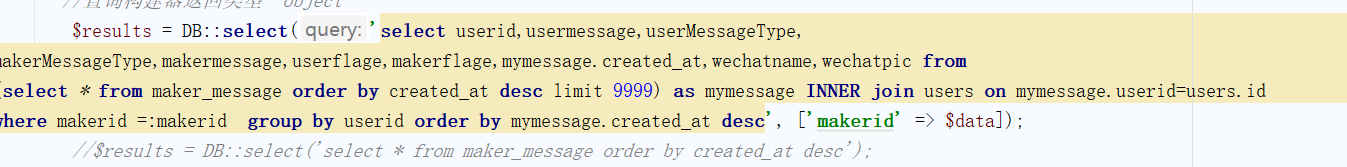
### mysql5.7子查询中order by 要加limit

1. 
2. Access denied for user 'root'@'localhost' (using password: YES)

My.conf添加 skip-grant-tables跳过密码认证

mysql修改账号密码

update mysql.user set authentication\_string='123456' where user='root';

flush privileges;更新数据库

## 简单操作

### 数据库连接

mysql –u root -p

### 选择数据库

use databaseName(数据库名);

**例如**

use score;

### 查看数据库列表

show databases;

### 查看表列表

show tables;

### 显示表列

Book表

show columns from book;

## 创建、删除,选择数据库

create database 数据库名；

create database book;

drop database 数据库名；

drop database book;

use 数据库名；

use book；

## 创建删除数据表

crate table 数据表名；

drop table 数据表名；

CREATE TABLE IF NOT EXISTS `runoob\_tbl`(

`runoob\_id` INT UNSIGNED AUTO\_INCREMENT,

`runoob\_title` VARCHAR(100) NOT NULL,

`runoob\_author` VARCHAR(40) NOT NULL,

`submission\_date` DATE,

PRIMARY KEY ( `runoob\_id` )

)ENGINE=InnoDB DEFAULT CHARSET=utf8;

## 插入数据

Insert into 数据表名 （字段名,..,字段名）values（字段值,..,字段值）;

Insert into book (‘name’,’price’) values(‘php’,’120’);

## 更新数据

update 数据表名 set 字段名=字段值…字段名=字段值 where ..

update book set name=”java”,writer=”caa” where price>120;

## 删除数据

DELETE FROM table\_name [WHERE Clause]

delete from book where id=1;

 delete a from Person a , Person b  where a.Email=b.Email and a.Id>b.Id;

### 多表删除

---------------------------------------------查询语句

## 单表查询与多表查询见sqlServer的word

## MySQL ALTER命令

当我们需要修改数据表名或者修改数据表字段时，就需要使用到MySQL ALTER命令。

### 删除，添加或修改表字段

Alter table 数据表名 drop 字段名；

Alter table 数据表名 add 字段名 类型；

ALTER TABLE testalter\_tbl DROP i;

ALTER TABLE testalter\_tbl ADD i INT FIRST; 指定新增字段的位置

ALTER TABLE testalter\_tbl DROP i;

ALTER TABLE testalter\_tbl ADD i INT AFTER c; 指定新增字段在哪个字段之后

**改字段类型及名称**

Alter table 数据表名 modify 字段名 类型

Alter table 数据表名 change原字段名 新字段名 类型

ALTER TABLE testalter\_tbl MODIFY c CHAR(10);

ALTER TABLE testalter\_tbl CHANGE j j INT;

## 索引

### 普通索引



Create index 索引名 on 表名（字段名(长度)）

Create index indexName on table(username(10))

