[May30 3](#_Toc517893054)

[1 MySQL概述 3](#_Toc517893055)

[1.1什么是数据库？ 3](#_Toc517893056)

[1.2提供数据库服务的软件 3](#_Toc517893057)

[1.3 MySQL的特点 3](#_Toc517893058)

[1.4 数据库软件、数据库、数据仓库 3](#_Toc517893059)

[2 MySQL安装 4](#_Toc517893060)

[2.1 Ubuntu安装MySQL 4](#_Toc517893061)

[2.2 Ubuntu安装软件 4](#_Toc517893062)

[2.3 启动和连接 4](#_Toc517893063)

[3 基本MySQL命令 4](#_Toc517893064)

[3.1 SQL命令的使用规则 4](#_Toc517893065)

[3.2 库的管理 4](#_Toc517893066)

[3.3 库的命名规则 5](#_Toc517893067)

[4 表的管理 5](#_Toc517893068)

[4.1 创建表 5](#_Toc517893069)

[4.2 查看创建表的语句（字符集、存储引擎） 5](#_Toc517893070)

[4.2 表记录的管理 5](#_Toc517893071)

[5 注意 5](#_Toc517893072)

[6 更改库的默认字符集 5](#_Toc517893073)

[7 客户端把数据存储到数据库服务器上的过程 6](#_Toc517893074)

[8 MySQL的数据类型 6](#_Toc517893075)

[8.1数值类型 6](#_Toc517893076)

[8.2 字符类型 7](#_Toc517893077)

[8.3枚举类型 7](#_Toc517893078)

[8.4日期时间类型 7](#_Toc517893079)

[May31 9](#_Toc517893080)

[1 日期时间的函数 9](#_Toc517893081)

[1.1 日期时间的运算 9](#_Toc517893082)

[2 表字段的操作 9](#_Toc517893083)

[3表记录管理 12](#_Toc517893084)

[4 运算符操作 12](#_Toc517893085)

[4.1数值比较&字符比较 12](#_Toc517893086)

[4.2 逻辑比较 13](#_Toc517893087)

[4.3 范围内比较 13](#_Toc517893088)

[4.4 匹配空、非空 14](#_Toc517893089)

[4.5 模糊比较 15](#_Toc517893090)

[5 SQL查询 15](#_Toc517893091)

[5.1 order by 15](#_Toc517893092)

[5.2 limit 16](#_Toc517893093)

[5.3 聚合函数 17](#_Toc517893094)

[5.4 group by（分组） 17](#_Toc517893095)

[5.5 having 18](#_Toc517893096)

[5.6 distinct（不显示字段的重复值） 18](#_Toc517893097)

[5.6 查询表记录时做数学运算 19](#_Toc517893098)

[6 约束 19](#_Toc517893099)

[儿童节 21](#_Toc517893100)

[1 索引index 21](#_Toc517893101)

[2 索引类型 21](#_Toc517893102)

[2.1 普通索引index 21](#_Toc517893103)

[2.2 唯一索引index 23](#_Toc517893104)

[2.3 主键索引（primary key） 23](#_Toc517893105)

[2.4 外键索引（foreign key） 24](#_Toc517893106)

[2.5 使用规则 25](#_Toc517893107)

[3 数据导入 25](#_Toc517893108)

[4 数据导出 26](#_Toc517893109)

[5 表的复制 27](#_Toc517893110)

[5 嵌套查询（子查询） 27](#_Toc517893111)

[June 4 28](#_Toc517893112)

[1 多表查询 28](#_Toc517893113)

[1.1 普通索引index 28](#_Toc517893114)

[2 连接查询 28](#_Toc517893115)

[2.1 内连接 28](#_Toc517893116)

[2.2 外连接 29](#_Toc517893117)

[**2.2.1 左连接** 29](#_Toc517893118)

[**2.2.2 右连接** 29](#_Toc517893119)

[3 数据备份 29](#_Toc517893120)

[4 数据恢复 30](#_Toc517893121)

[5 事物和事物回滚 30](#_Toc517893122)

[5.1 事物应用 31](#_Toc517893123)

[6.1 常用存储引擎的特点 32](#_Toc517893124)

[**6.1.1 innodb特点** 32](#_Toc517893125)

[**6.1.2 myisam特点** 32](#_Toc517893126)

[**6.1.3 加锁的目的** 32](#_Toc517893127)

[**6.1.4 如何决定表使用哪种引擎** 32](#_Toc517893128)

May30

1 MySQL概述

1.1什么是数据库？

存储数据的仓库

1.2提供数据库服务的软件

1）软件分类

MySQL、SQL\_Server、Oracle、DB2、MonggoDB、Mariadb……

2）在生产环境中如何选择使用哪个数据库软件

①是否开源

|  |  |  |  |
| --- | --- | --- | --- |
| 开源软件 | MySQL | MongoDB | Mariadb |
| 商业软件 | Oracle | DB2 | SQL\_Server |

开源 != 免费

②是否跨平台

不跨平台 SQL\_Server

跨平台 MySQL MongoDB Mariadb Oracle DB2

1.3 MySQL的特点

跨平台、支持多种语言

1是一个关系型数据库

数据是以行和列的形式存储的（表格）

表中的每一行叫一条记录

表中的每一列叫一个字段

表和表之间的逻辑关联叫“关系”

① 示例：关系型数据库存储

表1 学生信息表

|  |  |  |
| --- | --- | --- |
| 姓名 | 年龄 | 班级 |
| 星矢 | 25 | 三班 |
| 水滨月 | 26 | 六班 |

表2 班级信息表

|  |  |
| --- | --- |
| 班级 | 班主任 |
| 三班 | 大空翼 |
| 六班 | 松人 |

②非关系型数据库存储储

{‘姓名’:’星矢’，’年龄’:’25’,’班级’:’三班’}

{‘姓名’:’水滨月’，’年龄’:’26’,’班级’:’六班’,’班主任’:’松人’}

1.4 数据库软件、数据库、数据仓库

1. 数据库软件

是一种软件，可以看得见，可操作，用来实现数据库逻辑功能

2. 数据库

是一种逻辑概念，用来存储数据的仓库，侧重存储

3. 数据仓库

从数据量上来说，数据仓库要比数据库庞大，主要用于数据挖掘和数据分析

2 MySQL安装

2.1 Ubuntu安装MySQL

sudo apt-get install mysql-server(安装服务端)

sudo apt-get install mtsql-client(安装客户端)

2.2 Ubuntu安装软件

1.sudo apt-get updata

2.sudo apt-get –f install修复依赖关系

2.3 启动和连接

1.服务端启动

sudo /etc/init.d/mysql start

sudo /etc/init.d/mysql status | stop | restart查看mysql状态

2.客户端连接

1.命令格式

mysql –h主机地址 –u用户名 –p密码（本地输入可以省略-h操作）

示例：

mysql –hlocalhost –uroot –p123456

3 基本MySQL命令

3.1 SQL命令的使用规则

1.每条命令必须以；结尾

2.SQL命令不区分大小写字母

3.使用\c终止当前命令执行

3.2 库的管理

|  |  |
| --- | --- |
| 查看当前已有的库 | show databases; |
| 创建库 （指定字符集） | create database 库名[character set utf8]; |
| 查看创建库的语句 | show create database 库名; |
| 查看当前所在库 | select database( ); |
| 切换库 | use 库名; |
| 查看库中已有表 | show tables; |
| 删除库 | drop database 库名; |

