第四,五章

数据库的安全性和完整性

本章主要内容

- 数据库安全性的概念
- 数据库安全性的措施
- 数据库完整性的概念
- 数据库完整性的措施
- 各种约束条件
- 断言
- 触发器

数据库的安全性

- 什么是数据库的安全性?

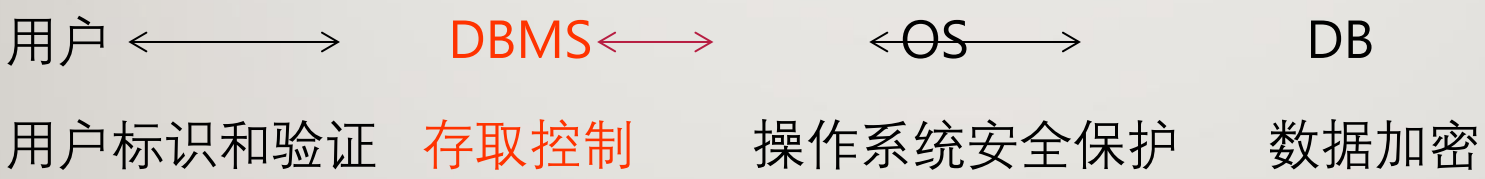
保护数据库以防止不合法的使用所造成的数据泄露、更改或破坏。

安全标准

计算机以及信息安全标准两个重要标准:

- TCSEC: 1985年美国国防部颁布的可信计算机系统评估准则. 分为四个组,七个等级(D,C1,C2,B1,B2,B3,A1)
- CC: 国际通用标准，安全功能要求(11类)和安全保证要求(7个类)

数据库的安全性控制



- 用户身份鉴别：通过帐号和密码来控制。
- **DBMS：定义用户权限，合法权限的检查。实现方法有自主存取控制和强制存取控制两种。**
- **视图机制**
- **审计：把用户对数据库的所有操作自动记录下来。**

存取控制的两种方法

- **自主存取控制方法**

GRANT，　REVOKE语句

- **强制存取控制方法** (用户不能直接感知)

主体：系统中的活动主体,代表实际的用户。
客体：系统中的被动实体.例如表,视图等。
敏感度标记：　（主体：许可证级别）
　　（客体：绝密，机密，可信，公开）
通过对比主体和客体的敏感度标签决定主体是否可以存取客体.

数据库安全性：自主存取控制

- 定义用户权限

Grant SELECT on student to ‘张三’;

