数据库系统原理

陈岭

浙江大学计算机学院

6 SQL语言（4）

- 事务
- 完整性约束
- 触发器
- 数据安全性
- 审计跟踪

事务

- 事务（transaction）由查询和更新语句的序列组成。SQL标准规定当一条SQL语句被执行，就隐式地开始了一个事务。下列SQL语句之一会结束一个事务:
  - Commit work:提交当前事务，也就是将该事务所做的更新在数据库中持久保存。在事务被提交后，一个新的事务自动开始
  - Rollback work：回滚当前事务，即撤销该事务中所有SQL语句对数据库的更新。这样，数据库就恢复到执行该事务第一条语句之前的状态

事务

- 动机示例：

```
    update account set balance = balance - 100
    where account_number = 'A-101' ;
    update account set balance = balance + 100
    where account_number = 'A-201' ;

COMMIT WORK;
```

  - 如果其中一个更新成功，另一个更新失败，会数据库中导致数据不一致问题
  - 因此，这两个更新要么全部成功，要么全部失败

- 原子性（atomic）
  - 一个事务或者在完成所有步骤后提交其行为，或者在不能成功完成其所有动作的情况下回滚其多有动作

事务

- 事务的四个性质：
  - 原子性（atomic）
  - 一致性(consistency)
  - 隔离性(isolation)
  - 持久性(durability)
- 在很多SQL实现中，默认方式下每个SQL语句自成一个事务，且一执行完就提交
  - 如果一个事务要执行多条SQL语句，就必须关闭单独SQL语句的自动提交，如何关闭自动提交也依赖于特定的SQL实现
- 一个较好的选择是，作为SQL:1999标准的一部分，允许多条SQL语句包含在关键字begin atomic … end之间

完整性约束

- 完整性约束保证授权用户对数据库所做的修改不会破坏数据的一致性
- 完整性约束的例子有：
  - 教师姓名不能为null
  - 任意两位教师不能有相同的教师标识
  - course关系中的每个系名必须在department关系中有一个对应的系名
  - 一个系的预算必须大于0.00美元
- 域完整性、实体完整性（主键的约束）、参照完整性（外键的约束）和用户定义的完整性约束
- 完整性约束是数据库实例(Instance)必须遵循的
- 完整性约束由DBMS维护

单个关系上的约束

- 单个关系上的约束
  - not null
  - unique
  - check（<谓词>）
  - 例，

```
CREATE TABLE instructor2
        ( ID char(5)  primary key ,
          name varchar(20) not null,
          dept_name varchar(20),
          salary  numeric(8,2) not null,
          check (salary>= 0));
```

域约束

- 域约束是完整性约束的最基本形式，可用于检测插入到数据库中的数据的合法性
- 从现有数据类型可以创建新的域

```
create domain Dollars as numeric(12, 2) not null
create domain Pounds as numeric(12,2);

create table instructor
     ( ID char(5) primary key,
       name varchar(20),
       dept_name varchar(20),
       salary Dollars,
       comm Pounds
      );
```

域约束

- check子句也可以应用到域上
  - 例，check子句可以保证教师工资域中只允许出现大于给定值的值

```
create domain YearlySalary numeric(8,2)
constraint salary_value_test check(value >= 29000.00);
```

  - *YearlySalary* 域有一个约束来保证年薪大于或等于29 000.00美元
  - constraint *salary_value_test* 子句是可选的，它用来将该约束命名为*salary_value_test*。系统用这个名字来指出一个更新违反了哪个约束
  - 作为另一个例子，使用in子句可以限定一个域只包含指定的一组值

```
create domain degree_level varchar(10)
constraint degree_level_test
check (value in (' Bachelors', ' Masters', or ' Doctorate'));
```

参照完整性

- 参照完整性约束定义：
  - 令关系r_1和r_2的属性集分别为R_1和R_2，主码分别为K_1和K_2
  - 如果要求对r_2中任意元组t_2，均存在r_1中元组t_1使得$t_1[K_1] = t_2[\alpha]$，我们称R_2的子集α为参照关系r_1中K_1的外码（foreign key）
  - 参照完整性约束也称为子集依赖，可写作：

$$\Pi_{\alpha}(r_2) \subseteq \Pi_{K1}(r_1)$$

参照完整性

- 回顾第二讲中外键的内容
- 假设存在关系r和s：r(A, B, C)，s(B, D)，则在关系r上的属性B称作参照s的**外码**，r也称为外码依赖的参照关系，s叫做外码被参照关系
  - 例，学生(学号, 姓名, 性别, 专业号, 年龄) – *参照关系*

