# 表

引用：<https://www.cnblogs.com/plf-Jack/p/11177019.html>

全部相关解说：<http://c.biancheng.net/view/2601.html>

## 索引

##### 简介

它是数据库性能调优技术的基础，常用于实现数据的快速检索

索引就是根据表中的一列或若干列按照一定顺序建立的列值与记录行之间的对应关系表，实质上是一张描述索引列的列值与原表中记录行之间一一对应关系的有序表。

**通常有以下两种方式访问数据库表的行数据：**

###### 顺序访问

顺序访问是在表中实行全表扫描，从头到尾逐行遍历，直到在无序的行数据中找到符合条件的目标数据。

使用顺序访问方式将会遍历所有的数据，花费大量的时间，显然会影响数据库的处理性能。

###### 索引访问

索引访问是通过遍历索引来直接访问表中记录行的方式。

使用这种方式的前提是对表建立一个索引，在列上创建了索引之后，查找数据时可以直接根据该列上的索引找到对应记录行的位置

**索引分类：**

###### B-树索引

B-树索引又称为 BTREE 索引，目前大部分的索引都是采用 B-树索引来存储的。

###### 哈希索引

哈希（Hash）一般翻译为“散列”，也有直接音译成“哈希”的，就是把任意长度的输入（又叫作预映射，pre-image）通过散列算法变换成固定长度的输出，该输出就是散列值。

##### 使用和注意

索引可以在一些情况下加速查询，但是在某些情况下，会降低效率。

**弊端：**

* 创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加
* 除了数据表占数据空间之外，每一个索引还要占一定的物理空间。如果要建立聚簇索引，那么需要的空间就会更大。
* 当对表中的数据进行增加、删除和修改的时候，索引也要动态地维护，这样就降低了数据的维护速度。

**如何创建索引才是高效率的呢**

* 在作为主键的列上创建索引，强制该列的唯一性，并组织表中数据的排列结构。
* 在经常使用表连接的列上创建索引，这些列主要是一些外键，可以加快表连接的速度。
* 在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，所以其指定的范围是连续的。
* 在经常需要排序的列上创建索引，因为索引已经排序，所以查询时可以利用索引的排序，加快排序查询。
* 在经常使用 WHERE 子句的列上创建索引，加快条件的判断速度

**索引失效的时候**

##### 创建

###### create index语句

**语法：**CREATE <索引名> ON <表名> (<列名> [<长度>] [ ASC | DESC])

Create index s\_number\_i on elective (s\_number)

###### create table语句

**语法：**KEY | INDEX [<索引名>] [<索引类型>] (<列名>,…)

**一般索引的创建：**

Index(id)

Key index\_name (id)

**创建唯一索引：**

Unique index (id)

Unique key index\_name (id)

以上没实践，实践后将会粘贴实践的代码

###### alter table 语句

**语法：**ADD INDEX [<索引名>] [<索引类型>] (<列名>,…)

Alert table elective add index c\_number\_i (c\_number);

##### 查看

##### 修改和删除索引

##### 索引设计原则

##### 索引种类

普通索引

唯一索引

全文索引

单列索引

多列索引

空间索引

<https://blog.csdn.net/weixin_41459547/article/details/87967502>

## 约束

##### 什么是约束

是一种限制，对某一个东西的限制。例如宪法规定了你违反的事情你是不能做的。这就是一种约束

数据库的约束，是对数据的安全性，完整性的保证

##### unique key 普通约束

唯一性约束，表示这个不能出现重复的值

在创建表的方式设置唯一约束

CREATE TABLE `test` (

`id` varchar(255) NOT NULL UNIQUE,

`goods\_id` varchar(255) DEFAULT NULL,

`name` varchar(255) DEFAULT NULL

# UNIQUE KEY `id` (`id`), 也可以这么设置

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

修改表的方式设置唯一约束

ALTER TABLE test MODIFY id varchar(255) UNIQUE not null

这个约束不能出现重复值，但是能重复插入null(如果没设置not null约束)

