MSBD 6000L
DATABASE SYSTEMS

ASSIGNMENT 3
SAMPLE SOLUTIONS

QUESTION 1

A company wants to develop a database for its 30,000 employees. A page is 512 bytes, a pointer to a page is 6 bytes and a pointer to a record is 7 bytes. There are six fields in the Employee file: empNo (4 bytes), name (30 bytes), address (40 bytes), phoneNo (8 bytes), email (30 bytes) and hkid (8 bytes). The Employee file is ordered according to the primary key empNo.

60% of queries to the Employee file involve retrieval according to empNo, 30% according to hkid and 10% according to the remaining fields.

Given that you are restricted to constructing only one single-level index, would you construct a primary index on empNo or a secondary index on hkid? Your goal is to minimize the number of page I/Os for the given types of queries.

Justify your answer by showing the overall average page I/O cost of querying the Employee file using

a) no index.

b) a primary single-level index on empNo.

c) a secondary single-level index on hkid.

Show your calculations. Answers with no calculations will incur a 50% penalty.

QUESTION 1 (CONT'D)

Preliminaries

$bf_{Employee} = \lfloor 512 / 120 \rfloor = \lfloor 4.26 \rfloor = 4$

$B_{Employee} = \lceil 30000 / 4 \rceil = 7{,}500$

Overall average page I/O cost of querying the Employee file:

0.6 * cost of retrieval according to empNo +
0.3 * cost of retrieval according to hkid +
0.1 * cost of retrieval according to other fields

a) no index

Page I/O cost for retrieval according to

1 mark i. empNo: $\lceil \log_2(7{,}500) \rceil = \lceil 12.87 \rceil = 13$ (file binary search)

1 mark ii. hkid: $\lceil 7{,}500/2 \rceil = 3{,}750$ (linear search - one record qualifies)

1 mark iii. other fields: 7,500 (linear search - multiple records may qualify)

2 marks **Overall average page I/O cost:** 0.6*13 + 0.3*3,750 + 0.1*7,500 = 1,882.8

QUESTION 1 (CONT'D)

b) a primary single-level index on empNo

1 mark Number of index entries: 7,500 (one to each *page* of the Employee file)

1 mark Index entry size: 4 bytes (for empNo) + 6 bytes (for page pointer) = 10

1 mark $bf_{\text{empNo index}}$: $\lfloor 512 / 10 \rfloor = \lfloor 51.2 \rfloor = 51$

1 mark Number of index pages needed: $\lceil 7{,}500 / 51 \rceil = \lceil 147.05 \rceil = 148$

Page I/O cost for retrieval according to

2 marks i. empNo: $\lceil \log_2(148) \rceil + 1 = \lceil 7.21 \rceil + 1 = 9$ (index binary search)

ii. hkid: $\lceil 7{,}500/2 \rceil = 3{,}750$ (linear search - one record qualifies)

iii. other fields: 7,500 (linear search - multiple records may qualify)

1.5 marks **Overall average page I/O cost:** 0.6*9 + 0.3*3,750 + 0.1*7,500 = 1,880.4

QUESTION 1 (CONT'D)

c) a secondary single-level index on hkid

1 mark Number of index entries: 30,000 (one to each *record* of the Employee file)

1 mark Index entry size: 8 bytes (for hkid) + 7 bytes (for record pointer) = 15 bytes

1 mark $bf_{\text{name index}}$: $\lfloor 512 / 15 \rfloor = \lfloor 34.13 \rfloor = 34$

1 mark Number of index pages needed: $\lceil 30{,}000 / 34 \rceil = \lceil 882.35 \rceil = 883$

Page I/O cost for retrieval according to

i. empNo: $\lceil \log_2(7{,}500) \rceil = \lceil 12.87 \rceil = 13$ (file binary search)

2 marks ii. hkid: $\lceil \log_2(883) \rceil + 1 = \lceil 9.78 \rceil + 1 = 11$ (index binary search)

iii. other fields: 7,500 (linear search - multiple records may qualify)

1.5 marks **Overall average page I/O cost:** 0.6*13 + 0.3*11 + 0.1*7,500 = 761.1

Conclusion

Despite its larger size the secondary index on hkid is the best solution because it saves a lot of page I/Os compared to linear search while the primary index on empNo does not save a lot of page I/Os because the file is already ordered according to empNo.

