数据库系统概论

An Introduction to Database System

第五章　数据库完整性

数据库完整性

- 数据库的完整性
  - 数据的正确性和相容性

- 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

第五章 数据库完整性

5.1.1 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*/
    );
```

第五章 数据库完整性

5.1 实体完整性

5.2 参照完整性

5.3 用户定义的完整性

5.4 完整性约束命名字句

*5.5 域中的完整性限制

5.6 触发器

5.7 小结

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 REFERENCES Student(Sno)，
       Cno    CHAR(4)  NOT NULL REFERENCES Course(Cno) ，
       Grade   SMALLINT，
       PRIMARY KEY (Sno，  Cno));
```

或

```
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)
);
```

5.2 参照完整性

- **5.2.1** 参照完整性定义
- **5.2.2** 参照完整性检查和违约处理

参照完整性检查和违约处理

可能破坏参照完整性的情况及违约处理

被参照表（例如Student）	参照表（例如SC）	违约处理
可能破坏参照完整性 ←	插入元组	拒绝
可能破坏参照完整性 ←	修改外码值	拒绝
删除元组 →	可能破坏参照完整性	拒绝/级连删除/设置为空值
修改主码值 →	可能破坏参照完整性	拒绝/级连修改/设置为空值

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

违约处理(续)

［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*/
    );
```

第五章 数据库完整性

5.1 实体完整性

5.2 参照完整性

5.3 用户定义的完整性

5.4 完整性约束命名字句

*5.5 域中的完整性限制

5.6 触发器

5.7 小结

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

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

5.3.1 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）

属性上的约束条件的定义(续)

- 1.不允许取空值

［例5］ 在定义SC表时，说明Sno、Cno、Grade属性不允许取空值。

```
CREATE TABLE SC
  （Sno  CHAR(9)  NOT NULL，
    Cno  CHAR(4)  NOT NULL，
    Grade  SMALLINT NOT NULL，
    PRIMARY KEY (Sno， Cno)，
  /* 如果在表级定义实体完整性，隐含了Sno，Cno不允许取空值，则在列级不允许取空值的定义就不必写了 */
  ）；
```

属性上的约束条件的定义(续)

- 2.列值唯一

［例6］ 建立部门表DEPT，要求部门名称Dname列取值唯一，部门编号Deptno列为主码

```
CREATE TABLE DEPT
  (Deptno  NUMERIC(2)，
    Dname  CHAR(9)  UNIQUE，/*要求Dname列值唯一*/
    Location  CHAR(10)，
    PRIMARY KEY (Deptno)
  );
```

属性上的约束条件的定义(续)

- 3. 用CHECK短语指定列值应该满足的条件

［例7］ Student表的Ssex只允许取“男”或“女”。

```
CREATE TABLE Student
    (Sno  CHAR(9) PRIMARY KEY，
     Sname CHAR(8) NOT NULL，
      Ssex  CHAR(2)  CHECK (Ssex IN (‘男’，‘女’))，
       /*性别属性Ssex只允许取'男'或'女' */
      Sage  SMALLINT，
     Sdept  CHAR(20)
     );
```

5.3.3 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］ 当学生的出生日期必须小于入学日期

```
CREATE TABLE Student
   (Sno    CHAR(9)，
    Sname  CHAR(8) NOT NULL，
    Ssex    CHAR(2)，
    Sage   SMALLINT，
    Sdept  CHAR(20)，
    Bdate  datetime,
    Idate    datetime,
    PRIMARY KEY (Sno)，
    CHECK (Bdate < Idate )
    /*定义了元组中Bdate和 Idate两个属性值之间的约束条件*/
   )；
```

第五章 数据库完整性

5.1 实体完整性

5.2 参照完整性

5.3 用户定义的完整性

5.4 完整性约束命名子句

***5.5** 域中的完整性限制

5.6 触发器

5.7 小结

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)

- **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

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);
```

第五章 数据库完整性

5.1 实体完整性

5.2 参照完整性

5.3 用户定义的完整性

5.4 完整性约束命名字句

*5.5 域中的完整性限制

5.6 触发器

5.7 小结

触发器

- 触发器是用编程的方法实现复杂的商业规则，它可以实现一般的数据完整性约束实现不了的复杂的完整性约束
- 不需要由用户调用执行，而是当用户对表中的数据进行UPDATE、INSERT或DELETE操作时自动触发执行的。

5.6.1 定义触发器

- CREATE TRIGGER语法格式

CREATE TRIGGER <触发器名>

{BEFORE | AFTER} <触发事件> ON <表名>

FOR EACH {ROW | STATEMENT}

[WHEN <触发条件>]

