
# 数 据 库 系 统 原 理 实 验 报 告

## 实验四 数据库完整性

姓　　名 李波

学　　号 22920202204570

院　　系 信息学院

专　　业 计算机科学与技术

指导教师 张东站

# 实验 1.1 实体完整性

1） 在数据库 School 中建立表 Stu_Union，进行主键约束，在没有违反实体完整性的前提下插入并更新一条记录

创建表

```
create table Stu_Union(
 sno char(5) not null unique,
 sname char(8),
 ssex char(5),
 sage int,
 sdept char(20),
 constraint  P1  primary key(sno) );
```

插入一条记录

```
insert into Stu_Union values ('0001','Wang','男',20,'CS');
```

结果

| | sno | sname | ssex | sage | sdept |
|---|---|---|---|---|---|
| 1 | 0001 | Wang | 男 | 20 | CS |

更新该记录

```
update Stu_Union SET sno='2' where sdept='CS'
```

| | sno | sname | ssex | sage | sdept |
|---|---|---|---|---|---|
| 1 | 2 | Wang | 男 | 20 | CS |

2） 演示违反实体完整性的插入操作

进行违反实体完整性的插入操作

```
insert into Stu_Union values('2','Li','男',20,'CS')
```

结果

```
消息 2627，级别 14，状态 1，第 17 行
违反了 PRIMARY KEY 约束"P1"。不能在对象"dbo.Stu_Union"中插入重复键。重复键值为 (2    )。
语句已终止。
```

3） 演示违反实体完整性的更新操作

进行违反实体完整性的更新操作

```
update Stu_Union  set sno= null where sno='2'
```

结果

```
消息 515，级别 16，状态 2，第 21 行
不能将值 NULL 插入列 'sno'，表 'School.dbo.Stu_Union'；列不允许有 Null 值。UPDATE 失败。
语句已终止。
```

4）演示事务的处理，包括事务的建立，处理以及出错时的事物回滚

提示：SQL2005 相关语句为

BEGIN TRAN

ROLLBACK TRAN

COMMIT TRAN

可以这样演示：新建一个包含两条语句的事务，使第一条成功而第二条失败，然后查看整个事务是否回滚。

重要提示：SQL 默认只回滚出错的语句，要回滚整个事务，需要预先执行以下语句：

SET XACT_ABORT ON

设置为回滚整个事务

```
set xact_abort on
```

执行事务的第一条语句

```
begin transaction T1
insert into Stu_Union values ('0003','W','男',50,'CS')
select * from  Stu_Union
```

成功插入记录

| | sno | sname | ssex | sage | sdept |
|---|---|---|---|---|---|
| 1 | 0003 | W | 男 | 50 | CS |
| 2 | 2 | Wang | 男 | 20 | CS |

执行事务的第二条语句

```
insert into Stu_Union values ('0003','W','男',50,'CS')
```

执行失败

```
消息 2627，级别 14，状态 1，第 30 行
违反了 PRIMARY KEY 约束"P1"。不能在对象"dbo.Stu_Union"中插入重复键。重复键值为 (0003 )。
```

回滚了整个事务

| | sno | sname | ssex | sage | sdept |
|---|---|---|---|---|---|
| 1 | 2 | Wang | 男 | 20 | CS |

5）通过建立 Scholarship 表，插入一些数据。演示当与现有的数据环境不等时，无法建立实

体完整性以及参照完整性。

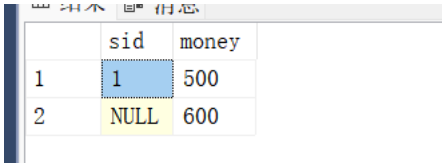
建立表

```
create table Scholarship
(
    sid char(8),
    money int
)
```

插入两条记录

```
insert into Scholarship values ('1',500);
insert into Scholarship values (null,600);
```

结果



| | sid | money |
|---|---|---|
| 1 | 1 | 500 |
| 2 | NULL | 600 |

无法建立实体完整性

```
alter table Scholarship add constraint P2 primary key(sid)
```

```
消息 8111，级别 16，状态 1，第 45 行
无法在表 'Scholarship' 中可为 Null 的列上定义 PRIMARY KEY 约束。
消息 1750，级别 16，状态 0，第 45 行
无法创建约束或索引。请参阅前面的错误。
```

