

实验 10、触发器

10.1 实验目的

通过实验使学生加深对数据完整性的理解，学会理解、创建和使用触发器。

10.2 实验内容

- (1) 为 **chenht_Teacher** 表建立触发器 T1，当插入或使更新表中的数据时，保证所操作的纪录的 **cht_Tage** 值大于 0。
- (2) 为 **chenht_Teacher** 表建立触发器 T2，禁止删除编号为 00001 的 CEO。
- (3) **chenht_Teacher** 表中的人员的编号是唯一且不可更变的，创建触发器 T3 实现更新中编号的不可改变性。
- (4) 演示违反 T1 触发器的约束的插入操作。
- (5) 演示违反 T1 触发器的约束的更新操作。
- (6) 演示违反 T2 触发器的约束的插入操作。
- (7) 演示违反 T2 触发器的约束的更新操作。

10.3 实验步骤

- (1) 仍然使用自定义完整性实验中的 **chenht_Teacher** 表。为此表建立触发器 T1，当插入或使更新表中的数据时，保证所操作的纪录的 **cht_Tage** 值大于 0。

在新建查询窗口中输入如下 SQL 语句

```
USE chenht_University_Mis
GO
CREATE TRIGGER T1 ON chenht_Teacher
FOR INSERT, UPDATE
AS
IF (SELECT cht_Tage FROM INSERTED)< 1
BEGIN
PRINT '职工年龄必须是大于 0 的正数！操作失败！'
ROLLBACK TRANSACTION
END
```

- (2) 为 **chenht_Teacher** 表建立触发器 T2，禁止删除编号为 S01 的 CEO。

在新建查询窗口中输入如下 SQL 语句

```
USE chenht_University_Mis
```

```

GO
CREATE TRIGGER T2 ON chenht_Teacher
FOR DELETE
AS
IF (SELECT cht_Tno FROM DELETED)='T01'
BEGIN
PRINT '此人是 CEO!删除操作失败'
ROLLBACK TRANSACTION
END

```

(3) **chenht_Teacher** 表中的人员的编号是唯一且不可更变的,创建触发器 T3 实现更新中编号的不可改变性。

在新建查询窗口中输入如下 SQL 语句

```

USE chenht_University_Mis
GO
CREATE TRIGGER T3 ON chenht_Teacher
FOR UPDATE
AS
IF UPDATE(cht_Tno)
BEGIN
PRINT '职工编号不能修改!'
ROLLBACK TRANSACTION
END

```

(4) 在新建查询窗口中输入如下 SQL 语句

```

USE chenht_University_Mis
INSERT INTO chenht_Teacher VALUES('T03','李宏','F',-10,'开发部')

```

(5) 在新建查询窗口中输入如下 SQL 语句

```

USE chenht_University_Mis
UPDATE chenht_Teacher SET cht_Tage = -7 WHERE cht_Tno = 'T01'

```

(6) 在新建查询窗口中输入如下 SQL 语句

```

USE chenht_University_Mis
DELETE FROM chenht_Teacher WHERE cht_Tname = '李用'

```

(7) 在新建查询窗口中输入如下 SQL 语句

```

USE chenht_University_Mis
UPDATE chenht_Teacher SET cht_Tno = 'T07' WHERE cht_Tsex = 'F'

```

10.4 实验结果

(1) 仍然使用自定义完整性实验中的 **chenht_Teacher** 表。为此表建立触发器 T1, 当插入或使更新表中的数据时, 保证所操作的纪录的 **cht_Tage** 值大于 0。

SQL 语句:

```

USE chenht_University_Mis
GO
CREATE TRIGGER T1 ON chenht_Teacher
FOR INSERT, UPDATE
AS
IF (SELECT cht_Tage FROM INSERTED)< 1