##### not null

`id` varchar(255) NOT NULL UNIQUE,

非空约束，表示这个字段的值不能为空

##### default

`goods\_id` varchar(255) DEFAULT NULL

默认值，用于给某一个字段设置默认值

##### primary key 主键约束

###### create table 语句

id char(19) primary key,

PRIMARY KEY (`id`)

id char(19) primary key auto\_increment

###### alter table 语句

ALTER TABLE test MODIFY id varchar(255) PRIMARY KEY not null

Alter tabel elective add coonstraint index\_id primary key (id);

##### foreign key 外键约束

数据出现了大量的重复

CREATE TABLE `test` (

`id` varchar(255) NOT NULL UNIQUE,

`goods\_id` varchar(255) DEFAULT NULL,

`name` varchar(255) DEFAULT NULL,

Foreign key(goods\_id) references table\_xxx(id)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

## 事务

##### 什么是事务

##### 事务的ACID特性

事务具有四个特征：原子性（ Atomicity ）、一致性（ Consistency ）、隔离性（ Isolation ）和持续性（ Durability ）。这四个特性简称为 ACID 特性。

**原子性。**事务是数据库的逻辑工作单位，事务中包含的各操作要么都做，要么都不做

**一致性。**事 务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。因此当数据库只包含成功事务提交的结果时，就说数据库处于一致性状态。如果数据库系统 运行中发生故障，有些事务尚未完成就被迫中断，这些未完成事务对数据库所做的修改有一部分已写入物理数据库，这时数据库就处于一种不正确的状态，或者说是 不一致的状态

**隔离性。**一个事务的执行不能其它事务干扰。即一个事务内部的操作及使用的数据对其它并发事务是隔离的，并发执行的各个事务之间不能互相干扰

**持续性：**也称永久性，指一个事务一旦提交，它对数据库中的数据的改变就应该是永久性的。接下来的其它操作或故障不应该对其执行结果有任何影响

##### MySql的四种隔离

###### Read Uncommitted(读未提交)

在该隔离级别，所有事务都可以看到其他未提交事务的执行结果。本隔离级别很少用于实际应用，因为它的性能也不比其他级别好多少。读取未提交的数据，也被称之为脏读（Dirty Read）

###### Read Committed(读提交)

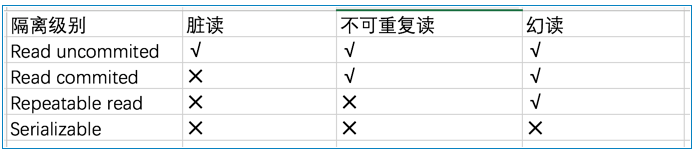
这是大多数数据库系统的默认隔离级别（但不是MySQL默认的）。它满足了隔离的简单定义：一个事务只能看见已经提交事务所做的改变。这种隔离级别 也支持所谓的不可重复读（Nonrepeatable Read），因为同一事务的其他实例在该实例处理其间可能会有新的commit，所以同一select可能返回不同结果

###### Repeatable Read(可重读)

这是MySQL的默认事务隔离级别，它确保同一事务的多个实例在并发读取数据时，会看到同样的数据行。不过理论上，这会导致另一个棘手的问题：幻读 （Phantom Read）。简单的说，幻读指当用户读取某一范围的数据行时，另一个事务又在该范围内插入了新行，当用户再读取该范围的数据行时，会发现有新的“幻影” 行。InnoDB和Falcon存储引擎通过多版本并发控制（MVCC，Multiversion Concurrency Control）机制解决了该问题

###### Serializable(可序列化)

这是最高的隔离级别，它通过强制事务排序，使之不可能相互冲突，从而解决幻读问题。简言之，它是在每个读的数据行上加上共享锁。在这个级别，可能导致大量的超时现象和锁竞争



<https://blog.csdn.net/m0_37450089/article/details/93380246?utm_medium=distribute.pc_relevant.none-task-blog-baidujs-7>

## 级联

##### 定义

级联操作指的就是，当你操作主表时，自动操作从表

##### 级联的两种方式

###### 表定义阶段进行定义

**级联的删除**

create table student\_b(

class\_id int,

name char(10),

age int,

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

on delete cascade

) charset utf8;

**级联的更新**

create table student\_b(

class\_id int,

name char(10),

age int,

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

on update cascade

) charset utf8;

**级联的删除和更新**

create table teacher(

id int primary key auto\_increment,

name char(20),

gender char(21),

dept\_id int,

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

on update cascade

on delete cascade

) charset utf8;

