



# 实验报告

实验（四）

姓 名	熊恪峥
学 号	22920202204622
日 期	2023年5月18日
学 院	信息学院
课程名称	数据库

## 实验（四）

### 目录

1	实体完整性	1
2	参照完整性	2
3	触发器的应用	7
4	索引的建立和作用	9
5	实验总结	11

## 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

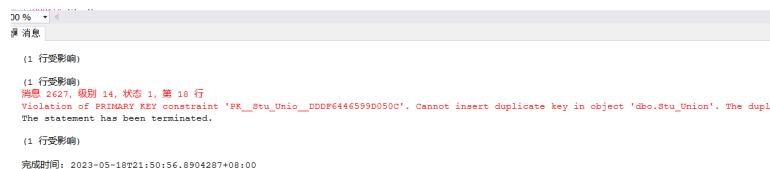
```
CREATE TABLE Stu_Union
(
    sno varchar(10) PRIMARY KEY,
    sname varchar(50),
    gender varchar(10),
    age int,
    major varchar(50)
);

INSERT INTO Stu_Union(sno, sname, Gender, Age, Major)
VALUES('201001', '张三', '男', 22, '计算机科学与技术');
INSERT INTO Stu_Union(sno, sname, Gender, Age, Major)
VALUES('201002', '李四', '男', 23, '软件工程');
```

2. 演示违反实体完整性的插入操作

```
INSERT INTO Stu_Union(sno, sname, Gender, Age, Major)
VALUES('12345678', '王五', '男', 23, '软件工程');

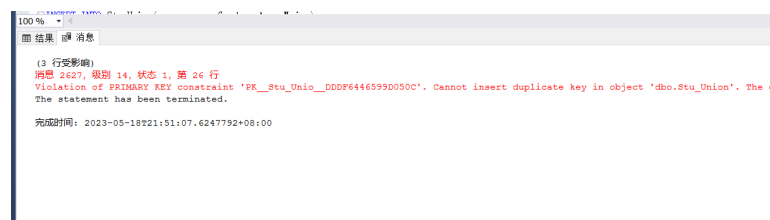
INSERT INTO Stu_Union(sno, sname, Gender, Age, Major)
VALUES('201001', '李四', '男', 23, '软件工程');
```



30 %  
消息  
(1 行受影响)  
(1 行受影响)  
消息 2627, 级别 14, 状态 1, 第 10 行  
Violation of PRIMARY KEY constraint 'PK\_Stu\_Union\_DDDF6446599D050C'. Cannot insert duplicate key in object 'dbo.Stu\_Union'. The dupl  
The statement has been terminated.  
(1 行受影响)  
完成时间: 2023-05-10T21:50:56.8904287+08:00

3. 演示违反实体完整性的更新操作

```
SELECT * FROM Stu_Union
UPDATE Stu_Union
SET sno = '201002'
WHERE sno = '201001';
```

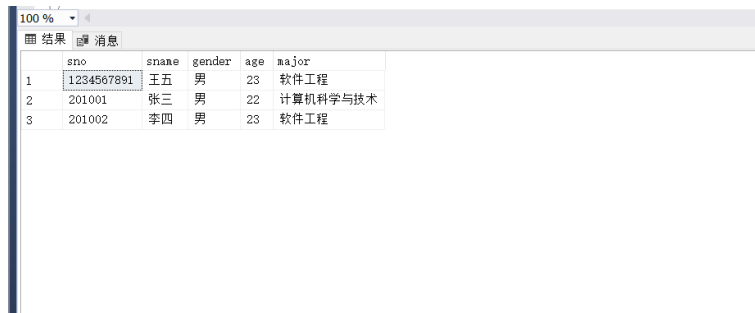


100 %  
消息  
(3 行受影响)  
消息 2627, 级别 14, 状态 1, 第 26 行  
Violation of PRIMARY KEY constraint 'PK\_Stu\_Union\_DDDF6446599D050C'. Cannot insert duplicate key in object 'dbo.Stu\_Union'. The  
The statement has been terminated.  
完成时间: 2023-05-10T21:51:07.6247792+08:00

