02_数据定义语言DDL

2024全新MySQL企业开发版

目录
Contents

1 数据定义语言概述

2 SQL命名规定和规范

3 数据定义语言之库管理

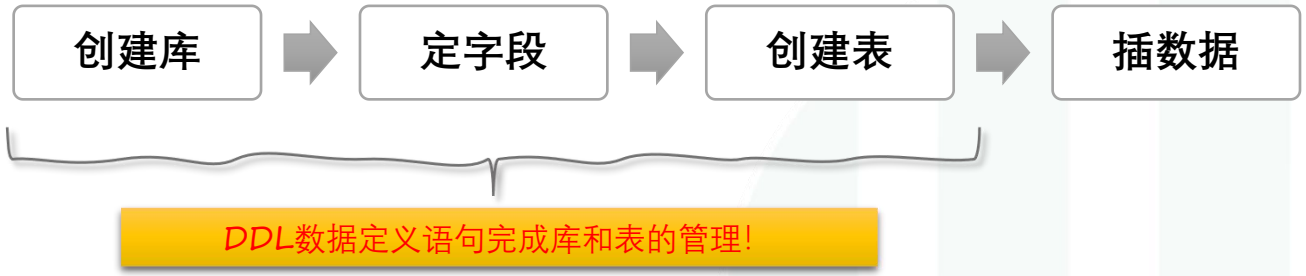
4 数据定义语言之表管理

1 数据定义语言概述

1 数据定义语言概述

一条数据的存储过程

在 MySQL 中，一个完整的数据存储过程总共有 4 步，分别：



数据库定义语句介绍

数据定义语言（Data Definition Language）。
DDL 用于定义和管理数据库的结构，包括**库**、**表**、索引、视图等数据库对象的创建、修改和删除。
DDL 不涉及对数据的操作，而是关注数据库的结构和元数据（容器）。

DDL关键字	**CREATE：** 用于创建数据库、表、索引、视图等。
	ALTER： 用于修改数据库对象的结构，如修改表结构、添加列、删除列等。
	DROP： 用于删除数据库对象，如删除表、删除索引等。

目录
Contents

1 数据定义语言概述

2 SQL命名规定和规范

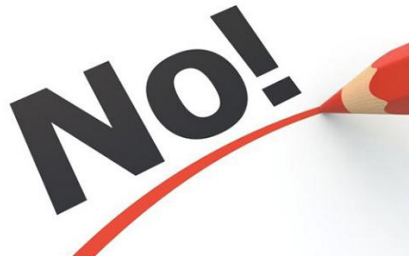
3 数据定义语言之库管理

4 数据定义语言之表管理

2 SQL命名规定和规范

2 SQL命名规定和规范

前置1：标识符命名规定



1. 数据库名、表名不得超过30个字符，变量名限制为29个
2. 必须只能包含 A–Z, a–z, 0–9, _共63个字符，而且不能数字开头
3. 数据库名、表名、字段名等对象名中间不能包含空格
4. 同一个MySQL软件中，数据库不能同名；同一个库中，表不能重名；同一个表中，字段不能重名
   https://dev.mysql.com/doc/refman/8.0/en/keywords.html
5. 必须保证你的字段没有和保留字、数据库系统或常用方法冲突。如果坚持使用，请在SQL语句中使用`（着重号）引起来

可以起的数据库名：

1. mycompanydatabase
2. sales_data
3. Customer_Records_DB
4. ecommerce_DB

不能起的数据库名：

6. my company database
7. sales&data
8. SELECT_db
9. 123_database
10. inventory database v1.0

