验证性实验

（1）用 ROLLBACK TRAN 实现事务的回滚操作

USE LMS

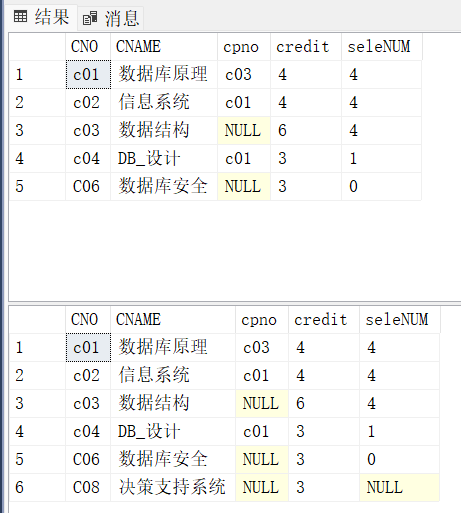
GO

BEGIN TRAN

SELECT \* FROM C

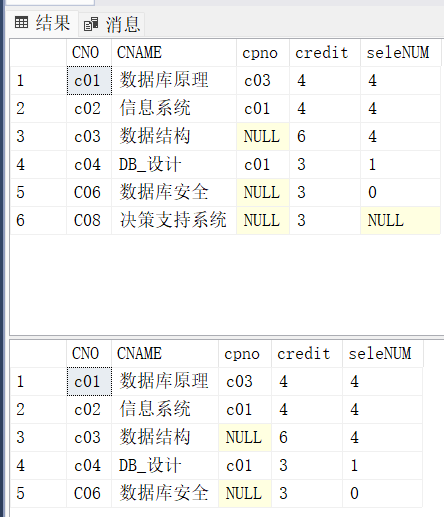
INSERT INTO C VALUES('C08', '决策支持系统', NULL, 3,NULL)

SELECT \* FROM C



ROLLBACK TRAN

SELECT \* FROM C



观察每次显示C表中数据记录条数变化，为什么有这种变化？

（2）事务回滚点的保存

USE LMS

GO

BEGIN TRAN Mytran

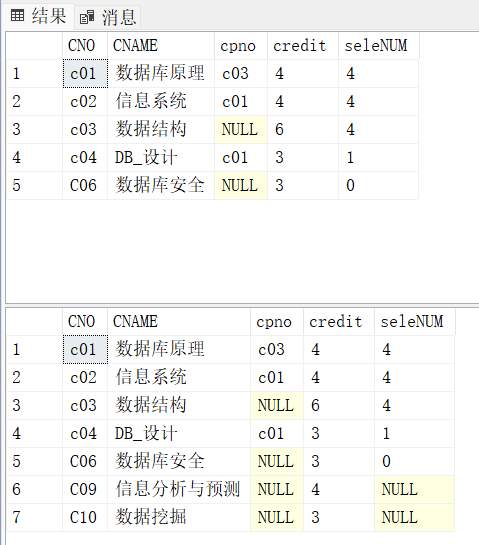
SELECT \* FROM C

INSERT INTO C VALUES('C09', '信息分析与预测', NULL,4,NULL)

SAVE TRAN POINT1

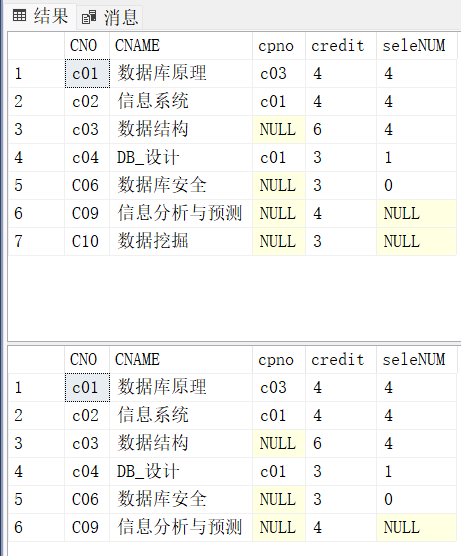
INSERT INTO C VALUES('C10', '数据挖掘', NULL, 3,NULL)

