**MySQL**

————————————————————————————————

>>数据库概念及作用（第一课）

- 知道数据库和数据库管理系统的关系

数据管理系统，是为管理数据而设计的软件系统，包括三大部分构成

1.数据库文件集合，主要是一系列的数据文件，作用是存储数据

2.数据库服务端，主要负责对数据文件以及文件中的数据进行管理

3.数据库客户端，主要负责和服务端通信，向服务端传输数据或从服务端获取数据

—————————————————————————

>>Windows下的mysql命令（第二课）

**- win+r , services.msc = 服务器名**

————————————————————————————————

1.net start mysql80  **- 开启服务器**

- net start [服务器名]

2.net stop 服务器名  **- 关闭服务器**

3.mysql -h localhost -u root -p  **- 登入服务器**

- mysql -h [服务器ip] -u [mysql用户名] -p

- 跑到mysql安装目录的bin文件下使用

- h : ip地址

- u : 用户

- p : 密码

————————————————————————————————

4.show databases; **- 显示所有数据库**

5.use [数据库名称]； **- 跳转到指定数据库**

6.show tables;  **- 查询当前数据库下的所有表**

————————————————————————————————

7.exit  **- 退出服务器**

8.quit  **- 退出服务器**

—————————————————————————

>>数据库完整性和约束（第三课）

- 知道 primary key , not null , unique , default 约束的作用

**1.实体完整性**

**- 主要用于保证操作的数据（记录）非空，唯一且不重复**

**- 主键索引，不重复不可以为空**

**- 唯一索引，不重复可以为空**

**2.域完整性**

**- 列必须满足某种特定的数据类型或约束**

**3.参照完整性**

**- 在更新，插入，删除记录时，如果只改其一，就会影响数据的完整性**

**如删除父表的某个记录后，子表的响应记录未删除**

**4.用户定义完整性**

**- 字段的值域，字段类型和字段的有效规则等约束**

约束类型 约束说明

1.实体完整性

PRIMARY KEY **主键约束(唯一性，非空性)**

UNIQUE KEY **唯一约束(唯一性，可以空，但只能有一个)**

————————

2.域完整性约束

NOT NULL  **非空约束(该字段不能为空)**

DEFAULT **默认约束(该数据的默认值)**

————————

3.参照完整性

FOREIGN KEY **外键约束(需要建立两表间关系)**

————————

4.用户定义完整性

字典的值必须是设定的范围之内

—————————————————————————

>>客户端Navicat使用（第四课）

- 能够使用 Navicat 创建数据库

- 能够使用 Navicat 在数据库中创建数据表

- 能够使用 Navicat 在数据库中添加数据

—————————————————————————

>>MySQL数据类型（第五课）

- 知道常用的数据类型：int,decimal,varchar,time

**1.整形类型**

TINYINT - 128 —— 127

SMALLINT - 32768 —— 32767

MEDIUMINT - 8388608 —— 8388607

INT - 2147483648 —— 2147483647

BIGINT - ？？？ —— ？？？

**2.浮点型**

float : 单精度型：只保证6位有效数字的准确性

double : 双精度型：只保证16位有效数字的准确性

- 定义方式：

- float(M,D)

- double(M,D)

- float(5,2) = 000.00 - 999.99

**3.定点数**

- 保存精确的小数

- decimal(M,D)

**4.字符串**

- 1.char : 最多保存 255 个字符（英文/中文/符号） - 定长字符串

- 2.varchar : 最多保存 65535 个字节 - 边长字符串

- UTF8 = 3字节/字符

- GBK = 2字节/字符

tinytxet = 255字节

text = 65535字节

mediumtext = 16777215字节

longtext = 4294967295字节

**5.枚举类型**

- 1.在定义字段时就规定好固定的几个值，然后插入记录时选一个

- 2.语法：

- gender enum("男","女","妖")

- 一个 enum 最多可以设置 65535 个值

**6.时间类型**

- 1.date : 年-月-日 2010-10-10

- 2.datetime : 年-月-日 时-分-秒 2010-10-10 10:10:10

- 3.timestamp : 年-月-日 时-分-秒 2010-10-10 10:10:10

- 4.time : 时:分:秒 10:10:10

- 5.year : 年 2010

datetime 与 timestamp 区别：

datetime:1000-01-01 00:00:00 —— 9999-12-31 23:59:59

timestamp:1970-01-01 00:00:01 —— 2038-01-19 03:14:07

—————————————————————————

>>MySQL基本操作（第六课）

查看( show ),创建( create ),删除( drop ),使用( use )

