Database Management Systems
Written Examination

27.09.2006

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	8	
3	12	
4	8	
5	16	
6	8	
7	16	
8	12	
Total	100	

Exercise 1 Answer the following questions:

1. What are the three performance measures of hard disks?
2. What is a primary index?
3. Mention two index structures that are designed for efficient querying on multiple keys?
4. What is the number of block accesses for a merge-join, assuming that the relations are sorted and the join attribute(s) in one relation forms a key.
5. What are the two forms of materialized view maintenance?
6. How works cost-based optimization?
7. What are the ACID properties?
8. How can a conflict graph be used to test for conflict serializability?
9. What are two pitfalls (problems) of lock-based protocols?
10. What are the two ways to address deadlocks?

Exercise 2 Given is the following table with project assignments:

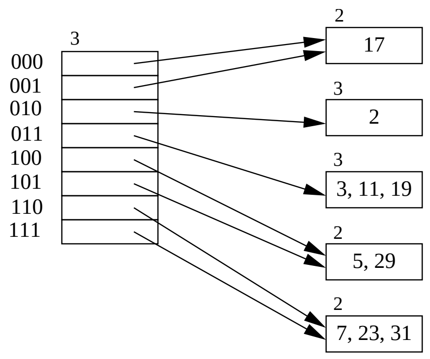
Name	Project	Hours
Jan	P7	800
Ann	P2	250
Jan	P1	400
Jan	P3	500

To store this information in a file, we assume the use of one variable-length record for each person, in which all project assignments for that person are stored.

a. Show the file organization using the reserved space method.

b. Show the file organization using the slotted page structure.

Exercise 3 Given is the following extandable hash table which uses the hash function $h(x) = x \bmod 8$ and a bucket size of 3:



Show the hash table after each of the following operations (applied in that order):
- Insert 1
- Insert 46
- Delete 11
- Delete 19

Exercise 4 Given is the following table `ProjAss` with project assignments in a company with 3 different departments:

	Name	Dept	Project	Hours
0	Jan	DB	P7	3100
1	Ann	DB	P2	700
2	Joe	AI	P1	1500
3	Bob	SE	P3	150
4	Pim	SE	P3	1850
5	Sue	DB	P10	5000
6	Bob	SE	P3	1300
7	Joe	AI	P17	180

a. Create a meaningful bitmap index on the attributes Dept and Hours (taking into consideration that this table might change and become large). Motivate your decision.

b. Describe and explain the use of this index for the evaluation of the query $\sigma_{Dept='DB' \wedge (Hours>3000 \vee Hours<200)}(\texttt{ProjAss})$, i.e., all employees from the DB department with very small (< 200) or very large (> 3000) project assignments.

Exercise 5 Consider a relation `r` with schema (`A`, `B`), a primary B$^+$ tree index on attribute `A`, and the following characteristics: $|r| = 4,000,000$; 100 index entries per block; 20 tuples per block; the values of `A` are uniformly distributed between 1 and $10,000,000$; there are no two tuples with identical values on `A`.

a. Determine the number of blocks (= number of nodes) used for the B$^+$-tree if index blocks are filled up to 80%.

b. Determine the number of block IOs (distinguishing between index blocks and data blocks) for the following queries:
   - Q1: $\sigma_{\texttt{A}<5Mio}(r)$
   - Q2: $\sigma_{\texttt{A}>5Mio}(r)$
   - Q3: $\sigma_{\texttt{A}\neq 5Mio}(r)$

c. Is the use of the B$^+$ tree index in all of the above queries useful? Explain your answer.

Exercise 6 Which of the following equivalence rules for query optimization are correct/not correct? Explain your answer and specify conditions under which a rule holds or does not hold.

a. $\sigma_{\theta_1 \wedge \theta_2 \wedge \theta_3}(E) = \sigma_{\theta_3}(\sigma_{\theta_2}(\sigma_{\theta_1}(E)))$
b. $\sigma_\theta(E_1 - E_2) = \sigma_\theta(E_1) - E_2$
c. $\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie E_2) = \sigma_{\theta_1}(E_1) \bowtie \sigma_{\theta_2}(E_2)$
d. $\pi_L(\sigma_\theta(E)) = \sigma_\theta(\pi_L(E))$

Exercise 7 Let relations $r_1(A, B, C)$ and $r_2(C, D, E)$ have the following properties: r_1 has 20,000 tuples, r_2 has 45,000 tuples, 25 tuples of r_1 fit on one block, and 30 tuples of r_2 fit on one block.

Compute the costs of the following evaluation plans for $r_1 \bowtie r_2$:

a. Plan p1: Nested-loop join with r_1 as outer relation
b. Plan p2: Block nested-loop join with r_1 as outer relation
c. Plan p3: Merge join if r_1 and r_2 are initially sorted

d. Plan p4: Merge join if r_1 and r_2 are not initially sorted
e. Plan p5: Hash join (assuming that no overflow occurs)

Exercise 8 Given is the following schedule that involves 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 answers:

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?