2 SQL命名规定和规范

2 SQL命名规定和规范

前置1：标识符命名规则

2 SQL命名规定和规范

前置1：标识符命名规则

1. mycompanydatabase
2. sales_data
3. Customer_Records_DB
4. ecommerce_DB

2 SQL命名规定和规范

前置1：标识符命名规则

1. mycompanydatabase
2. sales_data
3. Customer_Records_DB
4. ecommerce_DB

可以使用

好不好

延续性

2 SQL命名规定和规范

前置1：标识符命名规则

1. mycompanydatabase
2. sales_data
3. Customer_Records_DB
4. ecommerce_DB

可以使用

好不好

延续性

原因：我们只知道规定，不知道规范！没有标准！

2 SQL命名规定和规范

前置1：标识符命名规则

1. mycompanydatabase
2. sales_data
3. Customer_Records_DB
4. ecommerce_DB

可以使用

好不好

延续性

原因：我们只知道规定，不知道规范！没有标准！

前置2：标识符命名规范（基于阿里巴巴规范手册）

阿里巴巴的 SQL 规范建议通常是为了确保数据库操作的效率、安全性和可维护性。

2 SQL命名规定和规范

前置1：标识符命名规则

1. mycompanydatabase
2. sales_data
3. Customer_Records_DB
4. ecommerce_DB

可以使用

好不好

延续性

原因：我们只知道规定，不知道规范！没有标准！

前置2：标识符命名规范（基于阿里巴巴规范手册）

阿里巴巴的 SQL 规范建议通常是为了确保数据库操作的效率、安全性和可维护性。

1. 注释应该清晰、简洁地解释 SQL 语句的意图、功能和影响。
2. 库、表、列名应该使用小写字母，并使用下划线（_)或驼峰命名法。
3. 库、表、字段名应该简洁明了，具有描述性，反映其所存储数据的含义。
4. 库名应于对应的程序名一致 例如：程序名为 EcommercePlatform 数据库名命名为 ecommerce_platform"
5. 表命名最好是遵循 “业务名称_表”的作用 例如：*alipay_task* 、 *force_project*、 *trade_config*
6. 列名应遵循“表实体_属性”的作用 例如：*product_name* 或 *productName*

2 SQL命名规定和规范

前置1：标识符命名规则

1. mycompanydatabase
2. sales_data
3. Customer_Records_DB
4. ecommerce_DB

可以使用

好不好

延续性

原因：我们只知道规定，不知道规范！没有标准！

前置2：标识符命名规范（基于阿里巴巴规范手册）

阿里巴巴的 SQL 规范建议通常是为了确保数据库操作的效率、安全性和可维护性。

1. 注释应该清晰、简洁地解释 SQL 语句的意图、功能和影响。
2. 库、表、列名应该使用小写字母，并使用下划线（_)或驼峰命名法。
3. 库、表、字段名应该简洁明了，具有描述性，反映其所存储数据的含义。
4. 库名应于对应的程序名一致 例如：程序名为 EcommercePlatform 数据库名命名为 ecommerce_platform"
5. 表命名最好是遵循 “业务名称_表”的作用 例如：*alipay_task* 、 *force_project*、 *trade_config*
6. 列名应遵循“表实体_属性”的作用 例如：*product_name* 或 *productName*

数据库名 (Database Names):
1.customer_data_db
2.ecommerce_platform_db
3.inventory_management_db
表名 (Table Names):
1.user_accounts_table
3.order_details_table
4.product_catalog_table
列名 (Column Names):
1.customer_id
4.product_name
5.order_quantity

目录
Contents

1 数据定义语言概述

2 SQL命名规定和规范

3 数据定义语言之库管理

4 数据定义语言之表管理

3 数据定义语言之库管理

3 数据定义语言之库管理

3.1 库管理：创建库

创建库，我们必须指定**库名**，可能指定**字符集**或者**排序方式**!

