華中科技大學

数据库系统原理

第5章　数据库完整性

学习通，交流

基础篇

第1章 绪论
第2章 关系数据库*3
第3章 标准语言SQL*3
第4章 数据库安全性
第5章 数据库完整性
实验1

设计与应用开发篇

第6章 关系数据理论*2
第7章 数据库设计
实验2
第8章 数据库编程

系统篇

第9章 *关系查询处理和优化
实验3
第10章 数据库恢复技术
第11章 并发控制
实验4

内容提要

1 实体完整性

2 参照完整性

3 用户定义的完整性

4 完整性约束命名

6 断言

7 触发器

0. 概述

- 实体完整性
  - 关系模型中以主码作为唯一性标识，主属性不能取空值
- 参照完整性
  - 外码或者取空值，或者等于对应元组的某个存在的主码值
- 用户定义的完整性
  - 提供定义和检验这类完整性的机制

1. 实体完整性

[例5.1] 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)
);
```

（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)
);
```

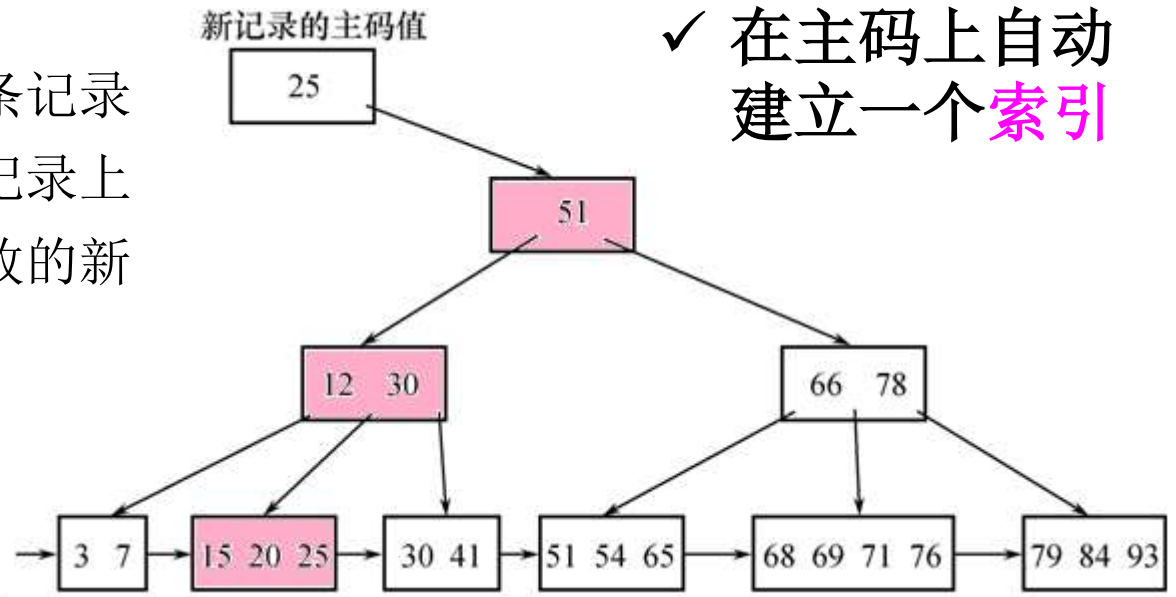
[例5.2] 将SC表中的Sno，Cno属性组定义为码

```
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—实现

- 插入或对主码列进行更新操作时，进行实体完整性规则检查：
  - 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
  - 检查主码值是否唯一，如果不唯一则拒绝插入或修改

- 全表扫描
  - 依次判断表中每一条记录的主码值与将插入记录上的主码值（或者修改的新主码值）是否相同



✓ 在主码上自动建立一个索引

2. 参照完整性

例如，关系SC中（Sno，Cno）是主码。Sno，Cno分别参照Student表的主码和Course表的主码

[例5.3]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—实现

- 如，表SC和Student有四种可能破坏参照完整性的情况：
  - SC表中增加一个元组，Sno属性的值在表Student中找不到对应元组
  - 修改SC表中的一个元组，修改后该元组的Sno属性的值在表Student中找不到一个元组，其Sno属性的值与之相等。
  - 从Student表中删除一个元组，造成SC表中原对应的元组不能参照
  - 修改Student表中一个元组的Sno属性，造成SC表中某些元组的Sno属性的值在表Student中找不到一个元组，其Sno属性的值与之相等

参照完整性—实现

表5.1 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

参照完整性—实现

[例5.4] 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*/
);
```