存放在数据字典里。

- 合法权限检查

  用户发出的操作都要进行权限检查。如超出定义的权限，系统拒绝执行。

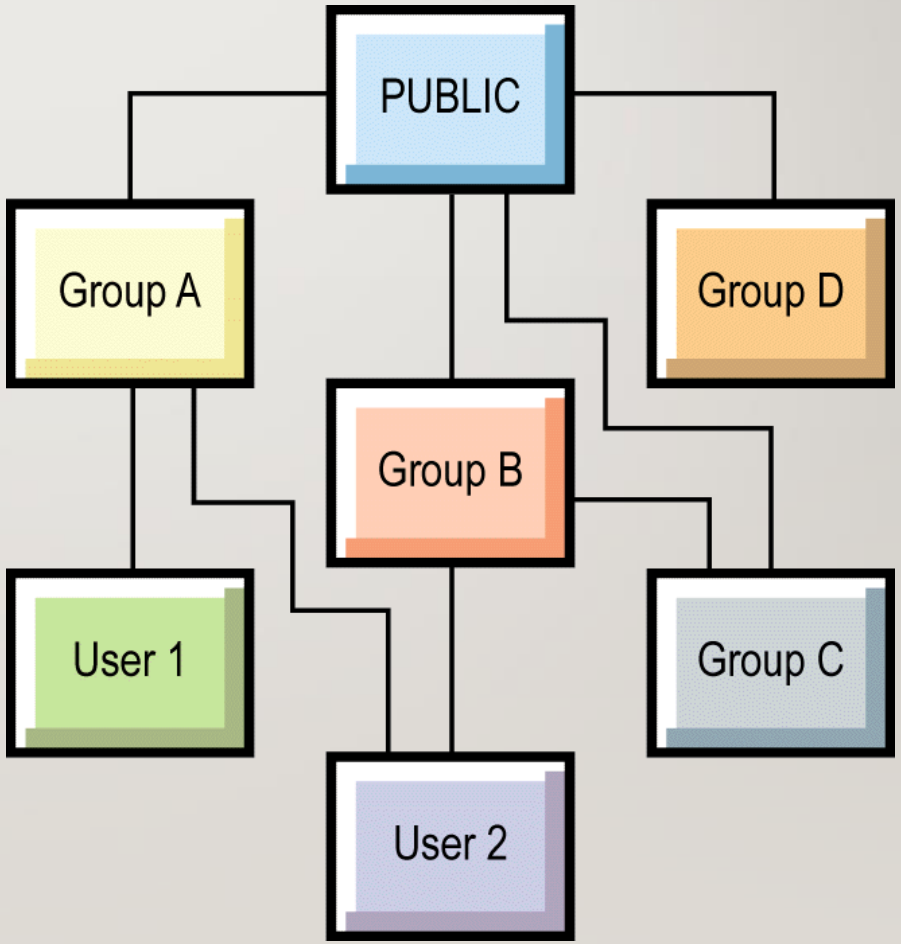
用户授权——用户/组

- 用户
  - 数据库的使用者
- 组
  - 具有相同用户权限的用户的集合
  - 便于管理
- 特性
  - 具有特定的角色
  - 用户名/口令
  - 权限（表/视图、操作）

用户授权——用户/组

- 创建用户组
- 将用户加入用户组

Public：是所有用户的总称。



数据库用户的三种权限

由DBA在创建用户时实现

- CREATE USER <username>{WITH][DBA I RESOURCE I CONNECT];

三种权限:

- DBA: 超级用户，创建用户，数据库，基本表等
- RESOURCE: 可以创建基本表和视图,进行操作，不能创建user,schema。
- CONNECT: 可以登录数据库,执行数据查询和操作（默认），不能创建。

用户权限

- DBA具有数据库中的所有权限
- 基本表的Owner具有对该表的一切操作权限
- 用户可被授予权限，也可能具有传播该权限的能力（如果指定了With Grant Option）

授权语句

Grant <权限>[,<权限>]...

[On <对象类型><对象名>]

To <用户>[,<用户>]...

[With Grant Option];

具有授权的权限

用户授权

- 用户的创建（各DBMS产品会有所不同）

  GRANT CONNECT TO userid , …

  IDENTIFIED BY password , …

- 例:

  GRANT CONNECT TO WANG

  IDENTIFIED BY WANG

用户授权

- 给特定的用户/用户组授予相应的权限

- 例:向用户WANG授予表student的查询权限

  Grant Select On Table student To WANG;

对象与权限

对　象	对象类型	操　作　权　限
属性列	TABLE	SELECT，INSERT，UPDATE，DELETE ，ALL PRIVILEGES
视　图	TABLE	SELECT，INSERT，UPDATE，DELETE ，ALL PRIVILEGES
基本表	TABLE	SELECT，INSERT，UPDATE，DELETE ，ALTER， INDEX，ALL PRIVILEGES
数据库	DATABASE	CREATETAB

数据库的角色

- 数据库角色是被命名的一组与数据库操作相关的权限,角色是权限的集合.使用角色管理数据库权限可以简化授权的过程.
- CREATE ROLE <角色名>
- GRANT<权限>[, <权限>] ON <对象类型>对象名 TO <角色>[, <角色>]

数据库的角色: 例子

- CREATE ROLE 教务员
- GRANT SELECT,UPDATE,INSERT ON TABLE STUDENT TO 教务员

- GRANT 教务员 TO 王品,张明,赵玲

用户授权

- 例：向用户 A 授予student表的查询以及对其中sdept属性的修改权限，并允许其传播该权限

```
Grant select , update(sdept)
        on Table student
        to A
        with grant option
```

用户授权

- 例：向用户A以及用户B授予student表上的所有权限

Grant All privileges

on Table student

to A，B

权限回收

- Revoke
- 回收由Grant授予的权限

Revoke <权限>[,<权限>]...

[On <对象类型><对象名>]

From <用户>[,<用户>]...[CASCADE I RESTRICT];

权限回收

- 级联回收

  -某用户（具有权限传播能力）的权限被回收后，由其授权的用户权限也被自动回收。

- 受限制回收

  - 如果该用户已把权限授予其它用户，则该回收被拒绝。

角色权限的回收

- Revoke <权限> >【, <权限>】 ON <对象类型> FROM <角色>【, <角色>】...

