Xi'an Jiaotong-Liverpool University

Question 1. Answer the following questions on indexing in database systems.

[25 marks]

A relation called *employee has* 20,000 tuples. The tuples are stored as fixed length and fixed format records; each has the length of 250 bytes. Tuples contain the key attribute *name* with length of 20 bytes. The tuples are stored sequentially in a number of blocks, ordered by *name*. Each block has the size of 4,096 bytes (i.e., 4 Kilobytes) and each tuple is fully contained in one block. Suppose that a primary index using B+tree on the *name* attribute, and a secondary index using extendable hashing on the non-key attribute *hometown* (stores the city where an employee is from) are to be created. The length of each index entry is 30 bytes: 20 bytes for the search key and 10 bytes for the pointer to data (an 8 byte block id and a 2 byte offset). Each index entry is also fully contained in one block.

a) Compute the number of disk blocks needed to store the relation *employee*.

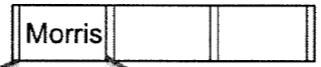
[4/25]

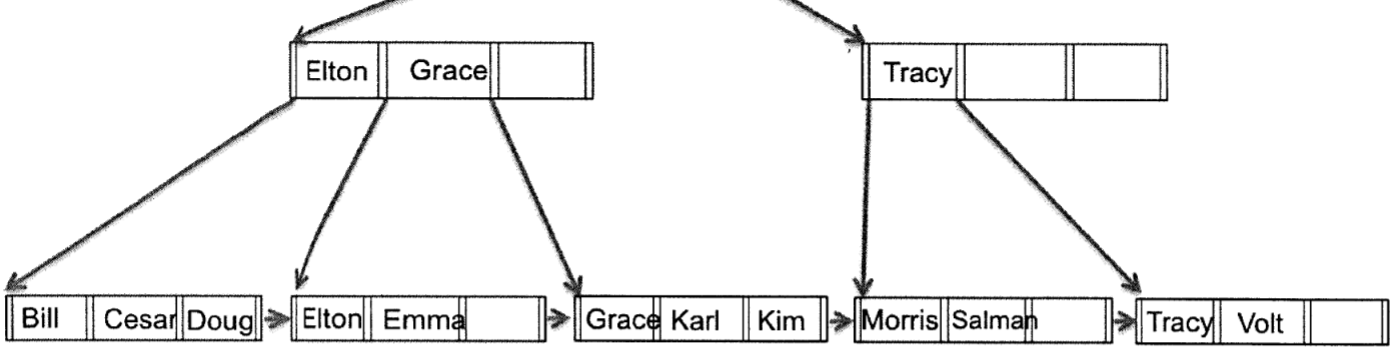
b) If the primary index on *name* is sparse, i.e., one index entry for one block, compute the number of blocks needed to store the primary index.

[4/25]

c) The diagram below shows the B+ tree during the index creation. The number of pointers in a node is 4. Describe the query process for the search key "Emma".

[2/25]





d) Based on B+ tree in Question 1.c), draw the tree after inserting two search keys "Amy" and "Linda".

[6/35]

e) Deletion in the B+ tree might cause nodes to become underfull. What are the two strategies for restoring the B+ tree properties? Briefly state under what circumstances these two strategies need to be applied.

[6/25]

f) Is a sparse secondary index on the *hometown* attribute useful? Why?

[3/25]

Xi'an Jiaotong-Liverpool University

Question 2. Consider the following three relations:

i) *author (authorID, name, affiliation, email)*
ii) *publishes (articleID, authorID)*
iii) *article (articleID, title, journal, year, publisher)*

"*authorId*" and "*articleID*" are the primary keys for relations *author* and *article*, respectively. *authorID* is the join attribute for *author* and *publishes*. *articleID* is the join attribute for *publishes* and *article*. Relation *author* has 5,000 tuples stored on 500 blocks. Relation *publishes* has 100,000 tuples stored on 100 blocks. Answer the following questions.

[25 marks]

a) Describe how the selection $\delta_{year<2014}$ on relation *article* can be evaluated in the most cost effective way by using a primary index on year and a secondary index on year, respectively.

