**数据库**

数据库概念

Mysql Navicat

SQL语言 查询（重点）

Mysql高级

**1.概念**

 数据库的作用就是把数据以表的形式存储起来，方便查询

**2.RDBMS**

关系型数据库管理系统

核心是以二维表存储数据

行：一条记录，一个事物的信息

列：字段。一个事物的某个属性

一个表中所有行是一类事物的集合

一个数据库由很多表组成

**3.SQL**

结构化查询语言，用于关系型数据库

不区分大小写

重点学习查询

**4.Mysql**

由瑞典公司开发，被卖给Sun,Sun被卖给Oracle

开源、免费、支持多平台

**5.navicat 操作**

连接数据库 用户名 密码 端口号（3306） ip

1、连接mysql服务端

 2、打开连接

3、连接下面显示mysql服务端中所有的数据库仓库

4、数据库的操作

打开数据库

数据库的创建

数据库的修改

数据库的删除

5、数据表的操作

创建表

修改表

字段 添加、删除、插入、修改

重命名

删除表

6、数据的操作

打开表

数据的增加、修改、删除

**6.数据类型**

整数：int，有符号范围（-2147483648 ~2147483647），无符号范围（0 ~ 4294967295）  长度不起作用

小数：decimal，如decimal(5,2)表示共存5位数，小数占2位，整数占3位

字符串：varchar，范围（0~65533），如varchar(3)表示最多存3个字符，一个中文或一个字 母都占一个字符

查看任何一个数据类型的使用  
  1、打开mysql命令行客户端  
  2、help 数据类型名称，如 help tinyint

约束

主键：int类型、无符号、自动递增

主键(primary key)：代表一条记录的唯一标识，此自动的值必须唯一不能重复，不能为空;  
   自动递增：从1开始，一直累加，值不会取以前的值  
   表清空：删除所有数据，不删表结构，自动递增的值继续累加  
   表截断：删除所有数据，不删表结构，自动递增的值从1开始

非空(not null)：此字段不允许填写空值

惟一(unique)：此字段的值不允许重复，通过sql语法演示

默认值(default)：当不填写此值时会使用默认值，如果填写时以填写为准

外键(foreign key)：维护两个表之间的关联关系，后面多表时讲

数据库的备份和恢复

备份：

鼠标右键点击某个数据库，点击转储SQL文件->结构和数据

恢复：

新创建一个数据库，名字随便起，字符集和排序规则必须和之前备份的数据库一致

双击打开刚才创建的数据库，鼠标右键点击此数据库，点击运行SQL文件

**7.sql语言**

**表操作**

创建表

-- 创建学生表，字段要求如下：注释 ctrl + /    -- 姓名(长度为10)

create table students(    name varchar(10)    )

-- 创建学生表，字段要求如下：取消注释 ctrl + shift + /    --    -- 姓名(长度为10)， 年龄

create table sutdents1(

name varchar(10),

 age tinyint unsigned    )

-- 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 students2(

id int unsigned primary key auto\_increment,

name varchar(10),

age tinyint unsigned,

height decimal(5,2)

)

删除表

-- 删除学生表

-- drop table students

-- drop table if exists sutdents1

删除并创建新表

drop table if exists students;

create table students(

name varchar(10),

age int

)

增删改查

查询

查询姓名和年龄

 -- select name,age from stu

查询所有列的数据

-- select \* from stu

查询性别为男的数据

select \* from stu where sex='男'

插入

-- 插入一个学生，设置所有字段的信息，值的顺序与表中字段的顺序对应

-- insert into students values('亚瑟',20)

-- 插入一个学生，只设置姓名，值的顺序与给出的字段顺序对应

-- insert into students(name) values('鲁班')

-- insert into students(age) values(30)

insert into students(age,name) values(30,'亚瑟2')

当表中有auto\_increment的字段时，添加数据时使用0或者 default 或者 null 来占位

insert into students values(0,'老夫子3',20);

插入多条数据

-- insert into students values(0,'老夫子3',20);

-- insert into students values(0,'老夫子4',20);

-- insert into students values(0,'老夫子5',20);

-- insert into students values (0,'老夫3',20),(0,'老夫4',20),(0,'老夫5',20)