索引使用列名称的前10个字符。

因为多数名称的前10个字符通常不同，所以此索引不会比使用列的全名创建的索引速度慢很多

### 唯一索引



## 视图

视图是虚拟的表，是动态的查询数据，数据表中的数据改变视图中的数据也会改变

### 为什么使用视图

重用SQL语句

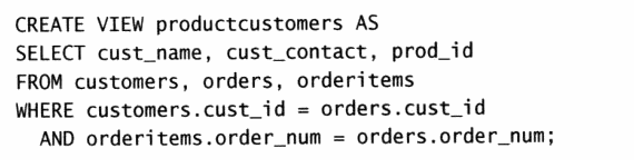
简化复杂的SQL操作，在编写查询后，可以方便的重用它而不必知道它的查询细节

使用表的部分而不是全部

保护数据。可以给用户授权特定部分的访问权限而不是全部表的权限

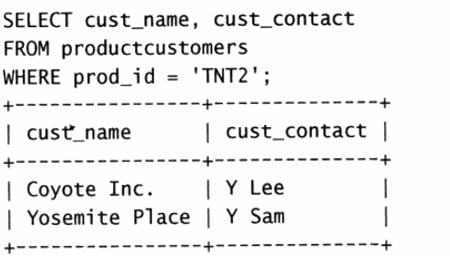
### 视图的创建

create view 视图名 as select语句；



### 检索视图

使用视图的时候就和表一样



### 删除视图

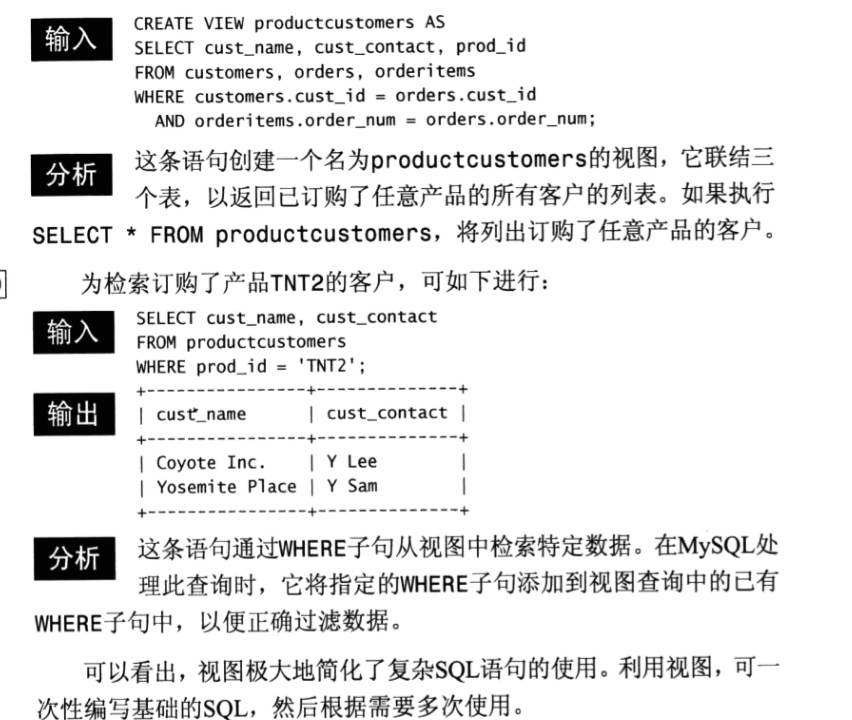
drop view 视图名；

### 查看创建视图语句

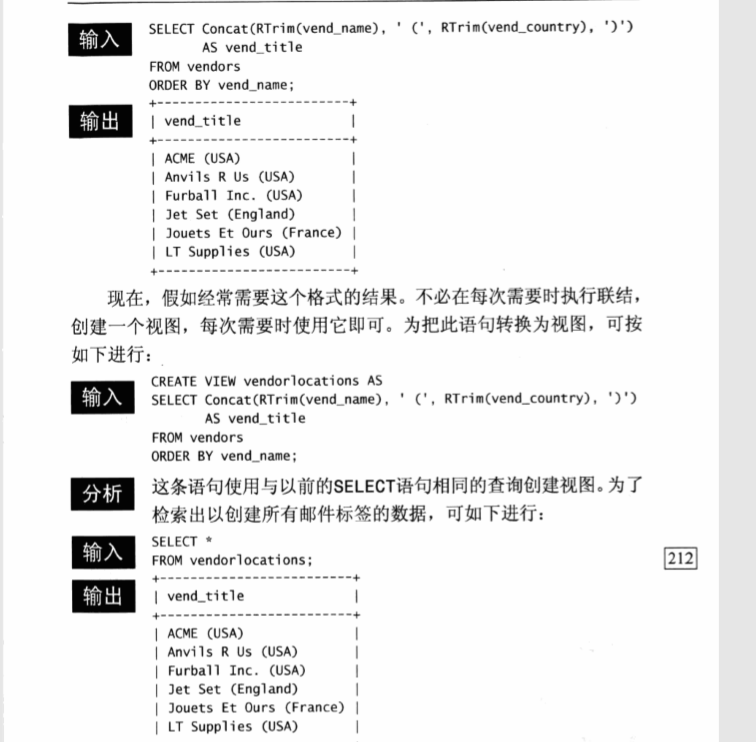
Show create view 视图名；

### 视图使用场景

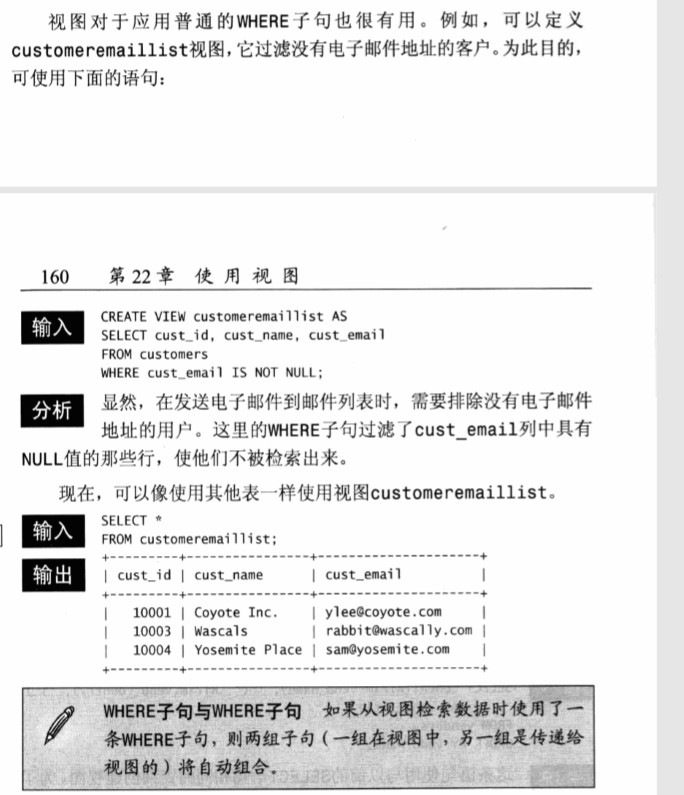
#### 视图简化复杂的联结



#### 视图重新格式化索引的数据



#### 过滤不想要的数据



## 存储过程

### 使用场景

在实际工作中可能会碰到这样的需求，我经常会进行一些操作，这些操作是固定的，但是每次操作都要写几十条语句，这时候就要用到存储过程

### 为什么使存储过程

组为了完成特定功能的SQL 语句集，它存储在数据库中，一次编译后永久有效，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。存储过程是数据库中的一个重要对象。

### 存储过程优点

通过封装复杂的SQL语句集来简化操作

由于不要求反复建立一系列处理步骤，这保证了数据的完整性

简化对变动的管理，如果表明、列明或业务逻辑有变化，只需要改变存储过程的代码。使用人员甚至都不需要知道这些变化

提高性能

### 创建存储过程

#### 不携带参数

Create produce name()

Begin

Select \* from book;

End;

#### 携带参数

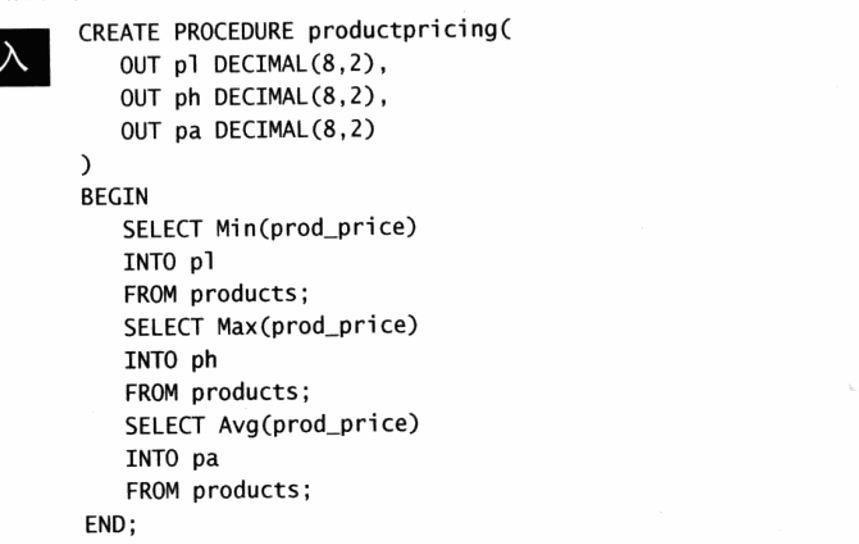
Create produce 存储过程名称（out 参数名 参数类型，…，out 参数名 参数类型）

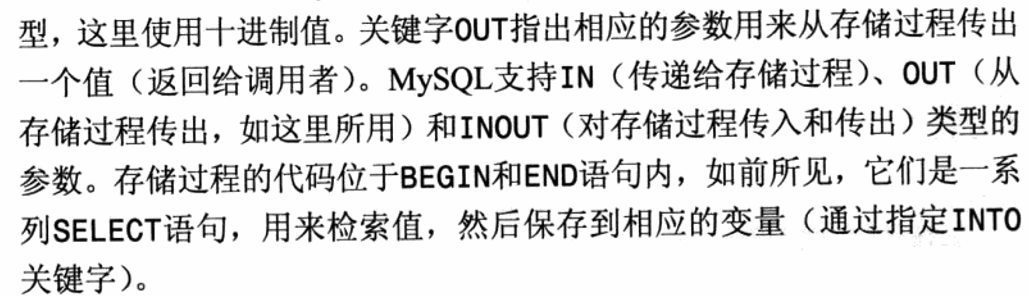
begin

select id into 参1 from A;

select name into 参数2from B;

end;



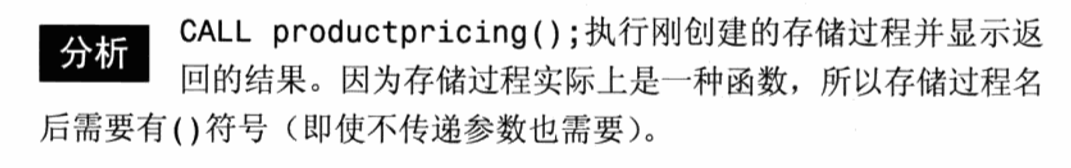


### 执行存储过程

#### 不携带参数

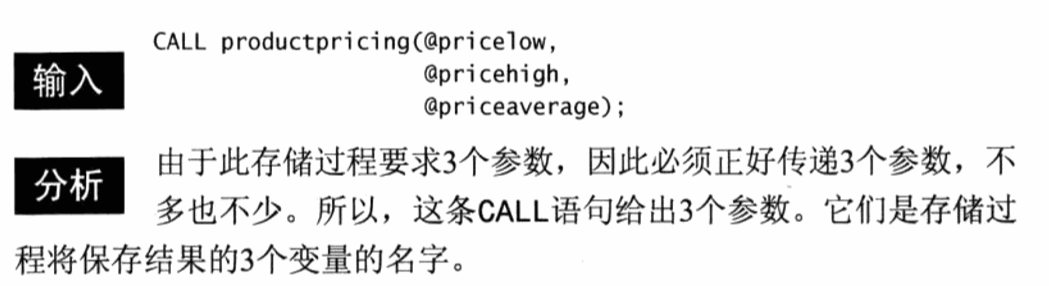
Call 存储过程名称；

Call name();



#### 携带参数

Call存储过程名称（@参数，…，@参数）



### 删除存储过程

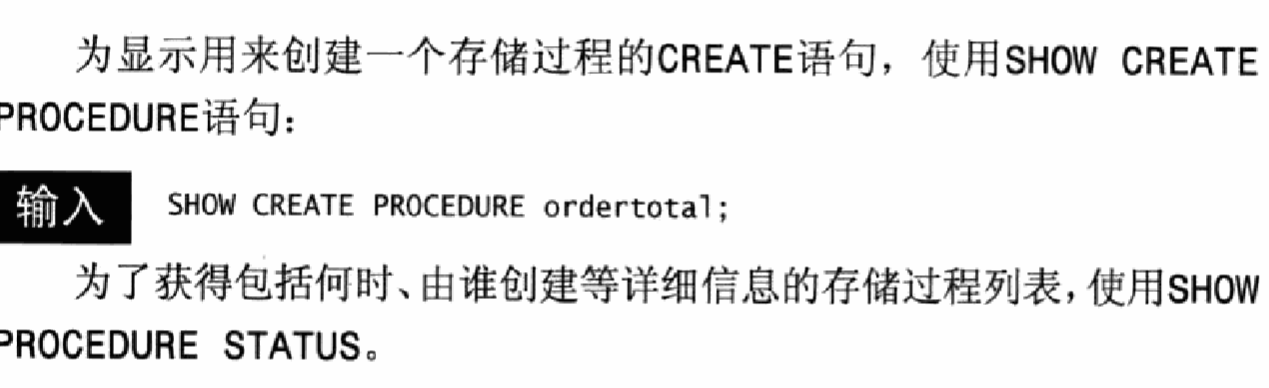
drop produce 存储过程名称；

drop produce name;