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

```
SET XACT_ABORT ON;
BEGIN TRAN;
INSERT INTO Stu_Union(sno, sname, Gender, Age, Major)
```

```
VALUES('111111111', '小明', '男', 20, '数学');
UPDATE Stu_Union
SET sname = 'xkz'
WHERE sno = '222222222';
-- 第一条语句插入成功，第二条语句更新失败，整个事务将回滚
ROLLBACK TRAN;
```

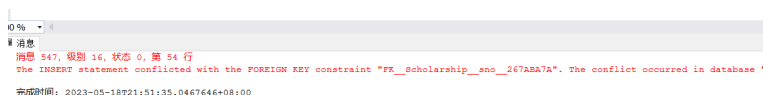


	sno	sname	gender	age	major
1	1234567891	王五	男	23	软件工程
2	201001	张三	男	22	计算机科学与技术
3	201002	李四	男	23	软件工程

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。

```
CREATE TABLE Scholarship
(
    ID int PRIMARY KEY,
    sno varchar(10),
    ScholarshipType varchar(50),
    FOREIGN KEY (sno) REFERENCES Stu_Union(sno)
);

INSERT INTO Scholarship(ID, sno, ScholarshipType)
VALUES(1, '333333333', '全额奖学金');
```



消息 547, 级别 16, 状态 0, 第 54 行  
The INSERT statement conflicted with the FOREIGN KEY constraint "FK\_Scholarship\_sno\_267ABA7A". The conflict occurred in database  
完成时间: 2023-05-18 21:51:35.0467646+08:00

## 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。

```
CREATE TABLE Course
(
    cno varchar(10) PRIMARY KEY,
    cname varchar(50),
    credit int
);

INSERT INTO Course(cno, cname, credit)
VALUES('01', '数据库原理', 4),
```

```
( '02', '操作系统', 3),  
( '03', '数据结构', 3),  
( '45', '数据结构2', 3)
```

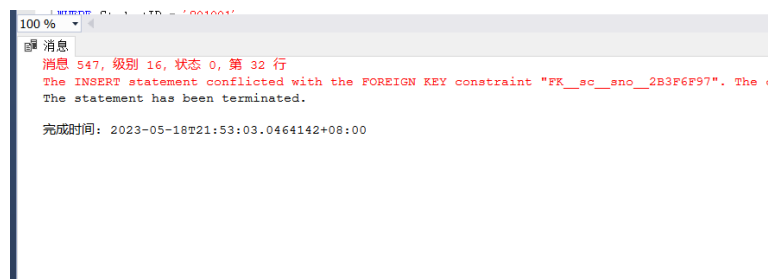
2. 建立表sc，另sno和cno分别为参照Stu\_Union表以及Course表的外键，设定为级连删除，并令(sno, cno)为其主键。在不违反参照完整性的前提下，插入数据。

```
CREATE TABLE sc  
(  
    sno varchar(10),  
    cno varchar(10),  
    grade int,  
    PRIMARY KEY (sno, cno),  
    FOREIGN KEY (sno) REFERENCES Stu_Union(sno) ON DELETE CASCADE,  
    FOREIGN KEY (cno) REFERENCES Course(cno) ON DELETE CASCADE  
);
```

```
INSERT INTO sc(sno, cno, grade)  
VALUES('201001', '01', 80),  
      ('201002', '01', 90),  
      ('201001', '02', 85),  
      ('201002', '02', 70);
```

3. 演示违反参照完整性的插入数据

```
INSERT INTO sc(sno, cno, grade)  
VALUES('201003', '04', 75);
```



4. 在Stu\_Union中删除数据，演示级连删除。

```
DELETE FROM Stu_Union  
WHERE StudentID = '201001';
```

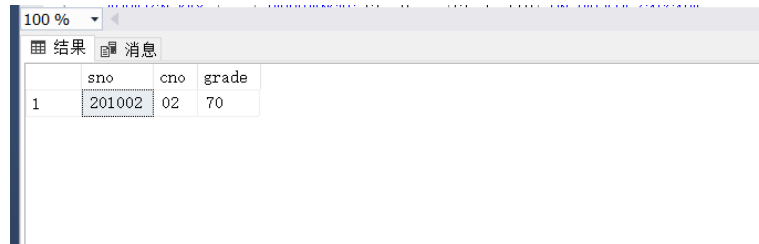
	sno	cno	grade
1	201002	01	90
2	201002	02	70

