DB2 and DBSA – April 11, 2017 – First Intermediate Test – Amended version

You can answer this test in English, Italian, or any mixture

1. Consider a schema R(<u>IdR</u>, A, B, …, IdS*), S(<u>IdS</u>,…), T(<u>IdT</u>, …, IdS*) and the following query

   SELECT *
   FROM R, T
   WHERE R.IdS = T.IdS And R.A ≤ 10 And R.B ≤ 200

Assume that R, S and T are stored as heap files. Primary keys are R.IdR, S.IdS and T.IdT, while R.IdS abd T.IdS are foreign keys that refers to S.
Assume that unclustered RID-sorted index are defined on all the primary and foreign keys, on R.A, on R.B, and a combined index on (R.A,R.B). Assume that every index on R has 4.000 leaves, every index on S has 10.000 leaves, every index on T has 200 leaves **(some part of this information is not needed for this exercise)**.
Assume the following table for the optimization parameters. If you need Cardenas formula Φ(n,k), approximate it with min(n,k).
Assume that every page is 4.000 bytes long, and that every attribute uses 4 bytes.

	NReg	NPag	NLeaf of Indexes	NKey	Min	Max
R	2.000.000	100.000	4.000			
S	5.000.000	500.000	10.000			
T	100.000	1.000	200			
Idx.R.A			See R	100	0	1.000
Idx.R.B			See R	10	0	1.000

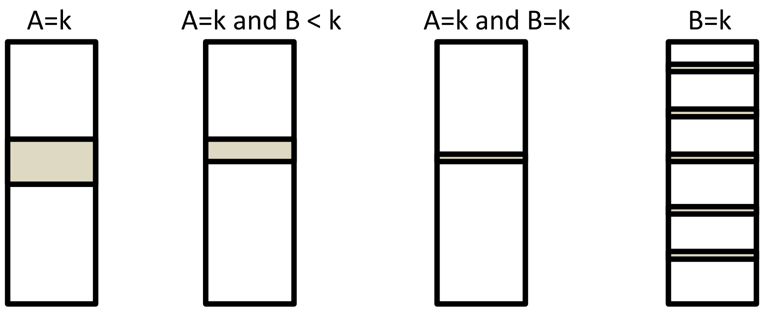
   a) Compute the cost of accessing all records of R that satisfy the condition R.A ≤ 10 And R.B ≤ 200 using:
      a. No index
      b. Index on R.A only
      c. Index on R.B only
      d. Both R.A and R.B indexes
   b) Using the cheapest access plan from (a), compute the cost of an IndexNestedLoop plan for the complete query, where R is the outer relation
   c) Compute the cost of an IndexNestedLoop plan for the complete query, where R is the inner relation
   d) How many records are in the result of the query?

2. Consider a combined index on attributes (A,B) of a table T. Assume the standard hypothesis of uniformity and independence of T.A and T.B, and also that, for every a in π_A(T) and b in π_B(T), the pair (a,b) belongs to π_{AB}(T). The following table reports the cost of accessing the data if the index is unclustered with sorted RIDs, if it is clustered on (A,B), or if it is unclustered with unsorted RIDs. The table assumes that RIDS are used to fetch the record in the same order as they are found in the index, we do not merge different RID lists and we do not sort RIDs in main memory.

We use the following abbreviations for the involved selectivity factors – Ae stands for 'equality condition on A' and Ar 'range condition on A'.
sfAe = 1/Nkey(A)
sfAr = (k – min(A)) / (max(A)-min(A))
sfBe = 1/Nkey(B)
sfBr = (k' – min(B)) / (max(B)-min(B))
sfXxYy = sfXx * sfYy