1.select version( ); **显示 MySQL 版本**

2.select now( ); **显示当前时间**

3.create database 名字; **创建数据库**

4.drop database 名字; **删除数据库**

5.use 名字; **切换数据库**

6.select database(); **查看当前选择的数据库**

7.show databases; **查看所有数据库**

8.show create database 名字; **查看数据库创建过程**

9.create database 名字 charset=utf8; **指定创建格式**

10.show tables; **查看当前数据库中所有表**

—————————————————————————

>>表结构的创建（第七课）

- 知道查看( show ) , 创建( create ) , 显示表字段( desc )命令格式

字段名称 数据类型 可选的约束条件

1.show tables ; **查看当前数据库中所有表**

2.int unsigend **无符号整型**

3.auto\_increment **表示自动增长**

4.not null **表示不能为空**

5.primary key **表示主键**

6.default **默认值**

7.create table 数据包名字 (字段 类型 约束[,字段 类型 约束]);

8.create monika(

字段名称 类型(int之类) 有无符号

);

9.desc 数据包的名字 **数据包的内容定义**

例子：

create table classes(

id int unsigned primary key auto\_increment,

name varchar(10) not null,

num tinyint

);

10.unsignde **应该是不出现标点符号**

11.show create table 名字 \G; **查看表创建过程**

—————————————————————————

>>表结构的修改（第八课）

- 知道select database表中添加( add ),修改字段名( change ),修改字段类型( modify ),删除( drop )

**1.添加字段**

alter table 表名 add 列表 类型;

例子:alter table students add birthday varchar(50) not null

**2.重命名字段 - 也可修改字段类型**

alter table 表名 change 原名 新名 类型及约束

例子：alter table students change birthdays birthday datetime;

**3.修改字段类型**

alter table 表名 modify 列名 类型及约束;

例子：alter table students modify birthday date not null;

**4.删除字段**

alter table 表名 drop 列名

例子：alter table students drop birthday;

**5.删除表**

drop table 表名；

drop database 数据库名；

—————————————————————————

>>表结构的增删改查（第九课）

- 知道 select insert update delete 能够增删改查数据库中的数据

创建 ( select )

更新 ( update )

读取 ( retrieve/read )

删除 ( delete )

**1.查询 select**

- 查询所有列

select \* from 表名；

例子：select \*from classes;

select \* from students where name = "二哥"; - 指定查询

select id,name from students; - 查询指定信息

select id as "编号",name as "名字" from students; - 指定编号别名查询

**2.插入字段 insert**

insert into 列名 values(1 , python20)

例子：insert into classes values(3028,"莫妮卡","18") - 指定插入

insert into students(id,name) values(null,"人"); - 指定插入单行

insert into students(id,name) values(null,"人"),(null,"大人"); - 指定插入多行

**3.修改 update**

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

例子：update students set name = 'monika' where name = '人';

update students set high = 1.2 , gender = "女" where name='二哥';

- 修改多行

**4.删除 delete**

格式：delete from 表名 [ where 条件判断 ] - 物理删除

例子：delete from students where id=1; - 物理删除

**5.逻辑删除**

- 用一个字段表示，这一条信息已经不在使用了

- 给 students 表添加一个 is\_delete 字段 bit 类型 默认为0

bit - 类型,只能保存 0 或者 1

alter table students add is\_delete bit default 0; - 拼写逻辑

update students set is\_delete=1 where id=3; - 更改数值

select \* from students where is\_delete = 0; - 查找数值

**default - 默认值**

总结：好像只是给字段加了一条信息，default(默认值) 与 bit(0,1) 更改默认值的bit，

将其更改为0好像就是逻辑删除?

**加了个字段做了个区分......**

**6.备份数据库**

mysqldump -u root -p 库名 > 名字.sql

- 备份到当前目录下 bin

**7.导入数据库**

mysql -u root -p 前面创建的库名 < 存放的名字.sql;

—————————————————————————

>>查询语句 - 练习（第十课）

1.select id 空格 "编号",name 空格 "名字" from students; - 指定编号别名查询

2.select students.id,students.name from students; - 查询指定信息

3.select s.id,s.name from students as s; - 查询指定信息 - 更改列表名查询

————-------————

**1.消除重复行(查性别)**

- distinct

select distinct 列 from 表名;

select distinct gender from students;