###### 表存在的情况下

先将之前的外键删除掉，然后通过alter增加外键级联

alter table teacher drop foreign key 外键的ID;

alter table teacher add constraint dept\_id foreign key(dept\_id) references dept(id) on delete cascade on update cascade;

## Sql语句分类

#### DDL（Data Definition Languages）

数据定义语言，这些语句定义了不同的数据段、数据库、表、列、索引等数据库对象的定义。常用的语句关键字主要包括 create、drop、alter等。

**数据库安装的时候自动生成的表的作用：**

**information\_schema**：主要存储了系统中的一些数据库对象信息。比如用户表信息、列信息、权限信息、字符集信息、分区信息等。

**cluster**：存储了系统的集群信息。

**mysql**：存储了系统的用户权限信息。

**test**：系统自动创建的测试数据库，任何用户都可以使用。

##### 创建数据库

create database test1;

##### 删除数据库

drop database dbname

##### 创建表

CREATE TABLE `test` (

`id` varchar(255) NOT NULL UNIQUE,

`goods\_id` varchar(255) DEFAULT NULL,

`name` varchar(255) DEFAULT NULL

# UNIQUE KEY `id` (`id`), 也可以这么设置

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

##### 删除表

DROP TABLE tablename

##### 修改表

###### 修改表类型

ALTER TABLE tablename MODIFY [COLUMN] column\_definition [FIRST | AFTER col\_name]

alter table emp modify ename varchar(20);

修改表引擎

ALTER TABLE tablename ENGINE = MyISAM;

###### 增加表字段

ALTER TABLE tablename ADD [COLUMN] column\_definition [FIRST | AFTER col\_name]

alter table emp add column age int(3);

###### 删除表字段

ALTER TABLE tablename DROP [COLUMN] col\_name

alter table emp drop column age;

###### 字段改名

ALTER TABLE tablename CHANGE [COLUMN] old\_col\_name column\_definition [FIRST|AFTER col\_name]

alter table emp change age age1 int(4) ;

###### 修改字段排列顺序

alter table emp add birth date after ename;

alter table emp modify age int(3) first;

###### 表改名

alter table emp rename emp1;

#### DML（Data Manipulation Language）

数据操纵语句，用于添加、删除、更新和查询数据库记录，并检查数据完整性，常用的语句关键字主要包括 insert、delete、udpate 和select 等。(增添改查）

##### 插入记录

insert into emp (ename,hiredate,sal,deptno) values('zzx1','2000-01-01','2000',1);

##### 删除记录

##### 查询记录

##### 修改记录。

#### DCL（Data Control Language）

数据控制语句，用于控制不同数据段直接的许可和访问级别的语句。这些语句定义了数据库、表、字段、用户的访问权限和安全级别。主要的语句关键字包括 grant、revoke 等。

