**3. СОЗДАНИЕ ТАБЛИЦ**

**3.1 Создание простейших таблиц**

**Примечания к задачам на создание таблиц**

Раньше для получения или изменения данных вы вставляли в редактор один SQL-запрос.  
В задачах на создание таблиц нужно одновременно выполнять **несколько SQL-запросов**:  
1. Создание таблицы (**CREATE TABLE**).  
2. Наполнение таблицы данными (**INSERT**).

Несколько SQL запросов, которые следуют друг за другом, нужно разделять точкой с запятой.  
Так запросы из урока «Создание простейших таблиц» могут выглядеть так:

CREATE TABLE users2 (

id INT,

first\_name VARCHAR(20),

last\_name VARCHAR(50),

birthday DATE,

age TINYINT,

active BOOL

);

INSERT INTO users2 (id, first\_name, last\_name, birthday, age, active)

VALUES (1, 'Дмитрий', 'Иванов', '1986-02-18', 31, True);

INT, TINYINT – числовые поля;

CHAR, VARCHAR, TEXT – текстовые поля;

VARCHAR(50) – поле не должно превышать 50 символов.

CHAR(50) – поле составляет ровно 50 символов.

BOOL – логическое поле.

**Пример 1.**

Создайте таблицу **messages** с со следующими полями:

1. **id** типа **INT**;
2. **subject** типа **VARCHAR** длиной 100 символов
3. **message** тип **TEXT**;
4. **add\_date** типа **DATETIME**;
5. **is\_public** логического типа.

Добавьте в таблицу сообщение с темой «Первое сообщение» и текстом «Это мое первое сообщение!». Дату установите 12 декабря 2016 года 14 часов, 16 минут. Сообщение должно быть публичным.

Таблица messages (итоговая таблица)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **subject** | **message** | **add\_date** | **is\_public** |
| 1 | Первое сообщение | Это мое первое сообщение! | 2016-12-12 14:16:00 | 1 |

**Решение:**

CREATE TABLE messages (

id INT,

subject VARCHAR(100),

message TEXT,

add\_date DATETIME,

is\_public BOOL

);

INSERT INTO messages (id, subject, message, add\_date, is\_public)

VALUES (1, 'Первое сообщение', 'Это мое первое сообщение!', '2016-12-12 14:16:00', 1);

**Пример 2.**

Создайте таблицу **rating**

1. **id** типа **INT**;
2. **car\_id** типа **INT**;
3. **user\_id** типа **INT**;
4. **rating** типа **FLOAT**.

Добавьте в неё 4 записи так, чтобы получалась таблица ниже:

Таблица rating

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **car\_id** | **user\_id** | **rating** |
| 1 | 1 | 1 | 4.54 |
| 2 | 1 | 2 | 3.34 |
| 3 | 2 | 3 | 4.19 |
| 4 | 2 | 11 | 1.12 |

**Решение:**

CREATE TABLE rating (

id INT,

car\_id INT,

user\_id INT,

rating FLOAT

);

INSERT INTO rating (id, car\_id, user\_id, rating)

VALUES (1, 1, 1, 4.54),

(2, 1, 2, 3.34),

(3, 2, 3, 4.19),

(4, 2, 11, 1.12);

**3.2 ЧИСЛОВЫЕ ПОЛЯ**

Если в поле **temperature** – температура, целое число в диапазоне от -100 до 100 градусов: temperature tinyint

Если число положительное, то пишем, например, для поля ID: id INT UNSIGNED

Для цены не более 1 млн. руб: price MEDIUMINT UNSIGNED

Если поле **size** — целочисленное поле для хранения размера файла в байтах. Только положительные числа. Могут храниться данные до 100 Гб: size BIGINT UNSIGNED

Если используется денежный тип, например, amount – сумма заказа:

amount DECIMAL (8,2)

Т.е. всего 8 знаков, но 2 после запятой.

