## 数据库操作

1 新增数据库

基本语法：create database + 数据库名称 + [库选项];

其中，库选项是用来约束数据库的，为可选项（有默认值），共有两种，分别为：

字符集设定：charset/ character set+ 具体字符集，用来表示数据存储的编码格式，常用的字符集包括GBK和UTF8等。

校对集设定：collate+ 具体校对集，表示数据比较的规则，其依赖字符集。

示例：create database TBL\_ERROR\_CODE charset utf8;

其中，数据库的名字不能用关键字（已经被占用的字符，例如 update 和 insert 等）或者保留字（将来可能会用的，例如 access 和 cast 等）。

如果非要使用数据库的关键字或者保留字作为数据库名称，那么必须用反引号将其括起来，例如：

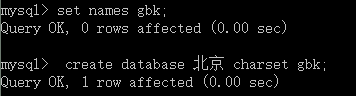
create databaseaccesscharset utf8;

如果还想使用中文作为数据库的名称，那就得保证数据库能够识别中文（强烈建议不要用中文命名数据库的名称），例如：

-- 设置中文名称的方法，其中 gbk 为当前数据库的默认字符集

set names gbk;

create database 北京 charset gbk;

[](https://camo.githubusercontent.com/8dd3f40e2ed88749701f98783825c59a2a688846/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730353033313330393434323733)

2 查询数据库

查看全部 --> 基本语法：show databases;

查看部分（模糊查询）--> 基本语法：show databases like 'pattern';

其中，pattern是匹配模式，有两种，分别为：

%：表示匹配多个字符；

\_：表示匹配单个字符。

此外，在匹配含有下划线\_的数据库名称的时候，需要在下划线前面加上反斜线\\_进行转义操作。

示例：show databases like 'TBL%';表示匹配所有TBL开头的数据库。

查看数据库的创建语句 --> 基本语法：show create database + 数据库名称;

在这里，查看的结果有可能与咱们书写的 SQL 语句不同，这是因为数据库在执行 SQL 语句之前会优化 SQL，系统保存的是优化后的结果。

3 更新数据库

在这里，需要注意：数据库的名字不可以修改。

数据库的修改仅限库选项，即字符集和校对集（校对集依赖字符集）。

基本语法：alter database + 数据库名称 + [库选项];

charset/character set[=] 字符集;

collate[=] 校对集;

示例：alter database TBL\_ERROR\_CODE charset gbk;表示修改此数据库的字符集为gbk.

4 删除数据库

基本语法：drop database + 数据库名称;

在这里，需要注意：在删除数据库之前，应该先进行备份操作，因为删除为不可逆操作，所以不要随意删除数据库。

## 数据表操作

1 新增表

基本语法：

create table [if not exists] + 表名(

字段名称 数据类型,

……

字段名称 数据类型 /\* 最后后一行，不需要加逗号 \*/

)[表选项];

其中，if not exists表示

如果表名不存在，就执行创建代码；如果表名存在，则不执行创建代码。

表选项则是用来控制表的表现形式的，共有三种，分别为：

字符集设定：charset/ character set+ 具体字符集，用来表示数据存储的编码格式，常用的字符集包括GBK和UTF8等。

校对集设定：collate+ 具体校对集，表示数据比较的规则，其依赖字符集。

存储引擎：engine+具体存储引擎，默认为InnoDB，常用的还有MyISAM.

由于任何表都归属于某个数据库，因此在创建表的时候，都必须先指定具体的数据库。在这里，指定数据库的方式有两种，分别为：

第 1 种：显式的指定表所属的数据库，示例

create table if not exists test.student(

name varchar(10),

age int, /\* 整型不需要指定具体的长度 \*/

grade varchar(10) /\* 最后后一行，不需要加逗号 \*/

)charset utf8;

第 2 种：隐式的指定表所属的数据库，示例

use test; /\* use + 数据库名称，表示切换到指定的数据库，这句命令其实不加分号也可以，但不建议这么做 \*/

create table if not exists student(

name varchar(10),

age int, /\* 整型不需要指定具体的长度 \*/

grade varchar(10) /\* 最后后一行，不需要加逗号 \*/

)charset utf8;

2 查询表

查看全部 --> 基本语法：show tables;

查看部分（模糊查询）--> 基本语法：show tables like 'pattern';

其中，pattern是匹配模式，有两种，分别为：

%：表示匹配多个字符；

\_：表示匹配单个字符。

此外，在匹配含有下划线\_的表名的时候，需要在下划线前面加上反斜线\\_进行转义操作。

示例：show tables like '%t';表示匹配所有以t结尾的表。

查看表的创建语句 --> 基本语法：show create table + 表名;

在这里，咱们也可以用\g和\G代替上述语句中的;分号，其中\g等价于分号，\G则在等价于分号的同时，将查的表结构旋转90度，变成纵向结构。

查看表中的字段信息 --> 基本语法：desc/describe/show columns from + 表名;

3 更新表

在这里，需要注意：表的修改，分为修改表本身和修改表中的字段。

第 1 类：修改表本身

修改表名，基本语法：rename table 旧表名 to 新表名;

修改表选项，基本语法：alter table + 表名 + 表选项[=] + 值;

第 2 类：修改表中的字段，新增、修改、重命名和删除

新增字段，基本语法：alter table + 表名 + add + [column] + 字段名 + 数据类型 + [列属性][位置];

其中，位置表示此字段存储的位置，分为first（第一个位置）和after + 字段名（指定的字段后，默认为最后一个位置）.

