```

Шаг 4: Задание (Посчитать сколько раз были в городе сотрудники)

Ссылка в интернете

Для каждого города посчитать, сколько раз сотрудники в нем были. Информацию вывести в отсортированном в алфавитном порядке по названию городов. Вычисляемый столбец назвать **Количество**.

Структура таблицы: **Sql запрос, создающий таблицу trip :**




```
CREATE TABLE trip (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(30),
    city VARCHAR(25),
    per_diem DECIMAL (8,2),
    date_first DATE,
    date_last DATE
);
```

Результат:

```
+-----------------+------------+
| city            | Количество |
+-----------------+------------+
| Владивосток     | 3          |
| Воронеж         | 1          |
| Москва          | 7          |
| Новосибирск     | 4          |
| Санкт-Петербург | 3          |
| Томск           | 2          |
+-----------------+------------+
```

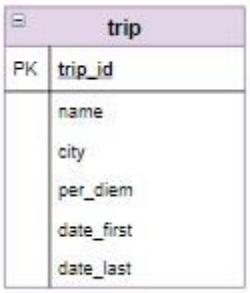
Шаг 5: Функции **LIMIT в ORDER BY**

Ссылка в интернете

Задание (Вывести два города)

Вывести два города, в которых чаще всего были в командировках сотрудники. Вычисляемый столбец назвать **Количество**.

Структура таблицы:



Sql запрос, создающий таблицу trip :

```
CREATE TABLE trip (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(30),
    city VARCHAR(25),
    per_diem DECIMAL (8,2),
    date_first DATE,
    date_last DATE
);
```

Результат:

```
+-------------+------------+
| city        | Количество |
+-------------+------------+
| Москва      | 7          |
| Новосибирск | 4          |
+-------------+------------+
```

Пояснение

1. Для запроса указать сортировку по убыванию количества командировок.

2. Ограничить вывод двумя строками, для этого используется ключевое слово `LIMIT`, после которого указывается количество строк. `LIMIT` размещается после перечисления полей в `ORDER BY`. Результирующая таблица будет иметь количество строк не более указанного после `LIMIT`.

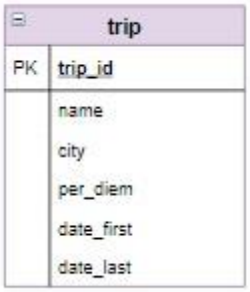
Шаг 6: Функция **DATEDIFF(дата_1, дата_2).**

Ссылка в интернете

Задание (Вывести информацию о командировках без Москвы и Санкт-Петербурга)

Вывести информацию о командировках во все города кроме Москвы и Санкт-Петербурга (фамилии и инициалы сотрудников, город, длительность командировки в днях, при этом первый и последний день относится к периоду командировки). Информацию вывести в упорядоченном по убыванию длительности поездки, а потом по убыванию названий городов (в обратном алфавитном порядке).

Структура таблицы:



Sql запрос, создающий таблицу `trip`:

```
CREATE TABLE trip (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(30),
    city VARCHAR(25),
    per_diem DECIMAL (8,2),
    date_first DATE,
    date_last DATE
);
```

Результат:

```
+--------------+-------------+--------------+
| name         | city        | Длительность |
+--------------+-------------+--------------+
| Ильиных Г.Р. | Владивосток | 22           |
| Баранов П.Е. | Новосибирск | 17           |
| Колесов С.П. | Новосибирск | 15           |
| Абрамова К.А. | Владивосток | 14          |
| Лебедев Т.К. | Томск       | 12           |
| Абрамова К.А. | Владивосток | 12          |
| Федорова А.Ю. | Новосибирск | 11          |
| Колесов С.П. | Новосибирск | 10           |
| Федорова А.Ю. | Томск       | 7            |
| Баранов П.Е. | Воронеж     | 7            |
+--------------+-------------+--------------+
```

Пояснение.

1. Для вычитания двух дат используется функция **`DATEDIFF(дата_1, дата_2)`**, результатом которой является количество дней между **дата_1** и **дата_2**. Например,

`DATEDIFF('2020-04-01', '2020-03-28')=4`

`DATEDIFF('2020-05-09','2020-05-01')=8`

2. Увеличьте разницу на 1, чтобы включить первый день командировки.

Комментарии учащихся:

1.

Елена Рудакова

добрый день!

Скажите ,пожалуйста, почему если прописать WHERE city <> ('Москва','Санкт-Петербург'), то код не работает, а если заменить <> на **NOT IN** - то всё ОК?

Никита Лень

@Елена_Рудакова, потому что оператор NOT IN и IN проверяет, не входит ли или входит поле city в написанный список, а если Вы используете операторы типа равно, больше, меньше, не равно, то надо одно значение ставить или перечислять их через and. Например, city <> 'Москва' and city <> 'Санкт-Петербург'

Nodirxon Amanxojayev

@Никита_Лень, у меня почему то не сработало с не равенствами пришлось not in использовать

2.

Aleksey Krivorot

Подскажите, почему решение данным способом выдает ошибку:

select name, city, datediff(date_last+1, date_first) as Длительность from trip
where city not in ('Москва','Санкт-Петербург')
order by Длительность desc, city desc;

```
Query result:
+---------------+-------------+--------------+
| name          | city        | Длительность |
+---------------+-------------+--------------+
| Ильиных Г.Р.  | Владивосток | 22           |
| Баранов П.Е.  | Новосибирск | 17           |
| Колесов С.П.  | Новосибирск | 15           |
| Абрамова К.А. | Владивосток | 14           |
| Абрамова К.А. | Владивосток | 12           |
| Федорова А.Ю. | Новосибирск | 11           |
| Колесов С.П.  | Новосибирск | 10           |
| Федорова А.Ю. | Томск       | 7            |
| Баранов П.Е.  | Воронеж     | 7            |
| Лебедев Т.К.  | Томск       | None         |
+---------------+-------------+--------------+
Affected rows: 10
Свернуть
```

и в чем существенное отличие от данного способа:

select name, city, datediff(date_last, date_first)+1 as Длительность from trip
where city not in ('Москва','Санкт-Петербург')
order by Длительность desc, city desc;

Галина Озерова

@Aleksey_Krivorot, сложение даты с числом осуществляется с помощью функции DATE_ADD, а простое сложение верно только если не при сложении не меняется месяц. В последней записи date_last - последний день месяца, поэтому не считается правильно. Функция сложения дат будет рассмотрена позже.

3.

Алексей Стецко

я, если честно, не понимаю момента (это было уже в предыдущих заданиях), когда нужно сортировать по убыванию одно и другое. Это же могут быть два противоречащих друг другу результата. Как я могу поставить по убыванию длительность, а потом города? Если это две отдельные операции, то почему в задание включена только одна сортировка?

Никита Лень

@Алексей_Стецко, чтобы сортировать по двум полям в разном порядке, можно использовать конструкцию ORDER BY поле1 DESC, поле2 ASC.

ASC -- сортировка по возрастанию, DESC -- по убыванию.

Алексей Стецко

@Никита_Лень, это-то понятно, у меня вопрос, зачем сортировать два поля, а не одно?

Никита Лень

@Алексей_Стецко, ну такое задание, иногда это удобно.

Галина Озерова

@Алексей_Стецко, Сортировка работает следующим образом: сначала сортируются данные по первому столбцу после ORDER BY, потом сортируются по второму столбцу данные, которые имеют ОДИНАКОВЫЕ

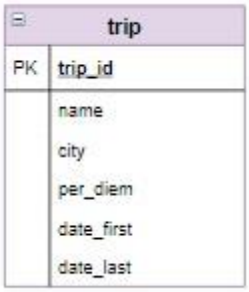
значения в первом. Если же в первом столбце все данные разные - то сортировка по второму столбцу ни на что не влияет.
Например, из задания:
|Лебедев Т.К. | Томск | 12 |
| Абрамова К.А. | Владивосток | 12 |
Сначала произведена сортировка по длительности, оказалось, что две командировки имеют одинаковую длительность. Для этих строк осуществляется сортировка в обратном алфавитном порядке по городам, поэтому запись с городом Томск в таблице расположена перед записью с Владивостоком.
Аналогично сортируются записи с длительностью командировки 7, на остальные строки сортировка по второму столбцу не влияет

Шаг 7: Задание (Вывести самые короткие командировки)

Ссылка в интернете

Вывести информацию о командировках сотрудника(ов), которые были самыми короткими по времени.

Структура таблицы:



Sql запрос, создающий таблицу trip :

```
CREATE TABLE trip (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(30),
    city VARCHAR(25),
    per_diem DECIMAL (8,2),
    date_first DATE,
    date_last DATE
);
```

Результат:

```
+--------------+-----------------+------------+------------+
| name         | city            | date_first | date_last  |
+--------------+-----------------+------------+------------+
| Семенов И.В. | Санкт-Петербург | 2020-06-01 | 2020-06-03 |
+--------------+-----------------+------------+------------+
```

Пояснение. Используйте вложенный запрос, чтобы найти длительность самой короткой командировки.

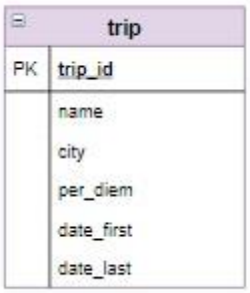
Шаг 8: Функция **MONTH(дата)**

Ссылка в интернете

Задание (Вывести командировки, начало и конец в одном месяце)

Вывести информацию о командировках, начало и конец которых относятся к одному месяцу. Результат отсортировать сначала в алфавитном порядке по названию города, а затем по фамилии сотрудника.

Структура таблицы: **Sql запрос, создающий таблицу trip :**




```
CREATE TABLE trip (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(30),
    city VARCHAR(25),
    per_diem DECIMAL (8,2),
    date_first DATE,
    date_last DATE
);
```

Результат:

```
+--------------+-----------------+------------+------------+
| name         | city            | date_first | date_last  |
+--------------+-----------------+------------+------------+
| Абрамова К.А. | Владивосток     | 2020-01-14 | 2020-01-27 |
| Абрамова К.А. | Владивосток     | 2020-07-02 | 2020-07-13 |
| Баранов П.Е.  | Воронеж         | 2020-07-19 | 2020-07-25 |
| Абрамова К.А. | Москва          | 2020-04-06 | 2020-04-14 |
| Баранов П.Е.  | Москва          | 2020-01-12 | 2020-01-17 |
| Баранов П.Е.  | Москва          | 2020-02-14 | 2020-02-22 |
| Колесов С.П.  | Москва          | 2020-02-01 | 2020-02-06 |
| Лебедев Т.К.  | Москва          | 2020-03-03 | 2020-03-06 |
| Семенов И.В.  | Москва          | 2020-01-23 | 2020-01-31 |
| Колесов С.П.  | Новосибирск     | 2020-06-03 | 2020-06-12 |
| Семенов И.В.  | Санкт-Петербург | 2020-06-01 | 2020-06-03 |
| Лебедев Т.К.  | Томск           | 2020-05-20 | 2020-05-31 |
| Федорова А.Ю. | Томск           | 2020-06-20 | 2020-06-26 |
+--------------+-----------------+------------+------------+
```

Пояснение. Для того, чтобы выделить номер месяца из даты используется функция **MONTH(дата)**. Например, **MONTH('2020-04-12')=4**.

Шаг 9: Функция **MONTHNAME(дата)**

Ссылка в интернете

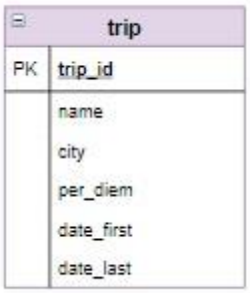
Задание (Вывести название месяца и количество месяцев)

Вывести название месяца и количество командировок для каждого месяца. Считаем, что командировка относится к некоторому месяцу, если она началась в этом месяце. Информацию вывести сначала в отсортированном по убыванию количества, а потом в алфавитном порядке по названию месяца виде. Название столбцов – **Месяц** и **Количество**.

Структура таблицы:

Sql запрос, создающий таблицу trip:

```
CREATE TABLE trip (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(30),
    city VARCHAR(25),
    per_diem DECIMAL (8,2),
    date_first DATE,
```




```
    date_last DATE
);
```

Результат:

```
+----------+------------+
| Месяц    | Количество |
+----------+------------+
| February | 4          |
| January  | 4          |
| June     | 3          |
| May      | 3          |
| April    | 2          |
| July     | 2          |
| March    | 2          |
+----------+------------+
```

Пояснение

1. Для того, чтобы выделить название месяца из даты используется функция **MONTHNAME(дата)**, которая возвращает название месяца на английском языке для указанной даты. Например, **MONTHNAME('2020-04-12')='April'**.
2. Если группировка осуществляется по вычисляемому столбцу (в данном случае «вычисляется» название месяца), то после GROUP BY можно указать как вычисляемое выражение, так и имя столбца, заданное с помощью AS.

Комментарии учащихся:

1.

Александр Цимбулов

Осталась не понятной одна вещь по порядку выполнения операций. Из пункта 1.3 курса читаю:
Сначала определяется таблица из которой выбираются данные (FROM), затем из этой таблицы отбираются записи в соответствии с условием WHERE, выбранные данные агрегируются (GROUP BY), из агрегированных записей выбираются те, которые удовлетворяют условию после HAVING. Потом формируются данные результирующей выборки как это указано после SELECT (вычисляются выражения, присваиваются имена и пр.). Результирующая выборка сортируется как указано после ORDER BY.
Если это верно, то тогда каким образом GROUP BY понимает имя, заданное в SELECT. По идее во время группировки имя столбца еще не должно быть вычеслено.

Галина Озерова

@Александр_Цимбулов, Совершенно верно, в соответствии со стандартом SQL имя из SELECT в GROUP BY не должно быть доступно. Но некоторые интерпретаторы SQL (включая эту платформу) порядок обработки запроса нарушают.
Я считаю, что нужно придерживаться стандарта, а не использовать особенности интерпретатора...

Александр Цимбулов

@Галина_Озерова,
Из пункта 2 пояснений к этому заданию:
"Если группировка осуществляется по вычисляемому столбцу (в данном случае «вычисляется» название месяца), то после GROUP BYможно указать как вычисляемое выражение, так и имя столбца, заданное с помощью AS."

Желательно указать в пункте 2 пояснений, хотя бы то, что указанное поведение нарушает стандарт, но иногда может встречаться на реальных платформах.
Станислав Гришин
@Александр_Цимбулов, Прекрасное замечание, полностью поддерживаю! А то тем людям, которые только начинают свой путь в SQL и принимают информацию за чистую монету и пытаются создать у себя в голове некую структуру и правила, очень тяжело, когда такие вещи не поясняются, а просто как "ну, бывает и такое".
Сергей Т
@Галина_Озерова, а как же урок 1.2.3? Или я что-то не так понял?
Для того чтобы отобрать данные из определенных столбцов таблицы используется SQL запрос следующей структуры:

- ключевое слово SELECT;
- список столбцов таблицы через запятую;
- ключевое слово FROM;
- имя таблицы.

Результатом является таблица, в которую включены все данные из указанных после SELECT столбцов исходной таблицы.
Запрос:

```
SELECT title, amount FROM book;
```

Андрей Голышев
@Александр_Цимбулов, правильно ли я понимаю, что в соответствие со стандартом SQL необходимо писать GROUP BY MONTHNAME(date_first) вместо GROUP BY Месяц, чтобы не зависеть от платформы, на которой выполняется запрос?
Галина Озерова
@Станислав_Гришин, **@Александр_Цимбулов** Я с вами согласна, добавила Ваше пояснение. Только сейчас заметила, что пропустила комметрии эти. Спасибо!
2.
Олег Макаров
Хотелось бы узнать как можно вывести названия месяцев по-русски?
Галина Озерова
@Олег_Макаров, Перед запросом добавить

```
SET @@lc_time_names = 'ru_UA';
```

Шаг 10: Функции день(**DAY()**), месяц (**MONTH()**), год(**YEAR()**)

Ссылка в интернете

Задание (Вывести сумму суточных)

Вывести сумму суточных (произведение количества дней командировки и размера суточных) для командировок, первый день которых пришелся на февраль или март 2020 года. Информацию отсортировать сначала в алфавитном порядке по фамилиям сотрудников, а затем по убыванию суммы суточных.

Структура таблицы:	**Sql запрос, создающий таблицу trip :**
	`CREATE TABLE trip (`
	`  trip_id INT PRIMARY KEY AUTO_INCREMENT,`
	`  name VARCHAR(30),`
	`  city VARCHAR(25),`
	`  per_diem DECIMAL (8,2),`
	`  date_first DATE,`
	`  date_last DATE`

);

Результат:

```
+---------------+-----------------+------------+---------+
| name          | city            | date_first | Сумма   |
+---------------+-----------------+------------+---------+
| Абрамова К.А. | Москва          | 2020-02-23 | 5600.00 |
| Баранов П.Е.  | Москва          | 2020-02-14 | 6300.00 |
| Колесов С.П.  | Новосибирск     | 2020-02-27 | 6750.00 |
| Колесов С.П.  | Москва          | 2020-02-01 | 4200.00 |
| Лебедев Т.К.  | Москва          | 2020-03-03 | 2800.00 |
| Семенов И.В.  | Санкт-Петербург | 2020-03-29 | 5600.00 |
+---------------+-----------------+------------+---------+
```

Пояснение. В SQL есть функции, которые позволяют выделить часть даты: день(**DAY()**), месяц (**MONTH()**), год(**YEAR()**) . Например:

DAY('2020-02-01') = 1

MONTH('2020-02-01') = 2

YEAR('2020-02-01') = 2020

Шаг 11: Задание (Вывести сотрудников, которые были более чем 3 раза в командировках)

Ссылка в интернете

Вывести фамилию с инициалами и общую сумму суточных, полученных за все командировки для тех сотрудников, которые были в командировках больше чем 3 раза, в отсортированном по убыванию сумм суточных виде. Только для этого задания изменена строка таблицы **trip** :

4	Ильиных Г.Р.	Владивосток	450	2020-01-12	2020-03-02

Структура таблицы:

Sql запрос, создающий таблицу trip :

```
CREATE TABLE trip (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(30),
    city VARCHAR(25),
    per_diem DECIMAL (8,2),
    date_first DATE,
```


```
    date_last DATE
);
```

Результат:

```
+---------------+----------+
| name          | Сумма    |
+---------------+----------+
| Абрамова К.А. | 29200.00 |
| Баранов П.Е.  | 21300.00 |
+---------------+----------+
```

УРОК 1.7: Таблица "Нарушения ПДД", запросы корректировки

Шаг 1: Содержание урока

Ссылка в интернете

В этом уроке на каждом шаге используется таблица, в которой представлена информация о начисленных водителям штрафах за нарушения правил дорожного движения (ПДД). С помощью запросов корректировки необходимо выполнить следующие действия:

- создать таблицу с информацией о штрафах ;
- заполнить ее;
- занести сумму штрафа за каждое новое нарушение ПДД;
- если водитель на определенной машине совершил повторное нарушение, то сумму его штрафа за данное нарушение нужно увеличить в два раза (часть 1, часть 2) ;
- если водитель оплатил свой штраф в течение 20 дней со дня нарушения, то значение его штрафа уменьшить в два раза;
- создать новую таблицу, в которую включить информацию о всех неоплаченных штрафах;
- удалить информацию о нарушениях, совершенных раньше некоторой даты.

На четвертом шаге рассматривается временное именование таблиц (алиасы).

Структура и наполнение таблиц

В таблице **fine** представлена информация о начисленных водителям штрафах за нарушения правил дорожного движения (ПДД) (фамилия водителя, номер машины, описание нарушения, сумма штрафа, дата совершения нарушения и дата оплаты штрафа):

fine_id	name	number_plate	violation	sum_fine	date_violation	date_payment
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(30)	VARCHAR(6)	VARCHAR(50)	DECIMAL(8, 2)	DATE	DATE
1	Баранов П.Е.	P523BT	Превышение скорости (от 40 до 60)	500.00	2020-01-12	2020-01-17
2	Абрамова К.А.	O111AB	Проезд на запрещающий сигнал	1000.00	2020-01-14	2020-02-27
3	Яковлев Г.Р.	T330TT	Превышение скорости (от 20 до 40)	500.00	2020-01-23	2020-02-23
4	Яковлев Г.Р.	M701AA	Превышение скорости (от 20 до 40)		2020-01-12	

5	Колесов С.П.	К892АХ	Превышение скорости (от 20 до 40)		2020-02-01	
6	Баранов П.Е.	Р523ВТ	Превышение скорости (от 40 до 60)		2020-02-14	
7	Абрамова К.А.	О111АВ	Проезд на запрещающий сигнал		2020-02-23	
8	Яковлев Г.Р.	Т330ТТ	Проезд на запрещающий сигнал		2020-03-03	

В таблицу **traffic_violation** занесены нарушения ПДД и соответствующие штрафы (в рублях):

violation_id	Violation	sum_fine
INT PRIMARY KEY AUTO_INCREMENT	VARCHAR(50)	DECIMAL(8,2)
1	Превышение скорости (от 20 до 40)	500.00
2	Превышение скорости (от 40 до 60)	1000.00
3	Проезд на запрещающий сигнал	1000.00

Шаг 2: Задание (Создать таблицу fine)

Ссылка в интернете

Создать таблицу **fine** следующей структуры:

Поле	Описание
fine_id	ключевой столбец целого типа с автоматическим увеличением значения ключа на 1
name	строка длиной 30
number_plate	строка длиной 6
violation	строка длиной 50
sum_fine	вещественное число, максимальная длина 8, количество знаков после запятой 2

date_violation	Дата
date_payment	Дата

Результат:

Affected rows: 0

Комментарии учащихся:

1.

Anonymous 36233429

Прошло решение с типом **DATETIME** и **DATE**. Что предпочтительнее использовать?

Лариса Фернандес

@Anonymous_36233429, как следует из названия, они используются для хранения разных данных - даты и даты со временем http://www.mysql.ru/docs/man/DATETIME.html

Мне в работе чаще всего приходится datetime приводить к date, причем какими-нибудь кривыми способами типа CreateDate(GetYear('дата-со-временем"), GetMonth('дата-со-временем"), GetDay('дата-со-временем")) Это T SQL

##

Если вам дали задание и нет возможности проверить, попробуйте соорудить небольшую таблицу, например, здесь https://sqliteonline.com/syntax/select/ и проверьте, правильно ли выдается результат.

вы можете написать собственную функцию и посчитать регрессию по прошлым данным https://docs.microsoft.com/ru-ru/sql/t-sql/statements/create-function-transact-sql?view=sql-server-ver15

Также можно использовать оконные функции для расчета регрессии http://www.silota.com/docs/recipes/sql-linear-regression.html

Но все-таки эффективнее и проще выгрузить данные из БД и посчитать все в питоне (могу помочь, наверное).

Шаг 3: Задание (Занести три записи в таблицу fine)

Ссылка в интернете

В таблицу **fine** первые 5 строк уже занесены. Добавить в таблицу записи с ключевыми значениями 6, 7, 8.

fine_id	Name	number_plate	violation	sum_fine	date_violation	date_payment
1	Баранов П.Е.	P523BT	Превышение скорости (от 40 до 60)	500.00	2020-01-12	2020-01-17
2	Абрамова К.А.	O111AB	Проезд на запрещающий сигнал	1000.00	2020-01-14	2020-02-27
3	Яковлев Г.Р.	T330TT	Превышение скорости (от 20 до 40)	500.00	2020-01-23	2020-02-23
4	Яковлев Г.Р.	M701AA	Превышение скорости (от 20 до 40)		2020-01-12	
5	Колесов	K892AX	Превышение		2020-02-01	

	С.П.		скорости (от 20 до 40)			
6	Баранов П.Е.	Р523ВТ	Превышение скорости (от 40 до 60)		2020-02-14	
7	Абрамова К.А.	О111АВ	Проезд на запрещающий сигнал		2020-02-23	
8	Яковлев Г.Р.	Т330ТТ	Проезд на запрещающий сигнал		2020-03-03	

Пояснение.

1. Между формулировкой нарушения и открывающей скобкой ПРОБЕЛА НЕТ. Например, *Превышение скорости(от 40 до 60)*.
2. Для занесения пустых значений в поля используется оператор `Null`.

Результат (для сокращения записи ключевой столбец опущен):

```
Affected rows: 1

Affected rows: 1

Affected rows: 1

Query result:

+-------------------+-------------+-----------------------------------------------+----------+-----------------+-------------------+
| name              | n_plate     | violation                                     | s_fine   | date_violation| date_payment |
+-------------------+-------------+-----------------------------------------------+----------+-----------------+-------------------+
| Баранов П.Е.      | Р523ВТ      | Превышение скорости(от 40 до 60)| 500.00   | 2020-01-12    | 2020-01-17      |
| Абрамова К.А.| О111АВ | Проезд на запрещающий сигнал      | 1000.00| 2020-01-14    | 2020-02-27      |
| Яковлев Г.Р.      | Т330ТТ      | Превышение скорости(от 20 до 40)| 500.00   | 2020-01-23    | 2020-02-23      |
| Яковлев Г.Р.      | М701АА      | Превышение скорости(от 20 до 40)| None     | 2020-01-12    | None            |
| Колесов С.П.      | К892АХ      | Превышение скорости(от 20 до 40)| None     | 2020-02-01    | None            |
| Баранов П.Е.      | Р523ВТ      | Превышение скорости(от 40 до 60)| None     | 2020-02-14    | None            |
| Абрамова К.А.| О111АВ | Проезд на запрещающий сигнал      | None     | 2020-02-23    | None            |
| Яковлев Г.Р.      | Т330ТТ      | Проезд на запрещающий сигнал      | None     | 2020-03-03    | None            |
+-------------------+-------------+-----------------------------------------------+----------+-----------------+-------------------+
```

Шаг 4: Использование временного имени таблицы (алиаса)

Ссылка в интернете

Теоретический материал для этого шага подготовлен Михаилом Захаровым*. Большое ему спасибо!*

Чтобы не писать название таблицы каждый раз, удобно использовать алиасы. Алиас - это временное имя таблицы, которое можно задать самостоятельно, псевдоним. Распространяется алиас только на текущий запрос. Чтобы его объявить, достаточно написать нужный псевдоним сразу после названия таблицы:

```
UPDATE fine f, traffic_violation tv
```

или использовать ключевое слово AS:

```
UPDATE fine AS f, traffic_violation AS tv
```

Здесь **f** будет псевдонимом таблицы **fine**, а **tv** - псевдонимом таблицы **traffic_violation** (как правило, в качестве алиаса выбирают аббревиатуру названия таблицы).

В дальнейшем объявленные псевдонимы используют вместо полных названий таблиц в запросе:

```
WHERE f.violation = tv.violation
```

На небольших запросах использование псевдонимов не приносит существенной пользы, но вот при увеличении числа используемых таблиц (а их иногда может быть и 5 и 10 и более) псевдонимы помогают сделать запрос чище и читабельнее.

Пример

Для тех, кто уже оплатил штраф, вывести информацию о том, изменялась ли стандартная сумма штрафа.

Запрос:

```
SELECT  name, number_plate, f.violation,
      if(f.sum_fine = tv.sum_fine, "Стандартная сумма штрафа",
        if(f.sum_fine < tv.sum_fine, "Уменьшенная сумма штрафа",
          "Увеличенная сумма штрафа")) AS description
FROM  fine f, traffic_violation tv
WHERE tv.violation = f.violation and f.sum_fine IS NOT Null;
```

Результат:

```
+---------------+--------------+----------------------+--------------------------+
| name          | number_plate | violation            | description              |
+---------------+--------------+----------------------+--------------------------+
| Баранов П.Е.  | P523BT       | Прев …(от 40 до 60)  | Уменьшенная сумма штрафа |
| Абрамова К.А. | O111AB       | Проезд …сигнал       | Стандартная сумма штрафа |
| Яковлев Г.Р.  | T330TT       | Прев …(от 20 до 40)  | Стандартная сумма штрафа |
+---------------+--------------+----------------------+--------------------------+
```

Задание (Занести в таблицу штрафы, которые водитель должен оплатить)

Сформированная на предыдущих шагах таблица **fine** имеет вид (без ключевого столбца):

```
+-----------------+-----------+-----------------------------------------------+---------+----------------+-----------------+
| name            | n_plate   | violation                                     | s_fine  | date_violation | date_payment    |
+-----------------+-----------+-----------------------------------------------+---------+----------------+-----------------+
| Баранов П.Е.    | P523BT    | Превышение скорости(от 40 до 60)              | 500.00  | 2020-01-12     | 2020-01-17      |
| Абрамова К.А.   | O111AB    | Проезд на запрещающий сигнал                  | 1000.00 | 2020-01-14     | 2020-02-27      |
| Яковлев Г.Р.    | T330TT    | Превышение скорости(от 20 до 40)              | 500.00  | 2020-01-23     | 2020-02-23      |
| Яковлев Г.Р.    | M701AA    | Превышение скорости(от 20 до 40)              | None    | 2020-01-12     | None            |
| Колесов С.П.    | K892AX    | Превышение скорости(от 20 до 40)              | None    | 2020-02-01     | None            |
| Баранов П.Е.    | P523BT    | Превышение скорости(от 40 до 60)              | None    | 2020-02-14     | None            |
| Абрамова К.А.   | O111AB    | Проезд на запрещающий сигнал                  | None    | 2020-02-23     | None            |
| Яковлев Г.Р.    | T330TT    | Проезд на запрещающий сигнал                  | None    | 2020-03-03     | None            |
+-----------------+-----------+-----------------------------------------------+---------+----------------+-----------------+
```

Занести в таблицу **fine** суммы штрафов, которые должен оплатить водитель, в соответствии с данными из таблицы **traffic_violation**. При этом суммы заносить только в пустые поля столбца **sum_fine**.

Таблица **traffic_violation** создана и заполнена.

*Результат (*для сокращения записи ключевой столбец не показан*):*

Affected rows: 5

Query result:

name	n_plate	violation	s_fine	date_violation	date_payment
Баранов П.Е.	Р523ВТ	Превышение скорости(от 40 до 60)	500.00	2020-01-12	2020-01-17
Абрамова К.А.	О111АВ	Проезд на запрещающий сигнал	1000.00	2020-01-14	2020-02-27
Яковлев Г.Р.	Т330ТТ	Превышение скорости(от 20 до 40)	500.00	2020-01-23	2020-02-23
Яковлев Г.Р.	М701АА	Превышение скорости(от 20 до 40)	500.00	2020-01-12	None
Колесов С.П.	К892АХ	Превышение скорости(от 20 до 40)	500.00	2020-02-01	None
Баранов П.Е.	Р523ВТ	Превышение скорости(от 40 до 60)	1000.00	2020-02-14	None
Абрамова К.А.	О111АВ	Проезд на запрещающий сигнал	1000.00	2020-02-23	None
Яковлев Г.Р.	Т330ТТ	Проезд на запрещающий сигнал	1000.00	2020-03-03	None

Пояснение.

1. После ключевого слова UPDATE кроме обновляемой таблицы **fine** укажите таблицу **traffic_violation,** для того чтобы запрос видел таблицы источники. Сначала перечисляем **все источники,** потом выполняем необходимые действия.

2. Обновляйте только те записи таблицы **fine**, у которых значение столбца **violation** совпадает со значением соответствующего столбца таблицы **traffic_violation**, а также значение столбца **sum_fine** пусто.

3. Сравнение значения столбца с пустым значением осуществляется с помощью оператора **IS Null**.

Шаг 5: Задание (Вывести информацию о водителях, которые совершили более двух нарушений)

Ссылка в интернете

Вывести фамилию, номер машины и нарушение только для тех водителей, которые совершили повторное нарушение на одной машине два и более раз.

Пояснение к заданию

Скорректированная на предыдущих шагах таблица **fine** имеет вид (без ключевого столбца):

name	n_plate	violation	s_fine	date_violation	date_payment
Баранов П.Е.	Р523ВТ	Превышение скорости(от 40 до 60)	500.00	2020-01-12	2020-01-17
Абрамова К.А.	О111АВ	Проезд на запрещающий сигнал	1000.00	2020-01-14	2020-02-27

```
| Яковлев Г.Р.  | Т330ТТ   | Превышение скорости(от 20 до 40)| 500.00   | 2020-01-23   | 2020-02-23     |
| Яковлев Г.Р.  | М701АА  | Превышение скорости(от 20 до 40)| 500.00   | 2020-01-12   | None           |
| Колесов С.П.  | К892АХ  | Превышение скорости(от 20 до 40)| 500.00  | 2020-02-01   | None           |
| Баранов П.Е.  | Р523ВТ  | Превышение скорости(от 40 до 60)| 1000.00| 2020-02-14   | None           |
| Абрамова К.А.| О111АВ | Проезд на запрещающий сигнал     |1000.00 | 2020-02-23   | None           |
| Яковлев Г.Р.   | Т330ТТ  | Проезд на запрещающий сигнал     |1000.00 | 2020-03-03   | None           |
+------------------+------------+-----------------------------------------------+---------+----------------+------------------+
```

Под увеличение штрафа в два раза подходит водитель «Абрамова К.А.», который на машине с государственным номером «О111АВ» совершил повторное нарушение «Проезд на запрещающий сигнал», а также водитель «Баранов П.Е.», который на машине с номером «Р523ВТ» дважды совершил нарушение «Превышение скорости (от 40 до 60)».

Результат:

```
+---------------+--------------+-----------------------------------+
| name          | number_plate | violation                         |
+---------------+--------------+-----------------------------------+
| Абрамова К.А. | О111АВ       | Проезд на запрещающий сигнал      |
| Баранов П.Е.  | Р523ВТ       | Превышение скорости(от 40 до 60)  |
+---------------+--------------+-----------------------------------+
```

Шаг 6: Задание (Увеличить в два раза сумму неоплаченных штрафов)

Ссылка в интернете

Увеличить в два раза сумму неоплаченных штрафов для отобранных на предыдущем шаге записей.

Пояснение к заданию

Для всех нарушений, по которым штраф еще не оплачен, (тех, у которых **date_payment** имеет пустое значение **Null**), необходимо проверить, является ли данное нарушение для водителя и машины повторным, если да – увеличить штраф в два раза. Если водитель совершил нарушение на другой машине, ему увеличивать штраф не нужно. Этот запрос реализован на предыдущем шаге.

При реализации можно использовать вложенный запрос как отдельную таблицу, записанную после ключевого слова UPDATE, при этом вложенному запросу необходимо присвоить имя, например, **query_in**:

```
UPDATE fine, (SELECT ...) query_in
SET ...
WHERE ...
```

Другим способом решения является использование двух запросов: сначала создать временную таблицу, например, **query_in**, в которую включить информацию о тех штрафах, сумму которых нужно увеличить в два раза, а затем уже обновлять информацию в таблице **fine**:

```
CREATE TABLE query_in ...;
UPDATE fine, query_in
SET ...
```

WHERE ...;

После ключевого слова WHERE указывается условие, при котором нужно обновлять данные. В нашем случае данные обновляются, если и фамилия, и государственный номер, и нарушение совпадают в таблице **fine** и в результирующей таблице запроса **query_in**. Например, для связи по фамилии используется запись **fine.name = query_in.name**. Также в условии нужно учесть, что данные обновляются только для тех записей, у которых в столбце **date_payment** пусто.

Результат:

Affected rows: 2

Query result:

```
+-----------------+-----------+-----------------------------------------+---------+----------------+-----------------+
| name            | n_plate   | violation                               | s_fine  | date_violation | date_payment    |
+-----------------+-----------+-----------------------------------------+---------+----------------+-----------------+
| Баранов П.Е.    | P523BT    | Превышение скорости(от 40 до 60)        | 500.00  | 2020-01-12     | 2020-01-17      |
| Абрамова К.А.   | O111AB    | Проезд на запрещающий сигнал           | 1000.00 | 2020-01-14     | 2020-02-27      |
| Яковлев Г.Р.    | T330TT    | Превышение скорости(от 20 до 40)        | 500.00  | 2020-01-23     | 2020-02-23      |
| Яковлев Г.Р.    | M701AA    | Превышение скорости(от 20 до 40)        | 500.00  | 2020-01-12     | None            |
| Колесов С.П.    | K892AX    | Превышение скорости(от 20 до 40)        | 500.00  | 2020-02-01     | None            |
| Баранов П.Е.    | P523BT    | Превышение скорости(от 40 до 60)        | 2000.00 | 2020-02-14     | None            |
| Абрамова К.А.   | O111AB    | Проезд на запрещающий сигнал           | 2000.00 | 2020-02-23     | None            |
| Яковлев Г.Р.    | T330TT    | Проезд на запрещающий сигнал           | 1000.00 | 2020-03-03     | None            |
+-----------------+-----------+-----------------------------------------+---------+----------------+-----------------+
```

Важно. Если в запросе используется несколько таблиц или запросов, включающих одинаковые поля, то применяется полное имя столбца, включающего название таблицы через символ «.». Например, **fine.name** и **query_in.name**.

Шаг 7: Задание (Занести дату оплаты соответствующего штрафа)

Ссылка в интернете

Водители оплачивают свои штрафы. В таблице **payment** занесены даты их оплаты:

payment_id	name	number_plate	violation	date_violation	date_payment
1	Яковлев Г.Р.	M701AA	Превышение скорости (от 20 до 40)	2020-01-12	2020-01-22
2	Баранов П.Е.	P523BT	Превышение скорости (от 40 до 60)	2020-02-14	2020-03-15
3	Яковлев Г.Р.	T330TT	Проезд на запрещающий сигнал	2020-03-03	2020-03-21

Необходимо в таблицу **fine** занести дату оплаты соответствующего штрафа из таблицы **payment** и уменьшить начисленный штраф в таблице **fine** в два раза (только для новых штрафов, дата оплаты которых занесена в **payment**), если оплата произведена не более, чем за 20 дней со дня нарушения.

Результат:

Affected rows: 3

Query result:

name	n_plate	violation	s_fine	date_violation	date_payment
Баранов П.Е.	P523BT	Превышение скорости(от 40 до 60)	500.00	2020-01-12	2020-01-17
Абрамова К.А.	O111AB	Проезд на запрещающий сигнал	1000.00	2020-01-14	2020-02-27
Яковлев Г.Р.	T330TT	Превышение скорости(от 20 до 40)	500.00	2020-01-23	2020-02-23
Яковлев Г.Р.	M701AA	Превышение скорости(от 20 до 40)	250.00	2020-01-12	2020-01-22
Колесов С.П.	K892AX	Превышение скорости(от 20 до 40)	500.00	2020-02-01	None
Баранов П.Е.	P523BT	Превышение скорости(от 40 до 60)	2000.00	2020-02-14	2020-03-15
Абрамова К.А.	O111AB	Проезд на запрещающий сигнал	2000.00	2020-02-23	None
Яковлев Г.Р.	T330TT	Проезд на запрещающий сигнал	500.00	2020-03-03	2020-03-21

Пояснение. Для уменьшения суммы штрафа в два раза в зависимости от условия можно либо в SET использовать функцию **if()**, либо реализовать 2 запроса с разными условиями после WHERE.

Шаг 8: Задание (Создать новую таблицу back_payment)

Ссылка в интернете

Создать новую таблицу **back_payment**, куда внести информацию о неоплаченных штрафах (Фамилию и инициалы водителя, номер машины, нарушение, сумму штрафа и дату нарушения) из таблицы **fine**.

Результат:

Affected rows: 2

Query result:

name	n_plate	violation	sum_fine	date_violation
Колесов С.П.	K892AX	Пре…(от 20 до 40)	500.00	2020-02-01
Абрамова К.А.	O111AB	Проезд …	2000.00	2020-02-23

Пояснение. Для неоплаченных штрафов столбец **date_payment** имеет пустое значение.

Важно. На этом шаге необходимо создать таблицу на основе запроса! Не нужно одним запросом создавать таблицу, а вторым в нее добавлять строки.

Шаг 9: Задание (Удалить информацию о нарушениях)

Ссылка в интернете

Удалить из таблицы **fine** информацию о нарушениях, совершенных раньше 1 февраля 2020 года.

Результат:

Affected rows: 4

Query result (это выборка из таблицы fine после удаления записей):

name	n_plate	violation	s_fine	date_violation	date_payment
Колесов С.П.	К892АХ	Превышение скорости(от 20 до 40)	500.00	2020-02-01	None
Баранов П.Е.	Р523ВТ	Превышение скорости(от 40 до 60)	2000.00	2020-02-14	2020-03-15
Абрамова К.А.	О111АВ	Проезд на запрещающий сигнал	2000.00	2020-02-23	None
Яковлев Г.Р.	Т330ТТ	Проезд на запрещающий сигнал	500.00	2020-03-03	2020-03-21

МОДУЛЬ 2: Запросы SQL к связанным таблицам

—

В модуле рассматриваются связи между таблицами реляционной базы данных, а также различные виды запросов, построенных на связанных таблицах

УРОК 2.1: Связи между таблицами

Шаг 1: Содержание урока

Ссылка в интернете

Средствами SQL запросов можно выбирать и обрабатывать данные не только из одной таблицы, но из нескольких связанных таблиц. В данном уроке мы рассмотрим способы соединения таблиц:

- связь между таблицами «один ко многим»;
- связь между таблицами «многие ко многим»;
- создание таблицы с внешними ключами;
- действия при удалении записи главной таблицы;
- заполнение таблицы с внешними ключами;
- добавление данных в таблицу с внешними ключами.

Комментарии учащихся:

1.
Елена Черникова
https://habr.com/ru/post/488054/ Замечательное объяснение связей между таблицами, может быть, кому-то будет полезно!

Шаг 2: Связь «один ко многим»

Ссылка в интернете

Рассмотрим таблицу **book** (в ней столбец **author** переименован в **name_author**):

book_id	title	name_author	price	Amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2
5	Стихотворения и поэмы	Есенин С.А.	650.00	15

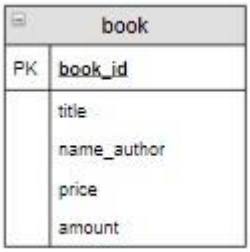
В этой таблице фамилии авторов повторяются для нескольких книг. А что, если придется вместо инициалов для каждого автора хранить его полное имя и отчество? Тогда, если в таблице содержится информация о 50 книгах Достоевского, придется 50 раз исправлять «Ф.М.» на «Федор Михайлович». При этом, если в некоторых записях использовать «Фёдор Михайлович» (с буквой ё), то мы вообще получим двух разных авторов...

Чтобы устранить эту проблему в реляционных базах данных создается новая таблица **author**, в которой перечисляются все различные авторы, а затем эта таблица связывается с таблицей **book**. При этом такая связь называется «**один ко многим**», таблица **author** называется главной, таблица **book** – связанной или подчиненной.

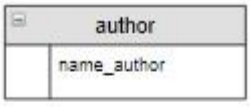
Связь «**один ко многим**» имеет место, когда одной записи главной таблицы соответствует несколько записей связанной таблицы, а каждой записи связанной таблицы соответствует только одна запись главной таблицы. Обозначается это так:

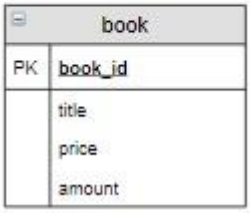
Этапы реализации связи «один ко многим»

Исходная таблица:

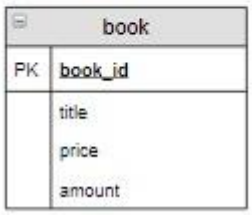


1. Создать таблицу **author**, в которую включить всех различных авторов из таблицы **book** (а затем удалить столбец с фамилиями авторов из таблицы **book**):

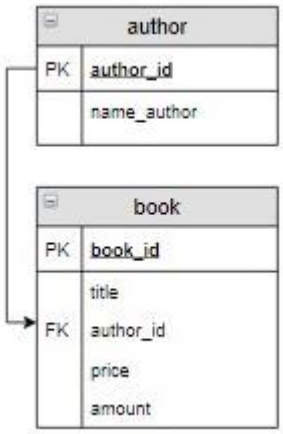




2. Обе таблицы должны содержать первичный ключ, в таблице **book** он уже есть, в таблицу **author** добавим ключ **author_id**:



3. Включить в таблицу **book** связанный столбец (внешний ключ, **FOREIGN KEY**), соответствующий по имени и типу ключевому столбцу главной таблицы (в нашем случае это столбец **author_id**). Для наглядности связь на схеме обозначается стрелкой от ключевого столбца главной таблицы к внешнему ключу связной таблицы:



Задание (Добавить новую характеристику книги)

Добавить новую характеристику книги – ее жанр, если считать, что каждая книга относится к одному жанру, то есть между ними определена связь «**один ко многим**». Расположите в правильном порядке этапы связывания таблицы с жанрами (**genre**) и таблицы **book**.

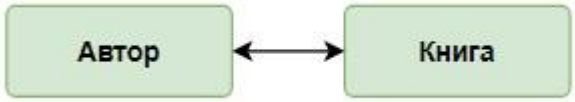
Шаг 3: Связь «многие ко многим»

Ссылка в интернете

На предыдущем шаге мы реализовали связь «**один ко многим**» для книг и авторов. Она означает, что каждый автор написал несколько книг, но каждую книгу написал только один автор. На самом деле, это не совсем верное утверждение. Например, книга «12 стульев» написана двумя авторами Ильфом И.А. и Петровым Е.П. С другой стороны, эти авторы написали и другие книги, например «Золотой теленок».

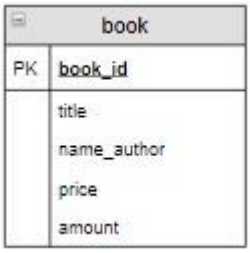
Для соединения таких таблиц используется связь «**многие ко многим**».

Связь «**многие ко многим**» имеет место когда каждой записи одной таблицы соответствует несколько записей во второй, и наоборот, каждой записи второй таблицы соответствует несколько записей в первой. Обозначается это так:

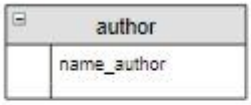


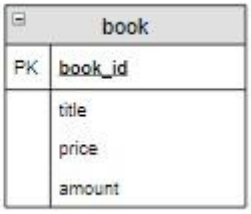
Этапы реализации связи «многие ко многим»

Исходная таблица:

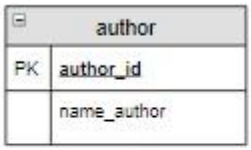


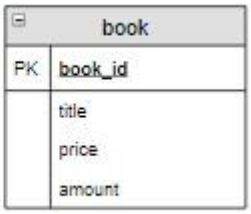
1. Создать таблицу **author**, в которую включить всех различных авторов из таблицы **book** (а затем удалить столбец с фамилиями авторов из таблицы **book**):



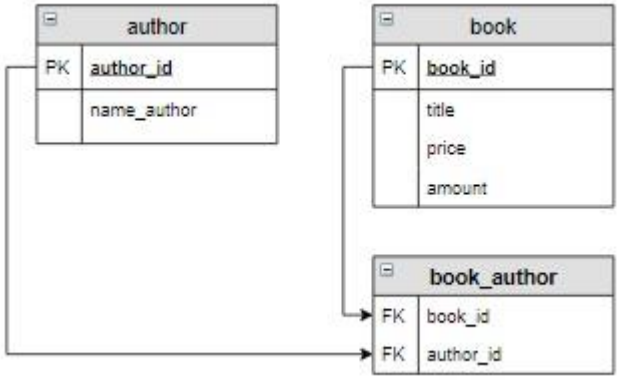


2. В обеих таблицах необходимо определить первичный ключ, в нашем случае в таблице **book** он уже есть, поэтому достаточно включить первичный ключ **author_id** в таблицу **author**:

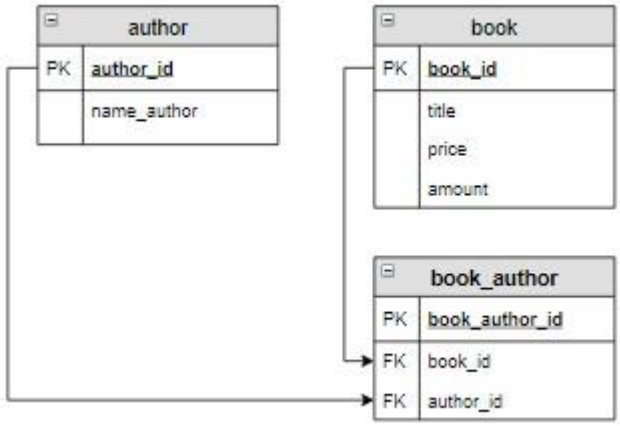




3. Создать новую таблицу-связку, состоящую из двух столбцов, соответствующих по имени и типу ключевым столбцам исходных таблиц. Каждый из этих столбцов является внешним ключом (**FOREIGN KEY**) и связан с ключевым столбцом каждой таблицы. Для наглядности связи на схеме обозначаются стрелкой от ключевого столбца исходной таблицы к внешнему ключу связной таблицы.



4. Дальше необходимо определиться с первичным ключом таблицы-связки. Можно сделать два ключевых столбца, тогда все записи в этой таблице должны быть уникальными, то есть не повторяться. Для связи автор-книга этот вариант подходит. Но в некоторых случаях записи в таблице-связке могут повторяться, например, если мы будем продавать книги покупателям (один человек может купить несколько книг, а одну и ту же книгу могут купить несколько человек). Тогда в таблицу-связку включают дополнительные столбцы для идентификации записей, например, дату продажи, также в таблицу-связку добавляют первичный ключ. Мы воспользуемся вторым способом:



Задание (Добавить новую характеристику к книге)

Добавить новую характеристику книги – ее жанр, если считать, что каждая книга может относиться к нескольким жанрам, а каждый жанр включает несколько книг, то есть между ними определена связь «**многие ко многим**».

Расположите в правильном порядке этапы связывания таблиц **genre** и **book**.

Шаг 4: Задание (Выберите тип связи)

Ссылка в интернете

Выберите тип связи, который подходит для описания пар информационных объектов.

Комментарии учащихся:

1.

Станислав Гришин

Если честно, пока вообще сложно понять, как это реализовывается. Схемки - это, конечно, хорошо, но все же, привели бы хоть один пример что за связи, как реализуются, что за ключи. И еще вопрос (тут его уже задавали, но почему-то не ответили на него, касательно случая про студента и книгу: " в конкретный момент времени конкретная книга только у одного студента, а у студента одновременно может быть несколько книг. Разве это не один к многим?"

Никита Лень

@Станислав_Гришин, нет, это многие ко многим, потому что эта конкретная книга через некоторое время может быть у другого студента. Про схемы, связи и ключи Вы можете поискать информацию в интернете, данный курс является тренажером именно по SQL. Возможно, в будущем будет добавлено больше теории баз данных.

Лариса Фернандес

@Станислав_Гришин, по поводу правильного создания ключей и связей в БД советую посмотреть курс "Погружение в СУБД" здесь же, на Степике

Станислав Гришин

@Никита_Лень, То есть получается, чтобы понять тип связи, нужно учитывать вообще все чисто гипотетические варианты (например "если я запрошу несколько раз в разное время кто взял эту книгу, могу ли я получить разные ответы?" и если ответ "Да, ответы могут быть разные", то значит связь многие-ко-многим) ?

Никита Лень

@Станислав_Гришин, да, составление ER-модели (схемы данных) должно основываться на описании конкретной предметной области, чем адекватнее Вы опишите предметную область, тем больше Ваша модель будет подходить для тех бизнес-процессов, которые должны быть автоматизированы с помощью СУБД (как пример).

Станислав Гришин

@Никита_Лень, если честно, из Вашего ответа я понял только "да" (на данном этапе мне, в принципе, достаточно). Но это ладно, я и так понимал, что передо мной еще настолько долгий и тернистый путь и столько неизвестного, что я пытаюсь не расстраиваться, когда не понимаю языка профессионалов.

Дмитрий Паньшин

@Станислав_Гришин, и я не понимаю

В библиотеке студент может взять несколько книг, одну и ту же книгу могут взять несколько студентов **(в разное время)**

В разное время - значит, один ко многим (ノ°益°)ノ 彡┻━┻

Шаг 5: Задание (Выберите одну или несколько схем)

Ссылка в интернете

Дана таблица **trip**. Выберите одну или несколько схем, которые позволяют правильно представить информацию из этой таблицы в виде нескольких связанных таблиц.

trip_id	name	city	per_diem	date_first	date_last
1	Баранов П.Е.	Москва	700	2020-01-12	2020-01-17
2	Абрамова К.А.	Владивосток	450	2020-01-14	2020-01-27
3	Семенов И.В.	Москва	700	2020-01-23	2020-01-31
4	Семенов И.В.	Владивосток	450	2020-02-12	2020-02-22

Пояснение. Для решения этой задачи необходимо:

1. Проанализировать информацию и выделить повторяющиеся данные в отдельные таблицы. Это будут Сотрудники и Города.

2. Определить тип связи между таблицами (многие ко многим или один ко многим), который позволит описать командировки сотрудников - выбрать верный вариант из предложенных.

3. Определить, как данная связь реализуется в реляционной модели - выбрать верный вариант.

Комментарии учащихся:

1.

Денис Герасименко

Давно не работал с БД, но когда работал связь многие-ко-многим использовались крайне редко. Их можно использовать только как временные, и нужно избегать подобных связей. Если я не прав - поправьте меня.

Галина Озерова

@Денис_Герасименко, В реляционной модели такая связь реализуется через вспомогательную таблицу, между которой и исходными таблицами связь один ко многим. При создании реляционных таблиц и связей между ними в базе данных она просто не может быть реализовано. Эту связь используют на этапе проектирования.

Шаг 6: Задание (Создать таблицу author)

Ссылка в интернете

Создать таблицу **author** следующей структуры:

Поле	Тип, описание
author_id	INT PRIMARY KEY AUTO_INCREMENT
name_author	VARCHAR(50)

Комментарии учащихся:

1.

Дмитрий Чернов

Пожалуйста, поправьте везде id int таблицы с PRIMARY KEY, что бы обязательно был UNSIGNED. Ну не может у ID быть отрицательного значения...

Галина Озерова

@Дмитрий_Чернов, Это потребует очень много изменить и в тексте, и в заданиях. Когда будем обновлять курс - добавим объяснение про это и изменим. А пока этот столбец вы нигде не заполняем, для него установлена автоматическая нумерация. она начинается с 0. Спасибо за замечание.

Александр Митрошин

@Галина_Озерова, Может - неможет, но, по-моему, не все СУБД поддерживают такой тип данных. В MS SQL Server, как минимум, до 2008 - точно не было типов данных UNSIGNED. UNSIGNED - не является стандартом SQL, это расширение языка добавленное некоторыми производителями СУБД, и, по мнению многих экспертов, бессмысленное. Скорее маркетинг, чем реальная необходимость.

Галина Озерова

@Александр_Митрошин, Я согласна с Вами.

Дмитрий Паньшин

@Александр_Митрошин, иногда лучше опираться на логику и здравый смысл, нежели на различные системы стандартизации. Зачем резервировать отрицательные значения, если они не используются?)

Шаг 7: Задание (Заполнить таблицу author)

Ссылка в интернете

Заполнить таблицу **author**. В нее включить следующих авторов:

- Булгаков М.А.
- Достоевский Ф.М.
- Есенин С.А.
- Пастернак Б.Л.

Результат:

Affected rows: 1

Affected rows: 1

Affected rows: 1

Affected rows: 1

Query result:

```
+-----------+------------------+
| author_id | name_author      |
+-----------+------------------+
| 1         | Булгаков М.А.    |
| 2         | Достоевский Ф.М. |
| 3         | Есенин С.А.      |
| 4         | Пастернак Б.Л.   |
+-----------+------------------+
```

Шаг 8: Создание таблицы с внешними ключами

Ссылка в интернете

При создании зависимой таблицы (таблицы, которая содержит внешние ключи) необходимо учитывать, что:

- каждый внешний ключ должен иметь тип данных **INT**;
- необходимо указать главную для нее таблицу и столбец, по которому осуществляется связь:

FOREIGN KEY (связанное_поле_зависимой_таблицы)

REFERENCES главная_таблица (связанное_поле_главной_таблицы)

Пример

Создать таблицу **book** следующей структуры:

Поле	**Тип, описание**	**Связи**
book_id	INT PRIMARY KEY AUTO_INCREMENT	
title	VARCHAR(50)	

author_id	INT	внешний ключ:главная таблица **author** ,связанный столбец **author.author_id**
price	DECIMAL(8, 2)	
amount	INT	

Запрос:

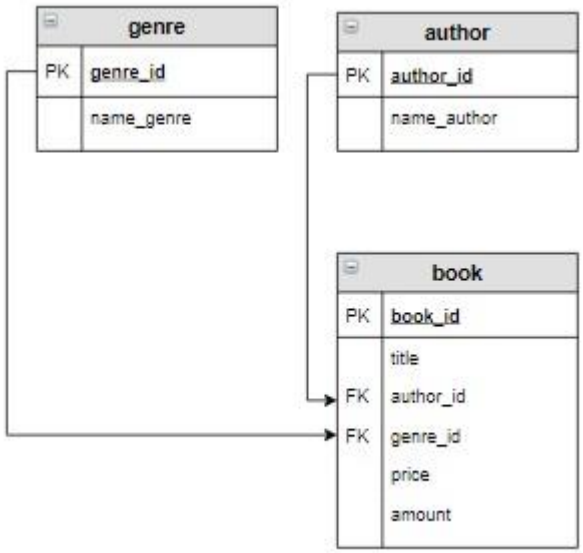
```
CREATE TABLE book (
    book_id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(50),
    author_id INT,
    price DECIMAL(8,2),
    amount INT,
    FOREIGN KEY (author_id)  REFERENCES author (author_id)
);
```

Задание (Дополнить запрос на создание таблицы book)

Дополнить запрос на создание таблицы **book** , чтобы ее структура соответствовала рисунку ниже (Связать таблицы **book** и **genre**) . В качестве главной таблицы для описания поля **genre_id** использовать таблицу **genre** следующей структуры:

Поле	**Тип, описание**
genre_id	INT PRIMARY KEY AUTO_INCREMENT
name_genre	VARCHAR(30)

Логическая схема (нужно создать только таблицу **book**):



Результат:

Affected rows: 0

Комментарии учащихся:

1.

Алексей Журавлёв

Странно что главных таблиц много, а зависимая одна.

Анастасия Давыденко

@Алексей_Журавлёв, эти главные таблицы - как справочники. Туда мы заносим просто максимально инфы по какому-то аспекту (например, кучу авторов) и оставляем, ведь новые авторы рождаются не так часто)
а в зависимую таблицу мы из таких справочников, как пазл, просто достаем информацию, в ней ее изначально нет, поэтому она зависима от кучи других справочных таблиц.

Шаг 9: Действия при удалении записи главной таблицы (**CASCADE, SET NULL, SET DEFAULT, RESTRICT**)

Ссылка в интернете

С помощью выражения `ON DELETE` можно установить действия, которые выполняются для записей подчиненной таблицы при удалении связанной строки из главной таблицы. При удалении можно установить следующие опции:

- `CASCADE`: автоматически удаляет строки из зависимой таблицы при удалении связанных строк в главной таблице.
- `SET NULL`: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение **`NULL`**. (В этом случае столбец внешнего ключа должен поддерживать установку **`NULL`**).
- `SET DEFAULT` похоже на `SET NULL` за тем исключением, что значение внешнего ключа устанавливается не в `NULL`, а в значение по умолчанию для данного столбца.
- `RESTRICT`: отклоняет удаление строк в главной таблице при наличии связанных строк в зависимой таблице.

Пример

Будем считать, что при удалении автора из таблицы **author**, необходимо удалить все записи о книгах из таблицы **book**, написанные этим автором. Данное действие необходимо прописать при создании таблицы.

Запрос:

```
CREATE TABLE book (
    book_id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(50),
    author_id INT,
    price DECIMAL(8,2),
    amount INT,
    FOREIGN KEY (author_id)  REFERENCES author (author_id) ON DELETE CASCADE
);
```

Задание (Создать таблицу, связанную с другими таблицами)

Создать таблицу **book** той же структуры, что и на предыдущем шаге. Будем считать, что при удалении автора из таблицы **author**, должны удаляться все записи о книгах из таблицы **book**, написанные этим автором. А при удалении жанра из таблицы **genre** для соответствующей записи **book** установить значение **Null** в столбце **genre_id**.

Результат:

```
Affected rows: 0
```

Пояснение. По умолчанию все столбцы **могут иметь пустое значение**. Для того чтобы не допускать пустых значений в столбце, при создании таблицы необходимо указать **NOT Null** после типа столбца.

Шаг 10: Заполнение таблицы с внешними ключами

Ссылка в интернете

На предыдущих шагах были созданы и заполнены таблицы **author**:

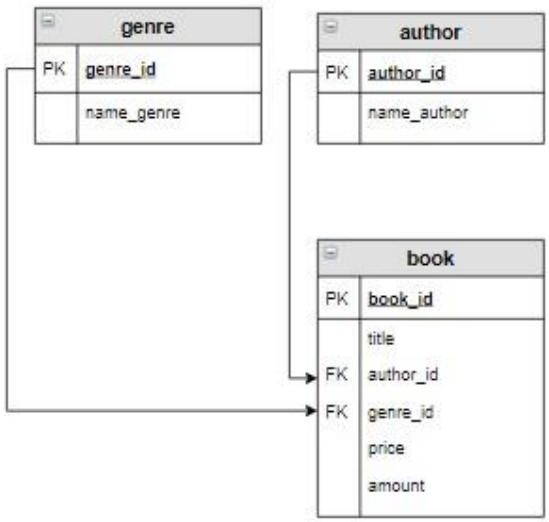
author_id	name_author
1	Булгаков М.А.
2	Достоевский Ф.М.
3	Есенин С.А.

4	Пастернак Б.Л.

и **genre**:

genre_id	name_genre
1	Роман
2	Поэзия

Эти таблицы являются главными для таблицы **book** и связаны с ней через внешние ключи:



При заполнении таблицы **book** в связанные столбцы необходимо заносить значения ключей главной таблицы. Например, Книгу «Игрок» написал Достоевский, поэтому значение поля **author_id** для этой записи должно быть 2, так как значение ключа для этого автора в таблице **author** равно 2. Значение поля **genre_id** для книги «Игрок» – 1, так как эта книга относится к жанру «Роман».

Задание (Для каждой строки book занести значения из author и genre)

Для каждой строки таблицы **book** занесите значения в поля **author_id** и **genre_id**. Считать, что книга Есенина относится к жанру «Поэзия», остальные книги – к жанру «Роман».

Через запятую перечислены значения полей **book_id**, **title**, **author_id**, **genre_id**, **price**, **amount** каждой записи таблицы **book**. Заполните пропуски.

Авторы и их произведения:

Название книги	Автор	Цена	Количество
Мастер и Маргарита	Булгаков М.А.	670.99	3
Белая гвардия	Булгаков М.А.	540.50	5
Идиот	Достоевский Ф.М.	460.00	10

Братья Карамазовы	Достоевский Ф.М.	799.01	3
Игрок	Достоевский Ф.М.	480.50	10
Стихотворения и поэмы	Есенин С.А.	650.00	15

Комментарии учащихся:

1.
Олег Григорьев

"Эти таблицы являются главными для таблицы **book** и связаны с ней через внешние ключи"

Если нетрудно, поясните, пожалуйста, на этом примере ещё раз, почему genre и author главные по отношению к book?

Андрей Окунев

@Олег, Первичные ключи таблиц genre (genre_id), author(author_id) являются внешними ключами для таблицы book

Шаг 11: Задание (Добавить три последние записи)

Ссылка в интернете

Добавьте три последние записи (с ключевыми значениями 6, 7, 8) в таблицу **book**, первые 5 записей уже добавлены:

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

Результат :

Affected rows: 1

Affected rows: 1

Affected rows: 1

Query result:

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	3
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

УРОК 2.2: Запросы на выборку, соединение таблиц

Шаг 1: Содержание урока

Ссылка в интернете

В запросах SQL могут участвовать несколько таблиц базы данных. При этом необходимо указать как эти таблицы соединены между собой.

Операция соединения **JOIN** предназначена для обеспечения выборки данных из двух таблиц и включения этих данных в один результирующий набор. При необходимости соединения не двух, а нескольких таблиц, операция соединения применяется несколько раз (последовательно).

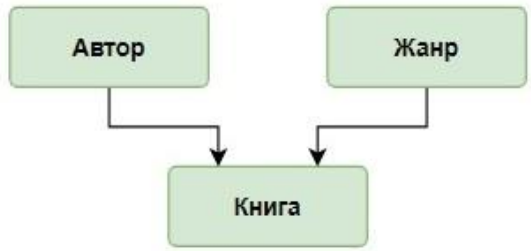
Операторы соединения входят в раздел **FROM** SQL запросов.

В данном уроке будут созданы запросы:

- для двух таблиц, внутреннее соединение INNER JOIN;
- для двух таблиц, внешние соединения LEFT JOIN и RIGHT JOIN;
- для двух таблиц, перекрестное соединение CROSS JOIN;
- выборки данных из нескольких таблиц;
- выборки данных из нескольких таблиц с группировкой;
- выборки данных с применением вложенных запросов;
- вложенные запросы в операторах соединения,
- операторы соединения, использование USING.

Структура и наполнение таблиц

Концептуальная схема базы данных:



Логическая схема базы данных:

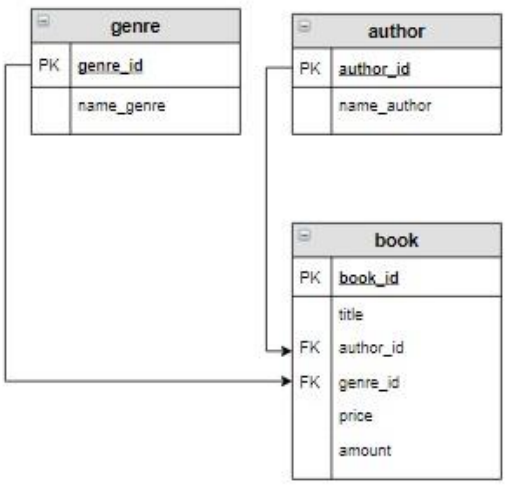


Таблица **author** (создание, заполнение):

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, заполнение, рассмотрено в качестве примеров):

genre_id	name_genre
1	Роман
2	Поэзия
3	Приключения

Таблица **book** (создание, заполнение):

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	3
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

Комментарии учащихся:

1.
Игорь Владимирович Лапшин
Хорошее объяснение JOIN: https://www.youtube.com/watch?v=EHvzvwAv7RU&t=15s
Alexey Karelskiy

@Игорь_Владимирович_Лапшин, Огромное спасибо! Очень понятное и доступное объяснение

2.
Arina Merkulova
Просто о JOIN (из серии видео SQL на котиках):
https://www.youtube.com/watch?v=PTAkqURmI0s
Мне это зашло лучше, чем всё :)

Шаг 2: Соединение **INNER JOIN**

Ссылка в интернете

Оператор внутреннего соединения `INNER JOIN` соединяет две таблицы. Порядок таблиц для оператора неважен, поскольку оператор является симметричным.

```
SELECT
 ...
FROM
 таблица_1 INNER JOIN  таблица_2
 ON условие
...
```

Результат запроса формируется так:

каждая строка одной таблицы сопоставляется с каждой строкой второй таблицы;

для полученной «соединённой» строки проверяется условие соединения;

если условие истинно, в таблицу результата добавляется соответствующая «соединённая» строка;

Пример

Вывести название книг и их авторов.

Запрос:

```
SELECT title, name_author
FROM
   author INNER JOIN book
   ON author.author_id = book.author_id;
```

Поскольку поля **author_id** в таблицах **book** и **author** называются одинаково, необходимо в запросах указывать полную ссылку на них (**book.author_id** и **author.author_id**).

Результат:

```
+----------------------+------------------+
| title                | name_author      |
+----------------------+------------------+
| Мастер и Маргарита   | Булгаков М.А.    |
```

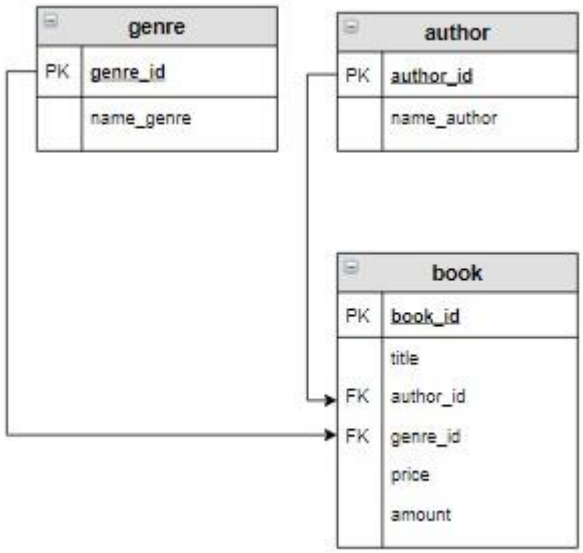
```
| Белая гвардия           | Булгаков М.А.     |
| Идиот                   | Достоевский Ф.М.  |
| Братья Карамазовы       | Достоевский Ф.М.  |
| Игрок                   | Достоевский Ф.М.  |
| Стихотворения и поэмы   | Есенин С.А.       |
| Черный человек          | Есенин С.А.       |
| Лирика                  | Пастернак Б.Л.    |
+-------------------------+-------------------+
```

В данном запросе осуществляется соединение главной таблицы **author** и зависимой таблицы **book** по ключевому столбцу **author.author_id** и внешнему ключу **book.author_id**. При этом в результирующую таблицу запроса включаются все строки, в которых значения этих столбцов совпадают. Другими словами строки зависимой таблицы **book** дополняются фамилией и инициалами авторов из таблицы **author**.

Задание (Вывести книги, количество которых больше 8)

Вывести название, жанр и цену тех книг, количество которых больше 8, в отсортированном по убыванию цены виде.

Логическая схема базы данных:



Результат:

```
+-------------------------+------------+--------+
| title                   | name_genre | price  |
+-------------------------+------------+--------+
| Стихотворения и поэмы   | Поэзия     | 650.00 |
| Игрок                   | Роман      | 480.50 |
| Идиот                   | Роман      | 460.00 |
+-------------------------+------------+--------+
```

Комментарии учащихся:

1.

Yury Popov

Немного синтаксического сахара:

1) Вместо `INNER JOIN` можно писать просто `JOIN`, разницы никакой;

2) Если условие соединения таблиц — это просто совпадение элементов в столбцах с одинаковыми именами в двух таблицах (а подавляющее большинство джойнов в этом курсе будет именно таким), то вместо

```
ON таблица_1.столбец_1 = таблица_2.столбец_1
AND таблица_1.столбец_2 = таблица_2.столбец_2
AND ...
```

можно написать просто

```
USING(столбец_1, столбец_2, ...)
```

.

Тогда для всех столбцов, по которым мы соединяли, в соединенную таблицу войдет только один столбец (в отличие от соединения по `ON`, в результате которого в таблице будут, например, столбцы `таблица_1.столбец_1` и `таблица_2.столбец_1`, содержащие дублирующиеся данные). В последовательных джойнах в курсе это позволит сильно уменьшить количество повторяющегося текста. Но вообще `ON` позволяет написать более общее условие соединения, так что про него не стоит забывать.

Еще есть `NATURAL JOIN`, который работает вообще без условий и просто соединяет таблицы по столбцам с одинаковым именем. Но, судя по всему, его лучше не использовать, ибо последствия могут быть непредсказуемыми %)

Почитать подробнее про USING и NATURAL JOIN можно вот здесь:
https://stackoverflow.com/questions/11366006/mysql-on-vs-using/11367066#11367066
https://stackoverflow.com/questions/8696383/difference-between-natural-join-and-inner-join

Дмитрий Ефремов

@Yury_Popov, данный синтаксис (USING) по-моему не поддерживается в T-SQL

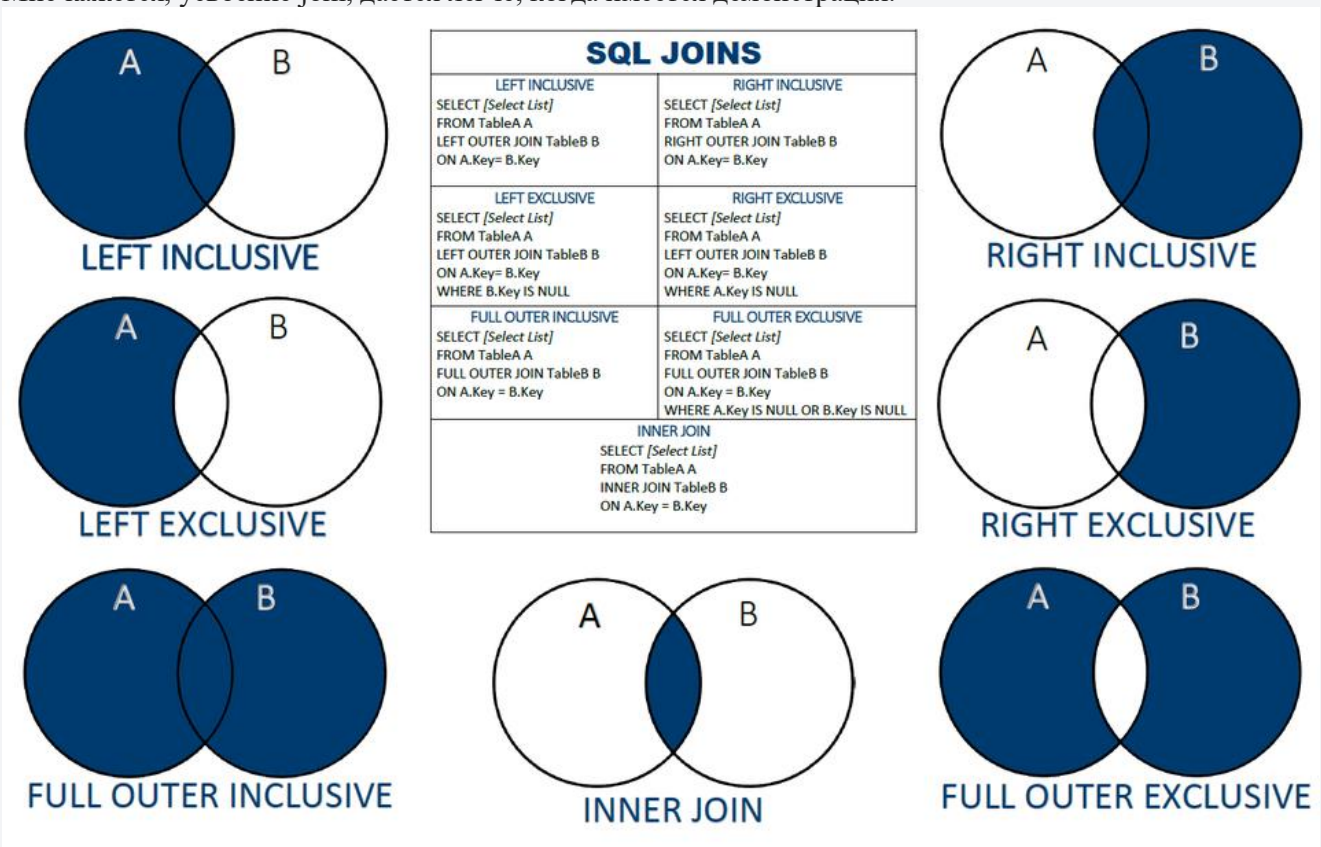
Галина Озерова

@Yury_Popov, На этом шаге USING рассматривается

2.

Abylaikhan Zulbukharov

Мне кажется, усвоение join, дается легче, когда имеется демонстрация.



Алексей Цибульников

@Abylaikhan_Zulbukharov, Если не ошибаюсь, когда смотрел интервью собеседований, то на техническом интервью очень не любят когда говорят про аналогию кругов венна и JOIN'ов. Это прямо как красное

полотно быку во время собеседования. Все потому что эти картинки верны только в ЧАСТНОМ случае, но не в ОБЩЕМ.
Кому интересно, в ссылке ниже детально объясняется почему эта аналогия плохая.
Не привыкайте к неправильному ! (*ссылка*)
Марина Савченко
@Алексей_Цибульников, спасибо, сломала мозг на ночь глядя =)))
Алексей Цибульников
@Марина_Савченко, у вас впереди целая ночь, чтобы его починить)

Шаг 3: Внешнее соединение **LEFT и RIGHT OUTER JOIN**

Ссылка в интернете

Оператор внешнего соединения `LEFT OUTER JOIN` (можно использовать `LEFT JOIN`) соединяет две таблицы. Порядок таблиц для оператора важен, поскольку оператор не является симметричным.