Solution 1

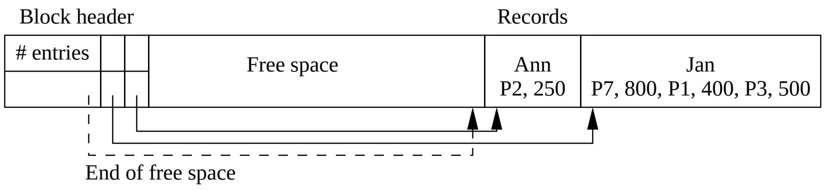
1. Access time, data-transfer rate, and mean time to failure.
2. The index whose search key specifies the sequential order of the file (table rows).
3. Grid files and bitmap index.
4. $b_r + b_s$, where b_r and b_s is the number of blocks of relation r and s, respectively.
5. Recomputation on every database update or incremental view maintenance.
6. Cost-based optimization works in 3 steps:
   1. Generating logically equivalent expressions (using equivalence rules);
   2. Annotating resultant expressions to get alternative query plans;
   3. Chooosing the cheapest plan based on estimated cost.
7. Atomicity (a transaction is atomic)
   Consistency (a transaction leads to a consistent state)
   Isolation (it appears to each transaction to be executed in isolation)
   Durability (after a transaction completes successfully, the changes persist)
8. A schedule is conflict serializable, if its conflict graph is acyclic.
9. Too early unlocking can lead to non-serializable schedules. Too late unlocking can lead to deadlocks.
10. Deadlock prevention (ensure that you never enter in a deadlock situation) and deadlock detection/recovery (deadlock has to be detected and then resolved by rolling back some transations)

Solution 2

a. Reserved space method for variable-length records

0	Jan	P7	800	P1	400	P3	500
1	Ann	P2	250	$\perp$	$\perp$	$\perp$	$\perp$

b. Slotted space structure for variable-length records

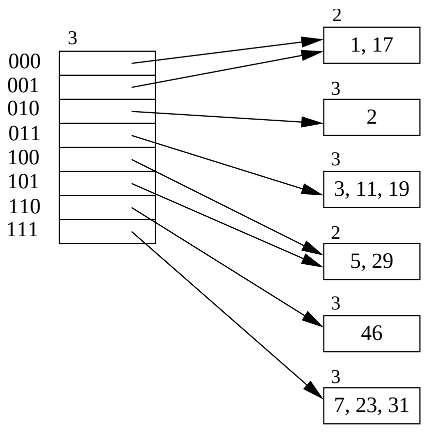


Solution 3

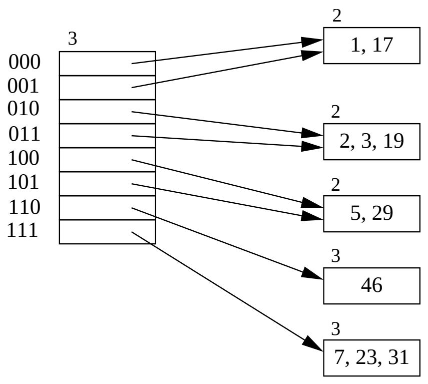
a. Insert 1: Key 1 is added to the first bucket, which becomes

$$\overset{2}{\boxed{1,\ 17}}$$

b. Insert 46: Key 46 is assigned to the last bucket and creates an overflow. The last bucket is split into two buckets, the bucket address table is updated, and the keys in this bucket are re-inserted.



c. Delete 11: Key 11 is removed from the third bucket. The second and third buckets are coalesced and the bucket address table is updated.



d. Delete 19: Key 19 is removed from the second bucket, which becomes

2

$\boxed{2,\ 3}$

Solution 4

a. Bitmap indexes:

One bitmap vector for each different department:

DB:	`[1 1 0 0 0 1 0 0]`
AI:	`[0 0 1 0 0 0 0 1]`
SE:	`[0 0 0 1 1 0 1 0]`

The values of the Hours attribute have to be split into a small number of ranges, and a bitmap vector is created for each range, e.g.,

H1 (< 200):	`[0 0 0 1 0 0 0 1]`
H2 (200-1000):	`[0 1 0 0 0 0 0 0]`
H3 (1000-2000):	`[0 0 1 0 1 0 0 0]`
H4 (2000-3000):	`[0 0 0 0 0 0 0 0]`
H5 (> 3000):	`[1 0 0 0 0 1 0 0]`

b. Take the bitmap vectors of H1 and H5 and compute the logical OR:

`[0 0 0 1 0 0 0 1]` OR `[1 0 0 0 0 1 0 0]` = `[1 0 0 1 0 1 0 1]`

Next, take this intermediate result and the bitmap vector of DB and compute the logical AND:

`[1 0 0 1 0 1 0 1]` AND `[1 1 0 0 0 1 0 0]` = `[1 0 0 0 0 1 0 0]`

Retrieve the tuple(s) with a '1' in the resulting bitmap vector, i.e., tuple 0 and tuple 5.

