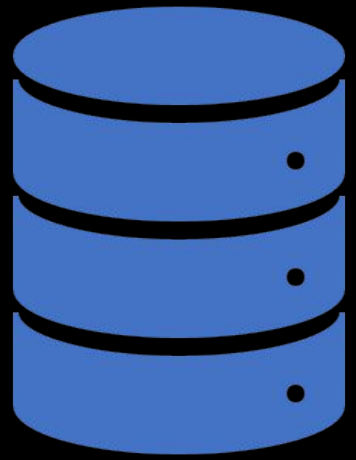

MYSQL

система управления реляционными базами данных

HISTORY

- Изначальным разработчиком данной СУБД была шведская компания MySQL AB. В 1995 году она выпустила первый релиз MySQL. В 2008 году компания MySQL AB была куплена компанией Sun Microsystems, а в 2010 году уже компания Oracle поглотила Sun и тем самым приобрела права на торговую марку MySQL. Поэтому MySQL на сеголняшней день развивается под эгодой Oracle.
- Текущей актуальной версией СУДБ является версия 8.0, которая вышла в январе 2018 года.
- MySQL обладает кроссплатформенностью, имеются дистрибутивы под самые различные ОС, в том числе наиболее популярные версии Linux, Windows, MacOS.
- Официальный сайт проекта: https://www.mysql.com/.

СОЗДАНИЕ БАЗЫ ДАННЫХ

Для создания базы данных используется команда CREATE DATABASE. Она имеет следующий синтаксис:

CREATE DATABASE [IF NOT EXISTS] имя_базы_даных;

В конце команды указывается имя базы данных.

Первая форма CREATE DATABASE имя_базы_даных пытается создать базу данных, но если такая база данных уже существует, то операция возвратит ошибку.

Вторая форма CREATE DATABASE IF NOT EXISTS имя_базы_даных пытается создать базу данных, если на сервере отсутствует бд с таким именем.

Например, в MySQL выполним следующую команду:

CREATE DATABASE productsdb;

Она создаст на сервере бд productsdb.

УСТАНОВКА СОЕДИНЕНИЯ С БАЗОЙ ДАННЫХ

После создания БД с ней производятся различные операции: создание таблиц, добавление и получение данных и т.д. Но чтобы установить производить эти операции, надо установить определенную базу данных в качестве используемой. Для этого применяется оператор USE:

USE productsdb;

УДАЛЕНИЕ БАЗЫ ДАННЫХ

Для удаления базы данных применяется команда DROP DATABASE, которая имеет следующий синтаксис:

DROP DATABASE [IF EXISTS] имя_базы_даных;

Первая форма DROP DATABASE имя_базы_даных пытается удалить базу данных, но если такая база данных отсутствует на сервере, то операция возвратит ошибку.

Вторая форма DROP DATABASE IF EXISTS имя_базы_даных пытается удалить базу данных, если на сервере имеется бд с таким именем.

Например, удалим выше созданную базу данных productsdb:

DROP DATABASE productsdb;

ТИПЫ ДАННЫХ MYSQL

- Символьные типы (CHAR, VARCHAR, TINYTEXT, TEXT, MEDIUMTEXT, LARGETEXT)
- Числовые типы (TINYINT, BOOL, TINYINT UNSIGNED, SMALLINT, SMALLINT UNSIGNED, MEDIUMINT, MEDIUMINT UNSIGNED, INT, INT UNSIGNED, BIGINT, BIGINT UNSIGNED, DECIMAL, FLOAT, DOUBLE)
- Типы для работы с датой и временем (DATE, TIME, DATETIME, TIMESTAMP, YEAR)
- Составные типы (ENUM, SET)
- Бинарные типы (TINYBLOB, BLOB, MEDIUMBLOB, LARGEBLOB)

СИМВОЛЬНЫЕ ТИПЫ

CHAR: представляет стоку фиксированной длины.

Длина хранимой строки указывается в скобках, например, CHAR(10) - строка из десяти символов. И если в таблицу в данный столбец сохраняется строка из 6 символов (то есть меньше установленной длины в 10 символов), то строка дополняется 4 проблеми и в итоге все равно будет занимать 10 символов

VARCHAR: представляет стоку переменной длины.

Длина хранимой строки также указывается в скобках, например, VARCHAR(10). Однако в отличие от CHAR хранимая строка будет занимать именно столько места, сколько необходимо. Например, если определеная длина в 10 символов, но в столбец сохраняется строка в 6 символов, то хранимая строка так и будет занимать 6 символов плюс дополнительный байт, который хранит длину строки.

TINYTEXT: представляет текст длиной до 255 байт.