—————————————————————————

>>where 之比较运算（第十一课）

- 能够说出比较常见的运算符，**结果为 true 的行为会出现在结果中**

- where:

- 比较运算符

- 逻辑运算符

- 模糊查询

- 范围查询

- 空判断

1.比较运算符

等于: =

大于: >

大于等于: >=

小于: <

小于等于: <=

不等于: != 或 <>

查询标号大于3的学生

select \* from students where id > 3;

查询标号不大于4的学生

select \* from students where id <= 3;

查询名字不是monika的学生

select \* from students where name != "monika";

查询没被删除的学生

select \* from students where is\_delete = 0;

—————————————————————————

>>where 之逻辑运算（第十二课）

- 能够说出逻辑运算符 and,or,not 的作用

查询标号大于3的女学生

select \* from students where id > 3 and gender = "女";

查询标号小于4或没被删除的学生

select \* from students where id <4 or is\_delete = 1

—————————————————————————

>>where 之模糊查询（第十三课）

- 能够说出 like 的使用

**1. like**

**2. % 表示任意多个任意字符**

**3. \_ 表示一个任意字符**

——————————————————————

查询姓黄的学生

select \* from students where name like "黄%";

select \* from students where name like "%月%";

- % 表示任意多个字符

查询姓黄并且 "名" 是一个字的学生

select \* from students where name like "黄\_";

select \* from students where name like "\_黄\_";

- \_ 表示一个任意字符

—————————————————————————

>>where 之范围查询（第十四课）

- 范围查询分为连续范围和非连续范围查询

- 能够说出 bettween-and 和 in 的区别

——————————————————————

查询编号是 1 或 3 或 8 的学生

select \* from students where id in(1,3,8);

select \* from students where id not in(1,3,8);

**- in 表示在一个非连续的范围内**

查询编号为 3 至 8 的学生

select \* from students where id between 3 and 8;

select \* from students where id not between 3 and 8;

**- between ... and 表示在一个连续的范围内**

查询编号是 3 至 8 的男生

select \* from students where (id between 3 and 8) and gender=1;

—————————————————————————

>>where 之空值判断（第十五课）

- 知道用什么判断空和非空

——————————————————————

查询没有填写身高的学生

select \* from students where height **is null;**

查询填了身高的学生

select \* from students where height **is not null;**

—————————————————————————

>>order 排序（第十六课）

- 能够说出排序使用的关键字

- 能够说出升序与降序分别使用的关键字

——————————————————————

排序查询语法

select \* from 表名 order by 列1 asc|desc [,列二 asc|desc,]

select \* from students where gender=2 and (

age between 18 and 34) order by height desc,id desc;

- 年龄在 18 - 34 女性，身高从高到矮，相同情况下按年龄排

**- 将行数据按照 列1 进行排序，如果某些行 列1 的值相同时，则按照 列2 排序，以此类推**

select \* from students where gender=1 and is\_delete=0 order by id desc;

**- asc 即为升序(如果不写默认升序)**

**- desc 即为降序(如果不写默认升序) \***

—————————————————————————

>>聚合函数的作用（第十七课）

- 知道如何求出集合中数据的最大值

- 知道如何求出集合中数据的最小值

- 知道如何求出集合中数据的数量

- 知道如何求出集合中数据的和

- 知道如何求出集合中数据的平均值

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

命令 作用 示例

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**count(\*)**  **计算总行数** select count(\*) from car;

select count(\*) "男" from students where gender=1;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**max(列)** **求此列的最大值** select max(max\_speed) from car;

select max(height) from students where gender=2;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**min(列) 求此列的最小值** select min(max\_speed) from car;

select min(height) from students where gender=2;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**sum(列) 求此列的和**  select sum(max\_speed) from car;

select sum(age) from students where gender=2;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**avg(列) 求此列的平均值** select avg(max\_speed) from car;

select avg(age) from students where gender=2;

select sum(age)/count(\*) from students where gender=2;

select round(sum(height)/count(\*),2) from students where gender=1;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**round(值，1) 保留 1 位小数** select round(avg(max\_speed)) from car;

select round(sum(age)/count(\*)) from students where gender=2;

select round(sum(height)/count(\*),2) from students where gender=1;

—————————————————————————

>>group分组（第十八课）

- 能够说出分组的目的

- 能够使用 group by 完成对数组的分组

- 能够说出 group\_concat 函数的作用

- 能够使用聚合函数完成对分组结果的统计

- 能够使用 having 完成对分组结果的条件过滤

、

———————————————————————

**1.按照性别分组，查询所有性别**

- select gender from students group by gender;

