UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR11199 ADVANCED DATABASE SYSTEMS

Monday 10 th May 2021

13:00 to 15:00

INSTRUCTIONS TO CANDIDATES

Answer any TWO of the three questions. If more than two questions are answered, only QUESTION 1 and QUESTION 2 will be marked.

All questions carry equal weight.

This is an OPEN BOOK examination.

MSc Courses

Convener: A.Pieris
External Examiners: W.Knottenbelt, M.Dunlop, E.Vasilaki.

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. This question is about file organisations, query minimisation, and joins.

   (a) The contact tracing department of NHS keeps track of people who have tested positive for COVID-19. At the beginning of each day, the department creates a brand new file in their database. Each time a person tests positive, a new tuple containing a unique test ID, the person's date of birth, and the person's 11-digit phone number is inserted into the file. On any given day, the department expects over 30,000 positive tests. At the end of each day, the department decides to contact only **one** person with a given test ID.

   Given this specification, you need to decide on the file organisation and page layout that will optimise the performance of the tasks the contact tracing department must perform.

   one search operation hence clustered overhead is not necessary. Heap file (search may scan entire pages on worst case), sorted (inserts are very slow)

   i. The following four file organisations are available: heap file, sorted file on test ID, clustered index on test ID, and unclustered index on test ID. Choose one file organisation that would be most suitable for these tasks. Explain your choice. [*3 marks*]

   ii. The following three page layouts are available: fixed length (packed), fixed length (unpacked), and slotted page. Choose one page layout that would be most suitable for these tasks. Explain your choice. [*3 marks*]

fixed length(packed) since record size is fixed and no deletions are performed. Since other options come with overhead to support efficient delete operations and variable length records respectively.

   (b) Consider the conjunctive query

   $$Q(x, y) :- P(x, y, z), R(x, a), S(y, w, x), P(x, y, a), S(a, a, x), S(y, y, x), R(x, z)$$

   where x, y, z, w are variables, and a is a constant value.

   Is the conjunctive query Q minimal? If Q is minimal, explain why that is the case. If Q is not minimal, compute a minimal conjunctive query that is equivalent to Q, showing all the intermediate steps of the computation. [*5 marks*]

   (c) We would like to join two relations, R and S, on a common attribute A. The R relation is organised as a sorted file with 300 pages and with tuples sorted on A. There are no duplicate A-values in R. The S relation is organised as a heap file with 200 pages. The buffer pool consists of 21 pages, one of which holds the evolving output page. Ignore the cost of writing the join output to disk.

   i. What is the *minimum* number of disk I/O operations for a Block Nested Loops Join between R and S? Show how you computed that number. [*3 marks*]

   ii. What is the *minimum* number of disk I/O operations for a Sort Merge Join between R and S? Show how you computed that number. [*3 marks*]

QUESTION CONTINUES ON NEXT PAGE

(d) We would like to join three relations, R, S, and T, on a common attribute. Instead of joining two relations at a time, we want to modify the Block Nested Loops Join algorithm to join three relations at a time.

Let P denote the number of buffer pages allocated to the outermost relation.

i. Write pseudocode of the modified Block Nested Loops Join algorithm. [*2 marks*]

ii. Assume that each of the three relations consists of N pages, and the buffer pool consists of B pages, one of which holds the evolving output page. Express the I/O cost of this algorithm in terms of N, B, and P. [*3 marks*]

iii. Continuing the previous question, assume that $N = 1000$ and $B = 102$. *Roughly* how many buffer pages should be allocated to the outermost relation such that the I/O cost of this algorithm is minimised? You can simplify your analysis by assuming $\lceil x/y \rceil = x/y$. Justify your answer. [*3 marks*]

2. This question is about query optimisation, indexing, and static query analysis.

(a) Consider a database consisting of relations Students, Courses, and Enrolled.