TEXT: представляет текст длиной до 65 КБ.

MEDIUMTEXT: представляет текст длиной до 16 МБ

LARGETEXT: представляет текст длиной до 4 ГБ

ЧИСЛОВЫЕ ТИПЫ

TINYINT: представляет целые числа от -127 до 128, занимает 1 байт

BOOL: фактически не представляет отдельный тип, а является лишь псевдонимом для типа TINYINT(1) и может хранить два значения 0 и 1. Однако данный тип может также в качестве значения принимать встроенные константы **TRUE** (представляет число 1) и **FALSE**(предоставляет число 0).

Также имеет псевдоним **BOOLEAN**.

TINYINT UNSIGNED: представляет целые числа от 0 до 255, занимает 1 байт

SMALLINT: представляет целые числа от -32768 до 32767, занимает 2 байта

SMALLINT UNSIGNED: представляет целые числа от 0 до 65535, занимает 2 байта

MEDIUMINT: представляет целые числа от -8388608 до 8388607, занимает 3 байта

MEDIUMINT UNSIGNED: представляет целые числа от 0 до 16777215, занимает 3 байта

ЧИСЛОВЫЕ ТИПЫ

INT: представляет целые числа от -2147483648 до 2147483647, занимает 4 байта

INT UNSIGNED: представляет целые числа от 0 до 4294967295, занимает 4 байта

BIGINT: представляет целые числа от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807, занимает 8 байт

BIGINT UNSIGNED: представляет целые числа от 0 до 18 446 744 073 709 551 615, занимает 8 байт

DECIMAL: хранит числа с фиксированной точностью. Данный тип может принимать два параметра precision и scale: DECIMAL(precision, scale).

Параметр precision представляет максимальное количество цифр, которые может хранить число. Это значение должно находиться в диапазоне от 1 до 65.

Параметр scale представляет максимальное количество цифр, которые может содержать число после запятой. Это значение должно находиться в диапазоне от 0 до значения параметра precision. По умолчанию оно равно 0.

Размер данных в байтах для DECIMAL зависит от хранимого значения.

Данный тип также имеет псевдонимы **NUMERIC**, **DEC**, **FIXED**

ЧИСЛОВЫЕ ТИПЫ

FLOAT: хранит дробные числа с плавающей точкой одинарной точности от $-3.4028 * 10^{38}$ до $3.4028 * 10^{38}$, занимает 4 байта

Может принимать форму FLOAT(M,D), где M - общее количество цифр, а D - количество цифр после запятой

DOUBLE: хранит дробные числа с плавающей точкой двойной точности от $-1.7976 * 10^{308}$ до $1.7976 * 10^{308}$, занимает 8 байт. Также может принимать форму DOUBLE(M,D), где M - общее количество цифр, а D - количество цифр после запятой.

Данный тип также имеет псевдонимы **REAL** и **DOUBLE PRECISION**, которые можно использовать вместо DOUBLE.

ТИПЫ ДЛЯ РАБОТЫ С ДАТОЙ И ВРЕМЕНЕМ

DATE: хранит даты с 1 января 1000 года до 31 деабря 9999 года (с "1000-01-01" до "9999-12-31"). По умолчанию для хранения используется формат yyyy-mm-dd. Занимает 3 байта.

TIME: хранит время от -838:59:59 до 838:59:59. По умолчанию для хранения времени применяется формат "hh:mm:ss". Занимает 3 байта.

DATETIME: объединяет время и дату, диапазон дат и времени - с 1 января 1000 года по 31 декабря 9999 года (с "1000-01-01 00:00:00" до "9999-12-31 23:59:59"). Для хранения по умолчанию используется формат "yyyy-mm-dd hh:mm:ss". Занимает 8 байт

TIMESTAMP: также хранит дату и время, но в другом диапазоне: от "1970-01-01 00:00:01" UTC до "2038-01-19 03:14:07" UTC. Занимает 4 байта

YEAR: хранит год в виде 4 цифр. Диапазон доступных значений от 1901 до 2155. Занимает 1 байт.

ТИПЫ ДЛЯ РАБОТЫ С ДАТОЙ И ВРЕМЕНЕМ

Тип Date может принимать даты в различных форматах, однако непосредственно для хранения в самой бд даты приводятся к формату "yyyy-mm-dd". Некоторые из принимаемых форматов:

- yyyy-mm-dd - 2018-05-25
- yyyy-m-dd - 2018-5-25
- yy-m-dd - 18-05-25

В таком формате двузначные числа от 00 до 69 воспринимаются как даты в диапазоне 2000-2069. А числа от 70 до 99 как диапазон чисел 1970 - 1999.

