Database Management Systems
Written Exam

05.09.2011

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	8	
3	20	
4	24	
5	8	
6	8	
7	12	
Total	100	

Exercise 1 (20 pt) Answer the following questions:

a. What is minimized by disk-arm-scheduling algorithms (e.g., by the elevator algorithm)?

b. What is the difference between a dense and a sparse index?

c. What are two index structures that can efficiently handle multiple-key queries?

d. Can bucket overflow be avoided in hashing?

e. Assume $M \geq 3$ main memory blocks. How many blocks shall be used for the outer relation in a (block) nested loop join?

f. Which strategy for the evaluation of complex expression is more efficient: materialization or pipelining?

g. Is $E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$ a correct equivalence rule (for query optimization)?

h. Is the following statement correct: Every view serializable schedule is also conflict serializable?

i. What is stored in the lock table?

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

Exercise 2 (8 pt) Consider the following file of variable-length records that stores the `exams` relation of exercise 4 by using the byte string representation.

r_0	Jan	AI	C	3	DB	A	2	DMS	B	2	$\perp$
r_1	Ann	AI	A	3	CSA	D	1	$\perp$			
r_2	Bob	OS	C	1	DB	C	2	$\perp$			
r_3	Sue	DMS	A	2	ITP	A	1	$\perp$			
r_4	Joe	DMS	B	2	$\perp$						

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

b. Show the file structure with fixed-length representation with pointers and anchor and overflow block.

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

	Name	*Course*	*Grade*
r_0	Tom	ITP	30
r_1	Tom	DMS	21
r_2	Aron	CSA	18
r_3	Ann	OS	18
r_4	Ann	DMS	25
r_5	Nick	ITP	27
r_6	Nick	DSA	23
r_7	Nick	IDB	26
r_8	Sue	ITP	28
r_9	Sue	CSA	19

Show the following index structures and file organisations:

a. A primary dense B$^+$-tree index on *Course*. Assume $n = 3$ for the B$^+$-tree. The tuples are inserted in the order $r_0, \dots, r_9$. Show the tree after inserting the first five tuples and after inserting all tuples.

b. An extensible hash table on *Grade* with the hash function $h(n) = n \mod 8$. Each bucket holds at most 2 tuples. The tuples are inserted in the order $r_0, \dots, r_9$. Show the structure after inserting the first five tuples and after inserting all tuples.

c. Assume that you have a B$^+$-tree index on *Course* and a (separate) B$^+$-tree index on *Grade*. Consider the following query:

```
SELECT * FROM r WHERE Course = 'ITP' AND Grade = 30
```

Describe 3 different evaluation strategies for this query that take advantage of the indexes, using one of the indexes or both.

Exercise 4 (24 pt) Let $r(A, B)$ and $s(A, C)$ be two relations with the following characteristics: $|r| = 45.000$, $|s| = 20.000$, A is primary key in both relations and equally distributed between 1 and 1.000.000, and s has a primary B$^+$-tree index on attribute A with 100 search-key/pointer pairs per node. A single block can contain 25 tuples of r, 30 tuples of s, or 1 node of the index.

a. Determine the number of blocks needed for r, s, and the index, respectively.

b. Determine the access strategy and determine the number of block IOs for the following selection queries:

- $\sigma_{A=100.000}(s)$
- $\sigma_{A<100.000}(s)$
- $\sigma_{A>100.000}(s)$

c. Determine the number of block IOs for the following evaluation plans for $r \bowtie s$ when 3 buffer blocks in memory are available:

- Plan p1: Block nested loop join
- Indexed nested loop join using the index in a)
- Plan p3: Hash join

d. For the hash join in plan p3 above a partition of s need to fit entirely in main memory. Assume a main memory buffer size of 12 blocks. How should the buffer blocks be used and what would be a useful hash function such that the number of s-partitions is minimal, i.e., the partitions are maximal. (Assume that the A-values are perfectly distributed)

Exercise 5 (8 pt) Assume two relations $r(A, C)$ and $s(B, D)$ and the following relational algebra expression:

$$\sigma_{A<10 \wedge B>100 \wedge A+B<200}(r \times s)$$

a. Transform this selection statement into a more efficient expression by applying some equivalence rules. Explain your choice and why the new expression is more efficient.

b. Assume that you can create one single-attribute index on either relation r or s to improve the evaluation of the expression obtained in a.). Which index (type, relation, attribute) would you create? Motivate your choice and briefly describe the evaluation strategy with this index.

Exercise 6 (8 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 $\langle T_1, T_2, T_3 \rangle$?

