

就业班 mysql-数据库

布尔教育 <http://www.itbool.com>

燕十八 念白白著

严禁传播 违者必究

第 1 课 数据库概念

留言本将留言存在哪里?-->文本文件

文本文件管理的不够好-->我们可以放在 数据库 里

数据库是什么呢?

它就是一个软件,它能帮我们管理数据的一个软件.

我们将数据给它,比如 一篇文章,一个人的年龄,名字,
放进数据库里,它能帮我们稳妥的管理起来,且效率挺高.
这种软件就叫 数据库

重要性

PHP->WEB 开发

PHP-----写入-->数据库

PHP<----查询---数据库

PHP 与 MySQL 就像炼钢和采矿的关系

学了 PHP 只是会炼钢,但矿石从哪儿来?还得会采矿

PHP的工作大部分是

PHP -> 连接数据库 -> 写入数据 -> 查出数据 -> 格式化数据 -> 显示出来

WEB开发想要做的好,学好数据库的知识是很重要的

面试在 8k 以上的工作, 数据库的知识 面试是必不可少的

数据库是管理数据的软件.(该类型软件肯定有多种)

一篇新闻,个人简介,评论内容等等,存储在哪儿了?

交给了计算机,由计算机的某个软件,管理着数据,这就软件称为数据库管理软件.

MySQL,SQLite, PostgreSQL,Oracle,DB2

安装 MySQL

这里我们用集成环境,wamp,不需要我们单独配置的安装mysql

第 2 课 客户端概念

安装完mysql之后,如何连接呢?

打开任务管理,查看 mysqld.exe, 数据由他管理.

它的位置在:

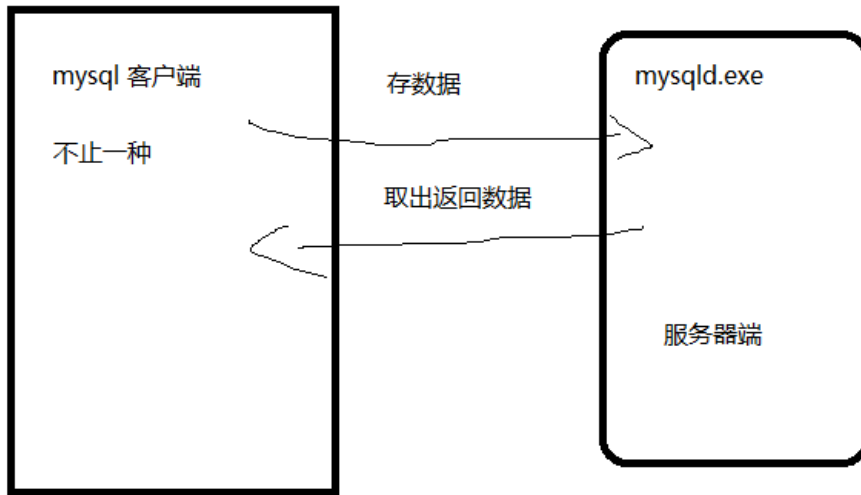
wamp\bin\mysql\mysql5.6.12\bin\mysqld.exe

通过这个进程,可以写数据,查数据.

但如何和这个进程"说话",需要客户端.

进程 -> mysqld.exe -> 是在一直运行的,数据全靠它在管理

mysql客户端 <=> mysqld.exe服务器端[24小时在运行] 存取数据是这两个在通话



C:\wamp\bin\mysql\mysql5.6.12\bin>mysql

使用客户端链接 mysql [以下是默认值]

-h localhost

-u root

-p

mysql客户端有多种

mysql(mysql.exe), phpmyadmin, navcat, mysql workbench

mysql.exe 是纯命令行的客户端,只能手敲命令

学习的时候,我们就使用它,来手敲sql来练习

我们安装的wamp集成环境中就有phpmyadmin

浏览器直接访问 localhost/phpmyadmin,就可以打开这个软件

它是用php写的客户端,可以通过浏览器来管理数据

第 3 课 表与库的概念

数据库管理数据,它是以什么样的形式组成的?

生活中的表---->表 table

多行多列,传统的数据库都是这样的;声明了表头,一个表创建好了,剩下的就是往表中添加数据

多张表放在档案中-->库 database

学生信息表,成绩表,记过表 -> 都是这个班级的信息,将这几张表放在一个文件夹中 -> 这个文件夹,文档就是 库

mysql服务器可以帮我们管理多个库

C:\wamp\bin\mysql\mysql5.6.12\data

数据库中的数据放在这个文件中, .MYD就是数据文件

档案袋管理员--> 服务器 server

数据库就是将我们的数据存储成文件,分文别类的管理起来

查看库 show databases;

选库 use test;

查看库下的表 show tables;

查表 select * from cat;

第 4 课 认识 SQL

SQL(Structured Query Language) 结构化查询语言

what 型的语言 where id > 3

而非 how 型的语言 if() else {}

php中

```
for($i=1;$i<=10;$i++) {  
    echo $i;  
}
```

是在告诉php如果去做

而what型的语言,是告诉它自己想要什么样的数据

直接描述自己想要什么样的数据就行;不需要考虑内部如何做的

cat_id = 3;name='lisi';

```
select * from msg where id=3;
```

sql语句大致可以分为三种:

DML is Data Manipulation Language statements. Some examples:数据操作语言, SQL 中处理数据等操作统称为数据操纵语言 [增删改查](#)

对我们而言,DML是我们的重点

DDL is Data Definition Language statements. Some examples:数据定义语言, 用于定义和管理 SQL 数据库中的所有对象的语言 (建设者的角度,建表,建库,建视图 等等, 15%)

一般来说,一个数据库在设计完毕之后很少会再改动了

DCL is Data Control Language statements. Some examples:数据控制语言, 用来授予或回收访问数据库的某种特权, 并控制数据库操纵事务发生的时间及效果, 对数据库实行监视等 (管理员角度,DBA[数据库管理员],5%)

这个用户是否有权限建表,等

第 5 课 insert 操作

DML,增删改查 -> 增

准备工作:

通过 phpmyadmin 导入"练习准备.sql"

desc user;

uid name age 三列

1.添加所有列

```
insert into user (uid,name,age) values (1,'lisi',23);
```

2.一行中有多个列,我们可以插入全部列,也可以插入部分列

但是:列与值要严格对应

```
insert into user (uid,name) values (1,'lucy');
```

3.主键自增插入 desc user;

```
insert into user (name) values ('yumi');
```

4.插入所有列的简写

```
insert into user values ('kimi',25); //报错:列计数不匹配值计数
insert into user values (3,'kimi',25);
```

数字可以加单引号,它也会转成int来理解

但是字符串必须加单引号,不加会理解为一个列名或者变量,会报错

```
insert into user values ('4','zhangsan','25');
insert into user values (5,zhangsan,25);
```

5.一次添加多行数据

```
insert into user values (5,'test1',44),(6,'test2',23),(7,'test3',18);
```

注意:

列与值, 严格对应 (id 自增列也必须对应)

数字和字符串的注意点

数字不必加单引号,字符串必须加单引号.

1 次添加多行数据,用逗号隔开

第 6 课 update 操作

增删改查 -> 改

改哪张表?

你需要改哪几列的值?

分别改为什么值?

在哪些行生效?

1.update 更改列

```
update user set age=99 where name='yumi';

update 表名 set
列1 = 新值1,
列2 = 新值2
where expr
```

1)sql是 what 型的语言,而不是how

如果是how,则需要这样写

```
for(所有行) {
    if(uid=2) {
        age = 23;
        name = 'nobody';
    }
}
```

what型

```
update user set age=23,name='nobody' where uid=2;
```

我们不用管它内部如何去做,只告诉它我们需要怎么修改即可

2.update 所有行

注意,不加 where 带来的后果

```
update user
set
name='mingming',
age=55;
```

数据是很宝贵的

如果我们update不加where条件,后果是很可怕的

mysql可以设置新手模式,在新手模式下,删除和更改不加where条件,它是拒绝执行的.

第 7 课 delete 操作

增删改查 -> 删

可不可以删除某一行中的某一列?delete...

其实这是个update操作

对于传统型数据库而言,一行就是它的原子型的单位

添加是一行,删除也要是一行

你要删除哪张表的数据?

你要删掉哪些行?

```
delete from 表名
where expr
```

*sql: *

```
delete from user where uid=7;
```

注意不加where条件

```
delete from user;
```

第 8 课 select 入门

增删改查 -> 查

查哪张表的数据?

你要选择哪些列来查询?

要选择哪些行?

select 列1,列2,...列n

from 表名

where expr

1.查询表的多有行所有列

```
select * from user;
```

开发中很少这样写,因为表中会有成千上万的数据,这样查询会增加数据库负担;

我们需要哪几条数据,就查询哪几条数据即可

2.查询一行

```
select * from user where uid=2;
```

3.查询多行

```
select * from user where uid>=2;
```

4.查询某几行的某几列,* 代表所有列

```
select uid,name from user where uid>=2;  
select name from user where uid=2;
```

dml占sql的80%,查占dml的80%;

我们所学的是最最基础的查询语句;

想要写出高难度的复杂的查询,我们还需专门讨论查询的模型的问题;

看我们是否能正真理解select,写出强大的查询语句,

要靠select的查询模型,非常之重要

第 9 课 select 查询模型(重要)

查询模型的重要性

```
Select * from user where uid=2;
```

为什么取出第 2 行?

1.列是变量

每一行,变量值在变

从上到下,遍历行,where 是表达式,当 值为真,则取出该行

这句话是什么意思?

```
select * from user where 1; //取出所有行  
select * from user where 0; //一行都取不出来
```

2.变量可以计算

计算明年多少岁

```
select name,age+1 from user where 1;
```

goods表

查询本店的商品比市场价便宜多少

```
select goods_id,goods_name,shop_price,market_price,market_price-shop_price from goods;
```

3.投影的概念

```
select name,age from user;
```

user表有三列,我们只取出2列(部分列),叫做**投影运算**

就像手电筒,只照到两列,投出影子显示出来

goods表查询出来 market_price-shop_price

两个列做运算,叫做**广义投影**

第 10 课 查询练习

goods表和category表

复习秘籍

开发中我们能碰到的这些查询,秘籍中基本都涵盖了

一定要把秘籍上的东西都好好练习一遍

工作中98%的查询,都是秘籍中所涵盖的查询类型

分析goods表结构,有哪些列

比较运算符

运算符	说明	运算符	说明
< <= = in	小于 小于或等于 等于 在某集合内	!= 或 <> >= > between	不等于 大于或等于 大于 在某范围内

逻辑运算符

运算符	说明
NOT 或 ! OR 或 AND 或 &&	逻辑非 逻辑或 逻辑与

秘籍第三部分,查询练习之 1.1-1.14 挨个讲解

1.10:取出第1个栏目下面的商品(注意:1栏目下面没商品,但其子栏目下有)

```
select goods_id,cat_id,goods_name,shop_price,click_count from ecs_goods
where cat_id=1;
```

```
select goods_id,cat_id,goods_name,shop_price,click_count from ecs_goods
where cat_id in (2,3,4,5);
```

栏目2,3,4,5是栏目1的子栏目,后面我们可以用子查询来直接查询栏目1下的商品

课后作业:

复习秘籍中第三部分第一小节的习题

第 1 小节后的面试题 1.15,1.16

第 11 课 习题讲解

讲解1.15,1.16 两道练习题

1.15

```
select floor(num/10)*10,num from mian;

update mian set num=floor(num/10)*10 where num between 20 and 39;

select * from mian;
```

1.16

提示:大胆的把列看成变量,参与运算,甚至调用函数来处理。

substring(),concat()

```
select goods_id,substring(goods_name,4) from goods where goods_name like '诺基亚%';

select goods_id,concat('HTC',substring(goods_name,4)) from goods where goods_name like '诺基亚%';

update goods set goods_name=concat('HTC',substring(goods_name,4)) where goods_name like '诺基亚%';
(为了不破坏表的数据,此处不予修改)
```

