Database Management Systems
Written Examination

01.07.2008

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	12	
7	8	
Total	100	

Exercise 1 (20 pt) Answer the following questions:

a. Describe briefly two different buffer replacement strategies.

b. When and why is a multi-level index recommended?

c. Can bucket overflow be eliminated in hashing? If yes, how?

d. Mention two index structures that can efficiently handle multiple-key queries.

e. What are the tree steps in query processing?

f. Consider a materialized view $v = r \bowtie s$. How can v be updated incrementally if i_r tuples are inserted into r?

g. Every view serializable schedule is also conflict serializable. Is this statement correct?

h. What is a deadlock?

i. In log-based recovery with deferred DB modifications: What actions are required after a rolled back transaction?

j. Consider log-based recovery with immediate DB modifications and the following log file: $\langle T_0, start \rangle$, $\langle T_0, A, 1000, 950 \rangle$, $\langle T_0, B, 2000, 1950 \rangle$. What actions are performed if the system crashes in this situation?

Exercise 2 (10 pt) The following table shows a file organization that represents variable-length records using the pointer method ("$\uparrow r_i$" denotes a pointer to record r_i, and $\perp$ denotes the end of a chain).

r_0	Jan	P1	400	$\uparrow r_2$
r_1	Joe	P3	350	
r_2		P2	500	$\perp$
r_3	Ann	P1	700	$\uparrow r_4$
r_4		P4	900	$\perp$

a. Show the file after the execution of the following steps:
   - Insert(Jan, P7, 800)
   - Insert(Ann, P2, 250)
   - Delete(Jan, P1, 400)

b. Transform the result of a) into a pointer representation that uses an anchor block and an overflow block.

c. What is the main disadvantage of the method in a) compared to the method in b)?

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

	Name	*Course*	*Grade*
r_0	Tom	ITP	30
r_1	Tom	DMS	30
r_2	Aron	CSA	18
r_3	Ann	OS	18
r_4	Ann	DMS	30
r_5	Nick	ITP	28
r_6	Nick	DSA	23
r_7	Nick	IDB	26
r_8	Sue	ITP	28
r_9	Sue	CSA	28

Show the following index structures and file organisations:

a. An index-sequential file organisation with a primary dense index on *Name* and a secondary index on *Grade*.

b. A primary dense B$^+$-tree index on *Course*. Assume $n = 3$ for the B$^+$-tree. The tuples are inserted in the order $r_0, r_1, r_2, \ldots, r_9$.

c. A static hash file organisation on *Grade* with hash function $h(n) = n \mod 4$. Each bucket holds at most 2 tuples. The tuples are inserted in the order $r_0, r_1, r_2, \ldots, r_9$.

d. For the primary index in a: briefly describe (in pseudocode) the insertion of a new tuple $(name, course, grade)$; describe both the index update and the data file update.

Exercise 4 (20 pt) Assume two relations $r(A, B)$ and $s(A, C)$ with $|r| = 15.000.000$ and $|s| = 800.000$. The block size is 2.000 Bytes, the tuple size 400 Bytes for both relations. The values of the integer attribute A are uniformly distributed between 1 and 500.000 in relation r. The disk performance is given as follows: latency time = 0.008 sec, seek time = 0.016 sec, transfer time = 0.001 sec.

a. Consider a primary B$^+$-tree index on attribute A in relation r, where each node contains 100 index entries. Determine the number of blocks at each level of the tree.

b. Determine the number of block IOs and the execution time for $\sigma_{A=x}(r)$, if
   - the index in a) is used
   - the index is not used, and r might or might not be sorted on A.

c. Determine the number of block IOs for the following evaluation plans for $s \bowtie r$, when $M = 3$ main memory buffer blocks are available:
   - Plan p1: Block nested loop join
   - Plan p2: Indexed nested loop join using the B$^+$ index in a)
   - Plan p3: Merge join (assume that the relations are already sorted)

d. Using the same strategies as in p1, p2, and p3: is $r \bowtie s$ a better join ordering, the same, or worse? Explain your answer.

Exercise 5 (10 pt) Consider relation $r(A, B, C)$ with an index on the key attribute A, relation $s(C, D, E)$ with an index on C, and a materialized view $v = r \bowtie s$ with no index.

a. Describe an evaluation strategy for the RA expression $\sigma_{A=10}(v)$.

b. Write an equivalent RA expression which allows a more efficient evaluation. Explain the optimization step(s) and the evaluation of the new expression. (Hint: consider the view v and the indexes)

c. Suppose that we have a third relation $t(E, F)$. What is the number of different join orderings for $r \bowtie s \bowtie t$?

Exercise 6 (12 pt) Given is the following schedule over transactions T_1, T_2, T_3:

T_1	T_2	T_3
	read(Z)	
	read(Y)	
	write(Y)	
		read(Y)
		read(Z)
read(X)		
write(X)		
		write(Y)
		write(Z)
	read(X)	
read(Y)		
write(Y)		
	write(X)	

Answer the following questions and explain your answers:

a. Draw the conflict graph of this schedule and show whether the schedule is conflict serializable or not.

b. Is the schedule view serializable to $\langle T_1, T_2, T_3 \rangle$?

