第五章　数据库完整性

关系的完整性

① 针对数据库系统关系数据模型提出；
② 是模型中数据及其联系所具有的制约和依存规则；
③ 用以限定符合数据模型的数据库状态以及状态的变迁；
④ 以保证数据的正确、有效与相容。

数据的正确性、有效性和相容性

- **正确性：** **指数据的合法性。**

例：

① 输入成绩时，应该输入数值（假如成绩定义为数值型），而实际输入了字符，即不正确。

② 其他，如学号、工号等必须**唯一**。

- **有效性：** **指数据是否在有效的范围之内。**

例：

① 输入年龄时，应该输入0-150之间的数据，而实际输入了-5，即无效。

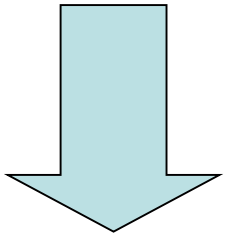
② 输入月份，1-12。

- **相容性：** **指表示同一事实的同一数据应该相同，或满足某一约束关系的一组数据不应发生互斥问题。**

例：

① 输入饲料配比百分比时，应该5种原料百分比之和加起来为100%，而实际输入数据加起来大于100%，即不相容。

怎样解决数据的正确性、有效性和相容性问题？



实现关系数据库的完整性约束。

数据库完整性

- 数据的完整性和安全性是两个不同概念
  - 数据的完整性
    - 防止数据库中存在不符合语义的数据，也就是防止数据库中存在不正确的数据
    - 防范对象：不合语义的、不正确的数据
  - 数据的安全性
    - 保护数据库防止恶意的破坏和非法的存取
    - 防范对象：非法用户和非法操作

数据库完整性

为维护数据库的完整性，DBMS必须：

- 1.提供定义完整性约束条件的机制
- 2.提供完整性检查的方法
- 3.违约处理
  - ✓ 拒绝该操作
  - ✓ 其他处理方法

完整性规则五元组表示

(D，O，A，C，P)

- D（Data）：约束作用的数据对象
- O（Operation）：触发完整性检查的数据库操作

  当用户发出操作请求需要检查该完整性规则是立即检查还是延迟检查；
- A（Assertion）：数据对象必须满足的断言或语义约束，这是规则的主体；
- C（Condition）：选择A作用的数据对象值的谓词
- P（Procedure）：违反完整性规则时触发的过程

例子

实例	学号不能为空	教授工资不得低于1000元
数据对象D	约束作用的对象为Sno属性	约束作用的对象为工资Sal属性
操作O	插入或修改Student 元组时	插入或修改职工元组时
断言A	Sno不能为空	Sal不能小于1000
谓词C	无（A可作用于所有记录的Sno属性）	职称=‘教授’ (A仅作用于职称=‘教授’的记录)
过程P	拒绝执行该操作	拒绝执行该操作

数据完整性概述

类型

- 实体完整性
- 域完整性：通过Foreign key约束check约束not null定义默认值和规则来确定可能值的范围
- 用户定义完整性：通过存储过程触发器等对象来实施具体数据库完整性

使用约束实施数据的完整性

SQL Server支持5类约束:

- PRIMARY KEY
- FOREIGN KEY

- CHECK
- NOT NULL
- UNIQUE

5.1 实体完整性

- 实体完整性定义
- 实体完整性检查和违约处理

实体完整性定义

- 关系模型的实体完整性
  - CREATE TABLE中用PRIMARY KEY定义
- 单属性构成的码有两种说明方法
  - 定义为列级约束条件
  - 定义为表级约束条件
- 对多个属性构成的码只有一种说明方法
  - 定义为表级约束条件

实体完整性定义(续)

［例1］ 将Student表中的Sno属性定义为码

(1)在列级定义主码

```
CREATE TABLE Student
  (Sno  CHAR(9)  PRIMARY KEY，
   Sname  CHAR(20) NOT NULL，
   Ssex  CHAR(2) ，
   Sage  SMALLINT，
   Sdept  CHAR(20));
```

实体完整性定义(续)

(2)在表级定义主码

```
CREATE TABLE Student
 (Sno  CHAR(9)，
  Sname  CHAR(20) NOT NULL，
  Ssex  CHAR(2) ，
  Sage  SMALLINT，
  Sdept  CHAR(20)，
  PRIMARY KEY (Sno)
);
```

实体完整性定义(续)

［例2］将SC表中的Sno，Cno属性组定义为码

```
CREATE TABLE SC
  (Sno   CHAR(9)  NOT NULL，
   Cno  CHAR(4)  NOT NULL，
   Grade    SMALLINT，
   PRIMARY KEY (Sno，Cno)     /*只能在表级定义主码*/
  );
```

实体完整性检查和违约处理

- 插入或对主码列进行更新操作时，RDBMS按照实体完整性规则自动进行检查。包括：
  - 1. 检查主码值是否唯一，如果不唯一则拒绝插入或修改
  - 2. 检查主码的各个属性是否为空，只要有一个为空就拒绝插入或修改

5.2 参照完整性

- 参照完整性定义
- 参照完整性检查和违约处理

5.2.1 参照完整性定义

- 关系模型的参照完整性定义
  - 在CREATE TABLE中用FOREIGN KEY短语定义哪些列为外码
  - 用REFERENCES短语指明这些外码参照哪些表的主码

参照完整性定义(续)

例如，关系SC中一个元组表示一个学生选修的某门课程的成绩，（Sno，Cno）是主码。Sno，Cno分别参照引用Student表的主码和Course表的主码

［例3］ 定义SC中的参照完整性

```
CREATE TABLE SC
  (Sno    CHAR(9)  NOT NULL，
   Cno     CHAR(4)  NOT NULL，
   Grade    SMALLINT，
   PRIMARY KEY (Sno， Cno)，    /*在表级定义实体完整性*/
   FOREIGN KEY (Sno) REFERENCES Student(Sno)，
    /*在表级定义参照完整性*/
   FOREIGN KEY (Cno) REFERENCES Course(Cno)
   /*在表级定义参照完整性*/
);
```