第 12 课 奇怪的 NULL

给user表插入null

```
insert into user values (4,'null',23);
desc user;
报错,因为我们在建表是声明了 not null
```

新建一个tmp表

```
create table tmp (
id int,
name char(10)
);
insert into tmp values (1,'liubei'),(2,null);
```

查询

where name=null;

where name!=null;

select null=null;

select null!=null;

where name is null;

where name is not null;

null比较特殊,它需要有自己的谓词来查,不便于优化,所以一半我们要尽量避免用null

第 13 课 group 分组与统计函数

统计函数

count() 计算行数

avg() 求平均函数

sum() 求总和

min() 求最小

max() 求最大

1.查询所有商品的平均价格

```
select avg(shop_price) from goods;
```

2.查询最贵的商品

```
select max(shop_price) from goods;
```

3.查询最便宜的商品

```
select min(shop_price) from goods;
```

4.查询一共有多少种商品

```
select count(*) from goods;
```

5.查询商品的总价格

```
select sum(shop_price) from goods;
```

6.查询本店积压的商品总货款

```
select sum(shop_price*goods_number) from goods;
```


经常统计的时候是分组统计 group by

1.帮我统计每个栏目下商品的平均价格

```
select cat_id,avg(shop_price) from goods group by cat_id;
```

分组查询,一半是比较耗费资源的,因为数据会先按照分组进行排序,再筛选出你所需的内容出来;如果索引建立的比较巧妙,在分组的时候有可能根据索引不需要排序;

所以分组计算能避免则避免,它比较浪费资源;

2.查询每个栏目下有多少种商品

```
select cat_id,count(*) from goods group by cat_id;
```

3.查询每个栏目下最贵的商品价格

```
select cat_id,max(shop_price) from goods group by cat_id;
```

4.查询每个栏目下商品的库存量

```
select cat_id,sum(goods_number) from goods group by cat_id;
```

最贵商品价格

最低商品价格

商品平均价格

商品库存量

商品种类

我们按照分组筛选取出cat_id,如果我们想取goods_id,那么会显示什么呢?按照栏目分组,那我们取出的每个栏目下的cat_id肯定是一样的,但是每个栏目下的商品的goods_id是不同的,如果我想取出goods_id,那会是哪个商品的goods_id呢?

例:统计205宿舍的平均身高,那我取出平均身高还想取出姓名,取谁的姓名呢?

```
select goods_id,cat_id,sum(goods_number) from goods group by cat_id;
```

既然取goods_id是有问题的,为什么还能取出数据呢?

在oracle和sql server 里面是会报错的,语法错误;

但是mysql允许这么写,稍微特殊一点,我们观察一下,取出的goods_id是谁的goods_id?

mysql会取出,在自然排序状态下,按照栏目分组,第一次出现的商品的goods_id;

我们要知道,这个语义的有问题的,不合法的

课后作业:

复习秘籍中第三部分第二小结,2.1--2.8

第 14 课 having 筛选

1.查询比市场价省钱200元以上的商品及该商品所省的钱(where和having分别实现)

```
select goods_id,shop_price,market_price,(market_price-shop_price) from goods where (market_price-shop_price)>200;
```

这个sql语句,计算了两次,没有必要.

我们知道,在php中,两个变量计算的结果可以赋值给另一个变量

```
select goods_id,shop_price,market_price,(market_price-shop_price) as sheng from goods where sheng>200;
```

会报错,找不到sheng;

什么呢？

我们知道mysql是数据库服务器,帮我们管理数据;说到底,他把我们的数据管理到哪里去了呢?管理到了文件上

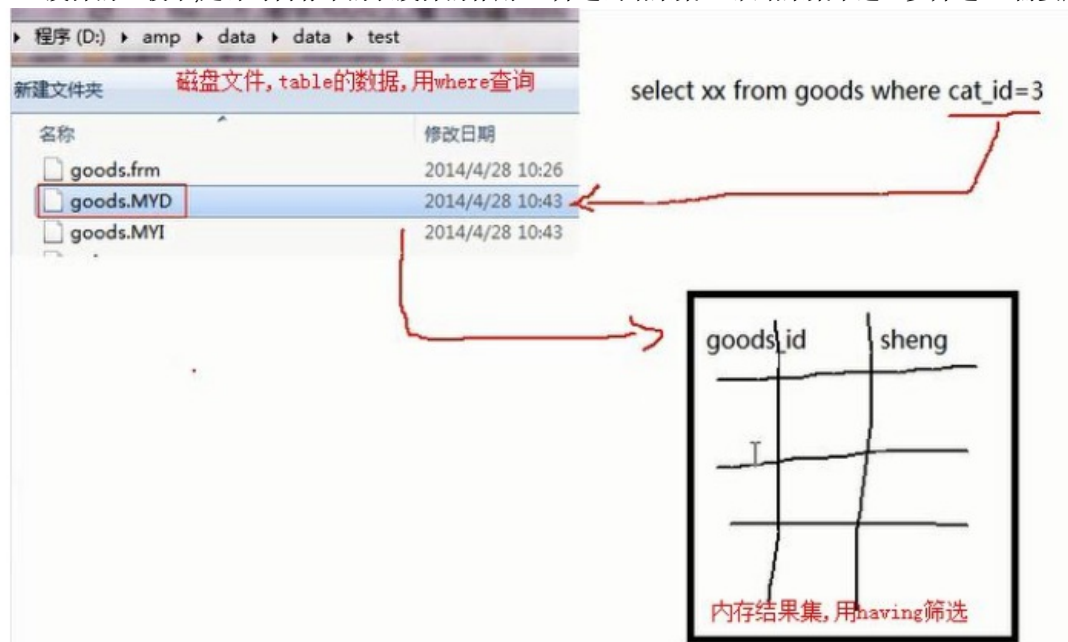
我们的商品表 -> data/test/goods.MYD

where -> 条件筛选的是磁盘上的表的字段

而sheng是结果集上的字段

我们需要筛选结果集上的字段,我们可以将结果集当成表,再一步筛选,只不过不能用where,需要用having;

where发挥的比较早,是针对内存中的表发挥的作用 -> 筛选出结果集 -> 从结果集中进一步筛选 -> 需要用到having



```
select goods_id,shop_price,market_price,(market_price-shop_price) as sheng from goods having sheng>200;
```

where和having都存在,where要放在having的前面;

因为where是对内存中的表进行筛选,而having是对进一步计算出来的结果集进行再一步筛选

课后作业:

复习秘籍中第三部分第3小节

给15分钟

3.7:where-having-group综合练习题

有如下表及数据

```
+-----+-----+-----+
| name | subject | score |
+-----+-----+-----+
| 张三 | 数学 | 90 |
| 张三 | 语文 | 50 |
| 张三 | 地理 | 40 |
| 李四 | 语文 | 55 |
| 李四 | 政治 | 45 |
| 王五 | 政治 | 30 |
+-----+-----+-----+
```

要求:查询出2门及2门以上不及格者的平均成绩

第 15 课 having 综合案例

1.张三挂科两门,计算分均成绩,需要把他的三门成绩都算上
result表

第一种错误,求平均,并 where score<60

```
select name,avg(score) from result where score<60 group by name;
```

第 2 种错误,计算挂科量,having 筛选

```
select name,count(score<60) as gk,avg(score) from result group by name having gk>=2;

insert into result values
('赵六','地理',80),
('赵六','八股',90);

添加完数据后,再运行语句
select name,count(score<60) as gk,avg(score) from result group by name having gk>=2;

能明显看出错误,看gk一行
```

count是计算行数,它取决于筛选的条件,而不是它括号里面的条件;
所以我们count(score<60)筛选的不是挂科数;count(*)返回结果是一样的

这个题目我们后面学到子查询,或者左右链接都可以做,但是反而会麻烦,我们这里用 where - having - group by 完全可以解决,不需要用其他的知识点

我们需要改变一下思路,sql是what型语句,我们要告诉他我们要什么

1.先求出所有人的平均成绩

```
select name,avg(score) from result group by name;
```

2.再看表中哪些人的科目是 score<60;

凡运算必有运算结果,比较运算的结果 返回真假1,0

```
select result.*,score<60 from result;
```

3.加挂科两门的人筛选出来

挂科数 sum(score<60) as gk

```
select name,avg(score),sum(score<60) as gk from result group by name having gk>=2;
```

第 16 课 order by 排序

在查出来的数据中,往往不是我们所需要的,我们需要的是按照某种规则排序后的结果;

排序在生活中也是随处可见的

有了排序才有评比 -> 有了评比才有幸福感

排序是按照某个属性来排的,身高,长相...

而在数据库中,是按照某个列来排的,那有正序也有倒序

我们在电话本和微信中,联系人是按照昵称拼音排序的

磁盘上的文件有可能是直接排好序的,也可能是没拍好序的

没拍好的需要到内存中再次排序,比较耗费资源,

我们也可以通过后面学的是数据库优化,建立良好的索引,在磁盘上直接完成

1.goods表按照价格排序

```
select goods_id,goods_name,shop_price from goods order by shop_price;
```

我们可以看到取出的数据,价格是递增的,那默认情况是升序排序,从低到高,asc

desc 是降序排列.从高到低

2.按价格由高到低排序

```
select goods_id,goods_name,shop_price from goods order by shop_price desc;
```

有可能出现,按照拼音无法排序的情况,拼音都一样
张丽 张力 -> 按笔画排序

3.按照栏目排序

```
select goods_id,goods_name,cat_id,shop_price from goods order by cat_id asc;
```

在相同的栏目中,又如何排序的呢,谁排前谁拍后呢?

4.按照栏目升序排列后,在每个栏目中按照价格降序排列

```
select goods_id,goods_name,cat_id,shop_price from goods order by cat_id asc,shop_price desc;
```

如果前面的条件,有比不出来的,可以继续往后追加条件继续比较

课后作业:

复习秘籍第三部分的 4.1-4.3

第 17 课 limit 限制取出条目

order by 一般和 limit 配合使用,功能才会更强大

limit 有选择的取出1条或多条

limit 限制取出条目

limit offset ,N , 跳过 offset 条,取 N 条

例:

取前 3 名, limit 0,3

取前 3 到前 5 名, limit 2,3

取最新的商品 order by goods_id desc limit 0,1;

对于 offset 为 0,可以简单为 limit N;

页码与 limit 的关系

1.取出价格最高的前三名商品

按照价格降序排列 -> 取出前三行

```
select goods_id,goods_name,shop_price from goods order by shop_price desc limit 0,3;
```

2.取出点击量是前三名到前无名的上屏

```
select goods_id,goods_name,shop_price,click_count from goods order by click_count desc limit 2,3;
```

3.取出最新商品

```
select goods_id,goods_name,shop_price from goods order by goods_id desc limit 0,1;
```

limit 做blog分页

假设每页显示5条

第一页取出最新的5篇文章 limit 0,5

第二页取出的文章为 limit 5,5;

第n页取出的文章数为 limit (n-1)*5,5

课后作业:

第 18 课 子句的查询陷阱

认识子句的查询顺序

select后的子句 where, group by, having, order by, limit

查询每个栏目下最新的产品(goods_id 最大即为最新)

要求: 只出现 1 次 select 查询

时间:20 分钟

只用一条select语句是无法查询出来的

1.常见的错误1

虽然能查出内容,但是语句是有问题的

```
select max(goods_id),goods_name,cat_id from goods group by cat_id;
```

取出的每行的内容是不正确的,goods_id正确,但是对应的goods_name确实不对的;它取出的是,按照cat_id分组排序后,第一个出现行的goods_name

2.常见错误2

先按照goods_id排序在分组

```
select max(goods_id),goods_name,cat_id from goods order by goods_id desc group by cat_id;
```

直接报错

1 条select 解决不了,需要子查询或连接查询等

想,如果有这样一张表,按照栏目拍好序,且每个栏目中,按照goods_id desc排序的

```
select goods_id,goods_name,cat_id from goods order by cat_id,goods_id desc;
```

针对我们上面的sql语句取出的内容

我们group by cat_id,就正好可以取出每个栏目下最新的商品

5种子句是有严格的顺序的

where,group by,having,order by,limit

这5种子句可以选择的写其中几种,但是顺序要按照上面的顺序来

第 19 课 where 型子查询

1: 查询最新的商品,以 goods_id 最大为最新

```
order by goods_id desc limit 1;
```

2: 不用 order by 如何做?

```
where goods_id=32;
```

3: 上条语句有何局限?

