**Памятка о типичных ошибках и способах их исправления**

Эта памятка написана, чтобы помочь Вам **самостоятельно находить ошибки в своих запросах**. Если ваш запрос не принимается системой, то возвращайтесь на эту страничку и пройдитесь по всем пунктам:

1. Приведите синтаксис запроса к общепринятому:

* если у вас есть время, стоит изучить руководство по стилю SQL <https://www.sqlstyle.guide/ru/>
* можете отформатировать ваш запрос с помощью, например, <https://codebeautify.org/sqlformatter>
* в любом случае, информации и примеров в курсе достаточно для того, чтобы писать запросы корректно.

2. Проверьте, что ключевые слова, названия столбцов и значения в ячейках, которые необходимо найти, написаны правильно. Особенно обратите внимание, чтобы в русских названиях столбцов не было английских букв.

3. Проверьте, что:

* количество открывающихся скобок равно количеству закрывающихся;
* запятые разделяют перечисление столбцов, но не ключевые слова;
* запросы разделяются точкой с запятой.

4. Проверьте, что последовательность команд указана верно (она отличается от последовательности выполнения команд в запросе):

SELECT 'столбцы или \* для выбора всех столбцов; обязательно'

FROM 'таблица; обязательно'

WHERE 'условие/фильтрация, например, city = 'Moscow'; необязательно'

GROUP BY 'столбец, по которому хотим сгруппировать данные; необязательно'

HAVING 'условие/фильтрация на уровне сгруппированных данных; необязательно'

ORDER BY 'столбец, по которому хотим отсортировать вывод; необязательно'

5.  Если запрос включает подзапросы, выполните сначала подзапросы и удостоверьтесь, что получаете ожидаемый результат.

6. **Прочитайте комментарии под заданием**: большинство трудностей уже обсуждалось не один раз.

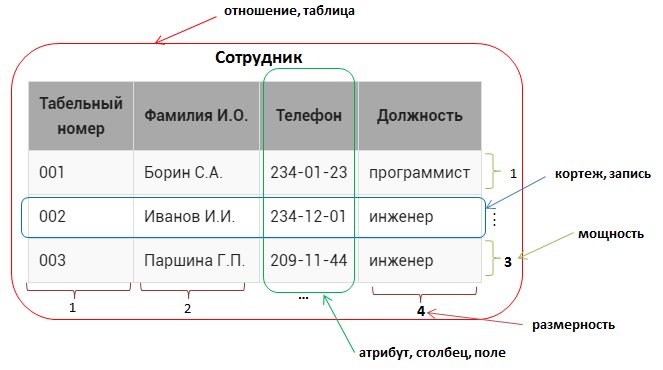
**Основные понятия реляционных баз данных**

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}.

**Задание**

**===========================================**

**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 |

**Задание**

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

**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 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);
* **priсe** - 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, с помощью которых их можно описать.

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.

**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).

**Задание**

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 |

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 \*/.

**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 (ключевое слово INTO можно пропустить);
* 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:*



**Задание**

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...

*Результат:*

Affected rows: 1

Query result:

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

| book\_id | title | author | price | amount |

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

| 1 | Мастер и Маргарита | Булгаков М.А. | 670.99 | 3 |

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

**Задание**

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**- кликните по этой строке, чтобы раскрыть пояснение

*Результат* :

Affected rows: 1

Affected rows: 1

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 |

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

Enter an SQL query

Correct answer from **68,824** learners

Total **51%** of tries are correct



1

--and

2

INSERT INTO book (title, author, price, amount) values ('Белая гвардия', 'Булгаков М.А.', 540.50, 5), ('Идиот', 'Достоевский Ф.М.', 460.00, 10), ('Братья Карамазовы', 'Достоевский Ф.М.', 799.01, 2);

3

SELECT \* FROM book;

4

​

5

​

6

​

7

​

8

​

Affected rows: 3 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 | +---------+--------------------+------------------+--------+--------+ Affected rows: 4

INSERT INTO book (title, author, price, amount) values ('Белая гвардия', 'Булгаков М.А.', 540.50, 5);

INSERT book (title, author, price, amount) values ('Идиот', 'Достоевский Ф.М.', 460.00, 10);

INSERT INTO book (title, author, price, amount) values ('Братья Карамазовы', 'Достоевский Ф.М.', 799.01, 2);