- 在 MySQL 默认情况下，select 后的字段只能出现一下两种情况

- 1.在 group by 后出现过

- 2.在聚合函数中出现

2.计算每种性别中人数

- select gender,count(\*) from students group by gender;

3.查询男，女 性别中的年龄最大值

- select gender,max(age) from students where gender in("男","女") group by gender;

———————————————————————

**2.group by + group\_concat(字段名)**

- 根据分组结果，使用 group\_concat() 来放置每一个分组中莫字段的集合

- select gender,max(age),group\_concat(name) from students where gender in ("男","女") group by gender;

**3.where 与 having**

- where 是一开始就筛选,having 是在组中筛选(分组完后再次进行过滤)

- 查询平均年龄超过30岁的性别,以及姓名

- select gender,avg(age) from students group by gender having avg(age) > 30;

- 查询每种性别的平均年龄和名字

- select gender,avg(age),group\_concat(name) from students group by gender;

- 查询每组性别中人数多余两个的性别或姓名

- select gender,count(\*),group\_concat(name) from students group by gender having count(\*) > 2;

**4.with rollup 汇总的作用**

- select gender,count(\*) from students group by gender with rollup;

+-----------+-----------+

| gender | count(\*)|

+-----------+-----------+

| 男 | 5 |

| 女 | 7 |

| 中性 | 1 |

| 保密 | 1 |

| NULL | 14 |

+-----------+-----------+

—————————————————————————

>>limit限制记录（第十九课）

- 知道 limit 关键字的作用

- 知道 limit 后面两个参数的作用

limit 的使用

- 可以使用 limit 限制取出记录的数量,但 **limit 要写在 sql 语句的最后**

- 语法:

**- limit 起始记录,记录数**

**- 起始记录是指从第几天记录开始取，第一条下标为 0**

- 记录数是从起始记录开始向后依次取的记录数

**- 从下标为 0 的记录开始取,取三条:**

**- select \* from students limit 0,3;**

标准格式

select > from > where > group by > order by > having > limit

—————————————————————————

>>分页查询（第二十课）

- 能够使用公式计算 limit 查询的起始位置

1.分页查询语法

- 从 start 开始,获取 count 条数据

- start 默认值为 0

- 也就是当前用户需要获取数据的前 n 条的时候可以直接写上 xxx limit n

- select \* from 表明 limit start=0 , count

—————————————————————————

>>连接查询（第二十一课）

连接查询

- 能够说出连接的作用

**- 当有多张表的时候，能够关联成一个大数据集去查询**

- 能够说出什么是笛卡儿积

**- 说白了就是遍历一遍表内的数据**

- 能够说出什么是内连接

**- 指定条件下 （where id 相等） 相互咨询**

- 能够说出什么是外连接

**- 左/右表为准,去查询彼此，没有则补 null**

- 能够使用 inner join 关键字来完成对多张表的内连接查询

- 能够使用 left join 关键字来完成对多张表的左连接查询,没查到会用null填补

- 能够使用 right join 关键字来完成对多张表的右连接查询,没查到会用null填补

1.连接两张表

- inner join ... on

- select \* from students inner join classes;

2.正确对接( 新增了where )

select \* from students inner join classes where students.cls\_id = classes.id;

3.去除无用( 只要姓名与班级 ) - 常规写法

select students.name,classes.name from students inner join classes where students.cls\_id = classes.id;

4.去除无用( 只要姓名与班级 ) - 真正写法

select s.name,c.name from students s,classes c where s.cls\_id = c.id;

5.查询学生信息 ( 左连接 ) - 没查到会用null填补

select \* from students left join classes on students.cls\_id = classes.id where classes.name = null;

—————————————————————————

>>自连接查询（第二十二课）

- 能够说出自连接的作用

- 能够说出自连接和普通连接上的区别

-- 自连接查询

-- 表和自己建立连接

-- 1、查询一共有多少个省

select count(\*) from areas where pid is null;

-- 2、查询省的名称为“山西省”的所有城市

select \* from areas city inner join areas province on city.pid = province.aid where province.atitle='山西省';

-- 3、查询省的名称为“广州市”的所有县

select \* from areas city inner join areas province on city.pid = province.aid where province.atitle='广州市';