### 检查存储过程

为了显示存储过程的create语句

Show create produce 存储过程名称；



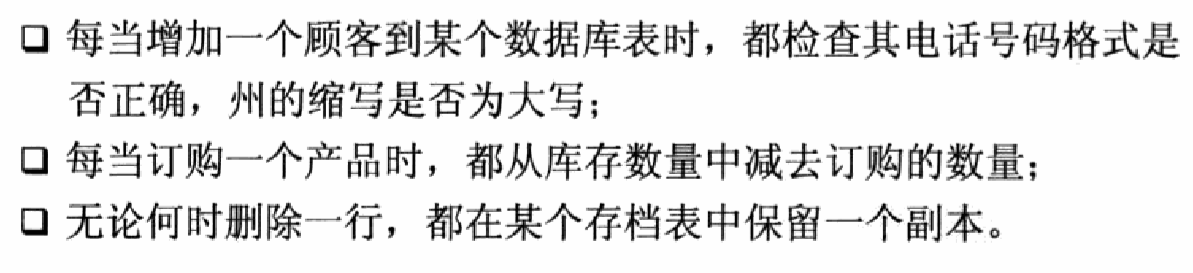
## 游标

以后补充

## 触发器

pass

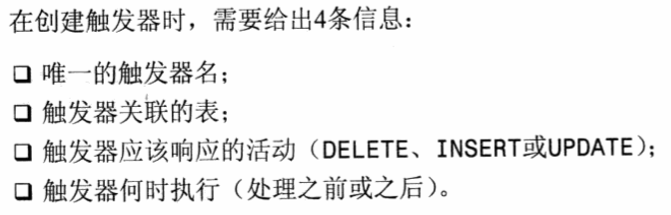
### 使用场景



### 什么是触发器

触发器是让想要的某些SQL语句自动执行的Sql语句；

### 创建触发器



Create trigger 触发器名称 after SQL语句 for SQL 语句

### 使用触发器

### 删除触发器

## Mysql事务

## Mysql自定义函数

参考：

<https://www.cnblogs.com/sharpest/p/9936857.html>

### 创建函数

CREATE FUNCTION 函数名（参数列表） return 返回类型

BEGIN

//设置变量

Set a=a-1;

函数体

END

### 调用函数

Select 函数名（参数列表）

### 查看函数

Show function status;

### 删除函数

Drop function if exists 函数名

### 事例

CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

BEGIN

   set N=N-1;

  RETURN (

      # Write your MySQL query statement below.

     select distinct Salary from Employee order by Salary desc limit N,1

    );

END

DELIMITER $

DROP FUNCTION IF EXISTS myf1;

CREATE FUNCTION myf1() RETURNS INT

BEGIN

DECLARE c INT DEFAULT 0;#定义局部变量

SELECT COUNT(\*) INTO c#赋值

FROM mall\_pro;

RETURN c;

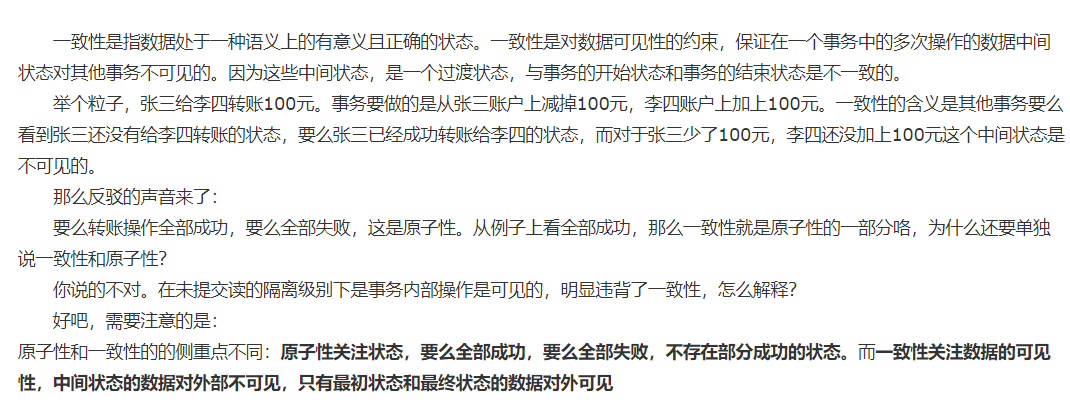
END $

delimiter就是告诉mysql解释器，该段命令是否已经结束了，是否可以执行了。  
默认情况下，delimiter是分号;，遇到分号就执行。  
后面的双美元符号 就是告诉mysql，遇到双美元符号再执行

### 使用场景

MySQL 事务主要用于处理操作量大，复杂度高的数据。比如说，在人员管理系统中，你删除一个人员，你既需要删除人员的基本资料，也要删除和该人员相关的信息，如信箱，文章等等，这样，这些数据库操作语句就构成一个事务！

### 事务特性



**原子性：**一个事务（transaction）中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。

**一致性：**在事务开始之前和事务结束以后，数据库的完整性没有被破坏。这表示写入的资料必须完全符合所有的预设规则，这包含资料的精确度、串联性以及后续数据库可以自发性地完成预定的工作。

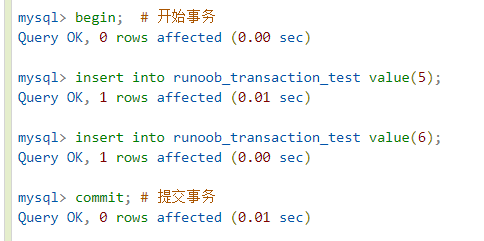
**隔离性：**数据库允许多个并发事务同时对其数据进行读写和修改的能力，隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致。事务隔离分为不同级别，包括读未提交（Read uncommitted）、读提交（read committed）、可重复读（repeatable read）和串行化（Serializable）。

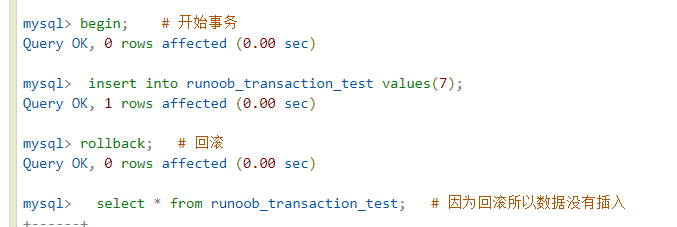
**持久性：**事务处理结束后，对数据的修改就是永久的，即便系统故障也不会丢失。

**BEGIN** 开始一个事务

**ROLLBACK** 事务回滚

**COMMIT** 事务确认





## 多张表有关联，每一个表都存着上一个表的外键，不在for循环里面使用Sql查询如何做到

## 多表删除

Mysql 时间字段默认设为Null

添加的时候 time=”” 直接添加是不行的 要用time=null

## Centos7 删除mysql

参考: <https://www.cnblogs.com/finalanddistance/p/10388818.html>

<https://www.jianshu.com/p/ef58fb333cd6>

## centos7 mysql创建用户并分配权限

起因：数据库被黑 ，root帐号不够安全，所以要新建用户，建好发现没有权限建立数据库..

参考：

<https://blog.csdn.net/w851685279/article/details/89810346>

<https://www.cnblogs.com/pxblog/p/10721939.html>

### 查看mysql数据数据库所有用户

**SELECT DISTINCT CONCAT('User: ''',user,'''@''',host,''';') AS query FROM mysql.user;**

### 2.创建新用户

create user ‘username’ @’host’ identified by ‘password’;

username 用户名

host指定哪个主机(ip指定)可以登录，如果是本地用户可用localhost，%表示该用户可以从任意主机登录

password 为密码

### 3.修改用户密码

set password for ‘username’@‘host’=password(‘newpassword’);

set password for 'SWE16081'@'%'=password('12345678');

### 4.删除用户

drop user ‘username’@’host’;

drop user ‘SWE16081’@’%’;

### 分配权限

# 刷新权限 权限更新后刷新才会起作用

FLUSH PRIVILEGES;

grant privileges on dbname.tablename to ‘username’@‘host’;

privileges：用户的操作权限，可以是all, 表示所有权限，也可以是select,update，insert,delete权限，多个权限的名词,相互之间用逗号分开

dbname：数据库名

tablename：表名，如果要授予该用户对所有数据库和表的相应操作权限则可用表示，如\*.\*

如果需要授权其他用户，用以下命令

grant privileges on dbname.tablename to ‘username’@‘host’ with grant option;

