22920192204097 吴雨娟 数据库实验

实验四 数据库完整性、触发器、索引

实验目的：熟悉 SQL SERVER 2019 的数据库完整性、触发器、索引机制

实验环境：WINDOWS10、SQL SERVER 2019

实验内容：

4.1 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。

```
create table Stu_Union(
    Sno char(10) primary key,
    Sname char(20),
    Ssex char(2),
    Sage int,
    Sdept char(2)
);
insert into Stu_Union values('95000','张明','男',22,'CS');
select * from Stu_Union;
```

插入记录后：

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	95000	张明	男	22	CS

```
update Stu_Union set Sage = 28 where Sno = '95000';
select * from Stu_Union;
```

更新记录后：

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	95000	张明	男	28	CS

2）演示违反实体完整性的插入操作。

①插入学号为空的信息：

```
insert into Stu_Union values(NULL,'王亮','男',24,'IS');
```

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

②插入学号重复的信息：

```
insert into Stu_Union values('95000','王亮','男',24,'IS');
```

```
消息
消息 2627，级别 14，状态 1，第 17 行
违反了 PRIMARY KEY 约束"PK__Stu_Unio__CA1FE46445AA5A10"。不能在对象"dbo.Stu_Union"中插入重复键。重复键值为 (95000
语句已终止。
```

3）演示违反实体完整性的更新操作。

将学号修改为重复值，会因学号重复导致违反实体完整性：

```
insert into Stu_Union values('95003','王亮','男',24,'IS');
update Stu_Union set Sno = '95000' where Sname = '王亮';
```

```
消息
消息 2627，级别 14，状态 1，第 21 行
违反了 PRIMARY KEY 约束"PK__Stu_Unio__CA1FE46445AA5A10"。不能在对象"dbo.Stu_Union"中插入重复键。重复键值为 (95000
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务回滚。

```
set xact_abort on;
begin transaction STU;
insert into Stu_Union values('95005','刘丽','女',21,'MA');
select * from Stu_Union;
insert into Stu_Union values('95005','程流','男',23,'CS');
commit transaction STU;
select * from Stu_Union;
```

执行第一次插入语句后，查询结果如下：

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	95000	张明	男	28	CS
2	95003	王亮	男	24	IS
3	95005	刘丽	女	21	MA

第二条插入语句违反了主键约束，所以插入失败：

```
消息
消息 2627，级别 14，状态 1，第 28 行
违反了 PRIMARY KEY 约束"PK__Stu_Unio__CA1FE46445AA5A10"。不能在对象"dbo.Stu_Union"中插入重复键。重复键值为 (95005
```

再次查询，发现第一条插入的数据已经消失了，说明出错时事物进行了回滚。

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	95000	张明	男	28	CS
2	95003	王亮	男	24	IS

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。

①建立 Scholarship 表，插入一些数据

```
create table Scholarship(
    No char(10),
    Sno char(10),
    Smoney int
);
insert into Scholarship values('001','95000',8000);
insert into Scholarship values('002','95002',5000);
insert into Scholarship values('002','95005',3000);
select * from Scholarship;
```

结果 消息

	No	Sno	Smoney
1	001	95000	8000
2	002	95002	5000
3	002	95005	3000

②尝试建立实体完整性

i）尝试定义 No 为主键

```
alter table Scholarship add constraint PK_Scholarship primary key(No);
```

消息

```
消息 8111，级别 16，状态 1，第 43 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 43 行
无法创建约束或索引。请参阅前面的错误。
```

因为 No 可以为空，所以不能创建主键约束。

ii）增加 No 非空的约束后，由于 No 存在重复值，所以仍然不能创建主键约束。

消息

```
消息 1505，级别 16，状态 1，第 44 行
因为发现对象名称 'dbo.Scholarship' 和索引名称 'PK_Scholarship' 有重复的键，所以 CREATE
消息 1750，级别 16，状态 1，第 44 行
无法创建约束或索引。请参阅前面的错误。
```

③尝试建立参照完整性

```
select * from Stu_Union;
```

查看 Stu_Union 表中的信息：

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	95000	张明	男	28	CS
2	95003	王亮	男	24	IS

```
select * from Scholarship;
```

查看 Scholarship 表中的信息：

结果 消息

	No	Sno	Smoney
1	001	95000	8000
2	002	95002	5000
3	002	95005	3000

尝试建立外键 Sno：