[4/25]

b) Assume that the memory can only hold one block for each relation. Using the blocked nested loop join algorithm and *publishes* as the outer relation, how many block transfers and seeks are needed to compute "*author* $\bowtie$ *publishes*", respectively?

[4/25]

c) Assume that a B+ tree index with the number of pointers in a node, $N=10$, is built for *author* on the attribute *authorID*. How many block transfers and seeks are needed to compute "*author* $\bowtie$ *publishes*" using the indexed nested loop join, respectively?

[6/25]

d) If both relations have been sorted and the merge join algorithm is used to evaluate "*author* $\bowtie$ *publishes*". Assume that the buffer for input and output, $b_b=1$; also, it takes $t_T=0.1ms$ for a block transfer and $t_S=4ms$ for a seek. Compared to the results from Question 2.b) and 2.c), which algorithm is the most cost effective in terms of total time for block transfers and seeks? Justify your answer.

[7/25]

e) Briefly outline the procedure of performing a hash join.

[4/25]

Xi'an Jiaotong-Liverpool University

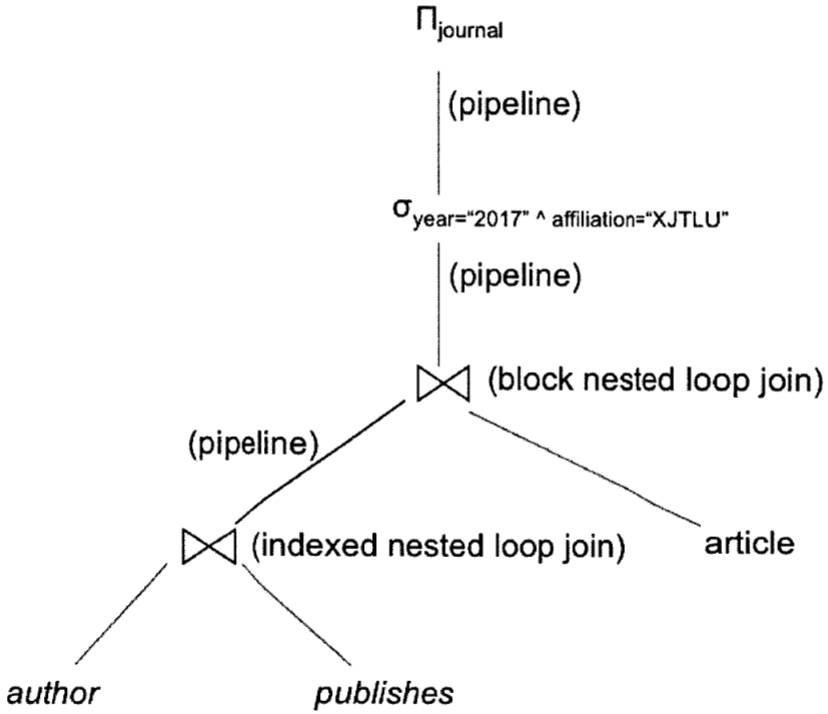
Question 3. Consider the three relations in Question 2 and their catalog information. "*authorId*" and "*articleID*" are the primary keys for relations *author* and *article*, respectively. The two attributes in *publishes* are the foreign keys referencing *author* and *article*, respectively.

i) author (authorID, name, affiliation, email)
ii) publishes (articleID, authorID)
iii) article (articleID, title, journal, year, publisher)

- number of records in *author*, n_{author} = 5,000;
- number of blocks in *author*, b_{author} = 500;
- number of records in *publishes*, $n_{publishes}$ = 100,000;
- number of blocks in *publishes*, $b_{publishes}$ = 100;
- number of records in *article*, $n_{article}$ = 20,000;
- number of blocks in *article*, $b_{article}$ = 2,000;
- index: a primary B^+-tree index of height 3 on the *authorID* attribute of *publishes* relation;
- number of distinct values for the attribute *affiliation* in the *author* relation, *V(author, affiliation) = 500;*
- the number of distinct values for the attribute *year* in the *article* relation, *V(article*

year) = 10;

A query evaluation plan is shown below.



Xi'an Jiaotong-Liverpool University

Answer the following questions.