```

```
BEGIN
PRINT '职工年龄必须是大于 0 的正数！操作失败！'
ROLLBACK TRANSACTION
END
```

实验结果：

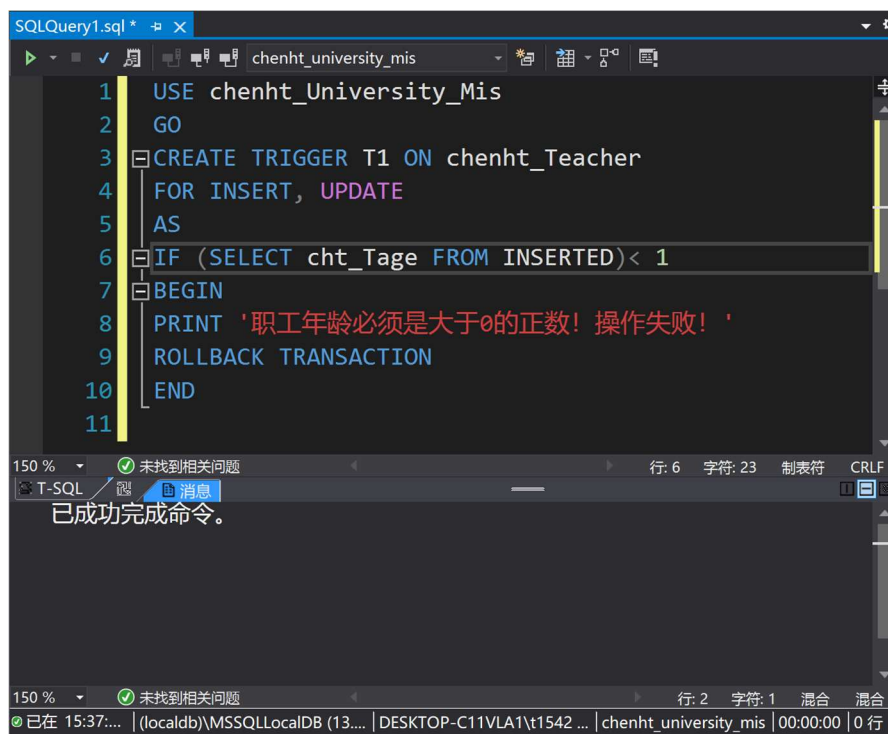


图 10-1 在 chenht_Teacher 插入年龄触发器的实验结果

(4) 在新建查询窗口中输入如下 SQL 语句。

SQL 语句：

```
USE chenht_University_Mis
INSERT INTO chenht_Teacher VALUES('T03','李宏','F',-10,'开发部')
```

实验结果：

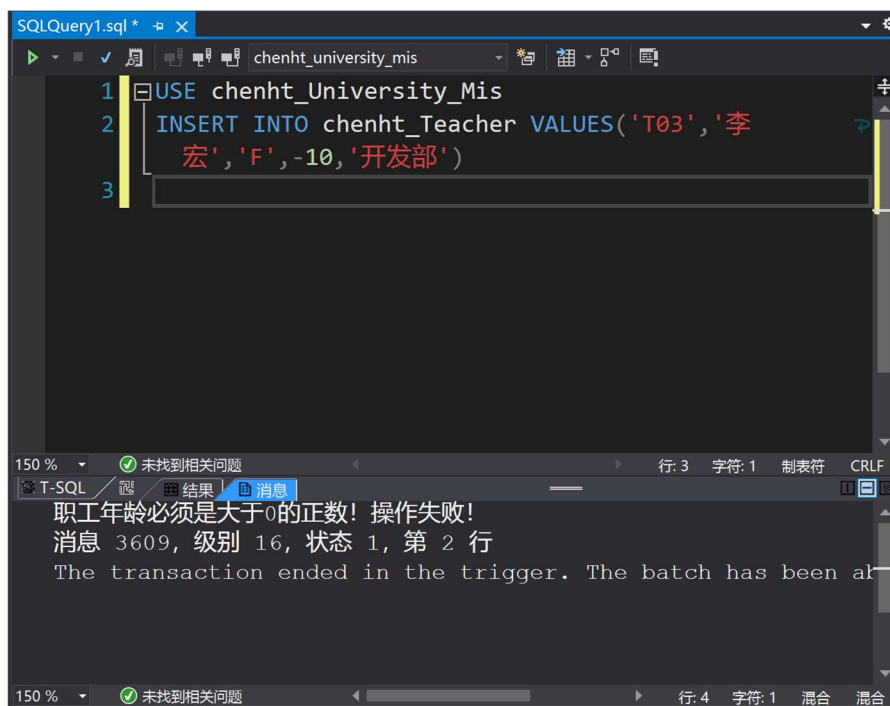


图 10-2 在 chenht_Teacher 中插入一条违反 cht_Tage 触发器的结果实验结果

错误信息：职工年龄必须是大于0的正数！操作失败！

消息 3609, 级别 16, 状态 1, 第 2 行

The transaction ended in the trigger. The batch has been aborted.

