Database Management Systems
Written Examination

02.02.2009

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 a clustering file organisation?

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

c. When is a sparse secondary index useful?

d. What is the main difference between a B$^+$-tree file organisation and a B$^+$-tree index?

e. What are the 3 steps of query processing?

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

g. What are the 5 states of a transaction?

h. What is a cascading rollback?

i. Which ACID properties are ensured by the recovery system?

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

Exercise 2 (10 pt) Consider the following relation:

Branch	Customer	Account
Downtown	Smith	237
Downtown	Jones	222
Mianus	Smith	250
Downtown	Turner	300
Mianus	Jackson	200
Mianus	Hayes	382
Downtown	Williams	180
Brighton	Jackson	290

Suppose that a branch with all its customer and accounts shall be stored in a variable-length record. Show the file organisation for the following methods:

a. Fixed-length representation with reserved space

b. Fixed-length representation with pointer

c. Slotted page structure

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

	Course	StudID	Grade
r_0	DMS	2100	18
r_1	ITP	2157	18
r_2	ITP	2230	30
r_3	DMS	2177	24
r_4	OS	2340	30
r_5	ITP	2200	23
r_6	DMS	2157	28
r_7	DB	2300	30
r_8	DMS	2263	25
r_9	DB	2299	28

Show the following index structures and file organisations:

a. An index-sequential file organisation with a primary sparse index on *StudID*. For a search-key k, an index entry is created if $k \mod 100 = 0$.

b. On top of the index-sequential structure in a), a secondary B^+-tree index on *Grade*. Assume $n = 3$ for the B^+-tree. The tuples are read sequentially as stored in the index-sequential file in a).

c. A hash file organisation using extendable hashing on *Grade* and the hash function $h(n) = n \mod 8$. Each bucket holds at most 2 tuples. Show the structure after inserting $r_0 - r_4$ and after inserting all tuples.

d. A bitmap index on *Course*.

Exercise 4 (20 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 main memory buffer blocks 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 (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 6 (10 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)	

a. Draw the precedence graph and show that the schedule is not conflict serializable.

b. Design a concurrent schedule of T_1, T_2, and T_3 that is conflict serializable. Specify also the equivalent serial schedule.

c. Assuming the timestamp order $TS(T_1) < TS(T_2) < TS(T_3)$: Is the schedule in a) possible under the timestamp ordering protocol? Explain your answer.

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: $A \rightarrow B$. 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. Records of several different relations are stored in the same file.

b. If the primary index does not fit entierely in main memory.

c. Never. Secondary indexes have always to be dense in order to be useful.

d. In the B^+-tree file organisation the leave nodes store the data records; in the B^+-tree index the leave nodes store pointers to the data records.

e. (i) Parsing and translation, (ii) optimization, (iii) evaluation

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

g. Active, partially committed, committed, failed, aborted

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

i. Atomicity and durability

j. No actions need to be done.

Solution 2

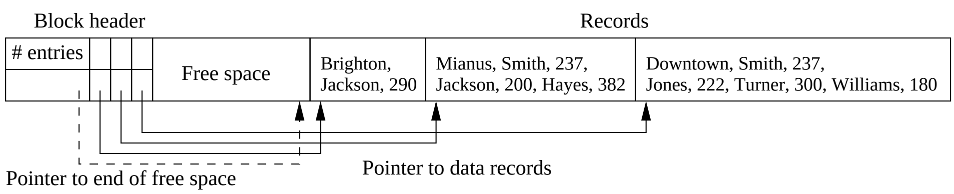
a. Fixed-length representation with reserved space:

0	Downtown	Smith	237	Jones	222	Turner	300	Williams	180
1	Mianus	Smith	250	Jackson	200	Hayes	382	⊥	⊥
2	Brighton	Jackson	290	⊥	⊥	⊥	⊥	⊥	⊥

b. Fixed-length representation with pointer:

0	Downtown	Smith	237	
1		Jones	222	
4	Mianus	Smith	250	
2		Turner	300	
4		Jackson	200	
5		Hayes	382	
3		Williams	180	
6	Brighton	jackson	290	

