Database Management Systems
Written Examination

22.06.2007

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	20	
6	8	
7	12	
8	12	
Total	100	

Exercise 1 Answer the following questions:

a. What is the buffer manager and what is its goal?

b. Which class of queries is better supported by the clustering file organization: selection or join?

c. What index is preferrable for a range query: primary index or secondary index? Explain your answer.

d. Which of the two index structures provides a more efficient access: B^+-tree or B-tree? Why?

e. What are the 3 steps of query processing?

f. Briefly describe the "linear search" algorithm (for the evaluation of selection queries).

g. What is a schedule?

h. Does the two-phase locking protocol ensure conflict serializable schedules?

i. What is stored in the lock table?

j. What are the two different approaches for log-based recovery?

Exercise 2 Consider the following file organization using fixed-length records with 4 data fields and a free list ("$\uparrow i$" denotes a pointer to record i).

header					$\uparrow 1$
record 0	BMW	1990	red	10	
record 1					$\uparrow 4$
record 2	BMW	1991	red	2	
record 3	Fiat	1990	white	3	
record 4					$\uparrow 6$
record 5	Fiat	1991	blue	3	
record 6					
record 7	Ford	1990	blue	1	
record 8	Ford	1990	red	7	
record 9	Ford	1991	red	3	

a. Show the structure of the file after each of the following steps (in that order):

(i) Insert(BMW,1991,blue,6)
(ii) Delete(record 2)
(iii) Insert(Ford,1990,white,7)

b. Describe a more *space* efficient representation of this file organization and show the initial situation above using this representation. (Hint: Consider which records contain free list pointers)

Exercise 3 Consider extensible hashing with a hash function $h(n) = n \bmod 8$ (n is a key), yielding a hash value of 3 bits. Each bucket can hold 3 data items.

a. Consider the insertion of the keys $0, 1, 2, 3, 4, 5, 6, 7$ (in that order).

b. Consider the insertion of the keys $7, 6, 5, 4, 3, 2, 1, 0$ (in that order).

For a.) and b.) draw the hash index after inserting the first four keys and after all the keys are inserted. Be sure to indicate the number of bits used in the bucket address table as well as in each bucket.

Exercise 4 Consider the following relation:

	Model	Year	Color	Sold
0	VW	1990	red	10
1	VW	1990	blue	5
2	VW	1991	blue	5
3	VW	1992	blue	6
4	Ford	1990	black	3
5	Ford	1990	blue	3
6	Ford	1991	red	2
7	Fiat	1990	red	20
8	Fiat	1990	blue	22
9	Fiat	1990	blue	22

a. Create a bitmap index for the attributes 'Model' and 'Color' and compute the size (in terms of Bytes) of the index.

b. Give a general formula for the size (Bytes) of a bitmap index for one attribute.

Exercise 5 Assume a relation `sales(pid, category, ...)` with $600,000$ tuples, where each tuple is 100 Bytes. The product ID `pid` is a key and is equally distributed between 1 and $3,000,000$. Furthermore, we have a block size of $2,000$ Bytes, seek time = 0.016 sec, latency = 0.008 sec, and transfer time = 0.001 sec.

a. Determine the number of data blocks needed to store the relation, if 25% of each data block is reserved for future insertions.

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

c. Consider the B^+-tree from b.) with the minimal number of blocks and assume that it is a primary index. Determine the number of IOs (data blocks + index blocks) and the execution time for the following two queries:

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

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

Exercise 6 Consider relation $r(A, B)$ with an index on the key attribute A, relation $s(B, C)$ with an index on B, and a materialized view $v = r \bowtie s$ with no index.

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

b. Write an equivalent RA expression which allows a more efficient evaluation. Explain the optimization step(s) and the evaluation of the new expression.

Exercise 7 Let relations $r(A, B)$ and $s(A, C)$ have the following properties: $|r| = 10,000$ with 50 tuples/block, $|s| = 30,000$ with 100 tuples/block, both relations are sorted on A, A is a key in r, and one r-tuple matches on average two s-tuples.

Compute the cost for the following evaluation plans for query $\sigma_{A=1000}(r \bowtie s)$:

a. Plan p1: Merge join followed by linear scan.

b. Plan p2: Merge join followed by binary search.

c. Which of the above plans allows pipelining between the join and the selection?