参照完整性检查和违约处理

- 参照完整性违约处理
  - 1. 拒绝(NO ACTION)执行
    - 默认策略
  - 2. 级联(CASCADE)操作
  - 3. 设置为空值（SET-NULL）
    - 对于参照完整性，除了应该定义外码，还应定义外码列是否允许空值

［例4］ 显式说明参照完整性的违约处理示例

```
CREATE TABLE SC
    (Sno   CHAR(9)  NOT NULL，
     Cno   CHAR(4)  NOT NULL，
     Grade  SMALLINT，
     PRIMARY KEY（Sno，Cno），
     FOREIGN KEY (Sno) REFERENCES Student(Sno)
        ON DELETE CASCADE     /*级联删除SC表中相应的元组*/
          ON UPDATE CASCADE，  /*级联更新SC表中相应的元组*/
     FOREIGN KEY (Cno) REFERENCES Course(Cno)
         ON DELETE NO ACTION
         /*当删除course 表中的元组造成了与SC表不一致时拒绝删除*/
         ON UPDATE CASCADE
       /*当更新course表中的cno时，级联更新SC表中相应的元组*/
    );
```

参照完整性的实现

例:职工一部门数据库包含职工表EMP和部门表DEPT

1 DEPT关系的主码为部门号Deptno

2 EMP关系的主码为职工号Empno, 外码为部门号Deptno

称DEPT为被参照关系或目标关系，EMP为参照关系

DEPT(Deptno, Dname)

EMP(Empno, Ename, Ssex, Sage, Deptno)

RDBMS实现参照完整性时需要考虑以下4方面：

1. 外码是否可以接受空值的问题

- 外码是否能够取空值：依赖于应用环境的语义
- 实现参照完整性：

  系统提供定义外码的机制

  定义外码列是否允许空值的机制

1. 外码是否可以接受空值的问题

DEPT(Deptno, Dname)

EMP(Empno, Ename, Ssex, Sage, Deptno)

例1：在职工一部门数据库中，

EMP关系包含有外码Deptno

某元组的这一列若为空值，表示这个职工尚未分配到任何具体的部门工作.和应用环境的语义是相符.

1. 外码是否可以接受空值的问题

SC：

Sno	Cno	Grade
95001	1	92
95001	2	85
95001	3	88
95002	2	90
95002	3	

Student：

Sno	Sname	Ssex	Sage	Sdept
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

例2：学生－选课数据库

Student关系为被参照关系，其主码为Sno。

SC为参照关系，外码为Sno。

若SC的Sno为空值：表明尚不存在的某个学生，或者某个不知学号的学生，选修了某门课程，其成绩记录在Grade中与学校的应用环境是不相符的，因此SC的Sno列不能取空值。

2.在被参照关系中删除元组时的问题

出现违约操作的情形：

删除被参照关系的某个元组（student）

而参照关系有若干元组(SC)的外码值与被删除的被参照关系的主码值相同

- 违约反应：可有三种策略
  - 级联删除（CASCADES）
  - 受限删除（RESTRICTED）
  - 置空值删除（NULLIFIES）

    这三种处理方法，哪一种是正确的，要依应用环境的语义来定

2.在被参照关系中删除元组时的问题

- 级联删除

  将参照关系中外码值与被参照关系中要删除元组主码值相对应的元组一起删除

- 受限删除

  当参照关系中没有任何元组的外码值与要删除的被参照关系的元组的主码值相对应时，系统才执行删除操作，否则拒绝此删除操作

- 置空值删除

  删除被参照关系的元组，并将参照关系中与被参照关系中被删除元组主码值相等的外码值置为空值。

2.在被参照关系中删除元组时的问题

例：要删除Student关系中Sno=95001的元组，

而SC关系中有3个元组的Sno都等于95001 。

- 级联删除：将SC关系中所有3个Sno= 95001的元组一起删除。如果参照关系同时又是另一个关系的被参照关系，则这种删除操作会继续级联下去 。
- 受限删除：系统将拒绝执行此删除操作。

SC：

Sno	Cno	Grade
95001	1	92
95001	2	85
95001	3	88
95002	2	90
95002	3	

Student：

Sno	Sname	Ssex	Sage	Sdept
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

- 问题：能否置空删除？

2.在被参照关系中删除元组时的问题

- 置空值删除：将SC关系中所有Sno= 95001的元组的Sno值置为空值。

- 在学生选课数据库中，显然第一种方法和第二种方法都是对的。第三种方法不符合应用环境语义。

3.在参照关系中插入元组时的问题

- 出现违约操作的情形
  - 需要在参照关系中插入元组，而被参照关系不存在相应的元组
- 违约反应
  - 受限插入
  - 递归插入

3.在参照关系中插入元组时的问题

- 受限插入
  - 仅当被参照关系中存在相应的元组，其主码值与参照关系插入元组的外码值相同时，系统才执行插入操作，否则拒绝此操作。

- 递归插入
  - 首先向被参照关系中插入相应的元组，其主码值等于参照关系插入元组的外码值，然后向参照关系插入元组。

3.在参照关系中插入元组时的问题

例：向SC关系插入（95009，1，90）元组，而Student关系中尚没有Sno= 95009的学生

- 受限插入：系统将拒绝向SC关系插入（ 95009 ，1，90）元组
- 递归插入：系统将首先向Student关系插入Sno= 95009的元组，然后向SC关系插入（ 95009 ，1，90）元组。

Student：

Sno	Sname	Ssex	Sage	Sdept
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

Student

被参照关系

SC：

Sno	Cno	Grade
95001	1	92
95001	2	85
95001	3	88
95002	2	90
95002	3	

Sc

参照关系

4. 修改主码的问题

- 要修改被参照关系中某些元组的主码值，而参照关系中有些元组的外码值正好等于被参照关系要修改的主码值
- 要修改参照关系中某些元组的主码值，而被参照关系中没有任何元组的外码值等于参照关系修改后的主码值

允许修改主码策略

违约反应 (1)

- 修改的关系是被参照关系：与删除类似
  - 级联修改
  - 受限修改
  - 置空值修改

允许修改主码策略

- 级联修改
  - 修改被参照关系中主码值同时，用相同的方法修改参照关系中相应的外码值。
- 受限修改
  - 拒绝此修改操作。只当参照关系中没有任何元组的外码值等于被参照关系中某个元组的主码值时，这个元组的主码值才能被修改。
- 置空值修改
  - 修改被参照关系中主码值，同时将参照关系中相应的外码值置为空值。

