8.3 授权与触发器

主要考点

1、授权(GRANT)
2、收回授权(REVOKE)
3、触发器概述
4、创建触发器
5、更改与删除触发器

授权(GRANT)

语句格式：　GRANT 权限 ON TABLE/DATABASE 表名/数据库名
TO 用户1,用户2.../PUBLIC
[WITH GRANT OPTION];

PUBLIC：表示将权限授予所有人

WITH GRANT OPTION：表示获得了这个权限的用户还可以将权限赋给其他用户。

对象	对象类型	操作权限
属性列	TABLE	SELECT，INSERT，UPDATE，DELETE， ALL PRIVILEGES（4种权限的总和）
视图	TABLE	SELECT，INSERT，UPDATE，DELETE， ALL PRIVILEGES（4种权限的总和）
基本表	TABLE	SELECT，INSERT，UPDATE，DELETE，ALTER，INDEX， ALL PRIVILEGES（6种权限的总和）
数据库	DATABASE	CREATETAB建立表的权限，可由DBA授予普通用户

授权(GRANT)

例：用户要求把数据库SPJ中供应商S、零件P、项目J表赋予各种权限。各种授权要求如下：

（1）将对供应商S、零件P、项目J的所有操作权限赋给用户User1及User2。

```
GRANT ALL PRIVILEGES
ON TABLE S,P,J
TO USER1,USER2;
```

（2）将对供应商S的插入权限赋给用户User1，并允许将此权限赋给其他用户。

```
GRANT INSERT
ON TABLE S
TO USER1 WITH GRANT OPTION;
```

（3）DBA把数据库SPJ中建立表的权限赋给用户User1。

```
GRANT CREATETAB
ON DATABASE SPJ
TO User1;
```

收回权限(REVOKE)

语句格式：　REVOKE 权限 ON TABLE/DATABASE 表名/数据库名

FROM 用户1,用户2.../PUBLIC

[RESTRICT | CASCADE];

RESTRICT：表示只收回语句中指定的用户的权限

CASCADE：表示除了收回指定用户的权限外，还收回该用户赋予的其他用户的权限。

例：将用户User1及User2对供应商S、零件P、项目J的所有操作权限收回：

REVOKE ALL PRIVILEGES ON TABLE S,P,J FROM User1,User2;

将所有用户对供应商S的所有查询权限收回：

REVOKE SELECT ON TABLE S FROM PUBLIC;

将User1用户对供应商S的供应商编号Sno的修改权限收回。

REVOKE UPDATE(Sno) ON TABLE S FROM User1;

例：收回用户li对表employee的查询权限，同时级联收回li授予其他用户的该权限，SQL语句为：（1）select ON TABLE employee FROM li（2）;

（1）A. GRANT　　B.GIVE　　C.CALLBACK　　D.REVOKE

（2）A. RESTRICT　　B.CASCADE　　C. WITH GRANT OPTION　　D.WITH CHECK OPTION

触发器概述

- 触发器主要有以下三方面的特点：

（1）当数据库程序员声明的事件发生时，触发器被激活。声明的事件可以是对某个特定关系的插入、删除或更新。

（2）当触发器被事件激活时，不是立即执行，而是首先由触发器测试触发条件，如果事件不成立，响应该事件的触发器什么都不做。

（3）如果触发器声明的条件满足，则与该触发器相连的动作由DBMS执行。动作可以阻止事件发生，可以撤销事件。

- 创建触发器时需指定：

（1）触发器名称

（2）在其上定义触发器的表

（3）触发事件：触发器将何时激发

（3）触发条件：满足什么条件时执行触发动作

（4）触发动作：指明触发器执行时应做的动作

- 触发器可以引用当前数据库以外的对象，但只能在当前数据库中创建触发器。
- 不能在临时表或系统表上创建触发器，但触发器可以引用临时表。

创建触发器

```
CREATE TRIGGER 触发器名称 [BEFORE | AFTER]
       [DELETE | INSERT | UPDATE OF 列名]                    //触发事件
   ON 表名
  [REFERENCING <临时视图名>]
  [FOR EACH ROW | FOR  EACH STATEMENT]
  [WHEN <触发条件>]          //WHEN后面跟触发条件，指明当什么条件满足时，执行下面的触发动作
  BEGIN
     <触发动作>              //BEGIN...END 中定义触发动作，即当触发条件满足时，需要数据库做什么
  END  [触发器名称]
```