Condition	Unclustered sorted RIDs	Clustered on A, B	Uncl. Unsorted RIDs
A=k	Nkey(B) *⌈Φ(sfAeBe*Nrec,Npag)⌉	sfAe*Npag	
A<k	⌈sfAr*NKey(A)*Nkey(B)⌉ *⌈Φ(sfAeBe*Nrec,Npag)⌉	sfAr*Npag	
B=k'	Nkey(A) *⌈Φ(sfAeBe*Nrec,Npag)⌉		
B<k'		Nkey(A)*⌈sfAeBr*NPag⌉	
A=k and B=k'	⌈Φ(sfAeBe*Nrec,Npag)⌉		
A=k and B<k'	⌈sfBr*NKey(B) ⌉ *⌈Φ(sfAeBe*Nrec,Npag)⌉	sfAeBr*Npag	
A<k and B=k'			
A<k and B<k'	⌈sfArBr*NKey(A)*Nkey(B) ⌉ *⌈Φ(sfAeBe*Nrec,Npag)⌉	sfAr*Nkey(A)*⌈sfAeBr*NPag⌉	sfArBr*Nrec

a. Consider the following graphical representation of the records that satisfy four of the conditions of the table below in a file that is sorted on (A,B). Complete that with the representation of the missing four conditions.



b. Draw a picture of an inverted-list combined index on (A,B)
c. Consider the case of 'A<k and B<k': explain how data is retrieved, in the three cases of the table, and also explain the three costs that are reported in the table
d. Give a general formula for the first and for the third columns – you may you sf(ϕ) for the selectivity factor of the condition in the current line
e. (Optional) Fill up the holes in second column

3. Consider the following log content. Assume that the DB was identical to the buffer before the beginning of this log, and consider a undo-redo protocol

(begin,T1) (W,T1,A,1,30) (begin,T2) (W,T2,B,1,20) (begin-ckp,{T1,T2}) (begin,T3) (W,T3,C,1,40) (commit,T1) (W,T2,A,30,50) (end-ckp) (begin,T4) (commit,T3) (W,T4,C,40,60)

a. Before starting this log, what was the content of A, B and C in the PS (Persistent Store)?
b. Assume there was a crash at the end of the logging period. At crash time, what was the content of A, B and C in the buffer? What can be said about the content of A, B and C in the PS?
c. At restart time, which transactions are undone? Which are redone?
d. List the operations that are redone, in the order in which are redone
e. After restart is finished, what is the content of A, B and C in the buffer?
f. Undo and Redo are executed in the buffer or on the PS?
g. After restart is finished, what is the content of A, B and C in the PS?
h. Assume now a Redo-NoUndo protocol. In this case, what can be said about the content of A, B and C in the PS at crash time, that is, after the completion of (W,T4,C,40,60)?

4. Assume that a system with no scheduler produces the following history:

$r_1[C]$, $r_2[A]$, $r_1[C]$, $w_3[B]$, $w_2[B]$, $w_1[C]$, c_1, $r_2[A]$, c_2, $w_3[B]$, c_3

a) Is this history serializable?
b) Exhibit a history that may be produced by a strict 2PL scheduler if presented with the above operations in that order, assuming that each transaction commits immediately after its last operation. In case of deadlock, assume that a transaction is aborted and is restarted later
c) Repeat (b) considering now a Wait-Die 2PL scheduler

DB2 and DBSA – April 11, 2017 – First Intermediate Test – SOLUTIONS, version 0.2

You can answer this test in English, Italian, or any mixture

1. Consider a schema R(<u>IdR</u>, A, B, …, IdS*), S(<u>IdS</u>,…), T(<u>IdT</u>, …, IdS*) and the following query

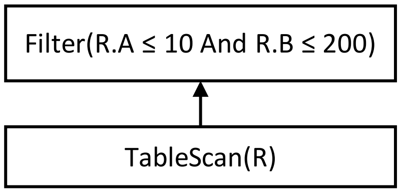
```
SELECT    *
FROM      R, T
WHERE     R.IdS = T.IdS And R.A ≤ 10 And R.B ≤ 200
```