SELECT \* FROM C



ROLLBACK TRAN POINT1

SELECT \* FROM C



观察最后的C表中的数据结果与前面C表显示结果的不同。

（3）触发器的建立

① 在S表建立建立如下触发器

CREATE TRIGGER TRIGONS

ON S FOR UPDATE

AS

SELECT \* FROM DELETED

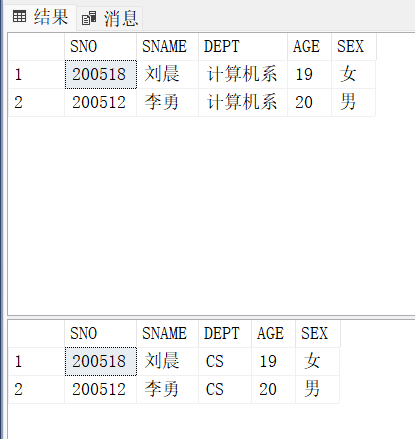
SELECT \* FROM INSERTED

选中上述触发器创建语句并执行，然后输入下面语句：

UPDATE S

SET DEPT ='CS' WHERE DEPT='计算机系'

SELECT \* FROM S



② 在S表上建立下面触发器：

CREATE TRIGGER TEMPTABLE

ON S FOR INSERT

AS

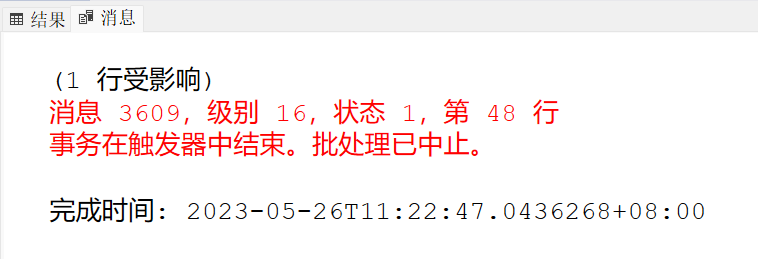
SELECT \* FROM INSERTED

ROLLBACK TRAN

执行上述触发器后在S表中插入数据，

INSERT INTO S(SNO,SNAME) VALUES(‘10012’,’ZXX’)

执行插入语句后打开S观察结果。



上面的代码创建了一个名为 "TEMPTABLE" 的触发器，它会在向 S 表插入新记录时触发。在触发器内部，使用 SELECT 语句从 INSERTED 表中获取插入的数据，并使用 ROLLBACK TRAN 语句回滚事务，使插入操作无效。

这个触发器的作用是，在插入新记录时，立即回滚事务，从而阻止数据的实际插入。

③ 在C表中加入一列seleNum表示选课数量，在SC表上插入一条选课记录后，将C表中

对应的课程的seleNum值加1。

CREATE TRIGGER SELECTCOURSE

ON SC FOR INSERT

AS

IF (SELECT COUNT(\*) FROM INSERTED)>1

BEGIN

PRINT 'YOU CAN INSERT ONE RECORD ONCE. SO THE RECORDS ARE NOT INSERTED

INTO THE TABLE'

ROLLBACK TRAN

END

ELSE

BEGIN

DECLARE @CNO VARCHAR(10)

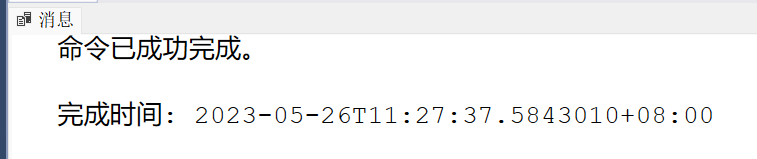
SELECT @CNO=CNO FROM INSERTED

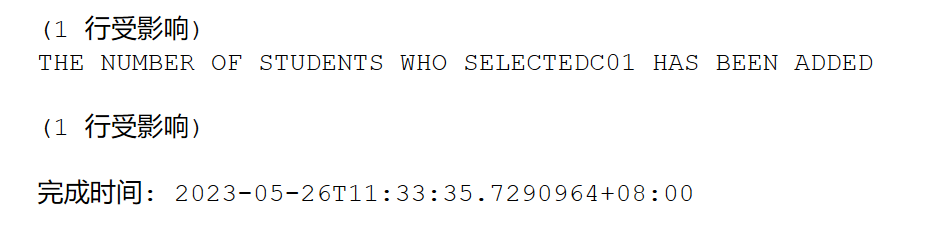
UPDATE C

SET seleNum=seleNum+1 WHERE CNO=@CNO

PRINT 'THE NUMBER OF STUDENTS WHO SELECTED'+@CNO +' HAS BEEN ADDED'

