CS 5/7330
Fall 2021
Homework 2

Due: Oct. (Late deadline: Dec 2nd 11:59pm. (Late deadline Dec 5th , 11:59pm)

1. Consider the following query

    SELECT *
    FROM A, B
    WHERE A.x = B.x

    Calculate (or provide lower and/or upper bound, if you don’t have enough information) the number of tuples returned from the query, under the following conditions (each part is separate from the other). Unless otherwise stated, table A has 10000 tuples, table B has 20000 tuples.

    a. A.x is the primary key of x, and B.x is the primary key of B
    b. A.x is the primary key of x, and B.x is the foreign key refer to A.x (but B.x is not unqiue)
    c. A.x and B.x is evenly distributed between 1-100.
    d. A.x is evenly distributed between 1-100, B.x is evenly distributed between 51-150.

2. Consider the following schedule for four transactions:

Time	T1	T2	T3	T4
1			X3 = Read(X)	
2	Y1 = Read(Y)			
3				Z4 = Read(Z)
4		X2 = Read(X)		
5			X3 = X3 + 1	
6	Z1 = Read(Z)			
7		X2 = X2 * 2		
8	Z1 = Z1 + Y1			
9	Write(Z, Z1)			
10				Z4 = Z4 * 3
11			Y3 = Y3 – X3	
12			Write(X, Y3)	
13		Y2 = Read(Y)		
14		Y2 = Y2 + 4		
15		Write(Y, Y2)		
16				Write(Z, Z4)
17		Write(X,X2)		

How to read the table:

- X, Y, Z are data stored on the database in the disk
- X1..X4, Y1..Y4, Z1..Z4 are local variables stored in memory and only accessible via the corresponding transactions (T1 for X1, Y1, T2 for X2, Y2 etc.). You can assume all local variables are initialized to 0.
- The Read() operation read in the corresponding data on the disk and store it in the local variable
- All the operations (*, +, - etc.) only operate on the local variable, and do not affect the data on the disk
- The Write() command write the value of the local variable onto the corresponding data on the disk.

a. Is the above schedule serializable? If it is, list all possible equivalent serial schedule. If not, explain why not.
b. Now suppose we want to implement 2-phase locking on the transactions. Assume we use the following convention:
   i. Every time a transaction wants to read/write an item in the disk for the first time, it will request a share/exclusive lock (respectively). We denote it as the operation S-Lock(), X-Lock() respectively
   ii. If a request is granted, the transaction can proceed immediately and execute the corresponding command
   iii. Otherwise, the transaction will use the wait-die policy to determine whether it will wait or abort.
   iv. If a transaction waits, it will wait until the lock is released and attempt to obtain it. It will try to obtain the lock immediately after it was released by another transaction. If it can obtain the lock, it will execute all commands up to the time that the lock is requested/obtained.
   v. If a transaction aborts, it will not be restarted
   vi. If there is more than one transaction waiting for the same lock, the lock is always granted to the transaction T_i with the smallest value of i.
   vii. We assume a transaction immediately commits and release all the locks after the last operation is successfully executed.

Below is a simple example

	Before adding lock command			After adding lock command		
	T1	T2	T3	T1	T2	T3
1	X1 = Read(X)			S-Lock(X); X1=Read(X)		
2		X2 = Read(X)			S-Lock(X); X2=Read(X)	
3			Y3 = Read(Y)			S-Lock(Y); Y3=Read(Y)

4			Y3 = Y3 + 1			Y3 = Y3 + 1
5			Write(Y, Y3)			X-Lock(Y); Write(Y, Y3)
6		Y2 = Read(Y)			S-Lock(Y); Wait	
7	Y1= Read(Y)			S-Lock(Y); Wait		
8	Y1 = Y1 + 1			---		
9			Z1 = Read(Z)			S-Lock(Z); Z1=Read(Z)
10			Z1=Z1+1			Z1=Z1+1
11			Write(Z, Z1)			X-Lock(Z); Write(Z, Z1); End T3,
				S-Lock(Y); Y1= Read(Y); Y1 = Y1 + 1		
12		X2 = Y2 + 1			----	
13	Write(Y, Y1)			X-Lock(Y); Write(Y, Y1)		

Apply the same notation to the 4 transactions on the first table, and show the execution in the table form as above.