05 数据库完整性

数据库的完整性

- 数据的正确性：数据符合现实世界语义，反映了当前实际状况
- 数据的相容性：数据库同一对象在不同关系表中的数据是符合逻辑的

为维护数据库的完整性，数据库管理系统必须：

- 提供定义完整性约束条件的机制
- 提供完整性检查的方法
- 违约处理

数据的完整性和约束性不相同

- 数据的完整性
  - 防止数据库中存在不符合语义的数据，即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

5.1 实体完整性

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

将Student表中的Sno属性定义为码

```SQL
/*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)
        )
/*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)
        )
```

将SC表中的Sno，Cno属性组定义为码

```SQL
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*/
        )
```

5.1.2 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

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

检查主码值是否唯一的一种方法是进行`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
- **表扫描的缺点**：十分耗时
- 为避免对基本表进行全表扫描，RDBMS核心一般都在主码上自动建立一个`索引` ——提高效率

5.2 参照完整性

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
- *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

定义SC中的参照完整性

```SQL
CREATE TABLE SC
        (Sno CHAR(9) NOT NULL,
        Cno CHAR(4) NOT NULL,
        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*/
        )
```

5.2.2 参照完整性检查和违约处理

- 一个参照完整性将两个表中的相应元组联系起来
- 对被参照和参照表进行增删改操作时有可能破坏参照完整性，必须进行检查

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

参照完整性违约处理

- **拒绝执行** `NO ACTION` ：不允许该操作执行，一般为默认策略
- **级联操作** `CASCADE` ：当删除或修改被参照表的一个元组造成了与参照表的不一致，则删除或修改参照表中的所有造成不一致的元组
- **设置为空值** `SET-NULL` ：当删除或修改被参照表的一个元组造成了不一致，则将参照表中的所有造成不一致的元组的对应属性设置为空值
- 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

```SQL
CREATE TABLE SC
        (Sno CHAR(9) NOT NULL,
        Cno CHAR(4) NOT NULL,
        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*/
        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*/
        )
```

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
- RDBMS提供，而不必由应用程序承担

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`

```SQL
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*/
        );

create table DEPT
        (Deptno numeric(2),
        Dname char(9) unique not null. /*要求Dname列值唯一且不能取空值*/
        Location char(10),
        primary key (Deptno)
        );

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'*/
        )
```

5.3.2 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

5.3.3 元组上的约束条件的定义

- 在 `Create Table` 时可以使用 `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

当学生的性别是男时，姓名不能以Ms.打头

```SQL
Create Table Student
        (Sno char(9) primary key,
        Sname char(8) not null,
        Ssex char(2) check (Ssex in ('男', '女')),
        Sage smallint,
        check (Ssex='女' or Sname not like 'Ms.%')
        /*性别是女性的元组都能通过检查，因为Ssex='女'成立*/
        /*当性别是男性时，要通过检查则名字不能以Ms.打头*/
        )
```

逻辑表达：若p则q，等价为**非p并q，即 非p or q**

5.3.4 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

1. 完整性约束命名字句

`constraint <完整性约束条件名><完整性约束条件>`

完整性约束条件： `primary key`、`foreign key`、`check`、`not null`、`unique`

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、女

```SQL
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' or '女')),
        constraint StudentKey primary key (Sno)
        )
```

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

```SQL
Alter Table Student Drop constraint C4;
/*修改：先删除，再增加新的约束条件*/
Alter Table Student Drop constraint C3;
Alter Table Student Add constraint C3 check (Sage < 40)
```

5.5 域中的完整性限制

sql支持域的概念，并可以用 `Create Domain` 语句建立一个域以及该域应满足的完整性约束条件

```SQL
Create Domain GenderDomain char(2) check (value in ('男' or '女'))
```

- SQL Sever中可以用户定义数据类型，其是系统提供的数据类型的别名，`sp_addtype`
- 在创建了用户数据类型之后，可以在 `Create Table` 或 `Alter Table` 中使用它，也可以将默认值和规则绑定到用户定义的数据类型

创建不允许空值的用户定义数据类型

```SQL
exec sp_addtype ssn, 'varchar(11)', 'not null'
```

- DBMS禁止两种数据类型之间进行没有定义的操作，所以不可将两种用户定义的数据类型相加

规则约束

- 创建：`CREATE RULE r1 as ...`
- 绑定：`sp_bindrule`
- 松绑：`sp_unbindrule`

```SQL
CREATE RULE range_rule as @range >= 1000 and @range <20000;
EXEC sp_bindrule 'range_rule', 'employees.[bonus]';
EXEC sp_unbindrule 'employees.bonus'
```