QUESTION 2

For the Employee file of Question 1, suppose that you can construct a B^+-tree index instead of a single-level index. Assume that all nodes (both leaf and internal) of the B^+-tree have *minimum occupancy*. What would be the overall average page I/O cost of querying the Employee file using

a) a clustering B^+-tree index on the primary key empNo.

b) a non-clustering B^+-tree index on hkid.

In addition to calculating the overall average page I/O cost of querying the Employee file using each index, also show for each index your calculations for

i. the number of values and pointers in the leaf and internal nodes.

ii. the height of the B^+-tree.

QUESTION 2 (CONT'D)

Preliminaries

From Question 1, for the B+-tree index on empNo we know that in each node (leaf and internal) an empNo value occupies 4 bytes and a page pointer occupies 6 bytes. For the B+-tree index on hkid, since it is a non-clustering index, we know that in each node (leaf and internal) a hkid value occupies 8 bytes while a (record) pointer occupies 7 bytes in a leaf node, and a (page) pointer occupies 6 bytes in an internal node.

For a B+-tree, we also need to consider that for the internal nodes there is one more page pointer (6 bytes) than the number of values and, for the leaf nodes, there is also a page pointer (6 bytes) to the next leaf node. Consequently, for a page size of 512 bytes, we have at most (512-6)=506 bytes to store index (pointer, value) entries.

a) clustering B+-tree index on the primary key empNo

For both leaf and internal nodes, we can get at most $\lfloor (512-6)/10 \rfloor = 50$ values (51 pointers) on a page (i.e., n is 51).

Thus, the minimum node occupancy for the B+-tree index on empNo is:

1mark for leaf nodes: $\lceil (n-1)/2 \rceil = \lceil 50/2 \rceil = 25$ values (25 pointers to data pages)

1 mark for internal nodes: $\lceil n/2 \rceil - 1 = \lceil 51/2 \rceil - 1 = 25$ values (26 pointers to B+-tree nodes)

QUESTION 2 (CONT'D)

1 mark Therefore, the effective fanout of internal nodes is 26.

At the leaf level we need to point to 7,500 pages (sparse index). Moreover, the number of page pointers is equal to the number of values at the leaf level.

1 mark Leaf level pages needed: $\lfloor 7{,}500/25 \rfloor = \lfloor 300 \rfloor = 300$

1 mark First internal level pages needed: $\lfloor 300/26 \rfloor = \lfloor 11.53 \rfloor = 11$

1 mark Second internal (root) level pages needed: 1

Note: To ensure the minimum node occupancy requirement, the floor function should be used rather than the ceiling function; else a node may have less than the minimum number of values. The root does not need to follow the minimum node occupancy requirement.

1 mark Thus, the height of the B$^+$-tree index on empNo is 3.

The B$^+$-tree height can also be calculated as $\lceil \log_{26} 7{,}500 \rceil = \lceil 2.73 \rceil = 3$.

Page I/O cost for retrieval according to

1.5 marks
i. empNo: 3 + 1 = 4 (B$^+$-tree search)

ii. hkid: $\lceil 7{,}500/2 \rceil = 3{,}750$ (file linear search - one record qualifies)

iii. other fields: 7,500 (file linear search - multiple records may qualify)

1 mark **Overall average page I/O cost:** 0.6*4 + 0.3*3,750 + 0.1*7,500 = 1,877.4

QUESTION 2 (CONTD)

b) non-clustering B+-tree index on hkid

For leaf nodes, we can get at most $\lfloor (512-6)/15 \rfloor = 33$ values (34 pointers to data records) on a page (i.e., *n* is 34 for leaf nodes).

For internal nodes, we can get at most $\lfloor (512-6)/14 \rfloor = 36$ values (37 pointers to B+-tree nodes) on a page (i.e., *n* is 37 for internal nodes).

