Xi'an Jiaotong-Liverpool University

西交利物浦大學

PAPER CODE	EXAMINER	DEPARTMENT	TEL
CSE201	[redacted]	Computer Science & Software Engineering	[redacted]

FIRST SEMESTER 2019/2020 FINAL EXAMINATIONS

Undergraduate – Year 3

DATABASE DEVELOPMENT AND DESIGN

TIME ALLOWED: 2 Hours

INSTRUCTIONS TO CANDIDATES

1、 This is a closed-book examination, which is to be written without books or notes.

2、 Total marks available are 100.

3、 This exam consists of four questions. Answer all questions. There is NO penalty for providing a wrong answer.

4、 Answer should be written in the answer booklet(s) provided.

5、 Only English solutions are accepted.

6、 All materials must be returned to the exam supervisor upon completion of the exam. Failure to do so will be deemed academic misconduct and will be dealt with accordingly.

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

[25 marks]

A relational database holds two relations: *professor (ID, name, email)* and *researchInterest (ID, researchTopic)* with the following information:

Relation *professor*:

- Tuples are stored as fixed-format, fixed-length records, each of 400 bytes.
- There are 5,000 tuples.
- Each tuple contains a key attribute *ID* of length 20 bytes; other fields and the record header make up the rest.

Relation *researchInterest*:

- Tuples are stored as fixed-format, fixed-length records, each of 100 bytes.
- There are 25,000 tuples.
- Tuples contain attribute *researchInterest.ID*, referencing *professor.ID*.

Assume that the size of one block is 4-kilobyte (4,096 bytes), and no tuple spans over more than one block. Each professor can specify up to five research interests, and space is allocated even if a professor specifies less than five interests. Tuples in *professor* are sequentially ordered by *ID*. The "clustered disk organisation" strategy is used, which means that, for each professor tuple, the five tuples of *researchInterest* also reside in the same block.

a) How many disk blocks are needed to store the two relations? Justify your answer.

[3/25]

b) A sparse primary index is created on *ID* for the *professor* relation, i.e., one index entry for each block. Each index entry includes a search key value and a 30-byte pointer to the tuples on disk. How many disk blocks are needed to store the index? Justify your answer.

[3/25]

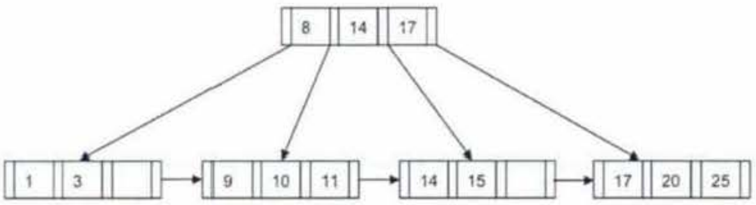
c) Suppose that a dense primary index is also created on *ID* for the *professor* relation, i.e., one index entry for each tuple. Each index entry includes a search key value and a 30-byte pointer to the tuples on disk. How many disk blocks are needed to store the index? Justify your answer.

[3/25]

d) An initial B+ tree index (N=4) on *professor.ID* is shown below. Draw the B+ trees after the following sequential operations: (1) insert 11; (2) insert 13; (3) delete 14; (4) delete 15; and (5) delete 13. Each subsequent operation must be performed based on the result of the previous operation.

[10/25]

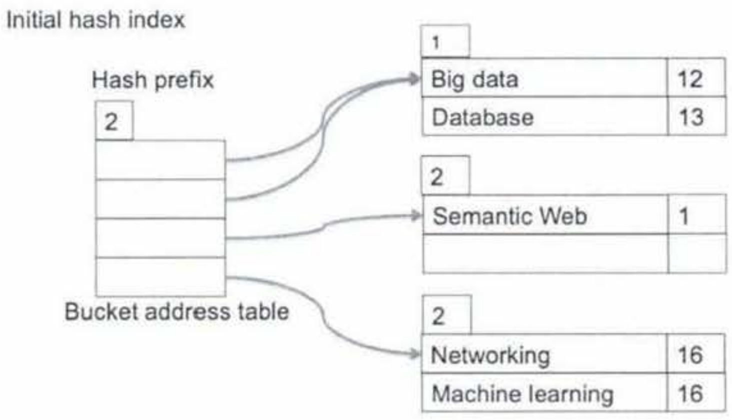
Initial tree



e) An extendable hash index on the *researchTopic* of the relation *researchInterest* is shown below. To simplify the problem, assume that each bucket can only contain up to two tuples. Based on the hash values in the table below, draw the hash indices after insertion of the following tuples (tuple in the form "*researchTopic, ID*"): (1) "Machine Leaning, 20"; and (2) "Security, 33".

