一、数据库的操作

目标

- 会使用SQL语句实现**数据库**的创建、使用、修改、删除操作

---

1. 主要操作

- 创建数据库
- 使用数据库
- 修改数据库
- 删除数据库

2.创建数据库

- 语法格式

```
#创建数据库
create database 数据库名 [charset] [字符编码] [collate] [校验规则];
#创建结果查看
show create database 数据库名;
```

- 举例说明

```
#创建一个叫python的数据库:
create database python charset=utf8 collate=utf8_general_ci;
#查看创建结果
show create database python;
```

3.使用数据库

- 语法格式

```
#使用（打开）数据库
use 数据库名;
#查看当前使用的数据库
select database();
```

- 举例说明

```
#使用（打开）python的数据库:
use python;
```

4.修改数据库

- 语法格式

```
alter database [数据库名]
[ default ] character set <字符集名>
[ default ] collate <校对规则名>;
```

- 举例说明

```
#创建testpython数据库，字符集为gb2312
create database testpython charset = gb2312
#修改testpython的指定字符集修改为 utf8mb4，默认校对规则修改为utf8mb4_general_ci：
alter database testpython
default character set utf8mb4
default collate utf8mb4_general_ci;
```

5.删除数据库

- 语法格式

```
drop database 数据库名;
```

- 举例说明

```
#删除python数据库
drop database python;
```

6.数据库其他操作

- 查看所有数据库

```
show databases;
```

- 备份数据库

```
mysqldump -uroot -p 数据库名 > python.sql;

# 按提示输入mysql的密码
```

- 恢复数据库

```
mysql -uroot -p 新数据库名 < python.sql

# 根据提示输入mysql密码
```

二、数据表操作

目标

- 会使用SQL语句实现**数据表**的新增、查询、删除操作

1. 创建表

- 语法格式

```
#创建表
create table 表名(
    字段名 类型 约束,
    字段名 类型 约束
    ...
)
#查看创建的表
show create table 表名;
```

- 举例说明

例：创建学生表，字段要求如下：

姓名(长度为10)

```
create table students(
    name varchar(10)
)
```

例：创建学生表，字段要求如下：

姓名(长度为10)， 年龄

```
create table students(
    name varchar(10),
    age int unsigned
)
```

例：创建学生表，字段要求如下：

姓名(长度为10)， 年龄，身高(保留小数点2位)

```
create table students(
    id int unsigned primary key auto_increment,
    name varchar(20),
    age int unsigned,
    height decimal(5,2)
)
```

2.查看表结构

- 语法格式

```
desc 表名;
```

- 举例说明

```
desc students;
```

3. 删除表

- 语法格式

```
格式一：drop table 表名;

格式二：drop table if exists 表名;
```

- 举例说明

例：删除学生表

```
drop table students;
或
drop table if exists students;
```

三、数据操作-增删改

目标

- 会使用SQL语句实现**数据**的新增、修改、删除操作

1. 添加数据

1.1 添加一行数据

格式一：所有字段设置值，值的顺序与表中字段的顺序对应

- 说明：主键列是自动增长，插入时需要占位，通常使用0或者 default 或者 null 来占位，插入成功后以实际数据为准

```
insert into 表名 values(...)
```

例：插入一个学生，设置所有字段的信息

```
insert into students values(0,'亚瑟',22,177.56)
```

格式二：部分字段设置值，值的顺序与给出的字段顺序对应

```
insert into 表名(字段1,...) values(值1,...)
```

例：插入一个学生，只设置姓名

```
insert into students(name) values('老夫子')
```

1.2 添加多行数据

方式一：写多条insert语句，语句之间用英文分号隔开

```
insert into students(name) values('老夫子2');
insert into students(name) values('老夫子3');
insert into students values(0,'亚瑟2',23,167.56)
```