Assume that R, S and T are stored as heap files. Primary keys are R.IdR, S.IdS and T.IdT, while R.IdS abd T.IdS are foreign keys that refers to S.
Assume that unclustered RID-sorted index are defined on all the primary and foreign keys, on R.A, on R.B, and a combined index on (R.A,R.B). Assume that every index on R has 4.000 leaves, every index on S has 10.000 leaves, every index on T has 200 leaves **(some part of this information is not needed for this exercise)**.
Assume the following table for the optimization parameters. If you need Cardenas formula Φ(n,k), approximate it with min(n,k).
Assume that every page is 4.000 bytes long, and that every attribute uses 4 bytes.

	NReg	NPag	NLeaf of Indexes	NKey	Min	Max
R	2.000.000	100.000	4.000			
S	5.000.000	500.000	10.000			
T	100.000	1.000	200			
Idx.R.A			See R	100	0	1.000
Idx.R.B			See R	10	0	1.000

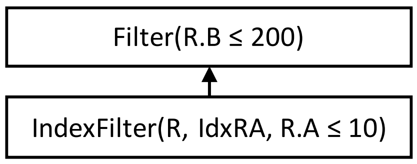
a) Compute the cost of accessing all records of R that satisfy the condition R.A ≤ 10 And R.B ≤ 200 using:
   a. No index
   b. Index on R.A only
   c. Index on R.B only
   d. Both R.A and R.B indexes

(a) No index. Access plan:



C = NPag(R) = 100.000

(b) Index on R.A only

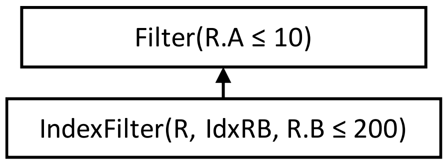


sf = 10/1000 = 1/100

C = CI + CD = sf*NLeaf + $\lceil$sf*NKey(A)$\rceil$*$\lceil\phi$(NRec/NKey(A),NPag)$\rceil$

= 4.000/100 + (100/100)*ϕ(2.000.000/100,100.000) = 40+20.000 = 20.040

(c) Index on R.B only

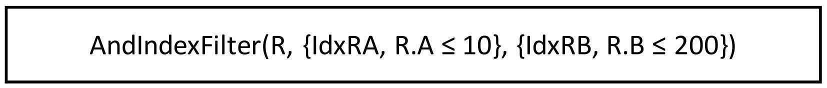


sf = 200/1000 = 1/5

C = CI + CD = sf*NLeaf + $\lceil$sf*NKey(B)$\rceil$*$\lceil\phi$(NRec/NKey(B),NPag)$\rceil$

= 4.000/5 + (10/5)*ϕ(2.000.000/10,100.000) = 800+2*100.000 = 200.800

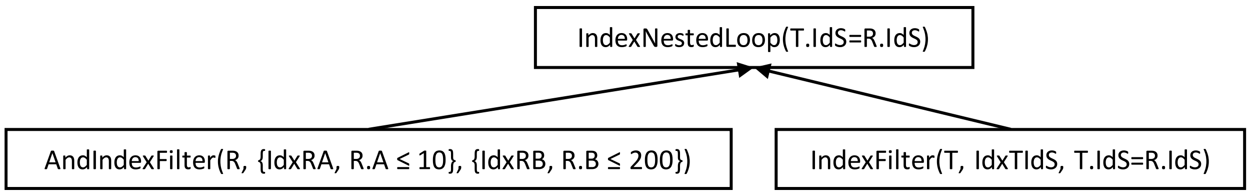
(d) Both R.A and R.B indexes



C = CI(A) + CI(B) + CD = sf(A)*NLeaf + sf(B)*NLeaf + sf(A)*sf(B)*NRec

= 40 + 800 + 1/5*1/100*2.000.000 = 4.840

b) Using the cheapest access plan from (a), compute the cost of an IndexNestedLoop plan for the complete query, where R is the outer relation


