Database Management Systems
Written Examination

24.06.2011

First name		Last name	
Student number		Signature	

Instructions for Students

- Write your name, student number, and signature on the exam sheet.
- Write your name and student number 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 120 minutes for the exam.

Reserved for the Teacher

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

Exercise 1 (20 pt) Answer the following questions:

a. What is minimized by the buffer manager? access?

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

c. What are the two properties of an ideal hash function?

d. Assuming M memory blocks, what is the best way to use these blocks in the block nested loop join?

e. The non-leave nodes of a B^{+}-tree form a dense index on the leave nodes. Is that correct?

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

g. What is a cascading rollback?

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

i. Does the two-phase-locking protocol ensure freedom from deadlocks?

j. For log-based recovery with immediate DB modifications: What actions are performed after a crash?

Exercise 2 (10 pt) Given is the following table with project assignments:

Employee	*Project*	*Hours*
Jan	P1	800
Ann	P1	250
Jan	P2	400
Jan	P3	500
Jan	P4	900
Joe	P3	350

Consider the storage of the table using a variable-length record for each employee.

a. Show the file organization using fixed-length representation with reserved space.

b. Show the file organization using fixed-length representation with pointer (without overflow block).

c. Assume the following memory requirements for the attributes: *Employee* = 20 Bytes, *Project* = 10 Bytes, *Hours* = 4 Bytes. A pointer requires 4 Bytes. Calculate the disk space that is required by the solutions in a) and b).

Exercise 3 (20 pt) Consider the following relation, which stores information about project assignments:

	Emp	Proj	Salary	Period
r_0	Joe	A	6	[1,6]
r_1	Joe	B	14	[1,12]
r_2	Ron	B	30	[4,24]
r_3	Ann	A	15	[7,18]
r_4	Jim	A	4	[7,12]
r_5	Ann	D	3	[10,12]
r_6	Lea	F	13	[12,24]
r_7	Ann	F	13	[13,24]
r_8	Jim	C	8	[13,18]
r_9	Max	B	7	[18,24]

Show the following index structures and file organisations (assume that the tuples are inserted in the order $r_0, r_1, \ldots$):

a. A primary B^+-tree index with $n = 3$ on *Proj* together with the data file. Show the structure after $r_0, \ldots, r_4$ have been inserted and after all tuples have been inserted.

b. A secondary index on *Emp* together with the data file, using index-sequential file organisation.

c. A static hash file organisation on *Salary* using hash function $h(n) = n \mod 4$. Each bucket can hold at most 2 tuples.

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 (10 pt) Assume two relations $r(A, B)$ and $s(B, C)$. Transform the following relational algebra expression into more efficient ones and motivate your choice:

a. $\sigma_{(A=1 \vee A=3) \wedge B<C}(r \bowtie s)$

b. $\pi_B(\sigma_{C>100}(r \bowtie s))$

Exercise 7 (10 pt) Consider the following schedule:

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

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

b. Assume the following order on the data items: $B \rightarrow A$. Is the schedule possible under the tree protocol? If yes, add lock and unlock instructions.

c. Suppose that none of the two transactions committed yet (e.g., additional operations might follow). Is the schedule cascadeless? Explain your answer. If no, where should a commit be placed in order to make it cascadeless?

Solution 1

a. The number of disk accesses

b. Primary index

c. Uniform and random

d. $M-2$ blocks for the outer relation, 1 block for the inner relation, 1 block for the output

e. No

f. $C = b_r + b_s$, where b_r and b_s is the number of blocks of the two relations

g. A single transaction leads to a series of transaction rollbacks

h. Deadlocks

i. No

j. Transaction T needs to be undone if the log contains a $\langle T, start \rangle$ record but not a $\langle T, commit \rangle$ record; T needs to be redone if the log contains both a $\langle T, start \rangle$ record and a $\langle T, commit \rangle$ record.

Solution 2

a. File organization with reserved space method for variable-length records:

0	Jan	P1	800	P2	400	P3	500	P4	900
1	Ann	P1	250	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
2	Joe	P3	350	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

b. File organization with pointer method for variable-length records:

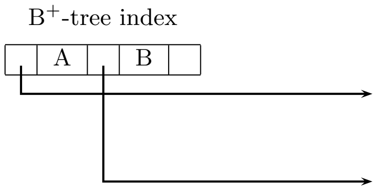
0	Jan	P1	800	
1	Ann	P1	250	
2		P2	400	
3		P3	500	
4		P4	900	
5	Joe	P3	350	

c. Memory requirements:
for a) 1 record = (20 + 4 x 14) = 76 Bytes, total = 3 x 76 = 228 Bytes
for b) 1 record = (20 + 14 + 4) = 38 Bytes, total = 6 x 38 = 228 Bytes