练习：

1.创建库testdb，指定字符集为utf8

2.进入到库testdb

3.查看当前所在库

4.创建库testdb2，指定字符集为latin1

5.进入到库testdb2

6.查看testdb2的默认字符集

7.删除库testdb

8.删除库testdb2

3.3 库的命名规则

1.可以使用数字、字母、\_,但是不能为纯数字

2.库名区分大小写

3.库名具有唯一性

4.不能使用特殊字符和mysql关键字

4 表的管理

4.1 创建表

create table 表名(

字段名1 数据类型, char 字符串

字段名2 数据类型,

字段名3 数据类型,

……);

4.2 查看创建表的语句（字符集、存储引擎）

|  |  |
| --- | --- |
| 查看创建表的语句 | show create table 表名； |
| 查看表结构 | desc 表名； |
| 删除表 | drop table 表名; |

4.2 表记录的管理

4.2.1.在表中插入记录

1.insert into 表名 values（值1），（值2），……；

2.insert into 表名（字段名1，字段名2，……）values（值1），（值2），……；

4.2.2 查询表记录

1.select \* from 表名[where 条件];

2.select字段名1，字段名2，……from 表名 [where 条件]；

5 注意

1.所有的数据都是以文件的形式存放在数据库的目录下

2.数据库目录：/var/lib/mysql

6 更改库的默认字符集

方法：

通过更改MySQL的配置文件实现

步骤：

1.获取root权限 sudo –i

2.备份mySQL的配置文件

cd /etc/mysql/mysql.conf.d/

cp mysqld.cnf mysql.cnf.bak

3.修改配置文件

vi mysqld.cnf

或subl mysqld.cnf

在mysqld下面添加：character\_set\_server = utf8

4.重启MySQL服务/重新加载配置文件(reload)

/etc/init.d/mysql restart | force-reload

7 客户端把数据存储到数据库服务器上的过程

1.连接到数据库服务器 mysql –h -u -p

2.选择库 use 库名

3.创建/修改表

4.断开与数据库服务器的连接exit/quit/ \q

8 MySQL的数据类型

8.1数值类型

**整型**

----------------------------int大整型-----------------------------------

（4个字节）取值范围：0~2\*\*32-1（42亿多）

-----------------------------tinyint微小整型---------------------------

（1个字节）有符号（signed默认）：-128~127

无符号unsigned：0~255

-----------------------------smallint小整型-----------------------------

（2个字节）

-----------------------------bigint极大整型-----------------------------

（8个字节）0~2\*\*64-1

**浮点型**

-----------------------------float-------------------------------------------

（4个字节，最多显示7个有效位）

用法：

字段名 float(m,n) m:总位数 n:小数位位数

float(5,2)取值范围？-999.99~999.99

注意：

1.浮点型插入整数时会自动补全小数位

2.小数位如果多于指定的位数，会对下一位四舍五入

-----------------------------double--------------------------------------

（8个字节，最多显示15个有效位）

-----------------------------decimal-------------------------------------

（最多显示28个有效位）

1.字段名 decimal（m,n）

2.存储空间（整数部分和小数部分分开存储）

规则：将9位数字的倍数包装成4个字节（对于每一个部分需要4个字节来存储9位数的每个倍数，剩余的数字所需的存储空间如下表）

|  |  |
| --- | --- |
| 剩余数字 | 字节 |
| 0 | 0 |
| 1-2 | 1 |
| 3-4 | 2 |
| 5-6 | 3 |
| 7-9 | 4 |

示例：decimal（19,9）

整数部分：10/9 = 商1余1 4字节 + 1 字节 = 5字节

小数部分：9/9 = 商1余0 4字节 + 0字节 = 4字节

8.2 字符类型

-----------------------------char（定长）-------------------------------

宽度取值范围：1~255 name char(255)

不给定宽默认为1

-----------------------------varchar（变长）----------------------------

宽度取值范围：1~65535

char和varchar的特点

char：浪费存储空间，性能好

varchar：节省存储空间，但是性能差

字符类型的宽度和数值类型的宽度的区别：

数值类型的宽度为显示宽度，仅仅用于select查询时显示，和占用的存储空间大小无关，可用zerofill（加在int后）查看效果。

字符类型的宽度超过则无法存储

8.3枚举类型

字段值只能在列举的范围内选择

-----------------------------单选----------------------------

字段名 enum（值1，值2，……，值n）最多65535个不同值

-----------------------------多选----------------------------

字段名 set（‘值1’，’值2’，……，’值n’）最多64个不同值

插入记录时’gril,python,mysql’

8.4日期时间类型

year : 年 YYYY

data : 日期 YYYYMMD

time : 时间 HHMMSS

datetime 年月日时分秒 YYYYMMDDHHMMSS

timestamp

插入记录时datetime不给值默认返回NULL，而timestamp字段默认返回系统时间。

作业：

1.MySQL中的数据类型有：整数类型、时间/日期类型、字符串类型

2.关系型数据库的核心内容是 关系 即 二维表

3.简述客户端把数据存储到数据库服务器上的过程

4.char和varchar的区别，各自的特点

5.创建一个学校的库school

6.在库中创建一个表students存储学生信息，字段如下：

学号（显示宽度为3，不够用0填充）、姓名、年龄（不能为负数）、手机号、成绩（浮点型）、性别（单选）、爱好（多选）、入学时间（年月日）

7.查看students的表结构

8.在表中随意插入一条记录

9.在表中的姓名、手机号、成绩、三个字段插入3条记录

10.查看所有学生的姓名、手机号和成绩

May31

1 日期时间的函数

|  |  |
| --- | --- |
| now( ) | 返回服务器当前时间YYYY-MM-DD HH:MM:SS |
| curdate( ) | 返回当前日期YYYY-MM-DD |
| curtime() | 返回当前时间HH:MM:SS |
| year(时间) | 返回指定时间的年份 |
| date(时间) | 返回指定时间的日期 |
| time（时间） | 返回指定时间的时间 |

insert into t11

1.1 日期时间的运算

语法：

select……from 表名 where 条件（字段名 运算符 [时间 interval 时间间隔单位]）;

interval：是间隔类型关键字

时间间隔单位: 1 day 2 hour | minute | year | month

示例：

查询1天以内的记录

select \* from t11 where

meeting > (now( ) – interval 1 day)

现在时间 – 1天时间 = 一天以前的时间点

查询1天以前3天以内的记录

select \* from t11 where

meeting<(now()-interval 1 day) and

meeting>(now()-interval 3 day);