Thus, the minimum page occupancy for the B+-tree on hkid is:

1 mark for leaf nodes: $\lceil (n-1)/2 \rceil = \lceil (34-1)/2 \rceil = 17$ values (17 pointers to data records)

1 mark for internal nodes: $\lceil n/2 \rceil - 1 = \lceil 35/2 \rceil - 1 = 18$ values (19 pointers to B+-tree nodes)

1 mark Therefore, the effective fanout of internal nodes is 19.

At the leaf level we need to point to 30,000 records (dense index). Moreover, the number of record pointers is equal to the number of values at the leaf level.

1 mark Leaf level pages needed: $\lfloor 30{,}000/17 \rfloor = \lfloor 1{,}764.70 \rfloor = 1{,}764$

1 mark First internal level pages needed: $\lfloor 1{,}764/19 \rfloor = \lfloor 92.84 \rfloor = 92$

1 mark Second internal level pages needed: $\lfloor 92/19 \rfloor = \lfloor 4.84 \rfloor = 4$

1 mark Third internal (root) level pages needed: 1

1 mark Thus, the height of the B+-tree index on hkid is 4.

The B+-tree height can also be calculated as $\lceil \log_{19} 30{,}000 \rceil = \lceil 3.50 \rceil = 4$.

QUESTION 2 (CONTD)

Page I/O cost for retrieval according to

i. empNo: $\lceil \log_2(7{,}500) \rceil = \lceil 12.87 \rceil = 13$ (file binary search)

1.5 marks ii. hkid: $4 + 1 = 5$ (B^+-tree search)

iii. other fields: 7,500 (file linear search - multiple records may qualify)

1 mark **Overall average page I/O cost:** $0.6*13 + 0.3*5 + 0.1*7{,}500 = 759.3$

QUESTION 3

For the query $\pi_{A,B,C,D}(R \bowtie_{A=C} S)$, assume the following:

- R is 10 pages; each R tuple is 300 bytes.
- S is 100 pages; each S tuple is 500 bytes.
- The combined size of attributes A, B, C and D is 450 bytes.
- A and B are in R and have combined size of 200 bytes; C and D are in S.
- A is a key for R.
- Each S tuple joins with exactly one R tuple.
- The page size is 1024 bytes.
- The buffer size *M* is 3 pages.

Preliminaries

Number of R tuples: $\lfloor 1{,}024 / 300 \rfloor * 10 = 30$ $bf_R = \lceil 30 / 10 \rceil = 3$

Number S tuples: $\lfloor 1{,}024 / 500 \rfloor * 100 = 200$ $bf_S = \lceil 200 / 100 \rceil = 2$

We assume that C and D form a key for S. Consequently, all tuples in the join are unique and, thus, there are no duplicates in the join result and no duplicate tuples are removed by the projection.

QUESTION 3 (CONTD)

a) What are the minimum page I/O costs if the join uses the (optimized) block nested-loop join method.

i. Page I/O cost for join eliminating all unwanted attributes during the join.

Use R (smaller relation as outer relation; eliminate unwanted attributes)

2 marks

Join page I/O cost: $\lceil B_r / (M\text{-}2) \rceil * B_s + B_r = \lceil 10 / (3\text{-}2) \rceil * 100 + 10 = \underline{1{,}010}$

Since each S tuple joins with exactly one R tuple, the join produces 200 tuples. Keeping only the attributes A, B, C, and D, the size of each join tuple is 450 bytes. Thus, $\lfloor 1{,}024 / 450 \rfloor = 2$ tuples fit on a page.

2 marks

Write join result page I/O cost: $\lceil 200 / 2 \rceil = \underline{100}$

Optimized Projection Strategy Using External Sorting

When there are an odd number of runs to merge, withhold one of the largest runs for later merging since this will reduce the page I/Os at each merge pass.

ii. Page I/O cost for projection using external sorting and removing duplicate tuples on-the-fly during the merge passes.

Read 100 pages of the join result and write 100 sorted pages producing $\lceil 100 / 3 \rceil = 34$ runs (33 runs of 3 pages each and 1 run of 1 page).

