Xi'an Jiaotong-Liverpool University

西交利物浦大学

Module Code	Examiner	Email of Examiner	Tel
CSE201			

1st SEMESTER 2019/20 Remote Open-Book Resit Exam

Undergraduate – Year 3

DATABASE DEVELOPMENT AND DESIGN

Exam Duration: *2 Hours*

Crash Time Allowed: *30 Minutes*

INSTRUCTIONS TO CANDIDATES

1、 This is a remote open-book exam. Please tick the integrity disclaimer *when uploading your answers on ICE* and complete the assessment independently and honestly.

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、 Only English solutions are accepted. *Answers need to be typed in a word document and submitted as a PDF via ICE.*

5、 The duration is *2 hours*, and an additional *30-minute* crash time beyond the exam duration will be allowed for you to report and resolve minor technical issues which may be encountered during the exam. Where there are any major problems preventing you from continuing the exam or submitting your answers in time, please do not hesitate to email the Module Examiner or Assessment Team of Registry.

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

[25 marks]

A relation called *student has* 30,000 tuples. The tuples are stored as fixed length and fixed format records; each has the length of 200 bytes. Tuples contain the key attribute *id* with length of 20 bytes. The tuples are stored sequentially in a number of blocks, ordered by *id*. 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 *id* attribute is created. The length of each index entry is 40 bytes: 20 bytes for the search key value and 20 bytes for the pointer to data. Each index entry is also fully contained in a block.

a) Compute the number of disk blocks needed to store the relation *student*. Justify your answer.

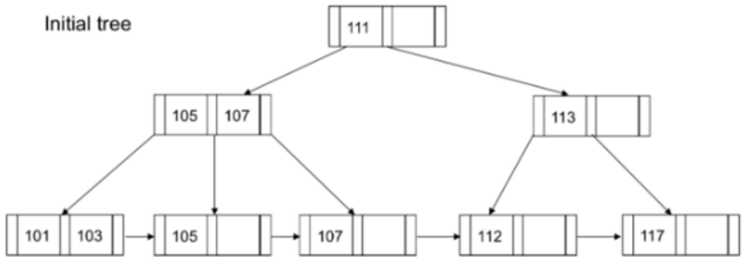
[4/25]

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

[4/25]

c) The diagram below shows the initial B+tree (N=3) on the *id* attribute for relation *student*. Draw the B+ trees after the following operations: (1) insert 102; (2) insert 104; and (3) delete 112. Operation must be performed based on the result of the previous operation.

[9/25]

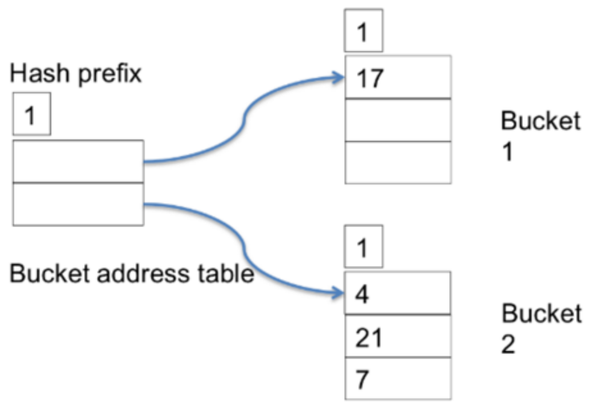


d) Suppose that the relation *student* also contains a non-key attribute, *city* (i.e. where a student is from). Is a sparse B+ tree index on the *city* attribute useful? Why?

[2/25]

e) Suppose an extendable hash index is created on the *city* attribute of *student*. Values of cities are represented as integers, e.g. 4 for London and 17 for Suzhou. A hash function, *h(x) = x mod 8,* is used (thus, the hash values are represented as 3-bit integers). To simplify the problem, assume that each bucket can only contain up to three tuples. With the initial hash index below, draw the hash indices after operations (1) insert 13; and (2) insert 29.

[6/25]


Hash prefix
1
Bucket address table
1
17
Bucket 1
1
4
21
7
Bucket 2

Question 2. Consider the following two relations:

i) *customer(customer_ID, customer_Name, address, phone, account_Number)*

ii) *account(account_Number, branch_Name, balance)*

The *customer_ID* and *account_Number* are the candidate keys for the relations *customer* and *account*, respectively. The *customer.account_Number* is the foreign referencing *account*. Tuples in *account* are sequentially ordered by *account_Number*. The number of tuples in *customer* is 60,000 and the number of blocks is 3,000. The number of tuples in *account* is 70,000 and the number of blocks is 2,000. Answer the following questions.

[25 marks]

a) Using linear scan to evaluate $\delta_{account\_Number=1337}$ in relation *account*, how many block transfers would be needed? How many seeks would be needed?

[2/25]

b) Using binary search to evaluate $\delta_{account\_Number=1337}$ in relation *account*, how many block transfers are needed? How many seeks are needed.

[2/25]

c) Describe how to use a B+ tree index on the *balance* attribute to evaluate $\delta_{balance>7,000}$?

[3/25]

d) Using *customer* as the outer relation, and the block nested-loop join algorithm to evaluate the natural join *account*$\bowtie$*customer*, how many block transfers would be needed? How many seeks would be needed?