2 表字段的操作

语法：

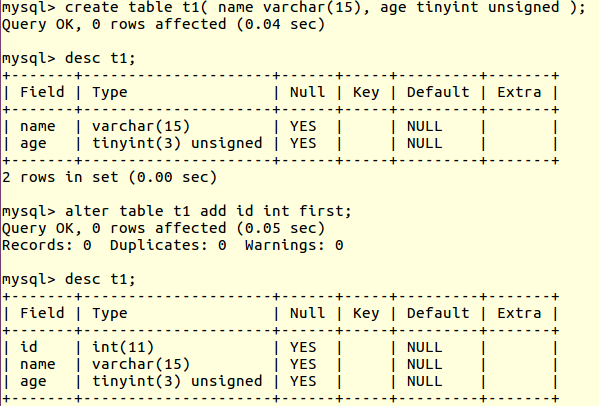
alter table 表名 执行动作；

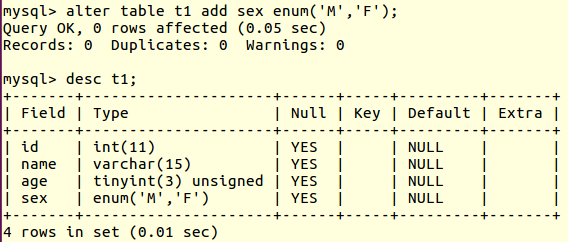
添加字段(add)

alter table 表名 add 字段名 数据类型；

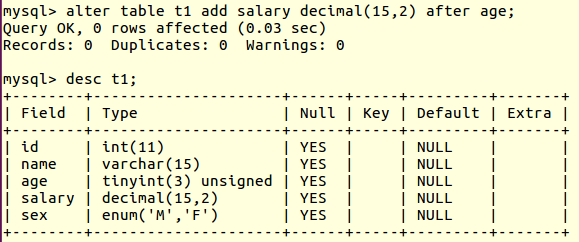
alter table 表名 add 字段名 数据类型 first；

alter table 表名 add 字段名 数据类型 after 字段名；



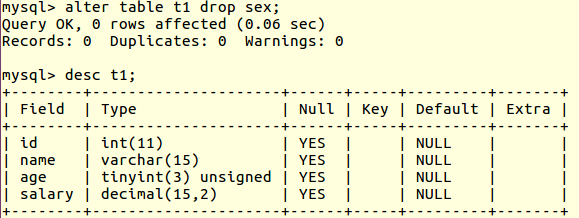


--------------------------------在age之后添加------------------------------------



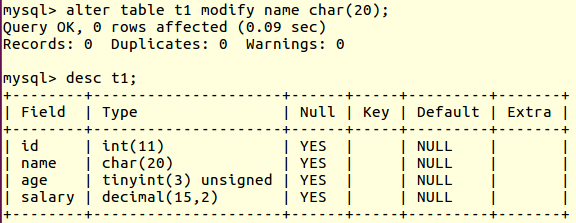
删除字段（drop）

alter table 表名 drop 字段名；



修改字段数据类型（modify）

alter table 表名 modify 字段名 新的数据类型；



练习：

1创建库studb2

create database studb2;

2在库中创建表stuinfo：姓名、年龄、手机号

create table stuinfo(

-> name varchar(20),

-> age tinyint unsigned,

-> phone char(11)

-> );

3查看表结构

desc stuinfo;

4在表中第一列添加一个字段：学号

alter table stuinfo add id int;

5把手机号的数据类型改为bigint

alter table stuinfo modify phone bigint;

6在表中最后一列添加一个字段：注册时间 register，数据类型timestamp

alter table stuinfo add register timestamp;

7在表中 学号、姓名、年龄、手机号四个字段插入2条记录

insert into stuinfo(

-> name,age,phone,id) values('张春林',25,15812345678,1);

8查询5分钟以内的记录明细

select \* from stuinfo where register > (now()-interval 20 minute);

3表记录管理

1.删除表记录

delete from 表名 where 条件；

注意：

delete语句后如果不加where条件，会将表中所有记录全部删除

2.更改表记录

update 表名 set 字段1 = 值1，字段2 = 值2 ……where 条件；

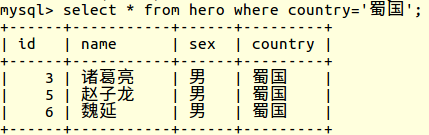
示例：

将字段id等于3 的改为33

update t1 set id =33 where id = 3

练习：

1查找所有蜀国英雄信息



2查找一下英雄的姓名和性别

select \* from hero where sex='女';

select name,sex from hero where sex = ‘女’;

3把id为2的记录姓名改为 司马懿，性别改为男，国家改为魏国

update hero set name='司马懿', sex='男' ,country='魏国' where id=2;

4删除所有的吴国人

delete from hero where country=’吴国’;

5把魏延的

update hero set id=66 where name=’魏延’;

6删除所有英雄

delete from hero;

4 运算符操作

4.1数值比较&字符比较

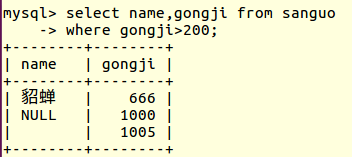
数值比较运算符

= != > >= < <=

字符比较

= !=

示例：



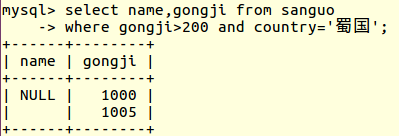
4.2 逻辑比较

and(连接两个或多个条件同时满足)

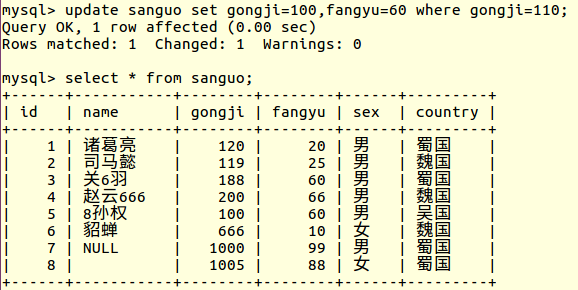
or

示例：

找出攻击力大于200的蜀国英雄的名字和攻击力



将吴国英雄中攻击力为110的英雄的攻击力值设置为100，防御值设置为60



查找蜀国和魏国的英雄

select \* from sanguo where country=’蜀国’ or country=’魏国 5’

4.3 范围内比较

运算符

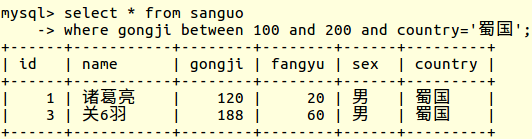
where 字段名 between 值1 and 值2

where 字段名 in（值1，值2，……）

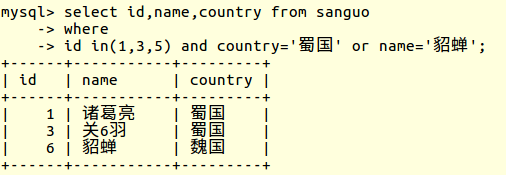
where 字段名not in（值1，值2，……）

示例：

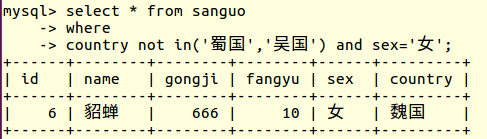
查找攻击力在100~200之间的蜀国英雄信息



查找英雄的编号为1,3或5的蜀国英雄和貂蝉的编号姓名和国家



查找蜀国和吴国之外的国家的女英雄信息

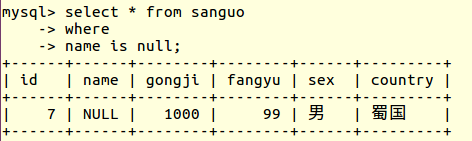


4.4 匹配空、非空

空：is null

非空；is not null

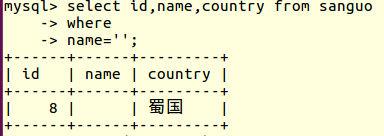
示例



查找姓名为null的蜀国女英雄信息

select \* from sanguo where name is null and country='蜀国' and sex = '女';

