Introduction to Database Systems
Homework 3 : Query Processing & 2PL

1. Query Processing:

2-1-1 Select Operation:

(1) (1000+50) +2* (1000*50) =101050

σ (position='Manager') ^ (city='London') ^ (Staff.branchNo=Branch.branchNo) (Staff X Branch)

pseudo code:

```
result_set = []

for each staff_member in Staff
     for each branch in Branch

          if staff_member.branchNo == branch.branchNo AND
              staff_member.position == 'Manager' AND
              branch.city == 'London'

               result_set.add ( (staff_member, branch) )

return result_set
```

(2) 2*1000+(1000+50)=3050

$\sigma_{(position='Manager') \wedge (city='London')}$ (Staff $\bowtie_{Staff.branchNo=Branch.branchNo}$ Branch)

pseudo code:

```
result_set = []

for each staff_member in Staff
     if staff_member.position == 'Manager'
          for each branch in Branch
               if branch.city == 'London' AND staff_member.branchNo ==
branch.branchNo

                    result_set.add((staff_member, branch))
return result_set
```

(3) 1000+2*50+5+(50+5)=1160

$(\sigma_{(position='Manager')}(Staff)) \bowtie_{Staff.branchNo=Branch.branchNo} (\sigma_{city='London'}(Branch))$

Pseudo code:

```
result_set = []

selected_staff = []
for each staff_member in Staff
    if staff_member.position == 'Manager'
        selected_staff.add(staff_member)

selected_branches = []
for each branch in Branch
    if branch.city == 'London'
        selected_branches.add(branch)

for each manager in selected_staff
    for each london_branch in selected_branches
        if manager.branchNo == london_branch.branchNo
            result_set.add((manager, london_branch))

return result_set
```

2-1-2 Join Operation:

a.Assume that we implement Nested-loops join, and both are sorted. If there are only 3 memory blocks available and the minimal block transfers are required, what is the outer relation that we should pick from r1 and r2, and the number of block transfers and seeks required. Explain your answer with pseudo code. (3% for each answer)

- Pseudo code:

```
outer_relation = r2  (3000 blocks)
inner_relation = r1  (2000 blocks)

// Read each tuple in the block of outer_relation
Seek =seek+1
for each tuple_outer in block of outer_relation (r2)
     block_transfer= block_transfer+1
     //for each tuples
     For data in block
          Seek = seek+1
          // Read each tuple in the block of inner_relation
          for each tuple_inner in block of inner_relation (r1)
               block_transfer= block_transfer+1
```

- Total block transfers = nr*bs+br=30000*2000+3000=60003000
- Total block seeks = br+nr =1+30000=30001

b. Assume that we implement Nested-loops join, and none of the sorted. If there are 102 memory blocks available and the minimal block transfers are required, what is the outer relation that we should pick from r1 and r2, and the number of block transfers and seeks required. Explain your answer with pseudo code. (3% for each answer)

- Pseudo Code:
  Outer relation = r2 (3000 blocks)
  Inner relation =r1 (2000 blocks)

  ```
  // Read each tuple in the block of outer_relation
  for each tuple_outer in block of outer_relation (r2)
        Seek =seek+1
        block_transfer= block_transfer+1
        // Read each tuple in the block of inner_relation
        for each chunk of blocks of inner_relation (r1)
              Seek =seek+100
              block_transfer= block_transfer+1
  ```

- Total block transfers: br+nr* (bs/102-2) =3000+30000 (2000/100) =3000+600000=603000
- Total block seeks: br+nr*bs=3000+30000*2000=3000+60000000=60003000

c. Assume that we implement Block Nested-loops join, and none of the sorted. If there are only 102 memory blocks available, what is the outer relation that we should pick from r1 and r2, and the number of block transfers and seeks required. Explain your answer with pseudo code.(3% for each answer)

- Pseudo code:

```
outer_relation = r1  (2,000 blocks)
inner_relation = r2  (3,000 blocks)

for each chunk of blocks in r1
    Seek =seek+1
    block_transfer= block_transfer+1
    //load chunk of outer_relation into buffer
    for each block in r2
        Seek =seek+100
        block_transfer= block_transfer+1
        for each tuple_outer in buffer of r1
            for each tuple_inner in block of r2
```

- Total blocks transfers : br+br*(bs/102-2)=2000+2000*(3000/100)=2000+60000=62000
- Seeks: br*bs+br=2000*3000+2000=6002000

2. 2PL:

2-2

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

Minimum time is 15s.

2-2-2

a.

T1 開始並鎖定了 **A**（排他鎖定），然後讀取 **A**，接著請求對 **C** 的共享鎖定。
T2 鎖定了 **B**（排他鎖定），然後讀取 **B**，接著請求對 **A** 的共享鎖定。
T3 鎖定了 **C**（排他鎖定），然後讀取 **C**，接著請求對 **B** 的共享鎖定。

從這個情況中，我們可以看到以下死鎖循環：

- **T1** 在等待 **T3** 釋放 **C**。
- **T3** 在等待 **T2** 釋放 **B**。
- **T2** 在等待 **T1** 釋放 **A**。

這形成了一個經典的死鎖循環，其中每個事務都在等待一個由另一個事務持有的資源，而這些事務又相互等待對方，形成了一個無法打破的等待循環。沒有一個事務能夠進行下去，因為它們都在等待其他事務釋放資源。

b.

T1	T2	T3
Lock-x (A)		
Read (A)		
		Lock-X (C)
		Read (C)
Wait for T3		
Wait for T3		
	(Abort)	
		Lock-S (B)
		Read (B)
Wait for T3		
Wait for T3		
Wait for T3		
	Write (B)	
	Unlock (B)	
	Unlock (A)	
		Write (C)
		Unlock (C)
		Unlock (B)
Lock-s (C)		
Read (C)		
Write (A)		
Unlock (A)		
Unlock (C)		
	(T2 restart) Lock-X (B)	
	Read (B)	
	Lock-s (A)	
	Read (A)	
	Write (B)	
	Unlock (B)	
	Unlock (A)	

c.

T1	T2	T3
Lock-x (A)		
Read (A)		
	Lock-X (B)	
	Read (B)	
Lock-s (C)		(Abort)
Read (C)		
	Wait for T1	
	Wait for T1	
Write (A)		
Unlock (A)		
Unlock (C)		
	Wait for T1	
	Wait for T1	
	Wait for T1	
	Lock-s (A)	
	Read (A)	
	Write (B)	
	Unlock (B)	
	Unlock (A)	
		(T3 restart) Lock-X (C)
		Read (C)
		Lock-s (B)
		Read (B)
		Write (C)
		Unlock (C)
		Unlock (B)

d.

T1	T2	T3
Lock-x (A)		
Read (A)		
		Lock-X (C)
		Read (C)
	Lock-X (B)	
	Read (B)	
Lock-s (C) (abort)		
Read (C)		
	Lock-s (A) (abort)	
	Read (A)	
		Lock-S (B)
		Read (B)
Write (A)		
Unlock (A)		
Unlock (C)		
	Write (B)	
	Unlock (B)	
	Unlock (A)	
		Write (C)
		Unlock (C)
		Unlock (B)