[6/25]

e) Assume that the memory size M is 50 blocks and four blocks are used to buffer the input and output, i.e., $b_b=4$. To sort the *customer* relation based on *account_Number* using external sort merge algorithm, how many block transfers would be needed? How many seeks would be needed?

[6/25]

f) Assume that both relations are physically sorted, to evaluate the natural join *account*$\bowtie$*customer* using the merge join algorithm with $b_b=4$, how many block transfers would be needed? How many seeks would be needed?

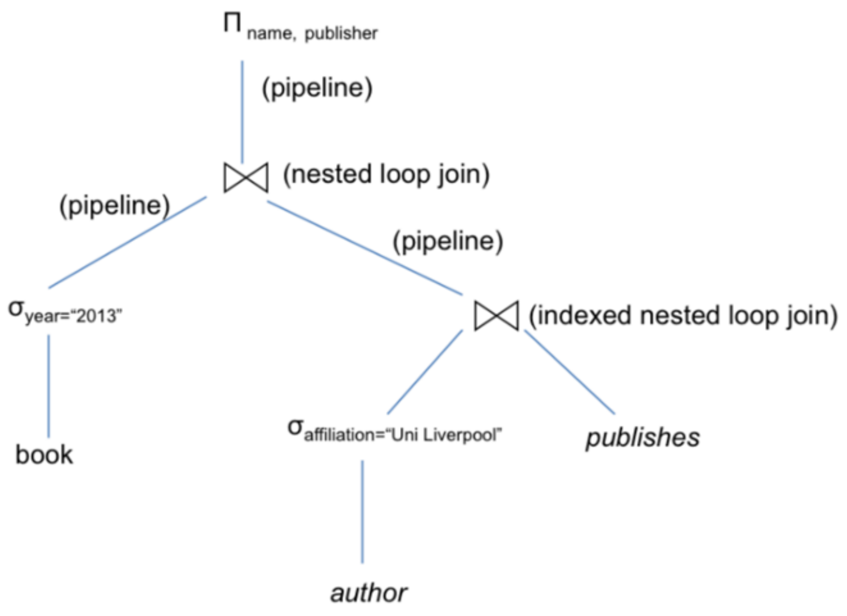
[6/25]

Question 3. Consider the three relations below and their catalog information. "*authorId*" and "*ISBN*" are the keys for relations *author* and *book*, respectively. The two attributes in *publishes* relation are the foreign keys referencing *author* and *book*, respectively.

i) *author (authorID, name, affiliation, email)*
ii) *publishes (ISBN, authorID)*
iii) *book (ISBN, title, edition, year, publisher)*

- Number of tuples in *author* is 10,000, stored in 1,000 blocks.
- Number of tuples in *publishes* is 50,000, stored in 2,000 blocks.
- Number of tuples in *book* is 40,000, stored in 4,000 blocks.
- Index: a secondary 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) = 1,000;*
- Number of distinct values for the attribute *year* in the *book* relation, *V(book, year) = 50;*

An annotated query evaluation tree is shown below.



Assume that linear scan is used to evaluate all the selection operations, and one author has 5 published books on average. Answer the following questions.

[25 marks]

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

[4/25]

b) Based on the given information, what is the estimated size of the selection $\sigma_{year=\text{"2013"}}(book)$?

[4/25]

c) Based on the given information, what is the estimated size of the operation "$\sigma_{affiliation=\text{"Uni Liverpool"}}(author) \bowtie publishes$"? Justify your answer.

[4/25]

d) Based on the given information, how many block transfers are needed to evaluate "$\sigma_{affiliation=\text{"Uni Liverpool"}}(author) \bowtie publishes$"? Justify your answer. Note that the indexed nested loop join algorithm is used.

[7/25]

e) How many block transfers are needed for the whole evaluation plan? Justify your answer. Note that no intermediate relations need to be saved if pipelining is used.

[6/25]

Question 4. Answer the following questions.

[25 marks]

a) Relational database systems must maintain four properties for transactions: atomicity, consistency, isolation and durability. Provide a short practical example for each of the four properties.

[4/25]

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

T1	T2	T3	T4
read(A)			
read(B)			
write(B)			
	read(B)		
		write(A)	
	read(C)		
			read(A)
			read(D)
			write(D)
	read(D)		

[4/25]

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

[4/25]

d) Is the schedule in Question 4.c recoverable? Justify your answer.

[3/25]

e) Consider following database logs and assume that database failure happens immediately after <T3 abort>. With the log-based recovery algorithm, (1) which transactions should be in the un-do list at the 2nd checkpoint? (2) Which transactions need to be redone? (3) Which transactions need to be undone?

Beginning of log
…
<checkpoint {T6}>
<T1 start>
<T1, B, 3000, 2050>
<T2 start>
<checkpoint {?}>
<T2, C, 300, 500>
<T2 commit>
<T3 start>

<T3, A, 1000, 950>
<T1, B, 3030>
<T3 abort>
Database failure

[3/25]

f) Assume that two distributed relations *r* and *s* are stored at London and Shanghai, respectively. Explain how the semi-join technique can be used to reduce the cost of the join between *r* and *s*?

[4/25]

g) Scalable systems designed for big data applications often employ a model called 'eventual consistency'. Provide a practical example on the usage of 'eventual consistency'.

[3/25]

END OF EXAM PAPER