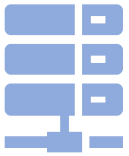
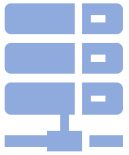


第七章 完整性约束

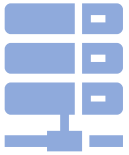


完整性约束

- 什么是数据库的完整性
  - 数据的正确性和相容性
  - 防止不合语义的数据进入数据库。

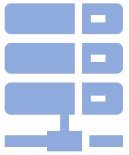
- 例：学生的年龄必须是整数，取值范围为14--29；
  - 学生的性别只能是男或女；
  - 学生的学号一定是唯一的；
  - 学生所在的系必须是学校开设的系；

- 完整性：是否真实地反映现实世界



完整性约束

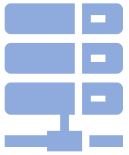
- 完整性控制机制
  - 完整性约束条件定义机制
    - ✓ 完整性约束条件：数据模型的组成部分，约束数据库中数据的语义
    - ✓ DBMS应提供定义数据库完整性约束条件的方法
  - 完整性检查机制
    - ✓ 检查用户发出的操作请求是否违背了完整性约束条件
  - 违约反应
    - ✓ 如果发现用户的操作请求使数据违背了完整性约束条件，则采取一定的动作来保证数据的完整性



完整性约束

■完整性约束条件

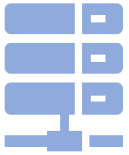
- 完整性约束条件作用的对象
  - ✓ 列：对属性的取值类型、范围、精度等的约束条件
  - ✓ 元组：对元组中各个属性列间的联系的约束
  - ✓ 关系：对若干元组间、关系集合上以及关系之间的联系的约束
- 对象的两种状态：
  - ✓ 静态：对静态对象的约束是反映数据库状态合理性的约束
  - ✓ 动态：对动态对象的约束是反映数据库状态变迁的约束。涉及新值和旧值



完整性约束

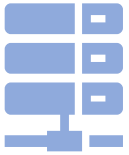
■完整性约束条件分类

- 六类完整性约束条件
  - ✓ 静态列级约束
  - ✓ 静态元组约束
  - ✓ 静态关系约束
  - ✓ 动态列级约束
  - ✓ 动态元组约束
  - ✓ 动态关系约束



完整性约束

- 静态列级约束
  - 对列的取值域的说明
  - 如取值类型、取值范围、是否为空值等约束
  - CHECK(price<1000)
- 静态元组约束
  - 规定元组的各个列之间的约束关系
  - 如有订货关系（ID，商品，订货量，发货量），要求每条记录中：发货量<=订货量
  - 用Check约束实现



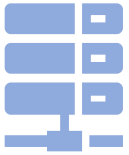
完整性约束

- 静态关系约束
  - 关系的各个元组之间或若干关系之间存在的各种联系或约束
  - 分类：
    - ✓ **实体完整性约束**
    - ✓ **参照完整性约束**
    - ✓ 函数依赖约束
    - ✓ 统计约束...

例：工资表（职工ID，，工资，是否部门经理），要求：

职工平均工资的2倍<=部门经理的工资<=职工平均工资的5倍

需用触发器实现



完整性约束

- 动态列级约束
  - 动态列级约束是修改列定义或列值时应满足的约束条件

    1) 修改列定义时的约束

    例：将原来允许空值的列改为不允许空值时:

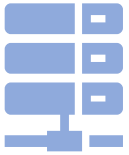
    该列目前已存在空值，则拒绝这种修改（数据库自动实现）

    2) 修改列值时的约束

    修改列值时新旧值之间要满足的约束条件（使用触发器实现）

    例：职工工资调整 >= 原来工资

    年龄只能增长



完整性约束

- 动态元组约束
  - 修改元组值: 各个字段之间要满足的约束条件

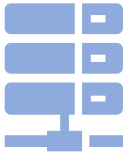
    例: 职工工资调整不得低于其原来工资 + 工龄*1.5

    用触发器实现

- 动态关系约束
  - 对关系变化前后状态的限制条件

    例: 事务一致性、原子性等约束条件

    具体内容在事务处理部分讲解



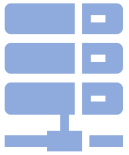
完整性约束

- 完整性约束另一种分类方法
  - 最重要的几类完整性约束条件：
    - ✓ 域完整性
    - ✓ 实体完整性约束
      - ✓ 取值唯一； 不能为空
    - ✓ 参照完整性约束

