

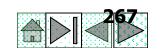
CREATE TABLE 部门

名称 VARCHAR (20),

经理名 CHAR(8),

(部门号 CHAR(3) PRIMARY KEY,

```
地址 VARCHAR (40),
CREATE TABLE 职工
(职工号 CHAR(5) PRIMARY KEY,
姓名 CHAR(8),
年龄 NUMBER(2) CHECK (年龄<=60),
职务 CHAR(10),
工资 NUMBER(7,2),
部门号 CHAR(3) FOREIGN KEY REFERENCES 部门(部门号)
);
```



- (2) 请用SQL的GRANT和REVOKE语句(加上视图机制)完成以下授权定义:
 - ①全体用户对两个表有SELECT权力。

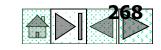
GRANT SELECT ON TABLE 职工, TABLE 部门 TO PUBLIC;

- ②用户U1、U2、U3对两个表有INSERT和DELETE权力。 GRANT INSERT, DELETE ON TABLE 职工, TABLE 部门 TO U1, U2, U3;
- ③用户刘明对职工表有SELECT权力,对工资属性列具有更新权力。 GRANT SELECT, UPDATE(工资) ON TABLE 职工 TO 刘明;
- ④用户吴新具有修改这两个表结构的权力。
 GRANT ALTER TABLE ON TABLE 职工, TABLE 部门 TO 吴新;
- ⑤用户李颖、周岚具有对两个表所有权力,并具有给其他用户授权的权力。 GRANT ALL ON TABLE 职工, TABLE 部门 TO 李颖, 周岚 WITH GRANT OPTION;
- ⑥用户杨青具有从每个部门职工中SELECT最高工资、最低工资、平均工资的权力,但他不能查看每个人的工资。

CREATE VIEW seldeptsal

AS SELECT 部门号, MAX(工资) 最高工资, MIN(工资) 最低工资, AVG(工资) 平均工资 FROM 职工 GROUP BY 部门号;

GRANT SELECT ON VIEW seldeptsal TO 杨青;



1

第6章 数据库保护与事务管理

7.

(1) 在关系Student中插入学生年龄值应在15~25之间

CHECK (age BETWEET 15 AND 25)

ALTER TABLE Student

<u>ADD CONSTRAINT</u> chk_Student_age CHECK (Sage >= 15 AND Sage <= 25);

(2) 在关系SC中插入元组时,其sno值和cno值必须分别在Student中和Course中出现

FOREIGN KEY (sno) REFERENCES student (sno)

FOREIGN KEY (cno) REFERENCES course (cno)

ALTER TABLE SC

ADD CONSTRAINT FK SC Student FOREIGN KEY (Sno) REFERENCES Student;

ALTER TABLE SC

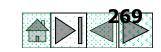
ADD CONSTRAINT FK_SC_Course FOREIGN KEY (Cno) REFERENCES Course;

(3) 在关系SC中修改grade值时,必须仍在0~100之间

CHECK (grade BETWEET 0 AND 100)

ALTER TABLE SC

ADD CONSTRAINT chk_SC_grade CHECK (grade >= 0 AND grade <= 100);





7.

(4) 在删除关系Course中一个元组时,首先要把关系SC中具有同样cno值的元组全部删去

FOREIGN KEY(cno) REFERENCES course(cno) ON DELETE RESTRICT
ALTER TABLE SC

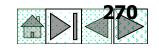
ADD CONSTRAINT FK_SC_Course FOREIGN KEY (Cno) REFERENCES Course ON <u>DELETE RESTRICT</u>;

(5) 在关系Student中把某一个sno值修改为新值时,必须同时把关系SC中那些同样的sno值也修改为新值

FOREIGN KEY(sno) REFERENCES student(sno) ON UPDATE CASCADE

ALTER TABLE SC

ADD CONSTRAINT FK_SC_Student FOREIGN KEY (Sno) REFERENCES Student ON UPDATE CASCADE;



学生必须在选修了数学课以后,才能选修其他的课程。

```
CREATE ASSERTION assel CHECK

( NOT EXISTS ( SELECT sno
FROM s_c X

WHERE NOT EXISTS ( SELECT *

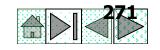
FROM s_c Y, course

WHERE Y. cno=course. cno AND
Y. sno=X. sno AND cname='物学')))
```