- 绑定到列的规则始终优于绑定到数据类型的规则

5.6 断言

- 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的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
- 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
  - 删除断言：`DROP ASSERTION <断言名>`

限制每一门课程最多60人选修

```SQL
CREATE ASSERTION Asse_SC_Cnum1 check
        (60 >= all
                (select count(*) from SC group by Cno))
```

5.7 触发器

触发器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

5.7.1 定义触发器

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>]`

触发器：事件-条件-动作规则

- 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

语法说明

1. 表的`拥有者`才可以在表上创建触发器
2. 触发器名
   - 触发器名可以包含模式名，也可以不包含模式名
   - 同一模式下，触发器名必须是唯一的
   - 触发器名和表名必须在同一模式下

3. 表名
    - 触发器只能定义在表上，不能定义在视图上
    - 当基本表的数据发生变化，将激活定义在该表上相应触发事件的触发器
4. 触发事件
    1. 触发事件可以是`INSERT`、`DELETE`或`UPDATE`，也可以是这几个事件的组合
    2. 还可以`UPDATE OF <触发列>`，即进一步指明修改哪些列时激活触发器
    3. `AFTER/BEFORE`是触发的时机
        1. `AFTER`表示在触发事件的操作执行之后激活触发器
        2. `BEFORE`表示在触发事件的操作执行之前激活触发器
5. 触发器类型
    1. `FOR EACH ROW`行级触发器
    2. `FOR EACH STATEMENT`语句级触发器
6. 触发条件
    1. 触发器被激活时，只有当触发条件为真时触发动作体才被执行，否则触发动作体不执行
    2. 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
7. 触发动作体
    1. 可以是一个匿名PL/SQL过程块，也可以是对已创建存储过程的调用
    2. 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
    3. 如果是语句级触发器，则不能在触发动作体中使用NEW或OLD进行引用
    4. 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，触发器的目标表或触发器可能影响的其他对象不发生任何变化

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

```SQL
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)
```

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

```SQL
create trigger Student_Count
after insert on Student /*指明触发器激活的时间在执行INSERT后*/
referencing
        new table as DELTA
for each statement
/*语句级触发器，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
```

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”

```SQL
create trigger Insert_or_Update_Sal
        before insert or update on Teacher
        for each row
        as begin
                if (new.Job='教授') and (new.Sal < 4000)
                then new.Sal = 4000
                end if;
        end
```

- SQL Sever中，允许为任何给定的INSERT、UPDATE或DELETE语句创建多个触发器

AFTER

- 触发器只有在触发SQL语句中指定的所有操作都已成功执行后才激发
- 所有的引用级联操作和约束检查也必须成功完成后，才能执行此触发器
- 如果仅指定`for`关键字，则AFTER是默认操作
- 不能在视图上定义AFTER触发器

INSTEAD OF

- 指定执行触发器而不是执行触发SQL语句，从而替代触发语句的操作
- 在表或视图上，每个`insert、update或delete`语句最多可以定义一个`instead of`触发器
- 然而，可以在每个具有`instead of`触发器的视图上定义视图

使用带有提醒消息的触发器

```SQL
create trigger reminder
on titles
for insert, update
as
raiserror (50009, 16, 10)
```

- 返回用户定义的错误信息并设系统标志，记录发生错误
- 通过使用 `raiserror` 语句，客户端可以从 sysmessages 表中检索条目，或者使用用户指定的严重度和状态信息动态地生成一条消息，这条消息在定义后就作为服务器错误信息返回给客户端
- 消息 50009 是 sysmessages 中的用户定义消息

虚拟表

- SQL Sever 中，使用触发器需要用到两个虚拟表：`inserted, deleted`
- `inserted` 保存的是 `insert 或 update` 之后所影响的记录形成的表
- `deleted` 保存的是 `delete 或 update` 之前所影响的记录形成的表
Sql命令	deleted	inserted
insert	不可用	新插入的记录
update	被更新前的记录	被更新后的记录
delete	被删除的记录	不可用

5.7.2 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`
  - 先执行表上的before触发器
  - 其次：激活触发器的SQL语句
  - 最后：执行表上的AFTER触发器

5.7.3 删除触发器

- **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

```SQL
drop trigger Insert_Sal on Teacher
```

5.8 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