由于在第（1）小题加入了触发器，在插入违反 cht_Tage>0 后，将会撤销当前的事务。因此此记录并没有插入到当前的数据集中。在执行 `select * from chenht_Teacher` 后，并没有该条记录的存在。

(6) 在新建查询窗口中输入如下 SQL 语句。

SQL 语句：

```
USE chenht_University_Mis
DELETE FROM chenht_Teacher WHERE cht_Tname = '李用'
```

实验结果：

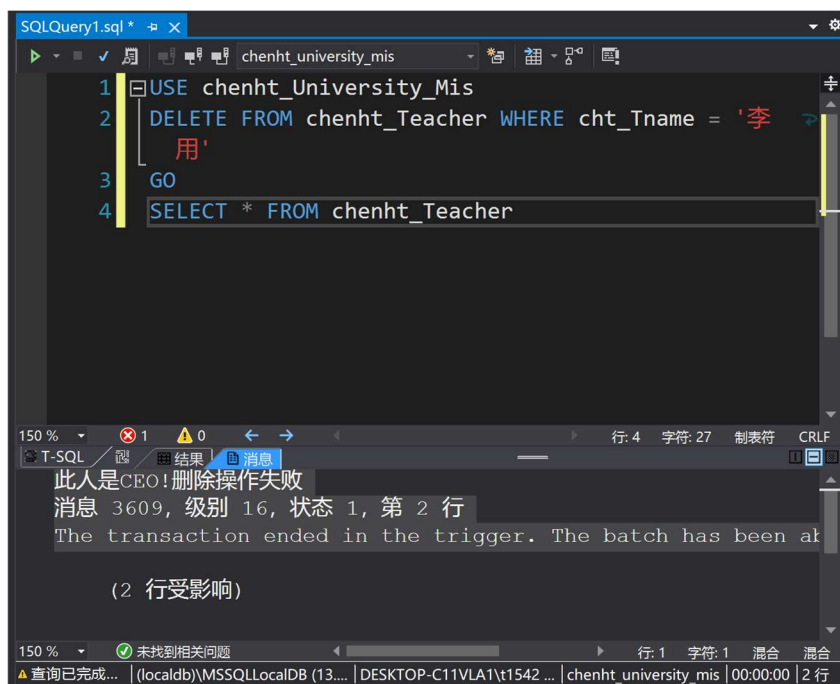


图 10-3 删除 chenht_Teacher 中违反触发器 T2 的数据行的实验结果。

错误信息：此人是CEO!删除操作失败

消息 3609, 级别 16, 状态 1, 第 2 行

The transaction ended in the trigger. The batch has been aborted.

分析：由于触发器 T2 的限制，在 chenht_Teacher 中删除 cht_Tno = 'T01' 的数据行时，将会回滚事务，而 cht_Tno='T01' 得数据行得 cht_Tname 得属性值为李用，因此并没有执行事务，对于数据没有影响。

(7) 在新建查询窗口中输入如下 SQL 语句。

SQL 语句：

```
USE chenht_University_Mis
UPDATE chenht_Teacher SET cht_Tno = 'T07' WHERE cht_Tsex = 'F'
```

实验结果：

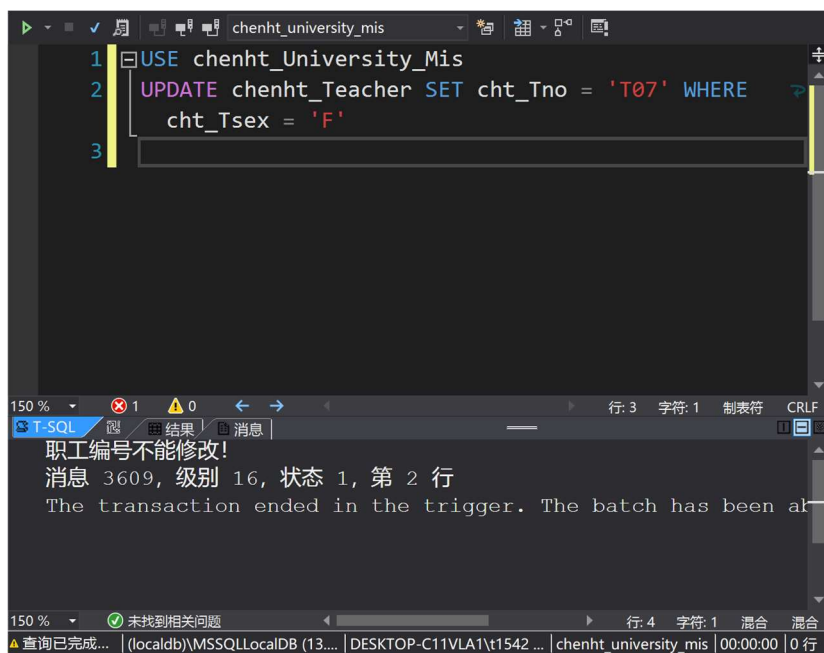


图 10-4 更改 chenht_Teacher 中违反触发器 T3 的实验结果

错误信息：职工编号不能修改！

消息 3609, 级别 16, 状态 1, 第 2 行

The transaction ended in the trigger. The batch has been aborted.