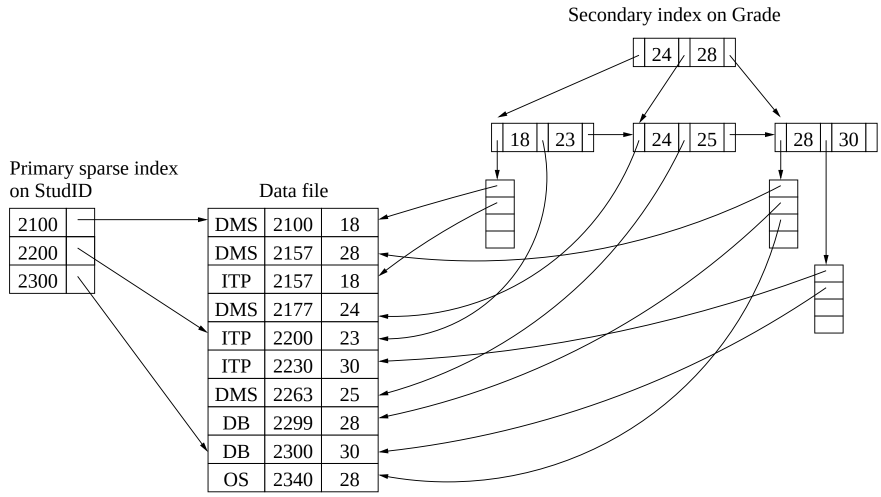
c. Slotted page structure:



Solution 3

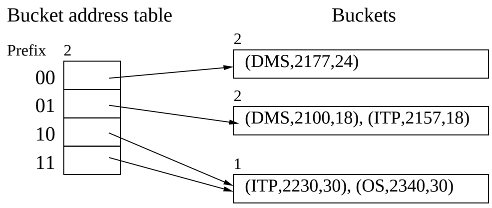
a. Index-sequential file organisation with primary index (see point b.)

b. Secondary B^+-tree index

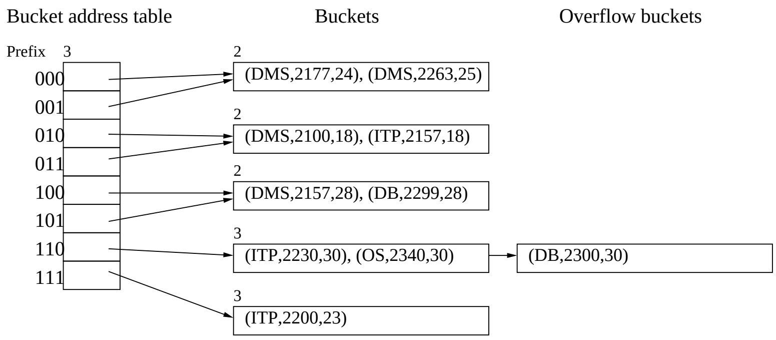


c. Extendable hashing

- after inserting $r_0, \ldots, r_4$:



- after inserting all tuples:



d. Bitmap index for *Course* and *Grade*:

DMS: [1 0 0 1 0 0 1 0 1 0]
ITP: [0 1 1 0 0 1 0 0 0 0]
OS: [0 0 0 0 1 0 0 0 0 0]
DB: [0 0 0 0 0 0 0 1 0 1]

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
– Thus, 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

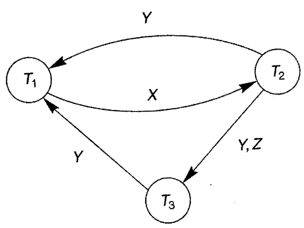
Solution 5

a. $\sigma_{(A=1 \vee A=3) \wedge B<C}(r \bowtie s)$:
– 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 6

a. Precedence graph:



There is a cycle in the precedence graph: $T_1 \xrightarrow{X} T_2$ and $T_2 \xrightarrow{Y} T_1$. Hence, the schedule is not conflict serializable.

b. The following schedule is conflict serializable to the serial schedule $\langle T_3, T_1, T_2 \rangle$

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

c. No. There are conflicting read and write operations on Y in T_1 and T_2, which are not executed in timestamp order: T_2 writes Y and then T_1 reads Y.

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. Yes, it is possible under the tree protocol.

	T_1	T_2
1	lock-X(A)	
2	read(A)	
3		lock-X(B)
4		write(B)
5		unlock(B)
6	lock-X(B)	
7	read(B)	
8	unlock(A)	
9	unlock(B)	

Note, that the tree protocol allows only X-locks.

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.