```
alter table Scholarship add
    constraint FK_Scholarship foreign key(Sno) references Stu_Union(Sno);
```

消息

消息 547，级别 16，状态 0，第 46 行
ALTER TABLE 语句与 FOREIGN KEY 约束"FK_Scholarship"冲突。该冲突发生于数据库"School"，表"dbo.Stu_Union"，column 'Sno'。

由于 Scholarship 中的 Sno 存在 Stu_Union 的 Sno 中不存在的值，所以无法建立参照完整性。

4.2 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 Course，令 cno 为其主键：

```
create table Course(
    Cno char(5) primary key,
    Cname char(20),
    Ccredit int
);
```

在 Stu_Union 中插入数据：

```
insert into Stu_Union values('95006','刘晨','男',21,'IS');
```

```
insert into Stu_Union values('95007','王敏','女',22,'CS');
insert into Stu_Union values('95008','张立','男',23,'MA');
select * from Stu_Union;
```

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	95000	张明	男	28	CS
2	95003	王亮	男	24	IS
3	95006	刘晨	男	21	IS
4	95007	王敏	女	22	CS
5	95008	张立	男	23	MA

在 Course 中插入数据：

```
insert into Course values('10001','Math',5);
insert into Course values('10002','DataBase',4);
select * from Course;
```

结果 消息

	Cno	Cname	Ccredit
1	10001	Math	5
2	10002	DataBase	4

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。

```
create table SC(
    Sno char(10) foreign key references Stu_Union(Sno) on delete
cascade,
    Cno char(5) foreign key references Course(Cno) on delete cascade,
    Grade int,
    primary key(Sno,Cno)
);

insert into SC values('95000','10001',91);
insert into SC values('95000','10002',81);
insert into SC values('95003','10002',89);
insert into SC values('95006','10001',78);
select * from SC;
```

结果 消息

	Sno	Cno	Grade
1	95000	10001	91
2	95000	10002	81
3	95003	10002	89
4	95006	10001	78

3）演示违反参照完整性的插入数据。

插入的 Cno 值在 Course 中不存在：

```
insert into SC values('95006','10003',87);
```

消息

消息 547，级别 16，状态 0，第 80 行
INSERT 语句与 FOREIGN KEY 约束"FK__SC__Cno__1EA48E88"冲突。该冲突发生于数据库"School"，表"dbo.Course"，column 'Cno'。

4）在 Stu_Union 中删除数据，演示级连删除。

```
delete from Stu_Union where Sno = '95000';
select * from SC;
```

结果 消息

	Sno	Cno	Grade
1	95003	10002	89
2	95006	10001	78

SC 表中关于学号为‘95000’的学生的信息也被删除了。

5）Course 中删除数据，演示级连删除。

```
delete from Course where Cno = '10001';
select * from SC;
```

结果 消息

	Sno	Cno	Grade
1	95003	10002	89

SC 表中关于课程号为‘10001’的信息也被删除了。

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。

```sql
create table Stu_Card(
    card_id char(5) primary key,
    stu_id char(10) foreign key references Stu_Union(Sno) on delete
cascade
);
insert into Stu_Card values('55001','95003');
insert into Stu_Card values('55002','95006');
insert into Stu_Card values('55003','95007');
insert into Stu_Card values('55004','95008');
select * from Stu_Card;
```

结果 消息

	card_id	stu_id
1	55001	95003
2	55002	95006
3	55003	95007
4	55004	95008

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。

```sql
create table ICBC_Card(
    bank_id char(5) primary key,
    stu_card_id char(5) foreign key references Stu_card(card_id) on
delete cascade
);
insert into ICBC_Card values('75001','55001');
insert into ICBC_Card values('75002','55002');
insert into ICBC_Card values('75003','55003');
select * from ICBC_Card;
```

结果 消息

	bank_id	stu_card_id
1	75001	55001
2	75002	55002
3	75003	55003

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

```sql
delete from Stu_Union where Sno = '95006';
select * from Stu_Card;
select * from ICBC_Card;
```

结果 消息

	card_id	stu_id
1	55001	95003
2	55003	95007
3	55004	95008

	bank_id	stu_card_id
1	75001	55001
2	75003	55003

删除 Stu_Union 表中的一条记录后，Stu_Card 表和 ICBC_Card 表中的相关记录也被删除。

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 Stu_Union 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。

①修改 ICBC_Card 表的外键属性，使其变为 On delete No action