方式1：创建数据库，使用默认的字符集和排序方式

```
CREATE DATABASE 数据库名;
```

方式2：判断并创建默认字符集库（推荐）

```
CREATE DATABASE IF NOT EXISTS 数据库名;
```

方式3：创建指定字符集库或者排序方式

```
CREATE DATABASE 数据库名 CHARACTER SET 字符集;
CREATE DATABASE 数据库名 COLLATE 排序规则;
```

方式4：创建指定字符集和排序规则库

```
CREATE DATABASE 数据库名 CHARACTER SET 字符集 COLLATE 排序规则;
```

MySQL8默认值(不同版本可能会有不同)：

字符集：utf8mb4 是一种广泛支持各种语言字符的字符集。

排序规则：utf8mb4_0900_ai_ci 是一种不区分大小写的排序规则

```
# 查看默认字符集和排序方式命令
SHOW VARIABLES LIKE 'character_set_database';
SHOW VARIABLES LIKE 'collation_database';
```

字符集和排序规则：

字符集就是我们常说的编码格式，决定了数据如何编码存储!
排序规则决定了如何比较和排序存储在数据库中的文本数据。

常见字符集（Character Set）：

1. **utf8**：早期版本的字符集，最多3字节存储一个字符，3字节无法覆盖全部unicode编码，有显示乱码可能。
2. **utfmb4（8+默认）**：解决utf8的存储限制，使用4字节进行字符存储，可以覆盖更广 Unicode 编码，包括表情符号等等。

常见排序规则（Collate）：

1. **utf8mb4_0900_ai_ci**：UTF-8 的不区分大小写的排序规则((mysql8+的默认排序规则)。
2. **utf8mb4_0900_as_cs**：UTF-8 的 Unicode 排序规则，区分大小写！。

3.2 库管理：查看和使用库

使用和查看库，包括展示和切换库等命令

方式1：查看当前所有库

```
SHOW DATABASES ;
```

方式2：查看当前使用库

```
SELECT  DATABASE();
```

方式3：查看指定库下所有表

```
SHOW TABLES FROM 数据库名;;
```

方式4：查看创建库的信息

```
SHOW CREATE DATABASE 数据库名;
```

方式5：切换库/选中库

```
USE 数据库名;
```

注意：要操作表格和数据之前必须先说明是对哪个数据库进行操作，先use库

3.3 库管理：修改库

方式1：修改库编码字符集

```
ALTER DATABASE 数据库名 CHARACTER SET 字符集;  # 修改字符集gbk utf8
ALTER DATABASE 数据库名 COLLATE 排序方式; #修改排序方式
ALTER DATABASE 数据库名 CHARACTER SET 字符集 COLLATE 排序方式; # 修改字符集和排序方式
```

注意：

DATABASE 不能改名。一些可视化工具可以改名，它是建新库，把所有表复制到新库，再删旧库完成的。

3.4 库管理：删除库

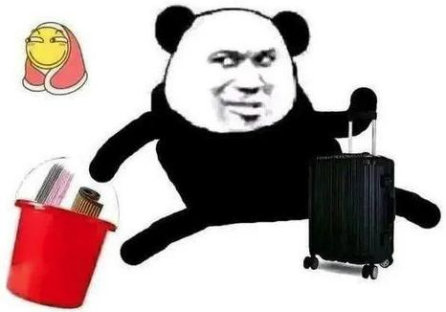
删除数据库前要三思，是不是有刁民要害朕，确认好再删除，否则真要提桶跑路！！

方式1：直接删除库

```
DROP DATABASE 数据库名;
```

方式2：判断并删除库（推荐）

```
DROP DATABASE IF EXISTS 数据库名;
```



提桶跑路

3.5 库管理实战练习

场景1：

假设你正在为一个多语言的博客平台设计数据库。你需要创建一个名为 blog_platform 的数据库，支持存储多语言的文章和评论。由于博客平台可能包含来自不同语言的用户，你决定使用 `utf8mb4`字符集，排序方式选择默认值，以支持广泛的 Unicode 字符。