IndexNestedLoop(T.IdS=R.IdS)
AndIndexFilter(R, {IdxRA, R.A ≤ 10}, {IdxRB, R.B ≤ 200})
IndexFilter(T, IdxTIdS, T.IdS=R.IdS)

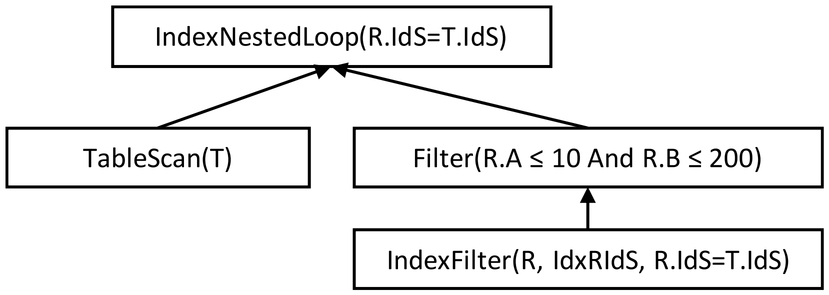
C(IndexFilter) = CI + CD = $\lceil$sf(T.IdS=R.IdS)*NLeaf(IdxTIdS)$\rceil$ + $\lceil\phi$(NRec(T)/NKey(T.IdS),NPag)$\rceil$

= $\lceil$200/5.000.000$\rceil$ + $\lceil\phi$(100.000/5.000.000, 1.000)$\rceil$ = 1+1 = 2

NReg(AndIndexFilter) = sf(AndIndexFilter)*NReg(R) = 1/5*1/100*2.000.000 = 4.000

C(IndexNestedLoop) = C(OE) + NReg(OE)*C(OI) = 4.840 + 4.000*2 = 12.840

c) Compute the cost of an IndexNestedLoop plan for the complete query, where R is the inner relation


IndexNestedLoop(R.IdS=T.IdS)
TableScan(T)
Filter(R.A ≤ 10 And R.B ≤ 200)
IndexFilter(R, IdxRIdS, R.IdS=T.IdS)

$C(\text{IndexFilter}) = CI + CD = \lceil sf(R.IdS=T.IdS)*NLeaf(IdxRIdS)\rceil + \lceil \phi(NRec(R)/NKey(R.IdS),NPag)\rceil$
$= \lceil 4.000/5.000.000\rceil + \lceil \phi(2.000.000/5.000.000, 100.000)\rceil = 1+1 = 2$
NReg(TableScan(T)) = 100.000
C(TableScan(T)) = NPag(T) = 1.000
C(IndexNestedLoop) = C(OE) + NReg(OE)*C(OI) = 1.000 + 100.000*2 = 201.000

d) How many records are in the result of the query?

NReg(R1 join R2) = NReg(OE) * NReg(OI) * sf(join condition) = 4.000*100.000*1/5.000.000 = 80