IDENTIFIED BY 指定用户的登录密码,该项可以省略(某些版本下回报错，必须省略)。

WITH GRANT OPTION 这个选项表示该用户可以将自己拥有的权限授权给别人。注意：经常有人在创建操作用户的时候不指定WITH GRANT OPTION选项导致后来该用户不能使用GRANT命令创建用户或者给其它用户授权。  
备注：可以使用GRANT重复给用户添加权限，权限叠加，比如你先给用户添加一个select权限，然后又给用户添加一个insert权限，那么该用户就同时拥有了select和insert权限。

### 6.撤销用户权限

revoke privilege on dbname.tablename from ‘username’@‘host’;

revoke all on \*.\* ‘SWE16081’@’host’;

7.查看用户权限

show grants for username;

show grants for SWE16081;

## mysql limit查找前几条数据

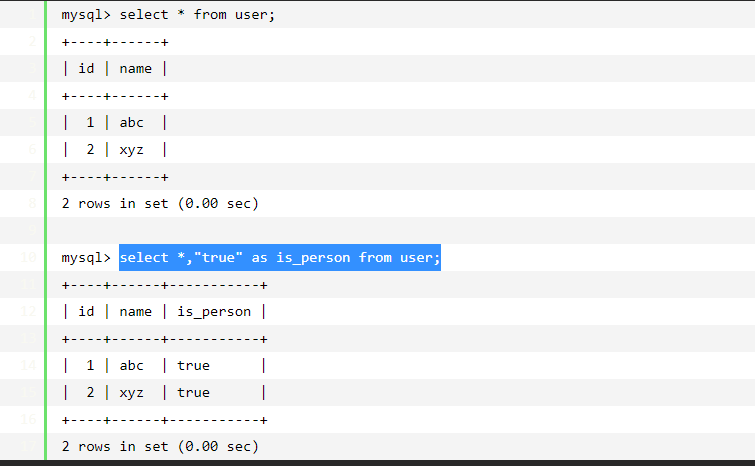
<https://www.cnblogs.com/xuxm2007/archive/2010/11/16/1878211.html>

select \* from a limit 3;//前三条

select \* from a limit 7,6;//从第八条开始往后取6条

## mysql 查询的时候添加字段

select \*,"true" as is\_person from user;



成绩排名，分数一样并列

select Score,(select count(distinct Score)+1 from Scores as b where  b.Score>a.Score) as "Rank"  from Scores as a order by a.Score desc;

## mysql原生高级查询

## 字段拼接

<https://www.cnblogs.com/Jason-Xiang/p/10441498.html>

## FIND\_IN\_SET

## 子查询问题

SELECT

id,

storeName

FROM

( (SELECT id FROM t\_store) AS id ,

(SELECT storeName FROM t\_store) AS storeName )

### In 可以匹配两个字段

select b1.Name as Department,a1.Name as Employee,Salary

from  Employee a1, Department b1

where  a1.DepartmentId=b1.Id

and (Salary,b1.Id) in(

select max(Salary) as Salary,b.Id

from  Employee a, Department b

where  a.DepartmentId=b.Id   group by b.Id

);

## Insert into select from

INSERT INTO table2  
SELECT \* FROM table1;

从一个表复制数据插入到另一个表中

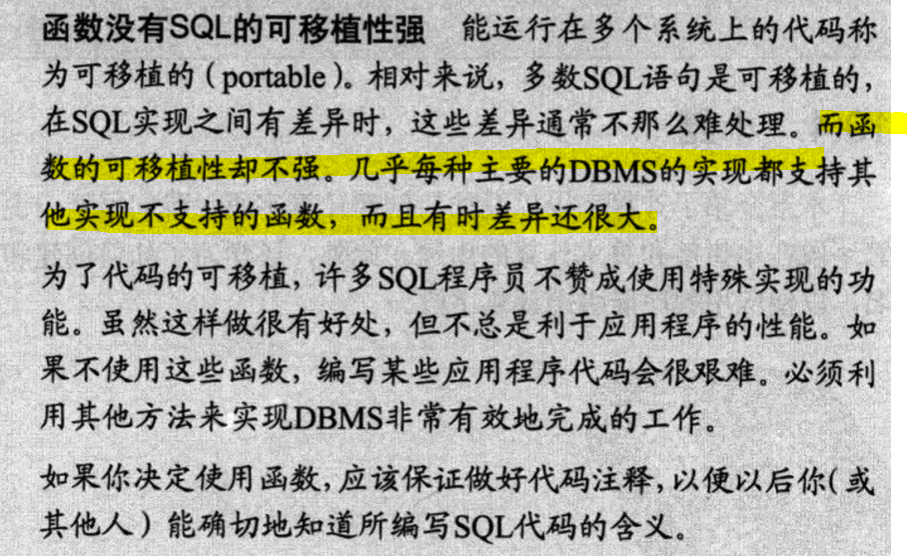
## Mysql IFNULL

## Mysql条件判断

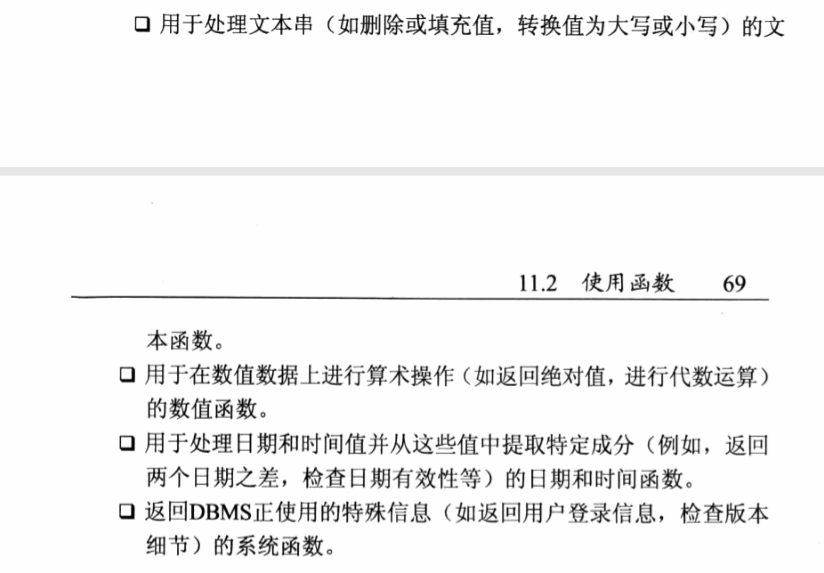
Mysql占位符

## Mysql常用函数

### 函数没有SQL的可移植性强

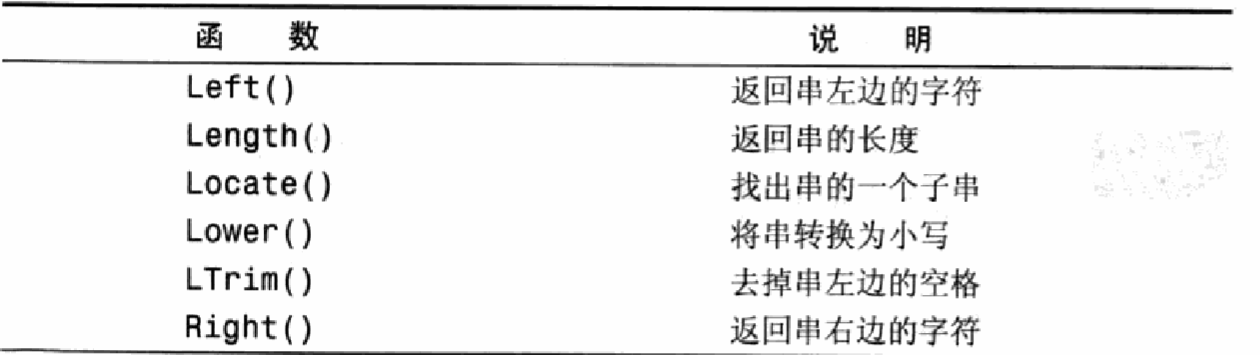


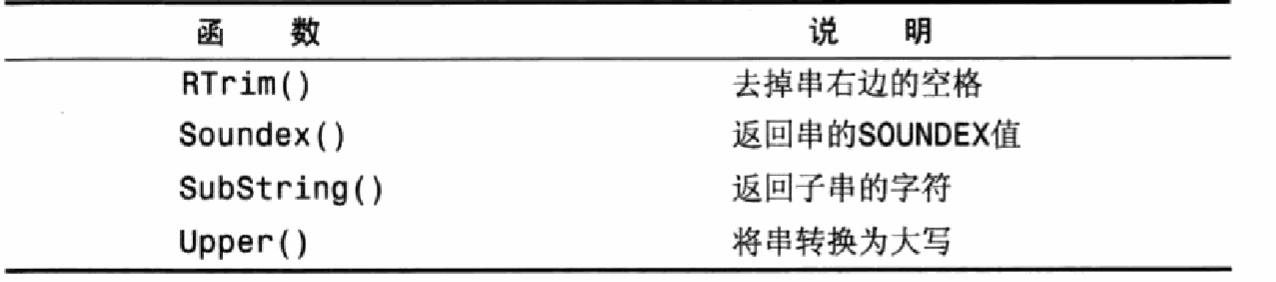
### 大多数SQL支持一下类型函数



### 1.文本处理函数

#### 常用的文本处理函数





#### Left(字段,number)返回从字段左侧到number个字符

select left(cachettagname,4) from cachet where cachetid=108;