- yyyymmdd - 20180525
- yyyy.mm.dd - 2018.05.25

Для времени тип Time использует 24-часовой формат. Он может принимать время в различных форматах:

- hh:mi - 3:21 (хранимое значение 03:21:00)
- hh:mi:ss - 19:21:34
- hhmiss - 192134

Примеры значений для типов DATETIME и TIMESTAMP:

- 2018-05-25 19:21:34
- 2018-05-25 (хранимое значение 2018-05-25 00:00:00)

СОСТАВНЫЕ ТИПЫ

ENUM: хранит одно значение из списка допустимых значений. Занимает 1-2 байта

SET: может хранить несколько значений (до 64 значений) из некоторого списка допустимых значений. Занимает 1-8 байт.

БИНАРНЫЕ ТИПЫ

TINYBLOB: хранит бинарные данные в виде строки длиной до 255 байт.
BLOB: хранит бинарные данные в виде строки длиной до 65 КБ.
MEDIUMBLOB: хранит бинарные данные в виде строки длиной до 16 МБ
LARGEBLOB: хранит бинарные данные в виде строки длиной до 4 ГБ

СОЗДАНИЕ ТАБЛИЦЫ

Для создания таблиц используется команда **CREATE TABLE**. Эта команды применяет ряд операторов, которые определяют столбцы таблицы и их атрибуты. Общий формальный синтаксис команды CREATE TABLE:

```
CREATE TABLE название_таблицы
(название_столбца1 тип_данных атрибуты_столбца1,
название_столбца2 тип_данных атрибуты_столбца2,
.....
название_столбцаN тип_данных атрибуты_столбцаN,
атрибуты_уровня_таблицы
)
```

После команды CREATE TABLE идет название таблицы. Имя таблицы выполняет роль ее идентификатора в базе данных, поэтому оно должно быть уникальным. Затем в скобках перечисляются названия столбцов, их типы данных и атрибуты. В самом конце можно определить атрибуты для всей таблицы. Атрибуты столбцов, а также атрибуты таблицы указывать необязательно.

СОЗДАНИЕ ТАБЛИЦЫ

Создадим простейшую таблицу. Для этого выполним следующий скрипт:

```
CREATE DATABASE productsdb;

USE productsdb;

CREATE TABLE Customers
(
    Id INT,
    Age INT,
    FirstName VARCHAR(20),
    LastName VARCHAR(20)
);
```

Таблица не может создаваться сама по себе. Она всегда создается в определенной базе данных. Вначале здесь создается база данных productsdb. И затем, чтобы указать, что все дальнейшие операции, в том числе создание таблицы, будут производиться с этой базой данных, применяется команда **USE**.

СОЗДАНИЕ ТАБЛИЦЫ

Далее собственно идет создание таблицы, которая называется Customers. Она определяет четыре столбца: Id, Age, FirstName, LastName. Первые два столбца представляют идентификатор клиента и его возраст и имеют тип INT, то есть будут хранить числовые значения. Следующие столбцы представляют имя и фамилию клиента и имеют тип VARCHAR(20), то есть представляют строку длиной не более 20 символов. В данном случае для каждого столбца определены имя и тип данных, при этом атрибуты столбцов и таблицы в целом отсутствуют.

И в результате выполнения этой команды будет создана база данных productsdb, в которой будет создана таблица Customers.

ПЕРЕИМЕНОВАНИЕ ТАБЛИЦ

Если после создания таблицы мы захотим ее переименовать, то для этого нужно использовать команду **RENAME TABLE**, которая имеет следующий синтаксис:

```
RENAME TABLE старое_название TO новое_название;
RENAME TABLE Customers TO Clients;
```

ПОЛНОЕ УДАЛЕНИЕ ДАННЫХ

Для полного удаления данных, очистки таблицы применяется команда **TRUNCATE TABLE**. Например, очистим таблицу Clients:

```
TRUNCATE TABLE Clients;
```

УДАЛЕНИЕ ТАБЛИЦ

Для удаления таблицы из БД применяется команда **DROP TABLE**, после которой указывается название удаляемой таблицы. Например, удалим таблицу Clients:

```
DROP TABLE Clients;
```

АТРИБУТЫ СТОЛБЦОВ И ТАБЛИЦ

С помощью атрибутов можно настроить поведение столбцов. Рассмотрим, какие атрибуты мы можем использовать.