Relation	Cardinality	Number of pages	Primary key
`Students(sid, sname, dept)`	1000	100	`sid`
`Courses(cid, cname, level)`	80	20	`cid`
`Enrolled(sid, cid, mark)`	5000	1000	`(sid,cid)`

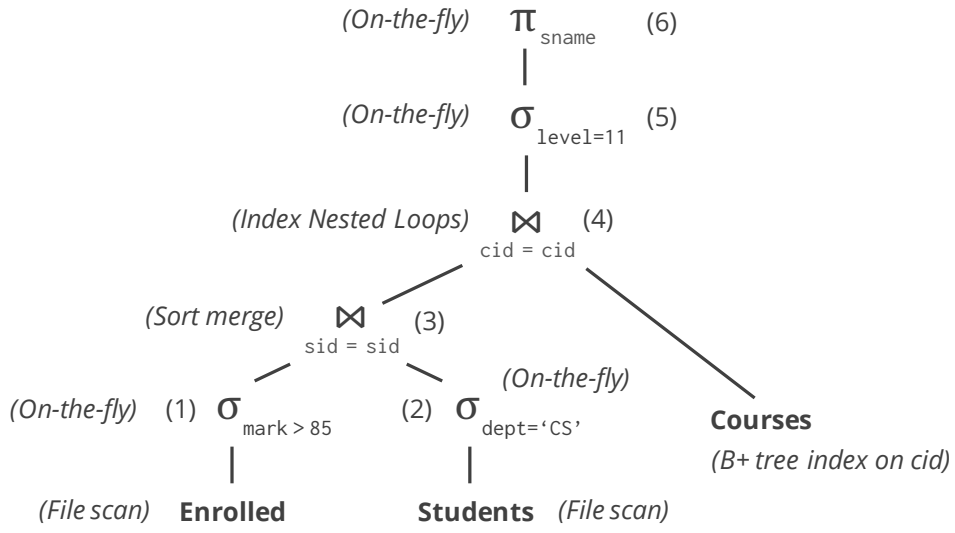
Consider the following SQL query that finds all students from the CS department who have a mark greater than 85 on a level-11 course.

```
SELECT S.name
  FROM Students S, Enrolled E, Courses C
 WHERE S.sid = E.sid AND E.cid = C.cid
   AND S.dept = 'CS' AND E.mark > 85 AND C.level = 11
```

Assume that:

- There are 25 different departments. Course marks are integers between 1 and 100 (both inclusive). There are 5 course levels (7, 8, 9, 10, 11).
- Enrolled.sid and Enrolled.cid are foreign keys that reference Students.sid and respectively Courses.cid. Attributes of primary keys are underlined.
- There is a variant B unclustered B+ tree index on Courses.cid. All index (leaf and non-leaf) pages are already in memory, outside the buffer pool.
- The buffer pool has 100 buffer pages.

The query optimiser is currently considering the below physical query plan.



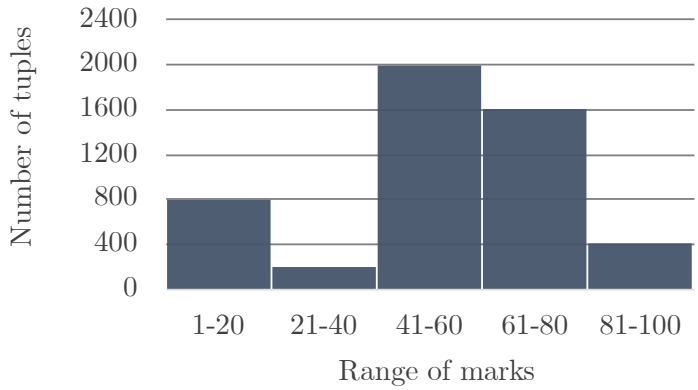
QUESTION CONTINUES ON NEXT PAGE

The query optimizer assumes even distributions of values in columns and independence between values in different columns, unless stated otherwise.

i. What is the estimated number of tuples in the output of operator 5? Explain how you computed the answer. [*3 marks*]

ii. What is the total estimated disk I/O cost of this query plan? Ignore the cost of writing the final result. Explain how you computed the answer. [*5 marks*]

iii. Suppose now that the query optimiser has access to the below histogram that shows the distribution of the mark values in the Enrolled relation.