无法建立参照完整性

```
alter table Scholarship add constraint P3 foreign key(sid)
   references Stu_Union(sname)
```

```
消息 1776，级别 16，状态 0，第 46 行
在被引用表 'Stu_Union' 中没有与外键 'P3' 中的引用列列表匹配的主键或候选键。
消息 1750，级别 16，状态 1，第 46 行
无法创建约束或索引。请参阅前面的错误。
```

# 实验 1.2 参照完整性

1） 为演示参照完整性，建立表 Course，令 cno 为其主键，并在 Stu_Union 中插入数据。为

下面的实验步骤做预先准备。

建立表

```
create table Course (
   cno char(5) not null unique,
   cname varchar(20) not null,
   cpoints int,
   constraint P4 primary key (cno));
```

在 Stu_Union 中插入数据

```
insert into Stu_Union values ('10001','王强','女',10,'AA')
```

结果

| | sno | sname | ssex | sage | sdept |
|---|---|---|---|---|---|
| 1 | 10001 | 王强 | 女 | 10 | AA |
| 2 | 2 | Wang | 男 | 20 | CS |

在 Course 中插入数据

```
insert Course values ('0001','computer',2);
insert Course values ('0002','database',3);
```

结果

| | cno | cname | cpoints |
|---|---|---|---|
| 1 | 0001 | computer | 2 |
| 2 | 0002 | database | 3 |

2） 建立表 sc，另 sno 和 cno 分别为参照 Stu_Union 表以及 Course 表的外键，设定为级连删除，并令(sno, cno)为其主键。在不违反参照完整性的前提下，插入数据。

建立表 sc

```
create table sc(
 sno char(5) references Stu_Union (sno) on delete cascade,
 cno char(5) references course(cno) on delete cascade,
 grade int,
 constraint p6 primary key (sno,cno)
 );
```

插入数据

```
insert into sc values ('10001','0001',2);
insert into sc values ('10001','0002',2);
```

结果

| | sno | cno | grade |
|---|---|---|---|
| 1 | 10001 | 0001 | 2 |
| 2 | 10001 | 0002 | 2 |

3） 演示违反参照完整性的插入数据

```
insert into sc values('55','55',99)
```

违反参照完整性，插入失败

```
消息
消息 547，级别 16，状态 0，第 82 行
INSERT 语句与 FOREIGN KEY 约束"FK__sc__sno__74AE54BC"冲突。该冲突发生于数据库"School"，表"dbo.Stu_Union"，column 'sno'。
```

4） 在 Stu_Union 中删除数据，演示级连删除。

删除 Stu_Union 中的数据

```
delete from Stu_Union where sno='10001'
```

Stu_Union 中的数据被删除

| | sno | sname | ssex | sage | sdept |
|---|---|---|---|---|---|
| 1 | 2 | Wang | 男 | 20 | CS |

sc 中的对应数据也被删除

结果 消息

| sno | cno | grade |
|---|---|---|

5） Course 中删除数据，演示级连删除。

```
delete from Course where cno='0001'
```

| | cno | cname | cpoints |
|---|---|---|---|
| 1 | 0002 | database | 3 |

sc 中的对应数据也被删除

结果 消息

| sno | cno | grade |
|---|---|---|

6） 为了演示多重级连删除，建立 Stu_Card 表，令 stu_id 为参照 Stu_Union 表的外键，令 card_id 为其主键，并插入数据。

插入数据

```
insert into Stu_Union values ('0004','00','男',20,'CS');
insert into Stu_Union values ('0002','11','男',20,'CS');
insert into Stu_Union values ('0003','22','男',20,'CS');
insert Course values ('0003','computer',2);
insert Course values ('0002','database',3);
```

结果

| | sno | sname | ssex | sage | sdept |
|---|---|---|---|---|---|
| 1 | 0002 | 11 | 男 | 20 | CS |
| 2 | 0003 | 22 | 男 | 20 | CS |
| 3 | 0004 | 00 | 男 | 20 | CS |
| 4 | 2 | Wang | 男 | 20 | CS |

| | cno | cname | cpoints |
|---|---|---|---|
| 1 | 0002 | database | 3 |
| 2 | 0003 | computer | 2 |