场景2：

查看数据库字符集和排序规则

场景3：

假设在后续的发展中，你决定将排序方式修改为 `utf8mb4_0900_as_cs`，以实现大小写敏感的比较。

场景4：

查看修改后数据库字符集和排序规则

场景5：

项目惨遭放弃，需要删除项目库，并且跑路

目录
Contents

1 数据定义语言概述

2 SQL命名规定和规范

3 数据定义语言之库管理

4 数据定义语言之表管理

4 数据定义语言之表管理

4 数据定义语言之表管理

创建库VS创建表

创建库，相当于创建一个excel表格文件，我们只需要指定**库名和字符集**即可！！

创建表，相当于创建一个表格内容页，我们不仅需要指定**表名和字符集，**还需要指定表中的**列名和列类型，**甚至会加入一些**约束，**例如：手机号必须填写等！

例子：创建了一个名为 **posts** 的博客文章表，包括一些常见的**数据类型、编码设置以及列注释！**

```
CREATE TABLE posts (
    post_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL COMMENT 'The title of the blog post',
    content TEXT NOT NULL COMMENT 'The content of the blog post',
    author_id INT NOT NULL COMMENT 'The ID of the author of the post',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP COMMENT 'The timestamp
when the post was created',
   CONSTRAINT fk_author FOREIGN KEY (author_id) REFERENCES authors(author_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT 'Table storing blog posts';
```

4 数据定义语言之表管理

4.1 表管理：创建表

```
CREATE TABLE posts (
    post_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL COMMENT 'The title of the blog post',
    content TEXT NOT NULL COMMENT 'The content of the blog post',
    author_id INT NOT NULL COMMENT 'The ID of the author of the post',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP COMMENT 'The timestamp
when the post was created',
   CONSTRAINT fk_author FOREIGN KEY (author_id) REFERENCES authors(author_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT 'Table storing blog posts';
```

4 数据定义语言之表管理

4.1 表管理：创建表

```
CREATE TABLE 表名 (
    post_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL COMMENT 'The title of the blog post',
    content TEXT NOT NULL COMMENT 'The content of the blog post',
    author_id INT NOT NULL COMMENT 'The ID of the author of the post',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP COMMENT 'The timestamp
when the post was created',
   CONSTRAINT fk_author FOREIGN KEY (author_id) REFERENCES authors(author_id)
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT 'Table storing blog posts';
```

核心要素1：指定表名

4 数据定义语言之表管理

4.1 表管理：创建表

```
CREATE TABLE 表名 (
    列名 INT AUTO_INCREMENT PRIMARY KEY,
    列名 VARCHAR(255) NOT NULL COMMENT 'The title of the blog post',
    列名 TEXT NOT NULL COMMENT 'The content of the blog post',
    列名 INT NOT NULL COMMENT 'The ID of the author of the post',
    列名TIMESTAMP DEFAULT CURRENT_TIMESTAMP COMMENT 'The timestamp when
the post was created',
    CONSTRAINT fk_author FOREIGN KEY (author_id) REFERENCES authors(author_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT 'Table storing blog posts';
```

核心要素1：指定表名

核心要素2：指定列名

4 数据定义语言之表管理

4.1 表管理：创建表

```
CREATE TABLE 表名 (
    列名 类型 AUTO_INCREMENT PRIMARY KEY,
    列名 类型 NOT NULL COMMENT 'The title of the blog post',
    列名 类型 NOT NULL COMMENT 'The content of the blog post',
    列名 类型 NOT NULL COMMENT 'The ID of the author of the post',
    列名 类型 DEFAULT CURRENT_TIMESTAMP COMMENT 'The timestamp when the post was created',
    CONSTRAINT fk_author FOREIGN KEY (author_id) REFERENCES authors(author_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT 'Table storing blog posts';
```

核心要素1：指定表名

核心要素2：指定列名

核心要素3：指定列类型

4 数据定义语言之表管理

4.1 表管理：创建表

```
CREATE TABLE 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] COMMENT 'The title of the blog post',
    列名 类型 [列可选约束] COMMENT 'The content of the blog post',
    列名 类型 [列可选约束] COMMENT 'The ID of the author of the post',
    列名 类型 [列可选约束] COMMENT 'The timestamp when the post was created',
    [列可选约束]
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT 'Table storing blog posts';
```