    可以方便地在数据库中设置
  - 其它的完整性约束条件：
    - ✓ 用户自定义完整性约束

    往往需编程实现（触发器或Check等）

动态关系完整性约束：
事务一致性、原子性约束



完整性控制

- 完整性控制机制：

  指完整性约束的定义、检查和违约反应规则

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

当用户发出什么操作请求时需要检查该完整性规则

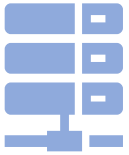
是立即检查还是延迟检查；

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。



完整性控制

- 完整性控制机制：

例：在“学号不能为空”的约束中

D 约束作用的对象为Sno属性

O 插入或修改Student 元组时

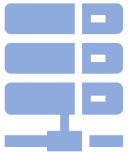
A Sno不能为空

C 无（A可作用于所有记录的Sno属性）

P 拒绝执行该操作

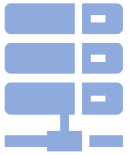
违约反应

- 关系数据库系统都提供了定义和检查实体完整性、参照完整性和用户定义的完整性的功能
- 违反实体完整性规则的操作一般是拒绝执行
- 违反参照完整性的操作的违约反应：
  - ✓ 可以拒绝执行
  - ✓ 也可以接受这个操作，同时执行一些附加的操作，以保证数据库的状态正确



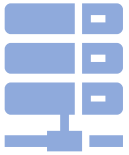
违约反应

- 参照完整性违约反应
  - 外键是否可以取空值：依赖应用环境语义而定
  - 在被参照关系中删除元组时，若参照关系有若干元组的外键值与被删除的被参照关系的主键值相同，此时的违约反应可以是：
    - ✓ 级联删除（CASCADES）
    - ✓ 受限删除（RESTRICTED）
    - ✓ 置空值删除（NULLIFIES）



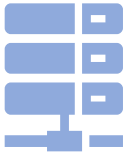
违约反应

- 参照完整性违约反应
  - 在参照关系中插入元组时，若被参照关系不存在相应的元组，则违约反应可以是：
    - ✓ 受限插入
    - ✓ 递归插入
  - 修改被参照关系中的主键，若主键在参照关系中已经被引用，则违约反应可以是：
    - ✓ 级联修改
    - ✓ 受限修改
    - ✓ 置空值修改



用户自定义约束的实现

- Check约束
- 触发器



用户自定义约束的实现

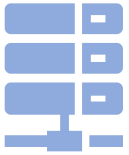
■ Check约束

CHECK 约束通过限制输入到列中的值来强制域的完整性。

常用于实现静态列级约束和静态元组约束。

可以通过任何基于逻辑运算符返回结果 TRUE 或 FALSE 的逻辑（布尔）表达式来创建 CHECK 约束。

表达式可包含列名、比较运算符（>、<、=等）、逻辑运算符(and、or等)、条件谓词（in、like等），可使用pattern（如_、%通配符，[]选择符等)



用户自定义约束的实现

■ Check约束

例：

欲将 salary 列的取值范围限制在 $15,000 至 $100,000 之间，则Check约束为：

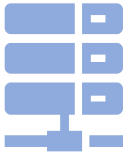
```
salary >= 15000 AND salary <= 100000
```

检查Email字段的格式：

```
email like  '%_@_%._%'
```

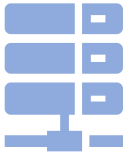
学号是一个五位字符串，前二位为字符，后三位为数字，则可定义CHECK约束如下：

```
SID like  '__[0,9][0,9][0,9]'
```



用户自定义约束的实现

- 触发器
  - 用于实现复杂逻辑的用户自定义约束的工具
  - 是一种特殊的存储过程
    - 与表（或视图、数据库等）紧密相连，不能脱离宿主存在
    - 由数据库自动调用执行，用户不能调用
    - 没有参数和返回值



用户自定义约束的实现

- 触发器
  - 是一种Event-Condition-Action规则的实现
    - Event事件： 引起数据库更新的事件，如执行Insert语句向表中插入数据
    - Condition条件：判断是否满足条件的SQL表达式
    - Action动作：任意的SQL语句

```
CREATE [ OR REPLACE ] TRIGGER trigger_name
  [BEFORE | AFTER | INSTEAD OF]
trigger_event  ON table_reference
  [FOR EACH ROW [ WHEN trigger_condition ] ]
  trigger_body;
```

用户自定义约束的实现

- 触发器

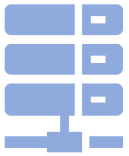
应用示例1：

Fruits (Fname, Price); Sells (Cname, Fname, Stime, Quantity)

需求： 当向Sell表中插入一条新的购买记录的时候，若Fname（水果名称）在fruits表中不存在，则向Fruits表中插入一条新的记录，其中Fname为新购买记录中的水果名称，Price为NULL。

分析： 可以用外键约束的级联插入实现。但是很多数据库不支持级联插入，怎么办？

用触发器实现



用户自定义约束的实现

- 触发器

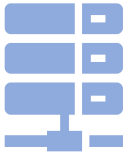
Fruits (Fname, Price); Sells (Cname, Fname, Stime, Quantity)

```
CREATE TRIGGER FruitTrig
    AFTER INSERT ON Sells
    REFERENCING NEW ROW AS NewTuple
    FOR EACH ROW
    WHEN (NewTuple.Fname NOT IN
        (SELECT Fname FROM Fruits))
    INSERT INTO Fruits(Fname)
        VALUES(NewTuple.Fname);
```