—————————————————————————

>>子查询（第二十三课）

- 能够说出什么是 标量 子查询

**- 只返回一个值，一行一列**

- 能够说出什么是 行子 查询

- 能够说出什么是 列子 查询

- 子查询是嵌入到主查询中

**- 列子查询：返回结果是一列(一列多行)**

**- 行字查询：返回结果是一行(一行多列)**

2 查出高于平均身高的信息

- select \* from students where height > (select avg(height) from students);

3 查询学生的学号和班级编号对应的学生信息

- select name from students where cls\_id in (select id from classes);

- 1 查出所有的班级id

- select id from classes;

- 2 查出能够对应上班级号的学生信息

- select \* from students where id in (select id from classes);

—————————————————————————

>>复习SQL（第二十四课）

- 注意作为一个库，应当**灵活多变**

—————————————————————————

>>外键使用（第二十五课）

- 知道数据库表 外键约束 的作用

- 能够为已经出存在的表添加外键约束

- 能够创建表时添加外键约束

**1.在以存在的表中添加外键约束**

**- 外键约束：对外键的值在更新和插入时进行插入和引用的表中字段数据进行对比**

- 关键字：foreign key 只有 innodb 数据库引擎 支持外键约束

- alter table 表明 add foreign key (当前表的字段) references 表名(字段);

- 给 brand\_id 添加外键约束和 goods\_brands 的 id 建立外键关联

- alter table goods add foreign key (cate\_id) references goods\_cates(id);

- 注意：建立连接的类型要一致

**2.创建表时就有外键约束，约束对象必须存在**

+------------------------------------+

| Tables\_in\_jing\_dong |

+------------------------------------+

| goods |

| goods\_brands |

| goods\_cates |

| goods\_test |

+------------------------------------+

create table goods\_test (

id int primary key auto\_increment,

name varchar(150) not null,

cate\_d int unsigned not null,

brand\_id int unsigned not null,

foreign key (cate\_id) references goods\_cates(id),

foreign key (brand\_id) references goods\_brands(id)

);

**3.删除外键约束**

show create table goods;

- 获取名称后就可以根据名称来删除外键约束

alter table goods drop foreign key 外键名称

alter table goods\_test drop foreign key goods\_test\_ibfk\_1;

**- 启用外键约束会极大降低表更新的效率**

—————————————————————————

>>视图（第二十六课）

- 能够说出视图的作用

视图:一个虚拟的，仅仅支持查询，**把复杂的SQL语句的功能封装起来的一个表**

—————————————————————————

>>视图的使用（第二十七课）

- 创建视图的 SQL 语句

- 使用视图的 SQL 语句

- 查看视图的 SQL 语句

- 删除视图的 SQL 语句

- 会根据库的变化而变化

**1.定义视图**

- 建议**以 v\_** 开头

create view v\_名称 as select 语句

- 顶部类别名称不能一致

**2.查看视图**

- select \* from 视图名

**3.删除视图**

- drop view 视图名

**4.注意**

- 视图只能进行搜索

—————————————————————————

>>事务概念即特性（第二十八课）

- 能够说出事务的使用场景

- 能够说出事务的四个特性以及分别含义

**1.事务的四大特性( ACID )**

**- 原子性**

**- 一致性**

**- 隔离性**

**- 持久性**

—————————————————————————

>>事务使用（第二十九课）

- 能够说出 commit 对事务的作用

- 能够说出 rollback 对事务的作用

1.查看数据库服务器支持的表数据引擎

**show engines;**

**- 在 ENGINE=InnoDB 情况下**

视图：

begin

|—————————— |

| |

| 内容无误 |

| |

———————————

commit

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

begin <-------------------------|

|—————————— | |

| | |

| 内容有误 | |

| | |

——————————— |

rollback --------------------------|

—————————————————————————

>>验证事务（第三十课）

- 验证事务的特性

**1.事务的四大特性( ACID )**

**- 原子性 A**

**- 一致性 C**

**- 隔离性 I**

**- 持久性 D**

—————————————————————————

>>数据库三范式（第三十一课）

- 知道一个表结构设计是否满足规范要求

1.第一范式:强调的是列的原子性,即使不能够分成其他几列

name | tel | addr |

张三 | 10086 | 山东 |

2.第二范式:满足 1NF ,另外包含两部分内容,一是表必须有一个主键;