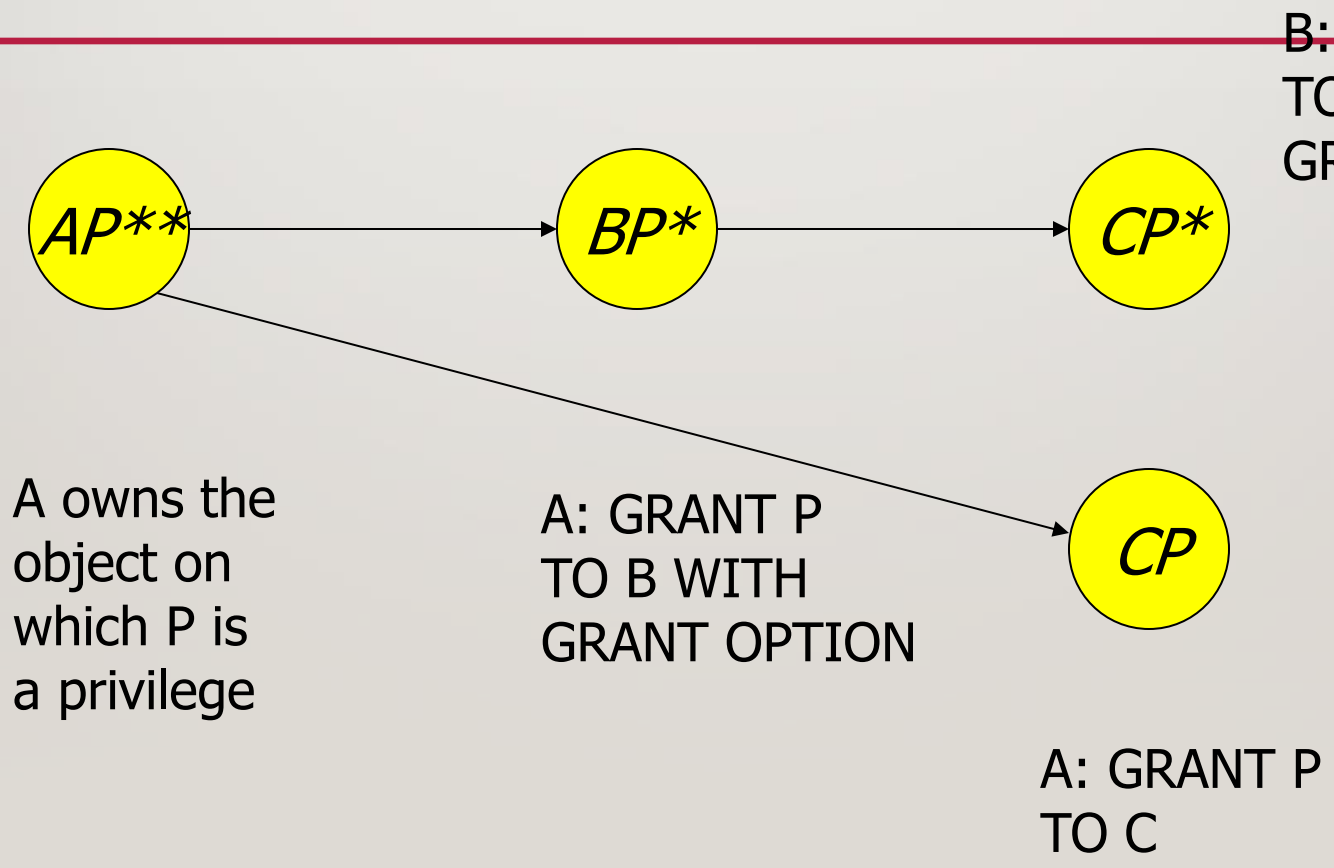
- Revoke 教务员 from 王品;