核心要素1：指定表名

核心要素2：指定列名

核心要素3：指定列类型

可选要素4：指定列约束【第五章约束讲解】

4 数据定义语言之表管理

4.1 表管理：创建表

```
CREATE TABLE 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] COMMENT 'The title of the blog post',
    列名 类型 [列可选约束] COMMENT 'The content of the blog post',
    列名 类型 [列可选约束] COMMENT 'The ID of the author of the post',
    列名 类型 [列可选约束] COMMENT 'The timestamp when the post was created',
    [列可选约束]
) [表可选约束] COMMENT 'Table storing blog posts';
```

核心要素1：指定表名

核心要素2：指定列名

核心要素3：指定列类型

可选要素4：指定列约束【第五章约束讲解】

可选要素5：指定表配置

4 数据定义语言之表管理

4.1 表管理：创建表

```
CREATE TABLE 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    [列可选约束]
) [表可选约束] [COMMENT '表可选注释'];
```

- 核心要素1：指定表名
- 核心要素2：指定列名
- 核心要素3：指定列类型
- 可选要素4：指定列约束【第五章约束讲解】
- 可选要素5：指定表配置
- 可选要素6：指定表和列的注释

4 数据定义语言之表管理

4.1 表管理：创建表

```
CREATE TABLE 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'], #列之间使用，分割
    [列可选约束]
) [表可选约束] [COMMENT '表可选注释'];
```

```
CREATE TABLE [IF NOT EXISTS] 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'], #列之间使用，分割
    [列可选约束]
) [表可选约束] [COMMENT '表可选注释'];
```

4 数据定义语言之表管理

4.1 表管理：创建表

```
CREATE TABLE 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'], #列之间使用，分割
    [列可选约束]
) [表可选约束] [COMMENT '表可选注释'];
```

关键点1：**表名、列名、类型必须填写**，其他可选

```
CREATE TABLE [IF NOT EXISTS] 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'], #列之间使用，分割
    [列可选约束]
) [表可选约束] [COMMENT '表可选注释'];
```

4 数据定义语言之表管理

4.1 表管理：创建表

```
CREATE TABLE 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'], #列之间使用，分割
    [列可选约束]
) [表可选约束] [COMMENT '表可选注释'];
```

关键点1：**表名、列名、类型必须填写**，其他可选

关键点2：推荐使用**if not exists**，直接创建存在报错

```
CREATE TABLE [IF NOT EXISTS] 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'], #列之间使用，分割
    [列可选约束]
) [表可选约束] [COMMENT '表可选注释'];
```

4 数据定义语言之表管理

4.1 表管理：创建表

```
CREATE TABLE 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'], #列之间使用，分割
    [列可选约束]
) [表可选约束] [COMMENT '表可选注释'];
```

关键点1：**表名、列名、类型必须填写**，其他可选

关键点2：推荐使用**if not exists**，直接创建存在报错

关键点4：**注释**不是必须的，但是是**很有必要的**

```
CREATE TABLE [IF NOT EXISTS] 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'], #列之间使用，分割
    [列可选约束]
) [表可选约束] [COMMENT '表可选注释'];
```

4 数据定义语言之表管理

4.1 表管理：创建表

```
CREATE TABLE 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'], #列之间使用，分割
    [列可选约束]
) [表可选约束] [COMMENT '表可选注释'];
```

```
CREATE TABLE [IF NOT EXISTS] 表名 (
    列名 类型 [列可选约束],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'],
    列名 类型 [列可选约束] [COMMENT '列可选注释'], #列之间使用，分割
    [列可选约束]
) [表可选约束] [COMMENT '表可选注释'];
```