4.查询最新商品的id(始终得到最大的 goods_id)

```
select max(goods_id) from goods;
```

5 合并语句

```
select goods_id,goods_name from goods where goods_id=(select max(goods_id) from goods);
```

内层的select 查询结果,放在了外层的查询条件里面;这种查询叫做where型子查询

有了where型子查询,可以查询每个栏目下最新的商品

6 分组查询每个栏目下最大的 goods_id

```
select cat_id,max(goods_id) from goods group by cat_id;
```

7 合并

```
select goods_id,goods_name from goods where goods_id in (select max(goods_id) from goods group by cat_id);
```

第 20 课 from 型子查询

```
select goods_id,goods_name,shop_price from goods where goods_id between 20 and 25;
```

以上sql输出的查询结果,也可以看成一个多行多列的表,针对这个查询出来的结果集,这个表,我们可以继续select

这里可以用from型子查询,查出每个栏目下最新的商品

1.先按cat_id asc,goods_id desc排序

```
select goods_id,cat_id,goods_name from goods order by cat_id asc,goods_id desc;
```

2.合并

对上面sql语句查出的结果看作一张tmp表,这张表的每个栏目的第一行刚好是该栏目下goods_id 最大的一行

```
select goods_id,goods_name,cat_id from (select goods_id,cat_id,goods_name from goods order by cat_id asc,goods_id desc) as tmp gro
```

from型子查询:

先select 查询出结果集看作一张表,起一个别名,再在外层select这张临时表中的内容;对于外层是select,5中子句依旧可以使用

第 21 课 exists 型子查询

题目:

查询所有有商品的栏目

1.什么叫有商品的栏目?以cat_id=1为例

```
select * from goods where cat_id=1;
```

以上sql,能查出内容,是不是就是该栏目下有商品

2.查询栏目表

```
select * from category
```

3.合并

查看栏目表,看这个栏目表对应的商品是否存在

查询栏目表,如果该cat_id在goods表能查出内容,则该栏目下有商品

exists 存在的意思

```
select * from category where exists (select * from goods where category.cat_id=goods.cat_id);
```

```
select * from category where exists (select * from goods where goods.cat_id=category.cat_id)
```

cat_id	cat_name	parent_id
1	手机类型	0
2	CDMA手机	1
3	GSM手机	1
4	3G手机	1
5	双模手机	1
6	手机配件	0
7	充电器	6
8	耳机	6
9	电池	6

✓ MySQL 返回的查询结果为空（即零行）。

```
select * from goods where cat_id=6
```

课后作业：

第三部分 7.1-7.5

第 22 课 新手 1+N 模式查询

题目：

查询价格大于2000元的商品及其栏目名称

1.价格大于2000元的商品

```
select goods_id,goods_name,cat_id,shop_price from goods where shop_price>2000;
```

```
$sql = 'select goods_id,goods_name,cat_id,shop_price from goods where shop_price>2000';
$rs = mysql_query($sql);
$data = array();
while($goods) {
    $data[] = $rs;
}

foreach ($data as $k=>$v) {
    $sql = 'select cat_name from category where cat_id='.$v['cat_id'];
}
```

1条sql语句 -> N条数据 -> N条查询

我们需要一条sql语句查询出来,那有没有可能从一张表中就能查出来？

不可能,查询的字段是分别在两张表中的,肯定需要从两张表中查询出来 -> 连接查询

第 23 课 内连接查询

1 取出 boy,girl 表数据

2 hid 为 homeid,相同者为一对

如何取夫妻双方的名称和hid

看 hid.doc 演示

经过匹配后,能取出几行

左边的表的每一行,都有一次机会跟右边表去试图连接

不过能不能连接上是有条件的

```
select boy.hid,boy.bname,girl.hid,girl.gname
from boy inner join girl
on boy.hid=girl.hid;
```

hid	bname	hid	gname
B	杨过	B	小龙女
C	陈冠希	C	张柏芝

第 24 课 左连接及右连接查询

- 1 左连接右连接类似,只是调个方向,
- 2 所有男士站出来,找到自己的另一半,没有的,以 NULL 补齐.

```
select boy.hid,boy.bname,girl.hid,girl.gname
from boy left join girl
on boy.hid=girl.hid;
```

hid	bname	hid	gname
B	杨过	B	小龙女
C	陈冠希	C	张柏芝
A	屌丝	NULL	NULL

左连接的特点:

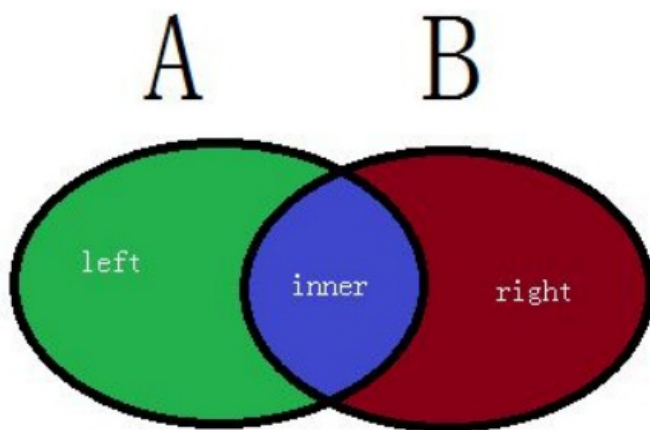
左表所有的数据全部取出来,然后到对应的右表去找数据,如果找不到数据,补null

右连接,以右表为准,全部取出

```
select boy.hid,boy.bname,girl.hid,girl.gname
from boy right join girl
on boy.hid=girl.hid;
```

hid	bname	hid	gname
B	杨过	B	小龙女
C	陈冠希	C	张柏芝
NULL	NULL	D	死宅女

图解左,右,内



内链接为左右连接的并集

那有没有左右连接的交集,并集是外连接?mysql中不支持外连接,oracle和sql server可以

习题:

5.1 取出所有商品的商品名,栏目名,价格

```
select goods_id,goods_name,cat_name,shop_price
from goods left join category
on goods.cat_id=category.cat_id;
```


注意:

为什么要在列名前加上 表名. goods.cat_id

因为goods表和category表都有cat_id这个字段,要说明这个是哪张表的cat_id

连接查询,通过on将两张表打通一个桥梁,拼接成一张大表

既然是一张大表,我们后面是不是可以跟着写那5个子句 where group by ...

5.2 取出第4个栏目下的商品的 商品名,栏目名,价格

```
select goods_id,goods_name,goods.cat_id,cat_name,shop_price
from goods left join category
on goods.cat_id=category.cat_id
where goods.cat_id=4;
```

5.4: 用友面试题

10分钟时间写这道题

第 25 课 经典连接查询面试题

m表和t表

5.4: 用友面试题

1.m表和t表内连接查出,hname

```
select m.*,t.tname
from m inner join t
on m.hid=t.tid;
```

mid	hid	gid	mres	matime	tname
1	1	2	2:0	2006-05-21	国安
2	2	3	1:2	2006-06-21	申花
3	3	1	2:5	2006-06-25	布尔联队
4	2	1	3:2	2006-07-21	申花

2.将上面sql查出来的表当作一张大表,继续inner join

```
select m.*,t1.tname as hname,t2.tname as gname
from m inner join t as t1
on m.hid=t1.tid
inner join t as t2
on m.gid=t2.tid;
```

mid	hid	gid	mres	matime	hname	gname
1	1	2	2:0	2006-05-21	国安	申花
2	2	3	1:2	2006-06-21	申花	布尔联队
3	3	1	2:5	2006-06-25	布尔联队	国安
4	2	1	3:2	2006-07-21	申花	国安

3.按照要求取出列

```
select hid,t1.tname as hname,mres,gid,t2.tname as gname,matime
from m inner join t as t1
on m.hid=t1.tid
inner join t as t2
on m.gid=t2.tid;
```

hid	hname	mres	gid	gname	matime
1	国安	2:0	2	申花	2006-05-21
2	申花	1:2	3	布尔联队	2006-06-21
3	布尔联队	2:5	1	国安	2006-06-25
2	申花	3:2	1	国安	2006-07-21

4.查出 2006-6-1 到2006-7-1之间举行的所有比赛

```
select hid,t1.tname as hname,mres,gid,t2.tname as gname,matime
from m inner join t as t1
on m.hid=t1.tid
inner join t as t2
on m.gid=t2.tid
where matime between '2006-06-01' and '2006-07-01';
```

注意:

t表被连续连接两次,需要起别名

第 26 课 union 查询

什么事union查询?

union查询就是把2条或多条sql的查询结果,合并成1个结果集

sql1 返回N行

sql2 返回M行

sql1 union sql2 ,返回 N+M行

1.用union查询栏目3和栏目4下面的商品

```
select goods_id,goods_name,cat_id,shop_price from goods where cat_id=3
union
select goods_id,goods_name,cat_id,shop_price from goods where cat_id=4;
```

2.查询网站此时用户,合并可以是两张相同的表,也可以是不同的表

用户表user和临时用户表tmp

```
select uid,name from user
union
select id,name from tmp;
```

union 语句必须满足 1 个条件:

各语句取出的列数相同

如果不相同,会报错:列数不同,如下sql

```
select uid,name,age from user
union
select id,name from tmp;
```

可以看出

列名称未必要一只,查询出的结果列名称会使用第一条sql的列名称为准

3.union 查询 a表和b表,看返回的是不是N+M行

```
select * from a
union
select * from b;
```

应该是返回8条数据,但是只返回了7条数据;

注意：

使用 union 时,完全相等的行,将会被合并.

合并是比较耗时的操作,两行会在比较看是否完全相等
一半不让union进行合并,使用"union all" 可以避免

```
select * from a
union all
select * from b;
```

实际中,一般使用union all来合并查询

4.讨论,union的子句中,不用写order by

```
(select * from a order by num desc)
union all
(select * from b order by num desc);
```

我们发现,写了 order by 并没有发挥作用;

sql合并后得到的总的结果,可以 order by,子句的order by 失去意义

比如:

比如,华西村,南街村,2 个村各出 3 人,去当兵,

那么在村子里排序是没有什么意义的, [x,f,d],[z, k, c]

到部队里,还会对这些人再进行排序

所以,合并后仍是无序的,因此子句内排序没什么意义.

union合并后得到的总的结果集,是可以去排序的

```
(select * from a order by num desc)
union all
(select * from b order by num desc)
order by num desc;
```

课后作业:

给10分钟写下面这道题

6.2:3期学员碰到的一道面试题

第 27 课 union 面试题

讲解:

6.2:3期学员碰到的一道面试题

1.查询a union b表

```
select * from a
union all
select * from b;
```

2.查询出来的结果当作一张表,在select

```
select id,sum(num)
from
(select * from a
union all
select * from b) as tmp
group by id;
```

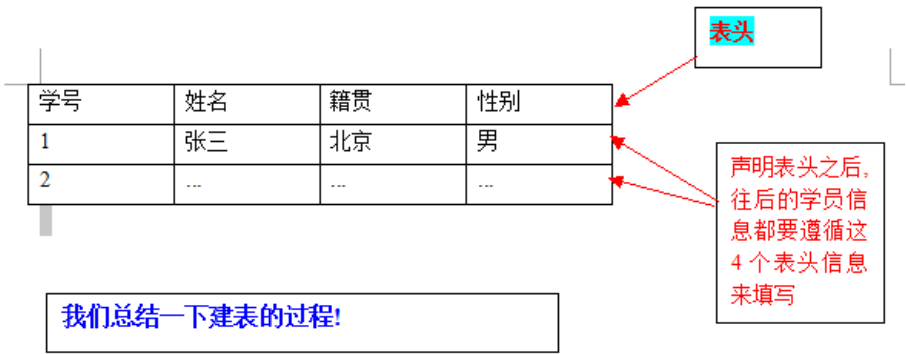
3.学员自己尝试用left join 左连接 完成该题

4.以上27课
我们学习了DML,数据操纵语言
今天我们学习的内容+复习秘籍上的内容
涵盖了工作中95%以上的查询问题 -> 复习秘籍一定要好好练习
至此,sql语句,DML部分,已经讲解完毕
下面我们将学习,数据库定义语言 -> DDL
目前别人给我们现成的表,我们会增删改查
但是我想自己创建表,创建库 -> 接下来将学习DDL