#### right(字段,number)返回从字段右侧到number个字符

select right(cachettagname,4) from cachet where cachetid=108;

#### Length()返回字段长度

select length(cachettagname,4) from cachet where cachetid=108;

#### locate(subStr,string) 查找子串

查找’aa’在cachettagname字段的位置

select locate('aa',cachettagname) from cachet where cachetid=108;

如果不存在则返回0

#### RTrim()去除列值右边的空格

select RTrim(cachettagname) from cachet where cachetid=108;

#### LTrim()去除列值左边的空格

select LTrim(cachettagname) from cachet where cachetid=108;

#### Trim()去除列值两边的空格

select Trim(cachettagname) from cachet where cachetid=108;

#### Upper()将文本转换成大写

select Upper(cachettagname) from cachet where cachetid=108;

#### soundex返回字段的soundex值

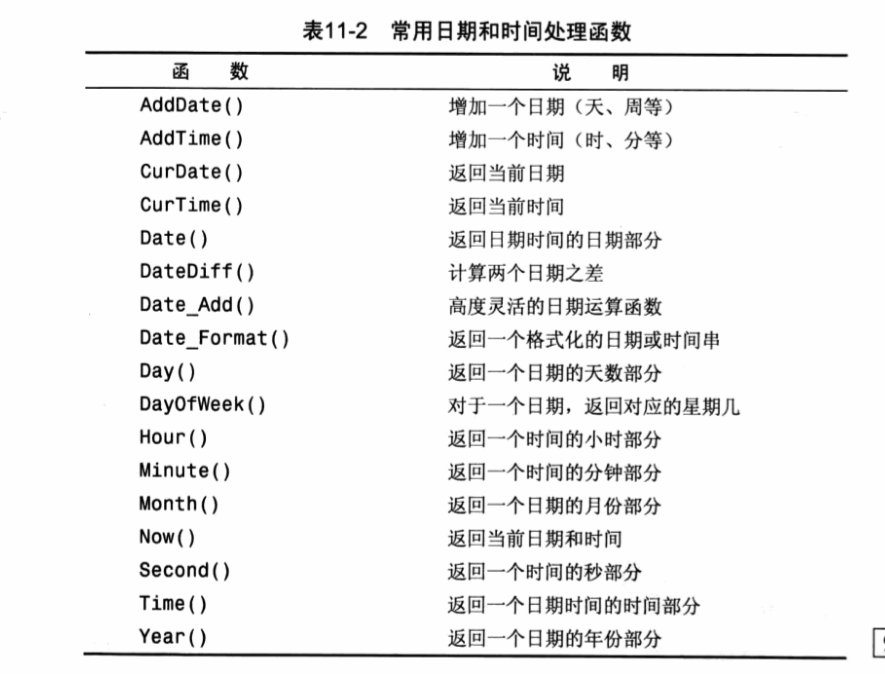
select soundex(cachettagname) from cachet where cachetid=108;

#### Lower()将文本转换成小写

select Lower(cachettagname) from cachet where cachetid=108;

### 2. 时间函数

#### 常见的时间函数

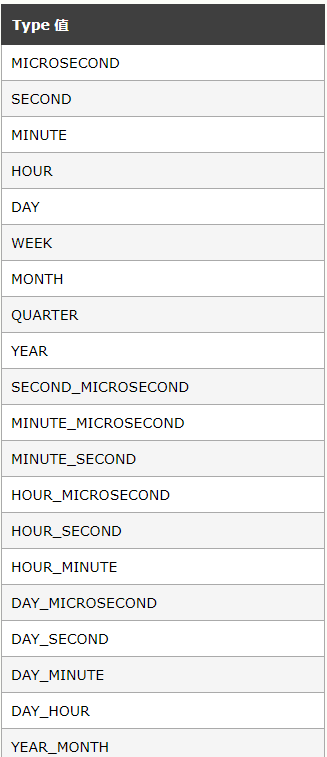


#### AddDate()向日期添加指定的时间间隔

DATE\_ADD(date,INTERVAL expr type)

==

AddDate()与DATE\_ADD是同义词



SECOND\_MICROSECOND下面addtime不能用？

select cachetId,DATE\_ADD(created\_at,interval 2 DAY) from cachet where cachetid=108;

#### CurDate()返回当前日期

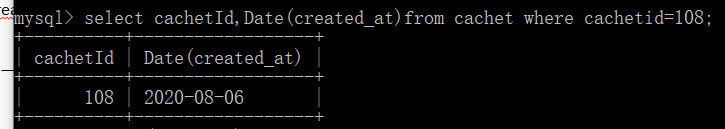
select cachetId,CurDate() from cachet where cachetid=108;

#### CurTime()返回当前时间

select cachetId,CurTime() from cachet where cachetid=108;

#### Date()返回日期时间的部分

select cachetId,Date(created\_at)from cachet where cachetid=108;



#### AddTime()增加一个时间

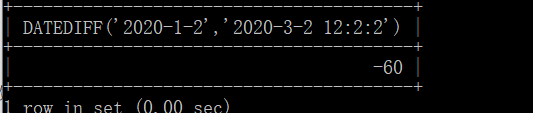
select cachetId,AddTime(created\_at,'2:2:2') from cachet where cachetid=108;

#### DATEDIFF()

DATEDIFF(date1,date2)

返回两个日期之间的天数，第一个减第二个

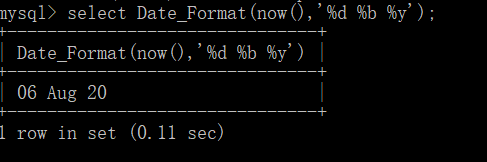
select DATEDIFF('2020-1-2','2020-3-2 12:2:2');



#### DATE\_FORMAT()函数用于以不同的格式显示日期/时间数据

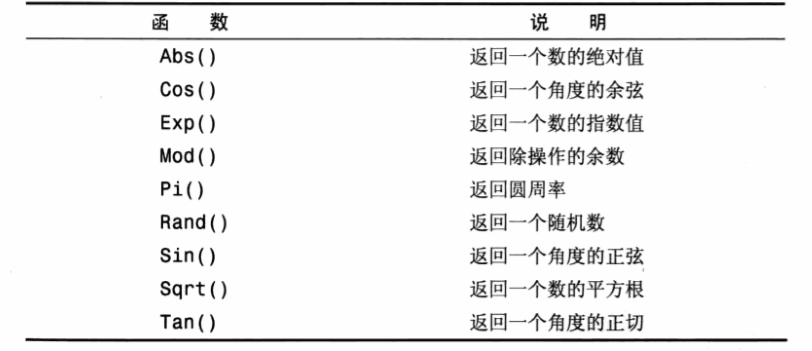
DATE\_FORMAT(date,format)



### 3.数值处理函数

主要用于代数，三角、几何运算

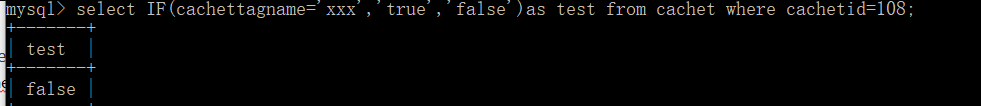


### 4.条件判断函数

#### IF(expr,v1,v2)

如果表达式expr成立，返回v1，否则返回v2

select IF(cachettagname='xxx','true','false')as test from cachet where cachetid=108;

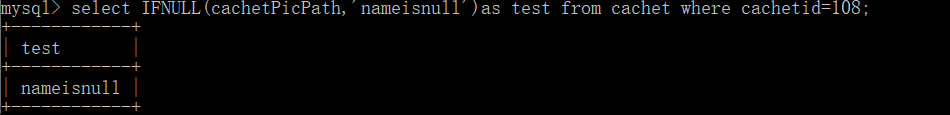


#### IFNULL(v1,v2)

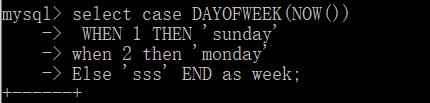
如果v1的值不为空，就显示v1的值，否则显示v2的值

前提是这个字段可以为空

select IFNULL(cachetPicPath,'nameisnull')as test from cachet where cachetid=108;



#### Case when





## MySQL计算时间的函数

### Timestampdiff()

Timestampdiff(interval,datetime1,datetime3)