What is then the total estimated disk I/O cost of this query plan? Ignore the cost of writing the final result. Explain how you computed the answer. [*3 marks*]

(b) Consider a B+ tree index of height $h = 3$ and order $d = 4$ with index entries stored using variant A. Assume that the index (search) key is the primary key of the relation for which the index is built. Note that index pages include both leaf and non-leaf pages.

i. What is the largest number of *leaf pages* that can exist in this index? Show how you computed that number. [*2 marks*]

ii. What is the smallest number of *leaf pages* that can exist in this index? Show how you computed that number. [*2 marks*]

iii. In the *worst case*, what is the largest number of *index pages* that will be *read* during a range search that returns exactly 7 tuples? Explain your answer. [*2 marks*]

iv. In the *worst case*, what is the largest number of *index pages* that will be *written to* during an insert? Consider both existing index pages and index pages created as part of the insert. Explain your answer. [*3 marks*]

QUESTION CONTINUES ON NEXT PAGE

(c) Consider the Boolean conjunctive query

$$Q :- R(x_1, x_2, x_3), P(x_2, x_4, x_5, x_1), R(x_5, x_6, x_7), S(x_8, x_7), S(x_3, x_9)$$

where $x_1, x_2, \ldots, x_9$ are variables.

i. Give the hypergraph $H(Q)$ of the conjunctive query Q. [*2 marks*]

ii. Is $H(Q)$ acyclic? If yes, then give a join tree of $H(Q)$; otherwise, prove that $H(Q)$ is indeed not acyclic. [*3 marks*]

3. This question is about query evaluation, distributed transactions, and hashing.

(a) Consider the database

$$\begin{aligned} D \;=\; \{&R(a,a), R(b,a), R(c,a),\\ &P(a,b,a), P(b,a,a), P(b,c,a), P(c,d,a), P(c,e,a),\\ &S(a,a,b), S(b,b,a), S(c,c,b), S(d,d,c), S(e,e,c)\} \end{aligned}$$

and the conjunctive query

$$Q(y) :- R(x,a), P(x,y,a), S(y,y,x)$$

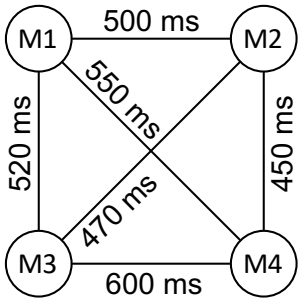
where x, y are variables, and a is a constant value.

i. Compute the set of tuples $Q(D)$. [*2 marks*]

ii. For each tuple t of $Q(D)$, provide the match of Q in D that witnesses the membership of t in $Q(D)$. [*3 marks*]

(b) A database runs on 4 machines and uses Two-Phase Commit (2PC). Machine 1 is the coordinator, while Machines 2, 3, and 4 are participants.

The machines are connected as in the graph below, with edge labels representing send times between two machines. For example, sending a message between Machine 1 and Machine 4 takes 550 milliseconds in either direction.



Each machine takes 20 milliseconds to generate and flush a log record. There is no need to re-flush log records if they already exist. Assume that everything else is instantaneous (e.g., processing time, recovery).

QUESTION CONTINUES ON NEXT PAGE

The coordinator starts 2PC for a transaction at 0 ms (the start time of the transaction). Assume that all participants vote 'yes'.

i. How long does the whole 2PC process take for a successful commit? [*2 marks*]

ii. What messages does the coordinator send and at what times? [*2 marks*]

iii. What messages does Machine 2 send and at what times? [*2 marks*]

iv. Consider a scenario where Machine 4 crashes at 1300 ms and recovers at 2000 ms. What is the earliest time at which the coordinator can finish the whole 2PC process? [*3 marks*]

v. Consider a different scenario where the coordinator crashes at 2000 ms and recovers at 3000 ms. What is the earliest time at which the coordinator can finish the whole 2PC process? [*3 marks*]

(c) Consider the `Users(uid, name, age)` relation consisting of 100 pages with 100 tuples per page. Each tuple has a unique `uid` value in the range $[0, 9999]$. We want to build a static hash index with chaining on `uid`, with index entries stored using variant A. The index consists of pages stored on a disk.

i. Consider a perfectly uniform hash function h_p that maps `uid` values to hash values in the range $[0, 19]$ (i.e., each hash value has the same number of collisions).
For the hash index constructed using h_p, compute the minimum, maximum, and expected number of disk I/O operations performed while searching for an existing user given a randomly selected `uid`. [*4 marks*]

ii. Consider now the following hash function h_d: the hash value $h_d(k)$ is computed as the number of digits in the base 10 representation of the integer k; for example, $h_d(13) = 2$ and $h_d(901) = 3$.
For the hash index constructed using h_d, compute the minimum, maximum, and expected number of disk I/O operations performed while searching for an existing user given a randomly selected `uid`. [*4 marks*]