[6/25]

Search key	Hash value
Big data	0001 1001 0000 1111
Database	0101 1001 0101 1100
Operating system	0011 1011 0000 1100
Machine learning	1100 1001 0101 1010
Networking	1110 0001 0101 1110
Semantic Web	1010 0101 1111 0010
Security	0101 1111 0101 0100



Question 2. Consider the following two relations and their catalogue information.

i) *customer(<u>customer_Number</u>, name, phone, address)*

ii) *transaction(<u>transaction_ID</u>, date, amount, customer_Number)*

The "*customer_Number*" is the key for the *customer* relation, and "*transaction_ID*" is the key for the *transaction* relation. The *transaction.customer_Number* is the foreign key referencing the *customer* relation. The *customer* relation has 5,000 tuples stored in 500 blocks, and the *transaction* relation contains 80,000 tuples stored in 7,000 blocks. Assume that the relation *transaction* has been sequentially sorted by the key attribute. Answer the following questions.

[25 marks]

a) Suppose that the linear search is used to evaluate the selection $\delta_{amount>200}$ in the *transaction* relation, how many block transfers would be needed? How many seeks would be needed?

[2/25]

b) How to use a B+Tree index on the *amount* attribute to evaluate $\delta_{amount>200}$ in the *transaction* relation in an efficient way?

[2/25]

c) Suppose that a sparse primary B+ tree index ($N=7$) has been created for the *transaction* relation on the key attribute. How many block transfers are needed for the selection $\delta_{transaction\_ID\ =7,360}$? How many seeks are needed? (Hint: in a sparse index, an index entry contains one pointer to each block)

[4/25]

d) Assume that none of the relations can fit in memory, time for one block transfer $t_T=0.1ms$, and time for one seek $t_S=4ms$. Which of the two algorithms, the block nested loop join and indexed nested loop join (using the sparse primary B+ tree index in Question 2.c), is more efficient to evaluate "*customer* $\bowtie$ *transaction*"? Justify your answer.

[9/25]

e) To sort the *customer* relation on the *customer_Number* attribute using external sort merge algorithm, assume that the memory size $M=50$ and the buffer for reading and writing $b_b=3$, how many block transfers are needed? How many seeks are needed?

[4/25]

f) Assume that the hash join algorithm is used to evaluate "*customer* $\bowtie$ *transaction*", the number of partitions, $n_h=30$, and the size of the buffer for reading and writing, $b_b=3$. How many block transfers are needed? How many seeks are needed?

[4/25]

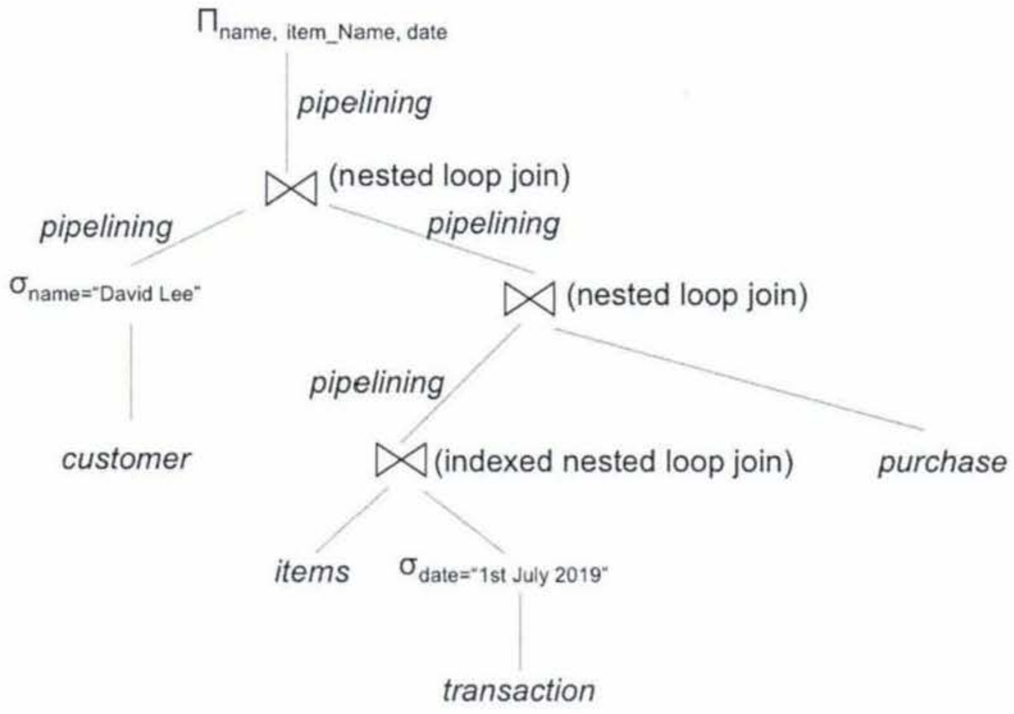
Question 3. Consider the following four relations and their catalog information.