```
SELECT
...
FROM
таблица_1 LEFT JOIN таблица_2
ON условие
...
```

Результат запроса формируется так:

- в результат включается внутреннее соединение (`INNER JOIN`) первой и второй таблицы в соответствии с условием;
- затем в результат добавляются те записи первой таблицы, которые не вошли во внутреннее соединение на шаге 1, для таких записей соответствующие поля второй таблицы заполняются значениями `NULL`.

Соединение `RIGHT JOIN` действует аналогично, только в пункте 2 первая таблица меняется на вторую и наоборот.

Пример

Вывести название всех книг каждого автора, если книг некоторых авторов в данный момент нет на складе – вместо названия книги указать `Null`.

Запрос:

```
SELECT name_author, title
FROM author LEFT JOIN book
    on author.author_id = book.author_id
ORDER BY name_author;
```

Результат:

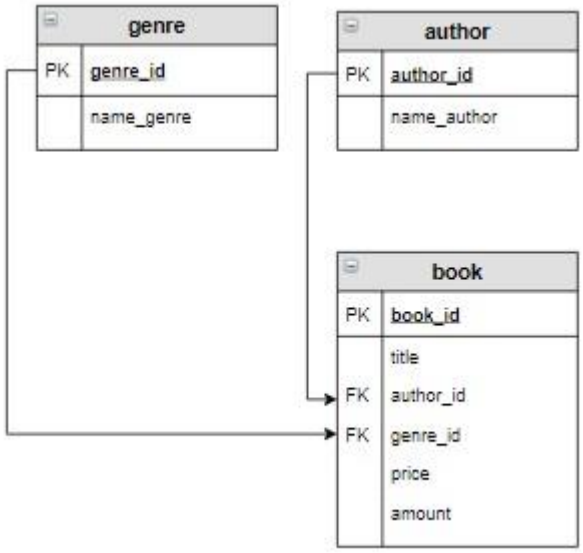
```
+------------------+----------------------+
| name_author      | title                |
+------------------+----------------------+
| Булгаков М.А.    | Мастер и Маргарита   |
| Булгаков М.А.    | Белая гвардия        |
| Достоевский Ф.М. | Игрок                |
| Достоевский Ф.М. | Идиот                |
| Достоевский Ф.М. | Братья Карамазовы    |
| Есенин С.А.      | Стихотворения и поэмы |
| Есенин С.А.      | Черный человек       |
| Лермонтов М.Ю.   | None                 |
| Пастернак Б.Л.   | Лирика               |
+------------------+----------------------+
```

Так как в таблице **book** нет книг Лермонтова, напротив этой фамилии стоит Null (None).

Задание (Вывести все жанры книг, которых нет на складе)

Вывести все жанры книг, которых нет на складе.

Логическая схема базы данных:



Результат:

```
+-------------+
| name_genre  |
+-------------+
| Приключения |
+-------------+
```

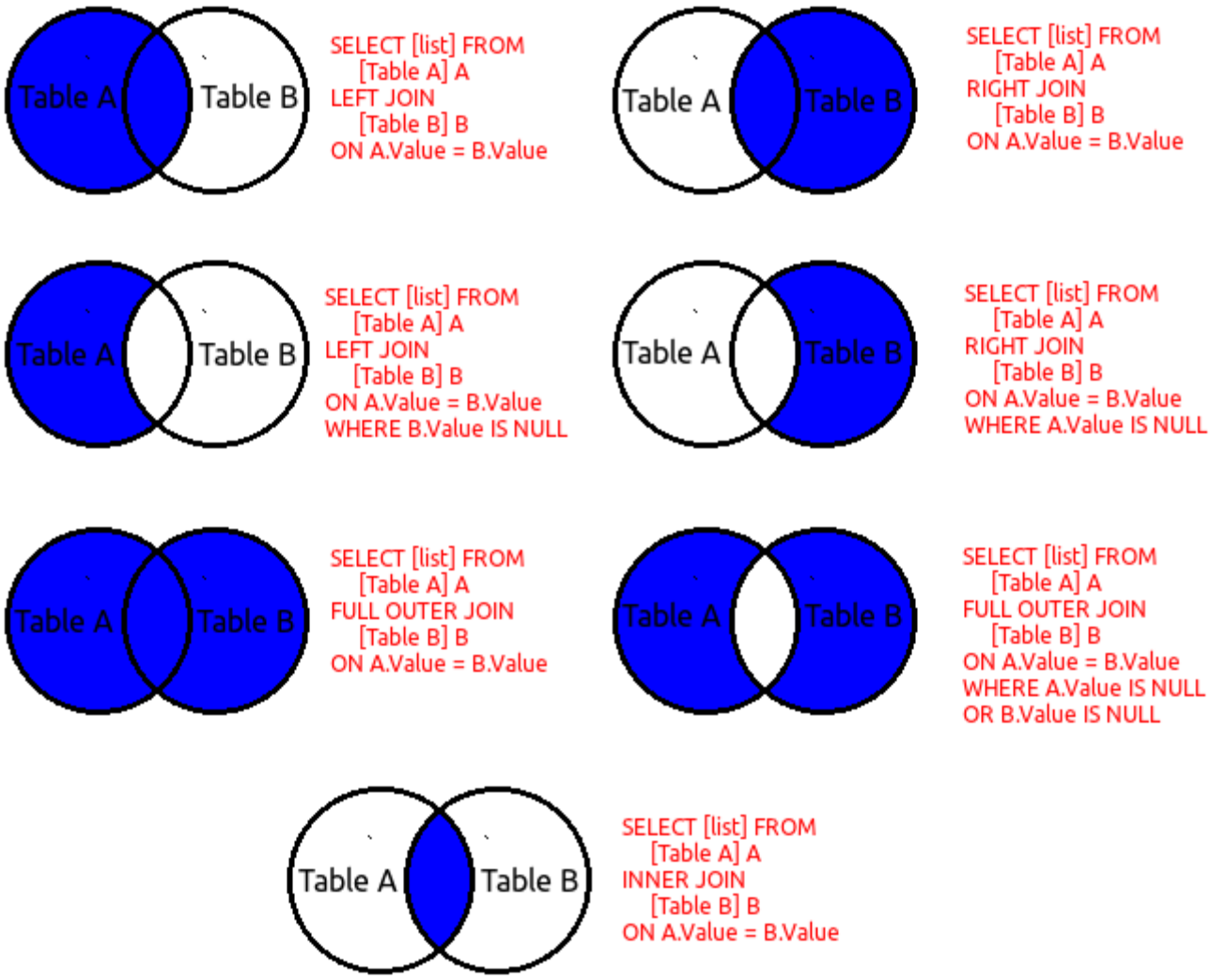
Пояснение. При использовании внешнего соединения названия книг и другие столбцы таблицы **book** для жанра тех книг, которого нет на складе, будут содержать значение Null.

Комментарии учащихся:

1.

Ilia Stepanov

Нашёл наглядное изображение джоинов



Шаг 4: Перекрестное соединение **CROSS JOIN, RAND(), FLOOR(), DATE_ADD()**

Ссылка в интернете

Оператор перекрёстного соединения, или декартова произведения `CROSS JOIN` (в запросе вместо ключевых слов можно поставить запятую между таблицами) соединяет две таблицы. Порядок таблиц для оператора неважен, поскольку оператор является симметричным. Его структура:

```
SELECT
 ...
FROM
 таблица_1 CROSS JOIN  таблица_2
...
```

или

```
SELECT
```

...

FROM

таблица_1, таблица_2

...

Результат запроса формируется так: каждая строка одной таблицы соединяется с каждой строкой другой таблицы, формируя в результате все возможные сочетания строк двух таблиц.

Например, запрос:

```
SELECT name_author, name_genre
FROM author, genre;
```

каждому автору из таблицы **author** поставит в соответствие все возможные жанры из таблицы **genre**:

```
+------------------+-------------+
| name_author      | name_genre  |
+------------------+-------------+
| Булгаков М.А.    | Роман       |
| Булгаков М.А.    | Поэзия      |
| Булгаков М.А.    | Приключения |
| Достоевский Ф.М. | Роман       |
| Достоевский Ф.М. | Поэзия      |
| Достоевский Ф.М. | Приключения |
| Есенин С.А.      | Роман       |
| Есенин С.А.      | Поэзия      |
| Есенин С.А.      | Приключения |
| Пастернак Б.Л.   | Роман       |
| Пастернак Б.Л.   | Поэзия      |
| Пастернак Б.Л.   | Приключения |
| Лермонтов М.Ю.   | Роман       |
| Лермонтов М.Ю.   | Поэзия      |
| Лермонтов М.Ю.   | Приключения |
+------------------+-------------+
```

Задание (Создать запрос проведения выставок)

Есть список городов, хранящийся в таблице **city**:

city_id	name_city
1	Москва
2	Санкт-Петербург
3	Владивосток

Необходимо в каждом городе провести выставку книг каждого автора в течение 2020 года. Дату проведения выставки выбрать случайным образом. Создать запрос, который выведет город, автора и дату проведения выставки. Последний столбец назвать **Дата**. Информацию вывести,

отсортировав сначала в алфавитном порядке по названиям городов, а потом по убыванию дат проведения выставок.

Структура таблицы:

Результат (даты при каждом запуске получаются разными, и не должны совпадать с приведенными значениями):

```
+----------------+-----------------+------------+
| name_city      | name_author     | Дата       |
+----------------+-----------------+------------+
| Владивосток    | Достоевский Ф.М. | 2020-12-04 |
| Владивосток    | Лермонтов М.Ю.  | 2020-10-21 |
| Владивосток    | Пастернак Б.Л.  | 2020-08-23 |
| Владивосток    | Есенин С.А.     | 2020-08-14 |
| Владивосток    | Булгаков М.А.   | 2020-01-08 |
| Москва         | Лермонтов М.Ю.  | 2020-09-30 |
| Москва         | Достоевский Ф.М. | 2020-07-21 |
| Москва         | Есенин С.А.     | 2020-06-23 |
| Москва         | Булгаков М.А.   | 2020-05-28 |
| Москва         | Пастернак Б.Л.  | 2020-04-08 |
| Санкт-Петербург | Булгаков М.А.   | 2020-11-05 |
| Санкт-Петербург | Лермонтов М.Ю.  | 2020-10-22 |
| Санкт-Петербург | Достоевский Ф.М. | 2020-09-19 |
| Санкт-Петербург | Есенин С.А.     | 2020-08-11 |
| Санкт-Петербург | Пастернак Б.Л.  | 2020-06-28 |
+----------------+-----------------+------------+
```

Пояснение.

1. Для генерации случайной даты можно к первому числу года ('2020-01-01') прибавить целое случайное число в интервале от 0 до 365.

Генерации случайных чисел в интервале от 0 до 1 (не включительно) осуществляется с помощью функции `RAND()`. Если эту функцию умножить на 365, то она будет генерировать вещественные числа от 0 до 365 (не включительно). Осталось только отбросить дробную часть. Это можно сделать с помощью функции `FLOOR()`, которая возвращает наибольшее целое число, меньшее или равное указанному числовому значению. Таким образом, случайное число от 0 до 365 можно получить с помощью выражения:

```
FLOOR(RAND() * 365)
```

2. Для сложения даты с числом используется функция:

DATE_ADD (дата, INTERVAL число единица_измерения),

где

единица_измерения (использовать строчные буквы) – это день (DAY), месяц(MONTH), неделя(WEEK) и пр.,

число – целое число,

дата – значение даты или даты и времени.

Функция к **дате** прибавляет указанное **число**, выраженное в днях, месяцах и пр. , в зависимости от заданного интервала, и возвращает новую дату.

Например:

DATE_ADD('2020-02-02', INTERVAL 45 DAY) возвращает 18 марта 2020 года

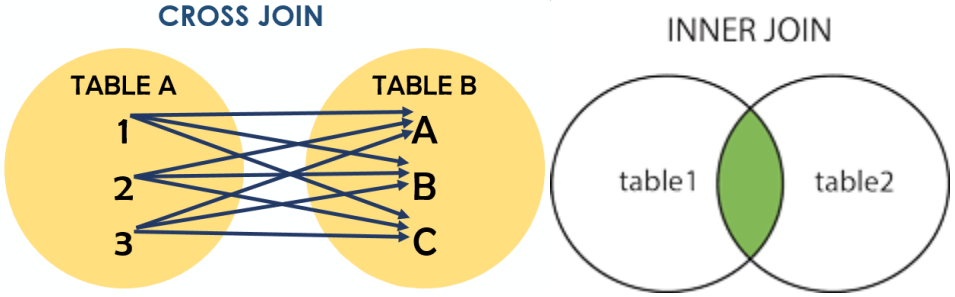
DATE_ADD('2020-02-02', INTERVAL 6 MONTH) возвращает 2 августа 2020 года

Комментарии учащихся:

1.

Никита Синчинов

Для тех, кто лучше воспринимает картинки, вот разница между CROSS JOIN и INNER JOIN.



2.

Дмитрий Чернов

Явно, тут много кто будет приводить в пример картинки с кругами (Диаграммы Венна) , самое лучшее понимание JOIN-ов = вы можете подчерпнуть из двух данных тредов на хабре:

Часть первая. Понимание джойнов сломано.

Часть вторая. Попытка альтернативной визуализации.

Dmitriy Novikov

@Дмитрий_Чернов, вторая статья показалась интересной, но не вижу ничего плохого в "пересечениях".

Олег Григорьев

@Дмитрий_Чернов, отличные статьи, большое спасибо.

Макс Качаев

@Олег_Григорьев,

С первого поста на хабре нашел такую запись "

Кстати, Еще маленький совет по производительности. Если нужно просто найти элементы в таблице, которых нет в другой таблице, то лучше использовать не 'LEFT JOIN… WHERE… IS NULL', а конструкцию EXISTS. Это и читабельнее, и быстрее."

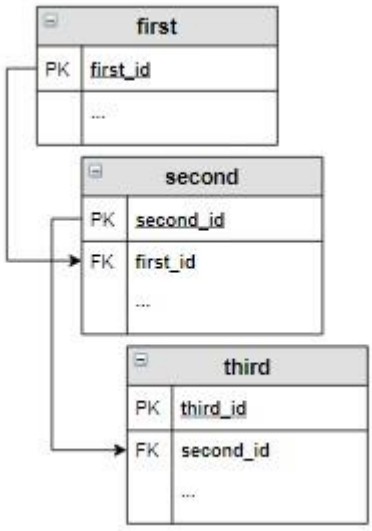
Интересно в этом курсе будет про EXISTS

Шаг 5: Запросы на выборку из нескольких таблиц

Ссылка в интернете

Запрос на выборку может выбирать данные из двух и более таблиц базы данных. При этом таблицы должны быть логически связаны между собой. Для каждой пары таблиц, включаемых в запрос, необходимо указать свой оператор соединения. Наиболее распространенным является внутренне соединение `INNER JOIN`, поэтому в примерах будем использовать его.

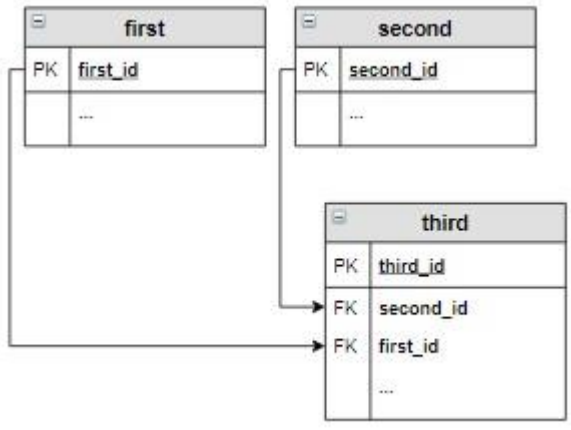
Пусть таблицы связаны между собой следующим образом:



тогда запрос на выборку для этих таблиц будет иметь вид:

```
SELECT
 ...
FROM
 first INNER JOIN  second
 ON first.first_id = second.first_id
     INNER JOIN  third
     ON second.second_id = third.second_id
...
```

Если же таблицы связаны так:



то запрос на выборку выглядит следующим образом:

```
SELECT
 ...
FROM
 first INNER JOIN  third
```

```
ON first.first_id = third.first_id
    INNER JOIN second
    ON third.second_id = second.second_id
...
```

В этом случае рекомендуется соединение таблиц записывать последовательно, «по кругу»: **first → third → second**.

Пример

Вывести информацию о тех книгах, их авторах и жанрах, цена которых принадлежит интервалу от 500 до 700 рублей включительно.

Запрос:

```
SELECT title, name_author, name_genre, price, amount
FROM
 author INNER JOIN  book
 ON author.author_id = book.author_id
    INNER JOIN genre
    ON genre.genre_id = book.genre_id
WHERE price BETWEEN 500 AND 700;
```

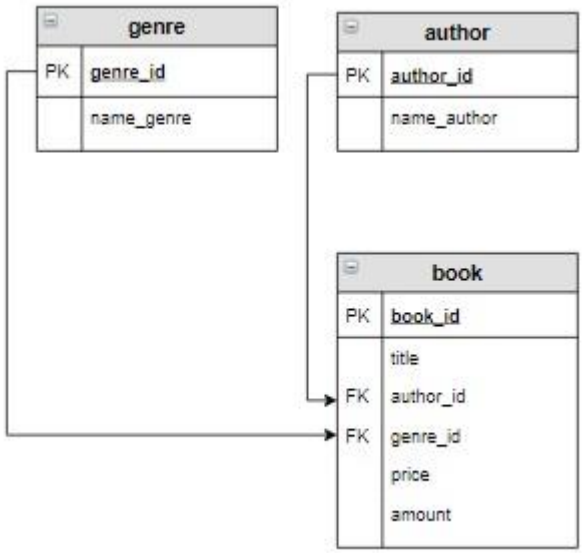
Результат:

```
+----------------------+----------------+------------+--------+--------+
| title                | name_author    | name_genre | price  | amount |
+----------------------+----------------+------------+--------+--------+
| Мастер и Маргарита   | Булгаков М.А.  | Роман      | 670.99 | 3      |
| Белая гвардия        | Булгаков М.А.  | Роман      | 540.50 | 5      |
| Стихотворения и поэмы | Есенин С.А.   | Поэзия     | 650.00 | 15     |
| Черный человек       | Есенин С.А.    | Поэзия     | 570.20 | 6      |
| Лирика               | Пастернак Б.Л. | Поэзия     | 518.99 | 2      |
+----------------------+----------------+------------+--------+--------+
```

Задание (Вывести книги, где есть слово «роман»)

Вывести информацию о книгах (жанр, книга, автор), относящихся к жанру, включающему слово «роман» в отсортированном по названиям книг виде.

Логическая схема базы данных:



Результат:

```
+------------+--------------------+------------------+
| name_genre | title              | name_author      |
+------------+--------------------+------------------+
| Роман      | Белая гвардия      | Булгаков М.А.    |
| Роман      | Братья Карамазовы  | Достоевский Ф.М. |
| Роман      | Игрок              | Достоевский Ф.М. |
| Роман      | Идиот              | Достоевский Ф.М. |
| Роман      | Мастер и Маргарита | Булгаков М.А.    |
+------------+--------------------+------------------+
```

Шаг 6: Запросы для нескольких таблиц с группировкой

Ссылка в интернете

В запросах с групповыми функциями могут использоваться несколько таблиц, между которыми используются различные типы соединений.

Пример

Вывести количество различных книг каждого автора. Информацию отсортировать в алфавитном порядке по фамилиям авторов.

Запрос:

```
SELECT name_author, count(title) AS Количество
FROM author INNER JOIN book
    on author.author_id = book.author_id
GROUP BY name_author
ORDER BY name_author;
```

Результат

```
+------------------+------------+
| name_author      | Количество |
+------------------+------------+
```

```
| Булгаков М.А.     | 2           |
| Достоевский Ф.М.  | 3           |
| Есенин С.А.       | 2           |
| Пастернак Б.Л.    | 1           |
+-------------------+-------------+
```

При использовании соединения INNER JOIN мы не можем узнать, что книг Лермонтова на складе нет, но предполагается, что они могут быть. Чтобы автор Лермонтов был включен в результат, нужно изменить соединение таблиц.

Запрос:

```
SELECT name_author, count(title) AS Количество
FROM author LEFT JOIN book
    on author.author_id = book.author_id
GROUP BY name_author
ORDER BY name_author;
```

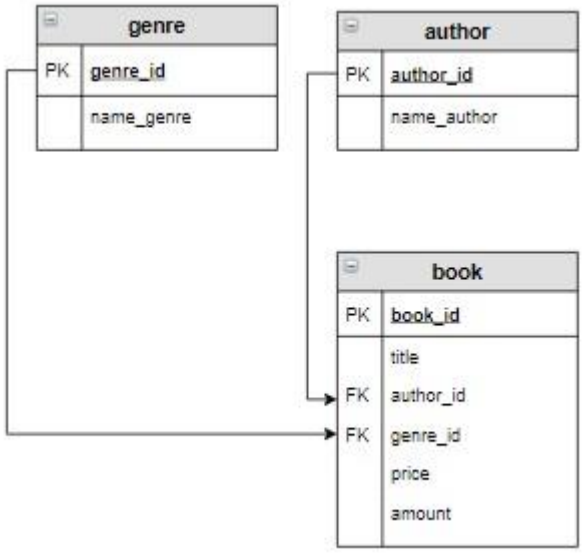
Результат:

```
+-------------------+-------------+
| name_author       | Количество  |
+-------------------+-------------+
| Булгаков М.А.     | 2           |
| Достоевский Ф.М.  | 3           |
| Есенин С.А.       | 2           |
| Лермонтов М.Ю.    | 0           |
| Пастернак Б.Л.    | 1           |
+-------------------+-------------+
```

Задание (Вывести авторов, количество книг которых меньше 10)

Посчитать количество экземпляров книг каждого автора на складе. Вывести тех авторов, количество книг которых меньше 10, в отсортированном по возрастанию количества виде. Последний столбец назвать **Количество**.

Логическая схема базы данных:



Результат:

```
+---------------+------------+
| name_author   | Количество |
+---------------+------------+
| Лермонтов М.Ю. | None       |
| Пастернак Б.Л. | 2          |
| Булгаков М.А.  | 8          |
+---------------+------------+
```

Пояснение. Чтобы в результат были включены авторы, книг которых на складе нет, необходимо в условии отбора, кроме того, что общее количество книг каждого автора меньше 10, учесть, что у автора вообще может не быть книг (то есть **COUNT(title) = 0**).

Шаг 7: Запросы для нескольких таблиц со вложенными запросами

Ссылка в интернете

В запросах, построенных на нескольких таблицах, можно использовать вложенные запросы. Вложенный запрос может быть включен: после ключевого слова SELECT, после FROM и в условие отбора после WHERE (HAVING).

Пример

Вывести авторов, общее количество книг которых на складе максимально.

Это достаточно сложный запрос, поэтому будем решать его по шагам (реализуя каждый запрос по отдельности), а потом объединим все запросы в один.

Шаг 1. Найдем суммарное количество книг на складе по каждому автору. Поскольку фамилии автора в этой таблице нет, то группировку будем осуществлять по **author_id**.

Запрос:

```
SELECT author_id, SUM(amount) AS sum_amount FROM book GROUP BY author_id
```

Результат:

```
+-----------+------------+
| author_id | sum_amount |
+-----------+------------+
| 1         | 8          |
| 2         | 23         |
| 3         | 21         |
| 4         | 2          |
+-----------+------------+
```

Шаг 2. В результирующей таблице предыдущего запроса необходимо найти максимальное значение, то есть 23. Для этого запросу, созданному на шаге 1, необходимо присвоить имя (например, **query_in**) и использовать его в качестве таблицы-источника после FROM. Затем уже находить максимум по столбцу **sum_amount**.

Запрос:

```
SELECT MAX(sum_amount) AS max_sum_amount
FROM (SELECT author_id, SUM(amount) AS sum_amount
```

```
    FROM book GROUP BY author_id) query_in
```

Результат:

```
+----------------+
| max_sum_amount |
+----------------+
| 23             |
+----------------+
```

Шаг 3. Выведем фамилию автора и общее количество книг для него.

Запрос:

```
SELECT name_author, SUM(amount) as Количество
FROM author INNER JOIN book
    on author.author_id = book.author_id
GROUP BY name_author
```

Результат:

```
+------------------+------------+
| name_author      | Количество |
+------------------+------------+
| Булгаков М.А.    | 8          |
| Достоевский Ф.М. | 23         |
| Есенин С.А.      | 21         |
| Пастернак Б.Л.   | 2          |
+------------------+------------+
```

Шаг 4. Включим запрос с шага 2 в условие отбора запроса с шага 3. И получим всех авторов, общее количество книг которых максимально.

Запрос:

```
SELECT name_author, SUM(amount) as Количество
FROM author INNER JOIN book
    on author.author_id = book.author_id
GROUP BY name_author
HAVING SUM(amount) =
    (SELECT MAX(sum_amount) AS max_sum_amount
     FROM (SELECT author_id, SUM(amount) AS sum_amount
           FROM book GROUP BY author_id) query_in);
```

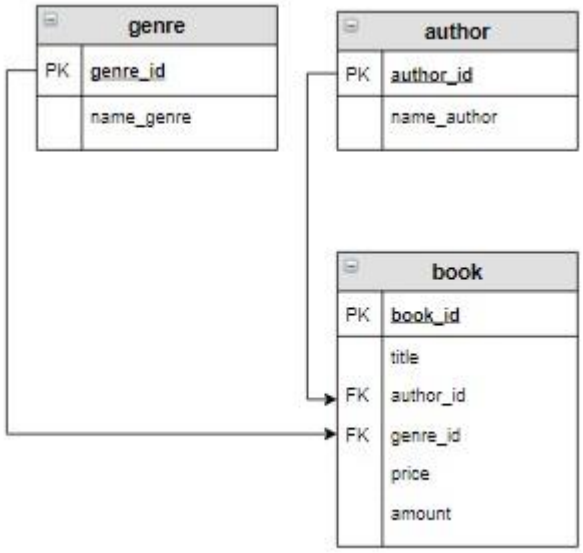
Результат:

```
+------------------+------------+
| name_author      | Количество |
+------------------+------------+
| Достоевский Ф.М. | 23         |
+------------------+------------+
```

Задание (Вывести авторов, которые пишут в одном жанре)

Вывести в алфавитном порядке всех авторов, которые пишут только в одном жанре. Поскольку у нас в таблицах так занесены данные, что у каждого автора книги только в одном жанре, для этого запроса внесем изменения в таблицу **book**. Пусть у нас книга Есенина «Черный человек» относится к жанру «Роман», а книга Булгакова «Белая гвардия» к «Приключениям» (эти изменения в таблицы уже внесены).

Логическая схема базы данных:



Результат:

```
+------------------+
| name_author      |
+------------------+
| Достоевский Ф.М. |
| Пастернак Б.Л.   |
+------------------+
```

Пояснение. Этот запрос рекомендуется реализовать по шагам (можно придумать и другой алгоритм):

- сначала отобрать различные записи по **id** авторов и жанров в таблице **book**;
- затем по предыдущему запросу посчитать количество жанров для каждого автора и отобрать тех, у которых один жанр (получится список из **id** авторов);
- и наконец, отобрать авторов, **id** которых есть в списке предыдущего запроса.

Шаг 8: Вложенные запросы в операторах соединения

Ссылка в интернете

Вложенные запросы могут использоваться в операторах соединения JOIN. При этом им необходимо присваивать имя, которое записывается сразу после закрывающей скобки вложенного запроса.

```
SELECT

 ...
```

```
FROM
 таблица ... JOIN  (SELECT ...) имя_вложенного_запроса
 ON условие
...
```

Вложенный запрос может стоять как справа, так и слева от оператора JOIN. Допускается использование двух запросов в операторах соединения.

Пример

Вывести авторов, пишущих книги в самом популярном жанре. Самым популярным считать жанр, общее количество экземпляров книг которого на складе максимально. Таких жанров может быть несколько, если они имеют одинаковое максимальное значение общего количества экземпляров. Только для этого шага изменена запись в таблице **book**.

book_id	Title	author_id	genre_id	price	amount
8	Лирика	4	2	518.9910	10

А также добавлены новые записи:

book_id	Title	author_id	genre_id	price	amount
9	Герой нашего времени	5	3	570.59	2
10	Доктор Живаго	4	3	740.50	5

Рассмотрим реализацию этого запроса по шагам.

Шаг 1. Найдем общее количество книг по каждому жанру, отсортируем его по убыванию и ограничим вывод одной строкой. Рекомендуется, если запрос будет использоваться в качестве вложенного (особенно в операциях соединения), вычисляемым полям запроса давать собственное имя.

Запрос:

```
SELECT genre_id, SUM(amount) AS sum_amount
FROM book
GROUP BY genre_id
ORDER BY sum_amount DESC
LIMIT 1
```

Результат:

```
+---------------+------------+
|  genre_id     | sum_amount |
+---------------+------------+
| 1             | 31         |
```

```
+---------------+------------+
```
Кажется, что, уже используя этот запрос, можно получить id самого популярного жанра. Но это не так, поскольку несколько жанров могут иметь одинаковую популярность. Поэтому нам необходим запрос, который отберет ВСЕ жанры, суммарное количество книг которых равно **sum_amount**.

Шаг 2. Используя запрос с предыдущего шага, найдем **id** самых популярных жанров.

Запрос:

```
SELECT query_in_1.genre_id
FROM (SELECT genre_id, SUM(amount) AS sum_amount
    FROM book
    GROUP BY genre_id
    )query_in_1
    INNER JOIN (SELECT genre_id, SUM(amount) AS sum_amount
            FROM book
            GROUP BY genre_id
            ORDER BY sum_amount DESC
            LIMIT 1
            ) query_in_2
   on query_in_1.sum_amount= query_in_2.sum_amount
```

Результат:

```
+----------+
| genre_id |
+----------+
| 1        |
| 2        |
+----------+
```

Шаг 3. Используя запрос с шага 2, выведем фамилии авторов, которые пишут в самых популярных жанрах. В этом запросе обязательно выполнить группировку по фамилиям авторов и **id** жанров, так как без этого фамилии авторов будут повторяться, поскольку в таблице book есть разные книги, написанные автором в одном жанре.

Запрос:

```
SELECT  name_author
FROM author INNER JOIN book
    on author.author_id = book.author_id
GROUP BY name_author, genre_id
HAVING genre_id IN
    (SELECT query_in_1.genre_id
     FROM (SELECT genre_id, SUM(amount) AS sum_amount
```

```
        FROM book
        GROUP BY genre_id
       )query_in_1
    INNER JOIN (SELECT genre_id, SUM(amount) AS sum_amount
            FROM book
            GROUP BY genre_id
            ORDER BY sum_amount DESC
            LIMIT 1
           ) query_in_2
     on query_in_1.sum_amount= query_in_2.sum_amount
    );
```

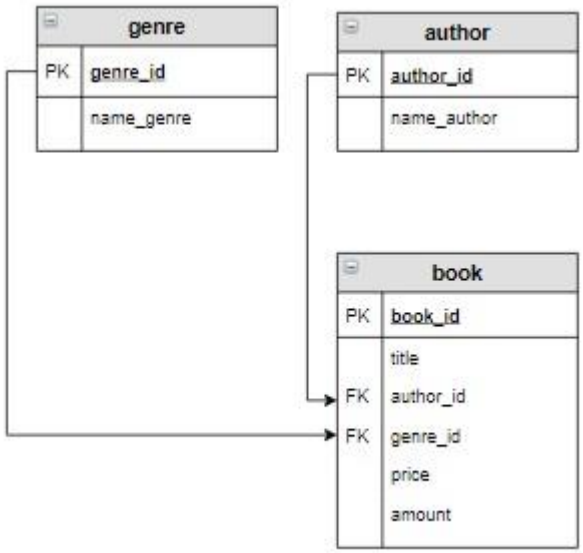
Результат:

```
+------------------+
| name_author      |
+------------------+
| Булгаков М.А.    |
| Достоевский Ф.М. |
| Есенин С.А.      |
| Пастернак Б.Л.   |
+------------------+
```

Задание (Вывести информацию о книгах, написанных в популярных жанрах)

Вывести информацию о книгах, написанных в самых популярных жанрах, в отсортированном в алфавитном порядке по названию книг виде.

Логическая схема базы данных:



Результат:

title	name_author	name_genre	price	amount
Белая гвардия	Булгаков М.А.	Роман	540.50	5
Братья Карамазовы	Достоевский Ф.М.	Роман	799.01	3
Игрок	Достоевский Ф.М.	Роман	480.50	10
Идиот	Достоевский Ф.М.	Роман	460.00	10
Лирика	Пастернак Б.Л.	Поэзия	518.99	10
Мастер и Маргарита	Булгаков М.А.	Роман	670.99	3
Стихотворения и поэмы	Есенин С.А.	Поэзия	650.00	15
Черный человек	Есенин С.А.	Поэзия	570.20	6

Комментарии учащихся:

1.

Paul Setchko

если начать отсеивать данные вот так:

```
select genre_id, sum(amount) as sum_amount
from book
group by 1;
```

то почему оно работает, гм, НЕОЖИДАННО, если добавить

```
having sum_amount = max(sum_amount)
```

(вернёт только одну строку вместо ожидаемых двух)

и почему вариант выше работает в принципе при том, что алиас sum_amount присваивается столбцу уже на этапе, когда сделано отсеивание информации:

1. from
2. where
3. group by
4. **having** (который не знает про алиас sum_amount)
5. **select** (тут присвоили алиас)
6. order by

при этом не работает аналогичная, как мне кажется до сих пор, фильтрация через

```
having sum(amount) = max(sum(amount))
```

Галина Озерова

@Paul_Setchko, Группировка выполняется по всей таблице (поскольку не указано GROUP BY) . А Max Вы пытаетесь вычислить среди уже сгруппированных данных. Фактически Вы пытаетесь реализовать "вложенную" группировку.... Поэтому и не идет.

Paul Setchko

@Галина_Озерова, а, т.е. вы имеете в виду, что в having sum(amount) у меня sum считается по оригинальному столбцу amount, а не по сгруппированным?
ок, но я всё ещё не знаю, почему having знает про алиас sum_amount и почему при использовании знака равенства возвращается одна строка, а не две :(
Галина Озерова
@Paul_Setchko, Запрос на таком количестве записей просто не показательный, он неправильный и не может использоваться для решения задачи (я про это sum_amount = max(sum_amount)). Почему Having видит алиас - тоже удивилась - но нашла информацию в интернете о том, что разработчики интерпретаторов SQL меняют порядок обработки запросов, и в некоторых системах он (порядок) отличается от классического. Насколько это правда - не знаю. Ну судя по Вашему запросу - это так. Порядок выполнения запросов в курсе вроде правильно написан, в соответствии с теорией. Я еще раз проверила.
Paul Setchko
@Галина_Озерова, я тоже думал про оптимизацию со стороны интерпретатора, но спасибо (:
неправильный запрос про sum_amount -- это вы сюда же отвечали, про то, что оно видно в пределах having, или там в плане логики в запросе целиком что-то не так?
и я всё ещё не понимаю, почему при использовании знака равенства возвращается одна строка, а не две (: (ну или тут, как я и выше предполагал, чото в плане логики я совсем не так понял, и вопрос абсолютно некорректный, или что)

2.
Инна Стародубцева
Кому интересно, этот пример можно решить еще вот так: то есть с помощью двух вложенных запросов a и b.
Пример:
Вывести авторов, пишущих книги в самом популярном жанре. Самым популярным считать жанр, общее количество экземпляров книг которого на складе максимально. Таких жанров может быть несколько, если они имеют одинаковое максимальное значение общего количества экземпляров. Только для этого шага изменена запись в таблице **book**.

```
1. select name_author
2. from author join (select distinct author_id
3. from book
4.  join (select genre_id, sum(amount) sum_amount
5.                 from book
6.                 group by genre_id
7.               order by sum_amount desc
8.                 limit 2) a using (genre_id)) b using (author_id)
```

еще лайфхаки:
1 - inner можно опускать, просто пишем join (предполагается по умолчанию inner)
2 - когда джойним, то можно писать так: `from book join author on author.author_id= book.author_id`, а можно так: `from book join author using (author_id)`
Важно!!!! второй способ используем только если уверены в том, что названия внешних ключей совпадают!!!

Шаг 9: Операция соединение, использование **JOIN… USING()**

Ссылка в интернете

Данный шаг добавлен по предложениям пользователей (Валерий Родькин, Todor Illia и другие).

При описании соединения таблиц с помощью **JOIN** в некоторых случаях вместо **ON** и следующего за ним условия можно использовать оператор **USING()**.

USING позволяет указать набор столбцов, которые есть в обеих объединяемых таблицах. Если база данных хорошо спроектирована, а каждый внешний ключ имеет такое же имя, как и соответствующий первичный ключ (например, **genre.genre_id = book.genre_id**), тогда можно использовать предложение **USING** для реализации операции **JOIN**.

При этом после **SELECT**, при использовании столбцов из **USING()**, необязательно указывать, из какой именно таблицы берется столбец.

Пример

Вывести название книг, фамилии и **id** их авторов.

Запрос:

Вариант с **ON**

```
SELECT title, name_author, author.author_id /* явно указать таблицу - обязательно */
FROM
  author INNER JOIN book
  ON author.author_id = book.author_id;
```

Вариант с **USING**

```
SELECT title, name_author, author_id /* имя таблицы, из которой берется author_id, указывать не обязательно*/
FROM
  author INNER JOIN book
  USING(author_id);
```

Результат (одинаковый для обоих запросов):

```
+----------------------+------------------+-----------+
| title                | name_author      | author_id |
+----------------------+------------------+-----------+
| Мастер и Маргарита   | Булгаков М.А.    | 1         |
| Белая гвардия        | Булгаков М.А.    | 1         |
| Идиот                | Достоевский Ф.М. | 2         |
| Братья Карамазовы    | Достоевский Ф.М. | 2         |
| Игрок                | Достоевский Ф.М. | 2         |
| Стихотворения и поэмы | Есенин С.А.     | 3         |
| Черный человек       | Есенин С.А.      | 3         |
| Лирика               | Пастернак Б.Л.   | 4         |
+----------------------+------------------+-----------+
```

Запись условия соединения с **ON** является более общим случаем, так как

- позволяет задавать соединение не только по одноименным полям;
- позволяет использовать произвольное условие на соединение таблиц, при этом в условие может включаться произвольное выражение.

Пример

В таблице **supply** занесена информация о книгах, поступивших на склад.

supply_id	Title	author	Price	amount
1	Доктор Живаго	Пастернак Б.Л.	618.99	3
2	Черный человек	Есенин С.А.	570.20	6
3	Евгений Онегин	Пушкин А.С.	440.80	5
4	Идиот	Достоевский Ф.М.	360.80	3

Если в таблицах **supply** и **book** есть одинаковые книги, вывести их название и автора.

Запрос:

```
SELECT book.title, name_author
FROM author INNER JOIN book
USING (author_id)
    INNER JOIN supply
    ON book.title = supply.title and author.name_author = supply.author;
```

Результат:

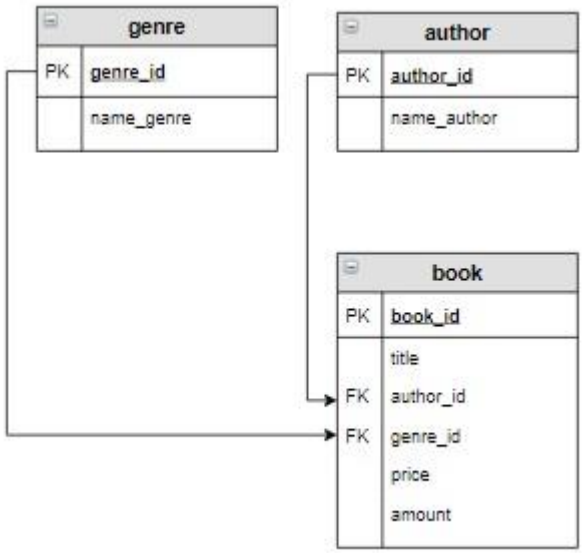
```
+----------------+------------------+
| title          | name_author      |
+----------------+------------------+
| Идиот          | Достоевский Ф.М. |
| Черный человек | Есенин С.А.      |
+----------------+------------------+
```

В данном примере для соединения **book** и **supply** использовать **USING** нельзя, так как в таблице **book** фамилий авторов вообще нет (их необходимо получить из таблицы **author**, столбец **name_author**), а в таблице **supply** фамилии занесены в столбец **author**.

Задание (Вывести информацию об одинаковых книгах из таблиц supply и book)

Если в таблицах **supply** и **book** есть одинаковые книги, которые имеют равную цену, вывести их название и автора, а также посчитать общее количество экземпляров книг в таблицах **supply** и **book**, столбцы назвать **Название**, **Автор** и **Количество**.

Схема данных:



Результат:

```
+----------------+-------------+------------+
| Название       | Автор       | Количество |
+----------------+-------------+------------+
| Черный человек | Есенин С.А. | 12         |
+----------------+-------------+------------+
```

Шаг 10: Задание (Придумать запрос для таблиц **book**, **author**, **genre** и **city)**

Ссылка в интернете

Придумайте один или несколько запросов для таблиц **book**, **author**, **genre** и **city**. Проверьте, правильно ли они работают.

При желании можно формулировку запросов разместить в комментариях.

Размещенные задания можно использовать для закрепления материала урока.

Оценивайте понравившиеся Вам запросы.

В последнем модуле создан отдельный урок, в котором мы разместим запросы, набравшие наибольшее количество лайков.

УРОК 2.3: Запросы корректировки, соединение таблиц

Шаг 1: Содержание урока

Ссылка в интернете

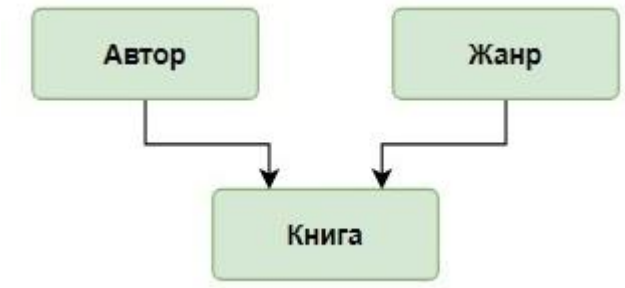
С помощью запросов корректировки данных решим задачу о занесении в базу книг, привезенных на склад поставщиком.

База данных о книгах включает три таблицы **genre**, **author** и book, информация о поставке занесена в таблицу **supply**. С разными типами книг из поставки необходимо выполнить разные действия:

- для книг, которые уже есть на складе по той же цене, что и в поставке, - увеличить их количество на значение, указанное в поставке (пример);
- для книг, которые уже есть на складе, но цена книги в поставке отличается, - увеличить количество экземпляров и вычислить новую цену, при расчете учесть количество имеющихся и новых экземпляров книг (задание);
- для книг, которых на складе нет, - проверить, есть ли автор книги в базе, если нет - занести фамилию автора, а потом добавить новую запись о книге, оставив поле для описания жанра пустым;
- задать жанр для новых книг.
- Еще одно типовое действие на складе - удаление устаревшей информации. С помощью запросов корректировки удаление данных о жанрах, авторах и книгах выполняется в зависимости от того, какие свойства внешних ключей были указаны в таблицах при их создании:
- каскадное удаление записей связанных таблиц;
- удаление записей в главной таблице с сохранением записей в зависимой;
- удаление записей с использованием информации из связанных таблиц.

Структура и наполнение таблиц

Концептуальная схема базы данных:



Логическая схема базы данных:

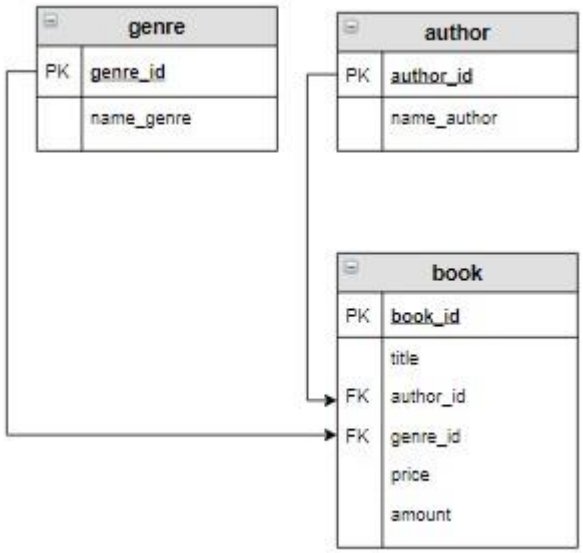


Таблица **author**(создание, заполнение):

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, заполнение, рассмотрено в качестве примеров):

genre_id	name_genre
1	Роман
2	Поэзия
3	Приключения

Таблица **book** (создание, заполнение):

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 supply(создание, заполнение):

supply_id	Title	author	price	amount
1	Доктор Живаго	Пастернак Б.Л.	380.80	4
2	Черный человек	Есенин С.А.	570.20	6
3	Белая гвардия	Булгаков М.А.	540.50	7
4	Идиот	Достоевский Ф.М.	360.80	3
5	Стихотворения и поэмы	Лермонтов М.Ю.	255.90	4
6	Остров сокровищ	Стивенсон Р.Л.	599.99	5

Шаг 2: Запросы на обновление, связанные таблицы (**UPDATE… JOIN…**)

Ссылка в интернете

В запросах на обновление можно использовать связанные таблицы:

```
UPDATE таблица_1
    ... JOIN таблица_2
    ON выражение
    ...
SET ...
WHERE ...;
```

При этом исправлять данные можно во всех используемых в запросе таблицах.

Пример

Для книг, которые уже есть на складе (в таблице book) по той же цене, что и в поставке (**supply**), увеличить количество на значение, указанное в поставке, а также обнулить количество этих книг в поставке.

Этот запрос должен отобрать строки из таблиц booki **supply** такие, что у них совпадают и автор, и название книги. Но в таблице **supply** фамилия автора записана не числом (**id**), а текстом. Следовательно, чтобы выполнить сравнение по фамилии автора нужно "подтянуть" таблицу author, которая связана с bookпо столбцу **author_id**. И в логическом выражении, описывающем соединение таблиц, можно будет использовать столбцы из таблиц book, authoru **supply**.

Запрос:

```
UPDATE book
    INNER JOIN author
    on author.author_id = book.author_id
          INNER JOIN supply
          on book.title = supply.title
            and supply.author = author.name_author
SET book.amount = book.amount + supply.amount,
   supply.amount = 0
WHERE book.price = supply.price;
SELECT * FROM book;
SELECT * FROM supply;
```

Результат:

Affected rows: 4

Query result:

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	12
3	Идиот	2	1	460.00	10
4	Братья Карамазовы	2	1	799.01	3
5	Игрок	2	1	480.50	10
6	Стихотворения и поэмы	3	2	650.00	15
7	Черный человек	3	2	570.20	12
8	Лирика	4	2	518.99	2

Affected rows: 8

Query result:

supply_id	title	author	price	amount
1	Доктор Живаго	Пастернак Б.Л.	380.80	4
2	Черный человек	Есенин С.А.	570.20	0
3	Белая гвардия	Булгаков М.А.	540.50	0
4	Идиот	Достоевский Ф.М.	360.80	3
5	Стихотворения и поэмы	Лермонтов М.Ю.	255.90	4
6	Остров сокровищ	Стивенсон Р.Л.	599.99	5

Под нужное нам условие подходят две книги «Белая гвардия» Булгакова и «Черный человек» Есенина. В таблице **book** их количество увеличилось, а в таблице **supply** - обнулилось.

Задание (Добавить книги и пересчитать цену)

Для книг, которые уже есть на складе (в таблице book), но по другой цене, чем в поставке (**supply**), необходимо в таблице book увеличить количество на значение, указанное в поставке, и пересчитать цену. А в таблице **supply** обнулить количество этих книг. Формула для пересчета цены:

$$price = \frac{(p_1 * k_1 + p_2 * k_2)}{k_1 + k_2}$$

где p_1, p_2 - цена книги в таблицах book и **supply**;

k_1, k_2 - количество книг в таблицах book и **supply**.

Результат:

Affected rows: 2

Query result:

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	437.11	13
4	Братья Карамазовы	2	1	799.01	3
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

Query result:

supply_id	title	author	price	amount
1	Доктор Живаго	Пастернак Б.Л.	380.80	4
2	Черный человек	Есенин С.А.	570.20	6
3	Белая гвардия	Булгаков М.А.	540.50	7
4	Идиот	Достоевский Ф.М.	360.80	0
5	Стихотворения и поэмы	Лермонтов М.Ю.	255.90	4
6	Остров сокровищ	Стивенсон Р.Л.	599.99	5

Пояснение. Пересчитаться должна цена только одной книги Достоевского «Идиот», для этой же книги увеличится количество в таблице **book** и обнулится количество в таблице **supply**.

Шаг 3: Запросы на добавление, связанные таблицы (INSERT INTO… SELECT)

Ссылка в интернете

Запросом на добавление можно добавить записи, отобранные с помощью запроса на выборку, который включает несколько таблиц:

```
INSERT INTO таблица (список_полей)
SELECT список_полей_из_других_таблиц
    FROM таблица_1 ... JOIN таблица_2
    ON ...
...
```

Пример

В таблице **supply** есть новые книги, которых на складе еще не было. Прежде чем добавлять их в таблицу **book**, необходимо из таблицы **supply** отобрать новых авторов, если таковые имеются.

Запрос:

```
SELECT name_author, supply.author
FROM author RIGHT JOIN supply
    on author.name_author = supply.author;
```

Поскольку таблица **author** и поле в таблице **supply** называются одинаково, необходимо указывать полную ссылку на поле (**supply.author**).

Результат:

```
+------------------+------------------+
| name_author      | author           |
+------------------+------------------+
| Булгаков М.А.    | Булгаков М.А.    |
| Достоевский Ф.М. | Достоевский Ф.М. |
| Есенин С.А.      | Есенин С.А.      |
| Пастернак Б.Л.   | Пастернак Б.Л.   |
| Лермонтов М.Ю.   | Лермонтов М.Ю.   |
| None             | Стивенсон Р.Л.   |
+------------------+------------------+
```

Выполнив правое внутреннее соединение таблиц, получили значение Null (None) в поле **name_author** в строке того автора, которого нет в таблице **book**, в нашем случае это Стивенсон.

Теперь достаточно в запросе задать условие отбора, и список новых авторов готов для включения в таблицу **author**.

Запрос:

```
SELECT supply.author
FROM author RIGHT JOIN supply
    on author.name_author = supply.author
WHERE name_author IS Null;
```

Результат:

```
+----------------+
| author         |
+----------------+
| Стивенсон Р.Л. |
+----------------+
```

Задание (Включить новых авторов –запрос на добавление)

Включить новых авторов в таблицу **author** с помощью запроса на добавление, а затем вывести все данные из таблицы **author**. Новыми считаются авторы, которые есть в таблице **supply**, но нет в таблице **author**.

Результат:

Affected rows: 1

Query result:

```
+-----------+------------------+
| author_id | name_author      |
+-----------+------------------+
| 1         | Булгаков М.А.    |
| 2         | Достоевский Ф.М. |
| 3         | Есенин С.А.      |
| 4         | Пастернак Б.Л.   |
| 5         | Лермонтов М.Ю.   |
| 6         | Стивенсон Р.Л.   |
+-----------+------------------+
```

Комментарии учащихся:

1.

Лариса Фернандес

В запросе примера, насколько я понимаю, можно было бы указать

```
SELECT supply.author
FROM supply
WHERE supply.author NOT IN (
                            SELECT name_author
                            FROM author)
```

Если это верно, какой запрос предпочтительней по производительности - соединение таблиц или отбор в WHERE?

Галина Озерова

@Лариса_Фернандес, JOIN выполняется раньше WHERE, поэтому соединение предпочтительнее, как мне кажется.

Артём Гришечко

```
Query result:
+----------+------------+-----------------------------------------------------------------+
| Query_ID | Duration   | Query                                                           |
+----------+------------+-----------------------------------------------------------------+
| 1        | 8.275e-05  | SHOW WARNINGS                                                   |
| 2        | 0.0003915  | SELECT supply.author                                            |
|          |            | FROM author RIGHT JOIN supply                                   |
|          |            |      on author.name_author = supply.author                      |
|          |            | WHERE name_author IS Null                                       |
| 3        | 0.00030825 | SELECT supply.author                                            |
|          |            | FROM supply                                                     |
|          |            | WHERE supply.author NOT IN (SELECT name_author FROM author)     |
+----------+------------+-----------------------------------------------------------------+
```

Через join немного медленнее, но не уверен, насколько валидны выводы для таблиц по 5 строк.
Время если что получено так: set profiling=1; show profiles;

Шаг 4: Запрос на добавление, связанные таблицы

Ссылка в интернете

Следующий шаг - добавить новые записи о книгах, которые есть в таблице **supply** и нет в таблице **book**. (В таблицах **supply** и **book** сохранены изменения предыдущих шагов). Поскольку в таблице **supply** не указан жанр книги, оставить его пока пустым (занести значение Null).

Пример

Прежде всего необходимо сформировать запрос с полями, которые соответствуют полям таблицы **book**, так как использовать только таблицу **supply** нельзя - в ней вместо кода автора стоит его фамилия.

Запрос:

```
SELECT title, author_id, price, amount
FROM author INNER JOIN supply
    ON author.name_author = supply.author;
```

Результат:

```
+----------------------+-----------+--------+--------+
| title                | author_id | price  | amount |
+----------------------+-----------+--------+--------+
| Доктор Живаго        | 4         | 380.80 | 4      |
| Черный человек       | 3         | 570.20 | 0      |
| Белая гвардия        | 1         | 540.50 | 0      |
| Идиот                | 2         | 360.80 | 0      |
| Стихотворения и поэмы | 5        | 255.90 | 4      |
| Остров сокровищ      | 6         | 599.99 | 5      |
+----------------------+-----------+--------+--------+
```

Далее необходимо отобрать только новые книги из таблицы **supply**. Как видно из таблицы с результатами запроса, в тех записях, которые нужно добавить, значения столбца **amount** не равны 0 (количество уже учтенных книг обнулены предыдущим запросом). Добавим это условие в запрос.

Запрос:

```
SELECT title, author_id, price, amount
```

```
FROM author INNER JOIN supply
    ON author.name_author = supply.author
WHERE amount <> 0;
```

Результат:

```
+----------------------+-----------+--------+--------+
| title                | author_id | price  | amount |
+----------------------+-----------+--------+--------+
| Доктор Живаго        | 4         | 380.80 | 4      |
| Стихотворения и поэмы | 5         | 255.90 | 4      |
| Остров сокровищ      | 6         | 599.99 | 5      |
+----------------------+-----------+--------+--------+
```

Задание (Добавить новые книги из supply в book)

Добавить новые книги из таблицы **supply** в таблицу **book** на основе сформированного выше запроса. Затем вывести для просмотра таблицу **book**.

Результат:

Affected rows: 3

Query result:

```
+---------+----------------------+-----------+----------+--------+--------+
| 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 | 12     |
| 3       | Идиот                | 2         | 1        | 437.11 | 13     |
| 4       | Братья Карамазовы    | 2         | 1        | 799.01 | 3      |
| 5       | Игрок                | 2         | 1        | 480.50 | 10     |
| 6       | Стихотворения и поэмы | 3         | 2        | 650.00 | 15     |
| 7       | Черный человек       | 3         | 2        | 570.20 | 12     |
| 8       | Лирика               | 4         | 2        | 518.99 | 2      |
| 9       | Доктор Живаго        | 4         | None     | 380.80 | 4      |
| 10      | Стихотворения и поэмы | 5         | None     | 255.90 | 4      |
| 11      | Остров сокровищ      | 6         | None     | 599.99 | 5      |
+---------+----------------------+-----------+----------+--------+--------+
```

Пояснение. Если нужно оставить какое-то поле пустым - его просто не указывают в списке полей таблицы, в которую добавляются записи.

Шаг 5: Запрос на обновление, вложенные запросы

Ссылка в интернете

После того, как новые книги добавлены в таблицу **book**, нужно указать к какому жанру они относятся. Для этого используется запрос на обновление, в котором можно указать значения столбцов из других таблиц, либо использовать вложенные запросы для получения этих значений.

Пример

Задать для книги Пастернака «Доктор Живаго» жанр «Роман».

Если мы знаем код этой книги в таблице **book** (в нашем случае это 9) и код жанра «Роман» в таблице **genre** (это 1), запрос будет очень простым.

Запрос:

```
UPDATE book
SET genre_id = 1
WHERE book_id = 9;
SELECT * FROM book;
```

Результат:

Affected rows: 1

Query result:

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	12
3	Идиот	2	1	437.11	13
4	Братья Карамазовы	2	1	799.01	3
5	Игрок	2	1	480.50	10
6	Стихотворения и поэмы	3	2	650.00	15
7	Черный человек	3	2	570.20	12
8	Лирика	4	2	518.99	2
9	Доктор Живаго	4	1	380.80	4
10	Стихотворения и поэмы	5	None	255.90	4
11	Остров сокровищ	6	None	599.99	5

Более сложным будет запрос, если известно только название жанра (результат будут точно таким же):

Запрос:

```
UPDATE book
SET genre_id = (SELECT genre_id
        FROM genre
        WHERE name_genre = 'Роман')
WHERE book_id = 9;
```

```
SELECT * FROM book;
```

Задание (Занести жанры для книг)

Занести для книги «Стихотворения и поэмы» Лермонтова жанр «Поэзия», а для книги «Остров сокровищ» Стивенсона - «Приключения». (Использовать два запроса).

Результат:

Affected rows: 2

Query result:

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	12
3	Идиот	2	1	437.11	13
4	Братья Карамазовы	2	1	799.01	3
5	Игрок	2	1	480.50	10
6	Стихотворения и поэмы	3	2	650.00	15
7	Черный человек	3	2	570.20	12
8	Лирика	4	2	518.99	2
9	Доктор Живаго	4	1	380.80	4
10	Стихотворения и поэмы	5	2	255.90	4
11	Остров сокровищ	6	3	599.99	5

Шаг 6: Каскадное удаление записей связанных таблиц (**ON DELETE CASCADE**)

Ссылка в интернете

При создании таблицы для внешних ключей с помощью `ON DELETE` устанавливаются опции, которые определяют действия , выполняемые при удалении связанной строки из главной таблицы.

В частности, `ON DELETE CASCADE` автоматически удаляет строки из зависимой таблицы при удалении связанных строк в главной таблице.

В таблице **book** эта опция установлена для поля **author_id**.

Пример

Удалим из таблицы **author** всех авторов, фамилия которых начинается на «Д», а из таблицы **book** - все книги этих авторов.

Запрос:

```
DELETE FROM author
WHERE name_author LIKE "Д%";
SELECT * FROM author;
SELECT * FROM book;
```

Результат:

Affected rows: 1

Query result:

author_id	name_author
1	Булгаков М.А.
3	Есенин С.А.
4	Пастернак Б.Л.
5	Лермонтов М.Ю.
6	Стивенсон Р.Л.

Affected rows: 5

Query result:

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	12
6	Стихотворения и поэмы	3	2	650.00	15
7	Черный человек	3	2	570.20	12
8	Лирика	4	2	518.99	2
9	Доктор Живаго	4	1	380.80	4
10	Стихотворения и поэмы	5	2	255.90	4
11	Остров сокровищ	6	3	599.99	5

Одним запросом удаляются связанные записи из главной и зависимой таблицы. В нашем случае удалился автор Достоевский и все его книги.

Задание (Удалить авторов и книги, количество меньше 20)

Удалить всех авторов и все их книги, общее количество книг которых меньше 20.

Результат:

Affected rows: 4

Query result:

author_id	name_author
2	Достоевский Ф.М.
3	Есенин С.А.

Affected rows: 2

Query result:

book_id	title	author_id	genre_id	price	amount
3	Идиот	2	1	437.11	13
4	Братья Карамазовы	2	1	799.01	3

```
| 5       | Игрок                  | 2         | 1        | 480.50 | 10      |
| 6       | Стихотворения и поэмы  | 3         | 2        | 650.00 | 15      |
| 7       | Черный человек         | 3         | 2        | 570.20 | 12      |
+---------+------------------------+-----------+----------+--------+---------+
```

Affected rows: 5

Пояснение. Для подсчета количества книг каждого автора используйте вложенный запрос.

Шаг 7: Удаление записей главной таблицы с сохранением записей в зависимой

Ссылка в интернете

При создании таблицы для внешних ключей с помощью `ON DELETE` устанавливаются опции, которые определяют действия, выполняемые при удалении связанной строки из главной таблицы.

Если задано `SET NULL`, то при удалении связанной строки из главной таблицы в зависимой, в столбце внешнего ключа, устанавливается значение **`NULL`**. (При этом в столбце внешнего ключа должно быть допустимо значение **`NULL`**)

В таблице **`book`** эта опция установлена на поле **`genre_id`**.

Пример

Удалим из таблицы **`genre`** все жанры, название которых заканчиваются на «я», а в таблице **`book`** - для этих жанров установим значение `Null`.

Запрос:

```
DELETE FROM genre
WHERE name_genre LIKE "%я";
SELECT * FROM genre;
SELECT * FROM book;
```

Результат:

Affected rows: 2

Query result:

```
+----------+------------+
| genre_id | name_genre |
+----------+------------+
| 1        | Роман      |
+----------+------------+
```

Affected rows: 1

Query result:

```
+---------+------------------------+-----------+----------+--------+---------+
| 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	12
3	Идиот	2	1	437.11	13
4	Братья Карамазовы	2	1	799.01	3
5	Игрок	2	1	480.50	10
6	Стихотворения и поэмы	3	None	650.00	15
7	Черный человек	3	None	570.20	12
8	Лирика	4	None	518.99	2
9	Доктор Живаго	4	1	380.80	4
10	Стихотворения и поэмы	5	None	255.90	4
11	Остров сокровищ	6	None	599.99	5

Affected rows: 11

В нашем случае удалились жанры «Поэзия» и «Приключения».

Задание (Удалить все жанры, количество книг меньше 3)

Удалить все жанры, количество различных книг в которых меньше 3. В таблицу **book** для этих жанров установить значение Null (None).

Результат:

Query result:

genre_id	name_genre
1	Роман
2	Поэзия

Affected rows: 2

Query result:

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	12
3	Идиот	2	1	437.11	13
4	Братья Карамазовы	2	1	799.01	3
5	Игрок	2	1	480.50	10
6	Стихотворения и поэмы	3	2	650.00	15
7	Черный человек	3	2	570.20	12
8	Лирика	4	2	518.99	2
9	Доктор Живаго	4	1	380.80	4
10	Стихотворения и поэмы	5	2	255.90	4
11	Остров сокровищ	6	None	599.99	5

Affected rows: 11

Пояснение. Для отбора жанров, количество различных книг которых меньше 3, использовать вложенный запрос.

Комментарии учащихся:

1.
Алексей Чичулин

Решая данное задание, стало очень интересно. Как, с помощью запроса при удалении информации из одной таблицы, отменить каскадное удаление в уже созданной и связанной внешним ключом таблицы?
Алексей Чичулин
@Anonymous_44648884, Что здесь не так?
ALTER TABLE book DROP FOREIGN KEY (author_id) REFERENCES author (author_id) ON DELETE CASCADE;
ALTER TABLE book ADD FOREIGN KEY (author_id) REFERENCES author (author_id) ON DELETE SET NULL;
Галина Озерова
@Anonymous_44648884, Должно работать следующее:
ALTER TABLE book DROP FOREIGN KEY author_id;
Почему не работает на платформе - не знаю
2.
Станислав Гришин
Спасибо за задания и за курс. Однако, здесь, если честно не понял по поводу этого using. Вопросы такие:
1) "можно использовать" - как альтернатива where из предыдущих заданий или есть какие-то критерии когда использовать using и в чем его преимущества или особенности (не раскрыто в материале)
2) Не совсем понятно, как он работает. В данном задании я сделал двойной inner join и потом через where отсеил нужные мне; однако, мне не совсем понятно как это работает - что именно происходит при этих двух пересечениях с помощью using и почему потом при добавлении условия с "Поэзией" она каким-то чудесным образом в конечном итоге удаляет нужных авторов?
Никита Лень
@Станислав_Гришин, если у Вас возникают вопросы по SQL, не стесняйтесь пользоваться гуглом, данный курс является тренажером по SQL. USING является заменой конструкции при использовании JOIN'ов, когда можно вместо talbe1 INNER JOIN table2 ON table1.n_id = table2.n_id написать table1 INNER JOIN table2 USING(n_id)
Галина Озерова
@Станислав_Гришин,
В запросах на удаление данных из некоторой таблицы, если необходима информация из других таблиц, используется USING. Этот оператор позволяет присоединить другие таблицы, чтобы можно было обратиться к их столбцам. Если таблицы перечислены (связаны) после USING, их столбцы можно использовать как в операторах соединения ON, так и после WHERE.

Шаг 8: Удаление записей, использование связанных таблиц (**DELETE FROM…USING…**)

Ссылка в интернете

При удалении записей из таблицы можно использовать информацию из других связанных с ней таблиц. В этом случае синтаксис запроса имеет вид:

```
DELETE FROM таблица_1
USING таблица_1 INNER JOIN таблица_2 ON ...
WHERE ...
```

Пример

Удалить всех авторов из таблицы **author**, у которых есть книги, количество экземпляров которых меньше 3. Из таблицы **book** удалить все книги этих авторов.

Запрос:

```
DELETE FROM author
USING author INNER JOIN book ON author.author_id = book.author_id
```

```
WHERE book.amount < 3;

SELECT * FROM author;

SELECT * FROM book;
```

Результат:

Affected rows: 1

Query result:

```
+-----------+------------------+
| author_id | name_author      |
+-----------+------------------+
| 1         | Булгаков М.А.    |
| 2         | Достоевский Ф.М. |
| 3         | Есенин С.А.      |
| 5         | Лермонтов М.Ю.   |
| 6         | Стивенсон Р.Л.   |
+-----------+------------------+
```

Affected rows: 5

Query result:

```
+---------+----------------------+-----------+----------+--------+--------+
| 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 | 12     |
| 3       | Идиот                | 2         | 1        | 437.11 | 13     |
| 4       | Братья Карамазовы    | 2         | 1        | 799.01 | 3      |
| 5       | Игрок                | 2         | 1        | 480.50 | 10     |
| 6       | Стихотворения и поэмы | 3         | 2        | 650.00 | 15     |
| 7       | Черный человек       | 3         | 2        | 570.20 | 12     |
| 10      | Стихотворения и поэмы | 5         | 2        | 255.90 | 4      |
| 11      | Остров сокровищ      | 6         | 3        | 599.99 | 5      |
+---------+----------------------+-----------+----------+--------+--------+
```

Книги из таблицы **book** будут удалены автоматически, так как для столбца **author_id** из таблицы **book** установлено каскадное удаление записей.

Задание (Удалить авторов, пишут в жанрах «Поэзия»)

Удалить всех авторов, которые пишут в жанре "Поэзия". Из таблицы **book** удалить все книги этих авторов. В запросе для отбора авторов использовать полное название жанра, а не его **id**.

Результат:

Affected rows: 3

Query result:

```
+-----------+------------------+
| author_id | name_author      |
+-----------+------------------+
| 1         | Булгаков М.А.    |
| 2         | Достоевский Ф.М. |
| 6         | Стивенсон Р.Л.   |
```

```
+-----------+------------------+
```
Affected rows: 3

Query result:

```
+---------+--------------------+-----------+----------+--------+--------+
| 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 | 12     |
| 3       | Идиот              | 2         | 1        | 437.11 | 13     |
| 4       | Братья Карамазовы  | 2         | 1        | 799.01 | 3      |
| 5       | Игрок              | 2         | 1        | 480.50 | 10     |
| 11      | Остров сокровищ    | 6         | 3        | 599.99 | 5      |
+---------+--------------------+-----------+----------+--------+--------+
```

Шаг 9: Задание (Придумать запрос корректировки данных для таблиц **book**, **author**, **genre** и **supply**)

Ссылка в интернет

Придумайте один или несколько запросов корректировки данных для таблиц **book**, **author**, **genre** и **supply** (в таблицы занесены данные, как на первом шаге урока). Проверьте, правильно ли они работают.

При желании можно формулировку запросов разместить в комментариях.

Размещенные задания можно использовать для закрепления материала урока.

Оценивайте понравившиеся Вам запросы.

В последнем модуле создан отдельный урок, в котором мы разместим запросы, набравшие наибольшее количество лайков.

УРОК 2.4: База данных «Интернет-магазин книг», запросы на выборку

Шаг 1: Предметная область

Ссылка в интернете

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.

В данном уроке сначала будет построена концептуальная модель базы данных, затем ее логическая модель. Также будут определены структура и содержание таблиц базы данных «Интернет-магазин книг».

Затем для разработанной базы данных рассматриваются следующие запросы:

1. Вывести фамилии всех клиентов, которые заказали определенную книгу.
2. Посчитать, сколько раз была заказана каждая книга.
3. Вывести города, в которых живут клиенты магазина.
4. Вывести информацию об оплате каждого заказа.
5. Вывести подробную информацию о каждом заказе.
6. Вывести информацию о движении каждого заказа.
7. Вывести заказы, доставленные с опозданием.
8. Вывести клиентов, которые заказывали книги определенного автора.
9. Вывести самый популярный жанр.
10. Сравнить ежемесячную выручку за текущий и прошлый год.

Шаг 2: Проектирование концептуальной модели базы данных

Ссылка в интернете

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. 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, автор, жанр, количество, цена;
- **Автор** – фамилия и инициалы;
- **Жанр** – название;
- **Покупатель (клиент)** – фамилия и имя, электронная почта, город;
- **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):

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 (Установить связь между информационными объектами)

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

Комментарии учащихся:

1.