```
alter table ICBC_Card drop
    constraint FK__ICBC_Card__stu_c__245D67DE;
alter table ICBC_Card add
    constraint FK_ICBC_Card foreign key(stu_card_id)
        references Stu_card(card_id) on delete no action;
```

②演示事务中进行多重级联删除失败的处理

```
begin transaction Del;
delete from Stu_Union where Sno = '95003';
commit transaction Del;
```

消息

消息 547，级别 16，状态 0，第 124 行
DELETE 语句与 REFERENCE 约束"FK_ICBC_Card"冲突。该冲突发生于数据库"School"，表"dbo.ICBC_Card"，column 'stu_car

结果 消息

	card_id	stu_id
1	55001	95003
2	55003	95007
3	55004	95008

	bank_id	stu_card_id
1	75001	55001
2	75003	55003

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。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。

①演示互参照问题（错误建立教师授课表和听课表）

```
create table Listen(
    Tid char(5) primary key,
    Tname varchar(10),
    Cid char(5) references Teach(Cid)
);

create Table Teach(
    Cid char(5) primary key,
    Cname varchar(10),
    Tid char(5) references Listen(Tid)
);
```

消息

```
消息 1767，级别 16，状态 0，第 128 行
外键 'FK__Listen__Cid__282DF8C2' 引用了无效的表 'Teach'。
消息 1750，级别 16，状态 1，第 128 行
无法创建约束或索引。请参阅前面的错误。
```

②解决方法（正确建立教师授课表和听课表）

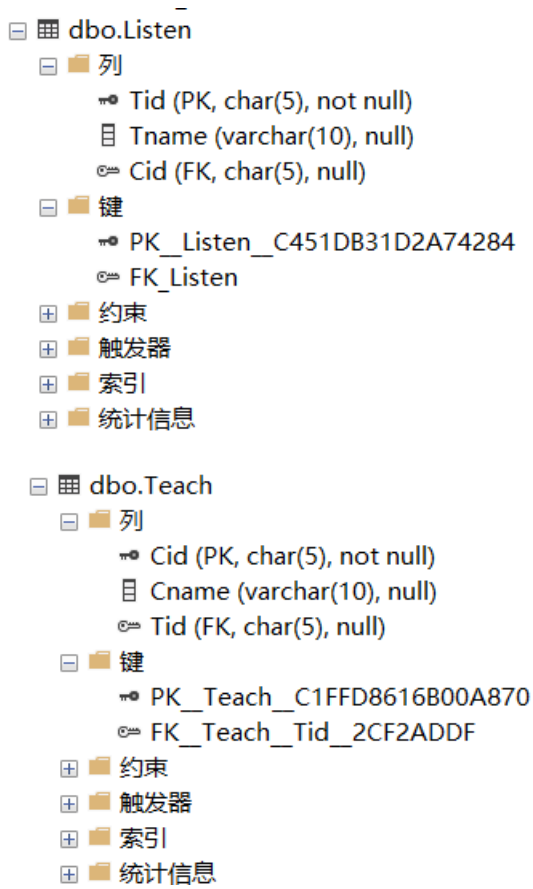
在 Listen 表中，无法引用尚未创建的表 Teach，所以将这个外键放到表 Teach 创建后再建立。

```
create table Listen(
```

```
    Tid char(5) primary key,
    Tname varchar(10),
    Cid char(5)
);

create Table Teach(
    Cid char(5) primary key,
    Cname varchar(10),
    Tid char(5) references Listen(Tid)
);
alter table Listen add
    constraint FK_Listen foreign key(Cid) references Teach(Cid);
```

结果如下；



4.3 触发器的应用

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。

①删除 SC 对 Stu_Union 的外键引用

```
alter table SC drop constraint FK__SC__Sno__1DB06A4F;
```

②创建真实记录表 RealSC

```
create table RealSC(
    Sno char(10),
    Cno char(5) foreign key references Course(Cno) on delete cascade,
    Grade int,
    primary key(Sno,Cno)
);
```

③创建触发器 T1 和 T2

```
create trigger T1 on SC for insert,update as update SC
set Grade = 60 from SC,inserted
where SC.Sno=inserted.Sno and SC.Cno=inserted.Cno and
inserted.Grade<60;

create trigger T2 on SC for insert as insert into RealSC
select Sno,Cno,Grade from inserted;
```

④演示触发器的 insert 操作

插入一条成绩低于 60 分的记录：

