[1.1. Отношение (таблица) 2](#_Toc56981925)

[1.1.04. Основные понятия реляционных баз данных 2](#_Toc56981926)

[1.1.05. Отношение, реляционная модель 4](#_Toc56981927)

[1.1.06. Выбор типов данных для полей 7](#_Toc56981928)

[1.1.07. Создание таблицы 9](#_Toc56981929)

[1.1.08. Вставка записи в таблицу 10](#_Toc56981930)

[1.2. Выборка данных 12](#_Toc56981931)

[1.2.02. Выборка всех данных из таблицы 12](#_Toc56981932)

[1.4. Вложенные запросы. 13](#_Toc56981933)

[1.4.02.Вложенный запрос, возвращающий одно значение 13](#_Toc56981934)

[1.4.03. Использование вложенного запроса в выражении 14](#_Toc56981935)

[1.4.04. Вложенный запрос, оператор IN 14](#_Toc56981936)

[1.4.05. Вложенный запрос, операторы ANY и ALL 15](#_Toc56981937)

[1.4.06. Вложенный запрос после SELECT 17](#_Toc56981938)

[1.5. Запросы корректировки данных. 19](#_Toc56981939)

[1.5.04. Добавление записей из другой таблицы 19](#_Toc56981940)

[1.5.05. Добавление записей, вложенные запросы 20](#_Toc56981941)

[1.5.06. Запросы на обновление 21](#_Toc56981942)

[1.5.07. Запросы на обновление нескольких столбцов 23](#_Toc56981943)

[1.5.08. Запросы на обновление, несколько таблиц 25](#_Toc56981944)

[1.5.09. Запросы на удаление 26](#_Toc56981945)

# 1.1. Отношение (таблица)

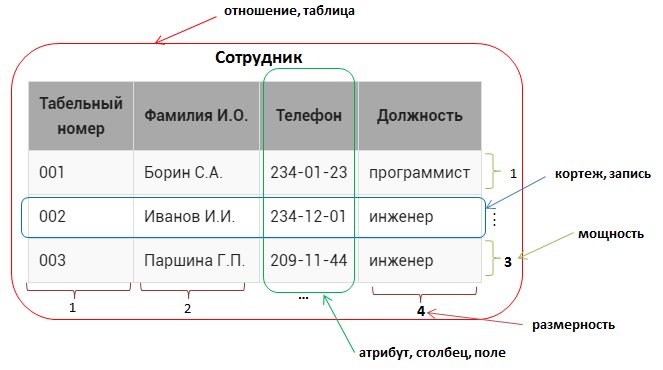
## 1.1.04. Основные понятия реляционных баз данных

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}.

## 1.1.05. Отношение, реляционная модель

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 ;
* **priсe** – 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 |

## 1.1.06. Выбор типов данных для полей

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 -1038+1 до 1038-1. DECIMAL(4,1)  NUMERIC(6,3) | 34.6 -3.294 |
| DATE | Дата в формате ГГГГ-ММ-ДД  26 июля 2020 года3 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);
* **priсe** - для описание денежного значения используется числовой тип данных с двумя знаками после запятой - DECIMAL(8,2);
* **amount** - целое число -INT.

## 1.1.07. Создание таблицы

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).

## 1.1.08. Вставка записи в таблицу

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:



# 1.2. Выборка данных

## 1.2.02. Выборка всех данных из таблицы

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;

Результат:

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

| 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 |

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

# 1.4. Вложенные запросы.

## 1.4.02.Вложенный запрос, возвращающий одно значение

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.

## 1.4.03. Использование вложенного запроса в выражении

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.

## 1.4.04. Вложенный запрос, оператор IN

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.

## 1.4.05. Вложенный запрос, операторы ANY и ALL

 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 которыхamount 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);

## 1.4.06. Вложенный запрос после SELECT

 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 |

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

# 1.5. Запросы корректировки данных.

## 1.5.04. Добавление записей из другой таблицы

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:

1. количество полей в таблице и количество полей в запросе должны совпадать;
2. 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.
3. 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.

## 1.5.05. Добавление записей, вложенные запросы

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.

## 1.5.06. Запросы на обновление

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»).

## 1.5.07. Запросы на обновление нескольких столбцов

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).

## 1.5.08. Запросы на обновление, несколько таблиц

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.

## 1.5.09. Запросы на удаление

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».