查找姓名为‘’的英雄的id姓名和国家



注意：

null是一个空值，必须用is或is not去匹配

‘’是一个空字符串，只能用 = 或者！=去匹配

4.5 模糊比较

语法：

where 字段名 like 表达式

表达式：

\_ ：匹配到单个字符

% : 匹配0到多个字符

select name from sanguo where name like’\_%\_’;匹配name里至少有两个字符的

select name from sanguo where name like’%’;匹配 0到多个字符

select name from sanguo where name like’\_ \_ \_’;匹配 3个字符

select name from sanguo where name like’赵%’;匹配 有‘赵’字符

5 SQL查询

总结（执行顺序）

第三步：select……聚合函数 from 表名

第一步：where

第二步：group by

第四步：having

第五步：order by

第六步：limit

5.1 order by

给查询结果排序

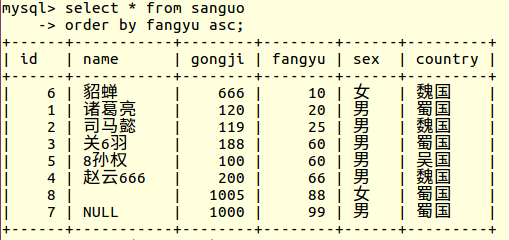
order by 字段名 排序方式

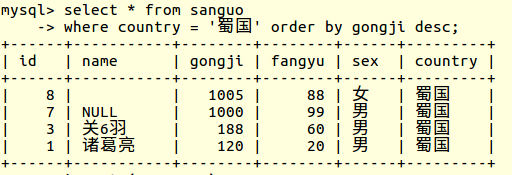
排序方式

asc（默认）：升序

desc：降序

示例：





将魏国蜀国男英雄中名字为三个字的英雄按防御值升序

select \* from sanguo where country in('蜀国','魏国') and sex = '男' and name like'\_\_\_' order by fangyu asc;

5.2 limit

作用：

限制显示查询记录的个数

用法：

limit n-->显示n条记录

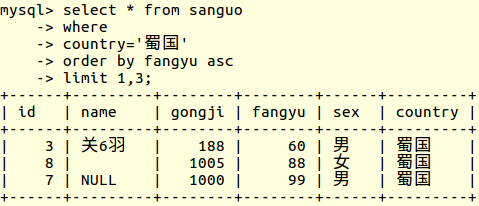
limit m,n-->从m+1条记录开始，显示n条记录

limit 2,3-->显示3,4,5条记录

m的值是从0开始计数，2则表示3条记录

示例：

查找防御值倒数第二名到倒数第四名的蜀国英雄记录



在蜀国英雄中且名字不为空，查找攻击值前三名的英雄的姓名、攻击值和国家

select name,gongji,country from sanguo where country='蜀国' and name is not null order by gongji desc limit 0,3;

5.3 聚合函数

分类：

avg(字段名) 求该字段的平均值

sum(字段名)

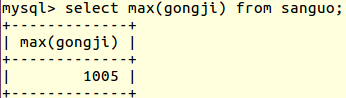
max(字段名)

min(字段名)

count(字段名) 统计该字段se记录的个数空值null不会被统计，空字符串‘’会被统计

示例：

攻击力最强值



统计表中id和name字段分别有多少条记录

select count(id),count(name) from sanguo;

蜀国英雄中攻击值大于200 的英雄个数

select count(\*)from sanguo where country='蜀国' and gongji>200;

5.4 group by（分组）

作用：

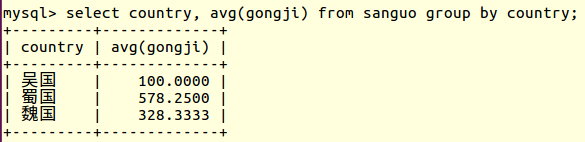
结果进行分组

用法：

group by 字段名

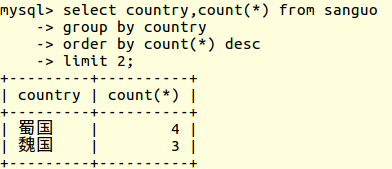
示例：

计算每个国家的平均攻击力



1分组 2 聚合 3去重

查找所有国家中英雄数量最为的前两名国家的国家名称和英雄数量



注意：

group by后的字段名必须要为select之后的字段名，如果查询字段和group by之后的字段不一致，则必须要对该字段进行聚合处理（聚合函数）

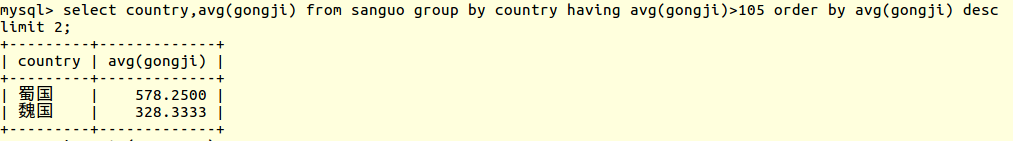
5.5 having

作用：

对查询的结果进行进一步的筛选

示例：

找出平均攻击力大于105的前两名国家



注意：

1.having语句通常与group by联合使用，用来过滤由group by语句返回的结果集

2.where只能操作表中实际存在的字段（desc 表名;）,having操作的是由聚合函数生成的显示列

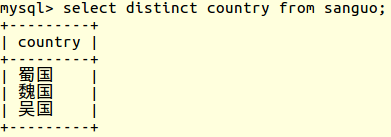
5.6 distinct（不显示字段的重复值）

用法：

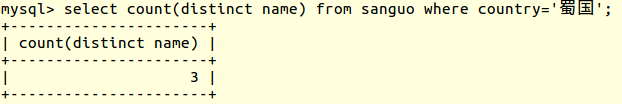
select distinct 字段名1，字段名2 ……from 表名；

示例：

统计sanguo表中统计多少个国家



计算蜀国的英雄中有多少名字



注意：

distinct处理的是distinct和from之间的所有字段，所有字段的值必须全部相同才能去重

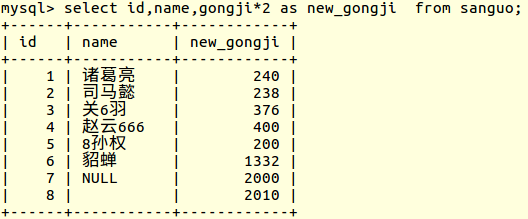
5.6 查询表记录时做数学运算

运算符

+ - \* / %

示例：

查询时显示所有英雄攻击力翻倍



6 约束

作用：

为了限制无效的数据插入到数据表中

约束分类：

默认约束( default)

插入记录时不给该字段赋值，使用默认值

字段名 数据类型 default 值

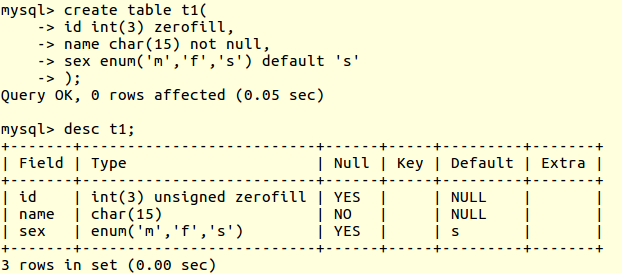
非空约束(not null)

不允许该字段的值有null记录

字段名 数据类型 not null

可连用

字段名 数据类型 not null default 值



作业：

select count(\*),user\_id from comment group by user\_id order by count(\*) desc;

儿童节

1 索引index

定义：

对数据库中表的一列或者多列的值进行安排的一种结构

（MySQL中索引用BTREE方式）

|  |  |
| --- | --- |
| 优点 | 缺点 |
| 加快数据的检索速度 | 当对表中的数据进行增加、删除和修改的时候，索引需要动态维护，降低了数据的维护速度 |
|  | 索引需要占用物理空间 |