4 marks

Sort page I/O cost pass 0: $100_{r} + 100_{w} = \underline{200}$

QUESTION 3 (CONTD)

Merge pass 1

Merge 34 runs, 33 runs of 3 pages and 1 run of 1 page, creating 17 runs, 16 runs of 6 pages and 1 run of 4 pages. Total runs remaining to merge 17. Read 100 pages and write 100 pages.

Merge pass 2

Merge 16 runs, 15 runs of 6 pages and 1 run of 4 pages, creating 8 runs, 7 runs of 12 pages and 1 run of 10 pages; withhold 1 run of 6 pages for merging in the next pass. Total runs remaining to merge 9. Read (15*6+4)=94 pages and write 94 pages.

Merge pass 3

Merge 8 runs, 6 runs of 12 pages, 1 run of 10 pages and 1 run of 6 pages withheld in pass 2, creating 4 runs, 3 runs of 24 pages and 1 run of 16 pages; withhold 1 run of 12 pages for merging in the next pass. Total runs remaining to merge 5. Read (6*12+10+6)=88 pages and write 88 pages.

Merge pass 4

Merge 4 runs, 2 runs of 24 pages, 1 run of 16 pages and 1 run of 12 pages withheld in pass 3, creating 2 runs, 1 run of 48 pages and 1 run of 28 pages; withhold 1 run of 24 pages for merging in the next pass. Total runs remaining to merge 3. Read (2*24+16+12)=76 pages and write 76 pages.

QUESTION 3 (CONTD)

Merge pass 5

Merge 2 runs, 1 run of 28 pages and 1 run of 24 pages withheld in pass 4, creating 1 run of 52 pages. Withhold 1 run of 48 pages for merging in the next pass. Total runs remaining to merge 2. Read (28+24)=52 pages and write 52 pages.

Merge pass 6

Merge 2 runs, 1 run of 52 pages and 1 run of 48 pages withheld in pass 5, creating 1 run of 100 pages. Total runs remaining to merge 0. Read (52+48) =100 pages and write 100 pages.

8 marks

Merge page I/O cost: $100_{r\ \text{pass 1}} + 100_{w} + 94_{r\ \text{pass 2}} + 94_{w} + 88_{r\ \text{pass 3}} + 88_{w} + 76_{r\ \text{pass 4}} + 76_{w} + 52_{r\ \text{pass 5}} + 52_{w} + 100_{r\ \text{pass 6}} = 920$

The page I/O cost of 100, to write the query result, is not included in the page I/O cost since the cost of writing the query result is never included.

iii. Total query processing page I/O cost (optimized strategy)

1,010 + 100 + 200 + 920 = 2,230

QUESTION 3 (CONTD)

Non-optimized Projection Strategy Using External Sorting

ii. Page I/O cost for projection using external sorting and removing duplicate tuples on-the-fly during the merge passes.

Read 100 pages of the join result and write 100 sorted pages producing $\lceil 100 / 3 \rceil = 34$ runs (33 runs of 3 pages each and 1 run of 1 page).

4 marks

Sort page I/O cost pass 0: $100_r + 100_w = 200$

Merge the 34 runs in $\lceil \log_2(100 / 3) \rceil = 6$ additional passes.

4 marks

Merge page I/O cost: $(100 * 2 * 6) - 100 = 1{,}100$

The page I/O cost of 100, to write the query result, is not included in the page I/O cost since the cost of writing the query result is never included.

iii. Total query processing page I/O cost

1,010 + 100 + 200 + 1,100 = 2,410

QUESTION 3 (CONT'D)

b) What are the minimum page I/O costs if the join uses merge join.