建立表

```
create table Stu_Card(
        card_id char(14),
        stu_id char (5) references stu_union(sno)
        on delete cascade,
        remained_money decimal (10,2),
        constraint P7 primary key (card_id)
)
```

插入数据

```
insert into  Stu_Card values ('05212567','0002',100.25);
insert into  Stu_Card values ('05212222','0003',200.50);
```

结果

| | card_id | stu_id | remained_money |
|---|---|---|---|
| 1 | 05212222 | 0003 | 200.50 |
| 2 | 05212567 | 0002 | 100.25 |

7） 为了演示多重级连删除，建立 ICBC_Card 表，令 stu_card_id 为参照 Stu_Card 表的外键，令 bank_id 为其主键，并插入数据。

建立表

```
create table ICBC_Card(
        bank_id char(20),
        stu_card_id char (14),
        constraint P10 foreign key (stu_card_id)
         references Stu_card(card_id) on delete cascade,
        restored_money decimal (10,2),
        constraint P8 Primary key (bank_id)
)
```

插入数据

```
insert into  ICBC_Card values ('9558844022312','05212567',15000.1);
insert into  ICBC_Card values ('9558844023645','05212222',50000.3);
```

结果

| | bank_id | stu_card_id | restored_money |
|---|---|---|---|
| 1 | 9558844022312 | 05212567 | 15000.10 |
| 2 | 9558844023645 | 05212222 | 50000.30 |

8） 通过删除 students 表中的一条记录，演示三个表的多重级连删除。

删除前数据

| | card_id | stu_id | remained_money |
|---|---|---|---|
| 1 | 05212222 | 0003 | 200.50 |
| 2 | 05212567 | 0002 | 100.25 |

| | bank_id | stu_card_id | restored_money |
|---|---|---|---|
| 1 | 9558844022312 | 05212567 | 15000.10 |
| 2 | 9558844023645 | 05212222 | 50000.30 |

删除 students 表中的一条记录

```
delete from Stu_Union where sno = '0002'
```

删除后数据

| | card_id | stu_id | remained_money |
|---|---|---|---|
| 1 | 05212222 | 0003 | 200.50 |

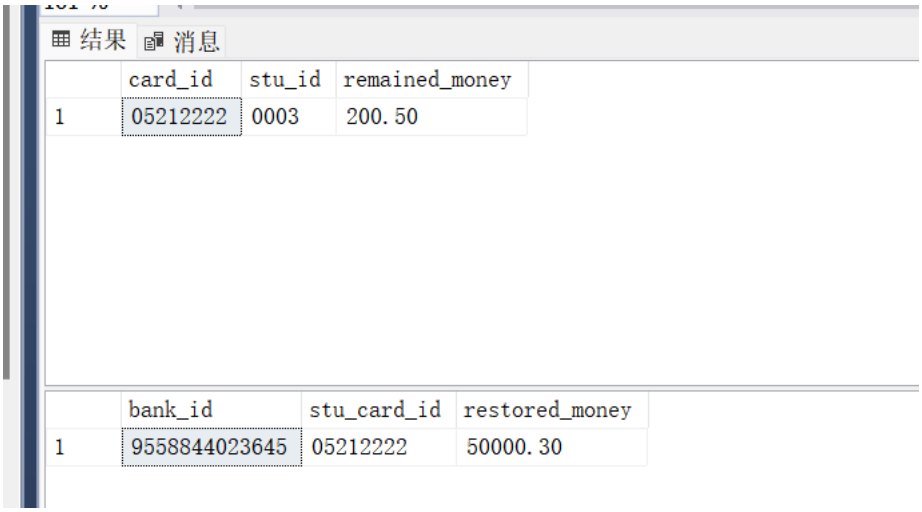
| | bank_id | stu_card_id | restored_money |
|---|---|---|---|
| 1 | 9558844023645 | 05212222 | 50000.30 |

9） 演示事务中进行多重级连删除失败的处理。修改 ICBC_Card 表的外键属性，使其变为 On delete No action，演示事务中通过删除 students 表中的一条记录，多重级连删除失败，整个事务回滚到事务的初始状态。