允许修改主码策略

例：将Student关系中Sno= 95001的元组中Sno值改为96001。而SC关系中有 3个元组的Sno= 95001

- 级联修改：将SC关系中3个Sno= 95001元组中的Sno值也改为96001 。如果参照关系同时又是另一个关系的被参照关系，则这种修改操作会继续级联下去。
- 受限修改：只有SC中没有任何元组的Sno= 95001时，才能修改Student表中Sno= 95001的元组的Sno值改为96001 。
- 置空值修改：？

SC：

Sno	Cno	Grade
95001	1	92
95001	2	85
95001	3	88
95002	2	90
95002	3	

Student：

Sno	Sname	Ssex	Sage	Sdept
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

允许修改主码策略

违约反应 (2)

- 修改的关系是参照关系：与插入类似
  - 受限修改
  - 递归修改

5.3 用户定义的完整性

- 用户定义的完整性就是针对某一具体应用的数据必须满足的语义要求
- RDBMS提供，而不必由应用程序承担

5.3 用户定义的完整性

- 属性上的约束条件的定义
- 属性上的约束条件检查和违约处理
- 元组上的约束条件的定义
- 元组上的约束条件检查和违约处理

属性上的约束条件的定义

- CREATE TABLE时定义
  - 列值非空（NOT NULL）
  - 列值唯一（UNIQUE）
  - 检查列值是否满足一个布尔表达式（CHECK）

NOT NULL约束(续)

- 用Transact-SQL设置NOT NULL

```
CREATE TABLE Student
        (Sno   CHAR(9) PRIMARY KEY,
         Sname  CHAR(20) UNIQUE,
         Ssex    CHAR(2), not null
         Sage   SMALLINT, not null
         Sdept  CHAR(20),not null
        )
```

惟一性约束(续)

用Transact-SQL设置：

```
CREATE TABLE Student
        (Sno   CHAR(9) PRIMARY KEY,
       Sname  CHAR(20) UNIQUE,
       Ssex    CHAR(2),
       Sage   SMALLINT,
       Sdept  CHAR(20)
      )
```

用Transact-SQL设置：

```
CREATE TABLE  SC
            (Sno  CHAR(9),
             Cno  CHAR(4),
             Grade    SMALLINT check ( grade>=0
                    and grade<=100))
```

属性上的约束条件检查和违约处理

- 插入元组或修改属性的值时，RDBMS检查属性上的约束条件是否被满足
- 如果不满足则操作被拒绝执行

元组上的约束条件的定义

- 在CREATE TABLE时可以用CHECK短语定义元组上的约束条件，即元组级的限制
- 同属性值限制相比，元组级的限制可以设置不同属性之间的取值的相互约束条件

元组上的约束条件的定义(续)

［例9］ 当学生的性别是男时，其名字不能以Ms.打头。

```
CREATE TABLE Student
    (Sno    CHAR(9)，
     Sname  CHAR(8) NOT NULL，
     Ssex    CHAR(2)，
     Sage   SMALLINT，
     Sdept  CHAR(20)，
     PRIMARY KEY (Sno)，
     CHECK (Ssex='女' OR Sname NOT LIKE 'Ms.%')
     /*定义了元组中Sname和 Ssex两个属性值之间的约束条件*/
    )；
```

- ✓ 性别是女性的元组都能通过该项检查，因为Ssex=‘女’成立；
- ✓ 当性别是男性时，要通过检查则名字一定不能以Ms.打头

元组上的约束条件检查和违约处理

- 插入元组或修改属性的值时，RDBMS检查元组上的约束条件是否被满足
- 如果不满足则操作被拒绝执行

5.4 完整性约束命名子句

- CONSTRAINT 约束

CONSTRAINT <完整性约束条件名>

［PRIMARY KEY短语

|FOREIGN KEY短语

|CHECK短语］

完整性约束命名子句(续)

［例10］ 建立学生登记表Student，要求学号在90000~99999之间，姓名不能取空值，年龄小于30，性别只能是“男”或“女”。

```
CREATE TABLE Student
 (Sno  NUMERIC(6)
   CONSTRAINT C1 CHECK (Sno BETWEEN 90000 AND 99999),
   Sname  CHAR(20)
   CONSTRAINT C2 NOT NULL,
   Sage  NUMERIC(3)
   CONSTRAINT C3 CHECK (Sage < 30),
   Ssex  CHAR(2)
   CONSTRAINT C4 CHECK (Ssex IN ( '男', '女')),
   CONSTRAINT StudentKey PRIMARY KEY(Sno)
 );
```

- ✓ 在Student表上建立了5个约束条件，包括主码约束（命名为StudentKey）以及C1、C2、C3、C4四个列级约束。

完整性约束命名子句(续)

［例11］ 建立教师表Teacher，要求每个教师的应发工资不低于3000元. 应发工资为实发工资列sal与扣除项deduct之和。

```
CREATE TABLE Teacher
 (Eno  NUMERIC(4) PRIMARY KEY,
   Ename  CHAR(10)  ,
   Job  CHAR(8)  ,
   Sal  NUMERIC(7,2) ,
   Deduct NUMERIC(7,2) ,
   Deptno NUMERIC(2) ,
  CONSTRAINT EMPFKey Foreign key (Deptno) References
DEPT(Deptno),
  CONSTRAINT C1 CHECK (Sal + Deduct >=3000)
);
```

完整性约束命名子句(续)

2. 修改表中的完整性限制

– 使用ALTER TABLE语句修改表中的完整性限制

- [例12] 去掉[例10]student表中的完整性限制

```
alter table student
drop constraint C4
```

完整性约束命名子句(续)

［例13］ 修改表Student中的约束条件，要求学号改为在900000~999999之间，年龄由小于30改为小于40

– 可以先删除原来的约束条件，再增加新的约束条件

```
ALTER TABLE Student
DROP CONSTRAINT C1;
ALTER TABLE Student
ADD CONSTRAINT C1 CHECK (Sno BETWEEN 900000 AND 999999),
ALTER TABLE Student
DROP CONSTRAINT C3;
ALTER TABLE Student
ADD CONSTRAINT C3 CHECK (Sage < 40);
```