Георгий Лолаев

Так и не дошло почему у заказа может быть несколько этапов. Ведь если один этап пройден, на его место должен встать другой этап. В чем моя ошибка??

Adilbek Kaliyev

@Георгий_Лолаев, мне кажется, потому что мы фиксируем дату/время ("*Фиксируется **дата каждой операции***").

Если мы просто будем менять в заказе название этапа, то непонятно как хранить дату. А так, мы создадим промежуточную таблицу, где будут колонки **Номер заказа, ИД этапа, дата** (еще не смотрел следующие степы, но предполагаю что так будет)

Алексей Стецко

@Георгий_Лолаев, может как-то так сформулировать связь: "Как один этап проходят несколько заказов, так и один заказ может проходить несколько этапов". То есть если заказ доставляется, то к нему относятся этапы "оплата", "упаковка", "передача курьеру". Ну это тоже мои мысли.

Макс Качаев

@Георгий_Лолаев, тоже непонятно почему так должно быть, ведь по сути для каждого заказа одновременно существует лишь один этап(статус если я правильно понял)

Алексей Стецко

@Макс_Качаев, нет, не один. Далее будут таблицы, там всё станет понятно. У заказа может быть несколько этапов, потому что всё **хранится в таблице**. Когда заканчивается один этап, запись не удаляется.

2.

Андрей Синельников

Строго говоря, каждая книга может быть написана больше чем одним автором. Также в зависимости от классификации книга может принадлежать к нескольким жанрам.

Галина Озерова

@Андрей_Синельников, Абсолютно верно, но база данных - это модель предметной области, а модель может отражать только существенные для определенного исследования аспекты. В данном случае такое упрощение - книга имеет одного автора и относится к одному жанру - сделано в учебных целях.

Шаг 3: Построение логической схемы базы данных

Ссылка в интернете

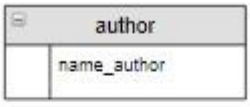
На предыдущем шаге получена концептуальная модель базы данных:

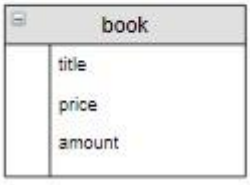
На основе этой модели создается логическая модель, в которой информационные объекты описываются в виде реляционных таблиц.

Для каждой пары таблиц необходимо выполнить следующие шаги:

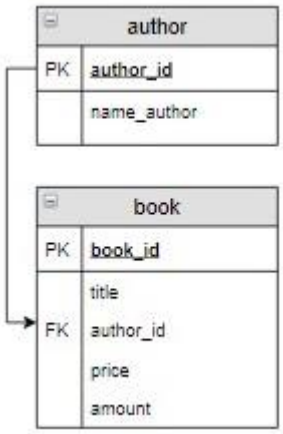
Шаг 1. Выбрать пару таблиц из схемы, например:

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»:



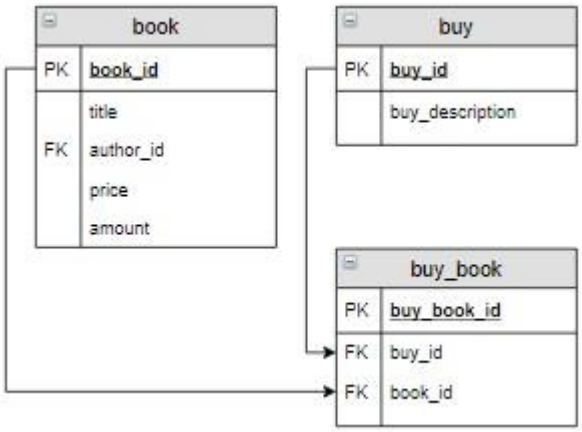
Шаг 4. Вернуться к описанию концептуальной модели и проверить, нужно ли включить какие-то характеристики, непривязанные к информационным объектам? В нашем случае ничего добавлять не надо. Чаще всего характеристики добавляются при реализации связи «многие ко многим».

Продолжим построение логической схемы:

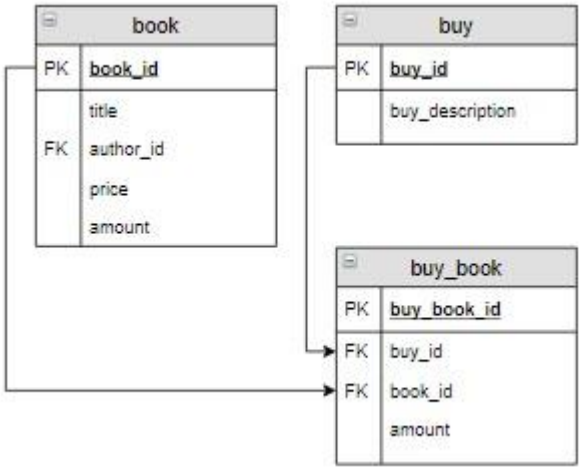
Шаг 1. Выберем следующую пару таблиц:

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

Шаг 3. Реализуем связь «многие ко многим»:



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.

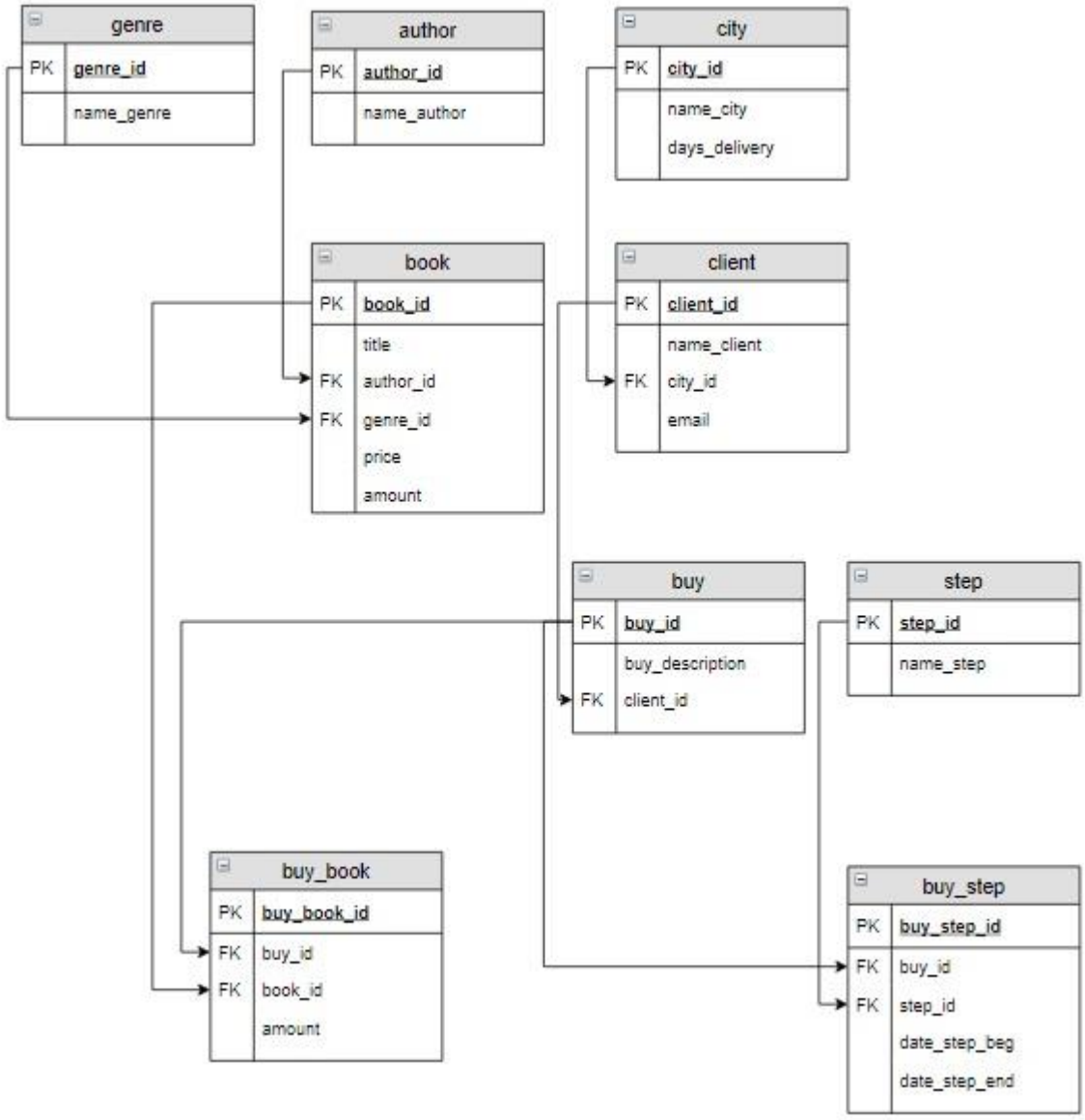
Задание (Сопоставить фрагменты концептуальной и логической моделей)

Сопоставьте фрагменты концептуальной модели с фрагментами логической модели.

Шаг 4: Создание базы данных

Ссылка в интернете

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, заполнение):

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, заполнение, рассмотрено в качестве примеров):

genre_id	name_genre
1	Роман
2	Поэзия
3	Приключения

Таблица **book** (создание, заполнение):

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	Семенов 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	2	8	2
4	3	3	2
5	3	2	1

6	3	1	1
7	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		

Шаг 5: Запросы на основе трех и более связанных таблиц

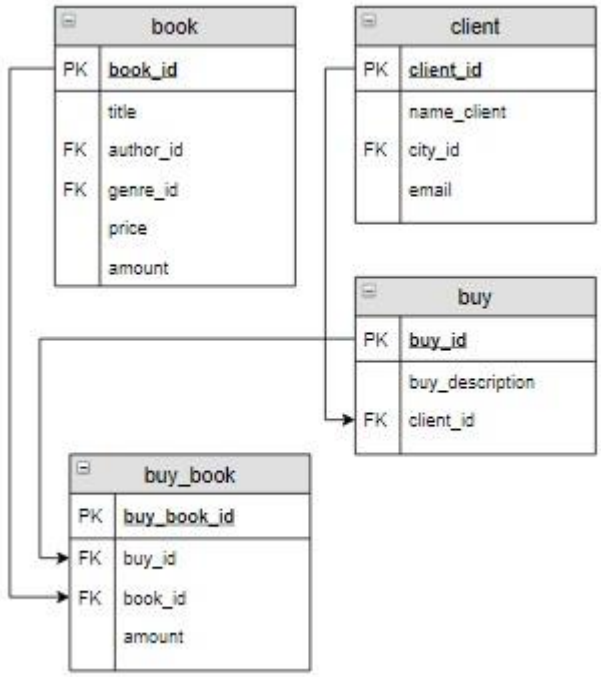
Ссылка в интернете

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**.

```
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';
```

В запросе отбираются различные 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)

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

Результат:

```
+--------+--------------------+--------+--------+
| buy_id | title              | price  | amount |
+--------+--------------------+--------+--------+
| 1      | Мастер и Маргарита | 670.99 | 1      |
| 1      | Черный человек     | 570.20 | 2      |
| 4      | Игрок              | 480.50 | 1      |
+--------+--------------------+--------+--------+
```

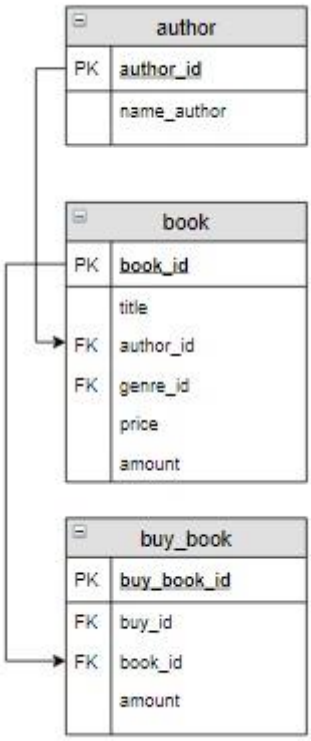
Пояснение. Если в нескольких таблицах столбцы называются одинаково – необходимо явно указывать из какой таблицы берется столбец. Например, столбец **amount** есть и в таблице **book**, и в таблице **buy_book**. В запросе нужно указать количество заказанных книг, то есть **buy_book.amount**.

Шаг 6: Задание (Посчитать, сколько раз была заказана каждая книга)

Ссылка в интернете

Посчитать, сколько раз была заказана каждая книга, для книги вывести ее автора (нужно посчитать, в каком количестве заказов фигурирует каждая книга). Результат отсортировать сначала по фамилиям авторов, а потом по названиям книг.

Фрагмент логической схемы базы данных:



Результат:

name_author	title	Количество
Булгаков М.А.	Белая гвардия	1
Булгаков М.А.	Мастер и Маргарита	2
Достоевский Ф.М.	Братья Карамазовы	0
Достоевский Ф.М.	Игрок	1
Достоевский Ф.М.	Идиот	1
Есенин С.А.	Стихотворения и поэмы	0
Есенин С.А.	Черный человек	1
Пастернак Б.Л.	Лирика	1

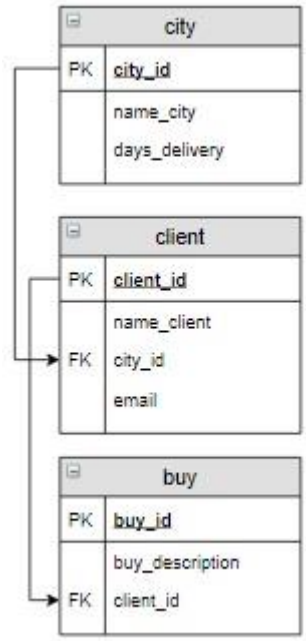
Пояснение. Для того, чтобы были выведены книги, которые клиенты не заказывали, использовать внешнее соединение.

Шаг 7: Задание (Вывести города, в которых живут клиенты, оформлявшие заказы в интернет-магазине)

Ссылка в интернете

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

Фрагмент логической схемы базы данных:



Результат:

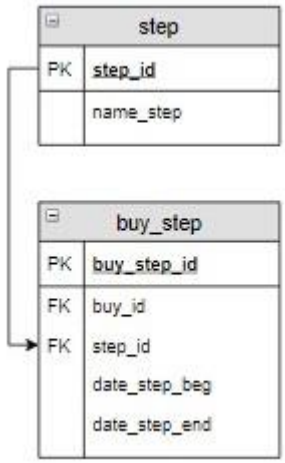
```
+-----------------+------------+
| name_city       | Количество |
+-----------------+------------+
| Владивосток     | 2          |
| Москва          | 1          |
| Санкт-Петербург | 1          |
+-----------------+------------+
```

Шаг 8: Задание (Вывести номера всех оплаченных заказов и даты)

Ссылка в интернете

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

Фрагмент логической схемы базы данных:

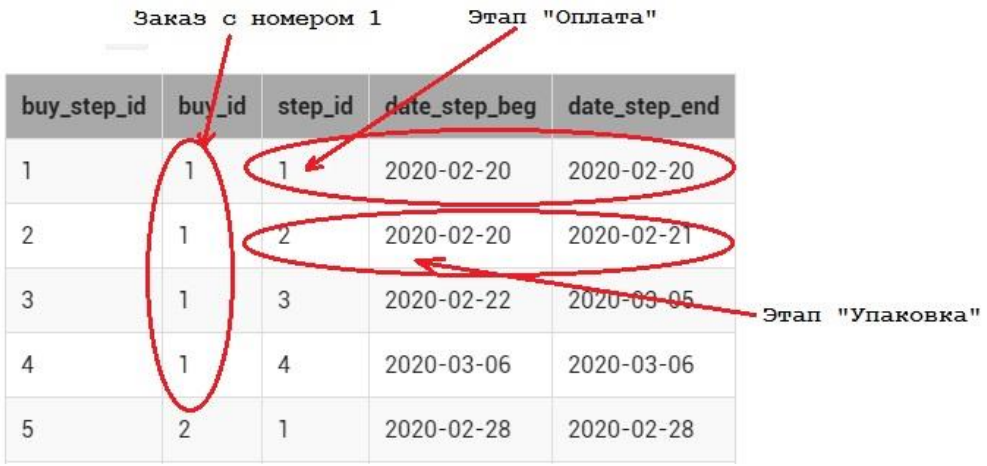


Результат:

```
+--------+---------------+
| buy_id | date_step_end |
```

```
+--------+--------------+
| 1      | 2020-02-20   |
| 2      | 2020-02-28   |
| 3      | 2020-03-05   |
+--------+--------------+
```

Пояснение. С каждым заказом в таблице **buy_step** связаны 4 записи, которые фиксируют этапы заказа.



buy_step_id	buy_id	step_id	date_step_beg	date_step_end
1	1	1	2020-02-20	2020-02-20
2	1	2	2020-02-20	2020-02-21
3	1	3	2020-02-22	2020-03-05
4	1	4	2020-03-06	2020-03-06
5	2	1	2020-02-28	2020-02-28

Для каждого заказа сначала выставляется счет на оплату (в запись с **step_id** со значением 1 («Оплата») в столбец **date_step_beg** заносится дата выставления счета по заказу). После того, как счет оплачен, в столбец **date_step_end** той же записи заносится дата оплаты заказа.

Затем в таблице **buy_step** заполняется **step_id** со значением 2 («Упаковка») для текущего заказа: после передачи заказа на упаковку заполняется поле **date_step_beg**, а после окончания упаковки – поле **date_step_end**. И так далее для оставшихся двух шагов («Транспортировка» и «Доставка»).

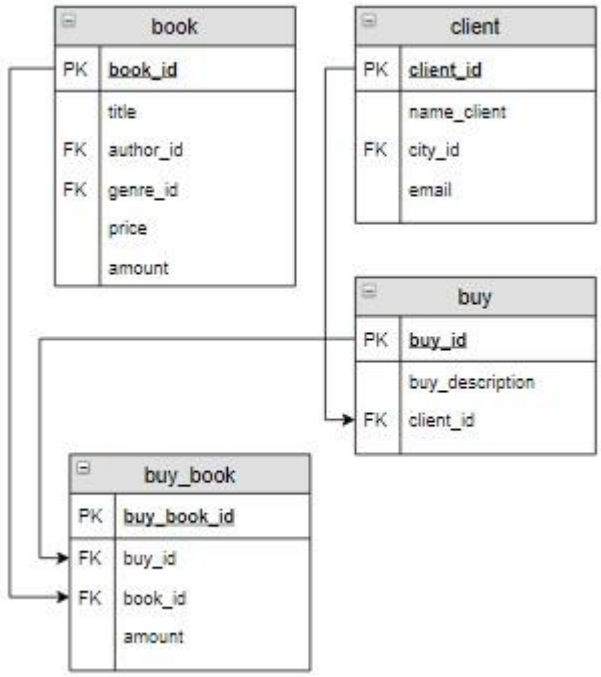
Для реализации запроса учитывать тот факт, что те заказы, которые не оплачены в таблице **buy_step** в записи с **step_id** со значением 1 («Оплата») в столбце **date_step_end** имеют значение Null.

Шаг 9: Задание (Вывести информацию о каждом заказе)

Ссылка в интернете

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.

Фрагмент логической схемы базы данных:



Результат:

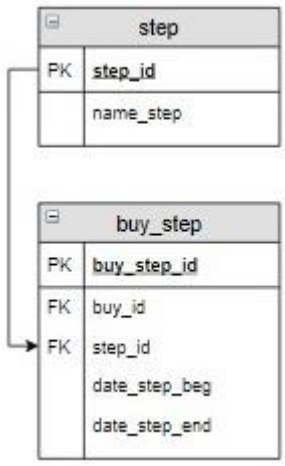
```
+--------+----------------+-----------+
| buy_id | name_client    | Стоимость |
+--------+----------------+-----------+
| 1      | Баранов Павел  | 1811.39   |
| 2      | Семенонов Иван | 1037.98   |
| 3      | Абрамова Катя  | 2131.49   |
| 4      | Баранов Павел  | 480.50    |
+--------+----------------+-----------+
```

Шаг 10: Задание (Вывести все заказы и названия этапов)

Ссылка в интернете

Вывести все заказы и названия этапов, на которых они в данный момент находятся. Если заказ доставлен – информацию о нем не выводить. Информацию отсортировать по возрастанию **buy_id**.

Фрагмент логической схемы базы данных:



Результат:

buy_id	name_step
2	Транспортировка
3	Доставка
4	Оплата

Шаг 11: Задание (Вывести количество дней за которое заказ реально доставлен в город)

Ссылка в интернете

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. Информацию вывести в отсортированном по номеру заказа виде.

Фрагмент логической схемы базы данных:

Результат:

buy_id	Количество_дней	Опоздание
1	14	2

```
| 3       | 4                  | 0          |
+--------+-----------------+-----------+
```

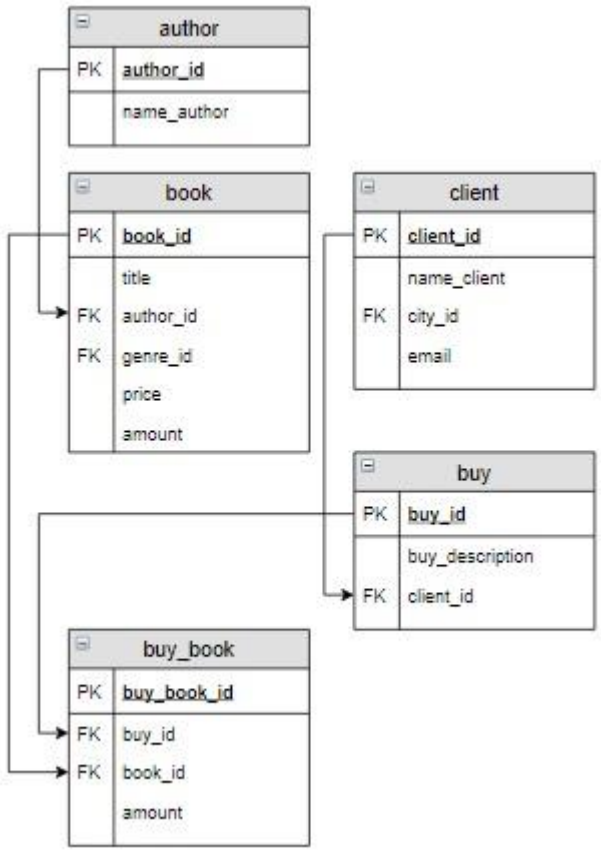
Пояснение. Для вычисления поля «Опоздание» используйте функцию `if()`, а для вычисления разности дат – функцию `DATEDIFF()`.

Шаг 12: Задание (Выбрать всех клиентов, которые заказывали книги Достоевского)

Ссылка в интернете

Выбрать всех клиентов, которые заказывали книги Достоевского, информацию вывести в отсортированном по алфавиту виде.

Фрагмент логической схемы:



Результат:

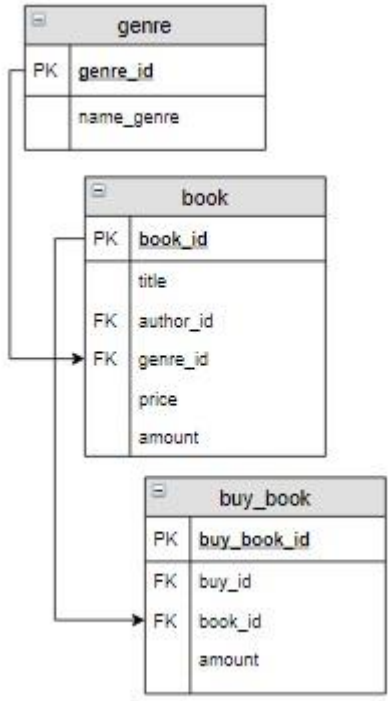
```
+---------------+
| name_client   |
+---------------+
| Абрамова Катя |
| Баранов Павел |
+---------------+
```

Шаг 13: Задание (Вывести жанр, который больше всего заказывали)

Ссылка в интернете

Вывести жанр (или жанры), в котором было заказано больше всего экземпляров книг, указать это количество.

Фрагмент логической схемы:



Результат:

```
+------------+------------+
| name_genre | Количество |
+------------+------------+
| Роман      | 6          |
+------------+------------+
```

Пояснение. Использовать вложенный запрос для вычисления максимального значения экземпляров книг. Рекомендуется запрос реализовывать по шагам.