修改 ICBC_Card 表的外键属性

```
Alter table ICBC_Card
    drop constraint P10;
Alter table ICBC_Card
    add constraint P9 foreign key (stu_card_id)
    references Stu_card(card_id) on delete no action ;
```

删除前数据



结果 消息

| | card_id | stu_id | remained_money |
|---|---|---|---|
| 1 | 05212222 | 0003 | 200.50 |

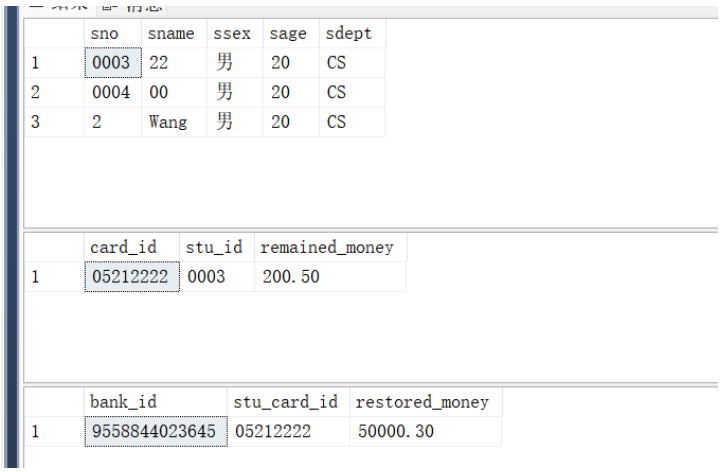
| | bank_id | stu_card_id | restored_money |
|---|---|---|---|
| 1 | 9558844023645 | 05212222 | 50000.30 |

执行事务删除数据

```
Begin Transaction del
delete from stu_union where sno='0003';
Commit Transaction del
```

删除失败

```
消息 547，级别 16，状态 0，第 150 行
DELETE 语句与 REFERENCE 约束"P9"冲突。该冲突发生于数据库"School"，表"dbo.ICBC_Card"，column 'stu_card_id'。
```



| | sno | sname | ssex | sage | sdept |
|---|---|---|---|---|---|
| 1 | 0003 | 22 | 男 | 20 | CS |
| 2 | 0004 | 00 | 男 | 20 | CS |
| 3 | 2 | Wang | 男 | 20 | CS |

| | card_id | stu_id | remained_money |
|---|---|---|---|
| 1 | 05212222 | 0003 | 200.50 |

| | bank_id | stu_card_id | restored_money |
|---|---|---|---|
| 1 | 9558844023645 | 05212222 | 50000.30 |

10）演示互参照问题及其解决方法。建立教师授课和课程指定教师听课关系的两张表，规定一个教师可以授多门课，但是只能去听一门课。为两张表建立相互之间的参照关系，暂时不

考虑听课教师和授课教师是否相同（有余力的同学可以尝试限定听课与授课教师不相同）。
提示：教师授课表以课程号为主键，教师号引用听课表的主键，
　　听课表以教师号为主键，课程号引用授课表的主键。

```
create table listen_course (
    teacher_id char(5),tname varchar(20),course_id char(5)
    constraint P10 primary key(teacher_id)
)
create table teach_course(
    course_id char(5),cname varchar(30),teacher_id char(5)
    constraint P12 primary key(course_id)
    constraint P13 foreign key(teacher_id)
    references listen_course(teacher_id)
)
alter table listen_course
      add constraint P11 foreign key(course_id)
      references teach_course(course_id);
```

# 实验 1.3 触发器的应用

**重要提示：在做以下练习前，先删除 sc 对 stu_union 的外键引用**

1）在表 sc 中演示触发器的 insert 操作，当学生成绩低于 60 分时，自动改为 60，并在事先创建的记录表中插入一条学生成绩低于 60 的记录。
提示：另外创建一个表记录成绩低于 60 分的学生的真实记录。
删除外键引用

```
drop table sc
create table sc(
sno char(5) ,
cno char(5) ,
grade int);
```

插入数据

```
insert into sc values ('0001','0001',2);
insert into sc values ('10001','0001',2);
insert into sc values ('10001','0002',2);
```