5.6 断言

- 断言的概念

  指定更具一般性的约束。可以定义涉及多个表或聚集操作的比较复杂的完整性约束。断言创建后，任何对断言中所涉及关系的操作都会触发关系数据库管理系统对断言的检查，任何使断言不为真值的操作都会被拒绝执行。

- SQL-92中断言的定义形式

  create assertion <断言名> check <谓词>

5.6 断言

- 例：限制数据库课程最多60名学生选修

```
Create assertion asse_sc_db_num
  Check (60>= ( select count(*)
                  from sc,course
          Where sc.cno=course.cno and course.cname='数据库')
          );
```

5.6 断言

- 例：限制每门课最多60名学生选修

```
Create assertion asse_sc_cnum1
  Check (60>= all ( select count(*)
                    from sc
                     group by cno)
          );
```

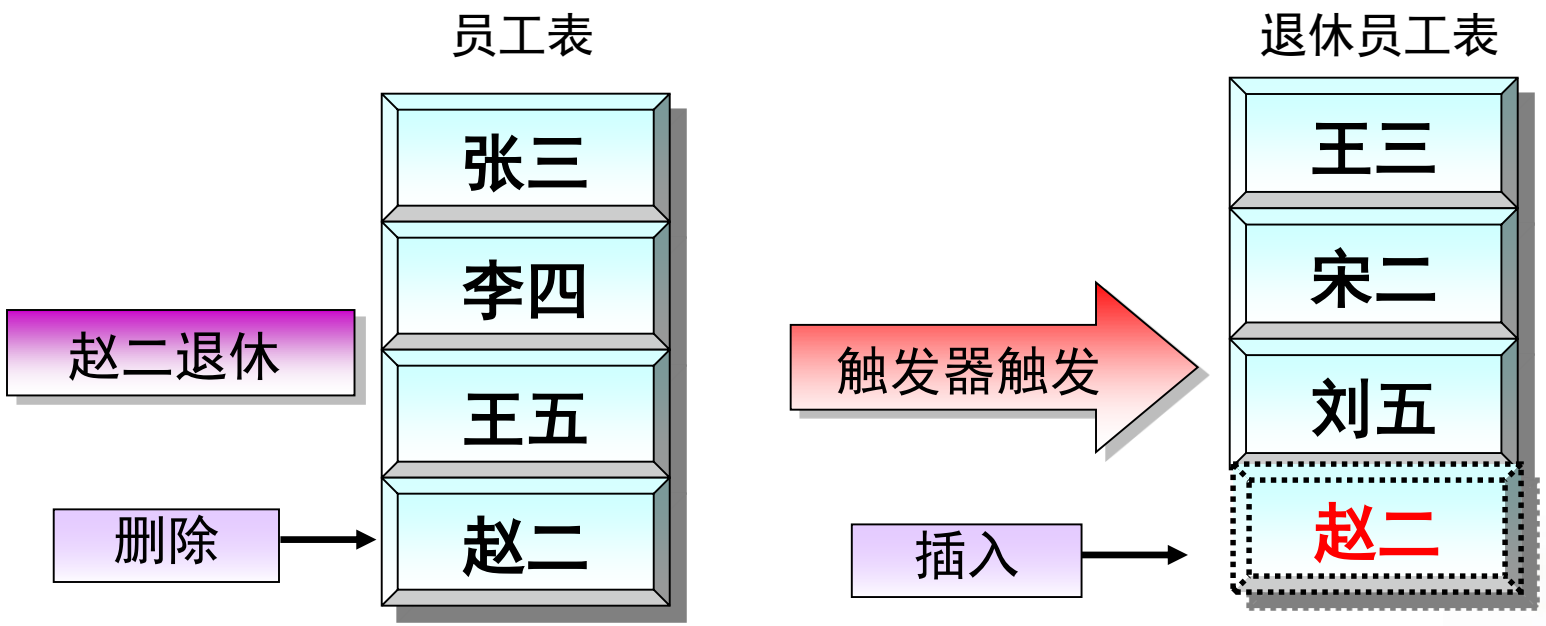
- 删除断言

  Drop assertion<断言名>

什么是触发器

- 触发器（Trigger）是SQL Server数据库中一种特殊类型的存储过程，不能由用户直接调用。
- 它是一个对数据库或数据表进行操作时触发执行的存储过程。
- 用户可以用它来强制实施复杂的业务规则，以此确保数据的完整性。
- 触发器本身是一个事务

什么是触发器



什么是触发器

手机话费计费场景

手机号表

序号	手机号码	当前手机余额	开户时间	等级
1	136***2721	300.00	2016/09/01	4星

话费记录表

No	手机号码	扣款日期	套餐	额外开销
…	******	2019/08/05	48.00	0.00

什么是触发器

手机号表

序号	手机号码	当前手机余额	开户时间	等级
1	136***2721	300.00	2016/09/01	4星

232.00

本月话费扣款
￥68.00=（48+20）

Trigger

话费记录表

No	手机号码	扣款日期	套餐	额外开销
…	******	2019/08/05	48.00	0.00
N	136***2721	2019/09/05	48.00	20.00

触发器：
比约束更为灵活，在多表之间执行特殊的业务规则。

触发器类型

按触发器被激活的时机可以分为以下两种类型：

- AFTER触发器（后触发器）
  - 在引发触发器的语句成功完成之后，执行触发器。
  - 如果操作语句因错误（如违反约束或语法错误）而失败，触发器将不会执行。
- INSTEAD OF触发器（替代触发器）
  - 执行触发器，而不是执行引发触发器的SQL语句，从而替代引发触发器的语句的操作。

触发器类型

按触发事件不同分为：

- DDL（数据定义语言）触发器

  是指当服务器或数据库中发生DDL事件时将启用，是对数据库对象进行操作的 DDL 语句（如 CREATE、ALTER 或 DROP）所激发

- DML（数据操纵语言）触发器

  是指在数据表中发生DML事件时将启用，即指在数据表或视图中修改数据的INSERT、UPDATE、DELETE语句激发

触发器的创建

- CREATE TRIGGER语句语法格式：

语句格式：

```
CREATE TRIGGER  触发器名
ON    { 表名|视图名 }
 { FOR | AFTER | INSTEAD OF }
{ [DELETE][,][INSERT][,][UPDATE] }
 AS
      SQL语句[...n ]
```

