



第8章 数据库的安全和完整性约束

数据库的破坏一般来自：

- 1.系统故障；
- 2.并发所引起数据不一致；
- 3.人为的破坏；数据库的安全保护(security protection)
- 4.数据的语义错误或对DB的错误操作引起的并发所引起数据库不一致

数据完整性约束



8.1 数据库的安全保护

主要讨论计算机系统在保证数据库安全方面的技术措施。

DBMS建立在OS之上，OS应能保证数据库中的数据必须经由DBMS访问，而不允许用户越过DBMS，直接通过OS访问。



8.1.1 视图的定义和查询修改

- (1) 定义视图，可以限制各个用户的访问范围；**
- (2) 有些DBMS没有视图功能，但是系统可以根据用户的访问限制条件，自动的修改查询条件，使其只能在给定访问范围内查询。**



8.1.2 访问控制

访问控制(access control)是对用户访问数据库各种资源的权力的控制。

└ **基表、视图、各种目录以及实用程序等**
└ **创建、撤销、查询、增、删、改等**

在同一DBMS下，可能建立多个数据库，访问控制在数据库之间是相互独立的。



数据库用户：

- 1.一般数据库用户；
- 2.具有支配数据库部分资源权限的数据库用户；
- 3.具有DBA特权的数据库用户

DBMS须解决： 用户的标识与鉴别**以及授权**（grant--revoke）**的问题。**



在数据库中，许多用户的权限相同，如分别授权，十分繁琐，可以为他们定义一个角色。

对角色授权，某用户承担某种角色就拥有该角色的权限，一个用户可以拥有多个角色和其他权限。

角色不是用户，不能用做登陆！



8.3 完整性约束检查

8.3.1 完整性约束的类型

以关系数据模型为例分类。

1. 静态约束 (static constraints)

(1) **固有约束**(inherent constraints) —— **第一范式**

(2) **隐含约束**(implicit constraints)

用DDL说明，例如：域完整性、实体完整性、引用完整性等。

(3) **显式约束**(explicit constraints)

依赖于数据的语义和应用。



2. 动态约束 (dynamic constraints)

不是对数据库状态的约束，而是数据库从一个状态转到另一个状态时要遵守的约束。



8.3.2 完整性约束的说明

约束的显式说明方法：

1. 用过程说明约束

让应用程序完成约束的说明和检验。

缺点： 检验分散在应用程序中，增加程序员的负担，约束改变会导致程序要修改。

优点： 容易实现，目前应用较多。



2.用断言(assertions)说明约束

DBMS提供断言说明语言，用此语言可以写出数据库完整性断言，由系统编译成约束库(constraint base)。

DBMS的完整性控制子系统，对每个更新事务，用相关断言进行检查，如果发现违反约束，就回卷该事务。

例如：Assert 余额约束 on 储蓄帐：余额 ≥ 0



优点：集中控制，用户不编程，维护方便；

缺点：实现复杂，开销大，处理单一。

3.用触发子(triggers)表示约束

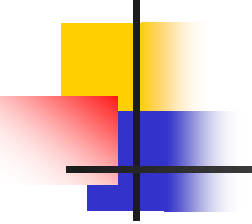
触发子是一种软件机制，形如：

whenever <条件> then <动作>

Event(激活触发器)

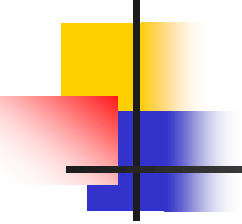
Condition(检验触发器的条件是否满足)

Actions(触发器运行后的动作)



传统的数据库系统只能按照用户或应用程序的要求，对数据库进行操作，而不能根据发生的事件或数据库的状态主动进行相应的处理，这样的数据库系统是被动的。

主动数据库系统就是具有主动数据库功能的数据库系统。



问题：主动数据库系统和关系数据库系统、面向对象数据库系统的区别和联系？

主动数据库只是数据库系统的一种功能！



假

设有下列三个关系：

Sailors(sid, sname, rating, birth, master)

/*分别为水手的编号、名字、级别、出生日期、师父的编号，每个水手的师父也是水手*/

Boats(bid, bname, color)

/*分别为船的编号、名字、颜色*/

Reserves(sid, bid, day)

/*分别为订船水手编号、所订船编号、日期*/



范例.引用完整性规则的实现

以Sailors,Boats,Reserves三张表为例，写出实现引用完整性约束的规则。

有哪些操作会影响到三张表间的引用完整性？

- Reserves表的Insert操作
- Reserves表的Update操作
- Sailors表的Delete操作
- Sailors表的Update操作
- Boats表的Delete操作
- Boats表的Update操作

是否对所有属性的Update操作都影响引用完整性？



规则1

创建触发器，对Reserves表的Insert操作进行监控，如果插入元组的外键属性在Sailors和Boats表中不存在，回卷插入该记录的操作。

Create trigger referential_integrity_check

Before Insert on Reserves

Referencing NEW as N

For Each Row



Event



```
When (not (exists(Select * From Sailors Where sid = N.sid)  
and  
(exists(Select * From Boats Where bid = N.bid))  
)
```

```
Rollback;
```

Action

Condition



规则2

创建触发器，对Boats表的Delete操作进行监控，如果删除元组的主键是Reserves表中的外键，回卷删除该记录的操作。

Create trigger boats_delete

Before Delete on Boats

Referencing OLD as O

For Each Row



Event



When (exists(Select * From Reserves
Where bid = O.bid))

Rollback;

Action

Condition



规则3

创建触发器，对Sailors表的Delete操作进行监控，如果删除元组的主键是Reserves表中的外键,则将Reserves表中的相关记录删除。

Create trigger sailors_delete

After Delete on Sailors

Referencing OLD as O

For Each Row



Event



When (exists(Select * From Reserves
Where sid = O.sid))

Delete From Reserves
Where sid = O.sid;

Condition

Action



规则4

创建触发器，对Reserves表的Update操作进行监控，如果修改元组sid和bid属性值在Sailors和Boats表中不存在，回卷修改该记录的操作。

Create trigger referential_integrity_check

Before Update of sid,bid on Reserves

Referencing NEW as N

For Each Row



Event



```
When (not (exists(Select * From Sailors Where sid = N.sid)
and
(exists(Select * From Boats Where bid = N.bid))
)
```

```
Rollback;
```

Action

Condition



规则5

Create trigger sailors_sid_update

Before Update of sid on Sailors

Referencing Old as O

For Each Row

When (exists(Select * From Reserves

Where sid = O.sid))

Rollback;

创建触发器，对Sailors表的Update操作进行监控，如果Reserves表中有元组引用修改前的sid值作为外键,回卷此修改操作。



规则6

监视Sailors表上的Insert操作，对每条Insert语句，判断其插入后的元组是否有年龄小于18的水手，将这样的水手自动插入到YoungSailors表中（YoungSailors表与Sailors表的模式相同）。

问题：与上面各题创建触发器的范例有什么不同？如何实现？



Create trigger young_sailor_update

After Insert on Sailors

Referencing New as N

For Each Statement

针对每条语句!

Insert into

YoungSailors(sid,name,age,rating)

Select sid,name,age,rating

From N

Where N.age<19 ;