Solution 5

a. Index blocks:
$0.8 * 100 = 80$ index entries per block
Index blocks required at each level:
- level 4: $\lceil 4,000,000/80 \rceil = 50,000$ blocks (leaf nodes)
- level 3: $\lceil 50,000/80 \rceil = 625$ blocks
- level 2: $\lceil 625/80 \rceil = 8$ blocks
- level 1: $\lceil 8/80 \rceil = 1$ block

$\Rightarrow 50,634$ index blocks are needed in total

b. Data blocks: $\lceil 4,000,000/20 \rceil = 200,000$ blocks

$Q1$:
4 index blocks (one at each level)
$100,000$ data blocks ($= 50\%$ of the data blocks)
$\Rightarrow 4 + 100,000 = 100,004$ blocks are transfered in total
Note: Instead of reading the 4 index blocks, we could immediately start to scan the data blocks.

$Q2$:
The same situation as for $Q1$ (except that the 4 index blocks <u>have</u> to be read)

$Q3$:
4 index blocks (one at each level)
$200,000$ data blocks (all data blocks have to be read)
$\Rightarrow 4 + 200,000 = 200,004$ blocks are transfered in total

c. In the first and the last query the index is not useful. In the first query, we always have to scan data tuples from the beginning (remember that we use a primary index). In the last query, all tuples have to be scanned. Hence, in both cases we can simply start to scan the data tuples.

Solution 6

a. Always correct. We can show this by repeatedely applying the following two equivalence rules:
(r1) $\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$
(r2) $\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$

$\sigma_{\theta_1 \wedge \theta_2 \wedge \theta_3}(E) \overset{r1}{=} \sigma_{\theta_1 \wedge \theta_2}(\sigma_{\theta_3}(E)) \overset{r2}{=} \sigma_{\theta_3}(\sigma_{\theta_1 \wedge \theta_2}(E)) \overset{r1}{=} \sigma_{\theta_3}(\sigma_{\theta_1}(\sigma_{\theta_2}(E))) \overset{r2}{=} \sigma_{\theta_3}(\sigma_{\theta_2}(\sigma_{\theta_3}(E)))$

b. Always correct. We proof this by showing set containment in both directions:
($\longrightarrow$) Assume that $\exists t \in \sigma_\theta(E_1 - E_2)$. Then t satisfies θ and $t \in E_1$ and $t \notin E_2$. Therefore we get $t \in \sigma_\theta(E_1)$. We get also $t \in \sigma_\theta(E_1) - E_2$ since $t \notin E_2$.
($\longleftarrow$) Assume that $\exists t \in \sigma_\theta(E_1) - E_2$. Then t satisfies θ and $t \in E_1$ and $t \notin E_2$. Therefore we get $t \in (E_1 - E_2)$, and since t satisfies θ we get further $t \in \sigma_\theta(E_1 - E_2)$.

c. Correct, if θ_1 uses only attributes in E_1 and θ_2 uses only attributes in E_2.

d. Correct, if θ uses only attributes in L.

Solution 7
Tuples of r_1: $n_{r_1} = 20,000$
Tuples of r_2: $n_{r_1} = 45,000$
Blocks required for r_1: $b_{r_1} = \lceil 20,000/25 \rceil = 800$ blocks
Blocks required for r_2: $b_{r_2} = \lceil 45,000/30 \rceil = 1500$ blocks

a. $cost(p1) = b_{r_1} + n_{r_1} * b_{r_2} = 800 + 20,000 * 1,500 = 30,000,800$

b. $cost(p2) = b_{r_1} + b_{r_1} * b_{r_2} = 800 + 800 * 1,500 = 1,200,800$

c. We assume that all tuples for any given value of the join attributes fit in memory:
$cost(p3) = b_{r_1} + b_{r_2} = 800 + 1,500 = 2,300$

d. We have to add the cost of sorting (using external sort-merge) to the cost of p3, i.e.,
$cost(p4) = cost(p3) + cost(sorting)$

$cost(\text{sorting relation } r) = b_r(2\lceil log_{M-1}(b_r/M)\rceil + 1)$, where M is the number of blocks in buffer. We have to add to these costs the output of the sorted relation, i.e., add b_r block transfers.
$\Rightarrow cost(\text{sorting}) = 800 * (2\lceil log_{M-1}(800/M)\rceil + 2) + 1,500 * (2\lceil log_{M-1}(1,500/M)\rceil + 2)$

e. $cost(p5) = 3 * (b_{r_1} + b_{r_2}) = 3 * (800 + 1,500) = 6,900$

Solution 8

a. No. For both possible serial schedules, $\langle T_1, T_2 \rangle$ and $\langle T_2, T_1 \rangle$, we get either a conflict with $write(A) - read(A)$ or with $write(B) - read(B)$.

b. No.
In the serial schedule $\langle T_1, T_2 \rangle$, 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 $\langle T_2, T_1 \rangle$, 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.

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