索引示例：

1开启运行时间检测

set profiling = 1；

或者 show variables like ‘%pro%’;模糊查询

set profiling = 1；

2执行查询语句

select name from t1 where name=’lucy100999’;

3查看执行时间

show profiles；

4在name字段创建索引

create index index\_name on t1(name);

5执行查询语句

select name from t1 where name=’lucy100999’;

6查看执行时间

show profiles；

2 索引类型

2.1 普通索引index

使用规则

一张表中可以有多个index字段

字段的值可以重复，且可以为null

经常把做查询条件的字段设置为index字段

index字段的key标志位mul

创建index

1创建表on

index（字段名），index（字段名）；

2已有表

create index 索引名 on 表名（字段名）；

查看索引

1.desc 表名；

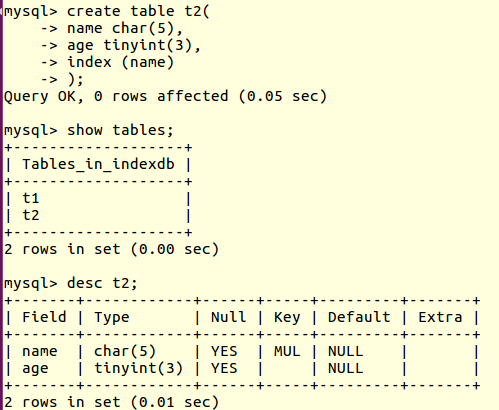
2.show index from 表名；

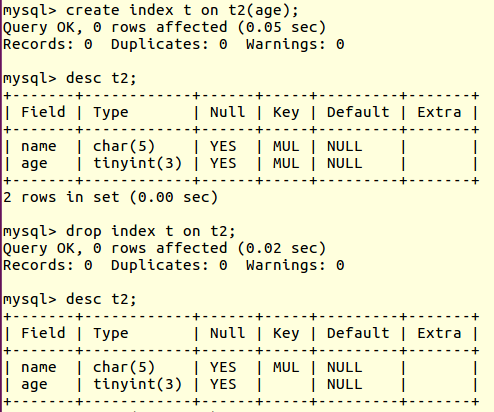
查看索引名

show index from 表名

删除索引

drop index 索引名 on 表名；一次只能删除一个





2.2 唯一索引index

使用规则

1.一个表中可以用多个unique字段

2.对应字段的值不允许重复，但可以为null

3.key标志：UNI

创建

1.创建表

……unique（字段名），unique（字段名））；

……unique（phonenumber），unique（cardnumber））；

2.已有表

create unique index 索引名 on 表名（字段名）；

3.查看/删除（同index）

drop index 索引名 on 表名；

2.3 主键索引（primary key）

自增长属性（auto\_increment）配合主键一起使用

使用规则

1.一个表中只能有一个主键

2.对应字段值不允许重复，且不能为null

3.key标志：PRI

4.把表中能够唯一标识的一条记录的字段设置为主键，通常把记录编号的字段设置为主键

创建

1.创建表

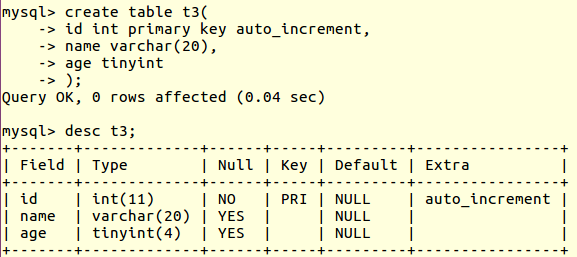
……id int primary key auto\_increment，

……） [auto\_increment = 100]从100开始记录；

方式二

……id int auto\_increment，

……primary key（id））;



2.已有表

alter table 表名 add primary key（字段名）;

alter table 表名 auto\_increment=100;

3.删除

删除auto\_increment属性 alter table 表名 modify 字段名 类型;

alter table 表名 drop primary key；

2.4 外键索引（foreign key）

定义：

让当前表的字段值在另一个表的范围内选择

语法格式：

foreigen key（参考字段名）

references被参考表名（被参考字段名）

on delete级联动作

on update级联动作

级联动作

1.cascade 数据级联更新

当主表删除记录或更改被参考字段值时，从表也会级联更新

2.restrict（默认）

当主表删除记录或更改被参考字段值时，从表中有相关联记录则不允许主表操作

3.set null

当主表删除记录或更改被参考字段值时，从表中相关联记录的字段值

4 no action

同restrict都是立即检查外键限制

使用规则

两张表的两个字段数据类型必须一致

被参考字段必须为key的一种

删除外键限制

alter table 表名 drop foreign key 外键名；

alter table bjtab drop foreign key bjtab\_ibfk\_1;

外键名查看方式

show create table 表名；

在已有表中添加外键限制

alter table 表名 add foreign key（）references 表名（字段名）

on delete 级联动作 on update 级联动作

alter table bjtab add foreign key(stu\_id) references jftab(id) on delete cascade on update cascade;

示例：

表1 ：缴费信息表（财务）

学号 姓名 班级 缴费金额

表2：班级信息表（班主任）

学号 姓名 金额

create table jftab(id int primary key,name varchar(20),class varchar(7),money int) character set utf8;

create table bjtab(stu\_id int,name varchar(20),money int,

foreign key(stu\_id) references jftab(id) on delete cascade on update cascade) character set utf8;

insert into jftab values(1,'唐伯虎','AID1804',200),(2,'点秋香','AID1804',300),(3,'祝枝山','AID1804',500);

2.5 使用规则

1.两张表参考字段和被参考字段数据类型一致

2.被参考字段必须为key的一种，通常primary key

3 数据导入

作用：

将文件系统的内容导入到数据库当中。

语法格式

load data infile‘文件名’

into table 表名

fields terminated by‘分隔符’

lines terminated by ‘\n’

示例：

把 /etc/passwd文件的内容导入到db3下的userinfo表

1在数据库中创建对应的表

create table userinfo(username varchar(20),password char(1),uid int,gid int,comment varchar(50),homedir varchar(50),shell varchar(50))character set utf8;

2把要导入的文件拷贝到数据库搜索路径中

①查看数据库搜索路径

show variables like ‘secure\_file\_priv’;

得到

sudo cp /etc/passwd /var/lib/mysql-files/

3执行数据导入语句

load data infile ‘/var/lib/mysql-files/passwd’ into table userinfo

fields terminated by ‘:’

lines terminated by ‘\n’;

4 数据导出

作用

将数据库中表的记录保存到系统文件里

语法

select … from 表名

into outfile‘文件名’

fields terminated‘分隔符’

lines terminated by ‘\n’；

示例

1.把userinfo表中用户名、密码和UID号三个字段的值导出

select username,passwd,uid from userinfo

into outfile ‘/var/lib/mysql-files/user1.txt’

fields terminated by ‘ ’

lines terminated by ‘\n’;

终端查看

sudo –i

cd /var/lib/mysql-files/

ls

cat user1.txt

2把mysql库下的user表中user、host导出来，user.csv

select user,host from mysql.user

-> into outfile'/var/lib/mysql-files/user.csv'

-> fields terminated by','

-> lines terminated by'\n';

Linux下

sudo cp /var/lib/mysql-files/user.csv ~

注意：

导出的内容由SQL查询语句决定

执行导出命令时路径必须指定对应的数据库目录下

5 表的复制

1.表的重命名

alter table 表名 rename 新表名；

2.复制语法格式

create table 表名 select …… from 表名……；

示例

复制AID1709表的全部记录和字段，aid17092

create table aid17092

-> select \* from AID1709;

