

# Database Management Systems

## Written Examination

14.02.2007

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	4	
7	16	
8	12	
Total	100	

**Exercise 1** Answer the following questions:

1. What is minimized by disk-arm-scheduling algorithms (e.g., by the elevator algorithm)?
2. What is the difference between a dense and a spares index?
3. What is an index-sequential file?
4. The non-leave nodes of a  $B^+$ -tree form a sparse index on the leave nodes. Is this statement true?
5. What are the two steps to evaluate  $\sigma_{A \geq v}(r)$ , if a primary index on  $A$  exists?
6. At what time during query processing does (query) optimization occur?
7. Is  $E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$  a correct equivalence rule (for query optimization)?
8. Mention two advantages of multiple transaction that are executed in parallel.
9. What are the phases of the two-phase locking protocol?
10. What is deadlock prevention?

**Exercise 2** Given is the following table with project assignments:

Employee	Project	Hours
Jan	P1	800
Ann	P1	250
Jan	P2	400
Jan	P3	500
Jan	P4	900
Joe	P3	350

- a. Show two different ways of a file organization using the reserved space method (variable-length records).
- b. Assume the following memory requirements for the attribute values: Name = 20 bytes, Project = 10 bytes, Hours = 4 Bytes. Calculate the disk space used by the two different solutions in (a).
- c. How much disk space is wasted (unused) by the two solutions in (a)?

**Exercise 3** Construct a  $B^+$ -tree for the following set of key values:

(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

Assume that the tree is initially empty, and values are added in ascending order. The number of pointers that will fit in one node is 4.

**Exercise 4** Consider the relation Grades(Stud,Grade) that contains the following tuples:  $(Jan, 25)$ ,  $(John, 25)$ ,  $(Ann, 25)$ ,  $(Sue, 18)$ ,  $(Pete, 30)$ ,  $(Sarah, 20)$ ,  $(Ron, 27)$ ,  $(Julia, 22)$ ,  $(Bob, 18)$ ,  $(Luk, 23)$ ,  $(Tim, 25)$ . Further, assume that only one tuple fits in a block, and the memory holds at most 3 blocks.

- a. Show the runs created on each pass of the sort-merge algorithm, when applied to sort the Grades relation.
- b. What is the total number of block transfers ? Explain your answer.

**Exercise 5** Assume two relations  $r(A)$  and  $s(A)$  with  $r$  being stored in a sequential (ordered) file and  $s$  being stored in an unordered file on the disk. The block size is 2,000 Bytes, the tuple size 10 Bytes, and the cardinality is 800,000 tuples for both relations (assume identical instances). The values of the integer attribute  $A$  are uniformly distributed between 5 Mio. and 9 Mio. and they are unique in both relations. The disk performance is given as follows: latency time = 0.008 sec, seek time = 0.016 sec, transfer time = 0.001 sec.

- Determine the number of block IOs and the execution time for the following queries on the two relations:  
 Q1:  $\sigma_{A=6,000,000}(x)$   
 Q2:  $\sigma_{A<5,009,500}(x)$
- Determine the number of blocks at each level for a  $B^+$ -tree for each of the two relations. Each node contains 100 index entries and fills an entire block.
- Determine the number of block IOs and the execution time for Q1 and Q2 when the  $B^+$ -tree is used.

**Exercise 6** Consider a bank database with the relation  $Branch(BN, BC, A)$  (BN = branch-name, BC = branch-city, A = assets) and the following relational algebra expressions:

- $\pi_{T.BN}(\sigma_{T.A>S.A \wedge S.BC='Brooklyn'}(Branch/T \times Branch/S))$

Write an equivalent relational algebra expression which is more efficient. Justify your choice.

**Exercise 7** Let relations  $r_1(A, B)$  and  $r_2(A, C)$  have the following properties:  $r_1$  has 10,000 tuples,  $r_2$  has 30,000 tuples, and 50 tuples of  $r_1$  fit on one block, and 100 tuples of  $r_2$  fit on one block. The values of  $A$  are unique in both relations.

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

- Plan p1: Nested-loop join with  $r_1$  as outer relation
- Plan p2: Nested-loop join with  $r_2$  as outer relation
- Plan p3: Index nested-loop join using a  $B^+$ -tree index with  $r_1$  as outer relation. Each node (= block) contains 200 index entries.
- Plan p4: Index nested-loop join using a hash index with  $r_1$  as outer relation.