BEFORE/AFTER：指明是在执行触发语句之前激发触发器还是执行触发语句之后激发触发器。

DELETE：当一个DELETE语句从表中删除行时激发触发器。

INSERT：当一个INSERT语句向表中插入行时激发触发器。

UPDATE/UPDATE OF(列名)：当UPDATE修改表中的值时，激发触发器，也可加（OF 列名）指定是某一列的值被修改时激发触发器。

REFERENCING：触发器运行过程中，系统会生成两个临时视图，分别存放更新前和更新后的值，对于行级触发器，为OLD ROW和NEW ROW，对于语句级触发器，为OLD TABLE和NEW TABLE。

REFERENCING new row AS nrow / REFERENCING old row AS orow 。

FOR EACH ROW：表示为行级触发器，对每一个被影响的元组（即每一行）执行一次触发过程。

FOR EACH STATEMENT：表示为语句级触发器，对整个事件只执行一次触发过程，为默认方式。

例：银行数据库关系模式如下：

Account(Account-no, branch-name, balance) 账户(账号，支行名称，余额)

Loan(Loan-no,branch-name,amount) 贷款(贷款号，支行名称，金额)

Depositor(customer-name,Account-no) 存款(存款人姓名，账号)

假设银行处理透支时，不是将账户的余额设为负值，而是将账户余额设置为0，并且建立一笔贷款，其金额为透支金额。这笔贷款的贷款号等于该透支账户的账号。

```
CREATE TRIGGER overdraft_trigger AFTER UPDATE ON Account      //在Account表上建立触发器，触发事件是“修改后”
  Referencing new row as nrow                                 //引用修改后的“行”为临时视图，命名为nrow
  For each row                                                //表示其为行级触发器，即对每一行都执行一次触发
WHEN nrow.balance<0                 //触发条件是修改后的行的balance属性值小于0，即账户余额为负。
BEGIN ATOMIC                        //ATOMIC关键字表示下面的语句为原子性的，即：要么都做，要么都不做。
    INSERT INTO borrower
            ( Select customer-name, Account-no
              From depositor
              Where nrow.account-no = depositor.account-no );      //将透支客户的信息单独存储在borrower表中
    INSERT INTO loan values
            ( nrow.account-no, nrow.branch-name, －nrow.balance );   //建立一笔贷款，将该笔记录插入到贷款表loan中
    UPDATE account set balance=0
        Where account.account-no = nrow.account-no;          //将Account表中的余额设为0
END
```

例：银行数据库关系模式如下：

Account(Account-no, mark)　　账户(账号，还款标志)

Repayment(Account-no, repaydate)　还款(账号，还款日期)

假设客户的信用卡在还款前还款标志为0，还款后为1，DBA希望不定期查看每个月还款的客户总数。比如在6月的时候希望查看3月份有多少客户完成了还款。

```
CREATE TRIGGER Account_trig
      AFTER UPDATE OF MARK ON Account      //在Account表上建立触发器，触发事件是 MARK属性的值修改后
  Referencing new row as nrow, old row as orow          //引用修改后的行，命名为nrow，修改前的行为orow
  For each row                                          //表示其为行级触发器，即对每一行都执行一次触发
WHEN  orow.mark=0  and  nrow.mark=1                     //触发条件是修改后的行的balance属性值为0
BEGIN
    INSERT INTO Repayment  values
          (nrow.account-no, getdate( ));                //在还款表中插入一行，用来记录还款客户的账号和还款日期。
    UPDATE Repayment set repaydate=getdate( )
        Where Repayment.account-no = nrow.account-no;   //如果还款表中已有所有客户的账号信息了，就可以
                                                        使用UPDATE语句直接修改还款时间
END
```