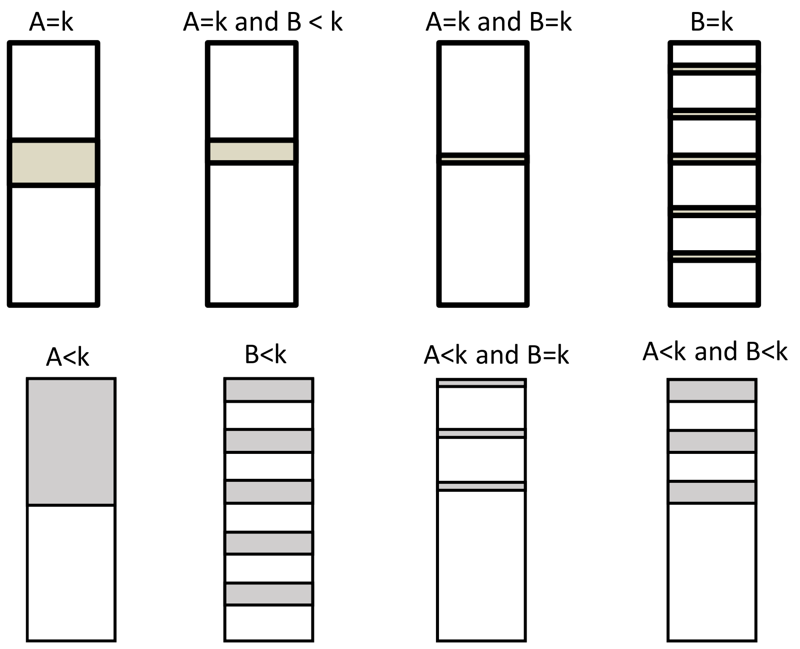
Or: NReg(R)*NReg(T)*sf(R.IdS = T.IdS And R.A ≤ 10 And R.B ≤ 200) =
= 2.000.000*100.000*1/100*1/5*1/5.000.000 = 80

2. Consider a combined index on attributes (A,B) of a table T. Assume the standard hypothesis of uniformity and independence of T.A and T.B, and also that, for every a in $\pi_A(T)$ and b in $\pi_B(T)$, the pair (a,b) belongs to $\pi_{AB}(T)$. The following table reports the cost of accessing the data if the index is unclustered with sorted RIDs, if it is clustered on (A,B), or if it is unclustered with unsorted RIDs. The table assumes that RIDS are used to fetch the record in the same order as they are found in the index, we do not merge different RID lists and we do not sort RIDs in main memory.
We use the following abbreviations for the involved selectivity factors – Ae stands for 'equality condition on A' and Ar 'range condition on A'.
sfAe = 1/Nkey(A)
sfAr = (k – min(A)) / (max(A)-min(A))
sfBe = 1/Nkey(B)
sfBr = (k' – min(B)) / (max(B)-min(B))
sfXxYy = sfXx * sfYy

Condition	Unclustered sorted RIDs	Clustered on A, B	Uncl. Unsorted RIDs
A=k	$Nkey(B) * \lceil \Phi(sfAeBe*Nrec,Npag)\rceil$	sfAe*Npag	
A<k	$\lceil sfAr*NKey(A)*Nkey(B)\rceil * \lceil \Phi(sfAeBe*Nrec,Npag)\rceil$	sfAr*Npag	

B=k’	Nkey(A) *⌈Φ(sfAeBe*Nrec,Npag)⌉		
B<k’		Nkey(A)*⌈sfAeBr*NPag⌉	
A=k and B=k’	⌈Φ(sfAeBe*Nrec,Npag)⌉		
A=k and B<k’	⌈sfBr*Nkey(B) ⌉ *⌈Φ(sfAeBe*Nrec,Npag)⌉	sfAeBr*Npag	
A<k and B=k’			
A<k and B<k’	⌈sfArBr*Nkey(A)*Nkey(B) ⌉ *⌈Φ(sfAeBe*Nrec,Npag)⌉	sfAr*Nkey(A)*⌈sfAeBr*NPag⌉	sfArBr*Nrec

a. Consider the following graphical representation of the records that satisfy four of the conditions of the table below in a file that is sorted on (A,B). Complete that with the representation of the missing four conditions.



b. Draw a picture of an inverted-list combined index on (A,B)

A	B	Rid List
1	1	[100, 200, 300]
1	2	[50,150,350,400]
1	3	[80,90]
2	1	[130,160,360,370]
2	2	[...]
2	3	[...]
3	1	[...]

c. Consider the case of 'A<k and B<k': explain how data is retrieved, in the three cases of the table, and also explain the three costs that are reported in the table

In each of the three cases, the index is scanned retrieving the RID lists associated to each couple of values (a,b) belonging to $\pi_{AB}(R)$. The number of retrieved lists can be estimated as