返回（时间2-时间1）的时间差，结果单位由interval参数给出

Interval可选值有

frac\_second 毫秒（低版本不支持，用second，再除于1000）

second 秒

minute 分钟

hour 小时

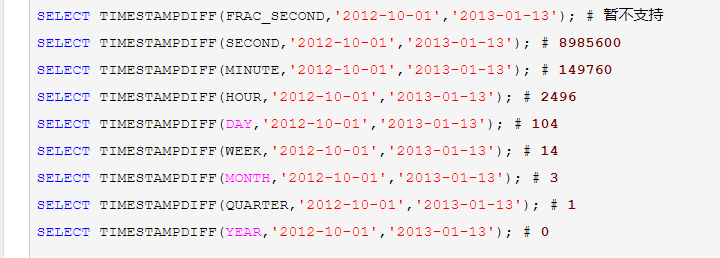
day 天

week 周

month 月

quarter 季度

year 年



### DATEDIFF()

DATEDIFF(date1,date2)

返回两个日期之间的天数

### Timediff()

timediff(time1,time2)

SELECT TIMEDIFF('2018-05-21 14:51:43','2018-05-19 12:54:43');

# 49:57:00  
注意：该方法两个参数必须为日期格式

### now()函数返回的是当前时间的年月日时分秒

### curdate()函数返回的是年月日信息

### curtime()函数返回的是当前时间的时分秒信息

### 对一个包含年月日时分秒日期格式化成年月日日期，可以使用DATE(time)函数

## date\_sub时间计算

## mysql先按分数排序如果一样再按时间排序

select \* from table\_name order by number asc,time asc

## mysql left join 多个字段

select a.\*, b.\*

from a

left join b on a1=b1 and a2=b2 and a3=b3

## 数据字典

参考：<https://blog.csdn.net/qq_39530754/article/details/85130249>

数据字典是描述数据的信息集合，是对系统中使用的所有数据元素的定义的集合。

### 两种形式

一、把主体的属性代码化放入独立的表中，不是和主体放在一起，主体中只保留属性的代码。这里属性的数量是不变的，而属性取值的数量可以是变化的。

二、用一个表来放结构相同的所有属性信息，不同属性的不同取值统一编码，用“类型”来区别不同的属性，主体中保留属性代码的列表。这样主体所拥有的属性数量就是可变的了。

## Mysql运行机制

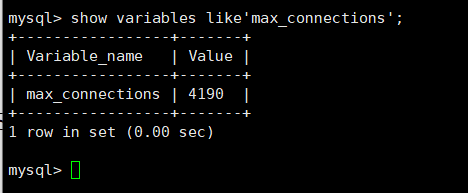
Mysql是一个单进程多线程的数据库

Mysql每一个连接实例都是一个多线程

## Mysql连接与释放

默认mysql没有设置，这是因为最大连接数决定于你服务器硬件配置，如果配置很低，将最大连接数修改为很大，也一样不会有效果的，对于百万并发量而言，如果什么处理都不做这就意味着，要建立百万的连接数，那么我们将连接数调成百万测试一下。

max\_connections=1000000



我自己机器的上线是4190。因此面对百万并发量时我的mysql数据库将直接瘫痪。

但是有的时候我连接上mysql不会自己主动释放，我们自然而然的想到mysql使用了什么机制释放我们的连接。

### Mysql连接断开机制

wait\_timeout 非交互连接的超时时间，非交互连接指连接池式、非客户端方式连接。

默认为28800即8小时，超过这个时间mysql服务器会主动低切断连接，将连接变为状态为sleep的连接

Interactive\_timeout针对交互式连接，即在mysql\_real\_connect()函数中使用CLIENT\_INTERACTIVE选项，通过mysql客户端连接数据库是交互式连接，通过jdbc连接数据库是非交互式连接

show processlist命令查看当前系统的连接状态

1. 控制连接最大空闲时长的wait\_timeout参数。

2. 对于非交互式连接，类似于jdbc连接，wait\_timeout的值继承自服务器端全局变量wait\_timeout。

    对于交互式连接，类似于mysql客户单连接，wait\_timeout的值继承自服务器端全局变量interactive\_timeout。

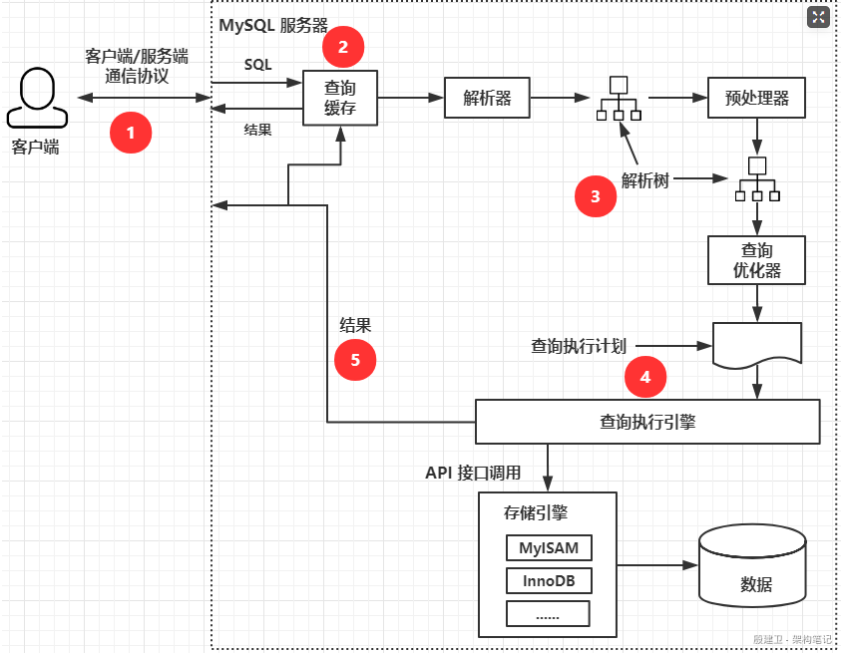
3. 判断一个连接的空闲时间，可通过show processlist输出Sleep状态的时间

当为交互连接

虽然超过一定时间会断开连接但是这样一样不能满足百万的并发量。那么mysql是如何处理高并发的场景的呢？在了解这个之前有一个问题困扰我，那就是mysql在他的并发量之内是如何处理的即执行sql语句的流程

### Mysql执行sql语句的流程

参考：<https://www.yuque.com/yinjianwei/vyrvkf/ri4ks7>



## Sql语句 where拆分

## mysql并发测试

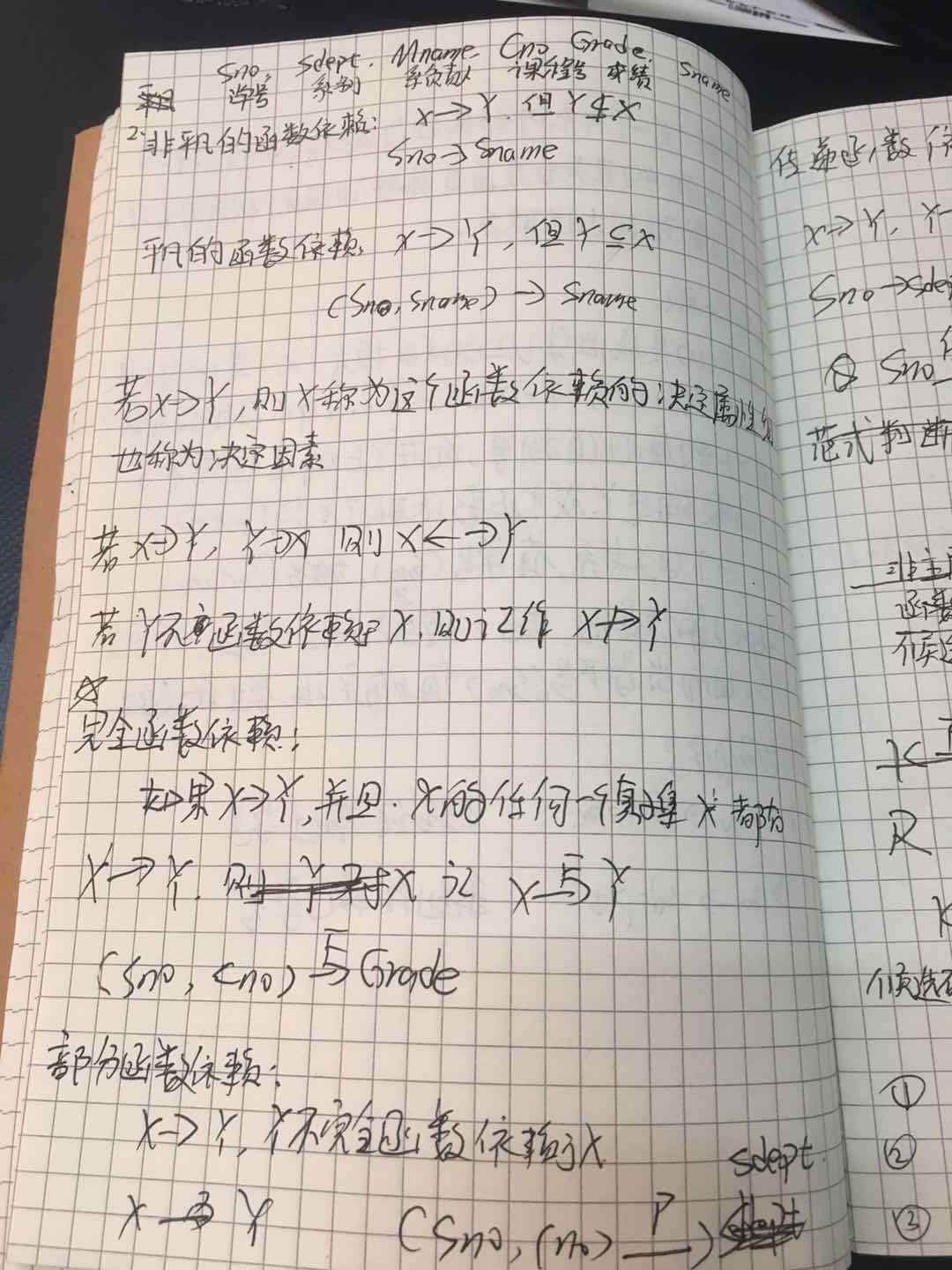
见G:\桌面\学习文件\后端\backEnd\数据库\mysql\ mysql并发测试

