Содержание урока

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.

**Связь «один ко многим»**

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» (c 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.

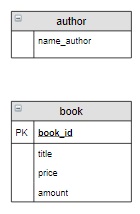
Связь «**один ко многим**» имеет место, когда одной записи главной таблицы соответствует несколько записей связанной таблицы, а каждой записи связанной таблицы соответствует только одна запись главной таблицы. Обозначается это так:



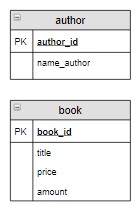
**Этапы реализации связи**«**один ко многим**» на следующем примере:

Один автор может написать несколько книг, а одна книга написана только одним автором. Для каждой книги известны ее количество и цена.

1. Создать таблицу **author**,  в которую включить уникальных авторов книг, хранящихся на складе:



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**

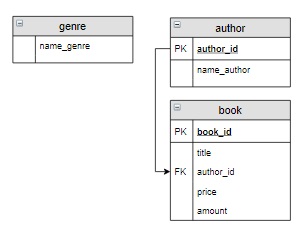
Добавить новую характеристику книги – ее жанр, если считать, что каждая книга относится к одному жанру, то есть между ними определена связь «**один ко многим**». Расположите в правильном порядке этапы связывания таблицы с жанрами (**genre**) и таблицы **book**.

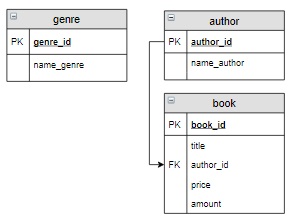
Sort a given list

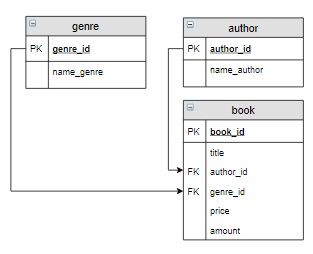
Correct answer from **32,472** learners

Total **75%** of tries are correct

 Great!





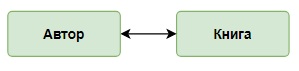


**Связь «многие ко многим»**

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**».

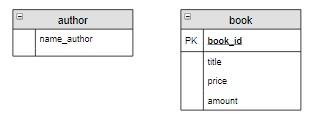
Связь «**многие ко многим**» имеет место когда каждой записи одной таблицы соответствует несколько записей во второй, и наоборот, каждой записи второй таблицы соответствует несколько записей в первой. Обозначается это так:



**Этапы реализации связи**«**многие ко многим**»  на следующем примере:

Один автор может написать несколько книг, а одна книга может быть написана несколькими авторами. Для каждой книги известны ее количество и цена.

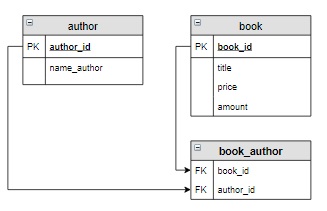
1. Создать таблицу **author**,  в которую включить уникальных авторов книг, хранящихся на складе:



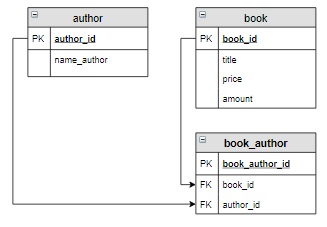
2. В обеих таблицах необходимо определить первичный ключ, в нашем случае в таблице**book** он уже есть, поэтому достаточно включить первичный ключ **author\_id**в таблицу **author**:



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 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**».

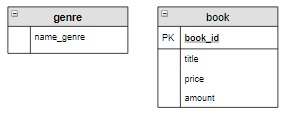
Расположите в правильном порядке этапы связывания таблиц **genre** и **book**.