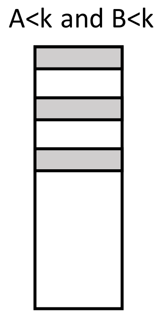
$$\lceil sfArBr*Nkey(A)*Nkey(B) \rceil.$$

The values of $\pi_{AB}(R)$ are ordered on (A,B), and the RID lists themselves are ordered in the 'clustered' and 'ordered RIDs' cases, while are not ordered in the 'unordered RIDs' case.
In the 'unclustered ordered RIDs' case, for each of the retrieved lists we follow an ordered list of RIDs whose length is on average 'sfAeBe*Nrec'. Since this list is sorted, the cost of following this list can be estimated as $\lceil \Phi(sfAeBe*Nrec,Npag) \rceil$, hence the total cost is

$$\lceil sfArBr*Nkey(A)*Nkey(B) \rceil * \lceil \Phi(sfAeBe*Nrec,Npag) \rceil$$

In the 'unclustered ordered RIDs' case, we must assume that every record is in a page that is unrelated to those that have been recently read, hence the total cost is just sfArBr*Nrec.

In the 'clustered case' we must consider how data is distributed in the file:



The picture makes it clear that the records are grouped in sfAr*Nkey(A) sequences, each containing sfAeBr records. Since the file is clustered, each seguence of sfAeBr records can be retrieved by reading $\lceil sfAeBr*NPag \rceil$ pages, hence the total cost is sfAr*Nkey(A)*$\lceil sfAeBr*NPag \rceil$.

d. Give a general formula for the first and for the third columns – you may you sf(ϕ) for the selectivity factor of the condition in the current line

Reasoning as in the previous point, we deduce that he general formula for the first column is $\lceil sf(\phi)*Nkey(A)*Nkey(B)\rceil*\lceil \Phi(sfAeBe*Nrec, Npag)\rceil$, and for the third column is $sf(\phi)*Nrec$.

e. (Optional) Fill up the holes in second column

We use the pictures on order to understand, in each case, how many clusters of consecutive records we have (say M) and what is the size of each cluster (say $\lceil sf\ldots*NPag\rceil$), and we write the corresponding formula $M*\lceil sf\ldots*NPag\rceil$.

Condition	Unclustered sorted RIDs	Clustered on A, B	Uncl. Unsorted RIDs
A=k	$Nkey(B) *\lceil \Phi(sfAeBe*Nrec,Npag)\rceil$	sfAe*Npag	
A<k	$\lceil sfAr*NKey(A)*Nkey(B)\rceil *\lceil \Phi(sfAeBe*Nrec,Npag)\rceil$	sfAr*Npag	
B=k’	$Nkey(A) *\lceil \Phi(sfAeBe*Nrec,Npag)\rceil$	**$Nkey(A)*\lceil sfAeBe*NPag\rceil$**	
B<k’	$\lceil NKey(A)*NKey(B)*sfBr\rceil *\lceil \Phi(sfAeBe*Nrec,Npag)\rceil$	$Nkey(A)*\lceil sfAeBr*NPag\rceil$	
A=k and B=k’	$\lceil \Phi(sfAeBe*Nrec,Npag)\rceil$	**sfAeBe*Npag**	
A=k and B<k’	$\lceil sfBr*NKey(B)\rceil *\lceil \Phi(sfAeBe*Nrec,Npag)\rceil$	sfAeBr*Npag	
A<k and B=k’	$\lceil sfAr*NKey(A)\rceil *\lceil \Phi(sfAeBe*Nrec,Npag)\rceil$	**$sfAr*Nkey(A)*\lceil sfAeBe*NPag\rceil$**	
A<k and B<k’	$\lceil sfArBr*NKey(A)*Nkey(B)\rceil *\lceil \Phi(sfAeBe*Nrec,Npag)\rceil$	$sfAr*Nkey(A)*\lceil sfAeBr*NPag\rceil$	sfArBr*Nrec