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

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 and explain the following questions:

a. Is this 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?

Solution 1

a. LRU: replace the block least recently used
MRU: replace the block most recently used

b. If the (primary) index does not fit entirely in main memory.

c. No

d. Grid files and bitmap index

e. Parsing and translation, Optimization, Evaulation

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

g. No

h. A system is in a deadlock state if there is a set of transactions such that every transaction in the set is waiting for another transaction in the set.

i. Nothing; the log is ignored.

j. *undo*(T_0), i.e., A is restored to 1000, and B is restored to 2000.

Solution 2

a. File after the 3 update operations:

r_0				
r_1	Joe	P3	350	
r_2	Jan	P2	500	$\uparrow r_5$
r_3	Ann	P1	700	$\uparrow r_4$
r_4		P4	900	$\uparrow r_6$
r_5		P7	800	$\perp$
r_6		P2	250	$\perp$

b. Pointer representation with anchor block and overflow block:

	r_0	Joe	P3	350	
Anchor block:	r_1	Jan	P2	500	$\uparrow s_0$
	r_2	Ann	P1	700	$\uparrow s_1$

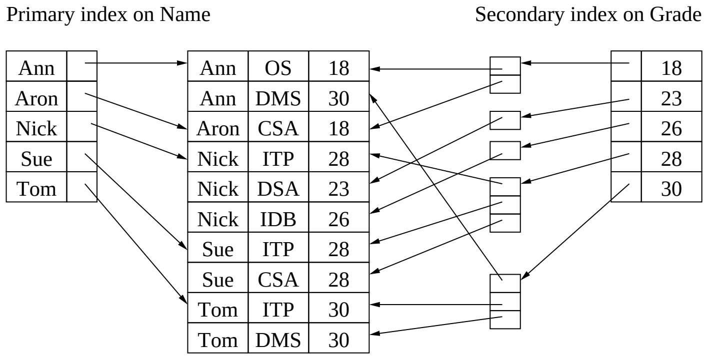
	s_0	P7	800	$\perp$
Overflow block:	s_1	P4	900	$\uparrow s_2$
	s_2	P2	250	$\perp$

Note: This method (immediately) follows the pointer chains during the transformation, thus $(P7, 800, \perp)$ is the first overflow record. If the data records are scanned sequentially, the order of tuples in the overflow block is different.

c. Space is wasted (i.e., the Name attribute is empty) in all records except the first in a chain.

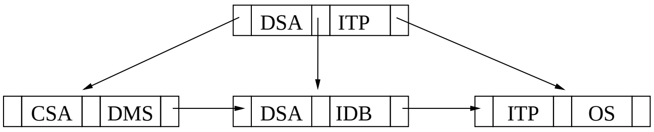
Solution 3

a. Index-sequential file organisation:



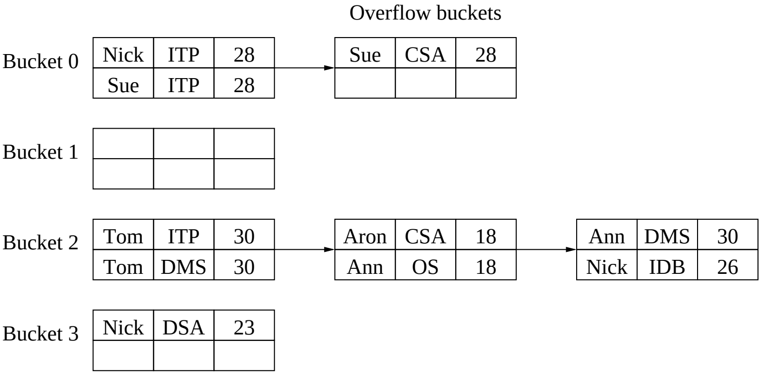
Note: Instead of buckets, several index entries might be used in the secondary index.

b. B^+-tree index



c. Hash file organization:

- The hash function computes the bucket: $18 \mod 4 = 2$, $23 \mod 4 = 3$, $26 \mod 4 = 2$, $28 \mod 4 = 0$, $30 \mod 4 = 2$
- Overflow buckets are used, if a bucket is already full.



d. Index and data update after inserting tuple $(name, course, grade)$:
  1. Perform a lookup in the index with search-key value *name*
  2. If *name* does not appear in index:
     – insert index record with *name*;
     – locate last data record with largest *Name* value smaller than *name*;
     – insert data record immediately after that record;
     – make new index record point to the new data record;
  3. Otherwise:
     – the index needs not to be updated;
     – follow index pointer to data file;
     – insert new data record after the other records with search-key value *name*