Optimized Strategy

Using merge join the projection can be done on-the-fly as the relations are joined. So, the projection cost is 0 and the only cost is for the join. Moreover, when initially sorting R and S, unwanted attributes can be removed, which reduces the number of pages written after sorting.

i. Page I/O cost to sort R (A and B are in R and have size 200 bytes)

Sort pass

Read 3 R pages into the buffer and eliminate unwanted attributes. Each R page contains 3 tuples. After eliminating unwanted attributes, these 3 tuples occupy $\lceil 9 / \lfloor 1{,}024 / 200 \rfloor \rceil = 2$ buffer pages. Therefore, read one more page into the buffer and sort 4 R pages at once creating a sorted run of 3 pages each containing 4 tuples for a total of 3 * 4 = 12 tuples. Do the same with the next 4 pages creating a second sorted run of 3 pages containing 12 tuples. Finally, read the final 2 R pages containing 6 tuples and create a third sorted run containing 6 tuples occupying $\lceil 6 / \lfloor 1{,}024 / 200 \rfloor \rceil = 2$ buffer pages. (Not all pages are full.)

Total 3 runs to merge, 2 runs of 3 pages and 1 run of 2 pages.

5 marks

Sort pass page I/O cost: $4_{r} + 3_{w} + 4_{r} + 3_{w} + 2_{r} + 2_{w} = 18$

QUESTION 3 (CONTD)

Merge pass 1

Merge 1 run of 3 pages (12 tuples) and 1 run of 2 pages (6 tuples) creating a sorted run containing18 tuples occupying $\lceil 18 / \lfloor 1{,}024 / 200 \rfloor \rceil = 4$ pages. Withhold 1 run of 3 pages for merging in the next pass.

Merge pass 2

Merge 1 run of 4 pages (18 tuples) and 1 run of 3 pages (12 tuples) creating a sorted output containing 30 tuples occupying $\lceil 30 / \lfloor 1{,}024 / 200 \rfloor \rceil = 6$ pages.

6 marks

Merge pass page I/O cost: $3 + 2 + 4 + 4 + 3 + 6 = 22$
(3, 4: read pass 1; 2, 6: write pass 1; 4, 3: read pass 2; 6: write pass 2)

(The cost to first merge the 3 pages and then the 2 pages is: pass 1 (read 3 + 3; write 5) + pass 2 (read 5 + 2; write 6) = 24).

Page I/O cost to sort R: $18 + 22 = 40$

ii. Page I/O cost to sort S (C and D are in S and have size 450-200=250 bytes)

Sort pass

Read 3 S pages into the buffer and eliminate unwanted attributes. Each S page contains 2 tuples. After eliminating unwanted attributes, these tuples occupy $\lceil 6 / \lfloor 1{,}024 / 250 \rfloor \rceil = 2$ buffer pages. Therefore, read one more page into the buffer. After removing unwanted attributes, the 8 tuples will still fit in $\lceil 8 / \lfloor 1{,}024 / 250 \rfloor \rceil = 2$ buffer pages.

QUESTION 3 (CONT'D)

Therefore, read one more page into the buffer. After removing unwanted attributes, the 10 tuples will fit in $\lceil 10 / \lfloor 1{,}024 / 250 \rfloor \rceil = 3$ pages. Sort these 10 tuples to create a sorted run of 3 pages. Do the same for each of the remaining 95 pages creating 20 sorted runs of 3 pages each containing 10 tuples. (Not all pages are full.)

5 marks

Sort pass page I/O cost: $100_{r} + 60_{w} = 160$

Merge pass 1

Merge 20 runs of 3 pages each creating 10 sorted runs each containing 20 tuples occupying $\lceil 20 / \lfloor 1{,}024 / 250 \rfloor \rceil = 5$ pages. Read 60, write 50 pages. Runs remaining to merge 10: 10 runs of 5 pages each.

Merge pass 2

Merge 10 runs of 5 pages each creating 5 sorted runs each containing 40 tuples occupying $\lceil 40 / \lfloor 1{,}024 / 250 \rfloor \rceil = 10$ pages. Read 50, write 50 pages. Runs remaining to merge 5: 5 runs of 10 pages each.

Merge pass 3

Merge 4 runs of 10 pages each creating 2 sorted runs each containing 80 tuples occupying $\lceil 80 / \lfloor 1{,}024 / 250 \rfloor \rceil = 20$ pages. Read 40, write 40 pages. Withhold 1 run of 10 pages for merging in the next pass. Runs remaining to merge 3: 2 runs of 20 pages and 1 run of 10 pages.