5. Course中删除数据，演示级连删除。

```
DELETE FROM Course
```

```
WHERE cno = '01';
```

-- 此时，sc表中所有选了该课程的学生，都会被级连删除。



	sno	cno	grade
1	201002	02	70

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。

```
CREATE TABLE Stu_Card
```

```
(
```

```
    card_id varchar(10) PRIMARY KEY,
```

```
    sno varchar(10),
```

```
    create_date datetime,
```

```
    FOREIGN KEY (sno) REFERENCES Stu_Union(StudentID) ON DELETE CASCADE
```

```
);
```

```
INSERT INTO Stu_Card(card_id, sno, create_date)
```

```
VALUES('20100101', '201001', '2021-01-01'),
```

```
      ('20100102', '201002', '2021-01-02');
```

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。

```
CREATE TABLE ICBC_Card
```

```
(
```

```
    bank_id varchar(10) PRIMARY KEY,
```

```
    stu_card_id varchar(10),
```

```
    expired_date datetime,
```

```
    FOREIGN KEY (stu_card_id) REFERENCES Stu_Card(card_id) ON DELETE CASCADE
```

```
);
```

```
INSERT INTO ICBC_Card(bank_id, stu_card_id, expired_date)
```

```
VALUES('955880001', '20100101', '2026-01-01'),
```

```
      ('955880002', '20100102', '2027-01-01');
```

100 % ▾

结果 消息

	sno	cno	grade	

	card_id	sno	create_date	
1	20100101	201001	2021-01-01 00:00:00.000	

	bank_id	stu_card_id	expired_date	
1	955880001	20100101	2026-01-01 00:00:00.000	

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。

```
DELETE FROM Stu_Union
WHERE StudentID = '201002';
-- 此时，sc表中该学生所选的所有课程，以及Stu_Card、ICBC_Card中与该学生相关的记录，都会被级连删除。
SELECT * FROM SC
SELECT * FROM Stu_Card
SELECT * FROM ICBC_Card
```

结果 消息					
sno	sname	gender	age	major	
1	1234567891	王五	男	23	软件工程
2	201001	张三	男	22	计算机科学与技术
3	201002	李四	男	23	软件工程

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。

```
SET XACT_ABORT ON;
BEGIN TRAN;
DELETE FROM Stu_Union
WHERE StudentID = '201001';
-- 通过修改外键属性，禁止级连删除
ALTER TABLE ICBC_Card
DROP CONSTRAINT FK_ICBC_Card_stu_card_id;

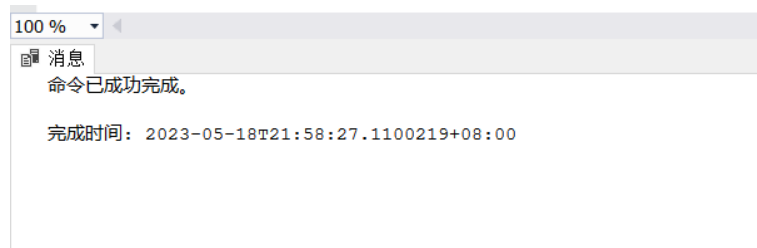
ALTER TABLE Stu_Card
DROP CONSTRAINT FK_Stu_Card_stu_id;

ALTER TABLE Stu_Union
DROP CONSTRAINT FK_sc_sno;

-- 在这种情况下，将无法删除sc表中与该学生相关的记录，整个事务会回滚
DELETE FROM Stu_Union
```

```
WHERE StudentID = '201001';
```

```
COMMIT TRAN
```