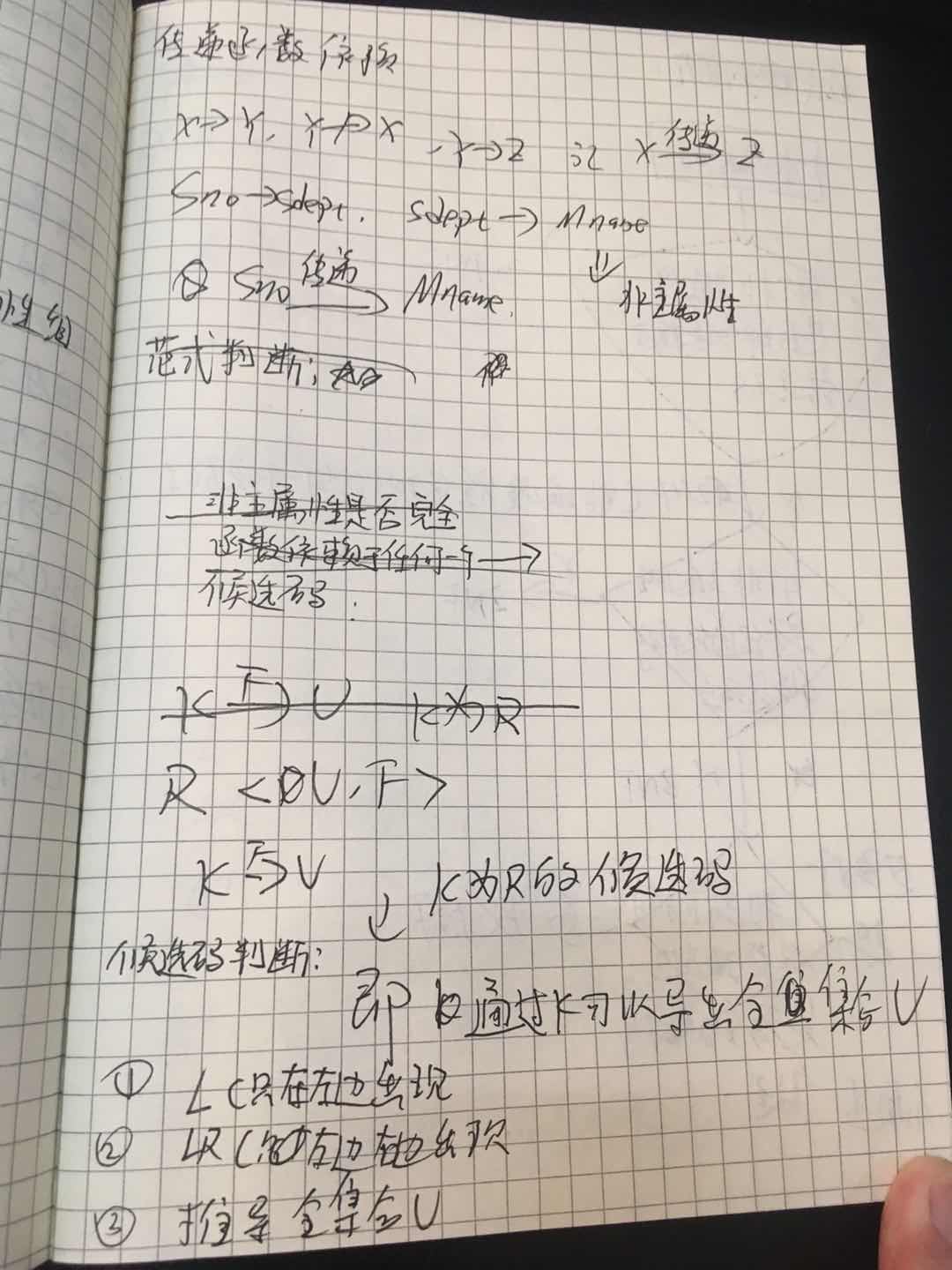
## 范式

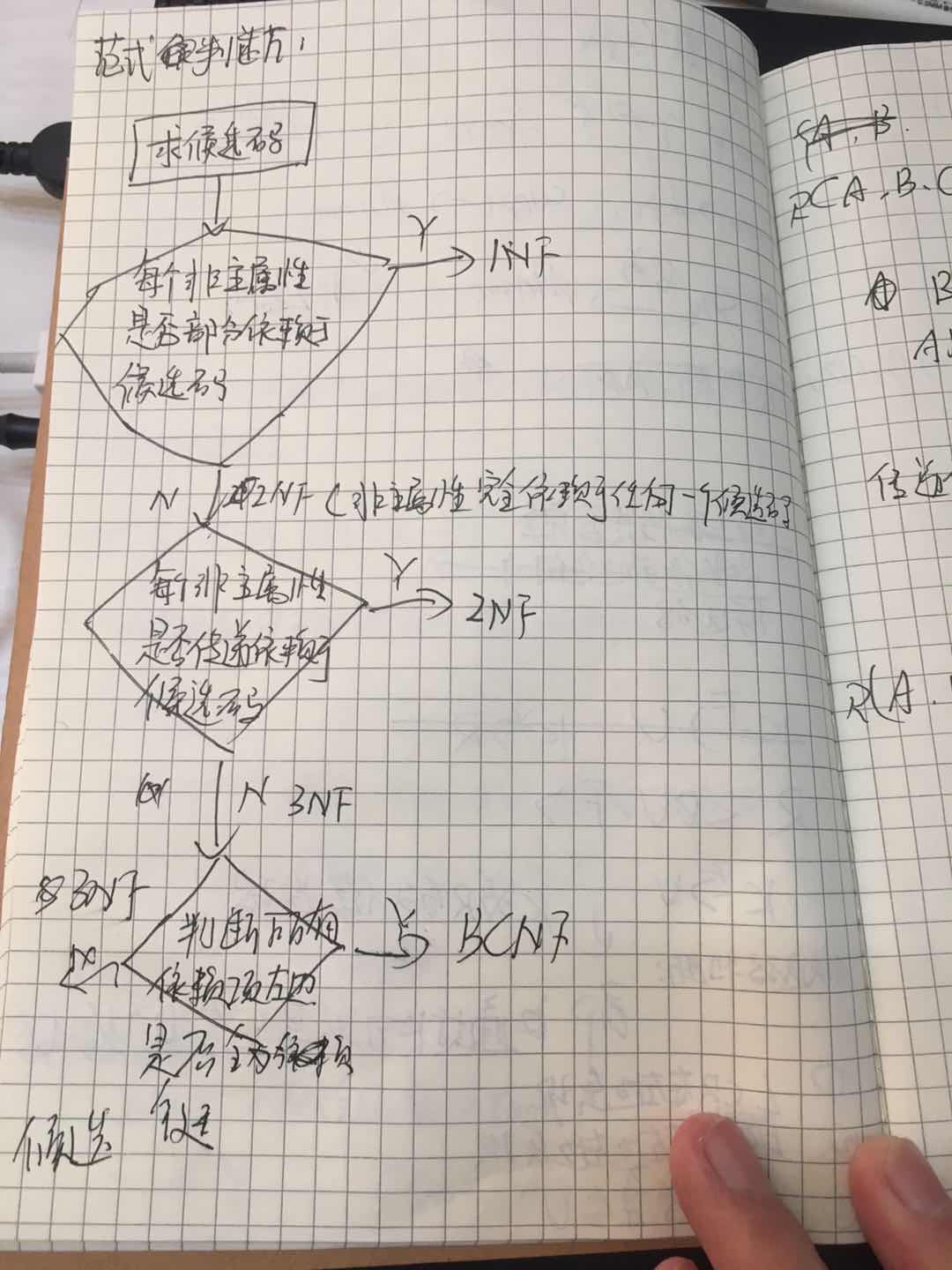
**第一范式：**每个列都不可以再拆分。

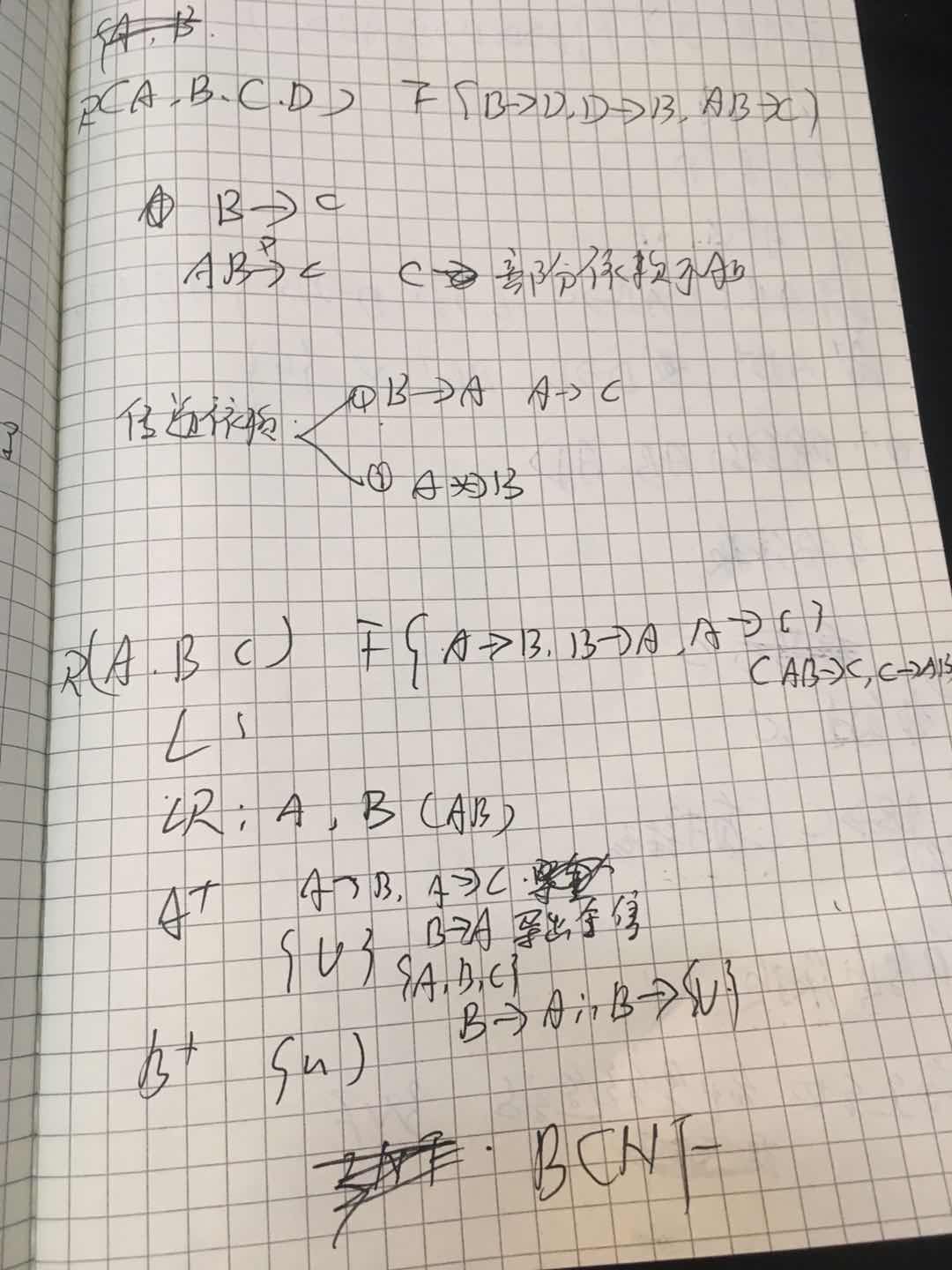