关键点1：**表名、列名、类型必须填写**，其他可选

关键点2：推荐使用**if not exists**，直接创建存在报错

关键点4：**注释**不是必须的，但是是**很有必要的**

关键点3：列之间使用,进行分割

4.2 表管理：小练习

场景1：

假设你正在设计一个简单的在线图书管理系统。需要创建一个名为 `book_libs` 的数据库,你决定使用 `utf8mb4` 字符集，排序方式选用大小写敏感的`utf8mb4_0900_as_cs`。

场景2：

创建一个图书表books，判断不存在再创建，并且手动设置books表字符集为utf8mb4，添加表注释内容 。

同时图书表books中应该以下列：

图书名称book_name列,类型为varchar(20)，添加注释。

图书价格book_price列,类型为double(4,1)，添加注释。

图书数量book_num列,类型为int，添加注释。

按以上要求完成图书表的创建!

4.3 表管理：数据类型（整数）

MySQL支持多种数据类型，包括**整数**、**浮点数**、**定点数**、**字符串**、**日期时间**等。
https://dev.mysql.com/doc/refman/8.0/en/data-types.html

MySQL支持SQL标准整数类型 INTEGER （或 INT ）和 SMALLINT 。作为标准的扩展，MySQL 还支持整数类型 TINYINT 、 MEDIUMINT 和 BIGINT 。下表显示了每种整数类型所需的存储和范围。

类型	存储（字节）	最小值有符号	最小值无符号	最大值有符号	最大值无符号
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^{63}	0	$2^{63}-1$	$2^{64}-1$

注意：无符号==无负号，整数类型都可以添加*unsigned* 修饰符，添加以后对应列数据变成无负号类型，值从0开始！！

示例：

```
stu_age tinyint  unsigned COMMENT '年龄字段，tinyint类型，无符号值从0开始',
stu_age tinyint  COMMENT '年龄字段，tinyint类型，无符号值从-128开始',
# unsigned 必须紧贴类型后放置
```

4.3 表管理：数据类型（浮点数）

FLOAT 和 DOUBLE 类型表示近似的数值数据值。MySQL 使用 4 个字节表示单精度值，使用 8 个字节表示双精度值。。下表显示了每种类型所需的存储和范围。

类型	存储（字节）	M（小数+整数位数）	D（小位数）
FLOAT(M,D)	4	M最大为24	D最大为8
DOUBLE(M,D)	8	M最大为53	D最大为30

注意： 从 MySQL 8.0.17 开始，不推荐使用非标准 FLOAT(M,D) 语法 DOUBLE(M,D)，未来版本中可能删除对它的支持。
支持unsigned修饰，添加修饰，只保留正值范围负值不会迁移到正值!

示例：

```
stu_height float(4,1)  unsigned COMMENT '身高，保留一位小数，多位会四舍五入',
```

4.3 表管理：数据类型（定点数）

DECIMAL 类型存储精确的数值数据值。当需要保持精确精度时，例如货币数据，商品价格等等!

类型	存储（字节）	M（小数+整数位数）	D（小位数）
BECIMAL(M,D)	动态计算	M最大为65	D最大为30

注意： DECIMAL 类型的存储空间是可变的，它的存储大小受 DECIMAL 数据类型定义时指定的精度和规模影响。
如果D小数位数为 0， DECIMAL 则值不包含小数点或小数部分。

示例：

```
emp_salary DECIMAL(8,1) COMMENT '工资，保留一位小数，多位会四舍五入',
```

4.3 表管理：数据类型（字符串）

CHAR和VARCHAR类型都可以存储比较短的字符串

字符串(文本)	特点	长度	长度范围（字符）	存储空间
CHAR(M)	固定长度	M	0 <= M <= 255	M*4个字节（utf8mb4）
VARCHAR(M)	可变长度	M	MySql一行数据最多65535字节	(M*4+1) 个字节（utf8mb4）

注意： CHAR(M) 类型一般需要预先定义字符串长度。如果不指定(M)，则表示长度默认是1个字符。
保存数据的实际长度比CHAR类型声明的长度小，则会在右侧填充空格以达到指定的长度。
当MySQL检索CHAR类型的数据时，CHAR类型的字段会去除尾部的空格。