创建触发器必须指定的选项：

- 触发器名称；
- 在其上定义触发器的表或视图；
- 触发器何时激发；
- 引发触发器的数据操纵语句；
- 触发器引发后所执行的语句；

触发器的创建

创建触发器注意事项：

- CREATE TRIGGER语句必须是批处理中的第一条语句。
- 只能在当前数据库中创建触发器，一个触发器只能对应一个表。
- AFTER触发器只能定义在表上，不能创建在视图上；INSTEAD OF触发器既可在表上定义，也可在视图上定义 。
- 可以为每个激活触发器的操作（如INSERT、UPDATE或DELETE）创建多个AFTER触发器；对于INSTEAD OF触发器，在一种操作上只能建立一个触发器。

与触发器相关的临时表

在触发器执行的时候，系统产生两个临时表：

inserted 表和deleted 表。

- 这两个表的结构与触发器所作用的表的结构相同
- 这两个表包含了在引发触发器的操作中插入、删除或更新的所有记录
- 这两个表中的数据是只读的，不允许修改
- 触发器执行完成后，系统自动删除这两个表

与触发器相关的临时表

INSERTED表和DELETED表存放的信息

T-SQL语句	deleted表	inserted表
INSERT	空	新增加的记录
UPDATE	旧记录	新记录
DELETE	删除的记录	空

- inserted 表
  - 临时保存了插入或更新后的记录行
  - 可以从inserted表中检查插入的数据是否满足业务需求
  - 如果不满足，则向用户报告错误消息，并回滚插入操作
- deleted 表
  - 临时保存了删除或更新前的记录行
  - 可以从deleted表中检查被删除的数据是否满足业务需求
  - 如果不满足，则向用户报告错误消息，并回滚删除操作

与触发器相关的临时表

分析下面触发器如何执行与执行结果。

```
create trigger liti_1
on 学生 for insert,update
as
 select * from inserted
 select *  from deleted
```

学生

学号	姓名	年龄	所在系
01	小孙	18	计算机
02	小李	19	外语
03	小王	19	生物

执行update 学生 set 年龄=年龄+1

inserted

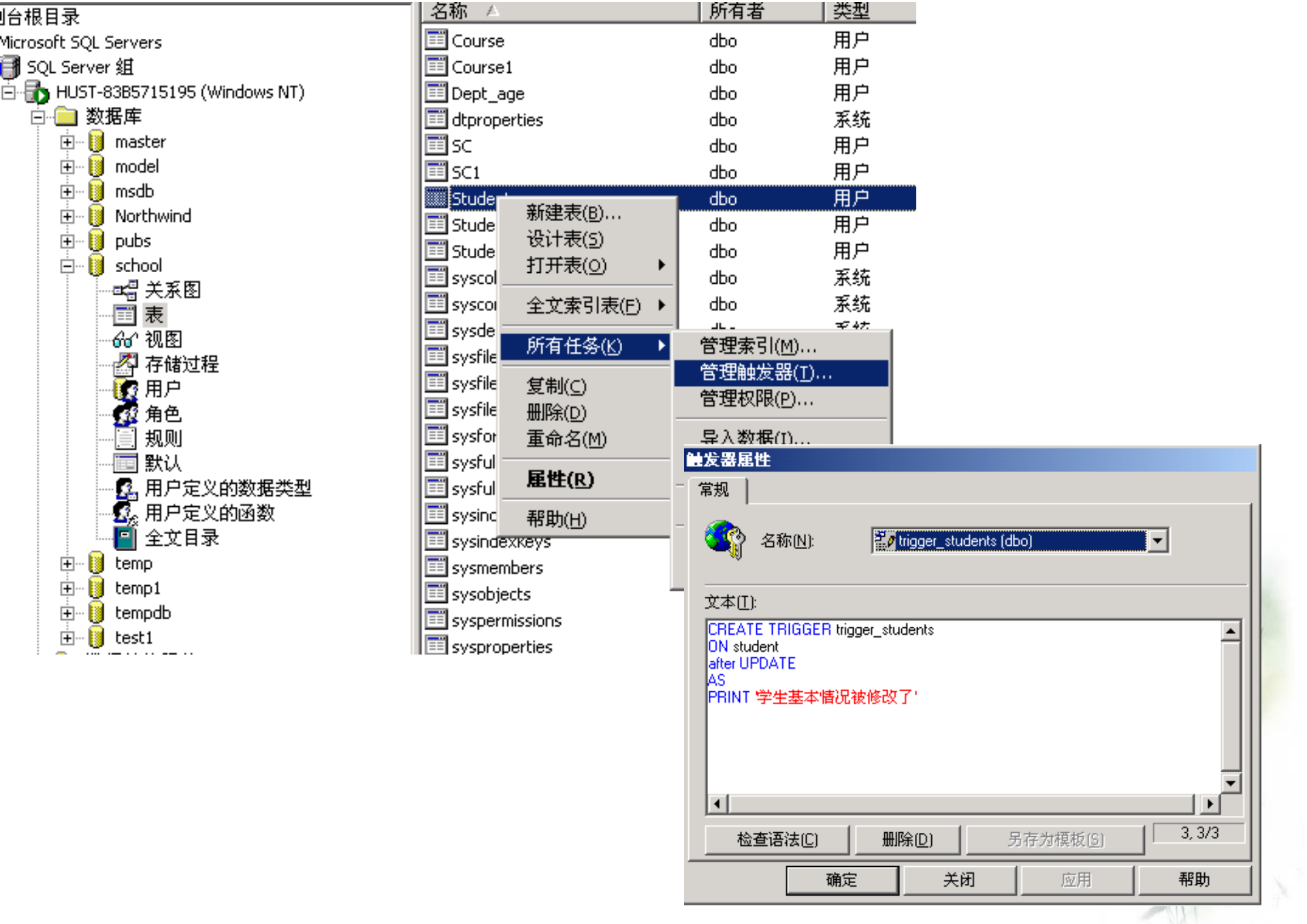
学号	姓名	年龄	所在系
01	小孙	19	计算机
02	小李	20	外语
03	小王	20	生物