创建触发器

```
create trigger T5  on sc
for insert as update sc set grade=60
from sc,inserted  where sc.sno=inserted.sno
    and sc.cno=inserted.sno and inserted.grade<60
```

结果

| | trigger_name | trigger_owner | isupdate | isdelete | isinsert | isafter | isinsteadof | trigger_schema |
|---|---|---|---|---|---|---|---|---|
| 1 | T5 | dbo | 0 | 0 | 1 | 1 | 0 | dbo |

插入数据

```
insert into sc values('02','02',2)
```

结果

| | sno | cno | grade |
|---|---|---|---|
| 1 | 0001 | 0001 | 2 |
| 2 | 10001 | 0001 | 2 |
| 3 | 10001 | 0002 | 2 |
| 4 | 02 | 02 | 60 |

2）在表 stu_union 中创建行级触发器，触发事件是 UPDATE。当更新表 stu_union 的 Sid 时，同时更新 sc 中的选课记录。

提示：这个触发器的作用实际上相当于具有 CASCADE 参数的外键引用。

创建行级触发器

```
create trigger T2 on stu_union
for update as if update(Sno) begin
update sc set sc.sno=i.sno
 from sc,inserted i,deleted d
  where sc.sno=d.sno
 end
```

3）在表 stu_union 中删除一学生的学号(演示触发器的 delete 操作)，使他在 sc 中关的信息同时被删除。

创建触发器使得两个表的信息同时删除

```
create trigger T3 on stu_union
 for delete as
delete sc from sc,deleted d
 where sc.sno=d.sno
```

删除前

结果 消息

| | sno | sname | ssex | sage | sdept |
|---|---|---|---|---|---|
| 1 | 0001 | 11 | 男 | 20 | CS |
| 2 | 0002 | 11 | 男 | 20 | CS |
| 3 | 0003 | 22 | 男 | 20 | CS |
| 4 | 0004 | 00 | 男 | 20 | CS |

| | sno | cno | grade |
|---|---|---|---|
| 1 | 0001 | 0001 | 2 |
| 2 | 10001 | 0001 | 2 |
| 3 | 10001 | 0002 | 2 |
| 4 | 02 | 02 | 60 |

删除数据

```
delete from stu_union where sno='0001'
```

删除后

结果 消息

| | sno | sname | ssex | sage | sdept |
|---|---|---|---|---|---|
| 1 | 0002 | 11 | 男 | 20 | CS |
| 2 | 0003 | 22 | 男 | 20 | CS |
| 3 | 0004 | 00 | 男 | 20 | CS |

| | sno | cno | grade |
|---|---|---|---|
| 1 | 10001 | 0001 | 2 |
| 2 | 10001 | 0002 | 2 |
| 3 | 02 | 02 | 60 |

4）演示触发器删除操作。

提示：SQL2005 创建触发器的语法

CREATE TRIGGER <触发器> ON <表名>

[WITH ENCRYPTION]

FOR {[DELETE][,][INSERT][,][UPDATE]}

[WITH APPEND]

[NOT FOR REPLICATION]

AS <SQL 语句组>

| | trigger_name | trigger_owner | isupdate | isdelete | isinsert | isafter | isinsteadof | trigger_schema |
|---|---|---|---|---|---|---|---|---|
| 1 | T3 | dbo | 0 | 1 | 0 | 1 | 0 | dbo |

```
drop trigger T3
```

结果 消息

| trigger_name | trigger_owner | isupdate | isdelete | isinsert | isafter | isinsteadof | trigger_schema |
|---|---|---|---|---|---|---|---|

# 实验 1.4 索引的建立和作用

1. 实验目的

学会在 SQL SERVER 中建立索引

通过本实验体会覆盖索引的作用，在以后的实践中，能适时地使用覆盖索引来提高数据库的性能。

通过实验体会聚簇索引的优缺点，学会根据具体情况创建聚簇索引

2. 实验内容

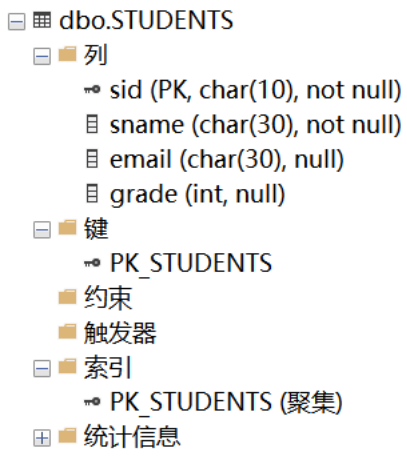
1） STUDENTS(sid,sname,email,grade)在 sname 上建立聚簇索引，grade 上建立非聚簇索引，并分析所遇到的问题