用户权限回收

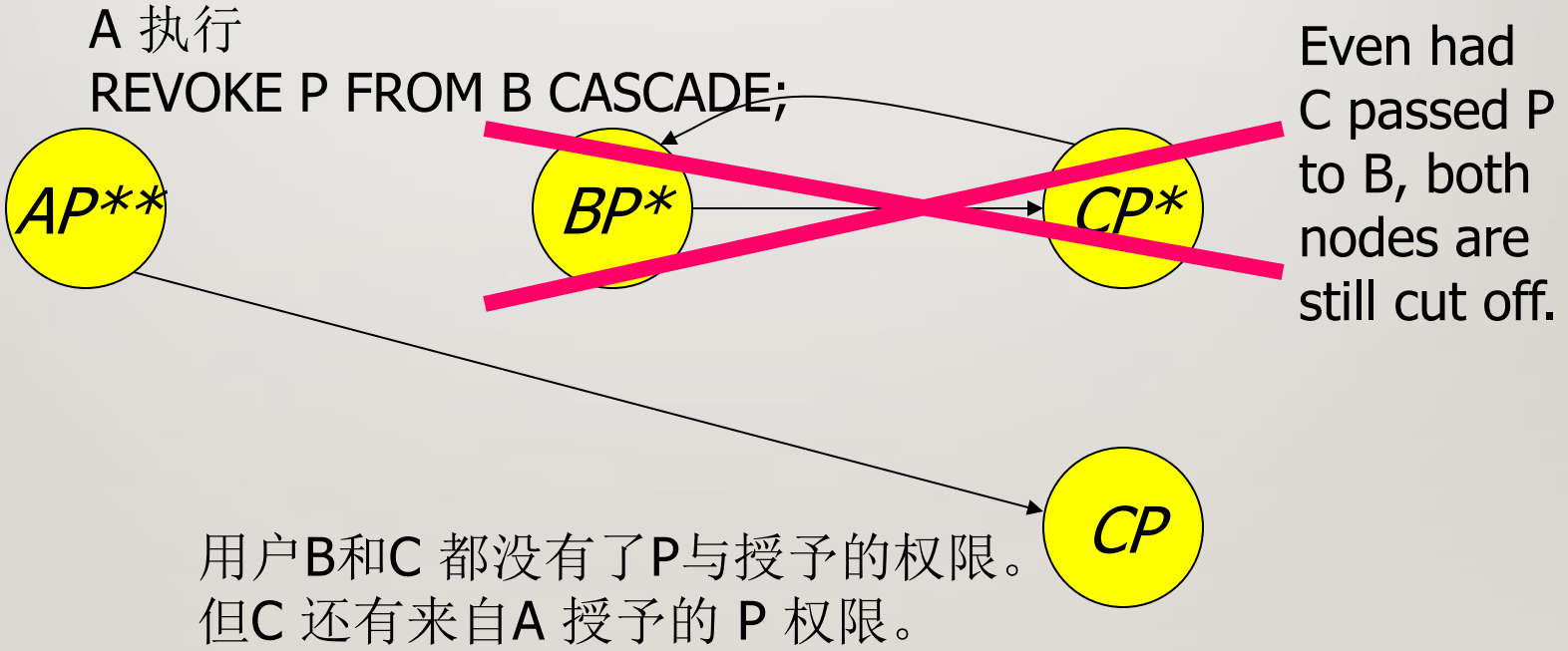
- 回收用户A在student表的sdept属性上的修改权限

```
Revoke Update(sdept)
    On Table student
    From A;
```

系统如何记录? 授权图


B: GRANT P
TO C WITH
GRANT OPTION
AP**
BP*
CP*
CP
A owns the
object on
which P is
a privilege
A: GRANT P
TO B WITH
GRANT OPTION
A: GRANT P
TO C