方式二：写一条insert语句，设置多条数据，数据之间用英文逗号隔开

```
格式一：insert into 表名 values(...),(...)...
例：插入多个学生，设置所有字段的信息
insert into students values(0,'亚瑟3',23,167.56),(0,'亚瑟4',23,167.56)
```

```
格式二：insert into 表名(列1,...) values(值1,...),(值1,...)...
例：插入多个学生，只设置姓名
insert into students(name) values('老夫子5'),('老夫子6')
```

2. 修改字段值

- 语法格式

```
update 表名 set 列1=值1,列2=值2... where 条件
```

- 举例说明

例：修改id为5的学生数据，姓名改为 狄仁杰，年龄改为 20

```
update students set name='狄仁杰',age=20 where id=5
```

3. 删除表记录

- 语法格式

```
格式一：delete from 表名 where 条件;
```

- 举例说明

例：删除id为6的学生数据

```
delete from students where id=6;
```

逻辑删除：对于重要的数据，不能轻易执行 delete 语句进行删除。因为一旦删除，数据无法恢复，这时可以进行逻辑删除。

1、给表添加字段，代表数据是否删除，一般起名 isdelete，0代表未删除，1代表删除，默认值为0

2、当要删除某条数据时，只需要设置这条数据的 isdelete 字段为1

3、以后在查询数据时，只查询出 isdelete 为0的数据

```
例：
1、给学生表添加字段(isdelete)，默认值为0，
如果表中已经有数据，需要把所有数据的isdelete字段更新为0
update students set isdelete=0
2、删除id为1的学生
update students set isdelete=1 where id=1
3、查询未删除的数据
select * from students where isdelete=0
```

```
格式二：truncate table 表名（删除表的所有数据，保留表结构）
```

例：删除学生表的所有数据

```
truncate table students
```

```
格式三：drop table 表名（删除表，所有数据和表结构都删掉）
```

例：删除学生表

```
drop table students
```

Truncate、Delete、Drop 的区别

1、Delete 删除数据时，即使删除所有数据，其中的自增长字段不会从1开始

2、Truncate 删除数据时，其中的自增长字段恢复从1开始

3、Drop 是删除表，所有数据和表结构都删掉

四、数据操作-查询

目标

- 熟练使用SQL语句实现**数据**的查询操作

1. 基础查询

1.1查询所有字段

- 语法格式

  ```
  select * from 表名;
  ```

- 举例说明

  ```
  #例：查询所有学生数据
  select * from students;
  ```

1.2查询部分字段

- 语法格式

```
select 字段1,字段2,... from 表名;
```

- 举例说明

```
#例：查询所有学生的姓名、性别、年龄
select name,sex,age from students
```

1.3起别名

- 语法格式

```
#给表起别名，在多表查询中经常使用
select 别名.字段1,别名.字段2,... from 表名 [as] 别名;

#给字段起别名，这个别名出现在结果集中
select 字段1 as 别名1,字段2 as 别名2,... from 表名
```

- 举例说明

```
#例：给学生表起别名
select s.name,s.sex,s.age from students as s;
```

```
#例：查询所有学生的姓名、性别、年龄，结果中的字段名显示为中文
select name as 姓名,sex as 性别,age as 年龄 from students;
```

1.4去重

- 语法格式

```
select distinct 字段1,... from 表名;
```

- 举例说明

```
#例：查询所有学生的性别，不显示重复的数据
select distinct sex from students;
```

2.复杂查询

> 在基础查询基础上，根据需求描述关系进行查询
>
> 实际应用中，往往是多种复合查询的组合使用

- 条件查询：按照一定条件筛选需要的结果
- 排序：按照一定的排序规则筛选所需结果
- 聚合函数：对一组数据进行计算得到一个结果的实现方法
- 分组：在同一属性（字段）中，将值相同的放到一组的过程
- 分页：对大批量数据进行设定数量展示的过程
- 连接查询：将不同的表通过特定关系连接的过程
- 自关联：将同一表通过特定关系连接的过程
- 子查询：在一个查询套入另一个查询的过程