直接使用 create 建立索引报错：

```
create clustered index SN on students(sname)
```

```
消息
消息 1902，级别 16，状态 3，第 213 行
无法对 表 'students' 创建多个聚集索引。请在创建新聚集索引前删除现有的聚集索引 'PK_STUDENTS'。
```

出错原因：表 STUDENTS 已经存在 1 个聚簇索引，而每个基本表最多可有一个聚簇索引，故发生冲突。需要将之前的聚簇索引删除。




使用 drop 语句删除原有索引报错：

```
drop index PK_STUDENTS on students
```

```
消息 3723，级别 16，状态 4，第 214 行
不允许对索引 'students.PK_STUDENTS' 显式地使用 DROP INDEX。该索引正用于 PRIMARY KEY 约束的强制执行。
```

出错原因：表中原来的聚簇索引是因为有主键的存在有系统自动建立。

故需要删除主键约束：

```
alter table students drop constraint PK_STUDENTS
```

```
消息 3725，级别 16，状态 0，第 215 行
约束 'PK_STUDENTS' 正由表 'CHOICES' 的外键约束 'FK_CHOICES_STUDENTS' 引用。
消息 3727，级别 16，状态 0，第 215 行
未能删除约束。请参阅前面的错误信息。
```

出错原因：由于 CHOICES 表存在对 STUDENTS 表的外键引用。

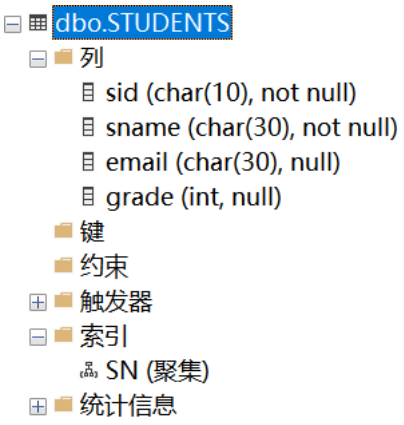
故此需先删除 CHOICES 表对 STUDENTS 表的为外键引用，再删除 STUDENTS 的主键约束，原表中的聚簇索引自动删除。

```
alter table choices drop constraint FK_CHOICES_STUDENTS
```

```
alter table students drop constraint PK_STUDENTS
```

此时在 sname 上建立聚簇索引：

```
create clustered index SN on students(sname)
```




创建 grade 上的非聚簇索引：

```
create index GD on students(grade)
```


2） 数据库 SCHOOL 的选课表 CHOICES 有如下结构：

CHOICES(no，sid，tid，cid，score)

假设选课表集中用于查询分析，经常执行统计某课程修读的学生人数查询访问要求：

A． 首先执行没有索引的实验（设数据库 CHOICES 表在 cid 列上没有索引）

B．然后做有索引的实验

C．对比试验结果，并进行分析

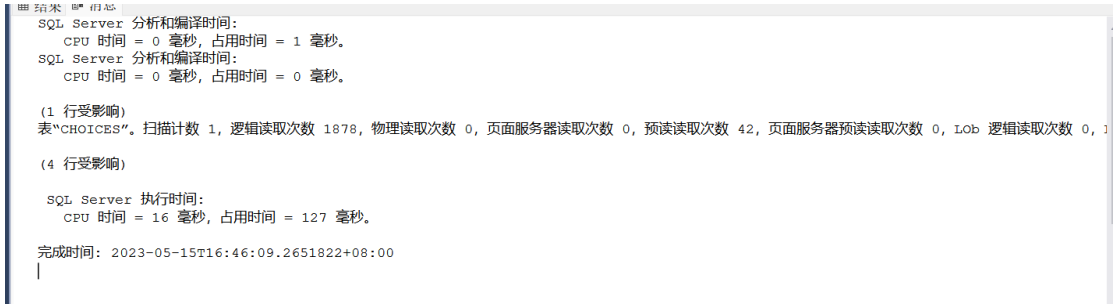
在进行实验前打开以下三个开关，显示语句执行的统计信息：

```
set statistics TIME on;
set statistics IO on;
set statistics profile on;
```

```
SQL Server 执行时间:
  CPU 时间 = 0 毫秒，占用时间 = 0 毫秒。

SQL Server 执行时间:
  CPU 时间 = 0 毫秒，占用时间 = 0 毫秒。

完成时间: 2023-05-15T16:27:49.4837868+08:00
```