END

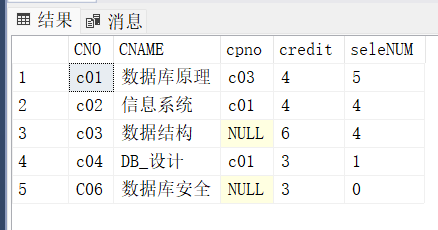




验证触发器的有效性：

INSERT INTO SC (SNO, CNO, SCORE)

VALUES ('200512', 'C01', 80)



（4）触发器的修改、删除

① 修改触发器

CREATE TRIGGER DELS

ON S FOR DELETE

AS

DECLARE @DEPT VARCHAR(20)

SELECT @DEPT=DEPT FROM INSERTED

IF @DEPT='计算机系'

BEGIN

PRINT 'YOU CAN NOT DELETE THE STUDENT FROM CS'

ROLLBACK

END

select \* from S

DELETE FROM S WHERE SNO = '200512'



执行上面的触发器，发现错误，则修改触发器。

ALTER TRIGGER DELS

ON S FOR DELETE

AS

DECLARE @DEPT VARCHAR(20)

SELECT @DEPT=DEPT FROM DELETED

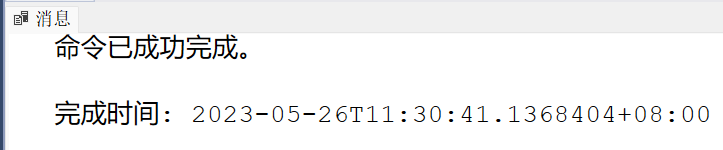
IF @DEPT=‘CS’

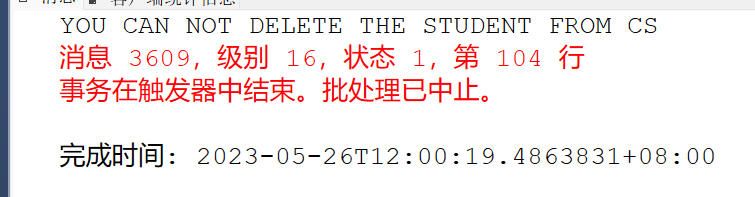
BEGIN

PRINT ‘YOU CAN NOT DELETE THE STUDENT FROM CS’

ROLLBACK

END

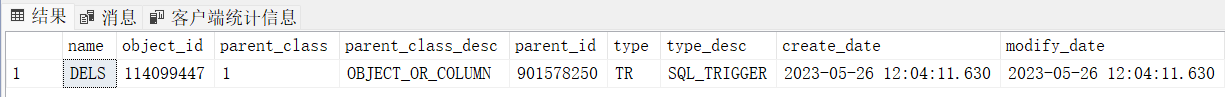


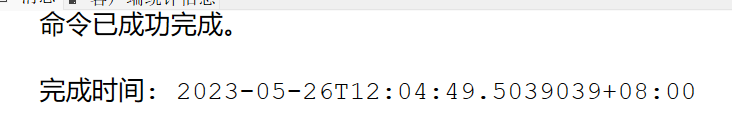


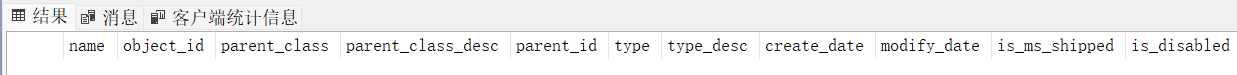
② 删除触发器

DROP TRIGGER DELS

SELECT \* FROM sys.triggers WHERE name = 'DELS'