-- insert into students(id,name) values (0,'老夫3'),(0,'老夫4'),(0,'老夫5')

修改数据

-- 更新数据 设置某一个学生的年龄加3岁

update students set age=age+3 where name='亚瑟3'

删除数据

-- 删除数据

delete from students where name='亚瑟3'

逻辑删除

1、-- 添加字段，标识数据是否被删除 is\_delete        --  默认设置为0，代表数据没有被删除

2、-- update students set is\_delete=0

3、-- 删除一条数据，只是修改了这条数据的is\_delete 改为1

-- update students set is\_delete=1 where name='老夫子6'

4、-- 查询所有学生时，不显示删除的学生

select \* from students where is\_delete=0

**8.sql查询**

给字段起别名

-- select name as 姓名,age as 年龄,hometown as 家乡 from students where name='王昭君'

-- select name 姓名,age 年龄,hometown 家乡 from students where name='王昭君'

给表起别名  -- select s.name,s.age from students as s

消除重复数据  -- select distinct age,class from students

select distinct \* from students

比较运算

-- 例1：查询小乔的年龄    -- select age from students where name='小乔'

-- 例2：查询20岁以下的学生

-- select \* from students where age<20

-- 例3：查询家乡不在北京的学生

-- select \* from students where hometown<>'北京'

逻辑运算

-- 例1：查询年龄小于20的女同学

-- select \* from students where age<20 and sex='女'

-- 例2：查询女学生或'1班'的学生

-- select \* from students where sex='女' or class='1班'

-- 例3：查询非天津的学生

-- select \* from students where not hometown='天津'

模糊查询

-- 例1：查询姓孙的学生

-- select \* from students where name like '孙%'

-- 例2：查询姓孙且名字是一个字的学生

-- select \* from students where name like '孙\_'

-- 例3：查询叫乔的学生

-- select \* from students where name like '%乔'

-- 例4：查询姓名含白的学生

select \* from students where name like '%白%'

范围查询

-- 例1：查询家乡是北京或上海或广东的学生

-- select \* from students where hometown in ('北京','上海','广东')

-- select \* from students where hometown not in ('北京','上海','广东')

-- 例2：查询年龄为18至20的学生  between 20 and 18 小值在前

-- select \* from students where age between 18 and 20

判空

-- 例1：查询没有填写身份证的学生

-- select \* from students where card is null

 非空

-- select \* from students where card is not null

判断身份证为空字符

-- select \* from students where card=''

排序 asc升序 可省略 desc降序

-- 例1：查询所有学生信息，按年龄从小到大排序

-- select \* from students order by age asc    --

-- 降序    -- select \* from students order by age desc

-- 例2：查询所有学生信息，按年龄从大到小排序，年龄相同时，再按学号从小到大排序

select \* from students order by age desc,studentNo

聚合函数

-- 例1：查询学生总数

-- select count(\*) as 学生总数 from students

-- select count(name) from students

-- count(card) 不统计为null数据

-- select count(card) from students

-- 例2：查询女生的最小年龄

-- select min(age) from students where sex='女'

-- 例3：查询1班的最大年龄

-- select max(age) from students where class='1班'

-- 例4：查询北京学生的年龄总和

-- select sum(age) from students

-- 例5：查询女生的平均年龄

select avg(age) from students where sex='女'

分组

-- 例1：查询各种性别的人数

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

-- 例2：查询各种年龄的人数

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

分组后过滤

-- 例1：查询男生总人数

-- select sex,count(\*) from students group by sex having sex='男'

select count(\*) from students where sex='男'

查询1班除外其他班级学生的平均年龄、最大年龄、最小年龄

-- 先过滤再分组

-- select class,avg(age),max(age),min(age) from students where class!='1班' group by class

-- 先分组再过滤

-- select class,avg(age),max(age),min(age) from students group by class having class!='1班'

分页

-- 获取部分行limit start,count  
-- 从start开始，获取count条数据 start索引从0开始  
-- 例1：查询前3行学生信息  
-- select \* from students limit 0,3  
-- 分页  
-- 每页显示m条数据，求：显示第n页的数据  
-- select \* from students limit (n-1)\*m,m  
-- 练习：  
-- 每页显示5条数据，显示每一页的数据  
-- select count(\*) from students  
-- 12/5=3 页  
-- 第一页  
-- select \* from students limit 0,5  
-- 第二页  
-- select \* from students limit 5,5  
-- 第三页  
-- select \* from students limit 10,5