## 同oracle的不同之处

<https://blog.csdn.net/weixin_41799903/article/details/89021062>

## ****MyISAM 和 InnoBD区别：****



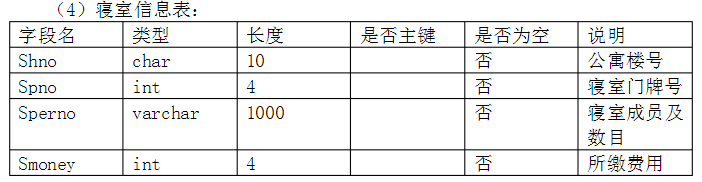
# 学生管理系统

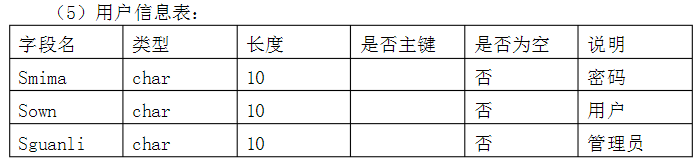
## 表的创建

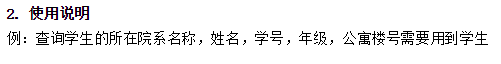


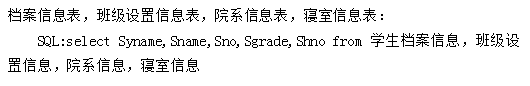




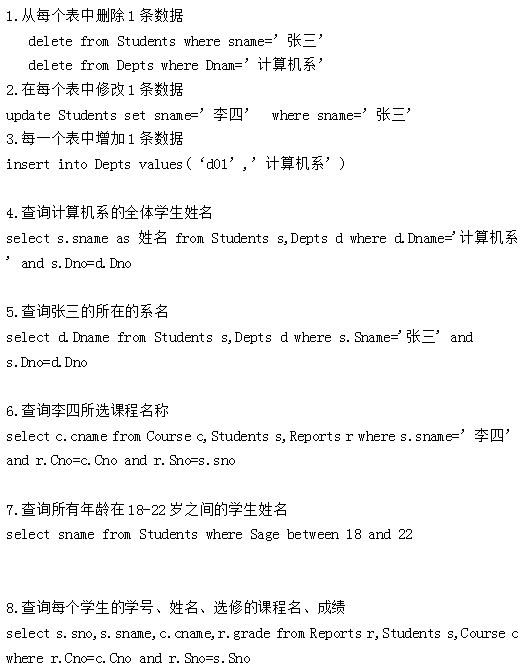




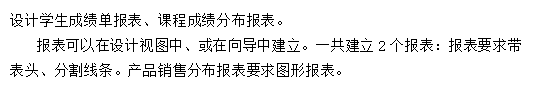




## 程序主要查询实现



## 主要报表、窗体



# x

# 简单的实践资料

## 1、约束

**向表中添加列**

alter table couse add 选课日期 datetime null

**删除表中指定的列**

ALTER TABLE dept DROP COLUMN dp\_count

**数据完整性**

空值约束( NULL )

操作3.1：将student表中的st\_sex列属性更改为NOT NULL

参考答案：

ALTER TABLE student ALTER COLUMN st\_sex nVarChar(8) NOT NULL

空值约束( NULL )

操作3.1：将student表中的st\_sex列属性更改为NOT NULL

参考答案：

ALTER TABLE student ALTER COLUMN st\_sex nVarChar(8) NOT NULL

**3．默认值对象空值约束( NULL )**

操作3.1：将student表中的st\_sex列属性更改为NOT NULL

参考答案：

ALTER TABLE student ALTER COLUMN st\_sex nVarChar(8) NOT NULL

操作3.3：创建默认值对象df\_today为当前日期，并将其绑定到couse表中的‘选课日期’列，然后取消绑定，最后删除默认值对象df\_today。

create default df\_today as getdate() //创建默认对象

go

exec sp\_bindefault df\_today,'couse.选课日期' //绑定到选课日期列

go

exec sp\_unbindefault 'couse.选课日期' //取消绑定

go

drop default df\_today //删除默认对象

Go

1. **检查约束 将couse 表中的 分数列 约束值为>=0或<=100**

alter table couse add check(分数>=0 and 分数<=100)

5.**规则约束对象，将student 表中的性别只允许设置为 男 或 女创建规则约束对象rl\_sex，用于检查性别的取值仅限于“男”和“女”，并将其绑定到student表中的st\_sex列，然后取消绑定，最后删除规则约束对象rl\_sex**

create rule r3\_sex as @学生性别='男' or @学生性别= '女'

go

EXEC sp\_bindrule r3\_sex, 'student.学生性别'

GO

EXEC sp\_unbindrule 'student.学生性别'

GO

DROP RULE r3\_sex

GO

**Student 表中的性别只允许男或女**

alter table student

add constraint r2\_sex check(学生性别 in('男','女'))

## 2、创建数据库

用T-SQL创建一个名为stj的数据库，一个数据文件一个日志文件，数据文件逻辑名为stj\_data，物理文件名分别为stj\_data.mdf，初始大小为5MB；日志文件逻辑名为stj\_log，物理文件名为stj\_log.ldf，初始大小为1MB，最大大小为7MB，增长速度为2MB。

create database stj

on

(

name="stj\_data",

filename="E:\sql2012\stj\_data.dmf",

size=5mb,

filegrowth=1mb

)

log on

(

name="stj\_log",

filename="E:\sql2012\stj\_lod.ldf",

size=1mb,

maxsize=7mb,

filegrowth=2mb

)

use stj

Go

## 3、表操作

stj数据库包括三个关系：

学生(学号,姓名,性别, 年龄, 系别)