第 28 课 创建表 table

之前我们着重学习的是 DML -> 数据操纵语言
别人给我们建好了表,我们可以增删改查数据,特别是查询,变化非常大
如果别人没有把表给我们,我们连上数据库,能不能自己建一张表,更具项目的要求,做一个博客,一个商城,具体分析这个商城项目应该如何建表
已建设者的角度,建表,建库,建视图 等等, -> DDL 数据定义语言
什么叫定义?
DML是操作表中一行一行数据的,而定义是建立这一张表到底是什么样子的;
最基本的 -> 如何创建一张表

首先在A4纸上建立一张表



1.我们往表中插入的数据,是否属于建表的过程?
不是 -> 属于 DML操作

2.表什么时候已经建好了?
我们把表头声明完之后,表就已经声明好了;
所谓建表的过程 -> 就是声明表头的过程
表头的每一个想 -> 我们叫做列
拿到数据中来说:
我们建表的过程 => 就是 声明列 的过程

3.第一个简单的建表语句

```
create table t1(  
  sn int,  
  name varchar(10)  
);
```

可以看到我们建的表已经存在了

```
show tables;
```

4.我们用word建表,每个列是等宽分布的
而实际在数据库中,表中的列并不是等宽的
像: 学号 姓名 家庭地址 性别

学号	姓名	家庭住址	性别
----	----	------	----

我们要根据这个字段有可能要存储的最大的长度,来给列分布一个最合理的宽度

如果分配的太宽了,A4纸就浪费了

如果分配的太窄了,有可能内容就放不下了

数据库将我们的内容放在磁盘上,我们的内存也是有限的

所以我们在建表的时候需要考虑列的属性

我们要分析,就是这个列,最大能存多少,然后我们选取一个合理的列类型,合理的列宽度;使之既能放下内容,又不浪费磁盘空间,还要查询速度快

建表的过程,就是 声明列 的过程

列 选 什么类型的列???

列 给 什么样的属性???

我们建好一张表,首先要学习 => 列的类型与其属性的知识

```
create table 表名 (  
    列1 列类型[ 列属性 默认值],  
    列2 列类型 [列属性 默认值],  
    .....  
    列n列类型 [ 列属性 默认值]  
);
```

第 29 课 整型列

我们现在就来讨论:

mysql中,各种列的类型及特点

列类型大致分为3类:

1.数值型

整型,浮点型,定点型

2.字符串

char,varchar,text

3.日期时间类型

2012-12-13

14:26:23

我们先看存数值用的 整型

1.我们声明了t1表的 sn int

```
insert into t1 values (1,'lisi');
```

那这个 1 在磁盘占据几个字节? 4个字节

1个字节有8个位,每个位上有0,1两种可能

0000 0000

那4个字节 int型 在磁盘存1 是如下的存法

00000000 00000000 00000000 00000001

int型 存储的范围是多少?

11111111 11111111 11111111 11111111
0 ~ 2^32-1 [40多亿]

说明:
一个列,占的字节越多,存储的范围越大

mysql手册 -> 列类型 -> 数值类型

类型	字节	最小值	最大值
		(带符号的/无符号的)	(带符号的/无符号的)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

没必要死记,记住大致范围即可

整型:
bigint 8
int 4 字节
mediumint 3
smallint 2
tinyint 1

人的年龄: tinyint 无符号类型

第 30 课 整型列的可选参数

理解并能应用 unsigned,zerofill 及 M 属性

以tinyint为例

-128 ~ 127
0 ~ 255
一. unsigned 无符号,列的值从0开始,不为负

1.建一个t2表

```
create table t2(  
  num tinyint  
);
```

2.给t2表插入数据试一下

```
insert into t2 values (255);  
insert into t2 values (-128);
```

默认是有符号类型,如果我们有一个age列,不需要有负数
可以设置属性

unsigned 无符号,列的值从0开始,不为负

3.给t2表增加一个 无符号类型的列

```
alter table t2 add unum tinyint unsigned;
```

二. zerofill 适用于 学号,编码等,固定宽度的数字,可以用 0 填充至固定宽度

zerofill 填充至多宽? M 来指定

学号--> 1 -> 0001

学号--> 123 -> 0123

1.给t2表增加一个学号列

```
alter table t2 add sn tinyint(5) zerofill;
```

2.给sn插入数据

```
insert into t2 (sn) values (1);
```

注意:

在生活中,碰到0填充的数字,在生活中它会是负数吗?

3.查看表结构

```
desc t2;
```

我们可以看出,插入的sn表,为unsigned 类型

注意:

zerofill 属性默认决定列为 unsigned

M必须要跟unsigned配合使用才有意义

插入的m列是不是只能插入0-9呢?

```
alter table t2 add m tinyint(1) unsigned;
```

```
insert into t2 (m) values (10);
```

M 宽度不是限制我们插入数字的范围,0-9就一个数字宽,不是的

而是指,0填充时,我给你填到多宽,

如果不用0填充,M参数完全可以不写,因为它没有任何意义

第 31 课 浮点列与定点列

float 浮点型

double 范围更大的浮点型

decimal 定点型

1 存小数

float -3.402823466E+38 到-1.175494351E-38、0 和 1.175494351E-38 到 3.402823466E+38。

这些是理论限制, 基于 IEEE 标准。实际的范围根据硬件或操作系统的不同可能稍微小些。

double 更大,没必要去记忆他们的范围,除非搞天文

2 float, (M,D) ,double(M,D)

FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]

整型的M遇到zerofill才会起作用

float的M和D 只要设置,就会立即起作用

M 是精度,总位数, D 标度, 小数点后面的位数

float(5,2) -999.99 – 999.99, 分别用范围之外的数字测试

```
create table t4(  
money float(5,2);
```

```
);
```

```
insert into t4 values (9999);
insert into t4 values (999.99);
insert into t4 values (1000);
insert into t4 values (-999.99);
insert into t4 values (-1000);
```

3 decimal(M,D) 定点

创建 float(9,2),decimal(9,2)并测试

```
create table t5(
f float(9,2),
d decimal(9,2)
);
```

给t5表插入数据

```
insert into t5 values (1234567.23,1234567.23);
```

查询t5表的内容

```
select * from t5;
```

我们可以清晰的看到,f列,我们插入的数据跟显示的数据是不同的

float/double , 有精度损失

decimal 定点型,更精确

定点型,是将整数部分和小数部分用分别用数字来存储的,所以定点型更精确

第 32 课 字符型列

字符型:

char varchar text/blob enum

char 定长存储内容

varchar 变长存储内容

char和varchar有什么区别呢?

就如投币公交与按站收费的长途公交的区别

char(M),varchar(M)

假设M是10,10个字符的宽度

1.char(10); 定长

给定列10个字符的宽度,最多能存10个字符

如果我们insert一个字符

那这一个字符也占据10个字符的宽度

char就好比你去吃自助餐,吃一次50;

那吃多吃少你都要花50元

2.varchar(10); 变长

给定列10个字符的宽度,最多能存10个字符

如果我们insert一个字符

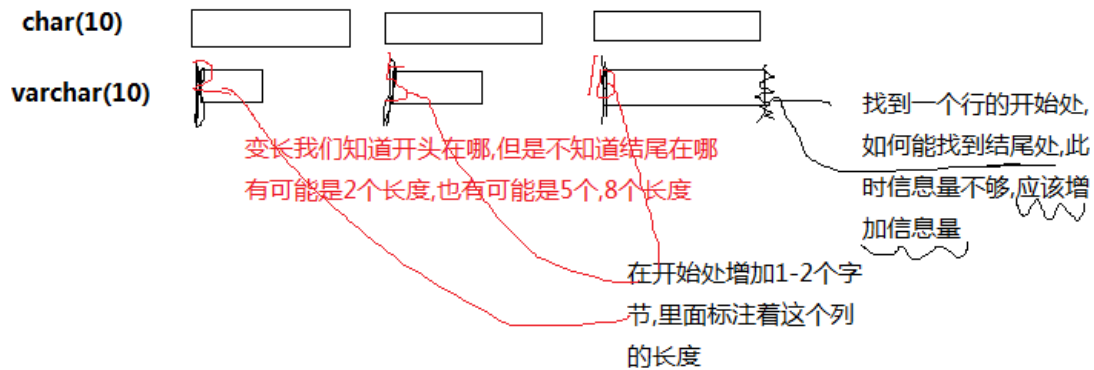
那么这一个字符占据多少个字符宽度?

肯定不会是占据1个字符的宽度,用多少占多少,利用率100%,就没有char存在的必要了

那么varchar到底占据多少?

讨论:

定长时,截 N 字节就是这一列的内容,如果变长,到底截多长呢?



类型	宽度	可存字符	实存字符($i \leq M$)	实占空间	利用率
Char	M	M	i	M	$i/M \leq 100\%$
Varchar	M	M	i	i 字节+(1-2)字节	$i/(i+1-2) < 100\%$

3.那到底是选择char还是varchar呢?

现在磁盘容量都非常的大,如果存储的字符不是很多,比如20个字符以内,其实都用char,速度会更快
定长,短列,适合.

而且,如果一张表所有列的字节都是定长的,这种表会被识别为 fixed 类型,速度很快.

实际开发中,存储的字符较少,都是用的定长,再浪费也浪费不到哪去

4.char没有存储指定的长度,也能占据指定的长度,是如何做到的呢?

char 不够指定长度时用"\0"(空格)来填充,取出时,会把右侧的空格全部抹掉

注:

这意味着,如果右侧有本身有空格,将会丢失

实验测试

```
create table t6 (
n1 char(10),
n2 varchar(10)
);
```

```
insert into t6 values (' hello ',' hello ');
select * from t6;
```

```
select concat('!',n1,'!'),concat('!',n2,'!') from t6;
```

```
! concat('!',n1,'!') ! concat('!',n2,'!') !
! ! hello! ! ! hello ! !
```

所有的空格都一样,mysql无法区分这是你本身的空格,所以char在取出时,将右侧的空格给删除掉了
而varchar型则不会删除后面本身的空格,因为在它的开始处有1-2个字节已经说明了这个列占几个字符

速度上:定长速度更快一些

注意:

char(M),varchar(M),限制的是字符,不是字节

即 char(2) charset utf8,能存2个utf8字符,比如'中国'

```
set names gbk;

create table t7 (
name char(10)
)charset=utf8;
```

```
insert into t7 values ('中华人民共和国万万岁');
```

char与varchar型的选择原因:

1.空间利用效率,四字成语表,char(4),利用率100%

个人简历,微博 140字 ,varchar(140)

2.速度

用户名:char

char/varchar 最大可存多少个字符:

手册中有说明

CHAR列的长度固定为创建表时声明的长度。长度可以为从0到255的任何值。

VARCHAR列中的值为可变长字符串。长度可以指定为0到65,535之间的值。(VARCHAR的最大有效长度由最大行大小和使用的字符集确定。整体最大长度是65,532字节)。

即 char不论字符集,在[0,255]之间,

varchar在[0,max]之间, 而max受字符集影响,总的原则是字符所占的字节不超65535(实际上还不到65535)

<http://www.zixue.it/thread-14292-1-1.html>

5.text 文本类型

TEXT[(M)]

可以存比较大的文本段,搜索速度稍慢

因此,如果不是特别大的内容,建议用char,varchar来代替

最大长度为65,535($2^{16}-1$)字符的TEXT列。

```
create table t8 (  
te text);  
  
insert into t8 values ('asdf asd fsdwerdsf ');  
  
select * from t8;
```

6.blob 二进制类型 BLOB[(M)]

blog和text是对应存在的

其实,如果不是存图像,blob基本用不上

blob是用来存储图像,音频等二进制信息

意义:2进制,0-255都有可能出现

用blob存储的好处:

我们在用字符存储的时候,知道有utf8字符集,gbk字符集

比如:

有可能在gbk字符集下,二进制中有个253,有可能是违法的,在入库的时候,被过滤了你的这一串信息,有可能被过滤掉一小部分,就丢失了

而blog用二进制来存储,就不用考虑字符集了,就是0,1

使用上和text一样,只不错在存储的时候,从不考虑字符集

所以我们存储什么都可以

如:

一张图片中有0xFF字节,这个在ascii字符集人为是非法的,在入库的时候,被过滤了

7.enum 枚举型

ENUM('value1','value2',...)