**9.连接查询**

-- 等值连接

-- 例1：查询学生信息及学生的成绩

-- select \* from students,scores where students.studentno=scores.studentno

-- 方式二内连接

-- 不会产生笛卡尔积，不会产生临时表，性能高

-- select \* from students

-- inner join scores on students.studentno=scores.studentno

 -- 例2：查询课程信息及课程的成绩

-- select \* from courses,scores where courses.courseNo=scores.courseNo

-- select \* from courses

-- inner join scores on courses.courseNo=scores.courseNo

-- 例3：查询学生信息及学生的课程对应的成绩

select stu.name 姓名,sc.score 成绩,cs.name 课程名 from students stu,scores sc,courses cs   where stu.studentNo=sc.studentNo and sc.courseNo=cs.courseNo

内连接

select stu.name,sc.score,cs.name from students stu

inner join scores sc on stu.studentNo=sc.studentNo

inner join courses cs on sc.courseNo=cs.courseNo

-- 例4：查询王昭君的成绩，要求显示姓名、课程号、成绩

-- select stu.name,cs.courseno,sc.score from students as stu

-- inner join scores as sc on stu.studentNo = sc.studentNo

-- inner join courses as cs on cs.courseNo = sc.courseNo

-- where stu.name='王昭君'

-- 例5：查询王昭君的数据库成绩，要求显示姓名、课程名、成绩

-- select stu.name,cs.name,sc.score from students as stu

-- inner join scores as sc on stu.studentNo = sc.studentNo

-- inner join courses as cs on cs.courseNo = sc.courseNo

-- where -- stu.name='王昭君'

-- and cs.name='数据库'

-- 例7：查询男生中最高成绩，要求显示姓名、课程名、成绩

-- SELECT stu. NAME,cs. NAME,sc.score FROM --  students AS stu

-- INNER JOIN scores AS sc ON stu.studentNo = sc.studentNo

-- INNER JOIN courses AS cs ON cs.courseNo = sc.courseNo

-- WHERE --  sex = '男'

-- ORDER BY sc.score DESC -- LIMIT 0, 1

-- 左连接 join前面的表称为左表，join右边的表为右表

-- 例1：查询所有学生的成绩，包括没有成绩的学生

-- select \* from students -- left join scores on students.studentno=scores.studentno

-- 例2：查询所有学生的成绩，包括没有成绩的学生，需要显示课程名

-- select \* from students

-- left join scores on students.studentno=scores.studentno

-- join前面生成的结果为左表，后面的为右表

-- left join courses on scores.courseno=courses.courseno

-- 右连接 -- join前面生成的结果为左表，后面的为右表,把右边的数据全部显示出来

-- insert into courses values (0, '语文'), (0, '数学');

-- 例1：查询所有课程的成绩，包括没有成绩的课程

-- select \*from scores

-- right join courses on scores.courseno=courses.courseno

-- 例2：查询所有课程的成绩，包括没有成绩的课程，包括学生信息

-- select \* from scores

-- right join courses on scores.courseno=courses.courseno

-- left join students on students.studentno=scores.studentno

 -- 自关联

atitle     aid         pid

130000 河北省   
130100 石家庄市 130000  
130400 邯郸市 130000  
130600 保定市 130000  
130700 张家口市 130000  
130800 承德市 130000  
410000 河南省   
410100 郑州市 410000  
410101 中原区 410100  
410102 二七区 410100  
410103 金水区 410100  
410300 洛阳市 410000  
410500 安阳市 410000  
410700 新乡市 410000  
410800 焦作市 41000

-- 例1：查询河南省的所有城

-- 先copy一份areas

-- select \*from areas,areas\_copy where areas.aid=areas\_copy.pid and areas.atitle='河南省'

-- 方法二：不需要copy，可以通过多次查自己，必须起别名

-- select \* from areas as sheng,areas as shi where sheng.aid=shi.pid and sheng.atitle='河南省'

-- 例2：查询郑州市的所有区县

-- select \* from areas as sheng

-- inner join areas as shi on sheng.aid=shi.pid

-- where sheng.atitle='郑州市'