Exercise 7 (12 pt) 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).

a. Add lock and unlock instructions to T_1 and T_2 so that they observe the two-phase locking protocol.

b. Show a concurrent schedule of T_1 and T_2 that results in a deadlock? Show also the evolution of the wait-for graph.

c. For the schedule in b.) what happens under the wait-die deadlock prevention protocol?

Solution 1

a. Disk-arm movement

b. A dense indes has an index record for every search-key value. A sparse index has an index record only for some search-key values.

c. Bitmap index and grid file index

d. No

e. $M - 2$

f. Pipelining

g. Yes

h. No

i. Granted locks and pending requests for locks

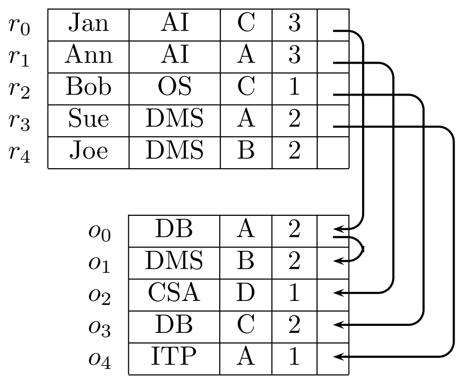
j. No actions need to be done

Solution 2

a. Fixed-length representation with reserved space

0	Jan	AI	C	3	DB	A	2	DMS	B	2
1	Ann	AI	A	3	CSA	D	1	⊥	⊥	⊥
2	Bob	OS	C	1	DB	C	2	⊥	⊥	⊥
3	Sue	DMS	A	2	ITP	A	1	⊥	⊥	⊥
4	Joe	DMS	B	2	⊥	⊥	⊥	⊥	⊥	⊥

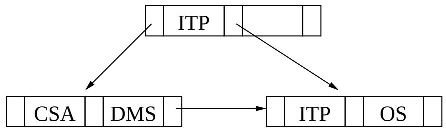
b. Fixed-length representation with pointers



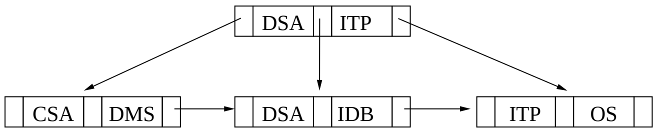
Solution 3

a. B^+-tree index

– after inserting $r_0, \ldots, r_4$:



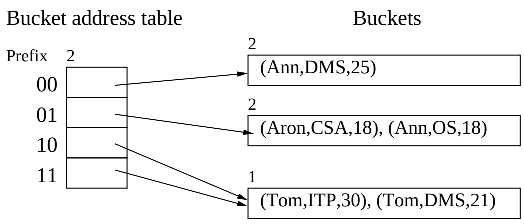
– after inserting all tuples:



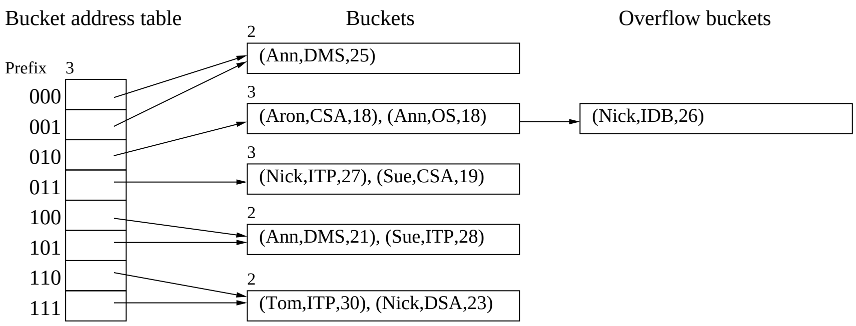
b. Extensible hash file organization:

- The hash function gives:
  $h(18) = 010$, $h(19) = 011$, $h(21) = 101$, $h(23) = 111$, $h(25) = 001$,
  $h(26) = 010$, $h(27) = 011$, $h(28) = 100$, $h(30) = 110$
- Overflow buckets are used, if a bucket is already full.

– after inserting $r_0, \ldots, r_4$:



– after inserting all tuples:



c. The 3 evaluation strategies are:

1. Use index on *Course* to find all tuples with *Course* = 'ITA'; then test for *Grade* = 30.
2. Use index on *Grade* to find all tuples with *Grade* = 30; then test for *Course* = 'ITA'.
3. Use index on *Course* to find pointers to all records with a *Course* = 'ITA'. Similarly, use index on *Grade* to find pointers to all records with a *Grade* = 30. Take the intersection of the two pointer sets.