transaction(<u>tID</u>, amount, date)
customer(<u>cID</u>, name, phone)
purchase(tID, cID)
items(tID, itemName, quantity)

where *tID* is the key for *transaction*, and *cID* is the key for *customer; purchase.tID* and *items.tID* are the foreign keys referencing *transaction; purchase.cID* is the foreign key referencing *customer*.

- Number of tuples in *transaction* is 20,000 stored in 2,000 blocks.
- Number of tuples in *customer* is 500 stored in 50 blocks.
- Number of tuples in *purchase* is 20,000 stored in 1,000 blocks.
- Number of tuples in *items* is 100,000 stored in 10,000 blocks.
- Number of distinct values for the attribute *date* in the *transaction* relation, *V(transaction, date) = 180.*
- Index: a four-level primary B^+-tree index (height=4, on disk) on the *tID* attribute of *items* relation.

Assume that all the selection operations are performed using linear scan, and one transaction contains at most 10 items. Consider the following annotated query evaluation tree and answer the questions below.


$\Pi_{\text{name, item\_Name, date}}$
pipelining
$\bowtie$ (nested loop join)
pipelining
pipelining
$\sigma_{\text{name="David Lee"}}$
$\bowtie$ (nested loop join)
pipelining
customer
$\bowtie$ (indexed nested loop join)
purchase
items
$\sigma_{\text{date="1st July 2019"}}$
transaction

[25 marks]

a) Write the algebra expression for the given annotated query evaluation tree.

[4/25]

b) Based on the given catalog information, what is the estimated size of the selection $\sigma_{date=\text{"1st July 2019"}}(transaction)$?

[4/25]

c) Based on the given catalog information, what is the estimated size of the join $\sigma_{date=\text{"1st July 2019"}}(transaction) \bowtie items$? Justify your answer.

[5/25]

抄格式

d) Based on the given catalog information, how many block transfers are needed to evaluate the join $\sigma_{date=\text{"1st July 2019"}}(transaction) \bowtie items$?

[6/25]

e) Based on the given catalog information, how many block transfers are needed for the whole evaluation plan?

[6/25]

Question 4. Answer the following questions.

[25 marks]

a) Draw a precedence diagram for the schedule below and determine if the schedule is conflict serialisable.

T1	T2	T3	T4
write(X);			
write(Y);			
	read(X);		
	write(Y);		
	read(Z);		
write(Z);			
commit;			
		read(Y);	
	commit;		
		write(Z);	
			read(Z);
		commit;	
			abort.

[3/25]

b) Is the schedule in Question 4.a recoverable? Justify your answer.

[3/25]

c) Is the schedule in Question 4.a cascadeless? Justify your answer.

[3/25]

d) Consider the following database logs for failure recovery. (1) Which transactions are in the un-do list at the checkpoint? (2) Which transactions need to be redone? (3) Which transactions need to be undone?

Time	**T1**	**T2**	**T3**
0			start
1	start		
2	read(X)		
3	------------------------Checkpoint---------------------		
4			read(Y)
5			Y=Y/3
6	X=X*2		
7		start	
8		read(Y)	
9		read(X)	
10		Y=Y+X	
11			write(Y)

12			commit
13		write(Y)	
14		X=X-10	
15	-------------------------Database Failure---------------------		

[3/25]

e) Describe how the bloom-join technique is used to reduce the cost of join operation between two distributed relations.

[4/25]

f) Describe how the two-phase commit protocol works in distributed database systems.

[6/25]

g) Briefly describe how data is stored in wide column data stores.

[3/25]

END OF EXAM PAPER