-- 例3：查询河南省的所有区县

-- select \* from areas as sheng

-- inner join areas as shi on sheng.aid=shi.pid

-- inner join areas as qu on shi.aid=qu.pid -- where sheng.atitle='河南省'

**11.子查询**

 标量子查询

-- 例1：查询班级学生的平均年龄   -- select avg(age) from students

-- 查询大于平均年龄的学生 21.4167   -- select \* from students where age>21.4167

-- 子查询返回的结果时一行一列：标量子查询

-- select \* from students where age>(select avg(age) from students)

-- 例2：查询王昭君的成绩，要求显示成绩

-- 先查询王昭君的学号   -- select studentNo from students where name='王昭君'

-- select \* from scores where studentNo=(select studentNo from students where name='王昭君')

 列子查询

-- 例3：查询18岁的学生的成绩，要求显示成绩

-- select \* from scores where studentNo='0' or studentno='2'

-- 先查询18岁的学生的学号

-- select studentNo from students where age=18

--select \* from scores where studentNo in('002','006')

-- 子查询返回的结果时一列多行：列级子查询

select \* from scores where studentNo in(select studentNo from students where age=18)

 行级子查询

-- 例4：查询男生中年龄最大的学生信息

-- select max(age) from students where sex='男'

-- select \* from students where sex='男'   -- and age=(select max(age) from students where sex='男')

-- select \* from students where (sex,age)=('男',30)

-- select sex,age from students where sex='男' order by age desc limit 1

-- 子查询返回的结果时一行多列：行级子查询

-- select \* from students where   -- (sex,age)=(select sex,age from students where sex='男' order by age desc limit 1)

 表级子查询

-- 例5：查询数据库和系统测试的课程成绩

-- select \* from scores sc

-- inner join courses cs on sc.courseNo=cs.courseNo

-- where cs.name in ('数据库','系统测试')

-- select \* from courses where name in ('数据库','系统测试')

-- 子查询返回的结果时多行多列：表级子查询

-- select \* from 数据源

select \* from scores sc

inner join (select \* from courses where name in ('数据库','系统测试')) c  on sc.courseNo=c.courseNo

-- 子查询中特定关键字使用

-- in 范围  
-- 格式: 主查询 where 条件 in (列子查询)   
-- any | some 任意一个  
-- 格式: 主查询 where 列 = any (列子查询) 在条件查询的结果中匹配任意一个即可,等价于 in  
-- >any 等于>some 大于子查询返回的结果中的任意一个  
-- all  
-- 格式: 主查询 where 列 = all(列子查询) : 等于里面所有   
-- 格式: 主查询 where 列 <>all(列子查询) : 不等一其中所有

**12.数据分表**

-- 创建类型表

-- create table good\_cates(

-- id int UNSIGNED PRIMARY KEY auto\_increment,

-- cate\_name varchar(10)

-- )

-- 把一个表查询出来的数据插入到另一个表中（要注明字段名）

-- insert into good\_cates(cate\_name) select distinct cate from goods

-- 创建品牌表

-- create table ...SELECT...

-- create table good\_brands(

-- id int UNSIGNED PRIMARY KEY auto\_increment,

-- brand\_name varchar(10) --

)

-- 查询出来的列必须对应表中的字段名，如果在表中找不到同名字段，会新建一个字段

-- select distinct brand\_name from goods

-- select \* from goods

-- inner join good\_cates on goods.cate=good\_cates.name

-- 将goods表和good\_cate连接起来

-- update goods

-- inner join good\_cates on goods.cate=good\_cates.cate\_name -- set goods.cate=good\_cates.id

--将goods表里的cate字段用good\_cates里的id替代

高级：

**13. 数据库的设计**

E-R模型 Entry-relationship实体关系模型

一对一（个人信息中的爱好） 不常用的表来维护

一对多 （班级与学生）多的表来维护

多对多（课程与学生） 再建一张表来维护

**14.命令行客户端的使用**

连接：mysql -uroot -p 回车 输入密码

查看所有仓库

show databases;

使用某个仓库

use 仓库名;

查看当前使用的数据库

select database();

创建数据库:

create database 数据库名 charset=uftf8;

删除数据库：

drop database 数据库名；

查看所有表：

show tables;