每个艺术系的学生最多选修20门课程

CREATE ASSERTION asse2 CHECK

(20>= SELECT COUNT(SC. CNO) FROM STUDENT, SC WHERE STUDENT. SNO=SC. CNO
AND STUDENT. DEPT="艺术系"GROUP BY SNO)



16. 设A,B的初值均为2,设T1,T2是以下两个事务:

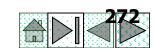
事务**T1**: 读**B**; **A=B+1**; 写回**A**

事务**T2**: 读**A**; **B**=**A**+**1**; 写回**B**

现给出对这两个事务不同的调度策略。

T ₁	T ₂
Slock B	
Y=R(B)=2	
Unlock B	
Xlock A	
A=Y+1=3	
W(A)	
Unlock A	
	Slock A
	X=R(A)=3
	Unlock A
	Xlock B
	B=X+1=4
	W(B)
	Unlock B

按T1→T2次序执行结果为 A=3, B=4 串行调度策略,正确的调度



16. 设A, B的初值均为2,设T1,T2是以下两个事务:

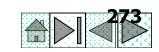
事务**T1**: 读**B**; **A=B+1**; 写回**A**

事务**T2**: 读**A**; **B**=**A**+**1**; 写回**B**

现给出对这两个事务不同的调度策略。

T ₁	T ₂
	Slock A
	X=R(A)=2
	Unlock A
	Xlock B
	B=X+1=3
	W(B)
	Unlock B
Slock B	
Y=R(B)=3	
Unlock B	
Xlock A	
A=Y+1=4	
W(A)	
Unlock A	

T2→T1次序执行结果 为B=3, A=4 串行调度策略,正确的 调度



16. 设A,B的初值均为2,设T1,T2是以下两个事务:

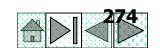
事务**T1**: 读**B**; **A**=**B**+**1**; 写回**A**

事务**T2**: 读**A**; **B**=**A**+**1**; 写回**B**

现给出对这两个事务不同的调度策略。

T_1	T ₂
Slock B	
Y=R(B)=2	
	Slock A
	X=R(A)=2
Unlock B	
	Unlock A
Xlock A	
A=Y+1=3	
W(A)	
	Xlock B
	B=X+1=3
	W(B)
Unlock A	
	Unlock B

A=3 B=3 执行结果与**前面**的结果 都不同 是错误的调度



16. 设A,B的初值均为2,设T1,T2是以下两个事务:

事务**T1**:读**B**; **A=B+1**;写回**A**

事务**T2**: 读**A**; **B**=**A**+**1**; 写回**B**

现给出对这两个事务不同的调度策略。

T ₁	T ₂
Slock B	
Y=R(B)=2	
Unlock B	
Xlock A	
	Slock A
A=Y+1=3	等待
W(A)	等待
Unlock A	等待
	X=R(A)=3
	Unlock A
	Xlock B
	B=X+1=4
	W(B)
	Unlock B

A=3 B=4 执行结果与第一种串行 调度的执行结果相同 是正确的调度

16. 设A,B的初值均为2,设T1,T2是以下两个事务:

事务**T1**: 读**B**; **A=B+1**; 写回**A**

事务T2: 读A; B=A+1; 写回B

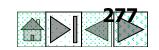
现给出对这两个事务不同的调度策略。

事务T₁	事务T ₂
Slock B	
R(B)=2	
	Slock A
	R(A)=2
Xlock A	
等待	Xlock B
等待	等待

遵守两段锁协 议,发生死锁



- 17.设A的初值为0, T1、T2、T3是如下的三个事务:
 - $T1:A=A+2; T2:A=A*2; T3:A=A^2$
- (1)若这三个事务允许并发执行,则有多少种可能的正确调度结果,请——列举出来;
- (2)写出一个可串行化的并发调度和结果;
- (3)写出一个非可串行化的并发调度和结果;
- (4)若这三个事务都遵守两段锁协议,请给出一个不产生死锁的可串行化调度;
- (5)若这三个事务都遵守两段锁协议,请给出一个产生死锁的调度。





解: (1) 4 种 A=16,8,4,2

T1 - T2 - T3 A=16

T1 - T3 - T2 A = 8

T2-T1-T3 或 T3-T1-T2 A=4

T2-T3-T1 或 T3-T2-T1 A=2

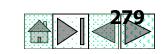


(2) 一个可串行化的调度及执行结果如下图所示:

至少有8种可串行化调度:

时间	T1	T2	Т3
t1	Slock A		
t2	X=A=0		
t3	Unlock A		
t4	Xlock A		
t5		Slock A	
t6	A=X+2	等待	
t7	写回 A (=2)	等待	
t8	Unlock A	等待	
t9		获得 Slock A	
t10		X=A=2	
t11		Unlock A	
t12		Xlock A	
t13			Slock A
t14		A=X*2	等待
t15		写回 A (=4)	等待
t16		Unlock A	等待
t17			获得 Slock A
t18			X=A=4
t19			Unlock A
t20			Xlock A
t21			$A=X^2$
t22			写回 A(=16)
t23			Unlock A

执行结果为 A=16, 是可串行化的调度。

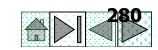




(3) 一个非串行化调度及执行结果如下图所示:

时间	T1	T2	Т3
t1	Slock A		
t2	X=A=0		
t3	Unlock A		
t4		Slock A	
t5		X=A=0	
t6	Xlock A		
t7	等待	Unlock A	
t8	获得 Xlock A		
t9	A=X+2		
t10	写回 A (=2)		Slock A
t11	Unlock A		等待
t12			获得 Slock A
t13			X=A=4
t14			Unlock A
t15			Xlock A
t16		Xlock A	
t17		等待	$A=X^2$
t18		等待	写回 A(=4)
t19		等待	Unlock A
t20		获得 Xock A	
t21		A=X*2	
t22		写回 A (=0)	
t23		Unlock A	

运行结果 A=0, 为非串行化调度。



(4) 若三个串行事务都遵守两段锁协议,下图是按 T3-T1-T2 顺序运行的一个不产生死锁的可串行化调度;

时间	T1	T2	T3
	11	12	
t1			Slock A
t2			X=A=0
t3			Xlock A
t4	Slock A		$A=X^2$
t5	等待		写回 A (=0)
t6	等待		Unlock A
t7	X=A=0		
t8	Xlock A		
t9	等待	Slock A	Unlock A
t10	A=X+2	等待	
t11	写回 A (=2)	等待	
t12	Unlock A	等待	
t13		X=A=2	
		Xlock A	
	Unlock A	等待	
		A=X*2	
		写回 A (=4)	
		Unlock A	
		Unlock A	

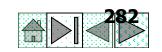
从上可见,按照 T3-T1-T2 的顺序执行结果 A=4 完全与串行化调度相同,所以是一个不产生死锁的可串行化的调度。



(5) 若三个事务都遵守两段锁协议,下图是一个产生死锁的调度。

时间	T1	T2	Т3
t1	Slock A		
t2	X=A=0		
t3		Slock A	
t4		X=A=0	
t5	Xlock A		
t6	等待		
t7		Xlock A	
t8		等待	
			Slock A
			X=A=0
			Xlock A
			等待

上例中, T1 申请对 X1 加写锁,由于 T2 对 X1 加了读锁,所以不成功,处于等待状态; T2 申请对 A 加写锁,由于 T1 对 A 加了读锁,所以不成功,处于等待状态; T3 申请对 A 加读锁,由于 T1 对 A 加了读锁,所以不成功,处于等待状态。因此,三个事务都处于等待状态,产生死锁。





考虑如下的调度,说明这些调度之间的包含关系。

- (1) 正确的调度;
- (2) 可串行化的调度;
- (3) 遵守两阶段封锁(2PL)的调度;
- (4) 串行调度。

包含关系如下:

串行调度 ⊂ 正确的调度

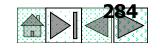
考虑T₁和T₂两个事务:

 $T_1: R(A); R(B); B=A+B; W(B)$

 T_2 : R(B); R(A); A=A+B; W(A)

(1) 改写T1和T2,增加加锁操作和解锁操作,并要求遵守两阶段封锁协议。

T_1	T ₂
Slock A;	Slock B
R(A)	R(B)
Slock B	Slock A
R(B)	R(A)
B=A+B	A=A+B
Xlock B	Xlock A
W(B)	W(A)
Unlock A	Unlock B
Unlock B	Unlock A





(2) 说明T1和T2的执行是否会引起死锁,给出T1和T2的一个调度并说明之。会的,示例如下:

T ₁	T ₂
SLock A R (A) Slock B R (B) B=A+B	
XLock B 等待	SLock B R(B) Slock A R(A) A=A+B XLock A 等待
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	