Example: Grant Diagram

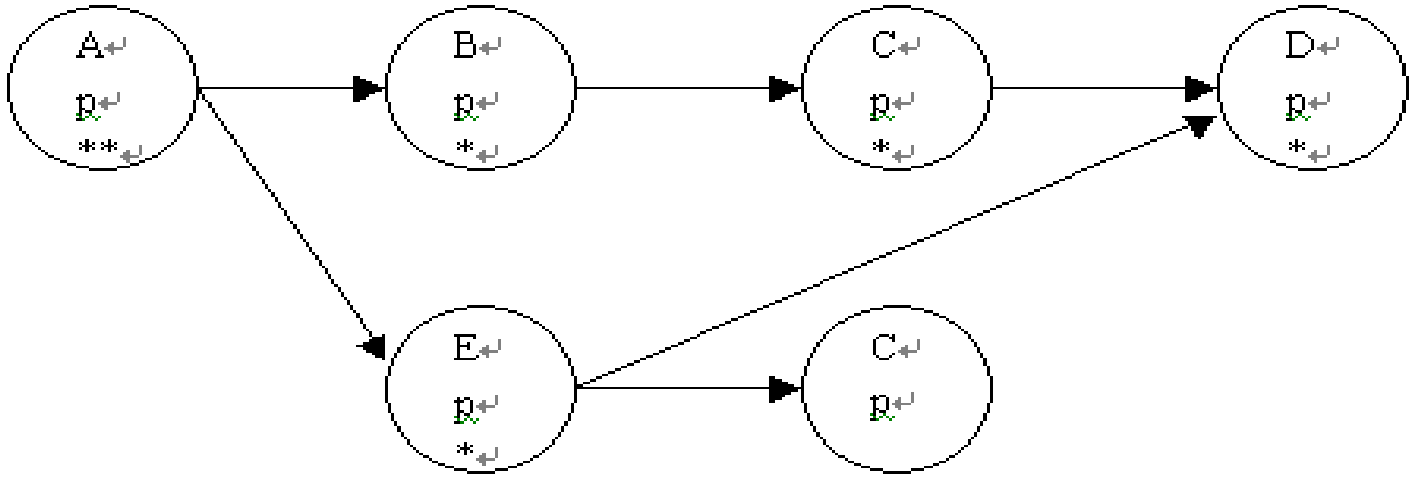


如果 A 执行下面的语句：

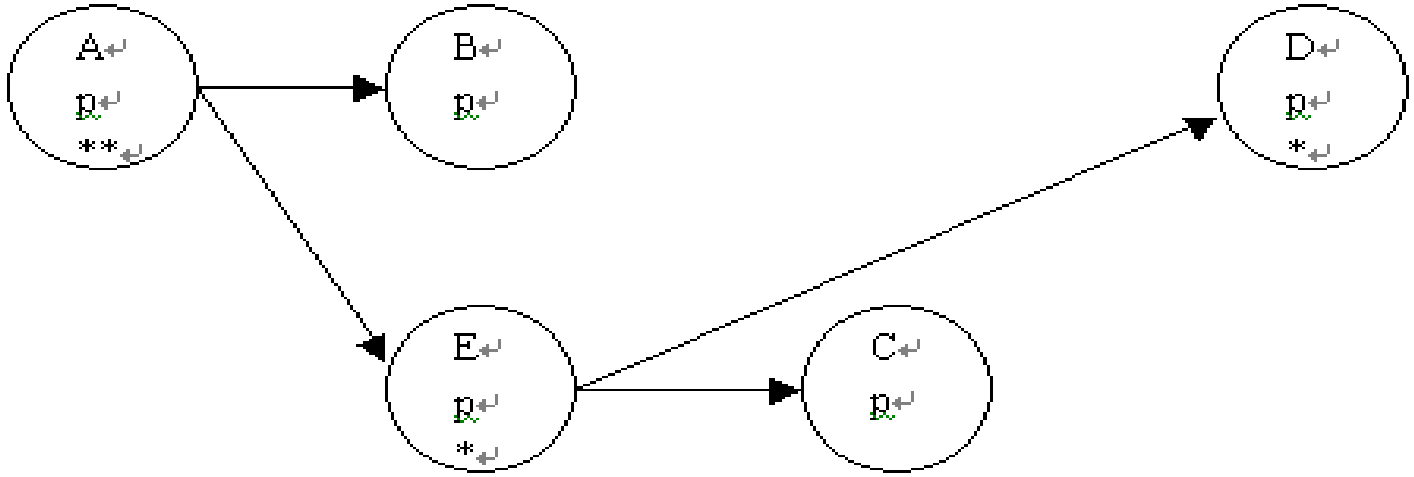
REVOKE P FROM B RESTRICT ??

课堂练习

1. User A(owner): grant p to B, E with grant option
2. User B: grant p to C with grant option
3. User C: grant p to D with grant option
4. User E: grant p to C
5. User E: grant p to D with grant option
6. User A: revoke grant option for p from B cascade



After step (6):



其它安全措施

- 视图机制

- 审计 (用于安全性较高的部门)

把用户对数据库的操作自动记录放在审计日志中.