```
insert into SC values('95005','10002',58);
select * from SC;
select * from RealSC;
```

结果如下：

结果 消息

	Sno	Cno	Grade
1	95003	10002	89
2	95005	10002	60

	Sno	Cno	Grade
1	95005	10002	58

在 SC 表中成绩被修改为 60，但是在真实记录表 RealSC 中，成绩为真实插入的 58。

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。

①创建触发器 T3：

```
create trigger T3 on Stu_Union for update as
if update(Sno)
begin
    update SC
    set SC.Sno=inserted.Sno
    from SC,inserted,deleted
    where SC.Sno=deleted.Sno
end;
```

②插入选课记录和学生信息：

```
insert into Stu_Union values('95001','李值','女',23,'IS');
insert into SC values('95001','10002',92);
select * from Stu_Union;
select * from SC;
```

此时表中信息如下：

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	95001	李值	女	23	IS
2	95003	王亮	男	24	IS
3	95007	王敏	女	22	CS
4	95008	张立	男	23	MA

	Sno	Cno	Grade
1	95001	10002	92
2	95003	10002	89
3	95005	10002	60

③修改 Stu_Union 表中的学号：

```
update Stu_Union set Sno='95000' where Sname='李值';
select * from Stu_Union;
select * from SC;
```

结果如下：

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	95000	李值	女	23	IS
2	95003	王亮	男	24	IS
3	95007	王敏	女	22	CS
4	95008	张立	男	23	MA

	Sno	Cno	Grade
1	95000	10002	92
2	95003	10002	89
3	95005	10002	60

在触发器的作用下，SC 表中的学号也进行了相应的修改。

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。

```
create trigger T4 on Stu_Union for delete as
delete SC from SC,deleted
where SC.Sno=deleted.Sno;

delete from Stu_Union where Sno='95000';
select * from Stu_Union;
select * from SC;
```

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	95003	王亮	男	24	IS
2	95007	王敏	女	22	CS
3	95008	张立	男	23	MA

	Sno	Cno	Grade
1	95003	10002	89
2	95005	10002	60

4）演示删除触发器操作。

删除之前查看触发器信息：

```
Sp_helptrigger Stu_Union;
```

结果 消息

	trigger_name	trigger_owner	isupdate	isdelete	isinsert	isafter	isinsteadof	trigger_schema
1	T3	dbo	1	0	0	1	0	dbo
2	T4	dbo	0	1	0	1	0	dbo

删除触发器：

```
drop trigger T3;
Sp_helptrigger Stu_Union;
```

结果 消息

	trigger_name	trigger_owner	isupdate	isdelete	isinsert	isafter	isinsteadof	trigger_schema
1	T4	dbo	0	1	0	1	0	dbo

可以看到触发器 T3 已被删除。

4.4 索引的建立和作用

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

①在 sname 上建立聚簇索引

```
create clustered index Stusname on STUDENTS(sname);
```

消息

消息 1902，级别 16，状态 3，第 212 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'。

分析遇到的问题：

因为在创建表时，会默认为主键建立聚簇索引，所以要建立其他聚簇索引前，要将其删除，否则会报错。

②在 grade 上建立非聚簇索引