是定义好,值,就在某几个枚举范围内

gender('boy','girl'),insert 时,只能选"boy","girl"

```
create table t10 (
```

```
gender enum ('boy','girl')
);

insert into t10 values ('girl');
insert into t10 values ('xxx');
insert into t10 values ('boy');
```

8.set 集合

SET('value1','value2',...)

一个设置。字符串对象可以有零个或多个值，每个值必须来自列值'value1'，'value2'，...SET列最多可以有64个成员
(自己在课后演示一下set)

第 33 课 日期时间列

日期时间类型

Year 年(1字节) 95/1995

范围[1901-2155]

在insert是,可以简写面的后2位,但是不推荐这样

如果我们只写两位,计算年份按:

[00-69]+2000

[70-99]+1900

即: 填2位,表示1970-2069

Date 日期 1998-12-31 [年月日]

范围:1000/01/01 ~ 9999/12/31

time 时间 13:56:23 [时分秒]

范围: -838:59:59 -> 838:59:59

datetime 日期时间 1998-12-31 13:56:23 [年月日 时分秒]

范围: 1000/01/01 00:00:00 -> 9999:12:31 23:59:59

时间戳:

是1970-01-01 00:00:00 到当前的秒数

一般存储注册时间,商品发布时间等,并不是用datetime存储,而是用时间戳

因为datetime虽然直观,但计算不便

```
create table t12 (
ye year,
dt date,
tm time,
dttm datetime
);
```

1.测试 year

```
insert into t12 (ye) values (1901);
insert into t12 (ye) values (07);
insert into t12 (ye) values (77);
```

如果我们只写两位 values (07),(77)

按如下方法计算年份:

[00-69]+2000

[70-99]+1900

不建议只写两位

2.测试 date

```
insert into t12 (dt) values ('1990-12-23');
```

3.测试 time

```
insert into t12 (tm) values ('12:23:59');
```

4.测试 datetime

```
insert into t12 (dtm) values ('1992-12-31 12:23:59');
```

5.timestamp 时间戳

如果我们插入改行,时间戳这列会自动插入当前时间戳

当我们更改这行数据时,timestamp列会自动更改时间为更改数据时的时间戳

```
create table t13 (  
id int,  
tt timestamp  
);  
  
insert into t13 (id) values (1);  
  
select * from t13;  
  
update t13 set id=2 where id=1;  
  
select * from t13;
```

tt列显示当前的时间

TIMESTAMP列的显示格式与DATETIME列相同

显示宽度固定在19字符, 并且格式为YYYY-MM-DD HH:MM:SS

不建议这样使用,如果需要插入精确的时间

用 int unsigned 来存储它的精确正整数时间戳即可

第 34 课 列的默认值

1. 某些列不插入内容,值是多少?

NULL

2. not null 又是干吗的?

1: NULL 查询不便

我们查询的时候需要用 is null 或者is not null

2: NULL 的索引效率不高,会影响我们的查询速度

所以,实用中,避免列的值为 NULL

3. 如何避免列的值不为null呢,这个列我没有插入具体的值

插入默认值

4. 如何设置默认值?

default

总结--声明列 NOT NULL default 默认值

```
create table t14 (  
id int not null default 0,  
name char(10) not null default ''  
);
```

```
insert into t14 values (1,'listi');
insert into t14 (id) values (2);

select * from t14;
select * from t14 where name='';
```

如果这个列没有插入数据,有个默认值可以填充,避免它为null

第 35 课 主键与自增

primary key, auto_increment

1.什么是主键?

primary key, 能够区分每一行的列

以会员为例,我们为了区分他们,往往给每个会员一个独一无二的会员号,这个会员号就是 主键

2.主键肯定不能重复, 那不重复的是不是一定就是主键呢?

不重复的列未必是主键(email 手机号 列一般不重复)

一半而言,一张表需要一个主键,但是如果你偏不声明,也是可以的

3.有两种声明主键的方式

```
create table t15 (
  id int primary key,
  name char(5)
);
```

```
create table t16 (
  id int,
  name char(5),
  primary key(id)
);
```

主键不能重复,我们插入相同主键测试一下

```
insert into t15 values (3,'list');
insert into t15 values (3,'list');
```

插入失败,会告诉你:主键重复定义

4.主键往往和 auto_increment 一起出现

当并不意味着,他们两个必须要绑定在一起使用

一般情况下,绝大部分,我们的主键是数字,1 2 3 4...

所以我们才让它递增

但有的时候,我就用 email 做主键是不是也可以?当然可以

5.不声明主键,直接用auto_increment,看是否能建表成功

auto_increment 必须有索引,且一般是主键索引

普通列也可以自增,但是必须有索引

```
create table t17 (
  id int auto_increment,
  name char(5),
);
```

报错:

there can be only one auto column and it must be defined as a key

一张表,只能有一个自增列,且必须要有索引(index/key)

```
create table t18 (
id int auto_increment,
name char(5),
key id(id)
);
```

一半使用中,自增需要配合 主键索引 共同使用

```
create table t19 (
id int primary key auto_increment,
name char(5)
);
```

6 oracle 没有自增,比较浪费资源,使用的是sequence,序列

7.看 准备练习.sql 中的goods表的建表语句

我们可以看明白建表语句中的内容

那表外面的如下字段怎么理解?

表引擎和字符集,我们后面会专门讨论

engine=myisam default charset=utf8;

第 36 课 综合建表案例

建表案例某高端白领私密社交网站

主键	用户名	性别	体重(KG)	生日	工资	上次登陆	个人简介
id	Username	gender	weight	birth	salary	Lastlogin	intro

查看 36综合建表案例.doc

id 一般 int unsigned

name,varchar(10)

lastlogin datetime

intro varchar(1500)

改进:

name 这种短列,一般定长,即使有轻度浪费

lastlogin 时间戳

intro ,不能再定长,但发动频率低,另存一张表.

定长与变长分离.

常用与不常用分离

第 37 课 列的删除增加与修改

1 表创建完毕后,能否添加 1 个列? 删除 1 个列? 修改一个列?

新增一个列,或者删除修改一个列,这属于 DDL 操作,数据库定义语言

区分数据的增删改,插入数据是指表中的数据,不会影响到表的结构

alter table 表名 add 列名 列类型 列属性... (新列 默认在表的最后)

alter table 表名 add 列名 列类型 列属性... after 列名 (新列出现指定列后)

alter table 表名 add 列名 列类型 列属性... first (新列为第 1 列)

alter table 表名 change 旧列名 新列名 新类型 列属性....

alter table 表名 modify 列名 新属性....

alter table 表名 drop [column] 列名

用到36课综合建表案例的sql语句,使用学生的一份建表语句

注意:

建表时的列名称一律小写

因为大小写在linux和windows下,有时候敏感有时候不敏感,所以在建表时一律小写

2.添加一个身高列,默认新增列追加在表的最后

```
alter table user1 add height tinyint unsigned not null default 0  
desc user1;
```

3.我想身高列放在体重列后面

删除身高列 column 列

```
alter table user1 drop column height;
```

在增加身高列,放在weight列的后面

```
alter table user1 add height tinyint not null default 0 after weight;  
  
desc user1;
```

4.现在人的身高越来越高,255的tinyint已经不够存了

我们需要改变列类型,改成 smallint

change 可以将表明一起修改了

```
alter table user1 change height shengao smallint not null default 0;
```

modify 也可以修改列

modify 跟 change 有什么不同呢?

modify 不可以修改列名,当我们不需要修改列名的时候,就用modify

```
alter table user1 modify shengao tinyint not null default 0;
```

5.查看表的结构

desc 表名

```
desc user1;
```

第 38 课 视图

视图 -> 叫 view

1.视图的概念

2.视图是干什么的

3.建视图

一张表多行多列,我不想看多行多列

我们将三行三列取出来 -> 形成一个结果集

拿着这个结果集,我们可以再次查询,结果集可以当作一张表,那么这张表就可以再次查询

如果某个结果集,我经常要对它进行查询,挺麻烦的,每次要写

from (select ...)

那么对于这个常用的结果集,我能不能把它存起来呢?

```
select goods_id,goods_name,(market_price-shop_price) as sheng from goods;
```

将上面的sql得到的结果集,存下来,以便我们下次从这个结果集中查询我们想要的数

据如何将这个结果集存储下来呢?

建视图

```
create view sgoods as select goods_id,goods_name,(market_price-shop_price) as sheng from goods;

select * from sgoods;
```

这个sgoods表的内容,就是上面select的结果集

1.视图的概念

view 又被成为虚拟表,view是sql语句的查询结果

也可以说 它是表到结果的映射

一个sql的查询结果,我用 view 存起来

下次你查这个view,就相当于这个sql发挥作用了

2.view有什么好处呢?

要view有什么用

1):权限控制时可以用

比如,某几个列,允许用户查询,其他列不允许查询

可以通过视图,开放其中一列或几列,起到权限控制的作用

如,我们只能让用户查询到goods表的goods_id,goods_name,shop_price

```
create view s2goods as select goods_id, goods_name,shop_price from goods;

select * from s2goods;
```

任你怎么查询s2goods表也无法得到market_price

2):简化复杂的查询

select 嵌套 select 查询

select * from (select....)

我们可以先将内层select语句的查询结果转为视图

在通过视图查询我们具体想要的内容

例:查询每个栏目下商品的平均价格,并按照平均价格排序,查出前三高的商品的栏目

1)我们先查出每个栏目下商品的平均价格

```
select cat_id,avg(shop_price) as pj from goods group by cat_id;
```

2)在从上面sql 得到的结果集中 desc 排序,取前三行即可

```
select * from (select cat_id,avg(shop_price) as pj from goods group by cat_id) as tmp order by pj desc limit 0,3;
```

3)通过view 我们可以简化上面的sql

```
create view pingjun as select cat_id,avg(shop_price) as pj from goods group by cat_id;

select * from pingjun order by pj desc limit 0,3;
```

view是什么 -> view就是你储存好的一条sql的查询结果

```
select * from pingjun order by pj desc limit 0,3;
```

这条sql的查询的pingjun这个表,我们能看出来pingjun是个表还是个视图吗?

不能,对于人的操作而言,视图看起来就是一张透明的表

我们

```
show tables;
```


可以看到我们的视图,也是存在的,我们都区分不出来它是表还是视图;

3.那视图可以更改,删除吗?

我们测试一下,新建一个vuser视图

```
create view vuser as select uid,name from user;  
  
select * from vuser;
```

1)将user表中的数据改一下

```
update user set name='Jane' where uid=4;
```

查询vuser视图表中的数据,看数据是否有变化

```
select * from vuser;
```

我们可以看出,vuser的内容随着user表的更改而变化了
因为视图,就是物理表经过某个运算(select)得到的一个投影
真实的表都变化了,这个投影肯定也会随之改变

2)我们修改vuser 视图表中的数据 会发生什么?

```
update vuser set name='who' where uid=4;
```

我们看到,user表和vuser表的数据都改成功了

3)我们在上面有个pingjun视图表

```
select * from pingjun;
```

那pingjun表中的pj列,能修改成功吗?