课程表(课程号,课程名)

选课表（学号，课程号，成绩）

1、用SQL语句定义上述表的结构

2、为三个表添加主键约束

3、为学生表姓名列添加唯一性约束

4、为选课表添加check约束,成绩输入只能在0-100之间

## 4、插入记录

学生表

CREATE TABLE student (

number CHAR ( 10 ) NOT NULL COMMENT '学号',

`name` CHAR ( 10 ) NOT NULL COMMENT '姓名',

sex bit NOT NULL COMMENT '性别',

age CHAR ( 10 ) NULL COMMENT '年龄',

`subject` CHAR ( 10 ) NULL COMMENT '系别'

) DEFAULT CHARSET=utf8;

课程表

CREATE TABLE courses (

number CHAR ( 10 ) NOT NULL COMMENT '课程号',

`NAME` CHAR ( 10 ) NOT NULL COMMENT '课程名',

`credit` CHAR ( 10 ) NOT NULL COMMENT '学分'

)DEFAULT CHARSET=utf8;

选课表

CREATE TABLE elective (

number CHAR ( 10 ) NOT NULL COMMENT '选课流水',

s\_number CHAR ( 10 ) NOT NULL COMMENT '学号',

c\_number CHAR ( 10 ) NULL COMMENT '课程号',

grade CHAR ( 10 ) NULL COMMENT '成绩' # 这个将要创建在学生表中

)DEFAULT CHARSET=utf8;

添加主键约束

给学生表、课程表和选课表添加主键

alter table student add constraint id primary key(number);

alter table student modify number char(10) not null primary key

alter table elective add constraint ld primary key(number)

添加索引

选课表的c\_number和s\_number要设置索引

alert table elective add index c\_number\_i (c\_number);

create index s\_number\_i on elective (s\_number)

Check 约束

选课表的成绩要在0和100之间

alter table elective add check(grade>=0 and grade<=100)

唯一性约束

学生表的name需唯一

alter table student add unique(`name`)

删除列

删除选课表的成绩（grade）字段

Alter table elective drop grade;

添加列

必须要默认为零，不然添加选课的时候插入触发器无法计算总和

Alter table student add column grade char(10) comment '成绩' default ‘0’

Alter table elective add column grade char(10) comment '成绩'

插入数据

insert into courses

values('1','SQL Serves','4'),

('2','VB','3'),

('3','专业英语','2')

insert into student

values('1','李明',0,'20','计算机系'),

('2','张然',1,'19','经管系'),

('3','许晴',1,'18','电子系')

insert into elective

values('1','1','1','90'),

('2','2','1','85'),

('3','3','2','76'),

('4','1','3','70')

## 修改记录

**把课程表中课程号为2的课程的学分修改为4。**

update 课程表

set 学分='4'

where 课程号='2'

## 查询

1. **查询选修表中选修1课程的学生学号**

select s\_number as number\_id

from elective

where c\_number='1'

1. **列出所有姓“李”且单名的学生**

select \*

from student

where `name` like '李\_'

1. **统计男女学生的人数**

select sex, count(\*) as 'total'

from student

group by sex

1. **查询学生选修课程成绩大于80分的学生情况。**

select s\_number, grade

from elective

where grade >=80

1. **查询选修了课程名为VB”的学生学号和姓名。**

select a.number, a.name

from student a,elective b,courses c

where a.number = b.s\_number

and c.number = b.s\_number

and c.name='VB'

1. **创建视图，检索“选修了SQL Server”课程的学生信息以及成绩。**

CREATE VIEW ccj1 AS SELECT

a.number,

a.NAME,

a.sex,

b.grade

FROM

student a,

elective b,

courses c

WHERE

a.number = b.s\_number

AND c.number = b.c\_number

AND c.NAME = 'SQL Serves'

desc ccj1 查看视图

Select \* from ccj1

1. **计算机系同学的学号，姓名和总学分**

Select 学号 as number,姓名 as name，总学分as mark

From xsb

Where 专业=’计算机’

1. **查找总分大于80分的人数**

select count(\*) as '总分>80分的人数'

from 选课表

where 成绩>80;