**Exercise 8** Given are two schedules  $S_1, S_2$  over three transactions  $T_1, T_2, T_3$ :

	$T_1$	$T_2$	$T_3$
S1:		read(A)	
	read(B)	write(A)	
			read(A)
	write(B)		write(A)
		read(B)	
		write(B)	

	$T_1$	$T_2$	$T_3$
S2:		read(A)	
	read(B)	write(A)	
		read(B)	read(A)
	write(B)		write(A)
		write(B)	
			write(A)

- Show the conflict graphs of  $S_1$  and  $S_2$ .
- Are  $S_1$  and  $S_2$  conflict serializable? (explain your answer)
- Are  $S_1$  and  $S_2$  view serializable? (explain your answer)

### Solution 1

1. The disk-arm movement.
2. A dense index has an index record for every search-key value. A sparse index has an index record for only some search-key values.
3. An index-sequential file is an ordered sequential file with a primary index.
4. Yes.
5. Step 1: Use index to find first tuple with  $A \geq v$ .  
Step 2: Scan relation sequentially from there.
6. After parsing the query and translating it to relational algebra.
7. Yes.
8. Increased processor and disk utilization.  
Reduced average response time for transactions.
9. Phase 1: Growing phase (transactions may only obtain locks)  
Phase 2: Shrinking phase (transactions may only release locks)
10. Deadlock prevention ensures that the system will never enter into a deadlock state.

### Solution 2

- a. File organization with reserved space method for variable-length records:

– Solution I: One record for each person

0	Jan	P1	800	P2	400	P3	500	P4	900
1	Ann	P1	250	⊥	⊥	⊥	⊥	⊥	⊥
2	Joe	P3	350	⊥	⊥	⊥	⊥	⊥	⊥

– Solution II: One record for each project

0	P1	Jan	800	Ann	250
1	P2	Jan	400	⊥	⊥
2	P3	Jan	500	Joe	350
3	P4	Jan	900	⊥	⊥

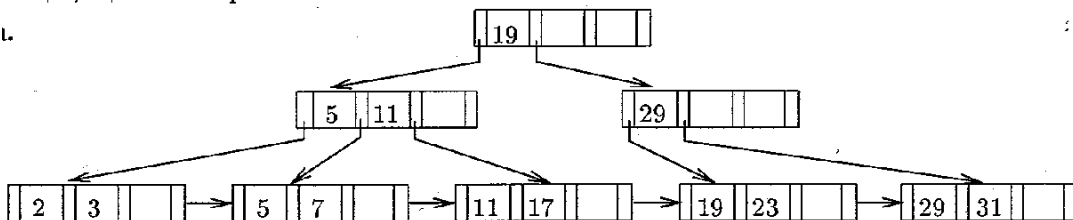
- b. Memory requirements:

- Solution I: 1 record =  $(20 + 4 \times 14) = 76$  Bytes, total =  $3 \times 76 = 228$  Bytes
- Solution II: 1 record =  $(10 + 2 \times 24) = 58$  Bytes, total =  $4 \times 58 = 232$  Bytes

- c. Unused disk space:

- Solution I:  $6 \times 14 = 84$  Bytes
- Solution II:  $2 \times 24 = 48$  Bytes

### Solution 3



## Solution 4

- a. In the following we use only the names to refer to the tuples.

Step 1: Create 4 sorted runs with 3 tuples each:

(Ann, Jan, John), (Pete, Sara, Sue), (Bob, Julia, Ron), (Luk, Tim)

Step 2: Merge pass that merges two runs into one run. Thus the number of runs decreases by the factor of 2:

(Ann, Jan, John, Pete, Sara, Sue), (Bob, Julia, Luk, Ron, Tim)

Step 3: The runs after the second merge pass are:

(Ann, Bob, Jan, John, Julia, Luk, Pete, Ron, Sara, Sue, Tim)

- b. Step 1:  $11 \times 2 = 22$  block transfers (read and write)  
Step 2:  $11 \times 2 = 22$  block transfers (read and write)  
Step 3:  $11 \times 1 = 11$  block transfers (only read)  
 $\Rightarrow 55$  block transfers

## Solution 5

- a.  $2,000/10 = 200$  tuples/block

$800,000/200 = 4,000$  blocks

$1 \text{ IO} = 0.008 + 0.016s + 0.001s = 0.025s$

Q1:  $\sigma_{A=6,000,000}(r)$ : Binary search

- Block IOs:  $\lceil \log_2 4,000 \rceil = 12$

- Time:  $0.025 \times 12 = 0.3$  sec

Q1:  $\sigma_{A=6,000,000}(s)$ : Sequential search

- Block IOs: on average read 2000 blocks (half of all) to find a unique value

- Time:  $0.025 \times 2000 = 50$  sec

Q2:  $\sigma_{A < 5,009,500}(r)$ : Sequential search

- Block IOs:

avg. distance between values:  $4Mio/800,000 = 5$

# of qualifying tuples:  $9,500/5 = 1,900$

# of qualifying blocks:  $\lceil 1,900/200 \rceil = 10$  block IOs

- Time:  $10 \times 0.025 = 0.25$  sec

Q2:  $\sigma_{A < 5,009,500}(s)$ : Sequential search

- Block IOs: 4000 blocks

- Time:  $0.025 \times 4,000 = 100$  sec

- b. Nodes (=index blocks): 100 index entries per node

Index blocks required at each level:

- level 3:  $\lceil 800,000/100 \rceil = 8,000$  blocks (leaf nodes)

- level 2:  $\lceil 8,000/100 \rceil = 80$  blocks

- level 1:  $\lceil 80/100 \rceil = 1$  block

$\Rightarrow 8,081$  index blocks are needed in total

The result is the same for  $r$  and  $s$

- c. Q1:  $\sigma_{A=6,000,000}(r)$ : B<sup>+</sup>-tree search

- Block IOs: 3 index blocks + 1 data block = 4 blocks

- Time:  $0.025 \times 4 = 0.1$  sec

Q1:  $\sigma_{A=6,000,000}(s)$ : the same as for  $r$

Q2:  $\sigma_{A < 5,009,500}(r)$ : Index makes no sense; hence the same result as in (a).

- Q2:  $\sigma_{A < 5,009,500}(s)$ : Follow leaf nodes in B<sup>+</sup>-tree
- Block IOs: Follow leaf nodes from the beginning to locate the data blocks
    - # of qualifying tuples (= # of qualifying key values): 1,900
    - # of qualifying blocks: max. 1,900 (each qualifying tuple in different block)
    - # of qualifying B<sup>+</sup>-tree nodes:  $\lceil 1,900/100 \rceil = 19$  (100 index entries/node)
  - Total:  $1,900 + 19 = 1,919$  block IOs
  - Time:  $(1,900 + 19) \times 0.025 = 47.975sec$

### Solution 6

$$\pi_{T.BN}(\pi_{T.BN,T.A}(Branch/T) \bowtie_{T.A > S.A} \pi_{S.A}(\sigma_{S.BC='Brooklyn'}(Branch/S)))$$

This expression reduces Branch/S to only those values corresponding to assets of branches in Brooklyn, which should be a small set. It also eliminates the unneeded attributes from Branch/T on the left-hand side. Finally, it replaces the Cartesian product by a theta-join.

### Solution 7

Tuples of  $r_1$ :  $n_{r_1} = 10,000$

Tuples of  $r_2$ :  $n_{r_2} = 30,000$

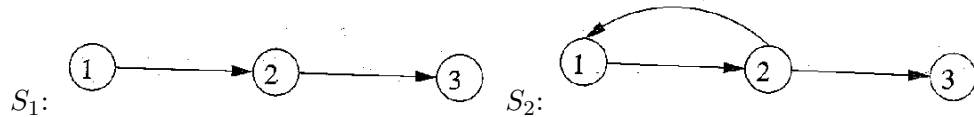
Blocks required for  $r_1$ :  $b_{r_1} = \lceil 10,000/50 \rceil = 200$  blocks

Blocks required for  $r_2$ :  $b_{r_2} = \lceil 30,000/100 \rceil = 300$  blocks

- a. Assume the worst case, where only one block of each relation fits in main memory:
  - $cost(p1) = n_{r_1} * b_{r_2} + b_{r_1} = 10,000 * 300 + 200 = 3,000,200$
- b. Assume the worst case, where only one block of each relation fits in main memory:
  - $cost(p2) = n_{r_2} * b_{r_1} + b_{r_2} = 30,000 * 200 + 300 = 6,000,300$
- c. Assume that only one block of  $r_1$  is in main memory and that for each tuple in  $r_1$  an index lookup in the B<sup>+</sup>-tree is performed:
  - Depth of B<sup>+</sup>-tree:  $2 \Rightarrow 2+1 = 3$  block IOs to get a tuple
  - $cost(p3) = b_{r_1} + n_{r_1} * (2 + 1) = 200 + 10,000 * 3 = 30,200$
- d. Assume that only one block of  $r_1$  is in main memory and that for each tuple in  $r_1$  a hash lookup is performed:
  - Cost of hash lookup: 1
  - $cost(p4) = b_{r_1} + n_{r_1} * 1 = 200 + 10,000 * 1 = 10,200$

### Solution 8

- a. Conflict graphs



- b.  $S_1$  is conflict serializable, since the conflict graph contains no cycles. It is equivalent to the serial schedule  $\langle T_1, T_2, T_3 \rangle$ .  
 $S_2$  is not conflict serializable, since there is a cycle in the conflict graph.
- c. Since  $S_1$  is conflict serializable, it is also view serializable.  
 $S_2$  is not view serializable, since both  $T_1$  and  $T_2$  read the initial value of  $B$  and modify  $B$ . Hence, it is impossible to create a serial schedule, where  $T_1$  and  $T_2$  read the initial value of  $B$ .