分析: 由于触发器 T3 中禁止了对 cht_Tno 的更新操作, 因此在执行 UPDATE 命令对 cht_Tno 中进行更新时, 将会回滚事务, 导致更新操作失败。

10.5 实验体会

1) 实验反思

- a 在使用触发器的过程中, 将会创建 INSERTED 和 DELETED 两张表, 当使用 INSERT INTO 同时插入多条数据时, 触发器 T1 会出现异常。可以修改 T1 触发器的检查方式为在 INSERTED 表中查找 Taged <= 0 的数据项, 如果为空, 则继续执行, 否则回滚事务。因此, 在前面的实验过程中, 应当考虑到 INSERTED 和 DELETED 两张表中的数据有多行的情况。

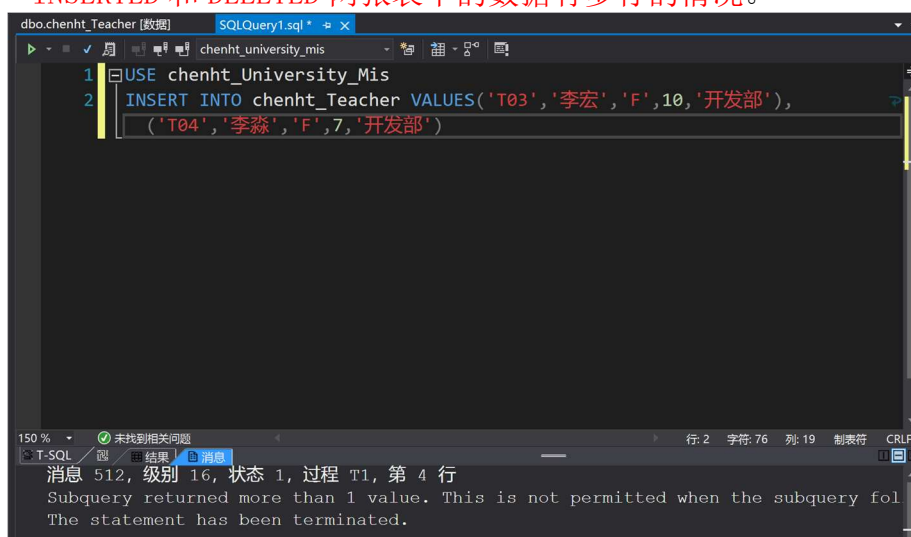
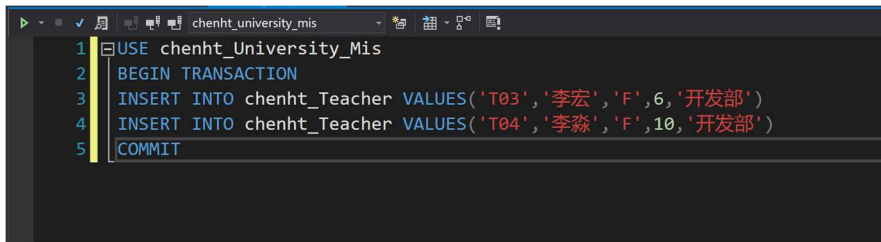


图 10-5 触发器异常的实验结果

- b 触发器的触发将有一条相关的 SQL 语句触发, 例如下面的语句将会两次触

发触发器。



```
1 USE chenht_University_Mis
2 BEGIN TRANSACTION
3 INSERT INTO chenht_Teacher VALUES('T03','李宏','F',6,'开发部')
4 INSERT INTO chenht_Teacher VALUES('T04','李森','F',10,'开发部')
5 COMMIT
```

图 10-4 在事务中引发触发器的实验结果

2) 实验收获

- a 触发器的触发条件是由动作来触发的，触发器内可以取消事务，如果引发触发器的语句是事务的一部分，在触发器中取消事务将会导致整个事务失败。
- b 触发器常用的地方在于动态的更新统计元素，例如，数据库中有一个字段保存教师的个数，那么可以建立触发器，统计 INSERTED 和 DELETED 的个数，对字段进行动态更改，从而减少 SELECT 带来的性能消耗。相当于对统计结果的缓存。