Solution 4

– 2.000/400 = 5 data tuples/block for both relations
– Number of blocks for r: 15.000.000/5 = 3.000.000
– Number of blocks for s: 800.000/5 = 160.000

a. Nodes (=index blocks): 100 index entries per node
Index blocks required at each level:
- level 3: $\lceil 500.000/100 \rceil = 5.000$ blocks (leaf nodes)
- level 2: $\lceil 5.000/100 \rceil = 50$ blocks
- level 1: $\lceil 50/100 \rceil = 1$ block
$\Rightarrow$ 5.051 index blocks are needed in total

b. $\sigma_{A=x}(r)$
The B$^+$-tree index is used:
– Traverse the tree: 3 index block IOs
– Read all data blocks with qualifying tuples (i.e., $A = x$)
  – On avg. $15.000.000/500 = 30$ qualifying data tuples $= 6$ data blocks
– Total block IOs: $3 + 6 = 9$
– Time: $1 \text{ IO} = 0.008 + 0.016s + 0.001s = 0.025s$
– $\Rightarrow 0.025 \times 9 = 0.225$ sec

B$^+$-tree index is not used, r is sorted on B: use binary search on A
– $\lceil \log_2 3.000.000 \rceil = 22$ blocks for binary search
– Total block IOs: $22 + 5 = 27$
– Time: $27 * 0.025 = 0.675$ sec

B$^+$-tree index is not used, r is not sorted on B: scan entire relation
– Total block IOs: 3.000.000
– Time: $3.000.000 * 0.025 = 75.000$ sec

c. $s \bowtie r$ Plan p1: Block nested loop join
– $Cost = b_s * b_r + b_s = 160.000 * 3.000.000 + 160.000 = 480.000.160.000$ block IOs

Plan p2: Indexed nested loop join
– $Cost = n_s * c + b_s = 800.000 * 9 + 160.000 = 7.360.000$ block IOs

Plan p3: Merge join
– Avg. number of tuples with same A-value in s: $800.000/500.000 = 2$
– All tuples with the same A-value fit in memory
– $Cost = b_s + b_r = 160.000 + 3.000.000 = 3.160.000$ IOs

d. $r \bowtie s$ Plan p1: Block nested loop join
– $Cost = b_r * b_s + b_r = 3.000.000 * 160.000 + 3.000.000 = 480.003.000.000$ IOs
$\Rightarrow$ worse

Plan p2: Indexed nested loop join
– not applicable since there is no index on s
– block nested loop join has to be used
$\Rightarrow$ worse

Plan p3: Merge join
With the simplifying assumption that all tuples with the same A-value fit in memory:
– $Cost = b_r + b_s = 3.000.000 + 160.000 = 3.160.000$ block IOs
$\Rightarrow$ the same

A more realistic assumption:
– Avg. number of tuples with the same A-value in r: 30 (= 6 blocks)

– s-Tuples with the same A-values do not fit in memory)
– Block nested loop join required between tuples with identical values
– ⇒ worse

Solution 5

a. $\sigma_{A=10}(v)$: Since there is no index and the data are not sorted, linear file scan is the only way to evaluate the query. On average, only 50% of the relation needs to be scanned, since at most one tuple with $A = 10$ exists.

b. Replace the view with its definition: $\sigma_{A=10}(r \bowtie s)$
Then push the selection down to r: $\sigma_{A=10}(r) \bowtie s$.

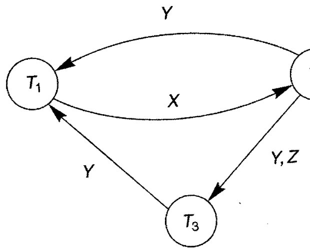
This expression is more efficient since it can take advantage of the indexes:
– use index scan on A to retrieve a single tuple that satisfies $A = 10$;
– use indexed nested-loop join to evaluate the join.

c. Number of join orderings for n relations: $(2(n-1))!/(n-1)!$
– For $r \bowtie s \bowtie t$: $(2 * (3-1))!/(3-1)! = 12$

– Join orderings:
$(r \bowtie s) \bowtie t$, $(s \bowtie r) \bowtie t$, $(r \bowtie t) \bowtie s$, $(t \bowtie r) \bowtie s$, $(s \bowtie t) \bowtie r$, $(t \bowtie s) \bowtie r$,
$r \bowtie (s \bowtie t)$, $r \bowtie (t \bowtie s)$, $s \bowtie (r \bowtie t)$, $s \bowtie (t \bowtie s)$, $t \bowtie (r \bowtie s)$, $t \bowtie (s \bowtie r)$

Solution 6

a. Conflict graph:



The schedule is not conflict serializable, since the conflict graph contains cycles.

b. No.
Example of violating a condition for view serializability: In the concurrent schedule T_2 reads the initial value of Y, and in $\langle T_1, T_2, T_3 \rangle$ the transaction T_2 reads the value of Y which is produced by T_1 (but should read the initial value).

c. No.
Because, for example, T_3 reads Y which was produced by T_2, hence T_2 must commit before T_3 commits in order for the schedule to be recoverable. In other words, if T_3 commits and later T_2 aborts, T_3 must be rolled back, since it used the value Y that was produced by T_2 and is no longer valid; but T_3 cannot roll back after the commit, so the schedule is not recoverable.

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 $T_0 = 0$ and $T_1 = 1$.
Then at step 4 the transaction T_1 sets the W-timestamp of B to 1.
Then at step 7 the $read(B)$ by T_2 is rejected, since the timestamp of T_0 is smaller than 1.