1. **分页查询学生数据**

每页显示2条数据，排序为升序

SELECT

\*

FROM

student

ORDER BY

number ASC

LIMIT 0, 2

1. **。**

## 创建存储过程

创建一存储过程getDetailByName，通过输入参数学生姓名（如“张然"），筛选出该学生的基本信息,对不存在此学生姓名的输入值，必须作一检测，打印信息“不存在此学生”。

Create procedure getDetailByName @Name nvarchar(10)

AS

IF (SELECT COUNT(\*) FROM 学生表 WHERE 姓名 = @Name) > 0

SELECT \*

FROM 学生表 Where 姓名 = @Name

ELSE

print '警示 ： 不存在姓名为 ' + @Name + ' 的学生资料'

execute getDetailByName '张然'

## 创建触发器

**语法：**CREATE <触发器名> < BEFORE | AFTER >

<INSERT | UPDATE | DELETE >

ON <表名> FOR EACH Row<触发器主体>

**创建触发器（INSERT,UPDATE触发器也要看）**

**在学生表中创建DELETE触发器，当删除学生表中某一学号时，同时删除选课表中该学生的相关信息。**

CREATE TRIGGER xsb AFTER DELETE ON student FOR EACH ROW

DELETE

FROM

elective

WHERE

s\_number = old.number

#删除触发器

DROP TRIGGER xsb

#删除表数据，测试

DELETE

FROM

student

WHERE

number = '3'

**创建insert 触发器，当向选课表中插入一名学生的成绩时，将表中该学生的总学分加上添加的课程的学分。**

CREATE TRIGGER xkb\_insert AFTER INSERT ON elective FOR EACH ROW

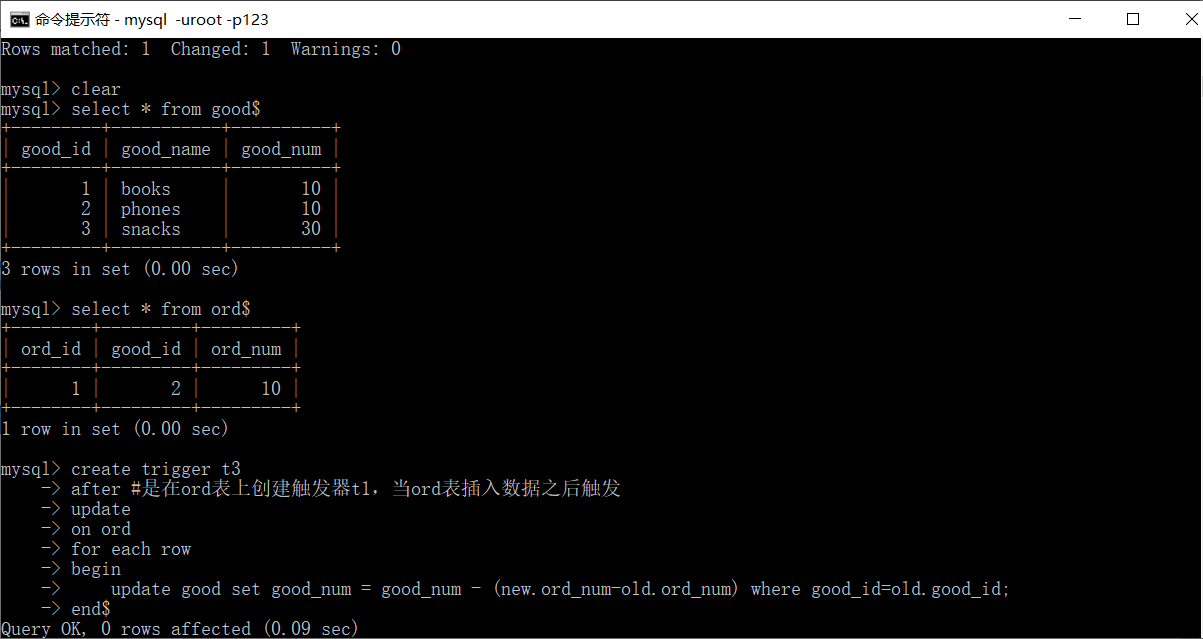
UPDATE student

SET grade = grade + new.grade

WHERE

number = new.s\_number

**更新触发器**



## X

## x