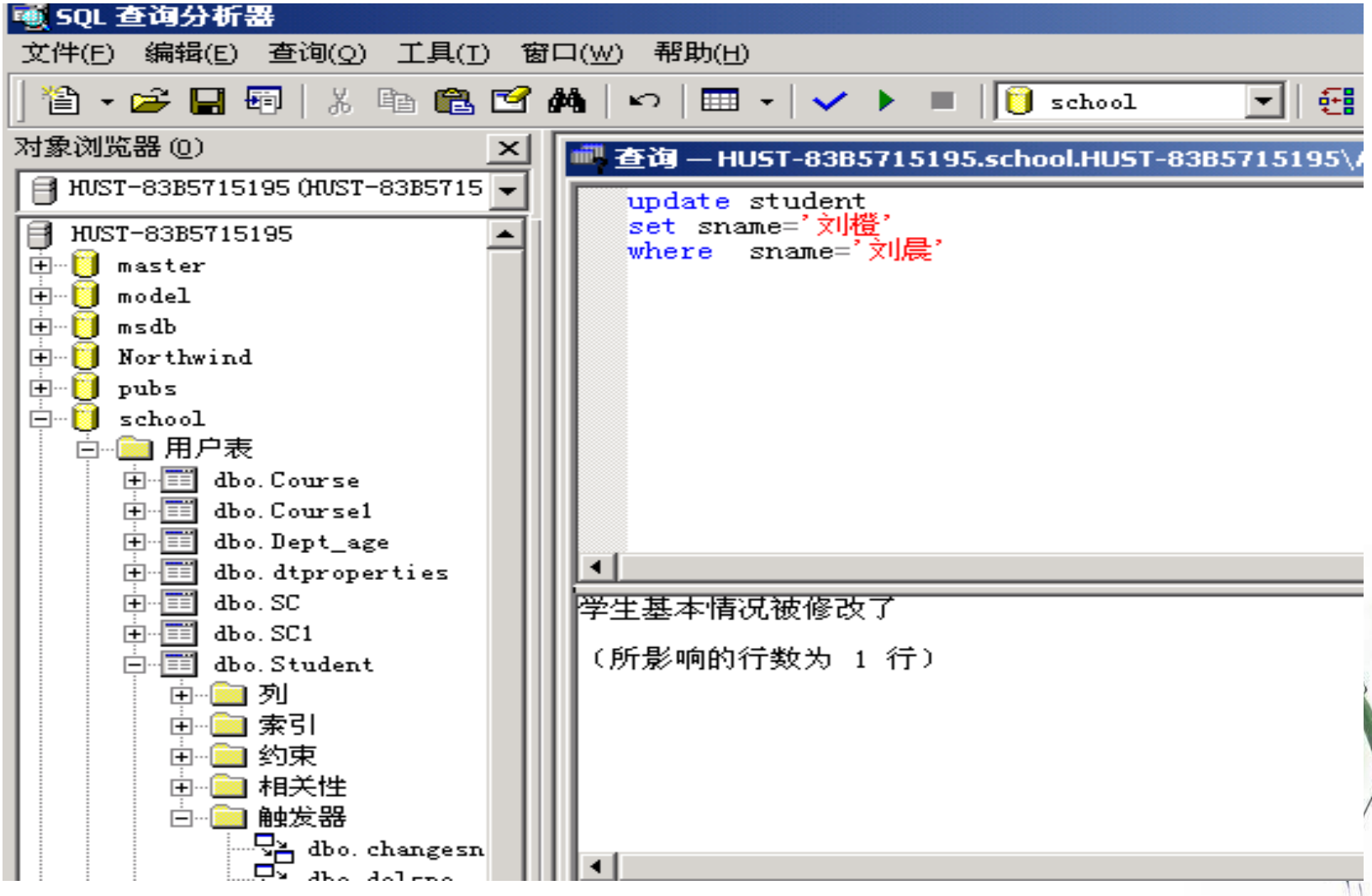
deleted

学号	姓名	年龄	所在系
01	小孙	18	计算机
02	小李	19	外语
03	小王	19	生物

例：创建触发器trigger_students，实现当修改students表中的数据时，显示提示信息“学生基本情况被修改了”。

- CREATE TRIGGER trigger_students
  ON student
  after UPDATE
  AS
  PRINT '学生基本情况被修改了'
  GO


控台根目录
Microsoft SQL Servers
SQL Server 组
HUST-83B5715195 (Windows NT)
数据库
master
model
msdb
Northwind
pubs
school
关系图
表
视图
存储过程
用户
角色
规则
默认
用户定义的数据类型
用户定义的函数
全文目录
temp
temp1
tempdb
test1
名称
所有者
类型
Course dbo 用户
Course1 dbo 用户
Dept_age dbo 用户
dtproperties dbo 系统
SC dbo 用户
SC1 dbo 用户
dbo 用户
dbo 用户
dbo 用户
syscol dbo 系统
sysco dbo 系统
sysde
sysfile
sysfile
sysfile
sysfor
sysful
sysful
sysinc
sysindexkeys
sysmembers
sysobjects
syspermissions
sysproperties
新建表(B)...
设计表(S)
打开表(O)
全文索引表(F)
所有任务(K)
复制(C)
删除(D)
重命名(M)
属性(R)
帮助(H)
管理索引(M)...
管理触发器(T)...
管理权限(P)...
导入数据(I)...
触发器属性
常规
名称(N):
trigger_students (dbo)
文本(T):
CREATE TRIGGER trigger_students
ON student
after UPDATE
AS
PRINT '学生基本情况被修改了'
检查语法(C)
删除(D)
另存为模板(S)
3, 3/3
确定
关闭
应用
帮助