QUESTION 3 (CONTD)

Merge pass 4

Merge 1 run of 20 pages from merge pass 3 and one run of 10 pages from merge pass 2 creating 1 run of 30 pages containing 120 tuples occupying $\lceil 120 / \lfloor 1{,}024 / 250 \rfloor \rceil = 30$ pages. Read 30 pages and write 30 pages. Withhold 1 run of 20 pages for merging in the next pass. Runs remaining to merge 2: 1 run of 30 pages and 1 run of 20 pages.

Merge pass 5

Merge 1 run of 30 pages from merge pass 4 and 1 run of 20 pages from merge pass 3 to create the final sorted run containing 200 tuples occupying $\lceil 200 / \lfloor 1{,}024 / 250 \rfloor \rceil = 50$ pages. Read 50 pages and write 50 pages.

6 marks

Merge pass page I/O cost: $\underset{r\ \text{pass 1}}{60} + \underset{w}{50} + \underset{r\ \text{pass 2}}{50} + \underset{w}{50} + \underset{r\ \text{pass 3}}{40} + \underset{w}{40} + \underset{r\ \text{pass 4}}{30} + \underset{w}{30} + \underset{r\ \text{pass 5}}{50} + \underset{w}{50}$

$= 450$

Page I/O cost to sort S: 160 + 450 = 610

iii. Page I/O cost to join R and S

2 marks

Join page I/O cost: $B_r + B_s = 6 + 50 = 56$

iv. Total query processing page I/O cost (optimized strategy)

40 + 610 + 56 = 706

QUESTION 3 (CONT'D)

Non-optimized strategy

Using merge join the projection can be done on-the-fly as the relations are joined. So, the projection cost is 0 and the only cost is for the join. Moreover, when initially sorting R and S, unwanted attributes can be removed, which reduces the number of pages written after sorting.

i. Page I/O cost to sort R (A and B are in R and have size 200 bytes)

Sort pass

Read 3 R pages into the buffer and eliminate unwanted attributes. Each R page contains 3 tuples. After eliminating unwanted attributes, these tuples occupy $\lceil 9 / \lfloor 1{,}024 / 200 \rfloor \rceil = 2$ buffer pages. Therefore, read 3 pages and write 2 pages. We do the same with the next 6 pages.

Finally, read and write the final R page. (Not all pages are full.)

Total 4 runs to merge, 3 runs of 2 pages and 1 run of 1 page.

3 marks

Sort pass page I/O cost: $3_r + 2_w + 3_r + 2_w + 3_r + 2_w + 1_r + 1_w = 17$

Merge pass 1

Merge 2 runs of 2 pages each creating 1 sorted run containing 18 tuples occupying $\lceil 18 / \lfloor 1{,}024 / 200 \rfloor \rceil = 4$ pages. Merge 1 run of 2 pages and 1 run of 1 page creating 1 sorted run containing 12 tuples occupying $\lceil 12 / \lfloor 1{,}024 / 200 \rfloor \rceil = 3$ pages. Read 7 pages, write 7 pages. Runs remaining to merge is 2.

QUESTION 3 (CONTD)

Merge pass 2

Merge 1 run of 4 pages with 1 run of 3 pages creating a final sorted output containing 30 tuples occupying $\lceil 30 / \lfloor 1{,}024 / 200 \rfloor \rceil = 6$ pages. Read 7 pages, write 6 pages.

4 marks

Merge pass page I/O cost: $7_{\text{read pass 1}} + 7_{\text{write}} + 7_{\text{read pass 2}} + 6_{\text{write}} = 27$

Page I/O cost to sort R: 17 + 27 = 44

ii. Page I/O cost to sort S (C and D are in S and have size 450-200=250 bytes)

Sort pass

Read 3 S pages into the buffer and eliminate unwanted attributes. Each S page contains 2 tuples. After eliminating unwanted attributes, these tuples occupy $\lceil 6 / \lfloor 1{,}024 / 250 \rfloor \rceil = 2$ buffer pages. Therefore, read 3 pages and write 2 pages. We do the same with the next 96 pages.