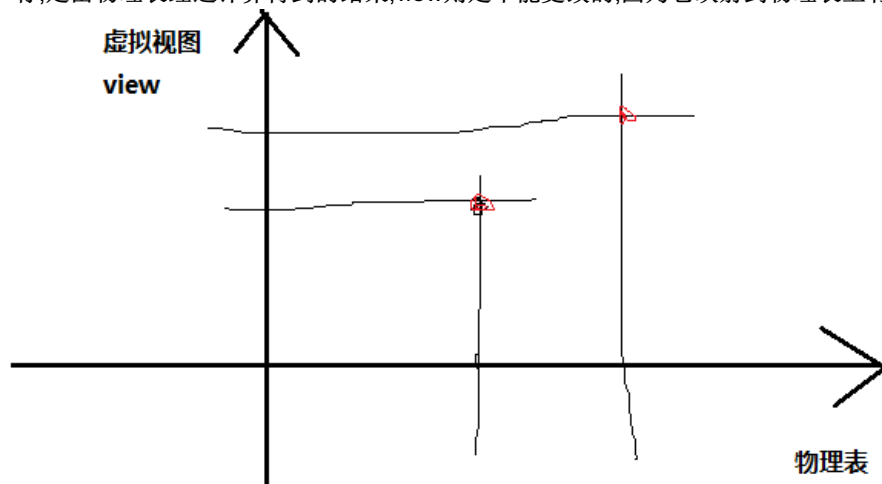
```
update pingjun set pj=80 where cat_id=15;
```

报错: pingjun表的pj字段不能被更改

我们将一个栏目下平均价格给改了,那这个栏目下那么多商品,这些商品的价格又该是多少呢?

如果视图的每一行,是与物理表一一对应的,它俩有且只有一个相互对应的字段,就像数学上的可逆函数,如果是这样就可以相互影响修改;

如果view某一行,是由物理表经过计算得到的结果,view则是不能更改的,因为它映射到物理表上有多行,不知道更改哪里



第 39 课 表/视图管理语句

1.查看所有表 show tables;

视图也会显示出来

2.查看表结构 desc 表名/视图名

3.查看建表过程 show create table 表名

4.查看建视图过程 show create view 视图名

5.删除表: drop table 表名

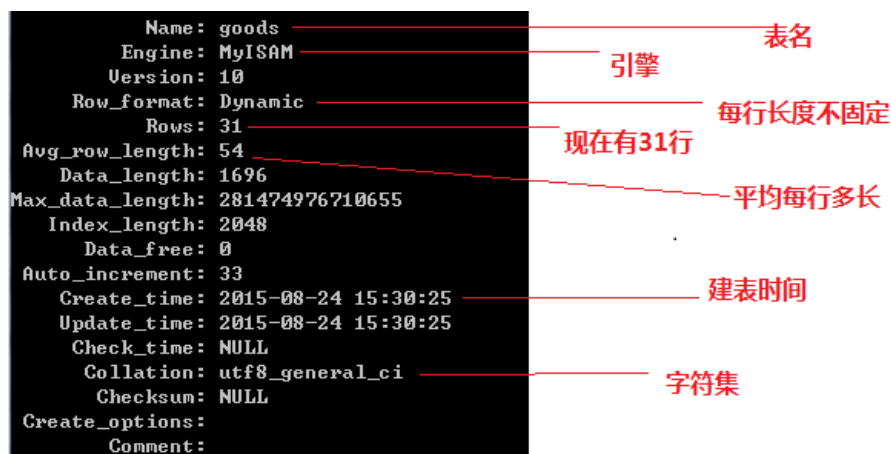
6.删除视图: drop view 视图名

7.查看库中的表信息 show table status;

如果表信息太多,看不清楚 可以加上\G 竖着显示

show table status \G

8.查看某张表详细信息 show table status where name='表名';



```

      Name: goods
      Engine: MyISAM
      Version: 10
      Row_format: Dynamic
      Rows: 31
      Avg_row_length: 54
      Data_length: 1696
      Max_data_length: 281474976710655
      Index_length: 2048
      Data_free: 0
      Auto_increment: 33
      Create_time: 2015-08-24 15:30:25
      Update_time: 2015-08-24 15:30:25
      Check_time: NULL
      Collation: utf8_general_ci
      Checksum: NULL
      Create_options:
      Comment:

```

Annotations (in red):

- 表名 (Table Name): points to Name: goods
- 引擎 (Engine): points to Engine: MyISAM
- 每行长度不固定 (Variable row length): points to Row_format: Dynamic
- 现在有31行 (Now 31 rows): points to Rows: 31
- 平均每行多长 (Average row length): points to Avg_row_length: 54
- 建表时间 (Table creation time): points to Create_time: 2015-08-24 15:30:25
- 字符集 (Character set): points to Collation: utf8_general_ci

通过查看表的详细信息,我们也能区分是否为视图

如果是视图: Comment: VIEW

如果不是视图: Comment

9.改表名 rename table oldName to newName

10.清空表数据 truncate (相当于删除表,再重建)

delete from 表名

truncate 表名

这两个的区别

1) delete from 表名

```
create table qing (
  id int primary key auto_increment
);
```

给qing表加入两行数据,删除一行,再添加一行,查看id的值

```
insert into qing values (null),(null);

delete from qing where id=2;

insert into qing values (null);
```

我们发现,id为自增长,即使删除一条数据后,在插入一条,id的值也不会重拍,依旧继续增长

将表的内容全部删除,并再次插入一条数据,观察id的变化

```
delete from qing;

insert into qing values (null);
```

看到id并识别从0开始,而是接着上次的id继续增长

2)我们删除qing这张变,再重新新建这张变,观察id是从0开始的还是继续上次的id增长,肯定是从0开始的

```
drop table qing;

create table qing (
id int primary key auto_increment
);

insert into qing values (null),(null);
```

3)truncate 表名

清空qing表的数据

```
truncate qing;

insert into qing values (null);
insert into qing values (null);
```

我们可以看到id的值,从0开始增长了

truncate 表,就是删除这张表,再重新新建一张一样的表,那表的自增长的信息就被初始化了

建表语句的括号外面一般这样写:

engine=myisam default charset=utf8

一个代表引擎,一个代表字符集

第 40 课 存储引擎的概念

什么事存储引擎?

数据库对同样的数据,有着不同的存储方式和管理方式

在mysql中,称为存储引擎

同样一段信息,你用什么方式来保存的?

记忆力好的可以保存在大脑中 -> 可以随时select

记忆力不好的可以抄在本子上

打开 mysql 数据库对应的目录

\\bin\\mysql\\mysql5.6.12\\data\\test\\

观看 table 与文件对应关系

以商品表为利息

建表的时候,每一列的属性,表结构,都存储在这个文件中 -> goods.frm

表中一行一行的数据,放在这个文件中 -> goods.MYD

索引文件 -> goods.MYI

我们可以看到,我们之前联系的t系列的表,只有.frm文件

没有.MYD和.MYI文件 **.ibd文件：单表表空间文件，每个表使用一个表空间文件（file per table），存放用户数据库表数据和索引。**

我们仔细观察: mysql\\mysql5.6.12\\data\\ibdata1 这个文件

ibdata1	2015/8/27 16:04	文件	77,824 KB
---------	-----------------	----	-----------

这个文件非常大,我们再

ibdata1、ibdata2等：系统表空间文件，存储InnoDB系统信息和用户数据库表数据和索引，所有表共用。

t10表中插入一行数据,在观察这个 ibdata1 文件

```
insert into t10 values ('girl');
```

我们可以看出,这个文件在我们插入数据后,变大了,而且最近修改时间也改变了
我们不难猜出,之前我们做联系所新建的t系列的表的数据,都存放在这个 ibdata1 文件中
都是表,为什么存储的位置和方式是不同的,如此差别对待
因为表的存储引擎不同,那数据存放的位置就不同了

我们通过查看表的详细信息,来看一下t系列表和goods表的存储引擎有何不同

```
show table status where name='t14' \G  
show table status where name='goods' \G
```

t14 => Engine: InnoDB

goods => Engine: MyISAM

引擎不同,它组织你数据的方式不同

mysql5以后,将 innodb 设置为默认引擎

我们之前在建t系列表的时候,直接结束,没有声明引擎,那mysql默认我们的引擎为 innodb

既然后不同的引擎,那不同的引擎之间肯定是有区别的

常用的引擎有三种 innodb,myisam,memory

memory建立起来的表,数据,全部存在内存里,不存放在磁盘上,它的速度是非常快的

但是也有一个问题,如果服务器一旦关机,内容就都没有了

所以一半,不需要持久保存的,可以用memory引擎

面试中最容易问到的就是,myisam和innodb的区别

设有张马虎,李小心两人,都是地铁口的自行车管理员.

每天都有很多人来存取自行车

张马虎的管理方式是:来存自己存,不记录存的是什么车,取时交5毛,也不检查取的是不是自己的车.

李小心呢,则在存取自己车时,记录存车人的特征与自行车的特征,当人来取钱车,还要小心核一下,人与车的特征是否对应

思考:

张马虎和李小心谁对业务的处理速度更高? 张马虎 -> myisam

二者谁对自行车管理更安全? 李小心 -> innodb

张马虎,myisam,数据库崩了,数据丢失就丢失了,想找回来的难度比较大

李小心,innodb,数据库崩了,但是它有丰富的日志,每一行操作都有记录,数据比较容易找回来

说到数据库优化,这是个非常大的题目,我们学习两天都无法入门这个数据库优化,

所以此处,大家记住两者的区别就可以了

myisam引擎有个优点,我们给别人建完数据库

想把数据给他,可以直接拷贝

data下的文件即可 test目录就是test库的所有数据

\\bin\\mysql\\mysql5.6.12\\data\\test

而innodb是不可以的,它是多张表,多个库混在一起写的,直接拷贝那个文件不好使的

特点	Myisam	InnoDB	BDB	Memory	Archive
批量插入的速度	高	低	高	高	非常高
事务安全		支持	支持		
全文索引	支持				
锁机制	表锁	行锁	页锁	表锁	行锁
存储限制	没有	64TB	没有	有	没有
B树索引	支持	支持	支持	支持	
哈希索引		支持		支持	
集群索引		支持			
数据缓存		支持		支持	
索引缓存	支持	支持		支持	
数据可压缩	支持				支持
空间使用	低	高	低	N/A	非常低
内存使用	低	高	低	中等	低
支持外键		支持			

第 41 课 字符集与乱码问题

牵涉到:

字符集,乱码

几乎80%的学员,在开发中,在学习,会遇到过乱码的困扰
我们现在就来学习,怎么样以后才能不乱码,不受乱码的干扰;

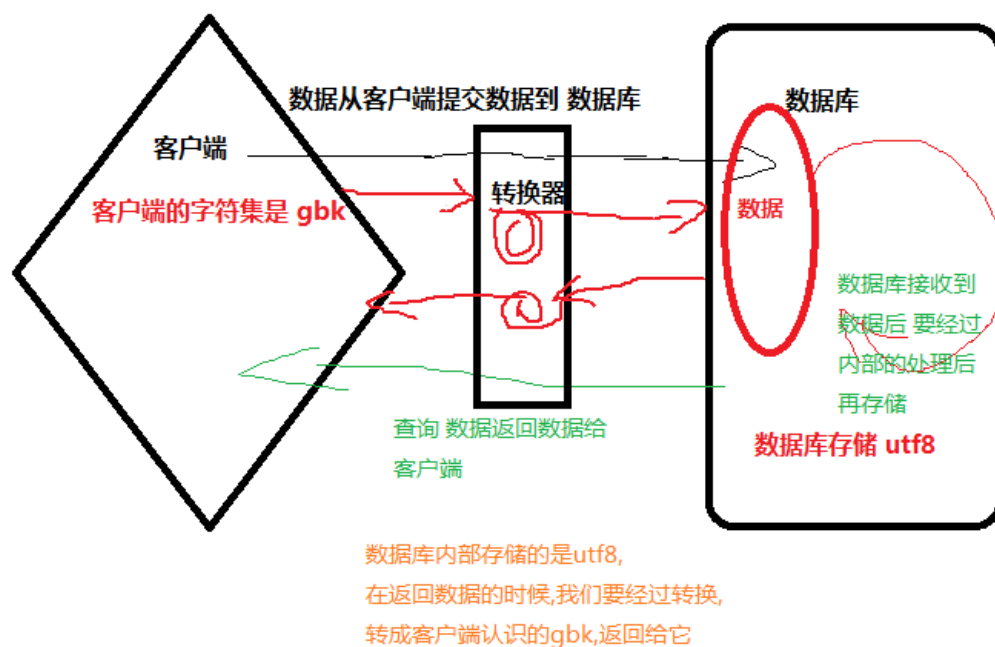
乱码:对计算机而言,没有这个乱码,只是对人而言,人看不懂

乱码只是 文字本来的字符集 和 展示的字符集 不一致

想要不乱码很简单: 输入的字符集和展示的字符集一致,自然就不会乱码;