SQL 查询分析器
文件(F) 编辑(E) 查询(Q) 工具(T) 窗口(W) 帮助(H)
school
对象浏览器 (O)
HUST-83B5715195 (HUST-83B5715
HUST-83B5715195
master
model
msdb
Northwind
pubs
school
用户表
dbo.Course
dbo.Course1
dbo.Dept_age
dbo.dtproperties
dbo.SC
dbo.SC1
dbo.Student
列
索引
约束
相关性
触发器
dbo.changesn
查询 — HUST-83B5715195.school.HUST-83B5715195\
update student
set sname='刘橙'
where sname='刘晨'
学生基本情况被修改了
(所影响的行数为 1 行)

例如：若修改**student**表中某学生学号，则表**sc**中与该学生相关的学号自动修改

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

- declare @声明的变量名称 变量类型
  如：declare @intDeclare int

例 数据库compet有如下两个数据表，score表是学生比赛得分表，total表是比赛得分汇总表。创建触发器，当在score表中添加记录时，自动计算学生的成绩总分，并添加到total表中

Score表

stid	stname	item1	item2	item3
0603170108	徐文文	80	85	78
2001160115	邓红艳	78	77	90
2602170105	杨平娟	82	86	74

total表

stid	stname	itemsum
0603170108	徐文文	243
2001160115	邓红艳	245
2602170105	杨平娟	242

例 数据库compet有如下两个数据表，score表是学生比赛得分表，total表是比赛得分汇总表。创建触发器，当在score表中添加记录时，自动计算学生的成绩总分，并添加到total表中

Score表

stid	stname	item1	item2	item3
0603170108	徐文文	80	85	78
2001160115	邓红艳	78	77	90
2602170105	杨平娟	82	86	74

total表

stid	stname	itemsum
0603170108	徐文文	243
2001160115	邓红艳	245
2602170105	杨平娟	242

```
CREATE TRIGGER addsore ON SCore
  FOR INSERT
  AS
    DECLARE @it1 int, @it2 int, @it3 int,@itsum int, @stno char(10), @sname char(20)
    SELECT @stno=stid , @sname =stname, @it1=item1, @it2=item2, @it3=item3
    FROM inserted
    SET @itsum=@it1+@it2+@it3
    INSERT INTO total VALUES(@stno,@sname,@itsum)
```

2、查看触发器数据

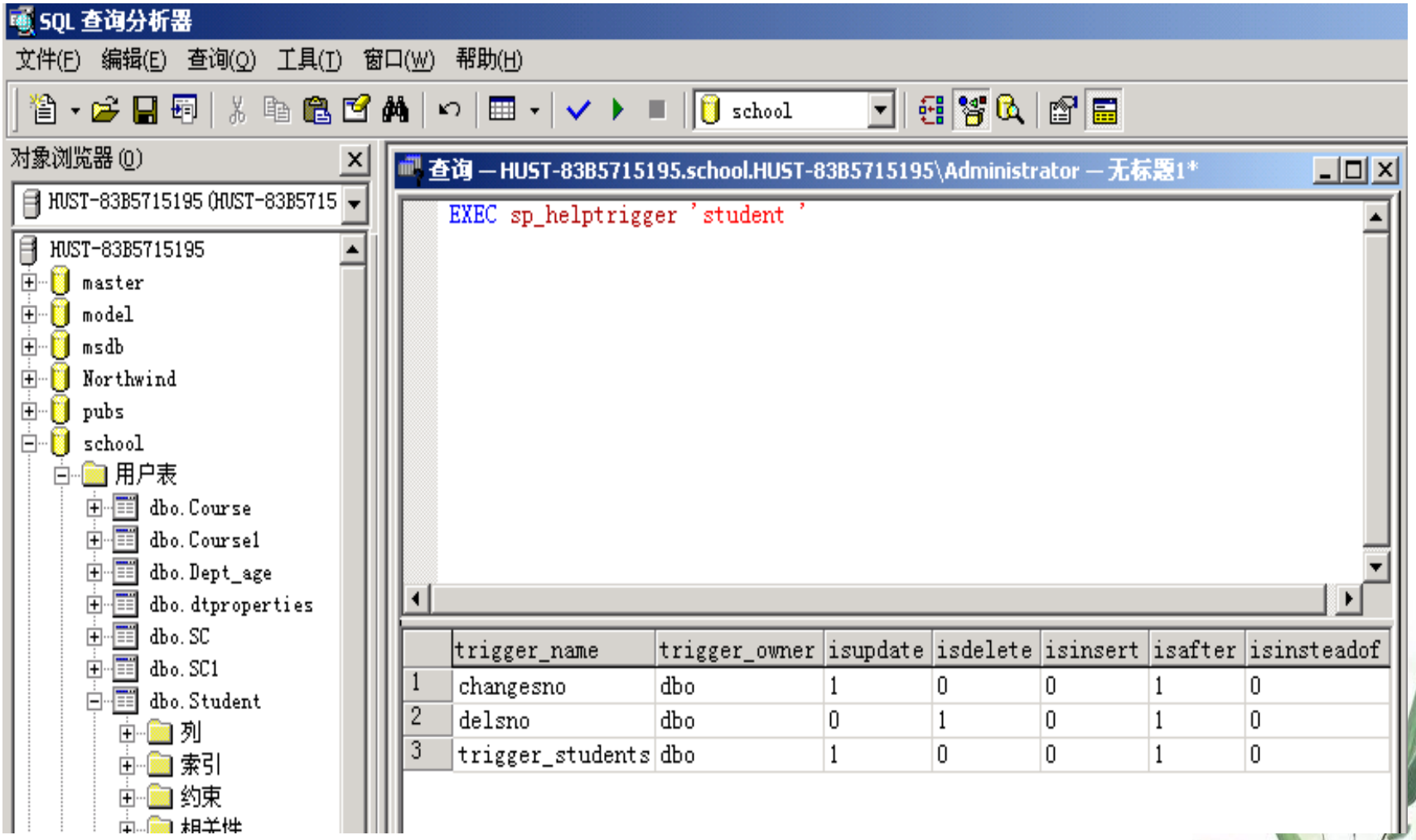
- 1）查看触发器
- 格式

  EXEC sp_helptrigger ‘table’ [,’type’]

- 例：查看刚才所建的触发器：

  EXEC sp_helptrigger 'student','changesno'

  或者EXEC sp_helptrigger 'student '