VARCHAR(M) 定义时，必须指定长度M，否则报错。
MySQL4.0版本以下，varchar(20)：指的是20字节，如果存放UTF8汉字时，只能存6个（每个汉字3字节）；
MySQL5.0版本以上，varchar(20)：指的是20字符。
检索VARCHAR类型的字段数据时，会保留数据尾部的空格。

通过显示将各种字符串值存储到 CHAR(4) and VARCHAR(4) 列的结果（假设使用单字节字符集，如 latin1 ）说明 和 VARCHAR 之间的 CHAR 差异。

插入值	CHAR(4)	所需存储	VARCHAR(4)	所需存储
''	' '	4 bytes	''	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes

4.3 表管理：数据类型（讨论M）

示例1：

```
CREATE TABLE 表名 (
    post_name  VARCHAR(16384) NOT NULL
) ENGINE=INNODB CHARSET=utf8mb4;

错误代码： 1074
Column length too big for column 'post_name' (max = 16383); use BLOB or TEXT instead
```

前提：在MySQL的设定中，单行数据最大能存储65535字节数据（注意：是一行非额外存储所有列总和且单位是字节）
原因：字符集为utf8mb4，每个字符占4个字节
字符串列需要1个额外字节空间记录是否为空
计算（65535-1）/ 4 = 16383.5 向下取余 16383，单列utf8mb4字符集类型最大字符限制16383

示例2：

```
CREATE TABLE 表名 (
    post_name  VARCHAR(16000) NOT NULL,
    post_name2  VARCHAR(384) NOT NULL
) ENGINE=INNODB CHARSET=utf8mb4;

错误代码： 1118
Row size too large. The maximum row size for the used table type, not counting BLOBs, is 65535. This includes storage overhead, check the manual. You have to change some columns to TEXT or BLOBs
```

解决：在MySQL的设定中，单行数据最大能存储65535字节数据，但是**TEXT和BLOB**类型不计算，他们属于额外存储!
这种场景可以使用TEXT进行某列类型替换即可!

4.3 表管理：数据类型（文本类型）

在MySQL中，TEXT用来保存文本类型的字符串，总共包含4种类型，分别为TINYTEXT、TEXT、MEDIUMTEXT 和 LONGTEXT 类型。
在向TEXT类型的字段保存和查询数据时，系统自动按照实际长度存储，不需要预先定义长度。这一点和 VARCHAR类型相同。

文本字符串类型	特点	长度(字符)	存储范围（字节）	占用的存储空间
TINYTEXT	小文本、可变长度	L	0 <= x <= 255	L + 2 个字节
TEXT	文本、可变长度	L	0 <= x <= 65535	L + 2 个字节
MEDIUMTEXT	中等文本、可变长度	L	0 <= x <= 16777215	L + 3 个字节
LONGTEXT	大文本、可变长度	L	0 <= x<= 4294967295	L + 4 个字节(最大4g)

示例：

```
CREATE TABLE 表名(
    tx TEXT
);
```

开发经验：

短文本，固定长度使用char　　例如：性别，手机号
短文本，非固定长度使用varchar 例如：姓名，地址
大文本，建议存储到文本文件，使用varchar记录文件地址，不使用TEXT，直接存储大文本，他性能非常较差！

4.3 表管理：数据类型（时间类型）

用于表示时态值的日期和时间数据类型为 DATE 、TIME、DATETIME、TIMESTAMP 和 YEAR 。每种类型都有一个有效值范围，换一种思路，可以理解时间类型就是`特殊格式的字符串`

类型	名称	字节	日期格式	小值	最大值
YEAR	年	1	YYYY或YY	1901	2155
TIME	时间	3	HH:MM:SS	-838:59:59	838:59:59
DATE	日期	3	YYYY-MM-DD	1000-01-01	9999-12-03
DATETIME	日期时间	8	YYYY-MM-DD HH:MM:SS	1000-01-01 00:00:00	9999-12-31 23:59:59
TIMESTAMP	日期时间	4	YYYY-MM-DD HH:MM:SS	1970-01-01 00:00:00	2038-01-19 03:14:07

注意：year类型赋00-99值对应年限，[00-69]对应[2000-2069],[70-99]对应[1970-1999],建议四位年值!
默认情况下，时间需要主动赋予默认值和修改值!