Solution 4

a. Data blocks for r: $b_r = \lceil 45.000/25 \rceil = 1.800$ blocks
Data blocks for s: $b_s = \lceil 20.000/30 \rceil = 667$ blocks
Index on s:
– level 3: $\lceil 20.000/100 \rceil = 200$ nodes
– level 2: $\lceil 200/100 \rceil = 2$ nodes
– level 1: $\lceil 2/100 \rceil = 1$ node
Total for index: 203 blocks

b. $\sigma_{A=100.000}(s)$:
– Traverse the B^+-tree to locate the matching tuple
– 3 index block IOs + 1 data block IO = 4 block IOs
$\sigma_{A<100.000}(s)$:
– Scan the data file from the beginning; the index is not needed.
– Avg. distance between A-values: $1.000.000/20.000 = 50$
– Tuples that match the selection predicate: $100.000/50 = 2.000$
– Thus, $\lceil 2.000/30 \rceil = 67$ data block IOs
$\sigma_{A>100.000}(s)$:
– Traverse the B^+-tree to locate the first matching tuple: 3 index blocks
– Scan the data file sequentially from that tuple
– Avg. distance between A-values: $1.000.000/20.000 = 50$
– Tuples that match the selection predicate: $900.000/50 = 18.000$
– Thus, $\lceil 18.000/30 \rceil = 600$ data block IOs
– Total block IOs: $3 + 600 = 603$

c. Plan p1: Block nested loop join (with r as outer relation):
– $C = b_r * b_s + b_r = 1.800 * 667 + 1.800 = 1.202.400$
Plan p2: Indexed nested loop join:
– Use the index to access matching tuples in s
– Cost c to access a matching tuple: $c = 3 + 1 = 4$ block IOs
– Cost for p2: $C = n_r * c + b_r = 45.000 * 4 + 1.800 = 181.800$
Plan p3: Hash join (partially filled blocks are ignored):
– $C = 3 * (b_r + b_s) = 3 * (1.800 + 667) = 7.401$

d. – Use 1 block for the result, 1 block for r-partitions, 10 blocks for s-partitions
– An s-partition can hold at most $30 * 10 = 300$ tuples
– Avg. distance between A-values in s: $\lceil 1.000.000/20.000 \rceil = 50$
– The range of A-values that fit in a partition is $300 * 50 = 15.000$
– A hash function that assigns 1.500 tuples to a partition: $h = A$ div 15.000

Solution 5

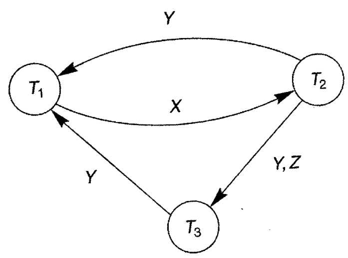
a. First, since condition $A < 10$ refers only to relation r, it can be pushed down to r. Similar, $B > 100$ can be pushed down to s, and we get $\sigma_{A+B<200}(\sigma_{A<10}(r) \times \sigma_{B>100}(s))$, which produces a smaller Cartesian product.

Second, the condition $A + B < 200$ can be pushed down to transform the Cartesian product into a join. The final expression is then: $\sigma_{A<10}(r) \bowtie_{A+B<200} \sigma_{B>100}(s)$

b. Create an ordered index on attribute B of relation s. The index can then be used to locate the first tuple with $B > 100$ and then continue to scan the relation. For the condition on relation r is not useful, since the relation is anyway scanned from the beginning.

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; in $\langle T_1, T_2, T_3 \rangle$, transaction T_2 reads the value of Y which is produced by T_1 (but should read the initial value).

Solution 7

a. Lock and unlock instructions:

T_1:	lock-S(A);	T_2:	lock-S(B);
	read(A);		read(B);
	lock-X(B);		lock-X(A);
	read(B);		read(A);
	if A=0 **then** B := B + 1;		**if** B=0 **then** A := A + 1;
	write(B).		write(A).
	unlock(A);		unlock(B);
	unlock(B);		unlock(A);

b. The following schedule results in a deadlock at step 6:

	T_1	T_2	Wait-for graph
1	lock-S(A);		
2		lock-S(B);	
3		read(B);	
4	read(A)		
5	lock-X(B)		$T_1 \longrightarrow T_2$ (T_1 waits for T_2)
6		lock-X(B);	$T_1 \rightleftarrows T_2$ (T_1 waits for T_2 and vice versa)

c. Wait-die deadlock prevention protocol: We assume that T_1 is the older transaction and T_2 is the younger transaction. Then at step 6, T_2 (the younger transaction) will not wait for T_1 (the older transaction) to release the lock. Instead, T_2 is rolled back, and the lock on B is released. T_1 can now continue.