Finally, read and write the final S page. (Not all pages are full.)

Total 34 runs to merge, 33 runs of 2 pages containing 6 tuples each and 1 run of 1 page containing 2 tuples.

3 marks

Sort pass page I/O cost: $100_{r} + (33*2_{w}+1) = 167$

QUESTION 3 (CONTD)

Merge pass 1

Merge 32 runs of 2 pages each creating 16 sorted runs each containing 12 tuples occupying $\lceil 12 / \lfloor 1{,}024 / 250 \rfloor \rceil = 3$ pages. Merge 1 run of 2 pages (6 tuples) and 1 run of 1 page (2 tuples) creating 1 sorted run containing 8 tuples occupying $\lceil 8 / \lfloor 1{,}024 / 250 \rfloor \rceil = 2$ pages. Read (33*2+1) = 67 pages and write (16*3+2) = 50 pages. Runs remaining to merge is 17, 16 runs of 3 pages and 1 run of 2 pages.

Merge pass 2

Merge 14 runs of 3 pages (12 tuples) each creating 7 runs each containing 24 tuples occupying $\lceil 24 / \lfloor 1{,}024 / 250 \rfloor \rceil = 6$ pages. Merge 1 run of 3 pages (12 tuples) and 1 run of 2 pages (8 tuples) creating 1 run containing 20 tuples occupying $\lceil 20 / \lfloor 1{,}024 / 250 \rfloor \rceil = 5$ pages. Withhold 1 run of 3 pages for merging in the next pass. Read (14*3+3+2) = 47 and write (7*6+5) = 47 pages. Runs remaining to merge 9: 7 runs of 6 pages, 1 run of 5 pages and 1 run of 3 pages.

Alternate strategy: Withhold 1 run of 2 pages for later merging. Merge 16 runs of 3 pages each creating 8 runs of 6 pages each. Read (16*3) = 48 pages, write (8*6) = 48 pages.

QUESTION 3 (CONTD)

Merge pass 3

Merge 6 runs of 6 pages (24 tuples) each creating 3 runs containing 48 tuples occupying $\lceil 48 / \lfloor 1{,}024 / 250 \rfloor \rceil = 12$ pages. Merge 1 run of 5 pages (20 tuples) and 1 run of 3 pages (12 tuples) creating 1 run containing 32 tuples occupying $\lceil 32 / \lfloor 1{,}024 / 250 \rfloor \rceil = 8$ pages. Withhold 1 run of 6 pages for merging in the next pass. Read (6*6+5+3) = 44 pages and write (3*12+8) = 44 pages. Runs remaining to merge 5: 3 runs of 12 pages, 1 run of 8 pages and 1 run of 6 pages.

Alternate strategy: Continue withholding 1 run of 2 pages for later merging. Merge 8 runs of 6 pages each creating 4 runs of 12 pages each. Read (8*6) = 48 pages, write (4*12) = 48 pages.

Merge pass 4

Merge 2 runs of 12 pages (48 tuples) each creating 1 run containing 96 tuples occupying $\lceil 96 / \lfloor 1{,}024 / 250 \rfloor \rceil = 24$ pages. Merge 1 run of 8 pages (32 tuples) and 1 run of 6 pages (24 tuples) creating 1 run containing 56 tuples occupying $\lceil 56 / \lfloor 1{,}024 / 250 \rfloor \rceil = 14$ pages. Withhold 1 run of 12 pages for merging in the next pass. Read (12*2+8+6) = 38 pages and write (24+14) = 38 pages. Runs remaining to merge 3: 1 runs of 24 pages, 1 run of 14 pages and 1 run of 12 pages.

Alternate strategy: Continue withholding 1 run of 2 pages for later merging. Merge 4 runs of 12 pages each creating 2 runs of 24 pages each. Read (4*12) = 48 pages, write (2*24) = 48 pages.