<触发动作体>

定义触发器(续)

- 例如,假设在［例11］的TEACHER表上创建了一个AFTER UPDATE触发器。如果表TEACHER有1000行，执行如下语句：

```
UPDATE TEACHER SET Deptno=5;
```

  - 如果该触发器为语句级触发器，那么执行完该语句后，触发动作只发生一次
  - 如果是行级触发器，触发动作将执行1000次

定义触发器(续)

［例18］ 定义一个BEFORE行级触发器，为教师表Teacher定义完整性规则“教授的工资不得低于4000元，如果低于4000元，自动改为4000元”。

```
CREATE TRIGGER Insert_Or_Update_Sal
    BEFORE INSERT OR UPDATE ON Teacher
    /*触发事件是插入或更新操作*/
    FOR EACH ROW          /*行级触发器*/
    BEGIN                 /*定义触发动作体，是PL/SQL过程块*/
        IF (:new.Job='教授') AND (:new.Sal < 4000) THEN
        :new.Sal :=4000;
        END IF;
    END;
```

定义触发器(续)

```
create trigger incordernitems after insert on lineitems
for each row
      begin
         update orders set n_items=n_items+1 where
   ordno=:old.ordno;
    end

create trigger decordernitems after delete on lineitems
for each row
      begin
         update orders set n_items=n_items-1 where
   ordno=:old.ordno;
    end
```

SQL server中创建触发器语法格式

CREATE TRIGGER 触发器名称

ON 表名 { FOR | AFTER | INSTEAD OF } { [INSERT] [,] [DELETE] [,] [UPDATE] }

AS SQL 语句 [... n]

注：SQL server中inserted相当于课本的new. deleted相当于课本的old

例1 :创建触发器,限制将SC表中不及格学生的成绩改为及格。

```
CREATE TRIGGER tri_grade
ON SC FOR UPDATE
AS
 IF UPDATE(Grade)
    IF EXISTS(SELECT * FROM INSERTED JOIN
DELETED ON INSERTED.Sno = DELETED.Sno WHERE
INSERTED.Grade >= 60 AND DELETED.Grade < 60)
   ROLLBACK /* 这里ROLLBACK代表回滚,取消更新操作
```

例2: 创建实现限制最低工资必须小于最高工资的触发器。

```
CREATE TRIGGER tri_job_salary2
ON 工作表 FOR INSERT, UPDATE
AS
IF EXISTS(SELECT * FROM INSERTED WHERE 最低工资 >= 最高工资 )
BEGIN
    PRINT '最低工资必须小于最高工资'
    ROLLBACK
END
```