查看表结构：

desc 表名;

show create table 表名;

中文乱码：

set charset gbk;

 命令行备份与恢复：

备份：以管理员身份打开cmd

cd C:\Program Files (x86)\MySQL\MySQL Server 5.1\bin 回车

mysqldump –uroot –p密码 数据库名 > back.sql

恢复：

先创建新的数据库

mysql -uroot –p密码 新数据库名 < back.sql

**15.函数**

-- 拼接字符串  select concat(12,34,'ab');

select \* from students

select name,sex,hometown,concat(name,'是',hometown,'的',sex,'生') as des from students

-- 包含字符个数

select length('abc');

-- 截取字符串

select left('abc',1)

select left(name,1) from students;

select substring('abc123',2,3); -- pos 指的位置时从1开始

-- 去除空格

select ltrim('  bar   '),

rtrim('  bar   '),

rtrim(ltrim('  bar   '));

-- 大小写转换

select lower('aBcD'),

upper('aBcD');

-- 求四舍五入值round(n,d)，n表示原数，d表示小数位置，默认为0

select round(1.6,1);

-- 求x的y次幂  select pow(2,3);

-- 获取圆周率PI()  select round(PI(),10)

-- 随机数rand()，值为0-1.0的浮点数  select rand();  -- 随机取表中的一条  select \* from students order by rand() limit 1

-- 日期时间函数  select current\_date(),current\_time(),now();  select date\_format(now(),'%Y-%m-%d/%h/%i/%s');

-- 流程控制  case语法

select  case 3

when 1 then 'one'

when 2 then 'two'

else 'zero'

end as result;

-- 姓 美女 帅哥

select left(name,1) ,sex,

case sex

when '男' then concat(left(name,1),'帅哥')

when '女' then concat(left(name,1),'美女')

else '保密'  end as res  from students

-- 自定义函数

1、在navicat查询中，执行下面sql语句，可以在navicat中的数据库下面的函数中找到创建的自定义函数

create function my\_trim(aaa varchar(100)) returns varchar(100)

begin   return rtrim(ltrim(aaa));

end

 2、使用自定义函数

 select my\_trim('   name    ')

16.存储过程

1、在navicat查询中，执行下面sql语句，可以在navicat中的数据库下面的函数中找到创建的存储过程

create procedure proc\_stu()

begin   select \* from students;

end

2、使用存储过程

call proc\_stu();

15、视图

-- 创建视图

-- 视图可以隐藏真正的表结构，对于比较重要的数据（如银行），只让别人访问视图，没有权限使用真正的表

create view v\_stu\_score\_course as

select stu.name,stu.class,sc.score,cs.name as a from students stu

INNER JOIN scores sc on stu.studentNo=sc.studentNo

INNER JOIN courses cs on cs.courseNo=sc.courseNo    -

-- 使用视图

select \* from v\_stu\_score\_course

16、事物

  提交

   两个命令行客户端 左边客户端 右边客户端

   1、左边客户端：查询学生信息    select \* from students;

   2、右边客户端：开启事务，插入数据    begin;    insert into students(studentNo,name) values ('013','abc');

   3、右边客户端：查询数据，此时有新增的数据    select \* from students;

   4、左边客户端：查询数据，发现并没有新增的数据    select \* from students;

   5、右边客户端：完成提交    commit;

   6、左边客户端：查询，发现有新增的数据    select \* from students;

  回滚

   两个命令行客户端 左边客户端 右边客户端

   1、左边客户端：查询学生信息    select \* from students;

   2、右边客户端：开启事务，插入数据    begin;    insert into students(studentNo,name) values ('014','aaa');

   3、右边客户端：查询数据，此时有新增的数据    select \* from students;

   4、左边客户端：查询数据，发现并没有新增的数据    select \* from students;

   5、右边客户端：回滚    rollback;

   6、左边客户端：查询，发现没有新增的数据    select \* from students;

   7、右边客户端：查询，发现没有新增的数据    select \* from students;

17、索引

  创建测试表testindex

   create table test\_index(title varchar(10));

  创建存储过程，在navicat查询中，执行下面sql语句

   begin

   declare i int default 0;

   while i<100000 do    insert into test\_index(title) values(concat('test',i));

   set i=i+1;

   end while;

   end

  调用存储过程，向表中添加数据

   call proc\_test()