示例：alter table student add column id int first;

修改字段，基本语法：alter table + 表名 + modify + 字段名 + 数据类型 + [列属性][位置];

其中，位置表示此字段存储的位置，分为first（第一个位置）和after + 字段名（指定的字段后，默认为最后一个位置）.

示例：alter table student modify name char(10) after id;

重命名字段，基本语法：alter table + 表名 + change + 旧字段名 + 新字段名 + 数据类型 + [列属性][位置];

其中，位置表示此字段存储的位置，分为first（第一个位置）和after + 字段名（指定的字段后，默认为最后一个位置）.

示例：alter table student change grade class varchar(10);

删除字段，基本语法：alter table + 表名 + drop+ 字段名;

示例：alter table student drop age;

注意：如果表中已经存在数据，那么删除该字段会清空该字段的所有数据，而且不可逆，慎用。

4 删除表

基本语法：

-- 可以一次删除多张表

drop table + 表1, 表2 ... ;

在这里，需要注意：此删除为不可逆操作，希望大家谨慎使用。

# 数据操作

查询

查看全部 --> 基本语法：select \* from + 表名 + [where 条件];

示例：select \* from test;

查看部分 --> 基本语法：select + 字段名称[,字段名称] + from + 表名 + [where 条件];

示例：select name,age,grade from test where age = '18';

select 查询字段(函数) from 表名

select \* from `students` 查询students表中所有的字段

select `name` from `students` 查询students表中的name字段

select `name`,`age`,`address` from `students` 查询students表中的name,age,address字段

count（\*） //统计条数

sum (字段) //求和

avg（字段）//求平均值

max（字段）//求最大值

min （字段）//求最小值

select count(\*) as a from `students` 统计students表中的数据数量

select sum(weight) from `students` 统计students表中weight字段的总和

select avg(age) from `students` 统计students中年龄的平均值

select max(height) from `students` 统计下students中身高的最大值

select min(height) from `students` 统计下students中身高的最小值

查询条件

select 字段 from 表名 where 条件

常用的条件:

1. = 等于 2.<>不等于 <= >= 大于等于、小于等于 3. in 包含 4. not in 不包含 5.like 模糊匹配 6.between 范围 7.not between 不在范围

常用的条件运算

and 逻辑与 or 逻辑或 ()

select \* from `students` where `height`=175 查询身高等于175的人

select \* from `students` where `address` <> '河南安阳' 查询下地址不是河南安阳的

select \* from `students` where `age` in (18,22) 查询下年龄里面包含18和22的

select \* from `students` where `age` not in (18,22) 不包括18,22的年龄

模糊匹配当中：%匹配一个或者多个字符 \_ 匹配一个字符 [charlist] 字符列中的任何一个字符

[^charlist] 不在字符列中的任何单一字符 [!charlist]效果一样

select \* from `students` where `name` like '李%' 查询姓李的

select \* from `students` where `weight` between 80 and 120 //查询体重在80-120之间

select \* from `students` where `weight` not between 80 and 120 //查询不在80-120体重的人

select \* from `students` where `height` = 175 and `weight` = 120 //查询下标准身材的人

select \* from `students` where `height` = 175 or `weight` = 120 //满足任何一个即可

查询结果排序、分组

select `age` from `students` where 1 group by `age` //分组查询，一般用在sum合计的时候，对结果进行分组

select \* from `students` order by `age` asc //正向排序ASC

select \* from `students` order by `age` desc //倒排DESC

select \* from `students` order by `id` limit 1,5 //limit 从哪儿开始,查询几条 常用与分页

添加

对于数据的新增操作，有两种方法。

第 1 种：给全表字段插入数据，不需要指定字段列表，但要求数据的值出现的顺序必须与表中的字段出现的顺序一致，并且凡是非数值数据，都需要用引号（建议使用单引号）括起来。

基本语法：insert into + 表名 + values(值列表)[,(值列表)];

示例：insert into test valus('charies',18,'3.1');

第 2 种：给部分字段插入数据，需要选定字段列表，字段列表中字段出现的顺序与表中字段的顺序无关，但值列表中字段值的顺序必须与字段列表中的顺序保持一致。

基本语法：insert into + 表名(字段列表) + values(值列表)[,(值列表)];

示例：insert into test(age,name) valus(18,'guo');

insert into 表名 (字段1，字段2) value (值1，值2)

insert into 表名 values (值1，值2)

insert into 表名 set 字段1=值1,字段2=值2

Insert into `students` (`name`,`sex`,`age`,`height`,`weight`,`phone`,`address`,`email`,`birthday`)

values ('于老师','女',25,160,80,13929364450,'河南郑州','14833@qq.com','1989-12-30')

更新

基本语法：update + 表名 + set + 字段 = 值 + [where 条件];

示例：update test set age = 20 where name = 'guo';

在这里，建议尽量加上where条件，否则的话，操作的就是全表数据。

此外，判断更新操作是否成功，并不是看 SQL 语句是否执行成功，而是看是否有记录受到影响，即affected的数量大于1时，才是真正的更新成功。

update 表名 set 字段=值,字段=值 where 条件

//将王振强的身高改为180

update `students` set `height`=180 where `name`='王振强'

删除

基本语法：delete from + 表名 + [where 条件];

示例：delete from test where grade = '3.1';

当然，我们也可以用drop来实现删除操作，不过与delete相比，drop的威力更强，其在执行删除操作的时候，不仅会删除数据，还会删除定义并释放存储空间；而delete在执行删除操作的时候，仅会删除数据，并不会删除定义和释放存储空间。

delete from 表名 where 条件

delete from `students` where `name`='于老师'