事件

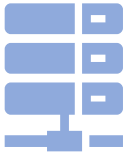
条件

动作



用户自定义约束的实现

- 触发器
  - 事件
    - ✓ 表/视图级事件：INSERT / DELETE / UPDATE
    - ✓ 数据库级事件：CREATE / ALTER / DROP TABLE / PROCEDURE / VIEW…
  - 触发的时机
    - ✓ AFTER /BEFORE /INSTEDA OF
    - ✓ Insteda Of 触发器会替换掉用户提交的操作语句。常用于不可更新视图的更新。



用户自定义约束的实现

- 触发器
  - 触发/执行粒度
    - 语句级触发器：缺省
      - ✓ 每条用户语句触发一次
    - 行级触发器： FOR EACH ROW
      - ✓ 对于表中每条受影响记录，触发一次触发器

用户语句：Update stu set Gender= ‘F’

--500条语句受影响

语句级触发器：

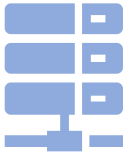
触发和执行一次

行级触发器：

触发和执行500次

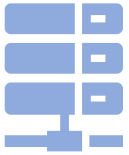
用户自定义约束的实现

- 触发器
  - Referencing引用
    - 名为“Inserted”的表，包含新插入的行
    - 名为“Deleted” 的表，包含删除的行
    - 在触发器业务逻辑中可以通过引用来处理新值、旧值
    - OpenGauss、MySQL等数据库中可以直接使用关键字New、Old来引用新、旧值：New.Attribute, Old.Attribute



用户自定义约束的实现

- 触发器
  - 动作
    - ✓ 实现自定义约束的业务逻辑
    - ✓ 通常用于进行关联数据更新，不能返回查询结果
    - ✓ 若需多于一条语句，则可用程序块实现：Begin...End
      - ✓ 自学存储过程编程技术
    - ✓ 拒绝更新的违约动作：引发系统错误，提示数据库拒绝更新，可用RaiseError（SQLServer）、Signal（MySQL）、Raise（OpenGauss）函数



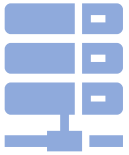
用户自定义约束的实现

- 触发器
  - 应用示例2：现有学生表Student（stu, class, age ），要求：班级总人数不超过30人

```
CREATE OR REPLACE FUNCTION Stu_num() RETURNS TRIGGER AS $$
        declare
          SSum int;
        BEGIN
           select count(sname) into snum from (select sname from student group by
sname having count(sname)>30) t;
            if ssum>0 then
              raise notice 'Too many students in one class';
              return null;
            end if;
            return new;
         END $$ LANGUAGE PLPGSQL;

create trigger TSC before insert on student execute procedure Stu_num();
```

OpenGauss的实现
--语句级触发器



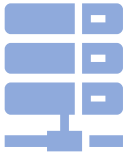
用户自定义约束的实现

- 触发器
  - 应用示例2：现有学生表Student（stu, class, age ），要求：班级总人数不超过30人

```
CREATE OR REPLACE FUNCTION Stu_num() RETURNS TRIGGER AS $$
        declare
          SSum int;
        BEGIN
           select count(*) into SSum from student where class=new.class;
            if ssum>30 then
               raise notice 'Too many students in one class';
               return null;
            end if;
            return new;
         END $$ LANGUAGE PLPGSQL;
```

OpenGauss的实现
--行级触发器

```
create trigger TSC before insert on student for each row execute procedure
Stu_num();
```



用户自定义约束的实现

- 触发器
  - Instead of 触发器
    - 将用户提交的语句替换成触发器代码
    - 在视图上创建的 Instead of 触发器可以实现不可更新视图的更新

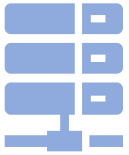
应用示例3：

基本表：

Fruits (Fname, Price); Sells (Cname, Fname, Stime, Quantity)

视图：

Create View Order as select s.cname, f.fname, price, quantity from Fruits f, Sells s where f.fname=s.fname



用户自定义约束的实现

- 触发器
  - Instead of 触发器

OpenGauss的实现

```
CREATE OR REPLACE FUNCTION tri_order() RETURNS TRIGGER AS
        $$  declare
        BEGIN
           if not new.price is null then
               update fruits set price = new.price where fname=new.fname;
            end if;
           return new;
        END
        $$ LANGUAGE PLPGSQL;

create trigger of1 instead of update on orderv for each row
      execute procedure tri_order();
```

MySQL不支持instead of 触发器，

也不支持在视图上建触发器

用户自定义约束的实现

- 触发器
  - 应用示例1的语句在MySQL中无法运行，因MySQL不支持When条件
  - 例1在MySQL中的实现：

```
create trigger ts1 after insert on sell for each row
        begin
            declare i int;
            select count(*) into i from fruits where fname=new.fname;
            if i=0 then
                insert into fruits values(new.fname, NULL);
            end if;
        end;
```

  - 例1在OpenGauss中也无法运行，因其When语句不支持子查询。请自行在OpenGauss中改造实现

作业

- 课本第173页第1， 3， 4， 5， 6， 7题
- 第8题的问题改为：在Male和Female表上各建立一个触发器，将来宾人数限制在50人以内。任选MySQL或OpenGauss数据库实现，提交触发器代码。
- 提交时间：下周上课之前