**A.**

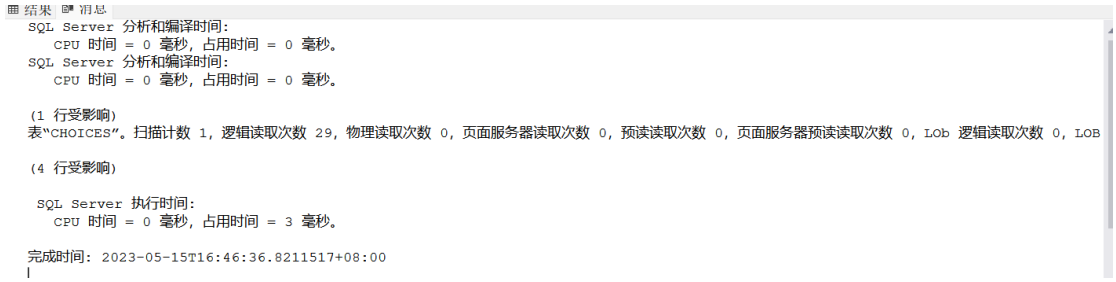
```
select count(*) from choices where cid='10010'
```



```
SQL Server 分析和编译时间:
   CPU 时间 = 0 毫秒，占用时间 = 1 毫秒。
SQL Server 分析和编译时间:
   CPU 时间 = 0 毫秒，占用时间 = 0 毫秒。

(1 行受影响)
表"CHOICES"。扫描计数 1，逻辑读取次数 1878，物理读取次数 0，页面服务器读取次数 0，预读读取次数 42，页面服务器预读读取次数 0，LOb 逻辑读取次数 0，

(4 行受影响)

 SQL Server 执行时间:
   CPU 时间 = 16 毫秒，占用时间 = 127 毫秒。

完成时间: 2023-05-15T16:46:09.2651822+08:00
```

**B.**

```
create index CID on choices(cid)
select count(*) from choices where cid='10010'
```



```
SQL Server 分析和编译时间:
   CPU 时间 = 0 毫秒，占用时间 = 0 毫秒。
SQL Server 分析和编译时间:
   CPU 时间 = 0 毫秒，占用时间 = 0 毫秒。

(1 行受影响)
表"CHOICES"。扫描计数 1，逻辑读取次数 29，物理读取次数 0，页面服务器读取次数 0，预读读取次数 0，页面服务器预读读取次数 0，LOb 逻辑读取次数 0，LOB

(4 行受影响)

 SQL Server 执行时间:
   CPU 时间 = 0 毫秒，占用时间 = 3 毫秒。

完成时间: 2023-05-15T16:46:36.8211517+08:00
```

**C.**

通过比较上述查询语句的 SQL server 执行时间的数据可以发现，建立索引后查询语句的 CPU 时间和占用时间明显小于没有索引时。由此可见，建立索引能够快速定位要查询的内容，有效加快查询速度。

3） 以数据库 SCHOOL 中 CHOICES 表为例，设建表时考虑到以后经常有一个用 sid 查询此学生所有选课信息的查询，考虑到一般学生不止选一门课，且要询问这些记录的所有信息，故在 sid 上建立索引，使相同 sid 的记录存在一起，取数据页面时能一起取出来，减少数据页面的存取次数