**第二范式：**在第一范式的基础上，非主键列完全依赖于主键，而不能是依赖于主键的一部分。

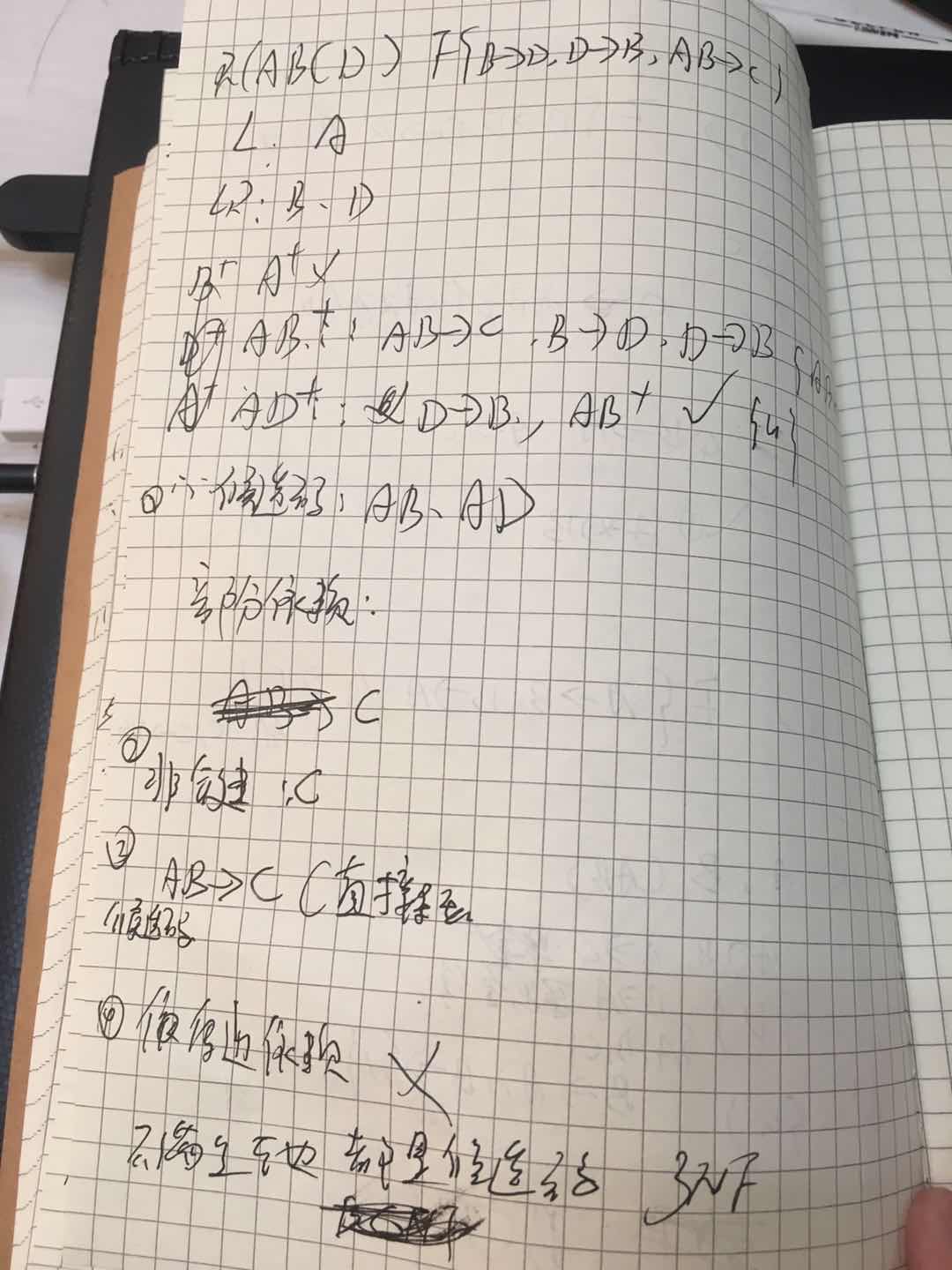
**第三范式：**在第二范式的基础上，非主键列只依赖于主键，不依赖于其他非主键











### 第一范式

表中每个属性不可分割

### 第二范式

表中有主键，其他字段都依赖于主键

### 第三范式