复制aid1709表中成绩低于65分的学生到表next

create table next

-> select \* from t2

-> where secor<65;

3.只复制表结构

create table 表名 select\*from 表名 where false；

5 嵌套查询（子查询）

定义：

把内层的查询结果作为外层查询的条件

语法

SQL查询语句where条件（SQL查询语句）

示例

把UID的值小于这个字段平均值的用户名和UID显示出来

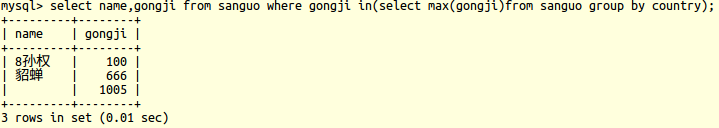
select avg(uid) from userinfo;#平均值

select username,uid from userinfo where uid<

select username,uid from userinfo

where uid<(select avg(uid)) from userinfo);

找出每个国家攻击力最高的英雄的名字和攻击值



作业：

June 4

1 多表查询

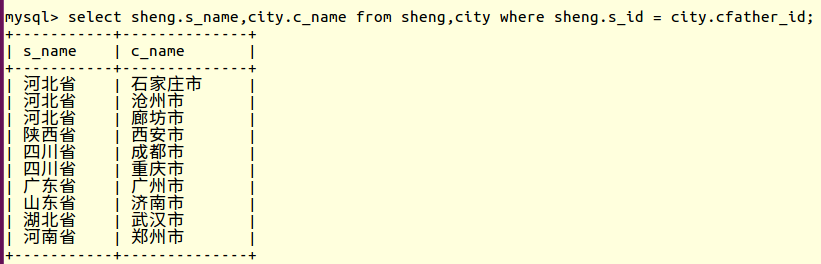
1.1 普通索引index

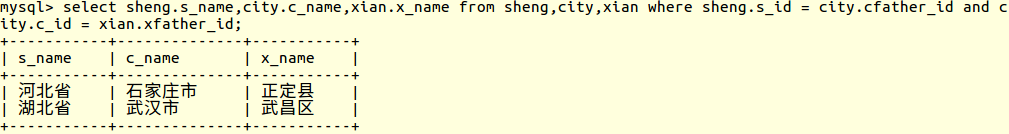
语法格式

select 字段名列表 from 表名列表 where 条件；

1 显示省市详细信息

select 字段名 from 表名 where 条件





2 连接查询

2.1 内连接

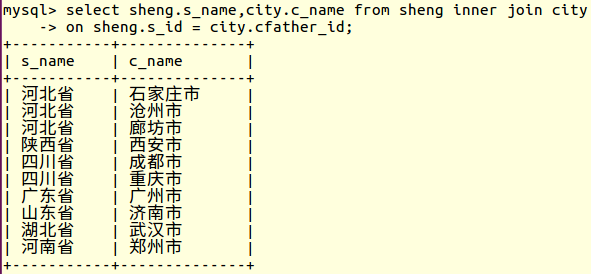
语法格式

select 字段名 from 表1 inner join 表2 on 条件

inner join 表3 on 条件；

示例：

内连接显示省和市的详细信息



2.2 外连接

**2.2.1 左连接**

定义

以左表为主显示查询结果（无论右边有或者无都以左表显示为主）

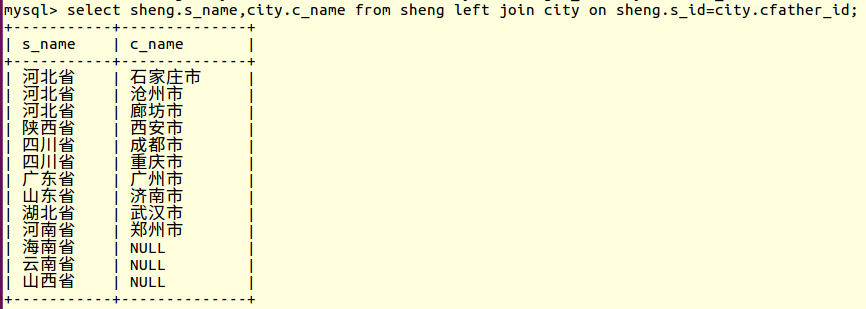
语法格式

select 字段名列表 from 表1 left join 表2 on 条件

left join 表3 on 条件；

示例

显示省、市的信息

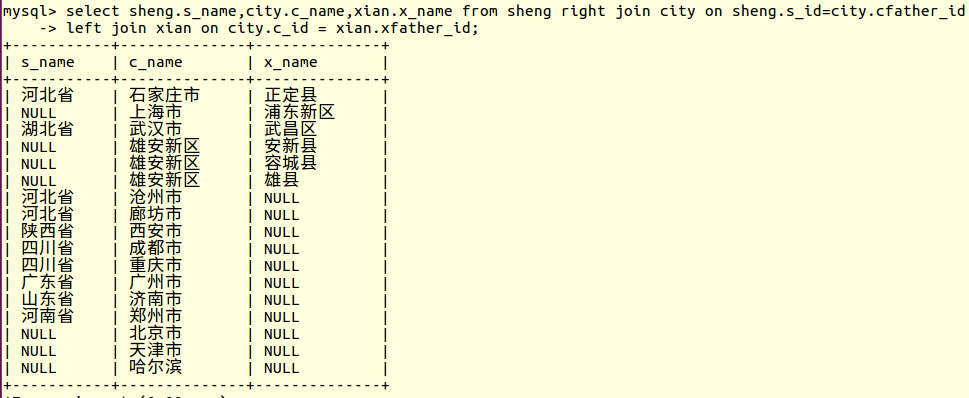


**2.2.2 右连接**

与左连建相反 left改为right

示例

显示省、市、县详细信息，要求所有的市全部显示



3 数据备份

在mysqldump，在Linux终端操作

命令格式

mysqldump –u用户名 –p源库名 >路径/\*\*\*.sql

示例

备份june4库

mysqldump –uroot –p june4 >~/mydata/june4.sql

源库名的表示方式

|  |  |
| --- | --- |
| --all-databases | 备份所有库 |
| 库名 | 备份单个库 |
| -B库1库2…… | 备份多个库 |
| 库名表1表2…… | 备份指定库的指定表 |

练习

1备份所有库all.sql放到mydata目录中

mysqldump -uroot -p --all-databases>all.sql

2备份june4库中的sheng、city和xian三张表为db4scx.sql

mysqldump -uroot -p june4 sheng city xian>db4scx.sql

4 数据恢复

mysql –u用户名 –p目标库名<路径/\*\*\*.sql

恢复june4库

1先创建空库

create database june4 character set utf8;

2恢复

mysql -uroot -p june4<db4scx.sql

从所有库的备份中恢复摸一个库（--one-database）

mysql –u用户名 –p –one-databases目标库名<all.sql

2在june4.sheng新增一条记录

insert into sheng values(null,300000,’测试’);

在june4库中新创建一张表t88

create table t88(id int);