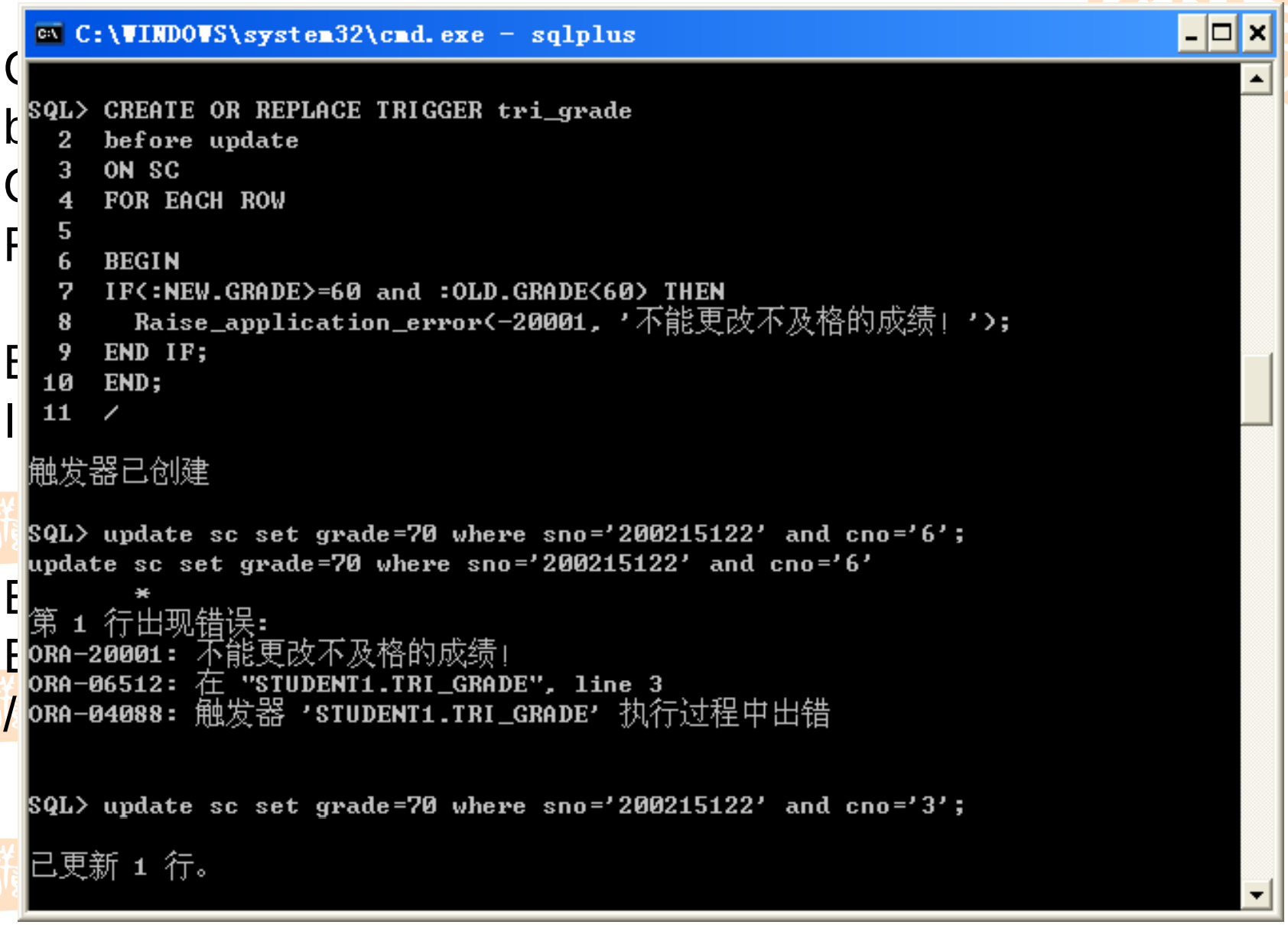
例3:创建实现限制雇员的工资必须在工作表的相应工作的最低工资和最高工资之间。

```
CREATE TRIGGER tri_emp_salary
ON 雇员表 FOR INSERT, UPDATE
AS
    IF EXISTS (SELECT * FROM INSERTED a JOIN 工作表 bON a.工作编号 = b.工作编号 WHERE 工资 NOT BETWEEN 最低工资 AND 最高工资 )
     ROLLBACK
```

Oracle 创建触发器语法格式

```
CREATE [OR REPLACE] TIGGER 触发器名
[BEFORE | AFTER ] 触发事件 ON 表名
[FOR EACH ROW]  [WHEN 触发条件]
[DECLARE
  说明部分]
BEGIN
  执行语句
END
```

用Oracle 实现限制将SC表中不及格学生的成绩改为及格



5.6 触发器

- **5.6.1** 定义触发器
- **5.6.2** 激活触发器
- **5.6.3** 删除触发器

5.6.2 激活触发器

- 触发器的执行，是由触发事件激活的，并由数据库服务器自动执行
- 一个数据表上可能定义了多个触发器
  - 同一个表上的多个触发器激活时遵循如下的执行顺序：
    - （1） 执行该表上的BEFORE触发器；
    - （2） 激活触发器的SQL语句；
    - （3） 执行该表上的AFTER触发器。

5.6 触发器

- **5.6.1** 定义触发器
- **5.6.2** 激活触发器
- **5.6.3** 删除触发器

5.6.3 删除触发器

- 删除触发器的SQL语法：

  DROP TRIGGER <触发器名> ON <表名>;

- 触发器必须是一个已经创建的触发器，并且只能由具有相应权限的用户删除。

［例21］ 删除教师表Teacher上的触发器Insert_Sal

```
DROP TRIGGER Insert_Sal ON Teacher;
```

注：ORACLE 和 SQL SERVER 2000 删除触发器的SQL语法

```
DROP TRIGGER <触发器名>
```

- 例：删除tri_grade触发器。

```
DROP TRIGGER tri_grade
```

第五章 数据库完整性

5.7 小结

- 数据库的完整性是为了保证数据库中存储的数据是正确的

- RDBMS完整性实现的机制
  - 完整性约束定义机制
  - 完整性检查机制
  - 违背完整性约束条件时RDBMS应采取的动作

作业

- 3,6
- 为本章的知识点出一道考试题，并给出正确的答案。