Атрибуты таблиц:

- **PRIMARY KEY**
- **AUTO_INCREMENT**
- **UNIQUE**
- **NULL и NOT NULL**
- **DEFAULT**
- **CHECK**
- **Оператор CONSTRAINT. Установка имени ограничений**

PRIMARY KEY

Атрибут **PRIMARY KEY** задает первичный ключ таблицы.

```
USE productsdb;

CREATE TABLE Customers
(
    Id INT PRIMARY KEY,
    Age INT,
    FirstName VARCHAR(20),
    LastName VARCHAR(20)
);
```

Первичный ключ уникально идентифицирует строку в таблице. В качестве первичного ключа необязательно должны выступать столбцы с типом int, они могут представлять любой другой тип.

PRIMARY KEY

Установка первичного ключа на уровне таблицы:

```
USE productsdb;
CREATE TABLE Customers
(
    Id INT,
    Age INT,
    FirstName VARCHAR(20),
    LastName VARCHAR(20),
    PRIMARY KEY(Id)
);
```

PRIMARY KEY

Первичный ключ может быть составным. Такой ключ использовать сразу несколько столбцов, чтобы уникально идентифицировать строку в таблице. Например:

```
CREATE TABLE OrderLines
(
    OrderId INT,
    ProductId INT,
    Quantity INT,
    Price MONEY,
    PRIMARY KEY(OrderId, ProductId)
)
```

Здесь поля OrderId и ProductId вместе выступают как составной первичный ключ. То есть в таблице OrderLines не может быть двух строк, где для обоих из этих полей одновременно были бы одни и те же значения.

AUTO_INCREMENT

Атрибут **AUTO_INCREMENT** позволяет указать, что значение столбца будет автоматически увеличиваться при добавлении новой строки. Данный атрибут работает для столбцов, которые представляют целочисленный тип или числа с плавающей точкой.

```
CREATE TABLE Customers
(
    Id INT PRIMARY KEY AUTO_INCREMENT,
    Age INT,
    FirstName VARCHAR(20),
    LastName VARCHAR(20)
);
```

В данном случае значение столбца Id каждой новой добавленной строки будет увеличиваться на единицу

UNIQUE

Атрибут **UNIQUE** указывает, что столбец может хранить только уникальные значения.

```
CREATE TABLE Customers
(
    Id INT PRIMARY KEY AUTO_INCREMENT,
    Age INT,
    FirstName VARCHAR(20),
    LastName VARCHAR(20),
    Phone VARCHAR(13) UNIQUE
);
```

В данном случае столбец Phone, который представляет телефон клиента, может хранить только уникальные значения. И мы не сможем добавить в таблицу две строки, у которых значения для этого столбца будет совпадать.

UNIQUE

Также мы можем определить этот атрибут на уровне таблицы:

```
CREATE TABLE Customers
(
    Id INT PRIMARY KEY AUTO_INCREMENT,
    Age INT,
    FirstName VARCHAR(20),
    LastName VARCHAR(20),
    Email VARCHAR(30),
    Phone VARCHAR(20),
    UNIQUE(Email, Phone)
);
```

NULL И NOT NULL

Чтобы указать, может ли столбец принимать значение **NULL**, при определении столбца ему можно задать атрибут **NULL** или **NOT NULL**. Если этот атрибут явным образом не будет использован, то по умолчанию столбец будет допускать значение NULL. Исключением является тот случай, когда столбец выступает в роли первичного ключа - в этом случае по умолчанию столбец имеет значение NOT NULL.

```
CREATE TABLE Customers
(
    Id INT PRIMARY KEY AUTO_INCREMENT,
    Age INT,
    FirstName VARCHAR(20) NOT NULL,
    LastName VARCHAR(20) NOT NULL,
    Email VARCHAR(30) NULL,
    Phone VARCHAR(20) NULL
);
```

В данном случае столбец Age по умолчанию будет иметь атрибут NULL.

DEFAULT

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.

```
CREATE TABLE Customers
(
    Id INT PRIMARY KEY AUTO_INCREMENT,
    Age INT DEFAULT 18,
    FirstName VARCHAR(20) NOT NULL,
    LastName VARCHAR(20) NOT NULL,
    Email VARCHAR(30) NOT NULL UNIQUE,
    Phone VARCHAR(20) NOT NULL UNIQUE
);
```

Здесь столбец Age в качестве значения по умолчанию имеет число 18.

ДОМАШНЕЕ ЗАДАНИЕ

- Создать 3 таблицы с первичными ключами. Таблицы авторов, книг и магазинов. В таблице книг будет 3 столбца: id, название книги, дата публикации. В таблице авторов будет 4 столбца: id, имя, фамилия, дата рождения. В таблице магазинов будет 2 столбца: id, название магазина.

- Самостоятельное изучение:
- Атрибут **CHECK и CONSTRAINT**
- Почитать про **FOREIGN KEY и нормализацию баз данных**