注意

1在恢复库时，如果恢复到原库会将表中数据覆盖，新增的表不会删除

2数据恢复如果要恢复的库不存在，则先要创建空库

5 事务和事务回滚

定义：

一件事才开始发生到结束的整个过程

属性：

1原子性

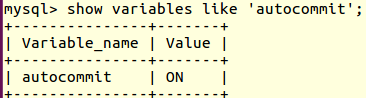
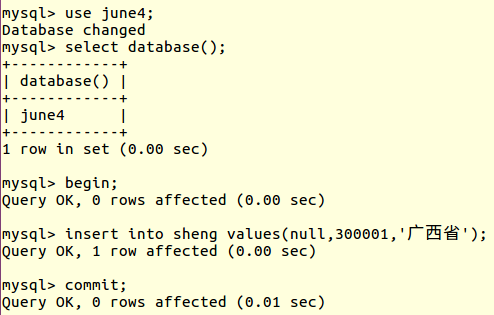
2一致性

3隔离性

4持久性

事务和事务回滚的应用

show variables like ‘autocommit’;

5.1 事物应用

开启事物

mysql>begin;

mysql>…一条或者多条sql命令##此时automatic被禁用

终止事物

mysql>commit;

mysql>rollback;

注意

事物回滚只针对于表记录的操作：增加、删除、修改有效，对库和表的操作无效

过程

表1:CCB

create table CCB(name char(15),money int);

insert into CCB values(‘zhuangqian’,10000);

表2：ICBC

create table ICBC(name char(15),money int);

insert into ICBC values(‘shouqian’,4000);

6 储存引擎

基本操作

|  |  |  |
| --- | --- | --- |
| 查看所有的存储引擎 | show engines; | |
| 查看已有表的存储引擎 | show create table 表名; | |
| 只有InnoDB才支持事务回滚 | | |
| 创建表时指定存储引擎 | | create table 表名（…）engine=myisam; |

常用存储引擎

InnoDB

myisam

[mysqld]

defaut – stotage – engine =myisam

6.1 常用存储引擎的特点

**6.1.1 innodb特点**

共享表空间



|  |  |
| --- | --- |
| 表名.frm | 表结构 |
| 表名.ibd | 表记录&索引信息 |

支持行级锁

**6.1.2 myisam特点**

独享表空间

|  |  |
| --- | --- |
| 表名.frm | 表结构 |
| 表名.myd | 表记录 |
| 表名.myi | 索引文件 |

支持表级锁

**6.1.3 加锁的目的**

解决客户端并发访问的冲突问题

锁类型

|  |  |
| --- | --- |
| 读锁（select）共享锁 | 加锁后不能更改表中内容但可以查询 |
| 写锁（）互斥锁排它锁 | 加锁后不能有任何操作 |

锁粒度

表级锁

行级锁

**6.1.4 如何决定表使用哪种引擎**

|  |  |
| --- | --- |
| 执行查询操作多的表使用 | myisam（使用innodb浪费资源） |
| 执行写操作多的表使用 | innodb存储引擎 |

7 mysql调优

1选择合适的存储引擎

2 SQL语句调优（尽量避免全表扫描）

①在select、where、order by字段上建立索引

②where子句中尽量少用’!=’，否则导致全表扫描

③尽量避免NULL值判断，否则导致全表扫描（可以在空值设置为0）

优化前：select id from t1 where number is null;

优化后：select id from t1 where number=0;

④尽量避免用or来连接条件

优化前：select id from t1 where id=10 or id=20;

优化后：select id from t1 where id=10

union all

select id from t1 where id=20;

⑤模糊查询尽量避免前置’%’,否则全表扫描

select id from id where name like’%c%’;

⑥尽量避免in和not in的使用

优化前：select id from t1 where number in(1,2,3);

对于连续的值，尽量用between and

⑦尽量避免使用\*号，要用具体的字段名代替星号，不要反回用不到的任何字段

8 与python交互

交互类型

python3：pymysql

pymysql $ sudo pip3 install pymysql

connection对象

创建与数据库连接对象（调用connect（）方法）

conn = pymysql.connect(参数列表)

class hello(object):

def \_\_init\_\_(self,……):

……

def fun1(self,……):

……

def fun2(self,……):

……

duixiang1=hello()

duixiang1.fun1(…)

参数列表

|  |  |
| --- | --- |
| 1.host | 主机地址，本机：’localhost’ |
| 2.user | 用户 |
| 3.port | mysql端口，默认3306 |
| 4.database | 数据库名 |
| 5.password | 连接密码 |
| 6.charset | 编码方式，推荐使用utf8 |

示例：

conn=pymysql.connect(host=’localhost’,user=’root’,password=’123456’,database=’db4’,port=3306,charset=’utf8’)

连接对象（如：conn）的方法

|  |  |
| --- | --- |
| close() | 关闭连接 |
| commit() | 提交到数据库执行 |
| rollback() | 事物回滚操作 |
| cursor() | 创建游标对象，用来执行SQL语句来获得结果 |

游标对象（cursor对象）

作用：

执行sql语句

创建游标对象：

调用连接对象的cursor()方法

示例：

cursor1=conn.cursor()

游标对象的方法

|  |  |
| --- | --- |
| execute(operation) | 执行sql语句 |
| close() | 关闭游标对象 |
| fetchone() | 获取结果集的第一条记录返回一个元组 |
| fetchmany(n) | 获取结果集的n条记录，返回一个大元组 |
| fetchall() | 获取结果集的所有记录，返回一个大元组 |

总结

1建立数据库连接conn

2创建游标对象cursor1=conn.cursor()

3利用游标对象的方法操作数据库

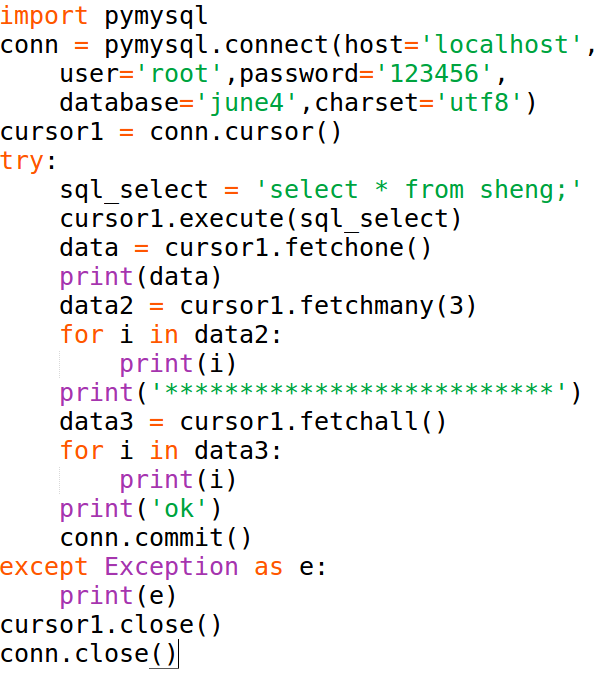
cursor1.execute(‘sql语句’)

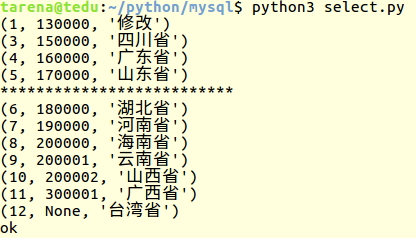
4提交cursor1.close()

