tags: 112 學年 上學期 讀書計畫

資料庫系統概論作業 HW 3 報告

2-1. Query Processing (39%)

2-1-1. select operation (12%, 4% each)

type 1

```
for each tuple s in Staff
    for each tuple b in Branch
        check if s.position = 'Manager' and
                 b.city = 'London' and
                 s.branchNo = b.branchNo
```

type 2

```
op1 <- empty set
for each tuple s in Staff
    for each tuple b in Branch
        if s.branchNo = b.branchNo then
                op1 <- op1 union {s,b} (except b.branchNo attribute)

for each tuple item in op1
    check if item.city = 'London' and item.position = 'Manager'
```

type 3

```
tempStaff <- empty set
for each tuple item in Staff
    if item.position = 'Manager' then
        tempStaff <- tempStaff union {item}

tempBranch <- empty set
for each tuple item in Branch
    if item.city = 'London' then
        tempBranch <- tempBranch union {item}

for each tuple s in tempStaff
    for each tuple b in tempBranch
        check if s.branchNo = b.branchNo
```

2-1-2. join operation (27%, 9% each)

a.

We will choose $R := r_2$ as the outer relation, because this way can minimize the block transfer.

We calculate some information first.

$$R := r_2,\ S := r_1$$
$$b_r = 40000/20 = 2000,\ b_s = 30000/10 = 3000,$$
$$n_r = 40000,\ n_s = 30000.$$

block transfer

We scan R tutples once: b_r
For each tuple in R, must scan S: $n_r \times b_s$

Total =
$b_r + n_r \times b_s = 3000 + 30000 \times 2000 = 6 \times 10^7 + 3 \times 10^3 = 60003000$

seeks

$n_r + 1$ (find S header cost n_r times and find R header cost 1 times)

Total = $n_r + 1 = 30000 + 1 = 30001$

pseudo code

```
for each tuple r in R # b_r transfer
    for each tuple s in S  # n_r * b_s transfer
        check if r.C == s.C
```

b.

Since the pseudo code is as the a., therefore, the number of block transfer is the same.

We will choose $R := r_2$ as the outer relation, because this way can minimize the block transfer.

We calculate some information first.

$R := r_2,\ S := r_1$
$b_r = 40000/20 = 2000,\ b_s = 30000/10 = 3000,$
$n_r = 40000,\ n_s = 30000.$

block transfer

We scan R tutples once: b_r
For each tuple in R, must scan S: $n_r \times b_s$

Total =
$b_r + n_r \times b_s = 3000 + 30000 \times 2000 = 6 \times 10^7 + 3 \times 10^3 = 60003000$

seeks

$n_r \times b_s + b_r$ (find S header cost $n_r b_s$ times and find R header cost b_r times)

Total = $n_r b_s + b_s = 60003000$

pseudo code

```
for each tuple r in R # b_r transfer
    for each tuple s in S  # n_r * b_s transfer
        check if r.C == s.C
```

c.

We will choose $r := r_1$ as the outer relation, because this way can minimize the block transfer.

We calculate some information first.

$$R := r_1,\ S := r_2$$
$$b_r = 40000/20 = 2000,\ b_s = 30000/10 = 3000$$
$$n_r = 40000,\ n_s = 30000.$$

block transfer

We scan S tutples once: $b_s/100$

Total = $b_r \times (b_s/100) + b_r = 2000 \times 30 + 2000 = 62000$

seeks

$b_r \times b_s/100 + b_r$ (find S header cost $b_r \times b_s/100$ times and find R header cost b_r times)

$b_r \times (b_s/100) + b_r = 2000 \times 30 + 2000 = 62000$

pseudo code

```
for each block in R # transfer b_r times
    for each group of 100 blocks in S # for each b_r, read S b_s / 10
        for each tuple r in one block
            for each tuple s in each group of 100 blocks in S
                check if r.C = s.C
```

2-2. 2PL (61%)

2-2-1 (21%)

T1	T2	T3	T4
lock-S(B)	lock-S(B)		lock-S(C)
Read(B)	Read(B)		Read(C)
	lock-S(A)		lock-S(A)
	Read(A)		Read(A)
	Unlock(A)		Unlock(C)
	Unlock(B)		Unlock(A)
lock-X(C)		lock-X(A)	
Write(C)		Write(A)	
Unlock(C)		lock-X(B)	
Unlock(B)		Write(B)	
		Unlock(A)	
		Unlock(B)	

2-2-2 (40%, 10% each)

a.

2-2-2 (40%, 10% each)

a. This schedule is under 2PL protocol, please find out where deadlock happens, if any, explain why.

T1	T2	T3
lock-X(A)	not Unlock(A)	
Read(A)		not Unlock(C)
		lock-X(C)
		Read(C)
	lock-X(B)	
wait	Read(B)	not Unlock(B)
lock-S(C)		
Read(C)		
	lock-S(A)	wait
	Read(A)	
	wait	lock-S(B)
		Read(B)
Write(A)		
Unlock(A)		
Unlock(C)		
	Write(B)	
	Unlock(B)	
	Unlock(A)	
		Write(C)
		Unlock(C)
		Unlock(B)

There is a cycle.
T3 -> T2 -> T1 - >T3.

b.

Let priority $T_1 > T_2 > T_3$.

row number	T1	T2	T3	reason
1	lock-X(A)			
2	Read(A)			
3			lock-X(C)	
4			Read(C)	
5		lock-X(B)		
6		Read(B)		
7	lock-S(C) wait for T3			since T3 lock-X(C)
8	Read(C)			
9		lock-S(A) abort all		since T1 lock-X(A)
10		Read(A)		
11			lock-S(B)	
12			Read(B)	
13	Write(A)			
14	Unlock(A)			
15	Unlock(C)			
16		Write(B)		
17		Unlock(B)		
18		Unlock(A)		
19			Write(C)	
20			Unlock(C)	
21			Unlock(B)	

c.

Let priority $T_1 > T_2 > T_3$.

row number	T1	T2	T3	reason
1	lock-X(A)			
2	Read(A)			
3			lock-X(C)	
4			Read(C)	
5		lock-X(B)		
6		Read(B)		
7	lock-S(C)		abort all	since T3 lock-X(C)
8	Read(C)			
9		lock-S(A) wait for T1		since T1 lock-X(A)
10		Read(A)		
11			lock-S(B)	
12			Read(B)	
13	Write(A)			
14	Unlock(A)			
15	Unlock(C)			
16		Write(B)		
17		Unlock(B)		
18		Unlock(A)		
19			Write(C)	
20			Unlock(C)	
21			Unlock(B)	

d.

Let priority $T_1 < T_2 < T_3$.

row number	T1	T2	T3	reason
1	Read(A)			
2			Read(C)	
3		Read(B)		
4	Read(C)			
5		Read(A)		
6			Read(B)	
7	Write(A) abort			since step 5 have read A
8		Write(B) abort		since step 6 have read B
9			Write(C)	