扩展：DATETIME和TIMESTAMP类型自动初始化和更新

```
# 方式1：  插入默认当前时间和修改自动更新当前时间
ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
dt DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
# 方式2：  插入默认当前时间
ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
dt DATETIME DEFAULT CURRENT_TIMESTAMP
```

4.4 表管理：数据类型总结

整数类型（Integer Types）
- INT：范围为 -2147483648 到 2147483647。
- TINYINT：范围为 -128 到 127。
- SMALLINT：范围为 -32768 到 32767。
- BIGINT：范围为 -9223372036854775808 到 9223372036854775807。

非整数类型（Floating-Point Types）
- FLOAT：单精度浮点数。
- DOUBLE：双精度浮点数。
- DECIMAL：固定精度十进制数。

字符串类型（String Types）
- VARCHAR：可变长度的字符串。
- CHAR：固定长度的字符串。
- TEXT：较大的文本数据（最大4G）。

日期和时间类型（Date and Time Types）
- DATE：日期值。
- TIME：时间值。
- DATETIME：日期和时间值。
- TIMESTAMP：日期和时间值，具有特殊的自动更新功能。
- YEAR：年份值。

确定数据值范围，选择符合范围且存储空间占有最小类型

不确定数据值范围，选择选择范围较大类型,避免值超范围异常

4.4 表管理：继续练习

场景1：

假设你正在设计一个简单的在线图书管理系统。需要创建一个名为 `book_libs` 的数据库,你决定使用 `utf8mb4` 字符集，排序方式选用大小写敏感的`utf8mb4_0900_as_cs`。

场景2：

创建一个学生表(student)来存储借书的学员信息，其中应包含学生姓名、年龄、身高、生日以及注册时间和更新时间等属性。

4.5 表管理：修改表

修改表中列（字段）

```
# 修改表，添加一列[可以指定X字段前或者后]
ALTER TABLE 表名 ADD 字段名 字段类型 【FIRST|AFTER 字段名】;
# 修改表，修改列名
ALTER TABLE 表名 CHANGE 原字段名 字段名 新字段类型 【FIRST|AFTER 字段名】;
# 修改表，修改列类型
ALTER TABLE 表名 MODIFY 字段名 新字段类型 【FIRST|AFTER 字段名】;
# 修改表，删除一列
ALTER TABLE 表名 DROP 字段名;
```

修改表名

```
# 修改表名
ALTER TABLE 表名 RENAME [TO] 新表名;
```

4.6 表管理：删除表

删除数据表

```
# 删除表
DROP TABLE [IF EXISTS] 数据表1 [, 数据表2, …, 数据表n];
```

清空表数据

```
# 清空表数据
TRUNCATE TABLE 表名;
```

注意: 删除表和清空表数据命令都是无法回滚的！动作不可以，执行之前需三思！

施主三思啊

4.7 综合练习

字段名	数据类型
emp_num	int(11)
last_name	varchar（50）
first_name	varchar(50)
mobile	varchar(25)
code	int
job_title	varchar(50)
birth	date
note	varchar(255)
sex	varchar(5)

要求1：创建表格employees
要求2：将表employees的mobile字段修改到code字段后面。
要求3：将表employees的birth字段改名为birthday;
要求4：修改sex字段，数据类型为char（1）。
要求5：删除字段note;
要求6：增加字段名favoriate_activity，数据类型为varchar（100）；
要求7：将表employees的名称修改为 employees_info

感谢观看