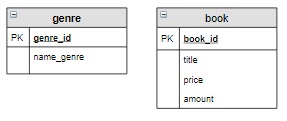
Sort a given list

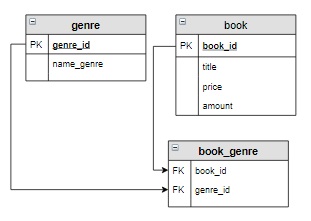
Correct answer from **31,406** learners

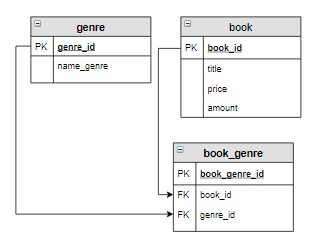
Total **75%** of tries are correct

 You are right, well done!









**Задание**

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

Fill in the blanks

Correct answer from **31,608** learners

Total **97%** of tries are correct

 All is correct.

В **группе** учатся **студенты**, каждый студент может учиться только в одной группе:                                           
**Студенты** изучают несколько **дисциплин**, одну и ту же дисциплину могут изучать несколько студентов:  
                                          
В библиотеке **студент** может взять несколько **книг**, одну и ту же книгу могут взять несколько студентов (в разное время):                                           
Каждый **студент** родился в определенном **городе**, в одном городе родились несколько студентов:                                           
Каждый **город** относится к определенной **стране**, в каждой стране есть несколько городов:                                         

**Задание**

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**

**Пояснение от пользователя @Дмитрий\_Ефремов**

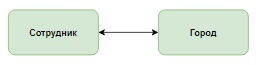
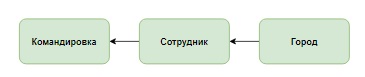
Select all correct options from the list

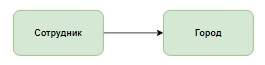
Correct answer from **29,560** learners

Total **22%** of tries are correct

 Correct.

You've solved a complex problem, congratulations! Now you can help other learners in [comments](https://stepik.org/lesson/308885/step/5?unit=291011#discussions) by answering their questions, or compare your solution with others on [solution forum](https://stepik.org/lesson/308885/step/5?thread=solutions&unit=291011#discussions).





**Задание**

Создать таблицу**author**следующей структуры:

|  |  |
| --- | --- |
| **Поле** | **Тип, описание** |
| author\_id | INT PRIMARY KEY AUTO\_INCREMENT |
| name\_author | VARCHAR(50) |

Enter an SQL query

Correct answer from **30,862** learners

Total **81%** of tries are correct

 Totally right.

Affected rows: 0 Affected rows: 0



1

CREATE TABLE author(

2

  author\_id INT PRIMARY KEY AUTO\_INCREMENT,

3

  name\_author VARCHAR(50)

4

);

5

​

6

SELECT \* FROM author;

7

​

## Задание

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

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

**Результат**

### Enter an SQL query

Correct answer from **30,464** learners

Total **71%** of tries are correct

 Yes!

Affected rows: 4 Query result: +-----------+------------------+ | author\_id | name\_author | +-----------+------------------+ | 1 | Булгаков М.А. | | 2 | Достоевский Ф.М. | | 3 | Есенин С.А. | | 4 | Пастернак Б.Л. | +-----------+------------------+ Affected rows: 4

Hide



1

INSERT INTO author(name\_author)

2

VALUES ('Булгаков М.А.'), ('Достоевский Ф.М.'), ('Есенин С.А.'), ('Пастернак Б.Л.');

3

​

4

SELECT\*FROM author;

5

​

6

​

**Создание таблицы с внешними ключами**

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

* каждый внешний ключ должен иметь такой же тип данных, как связанное поле главной таблицы (в наших примерах это **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)

По умолчанию любой столбец, кроме ключевого, может содержать значение **NULL**. При создании таблицы это можно переопределить,  используя  ограничение**NOT NULL** для этого столбца:

CREATE TABLE таблица (

    столбец\_1 INT NOT NULL,

    столбец\_2 VARCHAR(10)

);

 В созданной таблице в **столбец\_1** не может содержать пустое значение, а**столбец\_2** - может.

Для внешних ключей рекомендуется устанавливать ограничение **NOT NULL** (если это совместимо с другими опциями, которые будут рассмотрены в следующем шаге).