2013年下午试题二：

某航空公司要开发一个订票信息处理系统，该系统的部分关系模式如下：

航班（航班编号，航空公司，起飞地，起飞时间，目的地，到达时间，票价）
折扣（航班编号，开始日期，结束日期，折扣）
旅客（身份证号，姓名，性别，出生日期，电话，VIP折扣）
购票（购票单号，身份证号，航班编号，搭乘日期，购票金额）

有关关系模式的属性及相关说明如下：

（1）航班表中的起飞时间和到达时间不包含日期，同一航班不会在一天出现两次及两次以上；

（2）各航空公司会根据旅客出行淡旺季适时调整机票的折扣，旅客购买机票的购票金额计算公式为：票价×折扣×VIP折扣，其中旅客的VIP折扣与该旅客已购买过的机票的购票金额总和相关，在旅客每次购票后被修改。VIP折扣值的计算由函数float vip_value（char[18]身份证号）完成。

根据以上描述，回答下列问题。

【问题1】

请将如下创建购票关系的SQL语句的空缺部分补充完整，要求指定关系的主键、外键，以及购票金额大于零的约束。

```
CREATE TABLE 购票 （
        购票单号 CHAR(15) _____（a）_____ ,
        身份证号 CHAR(18),
        航班编号 CHAR(6),
        搭乘日期 DATE,
        购票金额 FLOAT _____（b）_____ ,
        _____（c）_____ ,
        _____（d）_____ ,
    ）;
```

【问题2】

(1)身份证号为210000196006189999的客户购买了2013年2月18日CA5302航班的机票，购票单号由系统自动生成。下面的SQL语句将上述购票信息加入系统中，请将空缺部分补充完整。

```
INSERT INTO 购票（购票单号，身份证号，航班编号，搭乘日期，购票金额）
SELECT ‘201303105555’，‘210000196006189999’，‘CA5302’，‘2013/2/18’，
              （e）
FROM  航班，折扣，旅客
WHERE    （f）    AND 航班.航班编号=‘CA5302’
    AND‘2013/2/18’BETWEEN  折扣.开始日期  AND   折扣.结束日期
    AND 旅客.身份证号 =‘210000196006189999’；
```

（2）需要用触发器来实现VIP折扣的修改，调用函数vip_value（）来实现。请将如下SQL语句的空缺部分补充完整。

```
CREATE  TRIGGER  VIP _TRG  AFTER   （g）   ON   （h）
    REFERENCING new row AS  nrow
    FOR EACH row
    BEGIN
        UPDATE 旅客
        SET   （i）
         WHERE （j）；
    END
```

【问题3】

请将如下SQL语句的空缺部分补充完整。

（1）查询搭乘日期在2012年1月1日至2012年12月31日之间，且合计购票金额大于等于10000元的所有旅客的身份证号、姓名和购票金额总和，并按购票金额总和降序输出。

```
SELECT 旅客.身份证号，姓名，SUM（购票金额）
FROM  旅客，购票
WHERE  ___（k）___
GROUP BY  ____（l）____
ORDER  BY  ___（m）___ ;
```

（2）经过中转的航班与相同始发地和目的地的直达航班相比，会享受更低的折扣。查询从广州到北京，经过一次中转的所有航班对，输出广州到中转地的航班编号、中转地和中转地到北京的航班编号。

```
SELECT ___（n）___
FROM 航班航班1，航班航班2
WHERE  ___（o）___ ;
```

更改和删除触发器

1、更改触发器

语句格式：

```
ALTER TRIGGER <触发器名> [BEFORE|AFTER]
    DELETE|INSERT|UPDATE OF [列名]
    ON 表名|视图名
    AS
    BEGIN
        要执行的SQL语句
    END
```

2、删除触发器

语句格式：

```
DROP TRIGGER <触发器名>
```