```
create index Stugrade on STUDENTS(grade);
```

- dbo.STUDENTS
  - 列
  - 键
  - 约束
  - 触发器
  - 索引
    - PK_STUDENTS (聚集)
    - Stugrade (不唯一，非聚集)
  - 统计信息

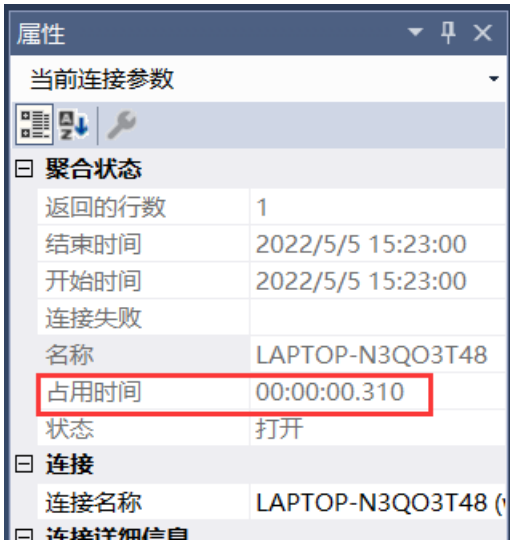
2） 数据库 SCHOOL 的选课表 CHOICES 有如下结构：

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）

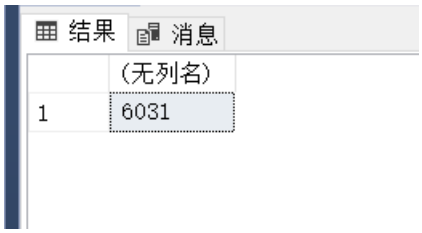
```
select COUNT(sid) from CHOICES where cid in(
select cid from COURSES where cname = 'c++');
```

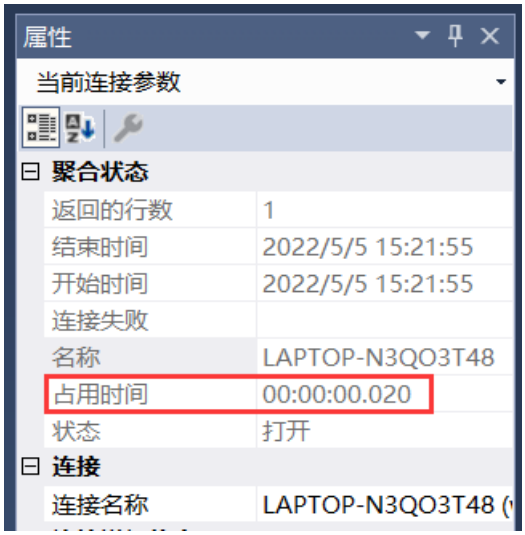


B. 然后做有索引的实验

建立索引并查询如下：

```
create index Chocid on CHOICES(cid);
select COUNT(sid) from CHOICES where cid in(
select cid from COURSES where cname = 'c++');
```





C. 对比试验结果，并进行分析

从结果图可以看到，没有索引的查询时间是 310ms，建立索引后的查询时间是 20ms。对比两个查询时间，可知建立索引可以加快查询速度。

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

```
select * from CHOICES where sid = '847061074';
```

结果 消息

	no	sid	tid	cid	score
1	500001270	847061074	292043491	10025	92
2	508884720	847061074	209134308	10031	NULL
3	571652576	847061074	233258310	10045	93
4	589635509	847061074	297427652	10024	98

查询已成功执行。 LAPTOP-N3QO3T48 (15.0 RTM) wyj (63) School 00:00:00 4 行

属性

当前连接参数

聚合状态

返回的行数	4
结束时间	2022/5/5 15:18:47
开始时间	2022/5/5 15:18:47
连接失败	
名称	LAPTOP-N3QO3T48
占用时间	00:00:00.068
状态	打开

连接

连接名称	LAPTOP-N3QO3T48 (

B. 在 sid 上建有非聚簇索引的情况

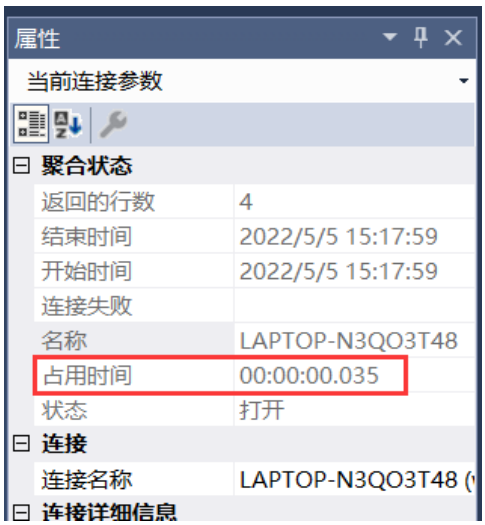
```
create index Chosid on CHOICES(sid);
select * from CHOICES where sid = '847061074';
```

结果如下：

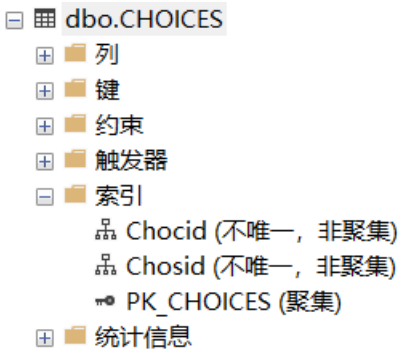
结果 消息

	no	sid	tid	cid	score
1	500001270	847061074	292043491	10025	92
2	508884720	847061074	209134308	10031	NULL
3	571652576	847061074	233258310	10045	93
4	589635509	847061074	297427652	10024	98

查询已成功执行。 LAPTOP-N3QO3T48 (15.0 RTM) wyj (63) School 00:00:00 4 行



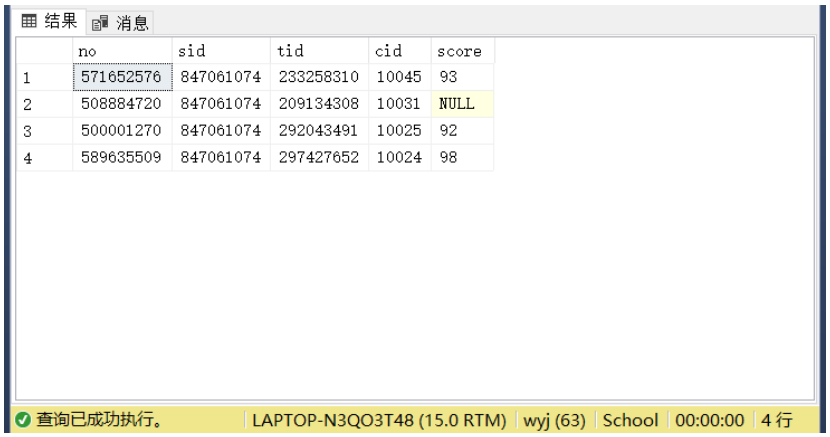
索引如下：



C. 在 sid 上建有聚簇索引的情况