Exercise 8 Given is the following schedule for transactions T_1 and T_2:

T_1	T_2
read(B)	
	read(B)
B := B - 50	
write(B)	
	read(A)
read(A)	
	display(A+B)
A := A + 50	
write(A)	
display(A+B)	

a. Show the conflict graph of the schedule.

b. Is the schedule conflict serializable? Explain your answer.

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

d. Is the schedule possible under the two-phase locking protocol? Explain your answer.

Solution 1

a. Buffer manager is the subsystem which is responsible for buffering disk blocks in main memory. Tries to minimize the number of disk accesses.

b. Join.

c. Primary index. Use the index only to locate the first data item; then you can scan the data file. With a secondary index, you need the index to locate every data item.

d. B-tree. Some search-key values (and corresponding data pointers) are in non-leaf nodes, and hence are found before reaching a leaf node. In a B^+-tree all data pointers are in the leaves, hence the entire tree is always traversed.

e. (i) Parsing and translation, (ii) Optimization, (iii) Evaluation

f. Linear search algorithm: Scan each file block and test all records to see whether they satisfy the selection condition.

g. Sequence of instructions from concurrent transactions indicating the chronological order in which these instructions are executed.

h. Yes.

i. The lock table stores granted locks and pending requests for locks.

j. Deferred DB modifications and immediate DB modifications

Solution 2

a. (i) Insert(BMW,1991,blue,6)

header					↑ 4
record 0	BMW	1990	red	10	
record 1	BMW	1991	blue	6	
record 2	BMW	1991	red	2	
record 3	Fiat	1990	white	3	
record 4					↑ 6
record 5	Fiat	1991	blue	3	
record 6					
record 7	Ford	1990	blue	1	
record 8	Ford	1990	red	7	
record 9	Ford	1991	red	3	

(ii) Delete(record 2)

header					↑ 2
record 0	BMW	1990	red	10	
record 1	BMW	1991	blue	6	
record 2					↑ 4
record 3	Fiat	1990	white	3	
record 4					↑ 6
record 5	Fiat	1991	blue	3	
record 6					
record 7	Ford	1990	blue	1	
record 8	Ford	1990	red	7	
record 9	Ford	1991	red	3	

(iii) Insert(Ford,1990,white,7)

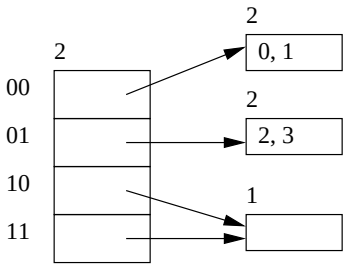
header					↑ 4
record 0	BMW	1990	red	10	
record 1	BMW	1991	blue	6	
record 2	Ford	1990	white	7	
record 3	Fiat	1990	white	3	
record 4					↑ 6
record 5	Fiat	1991	blue	3	
record 6					
record 7	Ford	1990	blue	1	
record 8	Ford	1990	red	7	
record 9	Ford	1991	red	3	

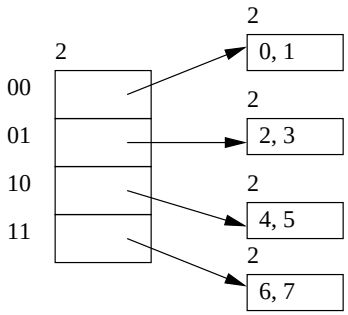
b. The main idea is that free list pointers are only stored in empty (deleted) records, hence the last column with the pointers is not needed. Instead, the pointers can be stored in the data fields.

header	↑ 1			
record 0	BMW	1990	red	10
record 1	↑ 4			
record 2	BMW	1991	red	2
record 3	Fiat	1990	white	3
record 4	↑ 6			
record 5	Fiat	1991	blue	3
record 6				
record 7	Ford	1990	blue	1
record 8	Ford	1990	red	7
record 9	Ford	1991	red	3

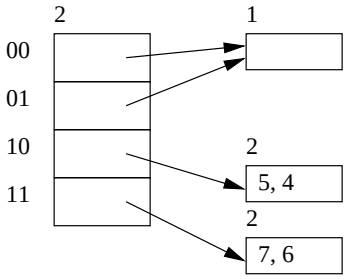
Solution 3

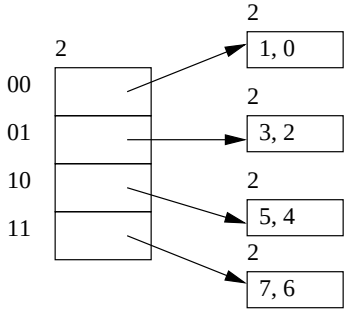
a.





b.