**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**, пустое значение не допускается |
| price | DECIMAL(8, 2) |  |
| amount | INT |  |

*Запрос:*

CREATE TABLE book (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(50),

author\_id INT NOT NULL,

price DECIMAL(8,2),

amount INT,

FOREIGN KEY (author\_id) REFERENCES author (author\_id)

);

**Задание**

Перепишите запрос на создание таблицы **book** , чтобы ее структура соответствовала структуре, показанной на логической схеме (таблица **genre** уже создана, порядок следования столбцов - как на логической схеме в таблице **book**, **genre\_id**  - внешний ключ) . Для **genre\_id** ограничение о недопустимости пустых значений **не задавать**. В качестве главной таблицы для описания поля  **genre\_id**использовать таблицу [**genre**](https://stepik.org/lesson/297508/step/5?unit=279268) следующей структуры:

|  |  |
| --- | --- |
| **Поле** | **Тип, описание** |
| genre\_id | INT PRIMARY KEY AUTO\_INCREMENT |
| name\_genre | VARCHAR(30) |

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



1

CREATE TABLE book(

2

  book\_id INT PRIMARY KEY AUTO\_INCREMENT,

3

  title VARCHAR(50),

4

  author\_id INT NOT NULL,

5

  genre\_id INT,

6

  price DECIMAL(8,2),

7

  amount INT,

8

   FOREIGN KEY (author\_id) REFERENCES author (author\_id),

9

   FOREIGN KEY (genre\_id) REFERENCES genre (genre\_id)

10

);

11

​

12

/\*для начала понять, какие таблицы вообще созданы в БД\*/

13

SELECT table\_name, engine

14

FROM information\_schema.tables

15

WHERE engine = 'InnoDB';

16

​

17

SHOW COLUMNS FROM book; /\*Можно посмотреть   какие строки получаются при создании таблицы\*/

18

​

19

/\*OR\*/

20

​

21

DESCRIBE book;

**Действия при удалении записи главной таблицы**

С помощью выраженияON 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.

**Важно!**Если для столбца установлена опция SET NULL, то при его описании нельзя задать ограничение на пустое значение.

**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 NOT NULL,

price DECIMAL(8,2),

amount INT,

FOREIGN KEY (author\_id) REFERENCES author (author\_id) ON DELETE CASCADE

);

**Задание**

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**.

1

CREATE TABLE book(

2

  book\_id INT PRIMARY KEY AUTO\_INCREMENT,

3

  title VARCHAR(50),

4

  author\_id INT NOT NULL,

5

  genre\_id INT,

6

  price DECIMAL(8,2),

7

  amount INT,

8

   FOREIGN KEY (author\_id) REFERENCES author (author\_id) ON DELETE CASCADE,

9

   FOREIGN KEY (genre\_id) REFERENCES genre (genre\_id) ON DELETE SET NULL

10

);

11

​

12

SELECT \* FROM book;

13

​

Для того, чтобы проще было понять тему, представьте, что база данных это множество таблиц состоящих из столбцов и строк. У каждой таблицы есть имя, мы его пишем когда создаем 'CREATE TABLE имя\_таблицы'.

Так вот таблицы нужны для того, чтобы было удобно и быстро искать данные, т.к. когда все лежит в одной таблице это сложно, медленно и неудобно. Данные будут дублироваться, таблица будет огромной по количеству строк.

А теперь представьте, что мы разбили большую таблицу на несколько маленьких и для связи между собой мы используем зависимости Первичный ключ (PK, главный, уникальный ключ-идентификатор, которые не повторяется в пределах этой таблицы) и Вторичный ключ (FK, нужен для того, чтобы зацепить столбец со столбцом из другой таблицы).

Ниже пример. В самую нижнюю таблицу мы добавляем FK, таким образом автоматически быстро привязываем Ивана Иванова с Математикой и ставим ему оценку "3", далее снова Иванова связываем с физикой и оценкой "4".

Вместо создания новой таблицы, куда мы копировали бы полностью все данные, мы всего лишь добавляем первичные уникальные ключи из других таблиц и заносим баллы (оценки). Таблица получается компактной, меньше весит и быстрее искать в ней что-то.