3. 用户定义完整性

[例5.5] 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 可以不写 * /
);
```

- 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，关系数据库管理系统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

用户定义完整性

[例5.6]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 NOT NULL,
        Location  CHAR(10),          PRIMARY KEY (Deptno)       );
```

[例5.7] 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)
    );
```

用户定义完整性

[例5.9]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

4. 完整性约束命名

CONSTRAINT <完整性约束条件名><完整性约束条件>

- <完整性约束条件>包括NOT NULL、UNIQUE、PRIMARY KEY短语、FOREIGN KEY短语、CHECK短语等

[例5.10]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)
  );
```

完整性约束命名

[例5.11]建立教师表TEACHER，要求每个教师的应发工资不低于3000元。应发工资是工资列Sal与扣除项Deduct之和。

```
CREATE TABLE TEACHER
    (   Eno     NUMERIC(4)  PRIMARY KEY     /*在列级定义主码*/
        Ename   CHAR(10),
        Job     CHAR(8),
        Sal     NUMERIC(7,2),
        Deduct  NUMERIC(7,2),
        Deptno  NUMERIC(2),
        CONSTRAINT TEACHERFKey FOREIGN KEY (Deptno)
                                REFERENCES DEPT(Deptno),
        CONSTRAINT C1 CHECK (Sal + Deduct >= 3000)
    );
```

完整性约束命名

- 使用ALTER TABLE语句修改表中的完整性限制

[例5.12]去掉例5.10 Student表中对性别的限制。

```
ALTER TABLE Student    DROP CONSTRAINT C4;
```

[例5.13] 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

```
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);
```

6. 断言Assertion

- SQL中，可以使用 CREATE ASSERTION语句，通过声明性断言来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。
- 断言创建以后，任何对断言中所涉及的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

创建断言的语句格式

- **CREATE ASSERTION<断言名><CHECK 子句>**
- **每个断言都被赋予一个名字，<CHECK 子句>中的约束条件与WHERE子句的条件表达式类似。**

断言Assertion

[例5.18] 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(*)        /*此断言的谓词涉及聚集操作count的SQL语句*/
        From Course,SC
        Where SC.Cno=Course.Cno and Course.Cname ='数据库')  );
```

[例5.20]限制每个学期每一门课程最多60名学生选修

首先需要修改SC表的模式，增加一个“学期（TERM）”属性

```
ALTER TABLE SC ADD TERM DATE;
```

然后，定义断言：

```
CREATE ASSERTION ASSE_SC_CNUM2
    CHECK(60 >= ALL (SELECT count(*)   FROM SC
                     GROUP by cno,TERM)
                );
```

✓ DROP ASSERTION <断言名>;

7. 触发器Trigger

- 触发器（Trigger）是用户定义在关系表上的一类由事件驱动的特殊过程
  - 触发器保存在数据库服务器中
  - 任何用户对表的增、删、改操作均由服务器自动激活相应的触发器
  - 触发器可以实施更为复杂的检查和操作，具有更精细和更强大的数据控制能力

（1）表的拥有者才可以在表上创建触发器

（2）触发器只能定义在基本表上，不能定义在视图上

触发器Trigger—定义

- CREATE TRIGGER语法格式