    专业(专业号, 专业名称) – *被参照关系（目标关系）*

    其中属性*专业号* 称为关系*学生*的*外码*

    选修(学号，课程号，成绩)
    课程(课程号, 课程名，学分，先修课号)
  - *instructor*(*ID*, *name*, *dept_name*, *salary*) – 参照关系

    department(*dept_name*, *building*, *budget*) – 被参照关系

参照关系中外码的值必须在被参照关系中实际存在或为null

数据库的修改

- 数据库的修改会导致参照完整性的破坏。这里列出对各种类型的数据库修改应做的测试，以保持如下的参照完整性约束：

$$\Pi_{\alpha}(r_2) \subseteq \Pi_{K1}(r_1)$$

  - 插入。如果向r_2中插入元组t_2，则系统必须保证r_1中存在元组t_1使得$t_1[K] = t_2[\alpha]$。即

$$t_2[\alpha] \in \Pi_K(r_1)$$

  - 删除。如果从r_1中删除元组t_1，则系统必须计算r_2中参照t_1的元组集合。即

$$\sigma_{\alpha = t1[K]}(r_2)$$

    - 如果集合非空，要么删除命令报错并撤销，要么参照t_1的元组本身必须被删除（可能导致级联删除）

数据库的修改

- 更新。必须考虑两种更新：对参照关系r_2做更新，以及对被参照关系r_1做更新
  - 如果关系r_2中元组t_2被更新，并且更新修改外码α上的值，则进行类似插入情况的测试。令t_2'表示元组t_2的新值，则系统必须保证

$$t_2'[\alpha] \in \Pi_K(r_1)$$

  - 如果关系r_1中元组t_1被更新，并且该更新修改主码K上的值，则进行类似删除情况的测试。系统必须用旧的t_1值（更新前的值）计算

$$\sigma_{\alpha = t1[K]}(r_2)$$

  如果该集合非空，则更新失败，或者以类似删除的方式做级联更新

数据库的修改

- 更新。必须考虑两种更新：对参照关系r_2做更新，以及对被参照关系r_1做更新
  - 如果关系r_2中元组t_2被更新，并且更新修改外码α上的值，则进行类似插入情况的测试。令t_2'表示元组t_2的新值，则系统必须保证

$$t_2'[\alpha] \in \Pi_K(r_1)$$

  - 如果关系r_1中元组t_1被更新，并且该更新修改主码K上的值，则进行类似删除情况的测试。系统必须用旧的t_1值（更新前的值）计算

$$\sigma_{\alpha = t1[K]}(r_2)$$

  如果该集合非空，则更新失败，或者以类似删除的方式做级联更新

SQL中的参照完整性

- 主码、候选码和外码可在SQL的create table语句中指明
  - primary key子句包含一组构成主码的属性
  - unique子句包含一组构成候选码的属性
  - foreign key子句包含一组构成外码的属性以及被修改外码所参照的关系名

SQL中的参照完整性

- 默认地，外码参照被参照关系中的主码

  foreign key (*dept_name*) references *department*

- 可以使用如下的简写形式定义单个列为外码

  dept_name *varchar* (20) references *department*

- 被参照关系中的属性可以被明确指定，但是必须被声明为主码或候选码

  foreign key (*dept_name*) references *department* (*dept_name*)

SQL中的参照完整性

- 例,

```
create table classroom
            (building varchar (15),
             room_number varchar (7),
             capacity numeric (4,0),
             primary key (building, room_number))
create table department
            (dept_name varchar (20),
             building varchar (15),
             budget numeric (12,2) check (budget > 0),
             primary key (dept_name))
```

SQL中的参照完整性

```
create table course
        (course_id varchar (8),
         title varchar (50),
         dept_name varchar (20),
         credits numeric (2,0) check (credits > 0),
         primary key (course_id ),
         foreign key (dept_name) references department)
create table instructor
        (ID varchar (5),
         name varchar (20), not null
         dept_name varchar (20),
         salary numeric (8,2), check (salary > 29000),
         primary key (ID),
         foreign key (dept_name) references department)
```

SQL中的级联动作

```
create table course(
    . . .
    foreign key(dept_name) references department
        [ on delete cascade]
        [ on update cascade]
    . . . );
```

- 由于有了与外码声明相关联的on delete cascade子句，如果删除*department*中的元组导致了此参照完整性约束被违反，则删除并不被系统拒绝，而是对course关系作“级联”删除，即删除了被删除系的元组
- “级联”更新也类似

SQL中的级联动作

- 如果存在涉及多个关系的外码依赖链，则在链一端所做的删除或更新可能传至整个链
- 但是，如果一个级联更新或删除导致的对约束的违反不能通过进一步的级联操作解决，则系统终止该事务
  - 即，该事务所做的所有改变及级联动作将被撤销

SQL中的级联动作

- 参照完整性只在事务结束时检查
  - 中间步骤可以破坏参照完整性，只要后续步骤解消这种破坏即可
  - 否则不可能建立某些数据库状态，例如插入两条互相有外键引用的元组
    - 例，关系*marriedperson* 的*spouse* 属性

      marriedperson (*name*, *address*, *spouse*)

SQL中的级联动作

- 除级联操作之外的其他选择:
  - on delete set null
  - on delete set default
- 外键属性上的空值使SQL的参照完整性语义变得复杂，最好用not null来防止
  - 若某外键属性为null，则该元组按定义是满足参照完整性约束的