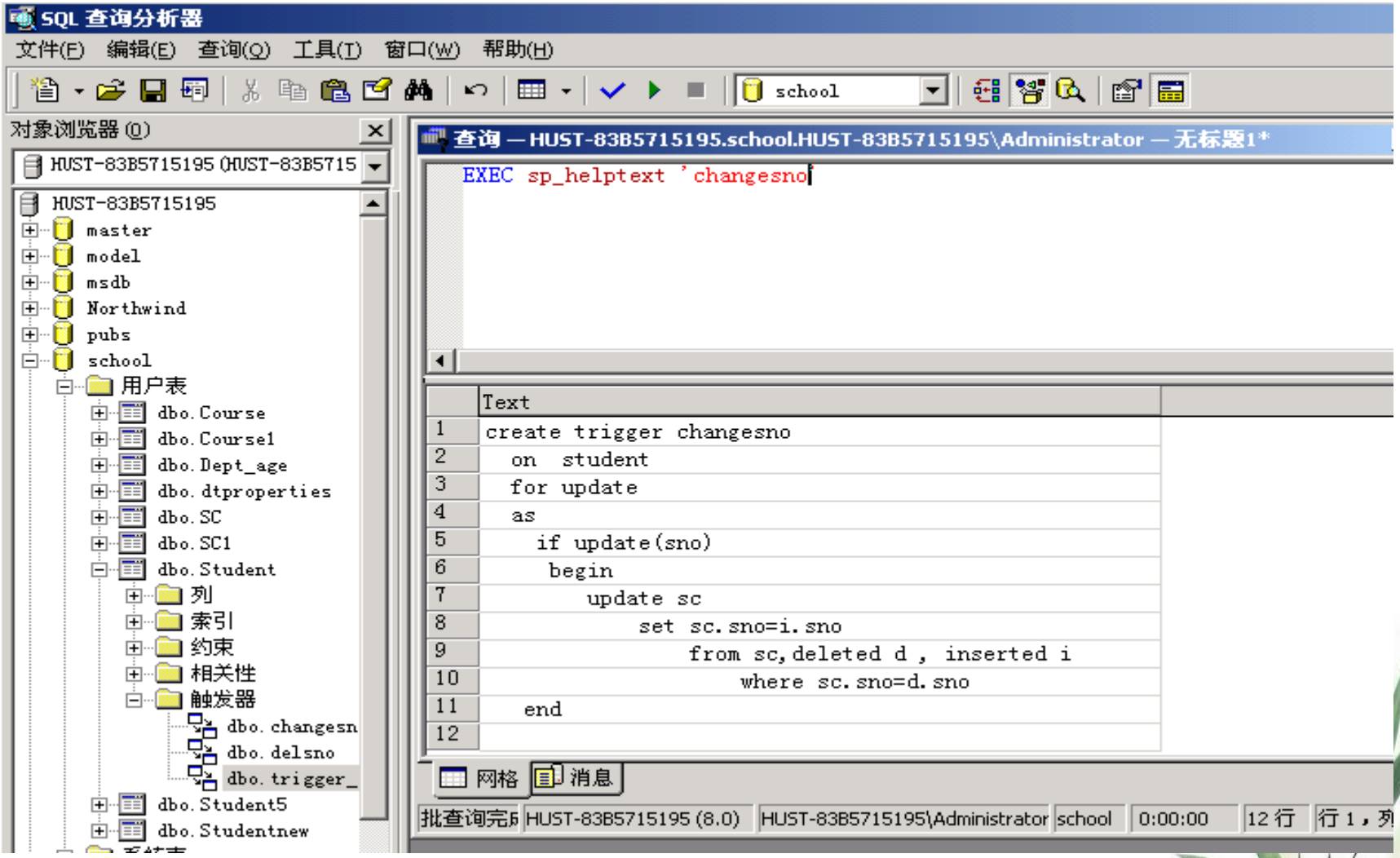

SQL 查询分析器
文件(F) 编辑(E) 查询(Q) 工具(T) 窗口(W) 帮助(H)
school
对象浏览器(O)
HUST-83B5715195 (HUST-83B5715
HUST-83B5715195
master
model
msdb
Northwind
pubs
school
用户表
dbo.Course
dbo.Course1
dbo.Dept_age
dbo.dtproperties
dbo.SC
dbo.SC1
dbo.Student
列
索引
约束
查询 — HUST-83B5715195.school.HUST-83B5715195\Administrator — 无标题1*
EXEC sp_helptrigger 'student'
trigger_name trigger_owner isupdate isdelete isinsert isafter isinsteadof
1 changesno dbo 1 0 0 1 0
2 delsno dbo 0 1 0 1 0
3 trigger_students dbo 1 0 0 1 0

- 2）查看触发器的定义文本

触发器的定义文本存储在系统表syscomments中，可用存储过程查看。

格式：

EXEC sp_helptext ‘trigger_name’


SQL 查询分析器
文件(F) 编辑(E) 查询(Q) 工具(T) 窗口(W) 帮助(H)
school
对象浏览器 (O)
HUST-83B5715195 (HUST-83B5715
HUST-83B5715195
master
model
msdb
Northwind
pubs
school
用户表
dbo.Course
dbo.Course1
dbo.Dept_age
dbo.dtproperties
dbo.SC
dbo.SC1
dbo.Student
列
索引
约束
相关性
触发器
dbo.changesn
dbo.delsno
dbo.trigger_
dbo.Student5
dbo.Studentnew
查询 — HUST-83B5715195.school.HUST-83B5715195\Administrator — 无标题1*
EXEC sp_helptext 'changesno
Text
1 create trigger changesno
2 on student
3 for update
4 as
5 if update(sno)
6 begin
7 update sc
8 set sc.sno=i.sno
9 from sc,deleted d , inserted i
10 where sc.sno=d.sno
11 end
12
网格
消息
批查询完成
HUST-83B5715195 (8.0)
HUST-83B5715195\Administrator
school
0:00:00
12 行
行 1 , 列

3、 删除触发器

- 删除触发器 DROP TRIGGER

语法格式：

DROP TRIGGER trigger_name[,…n]

- 例：删除触发器trigger_come, trigger_leave

  Drop trigger trigger_come, trigger_leave