开启运行时间监测：

set profiling=1;

查找第1万条数据test10000

select \* from test\_index where title='test10000';

查看执行的时间：

show profiles;

为表title\_index的title列创建索引：

**create index 索引名称 on 表名(字段名称(长度))**

**create index title\_index on test\_index(title(10));**

执行查询语句：

select \* from test\_index where title='test10000';

再次查看执行的时间

show profiles;

-- 分析查询

explain   select \* from test\_index where title='test10000'

建表时创建索引

创建表时，对于主键和unique字段，自动创建索引     
create table create\_index(

id int primary key,

name varchar(10) unique,

age int,

**key (age)**

);

18、外键

从表添加、修改数据时，受主表的约束

主表删除数据时，如果已经被从表依赖了，不能删除

方式一：创建数据表的时候设置外键约束  
create table class(

id int unsigned primary key auto\_increment,

name varchar(10)

);  
create table stu(

name varchar(10),

class\_id int unsigned,

foreign key(class\_id) references class(id)

);  
foreign key(自己的字段) references 主表(主表字段)

方式二：对于已经存在的数据表设置外键约束  
alter table 从表名 add foreign key (从表字段) references 主表名(主表字段);  
alter table stu add foreign key (class\_id) references class(id);

删除外键

需要先获取外键约束名称

show create table stu;

获取名称之后就可以根据名称来删除外键约束

alter table 表名 drop foreign key 外键名称;

 修改密码

使用root登录，修改mysql数据库的user表 使用password()函数进行密码加密 注意修改完成后需要刷新权限  
use mysql;  
update user set password=password('新密码') where user='用户名';  
例： update user set password=password('123') where user='root';  
刷新权限：flush privileges;

忘记 root 账户密码怎么办

1、配置mysql登录时不需要密码，修改配置文件  
Centos中：配置文件位置为/data/server/mysql/my.cnf Windows中：配置文件位置为C:\Program Files (x86)\MySQL\MySQL Server 5.1\my.ini  
修改，找到mysqld，在它的下一行，添加skip-grant-tables  
[mysqld] skip-grant-tables  
2、重启mysql，免密码登录，修改mysql数据库的user表  
use mysql;  
update user set password=password('新密码') where user='用户名';  
例： update user set password=password('123') where user='root';  
刷新权限：flush privileges;  
3、还原配置文件，把刚才添加的skip-grant-tables删除，重启

补充：

Mysql安装和使用

 2.1 Linux Centos

  服务端使用

   打开Terminal命令行

   查看mysql进程是否开启

    ps ajx | grep mysql -- 查看mysqld是否存在

   root用户执行下面命令

   启动mysql服务

    service mysqld start

   停止mysql服务

    service mysqld stop

   重启mysql服务

    service mysqld restart

  客户端使用

   navicat

    把navicat112\_mysql\_cs\_x64.tar.gz拷贝到centos中，解压     打开start\_navicat程序，点击两次取消     如果试用期到期了，删除用户目录下的

     rm -r /home/admin/.navicat64/

   命令行客户端

    打开Terminal命令行，输入

     mysql -u root --回车

    退出命令行客户端 ctrl + d

 2.2 Windows中

   Mysql服务端，安装：安装路径不要有中文

   客户端

    命令行客户端：开始菜单-->mysql-->mysql server 5.1-->>mysql commond line client     输入MySQL安装的密码

    navicat工具：激活时必须以管理员身份运行navicat\_chs.exe

 2.3 Mysql客户端和服务端搭配场景

  windows中客户端连接windows中服务端

   1、确保mysql服务端是开启的

   2、打开navicat，输入密码，点击测试连接，测试成功后，点击确定

  windows中客户端连接centos中服务端

   1、检查windows和centos的网络是否畅通     在centos中获取ip，打开terminal，输入ifconfg，拷贝ip     在windows中cmd命令行，输入 ping ip地址，如: ping 192.168.133.159

   2、centos中设置mysql服务端允许其他电脑连接服务端     打开terminal     mysql -u root

    use mysql;

    update user set host='%' where host='::1';

flush privileges;

3、打开navicat，输入centos中mysql服务端的ip，点击测试连接，测试成功后，点击确定