[25 marks]

a) What is the relational algebra expression represented by the query tree?

[4/25]

b) Based on the given information, what is the estimated size of the join operation "*author* ⋈ *publishes*"?

[4/25]

c) One of the heuristic rules for query optimisation is to perform selection operations as early as possible. In the above query tree, the selection predicate *affiliation*="XJTLU" can be performed on relation *author* first and *year*="2017" can be performed on relation *article* first. Write the equivalent algebra expression based on this heuristic rule and the equivalence rule for algebra expressions.

[4/25]

d) Based on the answer from Question 3.c), what is the estimated size of the operation "$\sigma_{affiliation="XJTLU"}(author)$ ⋈ *publishes*"?

[5/25]

e) Assume that linear scan is used to evaluate all the selection operations in the plan. What is the total number of block transfers for the optimised evaluation plan from Question 3.c)? Note that no intermediate relations need to be saved as the result of using pipelining.

[8/25]

Xi'an Jiaotong-Liverpool University

Question 4. Answer the following questions related to transaction, concurrency and recovery in database systems.

[25 marks]

a) What is meant by "a schedule is conflict serialisable?

[3/25]

b) Draw the precedence diagram for the schedule below and determine if it is conflict serialisable.
Schedule: T1:write(A); T1:read(B); T1:read(C); T3:write(B); T1:write(B); T3:write(C); T2:read(C); T2:write(B); T2:write(C); T3: read(A)

[5/25]

c) Is the following schedule view serialisable? If yes, what serial schedule is it equivalent to? Justify your answer.

T1	T2	T3
Read(A)		
	Write(A)	
Write(A)		
		Write(A)

[4/25]

d) What is meant by "a schedule is recoverable"?

[3/25]

e) Is the following schedule recoverable? Justify your answer.
Schedule: T1:write(X); T1:write(Y); T2:read(X); T2:write(Y); T2:abort; T1:write(Z); T1:commit; T3:read(Y); T3:write(Z); T3:commit.

[4/25]

f) Consider the following transaction logs. When recovering from the system crash, in the redo pass, which operations need to be redone? In the undo pass, which operations need to be undone? Justify your answer.

Start of the logs
<T1 start>
<T1, B, 900, 1000>
<T2 start>
<T1 commit>
<T4 start>
<T4, B, 1000, 1700>
<T5 start>
<T2, D, 10, 20>
<T3 start>
<checkpoint L{…}>
<T5, C, 300, 100>
<T5 commit>
<T2 commit>
<T6 start>
<T6, A, 700, 600>
<T4, B, 1000>
<T4 abort>
<T3, D, 20, 19>
←System crash, start recovery

Question 5. Answer the following questions.

[25 Marks]

a) The two-phase commit protocol is one of the most widely used protocols in distributed database systems to ensure transaction atomicity. Describe how the protocol works in distributed database systems.

[4/25]

b) Assume that the transaction coordinator fails during execution of the two-phase commit protocol in distributed database systems, how can the status of the transaction be determined based on the logs?

[4/25]

c) Consider the three relations in Question 2. Assume that they are stored in a distributed database at two sites:

XJTLU site:

author

publishes

IEEE site:

article

To answer the query "*find the names of XJTLU staff and titles of articles that were published by them*", one needs to compute the join of the three relations. It is known that only a small number of articles stored at the IEEE site were published by authors

stored at the XJTLU site. Assume that a query is made to the XJTLU site; describe how semi-join can be used to optimise the query (e.g., reduce cost).

[6/25]

d) Can the centralised deadlock detection approach be used to detect deadlock in distributed database system? If yes, describe how it works. If no, explain why.

[5/25]

e) Consider the following transaction table. Columns represent the items and rows represent transactions. Each cell represents whether an item is purchased in a transaction, "1" means yes and "0" means no. Compute the following:

(1) support (A)

(2) support (A->>C)

(3) confidence (A->>C)

ID\Item	A	B	C	D	E	F
T1	1	1	1	0	0	0
T2	1	0	1	0	0	0
T3	1	0	0	1	0	0
T4	0	1	0	0	1	1

[6/25]

END OF EXAM PAPER