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）。

```
CREATE TABLE Course_Teacher_Teaching (  
    CourseId INT NOT NULL,  
    TeacherId INT NOT NULL,  
    PRIMARY KEY (CourseId)  
);
```

```
CREATE TABLE Teacher_Listening (  
    TeacherId INT NOT NULL,  
    CourseId INT NOT NULL,  
    PRIMARY KEY (TeacherId)  
);
```

```
ALTER TABLE Course_Teacher_Teaching  
ADD CONSTRAINT fk_Course_Teacher_Teaching_Teacher_Listening  
FOREIGN KEY (TeacherId) REFERENCES Teacher_Listening(TeacherId);
```

```
ALTER TABLE Teacher_Listening  
ADD CONSTRAINT fk_Teacher_Listening_Course_Teacher_Teaching  
FOREIGN KEY (CourseId) REFERENCES Course_Teacher_Teaching(CourseId);
```

```
CREATE TRIGGER trg_Teacher_Listening  
ON Teacher_Listening  
FOR INSERT, UPDATE  
AS  
BEGIN  
    IF EXISTS (SELECT 1 FROM inserted i LEFT JOIN Course_Teacher_Teaching ct ON i.CourseId = ct.CourseId  
    BEGIN  
        RAISERROR ('Error: A teacher can only listen one course taught by himself', 16, 1)  
        ROLLBACK  
    END  
END
```



### 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。

```
-- 1
-- 创建记录表
CREATE TABLE LowScoreStudent
(
    sno VARCHAR(10) PRIMARY KEY,
    cno VARCHAR(10),
    grade int,
    FOREIGN KEY (sno) REFERENCES Stu_Union(sno) ON DELETE CASCADE,
    FOREIGN KEY (cno) REFERENCES Course(cno) ON DELETE CASCADE
)

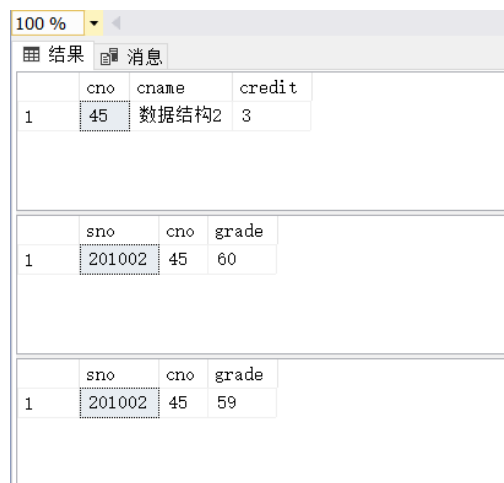
-- 创建触发器
CREATE TRIGGER Trigger_Sc_Insert ON sc
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @Sno VARCHAR(10), @Cno VARCHAR(10), @Score INT;

    SELECT @Sno = Sno, @Cno = Cno, @Score = Grade
    FROM inserted;

    IF (@Score < 60)
    BEGIN
        UPDATE sc SET grade = 60 WHERE Sno = @Sno AND Cno = @Cno;

        INSERT INTO LowScoreStudent (Sno, Cno, grade) VALUES (@Sno, @Cno, @Score);
    END
END
```



	cno	cname	credit
1	45	数据结构2	3

	sno	cno	grade
1	201002	45	60

	sno	cno	grade
1	201002	45	59

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。

图 1: 修改前

结果 消息		
	sno	sname
1	10001	李勇
2	10002	aa

结果 消息			
	sno	cno	grade
1	10001	0001	60

3. --2

-- 创建触发器

```
CREATE TRIGGER Trigger_StuUnion_UpdateSid ON stu_union
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE sc
    SET Sno = inserted.Sno
    FROM sc
    INNER JOIN inserted ON inserted.Sno = sc.Sno;
END
```

图 2: 修改后

结果 消息		
	sno	sname
1	10002	aa
2	10003	李勇

83 % 结果 消息			
	sno	cno	grade
1	10003	0001	60

4. 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。

图 3: 修改前

结果 消息		
	sno	sname
1	10002	aa
2	10003	李勇

83 % 结果 消息			
	sno	cno	grade
1	10003	0001	60

```
CREATE TRIGGER Trigger_Stu_Union_Delete ON stu_union
AFTER DELETE
AS
BEGIN
    SET NOCOUNT ON;
```

```

DECLARE @Sno VARCHAR(10);

SELECT @Sno = deleted.sno
FROM deleted;

DELETE FROM sc WHERE Sno = @Sno;
END

```

图 4: 修改后

结果 消息		
	sno	sname
1	10002	aa

结果 消息		
sno	cno	grade

5. 演示触发器删除操作。

```

DROP TRIGGER LowScoreTrigger;
DROP TRIGGER UpdateSidTrigger;
DROP TRIGGER DeleteStudentTrigger;