```
CREATE TRIGGER <触发器名>
{BEFORE | AFTER} <触发事件> ON <表名>
REFERENCING NEW|OLD ROW AS<变量>
FOR EACH {ROW | STATEMENT}
[WHEN <触发条件>]<触发动作体>
```

- 触发器又叫做事件-条件-动作（event-condition-action）规则。
- **当特定的系统事件发生时，对规则的条件进行检查，如果条件成立则执行规则中的动作，否则不执行该动作。规则中的动作体可以很复杂，通常是一段PL/SQL过程块 或 SQL存储过程。**

触发器Trigger—定义

- 触发事件
  - 触发事件可以是INSERT、DELETE或UPDATE
    也可以是这几个事件的组合
  - 还可以UPDATE OF<触发列，...>，修改哪些列时激活触发器
  - AFTER/BEFORE是触发的时机
    - AFTER表示在触发事件的操作执行之后激活触发器
    - BEFORE表示在触发事件的操作执行之前激活触发器
- 触发器类型
  - 行级触发器（FOR EACH ROW）
  - 语句级触发器（FOR EACH STATEMENT）
  - 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，触发动作将每行执行1次

触发器Trigger—定义

- 触发条件
  - 触发器被激活时，只有当触发条件为真时触发动作体才执行；否则触发动作体不执行。
  - 如果省略WHEN触发条件，则触发动作体在触发器激活后立即执行
- 触发动作体
  - 触发动作体可以是一个匿名SQL过程块，也可以是存储过程的调用
  - 如果是行级触发器，用户都可以在过程体中使用NEW和OLD引用事件之后的新值和事件之前的旧值
  - 如果是语句级触发器，则不能使用NEW或OLD进行引用
  - 如果触发动作体执行失败，激活触发器的事件就会终止执行，目标表或触发器可能影响的其他对象不发生任何变化

触发器Trigger—定义

[例5.21]当对表SC的Grade属性进行修改时，若分数增加了10%则将此次操作记录到下面表中：

SC_U(Sno,Cno,Oldgrade,Newgrade)

其中Oldgrade是修改前的分数，Newgrade是修改后的分数。


```
CREATE TRIGGER  SC_T
        AFTER UPDATE OF Grade ON SC
REFERENCING
      OLD row  AS  OldTuple,          NEW row AS  NewTuple
FOR EACH ROW
        WHEN (NewTuple.Grade >= 1.1*OldTuple.Grade)
        INSERT INTO SC_U(Sno,Cno,OldGrade,NewGrade)
          VALUES(OldTuple.Sno,OldTuple.Cno,OldTuple.Grade,NewTuple.Grade)
```

触发器Trigger—定义

[例5.22] 将每次对表Student的插入操作所增加的学生个数记录到表StudentInsertLog中。

```
CREATE TRIGGER Student_Count
AFTER INSERT ON Student
        /*指明触发器激活的时间是在执行INSERT后*/
REFERENCING
        NEW TABLE AS DELTA
FOR EACH STATEMENT
    /*语句级触发器, 即执行完INSERT语句后下面的触发动作体才执行一次*/
        INSERT INTO StudentInsertLog (Numbers)
        SELECT COUNT(*) FROM DELTA
```

触发器Trigger—定义

[例5.23] 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;
```

delimiter MySQL中的命令

- 告诉mysql解释器，该段命令是否已经结束了，mysql是否可以执行了。即改变输入结束符。
- 默认情况下，delimiter是分号“;”。
- 在命令行客户端中，如果有一行命令以分号结束，那么回车后mysql将会执行该命令。但有时候，不希望MySQL这么做。因为可能输入较多的语句，且语句中包含有分号。

```
drop trigger s_t.SC_trigger ;
delimiter //
create trigger SC_trigger after update on sc for each row
  begin
        if new.grade >= 95 then
            update student   set scholarship='是'
              where (scholarship='否') AND (student.sno=new.sno) ;
        end if;
end;//
```

触发器Trigger—触发、删除

- 触发器的执行，是由触发事件激活的
- 一个数据表上可能定义了多个触发器，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。
- 删除触发器的SQL语法

  DROP TRIGGER <触发器名> ON <表名>;
- 只能由具有相应权限的用户删除。

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

- **关系数据库管理系统完整性实现的机制**

  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采取的动作