Notice: The order of the data records in the buckets does not matter, hence the result after inserting all records is the same in both cases.

Solution 4

a. Bitmap index:
   – One bitmap vector for each different value of the Model attribute:
   VW: `[1 1 1 1 0 0 0 0 0 0]`
   Ford: `[0 0 0 0 1 1 1 0 0 0]`
   Fiat: `[0 0 0 0 0 0 0 1 1 1]`
   – One bitmap vector for each different value of the Color attribute:
   red: `[1 0 0 0 0 0 1 1 0 0]`
   blue: `[0 1 1 1 0 1 0 0 1 1]`
   black: `[0 0 0 0 1 0 0 0 0 0]`

   Each bitmap vector requires 2 Bytes, thus the entire index requires 12 Bytes.

b. For a relation r with an attribute A assuming m different values in r, the size (in Bytes) of a bitmap vector for A is:

$$size = m \times \lceil |r|/8 \rceil \text{ Bytes}$$

Solution 5

a. Data blocks:
   $0.75 \times 2,000 = 1,500$ Bytes/block can be used
   $\lfloor 1500/100 \rfloor = 15$ tuples/block
   $\lceil 600,000/15 \rceil = 40,000$ data blocks are needed

b. 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

c. 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.

   Time for 1 block IO: $0.016 + 0.008 + 0.001 = 0.025$ sec.

   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 + $\lceil 2,000/15 \rceil = 134$ data blocks $\Rightarrow$ 137 IOs
   – Execution time: $137 \times 0.025 = 3.43$ sec

   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 tuples are not needed to evaluate this query!
   – Block IOs: 3 index nodes + $2,000/200 = 10$ index leaf nodes $\Rightarrow$ 13 IOs
   – Execution time: $13 \times 0.025 = 0.33$ sec

d. 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 + 134 data blocks in the best case (or $2,000$ data blocks in the worst case) $\Rightarrow$ 147 IOs (or $2,013$ IOs)
– Execution time: $147 \times 0.025 = 3.68$ sec in the best case ($2,013 \times 0.025 = 50.33$ sec in the worst case

Q2: The same as in c.)

Solution 6

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.

Solution 7
– Blocks for r: $b_r = \lceil 10,000/50 \rceil = 200$ blocks
– Blocks for s: $b_s = \lceil 30,000/100 \rceil = 300$ blocks
– Size of join result: Assume that 40 result tuples fit on a block (i.e., a result tuple is a bit smaller than an r-tuple + an s-tuple). Further assume that the join attribute A is distributed in the same range in both relations. Then we get $10,000 \times 2 = 20,000$ result tuples.
– Blocks for $r \bowtie s$: $20,000/40 = 500$ blocks

a. Plan p1:
– Merge-join: $cost = b_r + b_s = 200 + 300 = 500$ block IOs
(+ 500 block IOs for writing result back to disk if pipelining is not possible)
– Linear scan: on avg. $2,000$ tuples have to be scanned
$cost = 2,000/40 = 50$ block IOs

b. Plan p2:
– Merge-join: $cost = 500$ block IOs
(+ 500 block IOs for writing result back to disk if pipelining is not possible)
– Binary search: $cost = \lceil log_2(20,000/40) \rceil = \lceil log_2(5,000) \rceil$ block IOs

c. Plan p1 allows piplined evaluation. Plan P2 does not, because binary search can only be done after all tuples are available.

Solution 8

a. Conflict graph

b. The schedule is conflict serializable, since the conflict graph contains no cycles. It is equivalent to the serial schedule $\langle T_2, T_1 \rangle$.

c. Lock and Unlock instructions:

T_1	T_2
Lock-X(B)	Lock-S(B)
read(B)	read(B)
B := B - 50	Lock-S(A)
write(B)	read(A)
Lock-X(A)	Unlock(B)
read(A)	Unlock(A)
A := A + 50	display(A+B)
write(A)	
Unlock(B)	
Unlock(A)	
display(A+B)	

d. No. Consider the following partial schedule:

T_1	T_2
Lock-X(B)	
read(B)	
	Lock-S(B)
	read(B)

Since T_1 holds an X-lock on B, transaction T_2 has to wait until T_1 releases this lock, which is not according to the schedule in a.).