Шаг 14: Оператор **UNION**

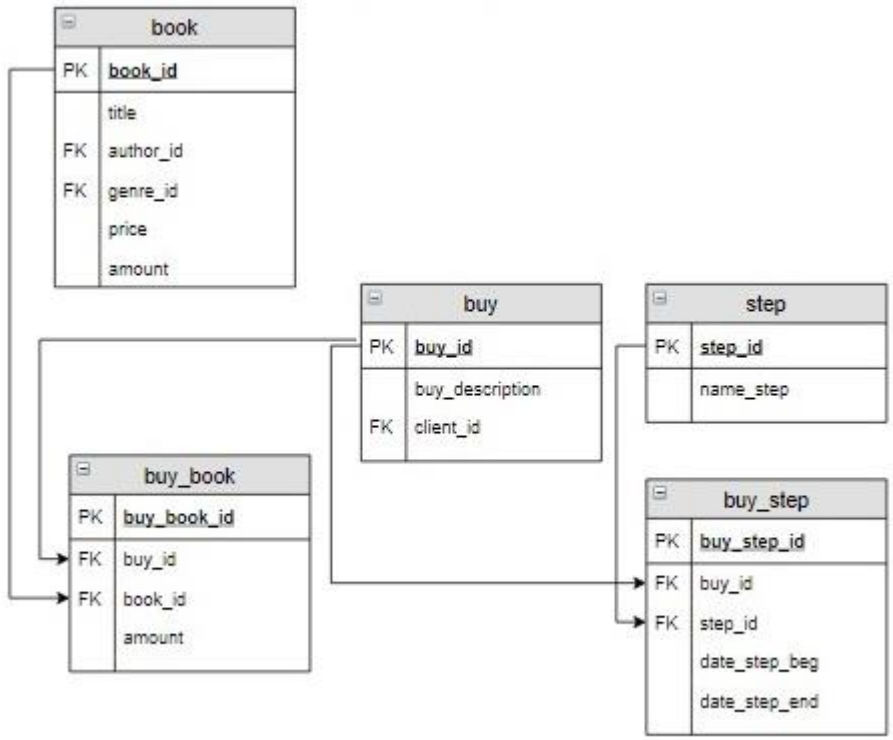
Ссылка в интернете

Задание (Сравнить выручку от продаж по годам)

Этот шаг добавлен по рекомендациям пользователей: Тимур Timmmyyy, Todor Illia, Лёха Last name, Игорь Владимирович Лапшин 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:



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**

Пояснение

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 информацию об оплаченных заказах за предыдущий и текущий год, информацию отсортировать по **id_client**.

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	1	2020-02-20	1	670.99
1	1	7	2020-02-20	2	570.20
2	3	8	2020-02-28	2	518.99
3	2	3	2020-03-05	2	460.00
3	2	2	2020-03-05	1	540.50
3	2	1	2020-03-05	1	670.99

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
4	1	6	2019-03-12	1	650.00
2	1	1	2019-02-21	2	670.60
2	1	3	2019-02-21	1	450.90
1	1	1	2020-02-20	1	670.99
1	1	7	2020-02-20	2	570.20
3	2	3	2020-03-05	2	460.00
3	2	2	2020-03-05	1	540.50
3	2	1	2020-03-05	1	670.99
5	2	1	2019-03-18	2	670.60
5	2	4	2019-03-18	1	780.90

```
| 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 |
| 2       | 3          | 8       | 2020-02-28   | 2       | 518.99 |
| 3       | 4          | 4       | 2019-03-05   | 4       | 780.90 |
| 3       | 4          | 5       | 2019-03-05   | 2       | 480.90 |
+--------+-----------+---------+--------------+--------+--------+
```

Если же необходимо выполнить сортировку в каждом запросе, входящем в **UNION**, необходимо каждый запрос заключить в круглые скобки.

Результат:

```
+------+----------+---------+
| Год  | Месяц    | Сумма   |
+------+----------+---------+
| 2019 | February | 5626.30 |
| 2020 | February | 2849.37 |
| 2019 | March    | 6857.50 |
| 2020 | March    | 2131.49 |
+------+----------+---------+
```

Пояснение.

1. Ежемесячная выручка рассчитывается как сумма произведений цены книги на заказанное пользователем в этом месяце количество.
2. Цена книги для текущего года хранится в таблице **book**, а для предыдущего в **buy_archive**.
3. Функция для для выделения месяца рассмотрена на этом шаге.

Комментарии учащихся:

1.

Дмитрий Азовцев

чем union отличается от union all, если результаты одинаковы ?

Анастасия Арсентьева

@Дмитрий_Азовцев, Если указано ключевое слово ALL, то в результат включаются все записи запросов, в противном случае - различные. В данной задаче результаты совпали.

Анастасия Будлянская

@Дмитрий_Азовцев, тоже не совсем поняла сначала, помогли разобраться следующие источники:

1. http://www.sql-tutorial.ru/ru/book_union.html

Предложение UNION приводит к появлению в результирующем наборе всех строк каждого из запросов. При этом, если определен параметр ALL, то сохраняются все дубликаты выходных строк, в противном случае в результирующем наборе присутствуют только уникальные строки. Заметим, что можно связывать вместе любое число запросов. Кроме того, с помощью скобок можно задавать порядок объединения.

2. https://info-comp.ru/obucheniest/340-sql-union-and-union-all.html

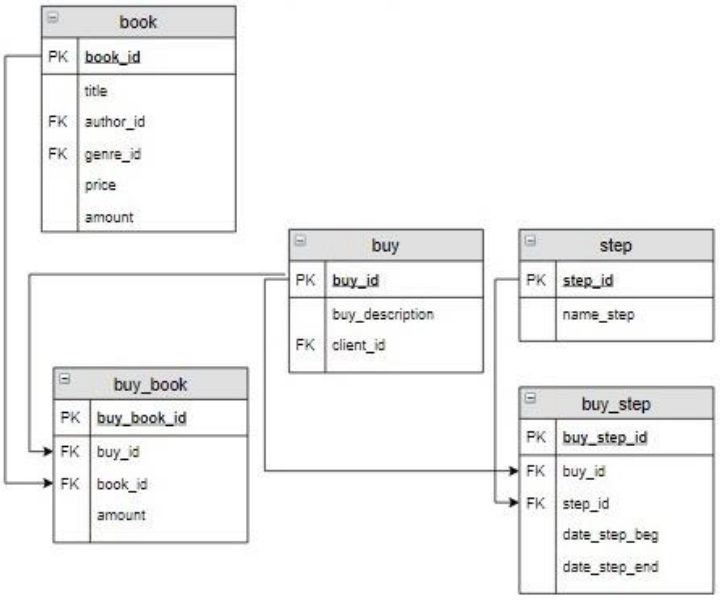
- **UNION** – это оператор SQL для объединения результирующего набора данных нескольких запросов, и данный оператор выводит только уникальные строки в запросах, т.е. например, Вы объединяете два запроса и в каждом из которых есть одинаковые данные, другими словами полностью идентичные, и оператор union объединит их в одну строку для того чтобы не было дублей;
- **UNION ALL** – это оператор SQL для объединения результирующего набора данных нескольких запросов, а вот данный оператор, выведет уже абсолютно все строки, даже дубли.

Шаг 15: Задание (Вычислить общее количество проданных книг по годам)

Ссылка в интернете

Вычислить общее количество проданных книг (столбец **Количество**) и их стоимость (столбец **Сумма**) за текущий и предыдущий год. Информацию отсортировать по убыванию стоимости.

Фрагмент логической схемы базы данных:



Информация о продажах прошлого года хранится в таблице **buy_archive** следующей структуры:

Пояснение

Запросы с **UNION** можно использовать как вложенные, это позволяет обрабатывать данные из объединенных запросов совместно.

Пример

Вывести клиентов, которые делали покупки в прошлом году, но не делали в этом. А также новых клиентов, которые делали заказы только в текущем году. Информацию отсортировать по возрастанию лет.

Шаг 1. Отберем клиентов прошлого года, указав дату самого раннего заказа, а также клиентов этого года, указав для них самую раннюю дату оплаты заказа.

Запрос

```
SELECT name_client, MIN(date_payment) AS first_payment
 FROM buy_archive INNER JOIN client USING(client_id)
 GROUP BY  name_client
UNION
 SELECT name_client, MIN(date_step_end)
```

```
 FROM buy INNER JOIN client USING(client_id)
 INNER JOIN buy_step USING (buy_id)
 GROUP BY name_client
```

Результат:

```
+-----------------+---------------+
| name_client     | first_payment |
+-----------------+---------------+
| Абрамова Катя   | 2019-02-10    |
| Баранов Павел   | 2019-02-21    |
| Яковлева Галина | 2019-03-05    |
| Абрамова Катя   | 2020-03-05    |
| Баранов Павел   | 2020-02-20    |
| Семенонов Иван  | 2020-02-28    |
+-----------------+---------------+
```

Как видно из таблицы, некоторые клиенты делали покупки как в прошлом, так и в этом году. Они встречаются в таблице 2 раза.

Шаг 2. Оставим только тех клиентов, которые встречаются в полученной таблице один раз, для этого используем предыдущий запрос как вложенный.

Запрос:

```
SELECT name_client, MIN(YEAR(first_payment)) AS Год
FROM
 (SELECT name_client, MIN(date_payment) AS first_payment
   FROM buy_archive INNER JOIN client USING(client_id)
   GROUP BY  name_client
  UNION
  SELECT name_client, MIN(date_step_end)
   FROM buy INNER JOIN client USING(client_id)
   INNER JOIN buy_step USING (buy_id)
   GROUP BY name_client) query_in
GROUP BY name_client
HAVING COUNT(*) = 1
ORDER BY 2
```

Результат:

```
+-----------------+------+
| name_client     | Год  |
+-----------------+------+
| Яковлева Галина | 2019 |
| Семенонов Иван  | 2020 |
+-----------------+------+
```

Результат:

title	Количество	Сумма
Братья Карамазовы	8	6247.20
Мастер и Маргарита	6	4024.38
Идиот	4	1821.80
Белая гвардия	3	1581.10
Черный человек	2	1140.40
Лирика	2	1037.98
Игрок	2	961.80
Стихотворения и поэмы	1	650.00

Пояснение. При вычислении Количества и Суммы для текущего года учитывать только те книги, которые уже оплачены (указана дата оплаты для шага "Оплата" в таблице buy_step).

Шаг 16: Задание (Придумать запрос на выборку из предметной области)

Ссылка в интернете

Придумайте один или несколько запросов на выборку для предметной области «Интернет-магазин книг» (в таблицы занесены данные, как на первом шаге урока). Проверьте, правильно ли они работают.

При желании можно формулировку запросов разместить в комментариях.

Размещенные задания можно использовать для закрепления материала урока.

Оценивайте понравившиеся Вам запросы.

В последнем модуле создан отдельный урок, в котором мы разместим запросы, набравшие наибольшее количество лайков.

УРОК 2.5: База данных «Интернет-магазин книг», запросы корректировки

Шаг 1: Содержание урока

Ссылка в интернете

С помощью запросов корректировки данных можно выполнить следующие действия по обработке заказов в интернет-магазине:

- включение нового клиента в базу данных;
- формирование нового заказа некоторым пользователем;
- включение в заказ одной или нескольких книг с указанием их количества;
- уменьшение количества книг на складе;
- создание счета на оплату (полный счет, итоговый счет);
- добавление этапов продвижения заказа;
- фиксация дат прохождения каждого этапа заказа (начало этапа, завершение этапа).

Предметная область, логическая структура базы данных, содержание таблиц – как на предыдущем уроке.

Шаг 2: Задание (Добавить нового клиента)

Ссылка в интернете

Включить нового человека в таблицу с клиентами. Его имя **Попов Илья**, его email **popov@test**, проживает он в **Москве**.

Фрагмент логической схемы базы данных:

Результат:

Affected rows: 1

Query result:

```
+-----------+-----------------+---------+----------------+
| client_id | name_client     | city_id | email          |
+-----------+-----------------+---------+----------------+
| 1         | Баранов Павел   | 3       | baranov@test   |
| 2         | Абрамова Катя   | 1       | abramova@test  |
| 3         | Семенонов Иван  | 2       | semenov@test   |
| 4         | Яковлева Галина | 1       | yakovleva@test |
| 5         | Попов Илья      | 1       | popov@test     |
+-----------+-----------------+---------+----------------+
```

Пояснение. В запросах на добавление можно одновременно заносить и константы, и данные из других таблиц. Для этого в той части запроса INSERT, где задается запрос на выборку, в качестве полей для вставки указываются не только поля других таблиц, но и константы:

```
INSERT INTO ...
SELECT 'Попов Илья', city_id, 'popov@test'
FROM city
WHERE ...;
```

Для просмотра той таблицы, в которую внесены изменения, используйте запрос вида:

```
SELECT * FROM таблица;
```

Шаг 3: Задание (Создать новый заказ для Попова Ильи)

Ссылка в интернете

Создать новый заказ для Попова Ильи. Его комментарий для заказа: «Связаться со мной по вопросу доставки».

Фрагмент логической схемы базы данных:

Результат:

Affected rows: 1

Query result:

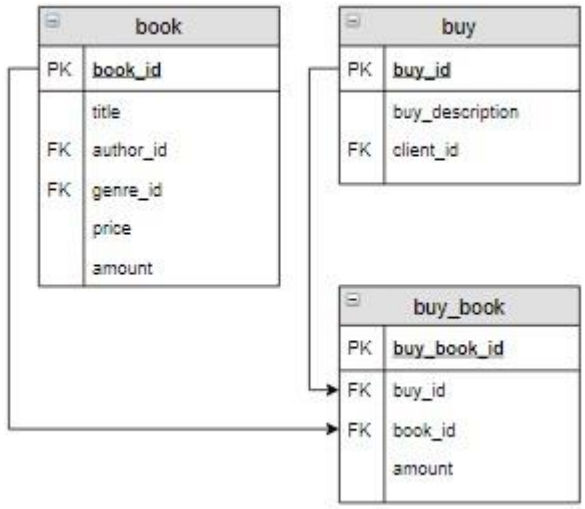
buy_id	buy_description	client_id
1	Доставка только вечером	1
2	None	3
3	Упаковать каждую книгу по отдельности	2
4	None	1
5	Связаться со мной по вопросу доставки	5

Шаг 4: Задание (Добавить заказ с номером 5)

Ссылка в интернете

В таблицу **buy_book** добавить заказ с номером 5. Этот заказ должен содержать книгу Пастернака «Лирика» в количестве двух экземпляров и книгу Булгакова «Белая гвардия» в одном экземпляре.

Фрагмент логической схемы базы данных:



Результат:

buy_book_id	buy_id	book_id	amount
1	1	1	1
2	1	7	2
3	2	8	2
4	3	3	2
5	3	2	1
6	3	1	1
7	4	5	1
8	5	8	2
9	5	2	1

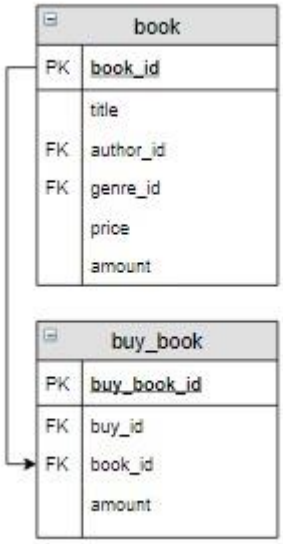
Пояснение. Для вставки каждой книги используйте отдельный запрос. Не забывайте между запросами ставить точку с запятой.

Шаг 5: Задание (Уменьшить количество книг на складе)

Ссылка в интернете

Количество тех книг на складе, которые были включены в заказ с номером 5, уменьшить на то количество, которое в заказе с номером 5 указано.

Фрагмент логической схемы базы данных:



Результат:

Affected rows: 2

Query result:

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	4
3	Идиот	2	1	460.00	10
4	Братья Карамазовы	2	1	799.01	3
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	0

Пояснение. Для изменения количества книг используйте запрос `UPDATE`.

Комментарии учащихся:

1.

Yury Popov

В MySQL можно написать

```
UPDATE table1
JOIN table2
ON ...
SET ...
WHERE (Условия на сджойненную таблицу)
```

и он обновит `table1` соответственно, то есть, не нужно `table1 JOIN table2...` как вторую таблицу в `UPDATE` подзапросом задавать (вот тут про другие СУБД: https://stackoverflow.com/questions/1293330/how-can-i-do-an-update-statement-with-join-in-sql-server). Я как-то прошляпил момент, в котором это объяснялось в курсе %) Но вообще эту задачу можно и без джойнов решить.

Галина Озерова

@Yury_Popov, Можно, но с JOIN запросы более эффективные, поскольку сначала выполняются операторы соединения, а потом условие после WHERE для уже отобранных операторами соединения данными. Если все условия поместить в WHERE, то они будут выполнятся над всеми данными из всех указанных таблиц.

Валерий Родькин

@Yury_Popov, лучше так

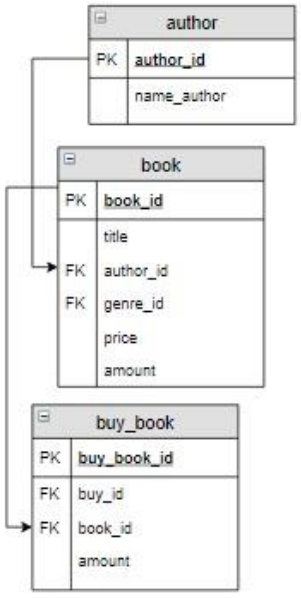
```
UPDATE table1
JOIN table2
USING ...
SET ...
WHERE (Условия на сджойненную таблицу) ,
```

Шаг 6: Задание (Создать счет на оплату заказа)

Ссылка в интернете

Создать счет (таблицу **buy_pay**) на оплату заказа с номером 5, в который включить название книг, их автора, цену, количество заказанных книг и стоимость. Информацию вывести в отсортированном по названиям книг виде.

Фрагмент логической схемы базы данных:



Результат:

Affected rows: 2

Query result:

```
+--------------+----------------+--------+--------+-----------+
| title        | name_author    | price  | amount | Стоимость |
+--------------+----------------+--------+--------+-----------+
| Белая гвардия | Булгаков М.А.  | 540.50 | 1      | 540.50    |
```

```
| Лирика          | Пастернак Б.Л. | 518.99 | 2       | 1037.98    |
+-----------------+----------------+--------+---------+------------+
```

Пояснение. Для создания таблицы используйте запрос CREATE.

Шаг 7: Задание (Создать общий счёт)

Ссылка в интернете

Создать общий счет (таблицу **buy_pay**) на оплату заказа с номером 5. Куда включить номер заказа, количество книг в заказе и его общую стоимость.

Фрагмент логической схемы базы данных:

Результат:

Affected rows: 1

Query result:

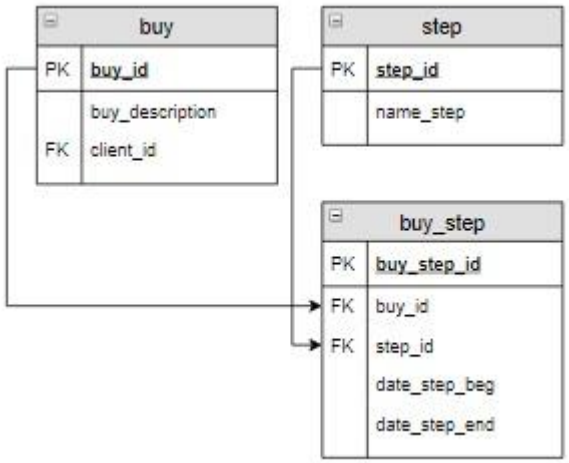
```
+--------+------------+---------+
| buy_id | Количество | Итого   |
+--------+------------+---------+
| 5      | 3          | 1578.48 |
+--------+------------+---------+
```

Шаг 8: Задание (Включить все этапы заказа)

Ссылка в интернете

В таблицу **buy_step** для заказа с номером 5 включить все этапы из таблицы **step**, которые должен пройти этот заказ. В столбцы **date_step_beg** и **date_step_end** всех записей занести Null.

Фрагмент логической схемы базы данных:



Результат:

Affected rows: 4

Query result:

buy_step_id	buy_id	step_id	date_step_beg	date_step_end
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	2020-03-06	2020-03-06
5	2	1	2020-02-28	2020-02-28
6	2	2	2020-02-29	2020-03-01
7	2	3	2020-03-02	None
8	2	4	None	None
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	None
13	4	1	2020-03-20	None
14	4	2	None	None
15	4	3	None	None
16	4	4	None	None
17	5	1	None	None
18	5	2	None	None
19	5	3	None	None
20	5	4	None	None

Пояснение. Все этапы в таблицу **buy_step** можно вставить одним запросом, для этого используется соединение CROSS JOIN для таблиц **buy** и **step**.

Комментарии учащихся:

1.
Алексей Карелин
Формат команды INSERT в MySQL имеет следующий вид:
INSERT [INTO] имя_таблицы [(список_столбцов)] VALUES (значение1, значение2, ... значениеN)
INTO - необязательное слово, но без него задание не принимается почему-то, хотя результат один и тот же.
Галина Озерова

@Алексей_Карелин, Традиционно в запросах INTO указывается, в курсе тоже показан формат записи с INTO. Когда задание принимало оба варианта - было очень много вопросов со стороны пользователя... Поэтому в валидатор добавлена проверка.
На степике используется версия SQL 5.7.2, многие возможности более высоких версий не доступны. А то что описано в шагах - точно работает. По этой причине мы просим пользователей использовать описанные в шагах возможности.
Но в принципе - Вы правы

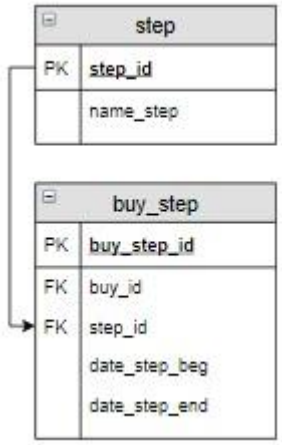
Шаг 9: Задание (Внести дату выставления счёта)

Ссылка в интернете

В таблицу **buy_step** занести дату 12.04.2020 выставления счета на оплату заказа с номером 5.

Правильнее было бы занести не конкретную, а текущую дату. Это можно сделать с помощью функции **Now()**. Но при этом в разные дни будут вставляться разная дата, и задание нельзя будет проверить, поэтому вставим дату 12.04.2020.

Фрагмент логической схемы базы данных:



Результат:

Affected rows: 1

Query result:

buy_step_id	buy_id	step_id	date_step_beg	date_step_end
17	5	1	2020-04-12	None
18	5	2	None	None
19	5	3	None	None
20	5	4	None	None

Пояснение. Для просмотра данных из таблицы **buy_step** выбраны не все записи, а только те, которые относятся к заказу с номером 5.

Шаг 10: Задание (Завершить этап «Оплата»)

Ссылка в интернете

Завершить этап «Оплата» для заказа с номером 5, вставив в столбец **date_step_end** дату 13.04.2020, и начать следующий этап («Упаковка»), задав в столбце **date_step_beg** для этого этапа ту же дату.

Реализовать два запроса для завершения этапа и начала следующего. Они должны быть записаны в общем виде, чтобы его можно было применять для любых этапов, изменив только текущий этап. Для примера пусть это будет этап «Оплата».

Фрагмент предметной области:

Результат:

Affected rows: 1

Affected rows: 1

Query result:

buy_step_id	buy_id	step_id	date_step_beg	date_step_end
17	5	1	2020-04-12	2020-04-13
18	5	2	2020-04-13	None
19	5	3	None	None
20	5	4	None	None

Пояснение. В таблицу **step** все необходимые этапы занесены последовательно. Если текущий этап «Оплата», его **id** 1, то у следующего этапа «Упаковка» **id** будет на единицу больше, то есть 2. Поэтому в условии отбора запроса, который обновляет дату начала следующего этапа, можно использовать вложенный запрос, который выбирает **id** этапа на 1 больше, чем у текущего:

```
SELECT step_id + 1
FROM step
WHERE name_step = 'Оплата'
```

Шаг 11: Задание (Придумать запрос корректировки данных)

Ссылка в интернете

Придумайте один или несколько запросов корректировки данных для предметной области «Интернет-магазин книг» (в таблицы занесены данные, как на этом шаге). Проверьте, правильно ли они работают.

При желании можно формулировку запросов разместить в комментариях.

Размещенные задания можно реализовать для закрепления материала урока.

Оценивайте понравившиеся Вам запросы.

В последнем модуле создан отдельный урок, в котором мы разместим запросы, набравшие наибольшее количество лайков.