- 数据加密

根据一定的算法将原始数据变换为不可直接识别的格式.

数据库的完整性

- 数据库的完整性指数据的**正确性**和**相容性**。
- 为防止数据库中存在不符合语义的数据，防止错误信息的输入和输出而制定的语义约束条件称为完整性约束条件。
- 检查数据是否满足完整性约束条件称为完整性检查。

完整性约束条件

- 完整性约束的对象可以是整个数据库、关系、元组和属性。
- 静态约束：数据库在每一确定状态时，数据对象应满足的约束条件。
- 动态约束：数据库从一种状态转变为另一种状态时应满足的条件。

完整性措施

- 实体完整性: 实体的码唯一,且不能为空。
- 参照完整性： 外码
- 用户自己定义的完整性

1. Check 子句（属性级别，表级别）
2. 断言
3. 触发器

完整性 举例

```
CREATE TABEL Student (
   SNO int PRIMARY KEY,
   sname char(6) NOT NULL,
   age int CHECK (age >=16 AND age < 50),    dept char(6)
 REFERENCES Department (dept) ON UPDATE CASCADE
 ON DELETE SET NULL );
```

级连修改
删除设置为空

用户自己定义的约束

属性上的约束:

- 列非空 : NOT NULL
- 列值唯一: UNIQUE
- 检查列值是否满足一个布尔表达式: CHECK

元组上的约束（表约束）： 检查元组是否满足一个布尔表达式: CHECK

基于属性的约束表达式

- 形式: CHECK (condition)

1) 条件中包含该属性.
2) 条件中可以包含其他属性，但只能在子查询中

- 何时检查呢？当该属性发生变化时，系统检查该约束条件。即，插入和修改操作时，检查该约束是否满足。

Example

```
CREATE TABLE SC (
 SNO CHAR (20) CHECK ( SNO IN SELECT SNO
   FROM STUDENT),
 CNO CHAR(20) CHECK ( CNO IN SELECT cno FROM
   course),
GRADE REAL CHECK ( grade >=0 and grade <=100)
 PRIMARY KEY (SNO,CNO));
```

基于元组的约束表达式

- 约束条件是基于多个属性
- 建表时说明为单独的约束条件，形式: 如 attribute-based check.

- 何时检查： 当数据发生变化，即插入，更新操作时。

Example

```
CREATE TABLE Student (
  sno CHAR (20) primary key,
  sname CHAR (20) not null,
  ssex CHAR(1),
  sage (int),
  sdept CHAR(20),
  CHECK (ssex= ‘女’  OR sname NOT LIKE ‘Ms.%’ ) );
```

定义了元组中姓名和性别之间的约束条件，避免错误的发生。

SQL Assertions 断言

- Database-schema 层次上的约束
- 有些数据库不支持.
- 断言中包含的数据库表发生变化，该断言被检查
- Syntax:

CREATE ASSERTION <name> CHECK (< condition>);

断言的实例

```
CREATE ASSERTION credit_check
   CHECK(
     30 <= ALL (SELECT SUM(credit)
                FROM SC,C
                WHERE SC.cno=C.cno
               GROUP BY sno )
  )
```

SC和C表的任何变化，系统都会自动检查该断言，是否满足条件。

各种约束的比较

约束类型	说明的地方	激活的条件	是否保证该约束条件？
Attribute-based Check	With attribute	On insertion to relation or attribute update	Not if subqueries
Tuple-based Check	Element of relation schema	On insertion to relation or tuple update	Not if subqueries
Assertion	Element of database schema	On change to any mentioned relation	Yes

命名约束

命名约束

例如：

1) ssex Char(1) **CONSTRAINT** NoAndro CHECK (ssex in (‘F’,’M’)),
2) Name Char(30) **CONSTRAINT** NameIsKey PRIMARY KEY,

修改约束

例如：

1) ALTER TABLE Student DROP CONSTRAINT NoAndro;

2) ALTER TABLE Student ADD CONSTRAINT NameIskey PRIMARY KEY(name);

触发器（Triggers）

基于 event-condition-action 规则

- Event= 指插入，删除和更新
- Condition= 指WHERE语句中的条件
- Action= 一系列的 SQL 语句

Triggers

- 不同于 checks, assertions and triggers.
- 由事件自动触发

具有更精细更强大的数据控制能力.
很多DBMS都实现了触发器,但语法梢有不同.