Solution 3

a. Primary B^+-tree index on *Proj*

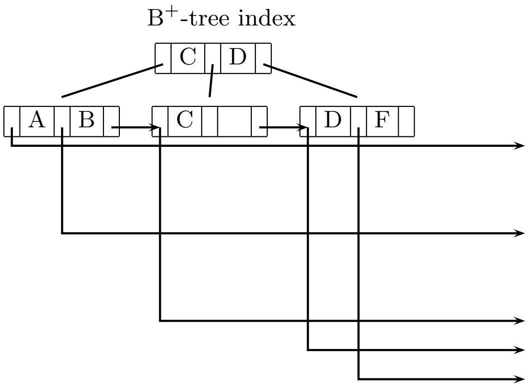
- after reading $r_0, \dots, r_4$:



Data file

Emp	Proj	Salary	Period
Joe	A	6	[1,6]
Ann	A	15	[7,18]
Jim	A	4	[7,12]
Joe	B	14	[1,12]
Ron	B	30	[4,24]

- after reading all tuples

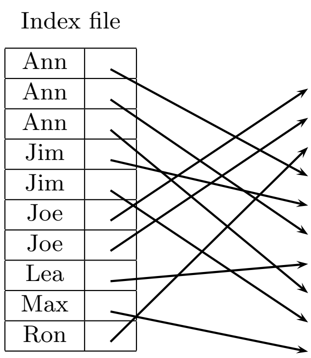


Data file

Emp	Proj	Salary	Period
Joe	A	6	[1,6]
Ann	A	15	[7,18]
Jim	A	4	[7,12]
Joe	B	14	[1,12]
Ron	B	30	[4,24]
Max	B	7	[18,24]
Jim	C	8	[13,18]
Ann	D	3	[10,12]
Lea	F	13	[12,24]
Ann	F	13	[13,24]

NOTE: For the data file we assume that tuples with identical *Proj* values are stored in the same order as in the original file, i.e., a new tuple is always stored at the end of a sequence of tuples with the same value.

b. Secondary index on *Emp* (with duplicate index entries)



Data file

Emp	Proj	Salary	Period
Joe	A	6	[1,6]
Joe	B	14	[1,12]
Ron	B	30	[4,24]
Ann	A	15	[7,18]
Jim	A	4	[7,12]
Ann	D	3	[10,12]
Lea	F	13	[12,24]
Ann	F	13	[13,24]
Jim	C	8	[13,18]
Max	B	7	[18,24]

c. Hash file organization

$h(4) = 0$; $h(8) = 0$; $h(13) = 1$; $h(6) = 2$; $h(14) = 2$; $h(30) = 2$; $h(3) = 3$; $h(7) = 3$; $h(15) = 3$;

	Bucket		Overflow buckets
Bucket 0:	(Jim, A, 4, [7,12]) (Jim, C, 8, [13,18])		
Bucket 1:	(Lea, F, 13, [12,24]) (Ann, F, 13, [13,24])		
Bucket 2:	(Joe, A, 6, [1,6]) (Joe, B, 14, [1,12])	⟶	(Ron, B, 30, [4,24])
Bucket 3:	(Ann, A, 15, [7,18]) (Ann, D, 3, [10,12])	⟶	(Max, B, 7, [18,24])

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. $\sigma_{(A=1 \vee A=3) \wedge B<C}(r \bowtie s)$:
   – Push condition $A = 1 \vee A = 3$ down to r
   – Push condition $B < C$ down to s
   – Both transformations reduce the arguments of the join
   – Thus, we get $\sigma_{A=1 \vee A=3}(r) \bowtie \sigma_{B<C}(s)$
   – An additional optimization might be to split the OR condition and replace it by a union: $(\sigma_{A=1}(r) \cup \sigma_{A=3}(r)) \bowtie \sigma_{B<C}(s)$

b. $\pi_B(\sigma_{C>100}(r \bowtie s))$:
   – Push down $\sigma_{C>100}$ to s followed by a projection to B
   – Project r to attribute B
   – Both operations reduce the argument relations of the join: the selection reduces the number of tuples, the projection reduces the size (in terms of blocks)
   – Thus, we get $\pi_B(r) \bowtie \pi_B(\sigma_{C>100}(s))$
   – Note that the join is needed, since there might be B-values in s that are not in r

Solution 7

a. Yes, it is possible under the two-phase locking protocol.

	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, not possible under the tree protocol with the order $B \rightarrow A$ (since in T_1 the first lock is on A, which does not allow to lock B later on).

c. No, the schedule is not cascadeless. If T_2 aborts, T_1 must be rolled back, since it uses a value of B that has been previously written by T_2.

   In order to make the schedule cascadeless, a commit must be placed immediately after $write(B)$ in T_2.