设计性实验

1) 将 S 表的 PRIMARY KEY 完整性约束删除，编写一个 S 表上的触发器来完成在 S

表上插入记录时实现如下目标：SNO 不能为空，且 SNO 不能重复。

CREATE TRIGGER TRG\_INSERT\_S

ON S

INSTEAD OF INSERT

AS

BEGIN

IF EXISTS (SELECT 1 FROM inserted WHERE SNO IS NULL)

BEGIN

RAISERROR ('SNO cannot be NULL', 16, 1)

ROLLBACK TRANSACTION

RETURN

END

IF EXISTS (SELECT 1 FROM inserted GROUP BY SNO HAVING COUNT(\*) > 1)

BEGIN

RAISERROR ('Duplicate SNO values are not allowed', 16, 1)

ROLLBACK TRANSACTION

RETURN

END

-- Insert the records into the S table

INSERT INTO S (SNO, SNAME, DEPT, AGE, SEX)

SELECT SNO, SNAME, DEPT, AGE, SEX

FROM inserted

END

此触发器将在插入记录到 S 表之前对 SNO 进行检查。如果 SNO 为 NULL 或者存在重复的 SNO 值，触发器将回滚事务并抛出相应的错误消息。

对触发器的验证：

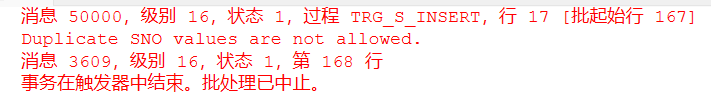
-- 插入一个有效的记录

INSERT INTO S (SNO, SNAME, DEPT, AGE, SEX) VALUES ('200001', '张三', '计算机系', 20, '男');



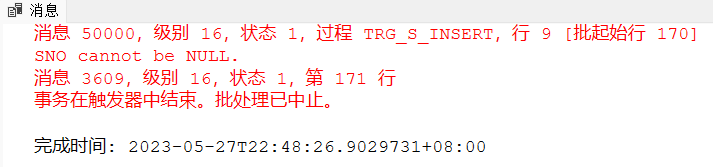
-- 插入一个重复的 SNO

INSERT INTO S (SNO, SNAME, DEPT, AGE, SEX) VALUES ('200001', '李四', '信息系', 19, '男');



-- 插入一个空的 SNO

INSERT INTO S (SNO, SNAME, DEPT, AGE, SEX) VALUES (NULL, '王五', '数学系', 18, '女');



2) 在 C 表修改记录时，若修改的是 CREDIT 字段，则修改值超过 6 时提示出错信息

并不将结果写出数据表

CREATE TRIGGER TRG\_C\_UPDATE

ON C

FOR UPDATE

AS

BEGIN

IF UPDATE(CREDIT)

BEGIN

DECLARE @NewCredit INT;

SELECT @NewCredit = CREDIT FROM INSERTED;

IF @NewCredit > 6

BEGIN

RAISERROR('CREDIT cannot exceed 6.', 16, 1);

ROLLBACK;

RETURN;

END;

END;

END;

-- 插入符合要求的记录

INSERT INTO C (CNO, CNAME, CREDIT)

VALUES ('C07', '课程1', 5);

INSERT INTO C (CNO, CNAME, CREDIT)

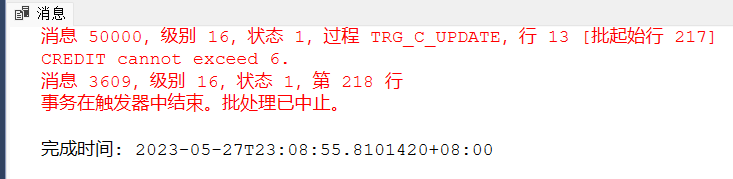
VALUES ('C08', '课程2', 4);

-- 修改 C 表中 CREDIT 字段的值，将其增加到超过 6，触发器应该阻止修改操作

UPDATE C

SET CREDIT = 8

WHERE CNO = 'C07';



-- 修改 C 表中 CREDIT 字段的值，将其修改为符合要求的值

UPDATE C

SET CREDIT = 4

WHERE CNO = 'C01';