QUESTION 3 (CONTD)

Merge pass 5

Merge 1 run of 14 pages (56 tuples) and 1 run of 12 pages (48 tuples) creating 1 run containing 104 tuples occupying $\lceil 104 / \lfloor 1{,}024 / 250 \rfloor \rceil = 26$ pages. Withhold 1 run of 24 pages for merging in the next pass. Read (14+12) = 26 pages and write 26 pages. Runs remaining to merge 2: 1 run of 26 pages and 1 run of 24 pages.

Alternate strategy: Continue withholding 1 run of 2 pages for later merging. Merge 2 runs of 24 pages each creating 1 run of 48 pages. Read (2*24) = 48 pages, write (1*48) = 48 pages.

Merge pass 6

Merge 1 run of 26 pages and 1 run of 24 pages creating the final sorted run containing 200 tuples occupying $\lceil 200 / \lfloor 1{,}024 / 250 \rfloor \rceil = 50$ pages. Read (26+24) = 50 pages, write 50 pages.

Alternate strategy: Merge 1 run of 48 pages and 1 run of 2 pages creating 1 run of 50 pages. Read (48+2) = 50 pages, write (1*50) = 50 pages.

4 marks

Merge pass page I/O cost: $67_{r\,\text{pass 1}} + 50_{w} + 47_{r\,\text{pass 2}} + 47_{w} + 44_{r\,\text{pass 3}} + 44_{w} + 38_{r\,\text{pass 4}} + 38_{w} + 26_{r\,\text{pass 5}} + 26_{w} + 50_{r\,\text{pass 6}} + 50_{w} = 527$

Alternate strategy merge page I/O cost: 549

Page I/O cost to sort S: 167 + 526 = 694

QUESTION 3 (CONT'D)

iii. Page I/O cost to join R and S

2 marks

Join page I/O cost: $B_r + B_s = 6 + 50 = \underline{56}$

iv. Total query processing page I/O cost (non-optimized strategy)

$44 + 694 + 56 = \underline{794}$

QUESTION 4

Consider the following two transactions:

T_1
read(A)
read(B)
if A = 0 then B := B + 1
write(B)

T_2
read(B)
read(A)
if B = 0 then A := A + 1
write(A)

Let the consistency requirement be A=0 ∨ B=0, with A=B=0 the initial values.

QUESTION 4 (CONT'D)

a) Show that every serial execution involving these two transactions preserves the consistency of the database

There are two possible serial executions: T_1T_2 and T_2T_1.

6 marks

Case 1: T_1T_2

	A	B
initially	0	0
after T_1	0	1
after T_2	0	1

Consistency met:
A = 0 v B = 0 ≡ T v F = T

Case 2: T_2T_1

	A	B
initially	0	0
after T_2	1	0
after T_1	1	0

Consistency met:
A = 0 v B = 0 ≡ F v T = T

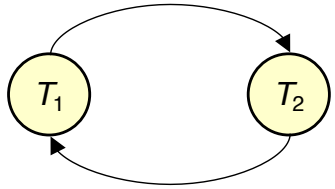
QUESTION 4 (CONT'D)

b) Give a concurrent execution of T_1 and T_2 that produces a non-serializable schedule and show why the schedule is non-serializable.

Any interleaving of T_1 and T_2 results in a non serializable schedule. For example, this schedule produces a cycle in the precedence graph.

6 marks

T_1	T_2
read(A)	
	read(B)
	read(A)
read(B)	
if A=0 then B:=B+1	
	if B=0 then A:=A+1
	write(A)
write(B)	



QUESTION 4 (CONT'D)

c) Is there a concurrent execution of T_1 and T_2 that produces a serializable schedule? Explain your answer.

There is no concurrent execution resulting in a serializable schedule.

From a) we know that a serializable schedule results in A=0 or B=0.

Suppose we start with T_1 read(A). Then when the schedule ends, no matter when we run the operations of T_2, B=1.

Now, suppose we start executing T_2 prior to the completion of T_1. Then T_2 read(B) will give B a value of 0. So, when T_2 completes, A=1.

Thus, $B=1 \wedge A=1 \rightarrow \neg(A=0 \vee B=0)$.

8 marks Similarly, for starting with T_2 read(B).