CH7 Constraints & Triggers约束和触发器

一、Keys and Foreign Keys

1. Kinds of Constraints

- Key （主键，唯一键）（单个关系）
- Foreign-key（参照完整性）（两个关系）
- Valued-based constraint（单个关系）
- Tupled-based constraint（单个关系）
- Assertion（两个关系）

2. Key

```
cert INT PRIMARY KEY,        -- 单属性键

PRIMARY KEY (title, year)    -- 多属性键
PRIMARY KEY (year, title)    -- 两种表达方式有差异
```

只能有一个主键，但是可以有很多UNIQUE属性

主键属性不可以赋值为NULL

3. Foreign Key

```
presC INT REFERENCES movieExec(cert)                -- 逻辑上定义外键前需要确保已定义原关系

FOREIGN KEY presC INT REFERENCES movieExec(cert)    -- 与上面定义方式等价
```

4. Unique

用于确保列中的每个值都是唯一的，但并不像主键那样自动创建索引或强制列非空。可以在表的多个列上定义唯一约束。

使用REFERENCES引用的属性必须通过PRIMARY KEY或UNIQUE声明（在主表里）

5. Actions

假设A和B是多对一关系$A - R \rightarrow B$

对A的INSERT或UPDATE可能会导致更新出了不存在于B中的数据，删除不会产生错误。

对B的DELETE或UPDATE可能会导致A中的部分数据不存在（因为没有相应更改）

在对B执行DELETE或UPDATE时有三种应对方式：

> Default : 拒绝
>
> Cascade : 同步修改或者删除A中的相关元组
>
> Set NULL : 将A中有关B的主键设为NULL

```
FOREIGN KEY presC INT REFERENCES movieExec(cert) ON UPDATE CASCADE ON DELETE SET NULL,  -- 常见搭配
```

同时向A和B中插入在A和B存在循环约束时不可取。

6. Deferred Checking

循环约束:

由于插入元组只能一条一条的插入，所以在向其中一个表中插入数据时必将引起另一个表的约束冲突。

**上述结论在可以使用事务的情况下不成立

```
presC INT REFERENCES MovieExec(cert) DEFERRABLE INITIALLY DEFERRED  -- 告诉DBMS在事务中推迟检查
```

任何的约束后面都客家加DEFERRABLE或NOT DEFERRABLE（默认）关键字

```
DEFERRABLE INITIALLY DEFERRED   -- 事务commit之后才检查
DEFERRABLE INITIALLY IMMEDIATE  -- 语句执行完毕立即检查
```

二、Constraints on Attributes and tuples

1. Not-Null

```
cert INT NOT NULL,
```

定义为PRIMARY KEY时，自动带有Not-null约束，

Not-null约束的结果：

> 外键更新时无法SET NULL
>
> 部分插入

2. Attribute-Based Check

```
gender CHAR(1) CHECK (gender IN ('F', 'M')),
```

基于属性的约束在执行INSERT和UPDATE时会触发。

DELETE操作不会触发。

$A \rightarrow B$

对B的修改不会触发A上的CHECK

3. Tuple-Based Check

`CHECK <condition>` 会被添加为关系模式的一个元素

***但是其他属性或关系需要一个子查询（postgresql中不支持）

```
CREATE TABLE ...(
    ...
    CHECK (gender = 'F' OR name NOT LIKE 'Ms.%')
)
```

每当对关系UPDATE或INSERT时都会触发，对元组的任意元素的修改都会触发。

同样不对外键约束敏感。

***多于两个的属性检查需要改写为元组检查

三、Modification of Constraints

1. Give name

```
column1 datatype CONSTRAINT constraint_name UNIQUE,
column1 datatype CONSTRAINT constraint_name PRIMARY KEY,

SET CONSTRAINTS constraint_name PRIMARY KEY;
```

2. Altering constraints on tables

```
SET CONSTRAINT constraintName DEFERRED (or IMMEDIATE);
ALTER TABLE relationName DROP CONSTRAINT constraintName;
ALTER TABLE relationName ADD CONSTRAINT constraintName CHECK(...);  -- 元组检查，且持有所有元组时才可添加
```

四、Assertions

1. Create

```
CREATE ASSERTION <name> CHECK (...);

DROP ASSERTION <name>

-- 任何表的更改都会触发断言
CREATE ASSERTION RichPres CHECK
    (NOT EXISTS
        (SELECT * FROM studio, movieExec WHERE presC#=cert# AND networth<1000000)
    );  -- 需要和外键配合
```

断言是一个Boolean值，且必须要在任何时候保持True（断言有可能建立失败）

五、Triggers

1. Definition

特定时间发生时触发，触发之后测试一个条件，条件满足，执行特定语句。

2. ECA rule

- Event : 某种特定操作，如insert on sells
- Condition : 任何SQL布尔表达式
- Action : 任何SQL语句

3. Create

```
CREATE TRIGGER BeerTrig
-- CREATE OR REPLACE TRIGGER BeerTrig
    AFTER INSERT ON Sells
    REFERENCING NEW ROW AS NewTuple
    FOR EACH ROW
    WHEN (NewTuple.beer NOT IN (SELECT name FROM Beers))
    INSERT INTO Beers(name) VALUES(NewTuple.beer);
```

4. Options

AFTER/BEFORE/INSTEAD OF（更适用于视图修改）

INSERT/DELETE/UPDATE

FOR EACH ROW/STATEMENT（默认）

REFERENCING [NEW/OLD] [TUPLE/TABLE] AS [name]