Triggers 标准语法

General form:

CREATE TRIGGER <name>

BEFORE | AFTER | INSTEAD OF <events>

FOR EACH ROW | STATEMENT //optional

<referencing clause> // optional

WHEN (<condition>) // optional

<action>

触发器语法

<events>: INSERT ON R

DELETE ON R

UPDATE [OF A1, A2, ..., An] ON R

<condition>: like general assertion

<action>: sequence of SQL statements

FOR EACH ROW: execute once for each tuple changed, if omitted, execute once for each relevant statement ("row-level" versus "statement-level") in either case, execute after statement completes

触发器语法（续）

REFERENCING <thing> AS <var>

<thing> can be:

OLD-TABLE - previous values of deleted or updated tuples for row-level or statement-level DELETE or UPDATE

NEW_TABLE - current values of inserted or updated tuples for row-level or statement-level INSERT or UPDATE

OLD - previous value of deleted or updated tuple row-level only DELETE or UPDATE

NEW - current value of inserted or updated tuple row-level only INSERT or UPDATE

触发器实例

Teacher (teacherID, name, age, salary)

```
CREATE TRIGGER updateSal
AFTER UPDATE ON Teacher
FOR EACH ROW
REFERENCING  OLD AS OldTuple,
                    NEW AS NewTuple
WHEN (OldTuple.salary>NewTuple.salary)
```

action {

```
UPDATE TEACHER
SET salary = OldTuple.salary
WHERE teacherID = NewTuple.teacherID
```

语句执行顺序:

1) 触发事件 update teacher
2) 触发器 updateSal 触发
3) 判断条件 WHEN语句
4) 满足 执行 action
5) 不满足,退出.
6) …

触发器的应用

```
CREATE TRIGGER 库存控制
AFTER UPDATE OF 库存量 ON 库存
REFERENCING NEW AS N
FOR EACH ROW
WHEN （N.库存量<N.库存下限
AND NOT EXISTS
（SELECT * FROM 在购定单 WHERE 零件号=N.零件号））
INSERT INTO 在购定单 VALUES（N.零件号，N.订购量，SYSDATE）；
```

触发器的删除

- DROP TRIGGER **updateSal ON TEACHER**

触发器的应用

- 提供完整性约束条件的检查
- 实现动态约束
- 使数据库变为主动数据库

小结

- 什么是数据的安全性?
- 安全性措施有哪些?
- 什么是完整性约束?
- 有哪些约束方法?

课堂练习

Database schema:

Student(sno, sname, gender,bdate,height)

Course(cno,lhour,Credit,Semester)

SC(sno,cno,grade)

用触发器来实现上述外码的约束

Foreign key declaration

```
CREATE TABLE SC (
  sno char(9) REFERENCES student (sno) ON DELETE
   CASCADE,
  cno CHAR(6) REFERENCES course(cno) ,
  grade INTEGER )
```

违背外码约束的几种情况

- Insert on SC table
- Delete on Course
- Delete on Student
- Update(sno,cno) on SC
- Update(cno) on Course
- Update(sno) on Student

INSERT ON SC TABLE

```
CREATE TRIGGER referential-check
AFTER INSERT ON SC
REFERENCING NEW ROW AS N
FOR EACH ROW
WHEN (NOT (EXISTS
  (SELECT * FROM STUDENT
   WHERE SNO=N.SNO) AND
   EXISTS (SELECT * FROM COURSE
   WHERE CNO=N.CNO)))
ROLLBACK;
```

rejection

DELETE ON STUDENT

```
CREATE TRIGGER student-delete
AFTER DELETE ON STUDENT
REFERENCING OLD ROW AS O
FOR EACH ROW
WHEN (EXISTS (SELECT * FROM SC WHERE SNO = O.SNO))
DELETE FROM SC
WHERE SNO = O.SNO;
```

Cascade deletion

课堂练习 （SQLITE ON 学生数据库）

- 完整性约束条件的实验
- 触发器实验：设计一个触发器，让每个CS新生都选择数据库原理作为学习课程

```
create trigger R1  after insert on students  for each row
when new.dept='cs'
begin
  insert into sc(sid,cid,cname) values (new.sid, 1,'database');
end;
```

```
Insert into students(sid,name,dept)
values(11,‘wangdong’,’cs’);

Select * from sc;
```