Но вставляемое значение пишем с точкой: 1240.00

Если поле **rating** – рейтинг, вещественное число. Принимает положительные значения от 0 до 10, то

rating FLOAT UNSIGNED

Если поле **bio** — биография, текст до 65000 символов:

bio TEXT

Если только дата, то тип данных DATE, если нужны и дата и время – DATETIME, если только год – YEAR

**3.6 NULL**

**NULL** - это специальное значение, которое используется в SQL для обозначения отсутствия данных. Оно отличается от пустой строки или нулевого значения, так как NULL означает отсутствие какого-либо значения в ячейке таблицы.

**Пример 1.**

Создайте таблицу **products** со следующими полями:

1. **id** — номер товара, целое положительное, не может принимать NULL.
2. **name** — название товара до 120 символов, не может принимать NULL.
3. **category\_id** — категория товара, целое положительное, может принимать NULL.
4. **price** — цена, денежный тип из 10 знаков, из них два после запятой. Не может принимать NULL.

Добавьте 4 записи так, чтобы получалась таблица ниже:

Таблица products

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **category\_id** | **price** |
| 1 | Подгузники (12 шт) | 3 | 700.00 |
| 2 | Подгузники (24 шт) | 3 | 1250.00 |
| 3 | Спиннер | NULL | 250.40 |
| 4 | Пюре слива | 4 | 47.50 |

**Решение:**

CREATE TABLE products (

id INT UNSIGNED not null,

name VARCHAR (120) not null,

category\_id INT UNSIGNED null,

price decimal (10,2) not null

);

INSERT INTO products (id, name, category\_id, price)

VALUES

(1, 'Подгузники (12 шт)', 3,700),

(2, 'Подгузники (24 шт)', 3,1250.00),

(3, 'Спиннер', null, 250.40),

(4, 'Пюре слива', 4, 47.50);

**3.8 BOOL, ENUM, SET**

Типы ENUM и SET — это особые строковые типы, значения которых выбираются из фиксированного списка значений. Главная разница между ними заключается в том, что значение типа ENUM должно содержать точно одно значение из указанного множества, а тип SET позволяет выбрать нескольких значений одновременно.

**Задача 3.8.1 Статьи**

Создайте таблицу **articles** для хранения данных о статьях. В таблице должны быть следующие поля:

* **id** — идентификатор, целое положительное, NULL запрещен
* **name** — название статьи, строка до 80 символов.
* **text** — текст статьи.
* **state** — статус статьи. Поле из 3 вариантов: draft (черновик), correction (корректура), public (опубликована).

Добавьте 3 записи так, чтобы получалась таблица ниже:

Таблица articles

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **text** | **state** |
| 1 | Новое в Python 3.6 |  | draft |
| 2 | Оптимизация SQL запросов | При больших объемах данных ... | correction |
| 3 | Транзакции в MySQL | По долгу службы мне приходится ... | public |

**Решение:**

CREATE table articles (

id int unsigned not NULL,

name VARCHAR (80),

text TEXT,

state ENUM ('draft', 'correction', 'public')

);

insert into articles (id, name, text, state)

values (1, 'Новое в Python 3.6', '', 'draft'),

(2, 'Оптимизация SQL запросов', 'При больших объемах данных ...', 'correction'),

(3, 'Транзакции в MySQL', 'По долгу службы мне приходится ...', 'public');

**Пример 2.**

Создайте таблицу **rooms** для хранения номеров в отеле:

1. **id** — идентификатор, целое положительное. NULL запрещен.
2. **number** — номер комнаты, целое положительное. Всего в отеле 107 комнат. NULL запрещен.
3. **beds** — количество спальных мест. Выбор из **1+1**, **2+1**, **2+2**. Можно выбрать только один вариант. NULL запрещен.
4. **additional** — дополнительные удобства в номере. Можно выбрать несколько вариантов из списка: **conditioner**, **bar**, **fridge** и **wifi**.

