Database Management Systems
Written Exam

09.02.2010

First name		Last name	
Student number		Signature	

Instructions for Students

- Write your name, student number, and signature on the exam sheet and on every solution sheet you hand in.
- This is a closed book exam: the only resources allowed are blank paper, pens, and your head. Use a pen, not a pencil.
- Write neatly and clearly. The clarity of your explanations affects your grade.
- You have 2 hours for the exam.

Good luck!

Reserved for the Teacher

Exercise	Max. points	Points
1	20	
2	10	
3	20	
4	20	
5	10	
6	12	
7	8	
Total	100	

Exercise 1 (20 pt) Answer the following questions:

a. What is minimized by the buffer manager?

b. What is important for a sequential file organization to be efficient for a (sequential) scan of the file?

c. What are the two steps to evaluate $\sigma_{A \geq v}(r)$ if a primary index on A exists?

d. What index is preferrable for range queries: primary index or secondary index?

e. What is the cost of the nested loop join in the best case?

f. What is an alternative to the nested loop/block nested loop join for the evaluation of $r \bowtie_{\theta_1 \vee \dots \vee \theta_n} s$?

g. How can the materialized view $v = r \bowtie s$ be updated incrementally when i_r tuples are inserted into r?

h. What are two pitfalls of lock-based protocols?

i. The wait-for graph is used to detect conflict serializability or deadlocks?

j. For log-based recovery with deferred DB modifications: What actions are performed if a transaction is rolled back?

Exercise 2 (10 pt) Consider the following relation:

Branch	Customer	Account
Downtown	Smith	237
Downtown	Jones	222
Mianus	Smith	250
Downtown	Turner	300
Mianus	Jackson	200
Mianus	Hayes	382
Downtown	Williams	180
Brighton	Jackson	290

Suppose that a branch with all its customer and accounts shall be stored in a variable-length record. Show the file organisation for the following methods:

a. Byte string representation

b. Fixed-length representation with pointer (using anchor and overflow block)

c. Slotted page structure

Exercise 3 (20 pt) Consider the following relation r:

	Course	StudID	Grade
r_0	DMS	2100	18
r_1	ITP	2157	18
r_2	ITP	2230	30
r_3	DMS	2177	24
r_4	OS	2340	28
r_5	ITP	2200	23
r_6	DMS	2157	28
r_7	DB	2300	30
r_8	DMS	2263	25
r_9	DB	2299	28

Show the following index structures and file organisations:

a. An index-sequential file organisation with a primary sparse index on *StudID*. For a search-key k, an index entry is created if $k \mod 100 = 0$.

b. Independent from a), a primary B^+-tree index on *Grade*. Assume $n = 3$ for the B^+-tree.

c. A static hash file organisation on *Grade* with hash function $h(n) = n \mod 4$ and buckets that hold at most 2 tuples.

d. A bitmap index on *Course*.

Exercise 4 (20 pt) Assume a relation `prod(pid, category, price, ...)` with 600.000 tuples, where each tuple is 100 Bytes. The product ID `pid` is a key and is equally distributed between 1 and 3.000.000. The block size is 2.000 Bytes.

a. Consider a B^+-tree index on the product ID `pid`, where the `pid` requires 4 Bytes and a pointer requires 6 Bytes; a tree node occupies an entire block. Determine the minimal and maximal number of blocks used for the tree.

b. Consider the B^+-tree from a) with the minimal number of blocks and assume that it is a primary index. Describe the evaluation of the following queries and determine the number of IOs (data blocks + index blocks):

```
Q1:SELECT * FROM prod WHERE pid BETWEEN 10000 AND 20000
Q2:SELECT CNT(*) FROM prod WHERE pid BETWEEN 10000 AND 20000
```

c. Repeat b) but assume the B^+-tree to be a secondary index.

Exercise 5 (10 pt) Consider a relation $Grades(Stud, Grade)$ that contains the following tuples: $(Jan, 25)$, $(John, 25)$, $(Ann, 25)$, $(Sue, 18)$, $(Pete, 30)$, $(Sarah, 20)$, $(Ron, 27)$, $(Julia, 22)$, $(Bob, 18)$, $(Luk, 23)$, $(Tim, 25)$. Further, assume that only one tuple fits in a block, and the memory holds at most 3 blocks.

a. Show the runs created on each pass of the sort-merge algorithm, when applied to sort the *Grades* relation.

b. What is the total number of block transfers ? Explain your answer.

Exercise 6 (12 pt) Given is the following schedule for transactions T_1 and T_2:

T_1	T_2
read(A)	
write(A)	
	read(A)
	read(B)
read(B)	
write(B)	

Answer the following questions and explain your answer:

a. Is the schedule conflict serializable?

b. Is the schedule view serializable?

c. Is the schedule recoverable if both transactions commit immediately after the last operation?

d. Is the schedule cascadeless?