3. Consider the following log content. Assume that the DB was identical to the buffer before the beginning of this log, and consider a undo-redo protocol

(begin,T1) (W,T1,A,1,30) (begin,T2) (W,T2,B,1,20) (begin-ckp,{T1,T2}) (begin,T3) (W,T3,C,1,40) (commit,T1) (W,T2,A,30,50) (end-ckp) (begin,T4) (commit,T3) (W,T4,C,40,60)

a. Before starting this log, what was the content of A, B and C in the PS (Persistent Store)?

1-1-1

b. Assume there was a crash at the end of the logging period. At crash time, what was the content of A, B and C in the buffer? What can be said about the content of A, B and C in the PS?

	Buffer	PS
A	50	30-50
B	20	20
C	60	1-40-60

c. At restart time, which transactions are undone? Which are redone?

(T2,T4) are undone. (T1,T3) are redone.

d. List the operations that are redone, in the order in which are redone

(W,T3,C,1,40)

e. After restart is finished, what is the content of A, B and C in the buffer?
f. Undo and Redo are executed in the buffer or on the PS?
g. After restart is finished, what is the content of A, B and C in the PS?

Undo and Redo are executed in the buffer, and their intermediate states MAY be flushed to the PS. The initial state of the buffer is undefined, and the possible initial states of the PS are those indicated at the point (b).

The following operations are undone:
UNDO: (W,T4,C,40,60) (W,T2,A,30,50) (W,T2,B,1,20)
And then the following are redone:
REDO: (W,T3,C,1,40)

Hence this is the state of Buffer and PS, at the end of the restart:

	Buffer	PS
A	30	30-50
B	1	20-1
C	40	1-40-60

h. Assume now a Redo-NoUndo protocol. In this case, what can be said about the content of A, B and C in the PS at crash time, that is, after the completion of (W,T4,C,40,60)?

In a NoUndo protocol, the effects of an update never go to the PS before the commit, hence this would be the state of the PS at crash time. Observe that, at checkpoint time, T1 was still running, hence the value of A could not be saved. For this reason, we cannot be sure that the value of A is already 30 at crash time.

	PS
A	1-30
B	1
C	1-40

4. Assume that a system with no scheduler produces the following history:

   $r_1[C], r_2[A], r_1[C], w_3[B], w_2[B], w_1[C], c_1, r_2[A], c_2, w_3[B], c_3$

   a) Is this history serializable?

The serialization graph is:

$$\begin{tikzcd} T1 & T2 \arrow[r, bend left] & T3 \arrow[l, bend left] \end{tikzcd}$$

The graph has a loop, hence the history is not c-serializable. This was the expected answer. One may also notice that, since T3 reads no value, the effect of this history is the same as that of a serial execution T1,T2,T3 and also the same as a serial execution T2,T1,T3, and, although it is not c-serializable, the history is serializable.

   b) Exhibit a history that may be produced by a strict 2PL scheduler if presented with the above operations in that order, assuming that each transaction commits immediately after its last operation. In case of deadlock, assume that a transaction is aborted and is restarted later

We report here the history. We will not indicate the lock requests that are granted, but only those that are not granted, indicated with a *

T1	T2	T3
r1[C]		
	r2[A]	
r1[C]		
		w3[B]
	wl2[B]*	
w1[C]		
c1		
		w3[B]
		c3
	w2[B]	
	r2[A]	
	c2	

   c) Repeat (b) considering now a Wait-Die 2PL scheduler

The history of point (b) has exactly one request of a conflicting lock, where an older transaction T2 asks for a lock that is currently held by the younger transaction T3. In the Wait-Die protocol, that is a 'Wait' situation, hence T2 waits until T3 commits, hence the produced history is identical to the one produced by the standard 'Wait-Wait' version of the 2PL protocol depicted at point (b).