要求：

A． 首先执行没有任何索引的情况

B．在 sid 上建有非聚簇索引的情况

C．在 sid 上建有聚簇索引的情况

D． 对比实验结果，并进行分析

**A.**

```
select count(*) from choices where sid = '823069829'
```

结果 消息

```
SQL Server 分析和编译时间:
   CPU 时间 = 0 毫秒，占用时间 = 4 毫秒。
SQL Server 分析和编译时间:
   CPU 时间 = 0 毫秒，占用时间 = 0 毫秒。

(1 行受影响)
表"CHOICES"。扫描计数 1，逻辑读取次数 2482，物理读取次数 0，页面服务器读取次数 0，预读读取次数 0，页面服务器预读读取次数 0，LOb 逻辑读取次数 0，L

(4 行受影响)

 SQL Server 执行时间:
   CPU 时间 = 46 毫秒，占用时间 = 104 毫秒。

完成时间: 2023-05-15T16:40:17.4271073+08:00
```

**B.**

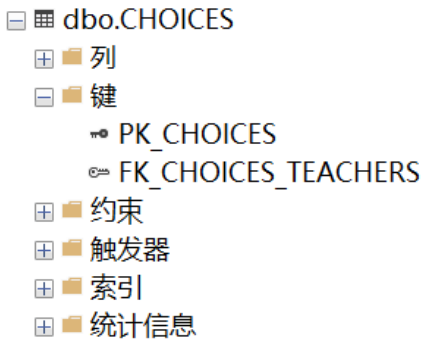
```
create index SID on choices(sid)
select count(*) from choices where sid = '823069829'
```

结果 消息

```
SQL Server 分析和编译时间:
   CPU 时间 = 0 毫秒，占用时间 = 0 毫秒。
SQL Server 分析和编译时间:
   CPU 时间 = 0 毫秒，占用时间 = 0 毫秒。

(1 行受影响)
表"CHOICES"。扫描计数 1，逻辑读取次数 3，物理读取次数 0，页面服务器读取次数 0，预读读取次数 0，页面服务器预读读取次数 0，LOb 逻辑读取次数 0，LOB

(4 行受影响)

 SQL Server 执行时间:
   CPU 时间 = 0 毫秒，占用时间 = 0 毫秒。

完成时间: 2023-05-15T16:44:13.3742753+08:00
```

**C.**

```
drop index SID on choices;
```




```
alter table choices drop constraint PK_CHOICES;
create clustered index SID on choices(sid)
select count(*) from choices where sid = '823069829'
```

结果 消息

```
SQL Server 分析和编译时间:
   CPU 时间 = 0 毫秒，占用时间 = 0 毫秒。
SQL Server 分析和编译时间:
   CPU 时间 = 0 毫秒，占用时间 = 0 毫秒。

(1 行受影响)
表"CHOICES"。扫描计数 1，逻辑读取次数 3，物理读取次数 0，页面服务器读取次数 0，预读读取次数 0，页面服务器预读读取次数 0，LOb 逻辑读取次数 0，LOB

(4 行受影响)

 SQL Server 执行时间:
   CPU 时间 = 0 毫秒，占用时间 = 0 毫秒。

完成时间: 2023-05-15T16:41:58.3208848+08:00
```

**D.**

根据实验数据中的 SQL Server 执行时间比较可知，建立索引的情况下查询语句的执行速度更快，同时建立聚簇索引的执行速度比建立非聚簇索引的执行速度更快。

# 实验总结

本次实验主要涉及到实体完整性，参照完整性，触发器和索引。通过多个表的建立和表与表之间关系的建立，我对实体完整性和参照完整性的理解更加深刻，对二者的建立条件更加了解，同时我也更深刻地体会到了触发器和索引的作用，熟悉了二者的建立。此外，我还学会了如何使用演示事务等操作。

本次实验相比之前难度较大，主要在于多个表之间的联系比较紧密，关系容易搞乱。所以，操作时需要进行大量的思考，否则可能会因误操作导致多个表的数据丢失和错乱，而发生错误就要把这部分重来，比较浪费时间。第四部分的实验比较困难，查阅资料花费了不少时间。

总的来说这次的实验任务量较大，涉及面广泛，难度较大，但很大程度上加深了我对理论知识的记忆与理解，也大大增强了我的实践能力。