Exercise 7 (8 pt) Given is the following schedule over transactions T_1, T_2:

T_1	T_2
read(A)	
	write(B)
read(B)	

Answer the following questions and explain your answer:

a. Is the schedule possible under the two-phase locking protocol? If yes, add the lock and unlock instructions.

b. Is the schedule possible under the timestamp protocol? (assume $TS(T_1) < TS(T_2)$)

Solution 1

a. The number of disk accesses

b. The records must physically be stored in search-key order (or close to)

c. Step 1: Use index to find first tuple with $A \geq v$
Step 2: Scan relation sequentially from there.

d. Primary index.

e. $C = b_r + b_s$ (where b_r and b_s is the number of blocks of the two relations)

f. Compute the union of the individual joins $r \bowtie_{\theta_i} s$, i.e., $r \bowtie_{\theta_1} s \cup \cdots \cup r \bowtie_{\theta_n} s$

g. $v^{new} = v^{old} \cup (i_r \bowtie s)$

h. Too early unlocking can lead to non-serializable schedules. Too late unlocking can lead to deadlocks.

i. Deadlocks

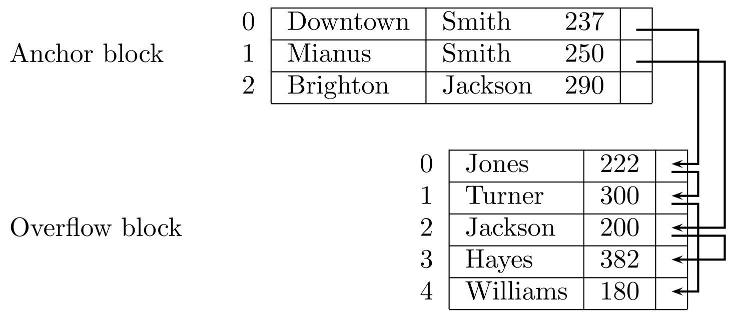
j. No actions need to be done.

Solution 2

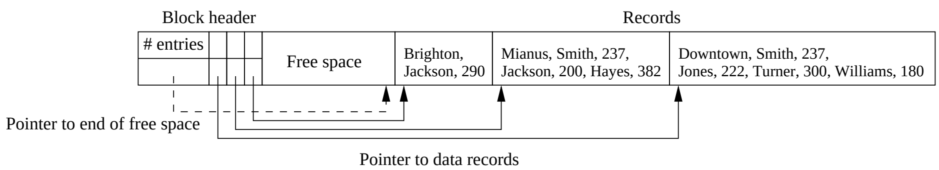
a. Byte string representation

0	Downtown	Smith	237	Jones	222	Turner	300	Williams	180	$\perp$
1	Mianus	Smith	250	Jackson	200	Hayes	382	$\perp$		
2	Brighton	Jackson	290	$\perp$						

b. Fixed-length representation with pointer (using anchor and overflow block):

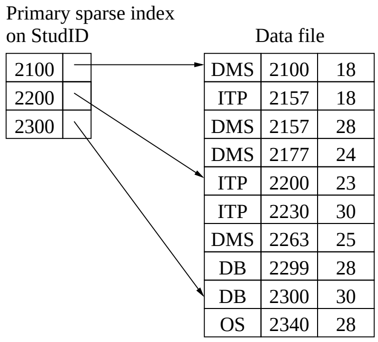


c. Slotted page structure:

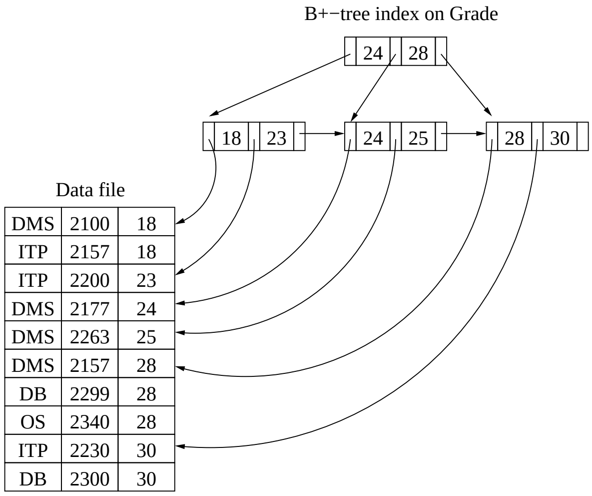


Solution 3

a. Index-sequential file organisation with primary index (and data file)



b. Primary B^+-tree index with data file



c. Hash file organisation
$h(18) = 2$, $h(23) = 3$, $h(24) = 0$, $h(25) = 1$, $h(28) = 0$, $h(30) = 2$

			Overflow buckets
Bucket 0:	(DMS, 2177, 24) (OS,2340,28)	⟶	(DMS,2157,28) (DB,2299,28)
Bucket 1:	(DMS,2263,25)		
Bucket 2:	(DMS,2100,18) (ITP,2157,18)	⟶	(ITP,2230,30) (DB,2300,30)
Bucket 3:	(ITP,2200,23)		

d. Bitmap index for *Course*:

DMS: [1 0 0 1 0 0 1 0 1 0]
ITP: [0 1 1 0 0 1 0 0 0 0]
OS: [0 0 0 0 1 0 0 0 0 0]
DB: [0 0 0 0 0 0 0 1 0 1]

Solution 4

a. Minimal number of index blocks when tree nodes are completely filled
$\lfloor 2.000/(4+6) \rfloor = 200$ index entries/block
- leaf nodes: $\lceil 600.000/200 \rceil = 3.000$ blocks
- level $n-1$: $\lceil 3.000/200 \rceil = 15$ blocks
- level $n-2$: $\lceil 15/200 \rceil = 1$ block
$\Rightarrow$ at least 3.016 index blocks are required

Maximal number of index blocks when tree nodes are only half full
$\lfloor 1.000/(4+6) \rfloor = 100$ index entries/block
- leaf nodes: $\lceil 600.000/100 \rceil = 6,000$ blocks
- level $n-1$: $\lceil 6.000/100 \rceil = 60$ blocks
- level $n-2$: $\lceil 60/100 \rceil = 1$ block
$\Rightarrow$ at most 6.061 index blocks are required

b. Average distance between `pid` values: $3.000.000/600.000 = 5$
$\Rightarrow Q1$ and $Q2$ retrieve $(20.000 - 10.000)/5 = 2.000$ tuples on average.

Data tuples/block: $\lceil 2.000/100 \rceil = 20$

Q1: Traverse the tree once to get the block of the first matching tuple, then scan the data blocks for the other tuples.
Block IOs:
– 3 index nodes + $\lceil 2.000/20 \rceil = 100$ data blocks $\Rightarrow$ 103 total IOs

Q2: Traverse the tree once to get the leaf node with the first matching search-key, then scan the leave nodes for the other matching keys. The data tuples are not needed to evaluate this query!
Block IOs:
– 3 index nodes + $\lceil 2.000/200 \rceil = 10$ index leaf nodes $\Rightarrow$ 13 IOs

c. Q1: Traverse the tree once to get the leaf node with the first matching search-key, then follow the leaf nodes for the other matching search-keys. For each matching search-key, follow the data pointer and retrieve the tuple.
Block IOs:
– $3 + 10 = 13$ index nodes (as in $Q2$ above);
– 2.000 data blocks (in the worst case, when each tuple is on separate block);
$\Rightarrow$ 2.013 IOs in total

Q2: The same as in b.)

Solution 5

a. In the following we use only the names to refer to the tuples (note that the relation shall be sorted on the *Stud* attribute).

   Step 1: Create 4 sorted runs with 3 tuples each:
   (Ann, Jan, John), (Pete, Sara, Sue), (Bob, Julia, Ron), (Luk, Tim)

   Step 2: Merge pass that merges two runs into one run. Thus the number of runs decreases by the factor of 2:
   (Ann, Jan, John, Pete, Sara, Sue), (Bob, Julia, Luk, Ron, Tim)

   Step 3: The runs after the second merge pass are:
   (Ann, Bob, Jan, John, Julia, Luk, Pete, Ron, Sara, Sue, Tim)

b. Step 1: 11 x 2 = 22 block transfers (read and write)
   Step 2: 11 x 2 = 22 block transfers (read and write)
   Step 3: 11 x 1 = 11 block transfers (only read)
   $\Rightarrow$ 55 block transfers

Solution 6

a. No. Transforming the schedule to each of the two serial schedules, $\langle T_1, T_2 \rangle$ and $\langle T_2, T_1 \rangle$, involves conflicting swaps, i.e., $write(A) - read(A)$ an $write(B) - read(B)$, respectively. (Alternatively, one could draw the conflict graph and show that it contains a cycle)

b. No.
   In the serial schedule $S' = \langle T_1, T_2 \rangle$, the following rule is violated for data item B: For each data item Q, if transaction T_i reads the inital value of Q in schedule S, then T_i must in scheule S' also read the initial value of Q.
   In the serial schedule $S' = \langle T_2, T_1 \rangle$, the following rule is violated for data item A: For each data item Q, if transaction T_i reads data item Q in schedule S and the value was produced by T_j, then T_i must in scheule S' also read the value of Q that was produced by T_j.

c. No. T_1 might fail after T_2 already committed (and a rollback is required).

d. No. If T_1 fails after T_2 executed the last operation (not yet committed), it causes T_2 to roll back.

Solution 7

a. Yes.

	T_1	T_2
1	lock-S(A)	
2	read(A)	
3		lock-X(B)
4		write(B)
5		unlock(B)
6	lock-S(B)	
7	read(B)	
8	unlock(A)	
9	unlock(B)	

b. No.
   We assume $TS(T_1) = 1$ and $TS(T_2) = 2$.
   Then at step 4 the transaction T_2 sets the W-timestamp of B to 2.
   Then at step 7 the $read(B)$ of T_1 is rejected, since the timestamp of T_1 is smaller than 2.