```

## 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
-- 创建 STUDENTS 表
CREATE TABLE STUDENTS (
    sno VARCHAR(10) PRIMARY KEY,
    sname VARCHAR(50),
    email VARCHAR(50),
    grade INT
);

-- 在 sname 字段上创建聚簇索引
CREATE CLUSTERED INDEX idx_sname ON STUDENTS(sname);

-- 在 grade 字段上创建非聚簇索引
CREATE NONCLUSTERED INDEX idx_grade ON STUDENTS(grade);

```

建立聚簇索引出错，因为默认为主键建立聚簇索引，要先删除主键的聚簇索引才能建索引。

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

```

-- 创建 CHOICES 表
CREATE TABLE CHOICES (
    no INT IDENTITY(1,1) PRIMARY KEY,
    sid VARCHAR(10),

```

```

    tid VARCHAR(10),
    cid VARCHAR(10),
    score INT
);

-- 为 cid 字段添加索引（实验2A）
SELECT COUNT(*) FROM CHOICES WHERE cid = '101';

-- 为 cid 字段添加非聚簇索引（实验2B）
CREATE NONCLUSTERED INDEX idx_cid ON CHOICES(cid);

-- 重新运行查询（实验2B）
SELECT COUNT(*) FROM CHOICES WHERE cid = '101';

```

(a) 没有索引

消息  
命令已成功完成。  
完成时间：2023-05-09T21:06:39.2848487+08:00

(b) 有索引

! %  
结果 消息  
(无列名)  
5859

使用索引可以显著提高查询性能，特别是当针对大型数据集进行查询时。因此，在设计数据库时，为了优化查询性能，我们应该考虑添加恰当的索引来支持常见的查询操作。例如，为经常使用的列添加索引，或者为经常连接的表添加外键索引，以提高查询的效率。然而，值得注意的是，过多的索引可能会对数据库性能产生负面影响。因此，在创建索引时必须在性能和查询效率之间找到平衡点。

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) 没有索引

消息  
命令已成功完成。  
完成时间：2023-05-09T21:15:43.3369330+08:00

(b) 有非聚簇索引

1	500250857	800554358	262175339	10037	91
2	512254732	800554358	283495419	10015	62
3	515656273	800554358	244897158	10047	NULL
4	562223324	800554358	281785755	10043	79
5	588918248	800554358	224600699	10003	51

(c) 有聚簇索引

1	515656273	800554358	244897158	10047	NULL
2	588918248	800554358	224600699	10003	51
3	500250857	800554358	262175339	10037	91
4	512254732	800554358	283495419	10015	62
5	562223324	800554358	281785755	10043	79

```

-- 为 sid 字段添加非聚簇索引（实验3B）
CREATE INDEX idx_sid ON CHOICES(sid);

-- 为 sid 字段添加聚簇索引（实验3C）
CREATE CLUSTERED INDEX idx_sid ON CHOICES(sid);

-- 运行查询
SELECT * FROM CHOICES WHERE sid = 'S01';

```

通过对比实验结果并进行分析，可以得出以下结论：

- (a) 在没有索引的情况下，每次查询都需要扫描整个表，效率较低，随着数据量的增加，查询语句的执行时间会逐渐变长。因此，在表的大小较大时，没有索引会显著影响查询性能。
- (b) 使用聚集索引进行查询时，它首先找到符合查询条件的第一条记录，然后根据索引中定义的列值顺序去查找下一条记录，直到找到满足所有查询条件的所有记录为止。由于聚簇索引是按照行的物理存储顺序排列的，因此查询时磁盘的读取次数会减少，从而提高查询速度。
- (c) 使用非聚簇索引进行查询时，它首先根据索引列的值查找对应的记录，然后使用指针定位实际行数据，最后返回符合查询条件的结果。与聚簇索引不同，使用非聚簇索引进行查询时需要进行额外的磁盘读取，因此查询速度可能会受到一定的影响。

## 5 实验总结

在本次实验中，我了解了实体完整性并学会了如何定义主键以保证唯一性。更好地理解实体与关系型表的关系我还了解了参照完整性以及如何用外键约束实现关系型表之间数据的一致性，学会了触发器的基本原理和用处。实际操作了通过触发器对表中的数据进行限制和管理的方法，练习了编写触发器所需的语法。同时，对不同的索引进行了对比，更好地理解了相关的知识。