选字符集 -> 就选 utf8 即可,这是大趋势

既然是utf8;为什么在mysql中,有的会 set names gbk;



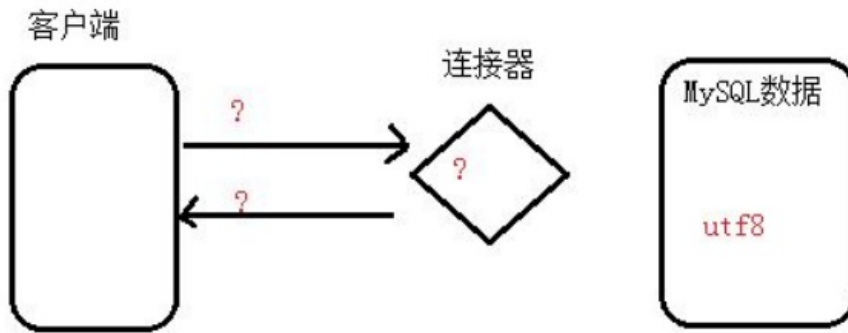
世界银行存储就只认黄金,不论我们带着哪国的钱过去了,都给你折换成金条存起来

下次你去取钱,你说,我要人民币,那么银行就会把你的黄金折换成人民币还给你

这个银行就是连接器

连接器既可以使用utf8,也可以使用gbk;

无论连接器是什么字符集,存储在数据库中的都是utf8



- 1: 服务器最终存utf8
- 2: 客户端传来的什么字符集? `character_set_client`
- 3: 连接器用什么字符集? `character_set_connection`
- 4: `select`查询返回什么字符集? `character_set_results`

`show variables like '%charact%';`

现在这三者保持一致,gbk,不过这三个字符集可以人为更改的

1.将客户端字符集改为 utf8;

但是我们的黑窗口是gbk编码的

插入内容,我们会发现乱码

```
create table test3 (  
name char(10) not null default ''  
)charset=utf8;
```

```
set character_set_client = utf8;  
  
insert into test3 values ('中国人民解放军');  
  
select * from test3;  
  
set character_set_client = gbk;
```

2.将数据库返回的数据 改为 utf8编码的

客户端是gbk编码的,返回utf8编码的数据,客户端肯定不认识
所以还是会乱码

```
set character_set_results = utf8;  
  
insert into test3 values ('中国人民解放军');  
  
select * from test3;  
set character_set_results = gbk;
```

3.中间的连接器,使用utf8和gbk都是可以的

`character_set_connection`

因为连接器始终需要 gbk=>utf8 因为始终都需要存储成utf8

注意:连接器不能使用较小的字符集,如ascii

在转换的时候会丢失一部分编码

而丢失的这部分编码是不可逆推回来的.

4.如果 3 者都一致,可以用简写

`set names gbk/utf8;`

5.我们知道,谁能连接数据库,谁就可以称为客户端
有时候我们用 .php文件 去连接数据库
我们的php就是客户端,那我们的php页面,编码统一使用的utf8;
我们只需要在 .php 文件里,
set names utf8; 即可保证不乱码
而有的学员是 在开发 php 的时 set names utf8;
在 命令行 黑窗口中还是 set names utf8;
这不就乱码了么,两个不同的客户端,各使各的参数就可以了

6.杜绝乱码的,检查的几个步骤

- 1)php 文件是 utf8
 - 2)html 页面的 meta 信息 也是utf8
 - 3)建表时,也是utf8
- ```
create table () charset=utf8;
```
- 4)php页面连接数据库时
- ```
set names utf8;
```

检查这四步,之后,如果仍然乱码,则继续检查这四步

第 42 课 校对集

charset是字符集,那校对集是什么呢?
校对集并不常用,我们知道它的概念即可

```
create table t21 (  
n char(2)  
);
```

插入数据

```
insert into t21 values ('a'),('B'),('c'),('D');
```

desc 查询 t21,猜测查询结果

```
select * from t21 order by n desc;
```

1)第一种排序: caDB

原因:

ascii字符集里,小写的a是96,大写的a是65

2)第二种排序: DcBa

a,b,c,d..按字母增长,到z最大,复合人的生活习惯

这其实就是标准的不统一

它牵扯到,字符排序时,谁拍前谁在后,那到底按谁说的算?

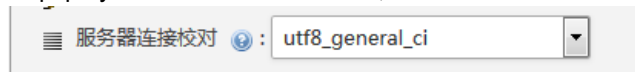
字符集排序到底以谁的标准拍? --> 校对集

以ascii字符集为例,ascii字符集只有一种,那它对应的校对集是否也只有一种呢?

我们刚才的两种排序都没有错,这就牵扯到看你选什么校对集来排序了

校对集就是字符集的排序规则,一套字符集可以有n套校对集

看 phpmyadmin的 服务器连接校对,里面可以选择服务器的校对集



以gbk为例

1)gbk_bin

就是按照二进制排序

2)gbk_chinese_ci

ci 指的是 common 普通的意思,就是指普通的排序,按照人的习惯来排序

存性别的时候,用tinyint 比 char(1) 快,因为char在查询的时候,往往要考虑校对集的问题,而tinyint 一个数字 完全不必考虑这个

第 43 课 索引概念

1.什么是索引?

index 中文有索引的意思

我们看的英文书,在目录部分,就写有index

数据库帮我们存储数据,存储在文件里,我们之前也看到了那个文件

MYD文件帮我存数据,当我们取数据的时候,我们希望取得比较快

那如何能让数据取的比较快呢?

当数据越来越多,MYD文件则越来越大

假设有 10000 行的数据

select .. where id=10000

那它应该迅速就来到文件的第10000行,取出这个数据

那它如何知道,这个第10000行,是在这个MYD文件中的第一个字节开始取呢?就是说,沿着这个文件,我们走多少字节,才能到第10000行呢

怎样才能快速帮我们查到呢,这个时候就需要靠索引了

select ... where id=10000

我们来找第10000行的数据

假设这似乎第10000行的数据



就像我们数的目录一样,我们需要找哪篇文章,看一下目录,是在第几页,直接就能很快的找到文章的所在

MYI 是索引文件

索引文件是干什么的?

它里面也是存储的数据,只不过它这个数据,是通过某种数据结构

帮你高效索引起来的

比如说它存储一个树形结构的,找起数据来就非常快速

就像电视上的猜价格节目一样

竞猜一个商品的价格

演员: 500

主持人: 低了

演员: 600

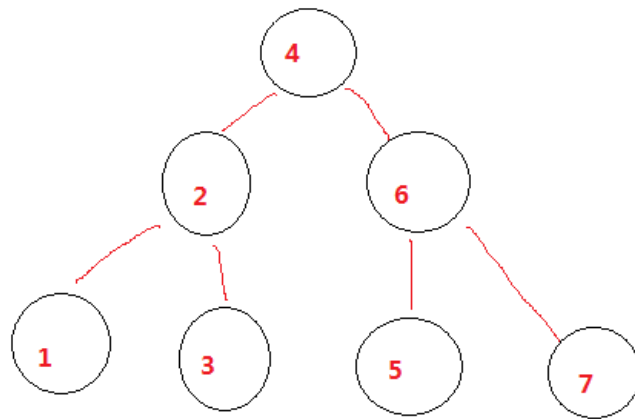
主持人: 高了

演员: 550

主持人: ..

演员: ..

MYI 索引文件 树状型



走几步能找到7?

索引如何查到7?

先找到4

7>4,右拐找到6

7>6,右拐找到7

3次就能查到

不用索引来查询,40亿的数据,运气不好,需要40亿此才能查到

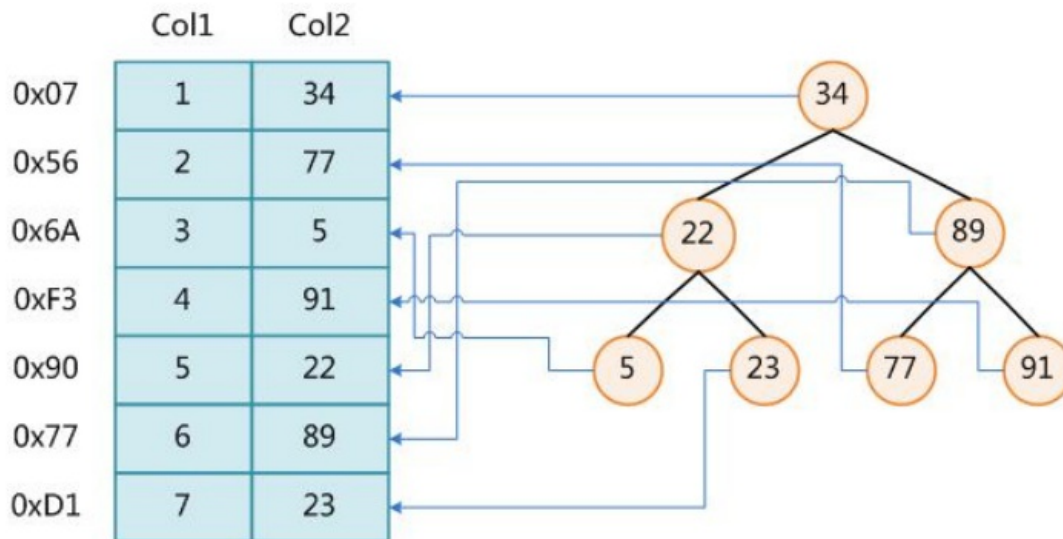
用索引,当数据40亿次的时候,它顶多查32次就能查到了

索引是高效组织的数据结构

我们查到 7 还不够,在7这里,还有一小行的数据,比如说是1281,这个代表了偏移字节

就是沿着MYD文件,便宜1281个字节,就能找到第7行的数据

这个就是索引的作用,它就是一个书的目录



索引是数据的目录,能快速定位行数据的位置

如果我们增加一行数据,删除一行数据,或者更改一行数据,

是不是索引也需要相应的操作呢?

因此:

索引提高了查询速度,降低了增删改的速度,并非加的越多越好

在实际中,有可能一张表的索引文件,比这张表的数据文件还要大 => 目录比书还厚

一般在查询频率的列上加,而且在重复度低列上加效果更好

2.索引的类型

mysql中 有以下几种索引

1)key, 普通索引

纯粹就是帮我们组织数据,提高查询速度

2)unique key, 唯一索引

起到2个作用

1是加快你的查询速度

2是帮你约束数据,怎么个约束法?