```
drop index CHOICES.Chosid;
alter table CHOICES drop constraint PK_CHOICES;
create clustered index Chosid on CHOICES(sid);
alter table CHOICES add constraint PK_CHOICES primary key(no);
select * from CHOICES where sid = '847061074';
```

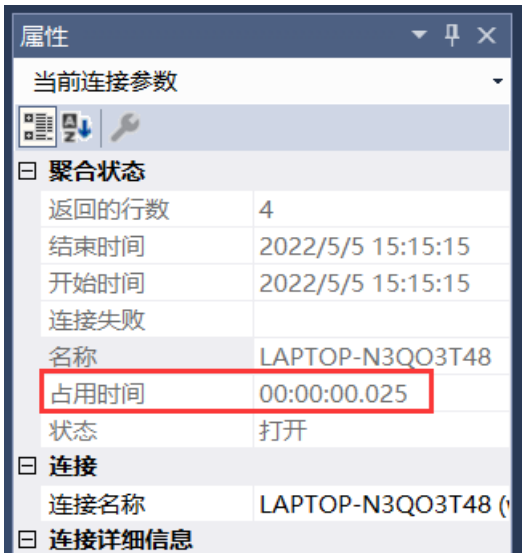
结果如下：



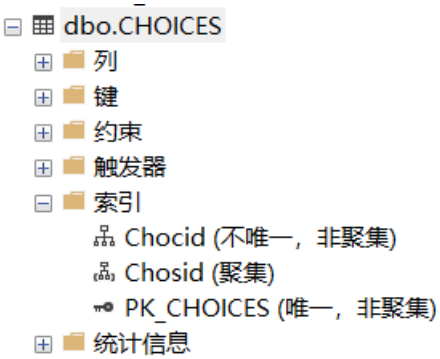
结果 消息

	no	sid	tid	cid	score
1	571652576	847061074	233258310	10045	93
2	508884720	847061074	209134308	10031	NULL
3	500001270	847061074	292043491	10025	92
4	589635509	847061074	297427652	10024	98

查询已成功执行。 LAPTOP-N3QO3T48 (15.0 RTM) wyj (63) School 00:00:00 4 行



索引如下：



D. 对比实验结果，并进行分析

从结果图可以看到，在没有任何索引的情况下，查询时间是 68ms；在 sid 上建有非聚簇索引的情况下，查询时间是 35ms；在 sid 上建有聚簇索引的情况下，查询时间是 25ms。所以建立索引后的查询比不建立索引的查询速度快。

实验总结：

通过本次实验，熟悉了数据库完整性、触发器、索引的相关操作。在数据库完整性的操作中，实验了一些不合法的操作，也警醒我们在有级联关系的表中操作时，要注意操作的合法性。触发器的使用可以很好地处理一些事件，当事件发生时会执行相应操作，是一个很实用也很强大的功能。索引的建立可以帮助我们提升查询速度，从而提升数据库性能。