Добавьте 3 записи так, чтобы получалась таблица ниже:

Таблица rooms

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **number** | **beds** | **additional** |
| 1 | 10 | 1+1 | conditioner,bar,wifi |
| 2 | 12 | 2+1 |  |
| 3 | 24 | 2+2 | fridge,bar,wifi |

**Решение:**

CREATE table rooms (

id int unsigned not NULL,

number tinyint unsigned not null,

beds ENUM ('1+1', '2+1', '2+2') not null,

additional set ('conditioner', 'bar', 'fridge', 'wifi')

);

insert into rooms (id, number, beds, additional)

values (1, 10, '1+1', 'conditioner,bar,wifi'),

(2, '12', '2+1', ''),

(3, '24', '2+2', 'fridge,bar,wifi');

**3.10 ЗНАЧЕНИЯ ПО УМОЛЧАНИЮ**

Атрибут **DEFAULT** определяет значение по умолчанию для столбца. Если при добавлении данных для столбца не будет предусмотрено значение, то для него будет использоваться значение по умолчанию.

**Пример 1.**

Создайте таблицу **orders** для хранения списка заказов:

1. **id** — идентификатор, целое положительное.
2. **user\_id** — идентификатор пользователя, который оформил заказ. Целое положительное, NULL запрещен.
3. **amount** — стоимость заказа. Целое положительное число не более 1 млн. NULL запрещен, по умолчанию 0.
4. **created** — дата и время создания заказа. NULL запрещен.
5. **state** — статус заказа.

Выбор из **new**, **cancelled**, **in\_progress**, **delivered**, **completed**. Можно выбрать только один вариант. NULL запрещен. По умолчанию должен стоять **new**.

Добавьте 3 записи так, чтобы получалась таблица ниже:

Таблица orders

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **user\_id** | **amount** | **created** | **state** |
| 1 | 56 | 5400 | 2018-02-01 17:46:59 | new |
| 2 | 90 | 249 | 2018-02-01 19:13:04 | new |
| 3 | 78 | 2200 | 2018-02-01 22:43:09 | new |

**Решение:**

create table orders (

id int unsigned not NULL,

user\_id int unsigned not null,

amount mediumint unsigned not null default 0,

created DATETIME not null,

state enum ('new', 'cancelled', 'in\_progress', 'delivered', 'completed') not null default 'new'

);

insert into orders (id, user\_id, amount, created, state)

values

(1, 56, 5400, '2018-02-01 17:46:59', 'new'),

(2, 90, 249, '2018-02-01 19:13:04', 'new'),

(3, 78, 2200, '2018-02-01 22:43:09', 'new');

**Пример 2.**

Создайте таблицу **users** для хранения списка пользователей сайта:

1. **id** — идентификатор, целое положительное.
2. **first\_name** — имя, строка до 20 символов. NULL запрещен.
3. **last\_name** — фамилия, строка до 20 символов. NULL запрещен.
4. **patronymic** — отчество, строка до 20 символов. NULL запрещен, по умолчанию пустая строка.
5. **is\_active** — отметка об активности пользователя. Логическое поле, по умолчанию TRUE.
6. **is\_superuser** — отметка администратора. Логическое поле, по умолчанию FALSE.

Добавьте 3 записи так, чтобы получалась таблица ниже:

Таблица users

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **first\_name** | **last\_name** | **patronymic** | **is\_active** | **is\_superuser** |
| 1 | Дмитрий | Иванов |  | TRUE | FALSE |
| 2 | Анатолий | Белый | Сергеевич | TRUE | TRUE |
| 3 | Андрей | Крючков |  | FALSE | FALSE |

**Решение:**

create table users (

id int unsigned not NULL,

first\_name varchar (20) not null,

last\_name varchar (20) not null,

patronymic varchar (20) not null default '',

is\_active bool default TRUE,

is\_superuser bool default FALSE

);

insert into users (id, first\_name, last\_name, patronymic, is\_active, is\_superuser)

values

(1, 'Дмитрий', 'Иванов', '', TRUE, FALSE),

(2, 'Анатолий', 'Белый', 'Сергеевич', TRUE, TRUE),

(3, 'Андрей', 'Крючков', '', FALSE, FALSE);