这个数据不可能重复了,unique 中文 唯一的意思

我们看一下key和unique key的使用

```
create table t22 (  
  name char(10),  
  email char(20),  
  key name(name),  
  unique key email(email)  
);
```

key name(name) 的意思是,针对name的列,创建索引,索引的名称也叫name

我们可以在所有的列声明完毕之后,再加索引

查看t22的表结构

```
desc t22;
```

给t22表插入两条相同的数据

```
insert into t22 values ('lisi','lisi@qq.com');  
insert into t22 values ('lisi','lisi@qq.com');
```

当我们插入第二条数据的时候,会报错

这个时候,它在维护索引的时候,会检查,这个email已经存在了,就不允许我们插入这个相同email这列值,这个是 unique key 的一个特点
比如我们在创建一个网站的时候,用户名,email这两个字段是不许重复的,我们可以用unique key ,既能提高查询速度,又能起到约束的作用

3)primary key , 主键索引

一张表中,只可能有一个 primary key

我们之前已经使用过了

```
create table t23 (  
  id int unsigned auto_increment primary key  
);
```

4)fulltext 全文索引

注意:

中文环境下,全文索引无效

为什么呢?

因为英文,每个单词是分开的,它将每个单词分为别类,建立索引

你查这行中的任意一个单词,都有可能查询到这行数据

而中文的每个字是连在一起的,它无法区分每个字

所以就导致,在中文环境下,全文索引是无效的

在中文环境下,需要 分词+索引才有效

一般用第三方解决方案,如sphinx

5)设置索引长度

设置索引长度是为了优化索引

建索引时,可以只定索引列的前一部分的内容

比如前10个字符

例:

email列 后面都是 @qq.com,@163.com, 等

@后面的内容区分度不高,都是类似的内容

那我们可不可以

将 @ 前的10个字符内容给截取下来建索引 -> 可以

```
create table t24 (  
  name char(10),  
  email char(20),  
  key name(name),  
  unique key email(email(10))  
);
```

注意:

这个并不是限制的email长度,只能填10个字符

而是索引只取10个字符

插入大于 10个字符 长度的email

```
insert into t24 values ('lisi','1234567892@qq.com');
```

6)多列索引

有的时候,一列索引还查询不出来什么效果

多列索引,就是把2列或多列的值,看成一个整体,然后建索引

例:

我们注册国外网站的会员时,会让我们分别注册

firstname,lastname 姓和名

分开存姓名,可以方便以后查询不同姓氏的注册用户

但大部分使用还是将姓名连接在一起查询的

```
create table t25 (  
  xing char(10),  
  ming char(10),  
  key xm(xing,ming)  
);  
  
insert into t25 values ('zhang','san');  
  
desc t25;
```

查看表的详细索引信息

```
show index from t25;
```

注意:

以下两个select 多列索引可以发挥作用

```
select * from t25 where xing='zhang' and ming='san';  
  
select * from t25 where xing='zhang';
```

下面这个select 索引是不能发挥作用的

```
select * from t25 where ming='san';
```

我们可以同 explain sql语句
来显示mysql如何使用索引来处理select语句以及连接表

```
explain select * from t25 where xing='zhang' and ming='san' \G

explain select * from t25 where xing='zhang' \G
```

possible_keys: xm 有可能使用到的key 是 xm
key: xm

```
explain select * from t25 where ming='san' \G
```

possible_keys: NULL 有可能使用到的key 是 null
key: xm

既然索引没有发挥作用,为什么key还是xm呢?

例:

一片文章,有标题,有内容

标题:<<静夜思>>为索引 -> 对应的内容 在第9页

1是我们看到内容在第9页,就去第9也看内容了

2是我们找到index索引标题,而不去看它的内容

从索引直接返回数据,不用走MYD文件去查询

下面这个语句可以使用上索引
因为mysql没那么傻,它会分析语义

```
explain select * from t25 where ming='san' and xing='zhang' \G
```

怎样区分它是否能使用上索引呢?

左前缀

一个数据,从左到右 abcdefg..

我们给书建目录,目录是有顺序的

我们前半部分能记得 abcd..后面记不得了,它可以帮我们定位大致的范围

那如果前面记不住,后面记得 ...efg ,那如何定位

前面的数据千变万化 有可能是zzzzefg,或这aaaaefg 等,根据右半部分无法定位

这个就是建使用的左前缀规则,从左往右侧查,索引还能部分发挥作用,左侧是确定的,相对好查询

右侧固定,左侧不知道,那则是千变万化,无法使用索引

十八哥详解:

<http://www.zixue.it/thread-11966-1-1.html>

7)冗余索引

就是在某个列上,可能存在多个索引

比如 xm(xing,ming) 我们单查ming,索引不发挥作用

ming(ming) 给ming也加上索引,单独查ming的时候,也是有索引发挥作用的,那如果 xing和ming都查,对于ming则会后两个索引发挥作用
在开发中,为了提高查询速度,我们也经常会用到冗余索引

```
create table t26 (
xing char(10),
ming char(10),
key xm(xing,ming),
key ming(ming)
);
```

```
show index from t26 \G
```

```
explain select * from t26 where ming='san' \G
```

possible_keys: ming

key: ming

第 44 课 索引操作

1. 查看一张表的索引

show index from 表名

show create table 表名 查看建表语句,后面也会显示索引

2. 删除索引

删除 alter table 表名 drop index 索引名

或 drop index 索引名 on 表名

3. 添加索引

alter table 表名 add index/unique xm(xing,ming);

4. 添加主键索引,不需要起名字,因为是唯一的

alter tables 表名 add primary key(id[主键名称]);

5. 删除主键索引

alter table 表名 drop primary key;

第 45 课 常用函数

mysql中的列可以当成变量来看,既然是变量,就可以参与运算(+*/)

```
select 3+2;  
select 3*2;
```

既然能参与运算,我们在php中可以用函数来处理某一个变量

在mysql中也有很多函数

打开常用函数.txt

这里面罗列着一些mysql常用函数

要求:

这些函数要统揽一边,准确的参数可以不必记得,但是要知道有这么一些东西,知道用的时候到哪去找

这几类分别举个例子,讲解几个函数

1) floor(x) 返回小于x的最大整数值

goods表的shop_price取出整数

```
select goods_id,goods_name,floor(shop_price) from goods;
```

2) rand() 返回 0 到 1 内的随机值,可以通过提供一个参数(种子)使rand()随机数生成器生成一个指定的值

生成随机5-10的值

```
select rand();  
select floor(5+rand()*5);
```

3) left(str,x) 返回字符串str中最左边的x个字符

```
create table t27 (  
email char(20)  
);
```

插入 email 数据

```
insert into t27 values ('abc@163.com'),('123456@qq.com');
```

我们要分别统计各邮箱种类的使用率
需要将email字段截取成两部分

```
select left(email,3) from t27;
```

我们需要获取@的位置,将@前面的截取出来
position(substr,str) 返回子串substr在字符串str中第一次出现的位置

```
select position('@' in email) from t27;  
  
select left(email,position('@' in email)-1) from t27;
```

4)now() 返回当前的日期和时间

```
select now();
```

date_format(date,fmt) 依照指定的fmt格式格式化日期date值
注意跟要加百分号%

```
select date_format(now(), '%Y/%m');
```

注意:不要在条件里用函数,该列将无法使用索引

第 46 课 事务

1.了解事务的概念

比如:

银行转帐

张三 -> 转账给李四500元

张三的钱-500 李四的钱+500 两个update操作 这次事务才算完成

那:

张三的钱刚-500,打雷闪电机房断电,李四的钱还没加上

最终这500快哪里去了???

日常生活中,汇款二字包含两个小动作,1扣张三的钱,2加李四的钱

汇款成功,那扣钱和增加钱都需完成才算汇款成功

事物就是给你一种保证,什么样的保证呢?

在一个业务的下的具体多个小的语句,要么让你都完成,要么让你都不完成,从而保证你数据的一个安全性

否则,张三钱少了,李四没收到钱...俩人决裂了

2.事务的特性

隔离性,原子性,一致性,持久性

引擎要选 innodb,因为myisam不支持事务

```
create table t29 (  
  id int,  
  name char(10),  
  money int  
)engine=innodb default charset=utf8;
```

插入zhangsan和lisi的数据

```
insert into t29 values (1,'zhangsan',5000),(2,'lisi',5000);
```

zhangsan借给lisi500,给lisi的钱+500

```
update t29 set money=money+500 where id=2;
```

在另外一个窗口查看lisi的钱是否多了500

```
select * from t29;
```

这个事务还没完成,zhangsan的钱还没减500,lisi不应该看到自己多的500
之前就有这种骗术,一个人在柜台存钱,另一个人在ATM取钱
银行职员经过一系列操作,已经将钱转过去了
存钱的人在最后一步确认签字前,告诉银行职员,自己不存钱了
而取钱的人在收到钱的时候立刻就取出来
这个钱肯定无法扣除了,总不能为负数把,即使为负数,反正卡也不要了
在事务没有完成的中间状态,你不应该看到自己账户上多了钱

A窗口启用事务,来试一试,看看是什么样的效果

```
start transaction;
```

先查看连个人各有多少钱,再次给lisi+500

```
select * from t29;  
update t29 set money=money+500 where id=2;
```

B窗口查看lisi的钱

```
select * from t29;
```

A窗口应该给zhangsan-500

```
update t29 set money=money-500 where id=1;
```

B窗口查看t29

```
select * from t29;
```

事务完成,A窗口结束事务

```
commit;
```

看不到事务的中间状态,只能看到事务的开始前的状态,和结束后的状态,这个叫做事务的**隔离性**;

事物的原子性

commite之后,事务就结束了

如果再想使用事务,需要重新 start transaction;

zhangsan继续借给了lisi500

```
update t29 set money=money+500 where id=2;  
update t29 set money=money-500 where id=1;
```

结果在commit之前,想撤销借出的钱

需要用到:回滚 rollback;

```
rollback;
```

```
select * from t29;
```

B窗口的lisi,无法看到,zhangsan转了500半道又撤销了,这叫做**隔离性**
原子性体现在哪里呢?

```
start transaction;
update t29 set money=money+500 where id=2;
update t29 set money=money+300 where id=2;
update t29 set money=money-600 where id=1;

rollback;
```

我们开启事务之后,不论将数据改的多乱,只要rollback 回滚,数据就全部回到了事务之前,要是一提交,commit,就全部来到了事务之后,半道如果有一个语句没有执行成功,断电了,只要回滚,数据即可恢复到事务开启之前.这个体现了事务的**原子性**和**隔离性**

事务的一致性如何体现?

一致性指的是事务的之前和之后,它的业务逻辑上保持总体的一致

什么意思呢?

看一下代码演示

```
alter table t29 change money money int unsigned;
```

张三还剩下4000,如果他要借给lisi5000,能否成功呢?

```
select * from t29;

start transaction;
update t29 set money=money+5000 where id=2;
update t29 set money=money-5000 where id=1;
```

zhangsan减钱的时候,他的钱已经不够5000,money字段不能为负数,所以mysql报错了

但是mysql它有几种语句的模式,有时候检测不是很严格,它存在一个语句检测模式问题

我们如果用严格模式,它超出存储范围,不能运行

如果采用的sql_mod,不是很严格,这句话是能够执行的

只不过它会将zhangsan的减5000,给减去4000,返回0

那zhangsan-4000,lisi+5000,平白无故的多了1000 => 合作生财之道

这个问题处在,汇款事务的前后,账面的总的额度变了,事务的一致性,指的就是事务前后的数据应该有保持**一致性**

事务的持久性

指的是,一旦commit之后,是无法回滚 rollback 的

因为:一旦commit,这个事实就已经发生了,就不可能撤销回去了

如:

取钱时,有可能发生,卡里的钱已经扣了,但是人民币没出来

遇到这种情况,不要离开ATM机,然后打银行电话

查看你的交易记录,确实是扣除了钱

不过还有一条,冲证记录

银行将钱又冲证,还给你了

一个事务发生了,是无法撤回的,只能再来一条补偿性的事务

将原来产生的恶果给你补偿回去

一个事情发生了,这就是铁一般的事实,这个流水在这呢,你不能将历史事实给去除

比如你小时候偷过一个西瓜,过80年之后,你之前还是偷过西瓜,这是铁一般的历史事实,你只能再买一个西瓜还回去,但是小时候的偷瓜记录还是在的,这个就叫**持久性**