二是非主键字段 必须完全依赖于主键 而不只依赖主键一部分

分表存储？

3.第三范式:满足 2NF ,另外非主键列必须直接依赖于主键,不能存在传递依赖

即不能存在:非主键列 A 依赖于非主键列 B ,非主键列 B　依赖于主键的情况

—————————————————————————

>>E-R模型及表间关系（第三十二课）

- 了解 E-R 模型的组成部分

- 能够举例说出生活中 1对1 1对多 多对多 的关系例子

- E-R模型即E-R图是指提供了表示实体型,属性和联系的方法,用来描述现实世界的

- 概念模型:关系有三种

1. 一对一

2. 一对多

3. 多对多( 需要三张表 )

—————————————————————————

>>索引-建立和删除（第三十三课）

- 能够说出 索引 的作用

- 能够说出创建 索引 的 SQL 语句

- 能够说出查看 索引 的 SQL 语句

- 能够说出删除 索引 的 SQL 语句

**1.查看表中已有索引**

**show index from goods;**

2.创建索引

- 如果指定字段是 字符串 ，需要指定长度，建议长度与定义字段时的长度一致

- 字段类型**如果不是字符串可以不填写长度部分**

create index 索引名称 on 表名( 字段名称( 长度 ) )

create index idx\_1 on goods (name(150));

3.删除索引

drop index 索引名称 on 表名

drop index idx\_1 on goods;

—————————————————————————

>>索引 - 检测执行结果（第三十四课）

- MySQL 的 Query Profiler 是一个使用非常方便的 Query 诊断分析工具

- 通过该工具可以获取一条 Query 在整个执行中的消耗情况

- 通过执行 set profiling 命令,可以开启关闭 Query Profiler 功能

- 开启运行时间检测

**set profiling = 1**

**- 查看结果**

**show profiles;**

—————————————————————————

>>用户管理（第三十五课）

- 能够使用 grant 创建一个用户并且授权

- MySQL 账户体系可以分为一下几种

- 1.服务实例级账号 : 启动了一个 mysqld 即为一个数据库实例

- 如果某用户如 root ,拥有服务实例级分配的权限,那么该账号就可以删除所有的数据库

- 连同这些库中的表

- 2.数据库级别账号 : 对特定数据库执行增删改查的所有操作

- 3.数据表级别账号 : 对特定表执行增删改查的所有操作

- 4.字段级别权限 : 对某些表特定字段执行操作

- 5.存储程序级别的账号 : 对存储程序执行增删改查的操作

- 跳转 mysql 库

use mysql;

- 查看 user 的表结构

desc user;

- 查看所有信息

select host,user,authentication\_string from user;

—————————————————————————

>>创建用户 - 授权（第三十六课）

- 需要使用实例级账户登陆后操作,以 root 为例

- 常用权限主要包括 :

**create / alter / drop / insert / update / delete / select**

- 如果分配所有权限,可以使用 **all privileges**

步骤一 : 创建用户

create user '用户名'@'主机' identified by '密码';

create user 'Monika'@'%' identified by '123456';

步骤二 : 授权

grant 权限 on 数据库.表明 to '用户名'@'主机名'；

grant select on \*.\* to 'Monika'@'%';

步骤三 : 刷新权限

flush privileges;

步骤四 : 查询某人权限

show grants for 'laowang'@'localhost';

show grants for 'Monika'@'%';

—————————————————————————

>>账户管理 - 修改权限（第三十七课）

- 能够修改一个用户的密码

- 能够修改一个用户的权限

- 修改权限

grant 权限名称 on 数据库 to 账户@主机 with grant option;

grant all privileges on jing\_dong to 'Monika'@'%' with grant option;

- 修改密码

alter user '名字'@'localhost/%'IDENTIFIED BY '新密码';

注意修改完成后需要刷新权限

**关于主权限密码忘记:**

**- 详细请看 bilibili :BVItJ4IIv7nz 13.4.2 8:42 处**

—————————————————————————

>>用户管理 - 远程登陆（第三十八课）

第一步：查看当前 root 的 host 值

use mysql

select host,user,authentication\_string,plugin from user;

第二步：授权(所有人只要知道密码与IP就可以登陆)

update user set host = '%' where user = 'root';

- % 可以变为IP地址

第三步：刷新权限

flush privileges;

—————————————————————————

>>用户管理 - 删除用户（第三十九课）

语法1：使用 root 登陆

drop user '用户名'@'主机'

drop user 'laoli'@'%'

语法2：使用 root 登陆,删除 mysql 数据库的 user 表中数据

delete from user where user ="用户名";

delete from user where user = "laoli";

**如果 语法1 删除失败则使用 语法2**