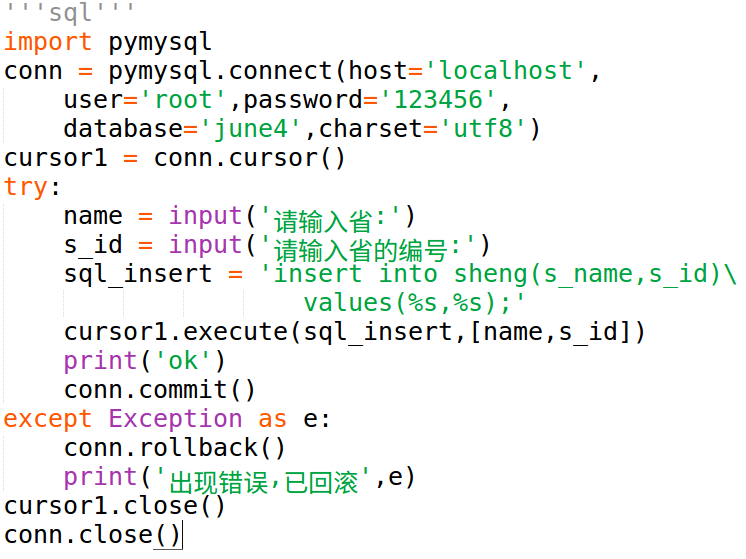
5关闭游标cursor1.close()

6关闭数据库连接conn.close()

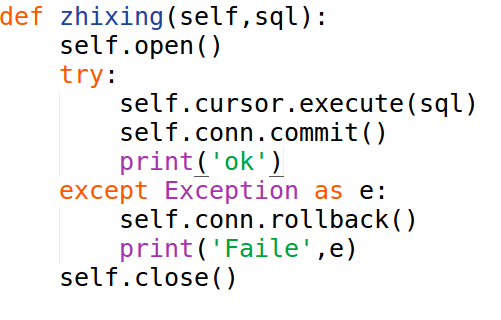
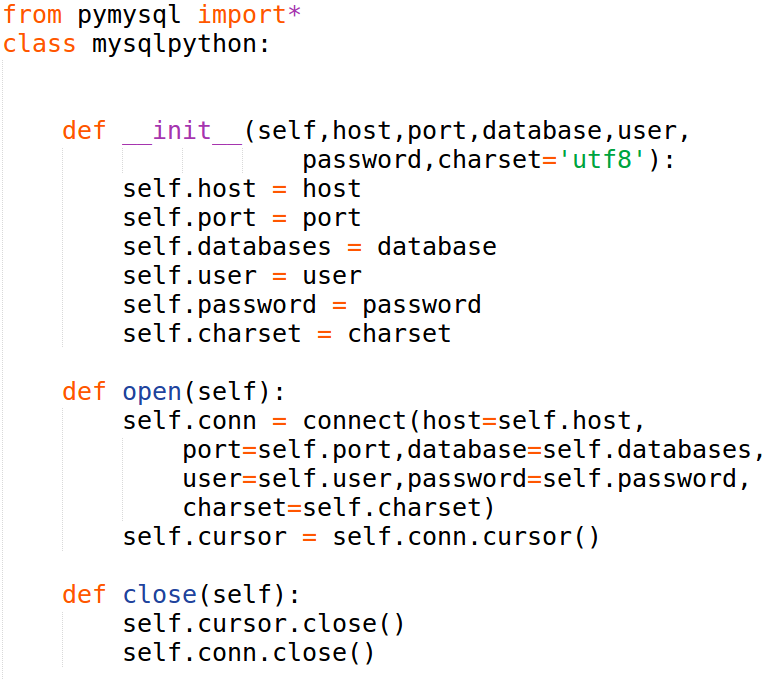




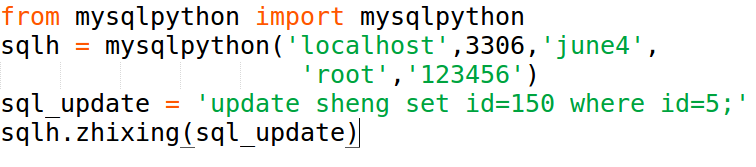




面向对象



调用



9 ER模型

Entry – Relation

1关系

1分类

1.一对一关系（1:1）：班级和班长

2.一对多关系（1：n）：公司和员工

3.多对多关系（m：n）：学生和课程

2 ER图的绘制

1矩形框代表实体，菱形块代表关系，椭圆框代表属性

**什么是事务？**

事务由一个或多个sql语句组成一个整体，如果所有的语句执行成功那么修改将会全部生效，如一条sql语句将销量+1，下一条再+1，倘若第二条失败，那么销量将撤销第一条sql语句的+1操作，只有在该事务中所有的语句都执行成功才会将修改加入到数据库中。

**事务的特性**

事务具体四大特性，也就是经常说的ACID   
1. 原子性（Atomicity）   
　　原子性是指事务包含的所有操作要么全部成功，要么全部失败回滚，因此事务的操作如果成功就必须要完全应用到数据库，如果操作失败则不能对数据库有任何影响。   
2. 一致性（Consistency）   
　　一致性是指事务必须使数据库从一个一致性状态变换到另一个一致性状态，也就是说一个事务执行之前和执行之后都必须处于一致性状态。

　　拿转账来说，假设用户A和用户B两者的钱加起来一共是5000，那么不管A和B之间如何转账，转几次账，事务结束后两个用户的钱相加起来应该还得是5000，这就是事务的一致性。

3.隔离性（Isolation）   
　　隔离性是当多个用户并发访问数据库时，比如操作同一张表时，数据库为每一个用户开启的事务，不能被其他事务的操作所干扰，多个并发事务之间要相互隔离。

　　即要达到这么一种效果：对于任意两个并发的事务T1和T2，在事务T1看来，T2要么在T1开始之前就已经结束，要么在T1结束之后才开始，这样每个事务都感觉不到有其他事务在并发地执行。

4.持久性（Durability）   
　　持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的，即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。Mysql中会保存有相应的操作日志，即使遭遇故障依然能够通过日志恢复最后一次更新。

　　例如我们在使用JDBC操作数据库时，在提交事务方法后，提示用户事务操作完成，当我们程序执行完成直到看到提示后，就可以认定事务以及正确提交，即使这时候数据库出现了问题，也必须要将我们的事务完全执行完成，否则就会造成我们看到提示事务处理完毕，但是数据库因为故障而没有执行事务的重大错误。

**MySql中支持事务的引擎**

在mysql中用的最多的存储引擎有：innodb，bdb，myisam ,memory 等。其中innodb和bdb支持事务而myisam等不支持事务。

InnoDB：

支持事务处理等

不加锁读取

支持外键

支持行锁

不支持FULLTEXT类型的索引

不保存表的具体行数，扫描表来计算有多少行

DELETE 表时，是一行一行的删除

InnoDB 把数据和索引存放在表空间里面

跨平台可直接拷贝使用

InnoDB中必须包含AUTO\_INCREMENT类型字段的索引

表格很难被压缩

MyISAM：

不支持事务，回滚将造成不完全回滚，不具有原子性

不支持外键

支持全文搜索

保存表的具体行数,不带where时，直接返回保存的行数

DELETE 表时，先drop表，然后重建表

MyISAM 表被存放在三个文件 。frm 文件存放表格定义。 数据文件是MYD (MYData) 。 索引文件是MYI (MYIndex)引伸

跨平台很难直接拷贝

MyISAM中可以使AUTO\_INCREMENT类型字段建立联合索引

表格可以被压缩

选择：

因为MyISAM相对简单所以在效率上要优于InnoDB.如果系统读多，写少。对原子性要求低。那么MyISAM最好的选择。且MyISAM恢复速度快。可直接用备份覆盖恢复。

如果系统读少，写多的时候，尤其是并发写入高的时候。InnoDB就是首选了。

两种类型都有自己优缺点，选择那个完全要看自己的实际类弄。