Xi'an Jiaotong-Liverpool University

PAPER CODE	EXAMINER	DEPARTMENT	TEL
CSE201		Computer Science & Software Engineering	

FIRST SEMESTER 2018/2019 FINAL EXAMINATIONS

BACHELOR DEGREE – Year 3

DATABASE DEVELOPMENT AND DESIGN

TIME ALLOWED: 2 Hours

INSTRUCTIONS TO CANDIDATES

1、 **Total marks available are 100. This will count for 80% in the final assessment.**

2、 **Answer ALL questions.**

3、 **The number in the column on the right indicates the marks for each section.**

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

5、 **The university approved calculator - Casio FS82ES/83ES can be used.**

6、 **All the answers must be in English.**

THIS PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

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

[25 marks]

A relational database holds two relations: *student (sID, name, email)* and *module (sID, title)* with the following information:

Relation *student*:

- Tuples are stored as fixed-format, fixed-length records, each of 300 bytes.
- There are 10,000 tuples.
- Each tuple contains a key attribute *sID* of length 20 bytes; other fields and the record header make up the rest.

Relation *module*:

- Tuples are stored as fixed-format, fixed-length records, each of 200 bytes.
- There are 50,000 tuples.
- Tuples contain attribute *module.sID* (ID of a student who takes a module), referencing to the *student.sID*.

Assume that each student must take exactly 5 modules. The student records are sequentially ordered by *sID* and the "clustered disk organisation" strategy is used. This means that, for each student record, the 5 module records also reside in the same block. Also, assume that a record does not span over more than one block. The records are stored in a collection of 4-kilobyte (4,096 bytes) disk blocks (i.e., the size of each block is 4 kilobytes).

a) Calculate the number of disk blocks needed to store the two relations.

[2/25]

b) Suppose that a dense primary index is to be created on *sID* for the student relation, i.e., one index entry for each tuple. Each index entry includes a key and a 10-byte pointer to data (an 8 byte block ID and a 2 byte offset). How many disk blocks are needed to store the index?

[4/25]

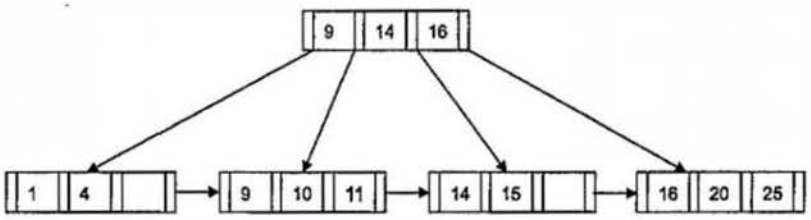
c) Suppose that a sparse primary index to be created on *sID* for the student relation, i.e., one index entry for each disk block. Each index entry includes a key and a 10-byte pointer to data (an 8 byte block ID and a 2 byte offset). How many disk blocks are needed to store the index?

[4/25]

d) Suppose that at a certain stage, a B+ tree index is created on *student.sID* as shown below. Draw the B+ trees after the following operations: (1) insert 12; (2) insert 13; (3) delete 15. Each subsequent operation should be performed based on the result of the previous operation.

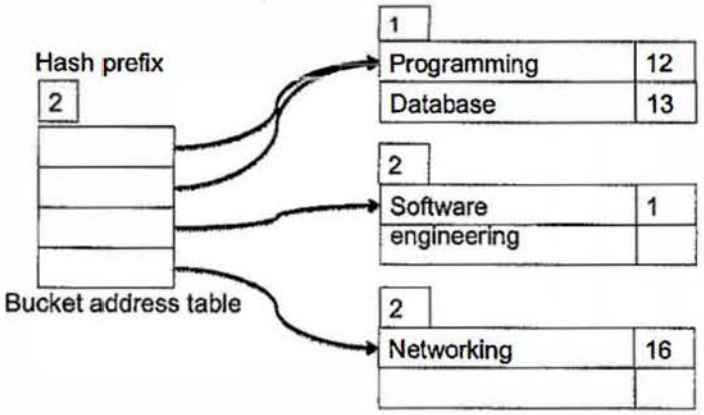
[9/25]

Initial tree



e) Suppose that at a certain stage, an extendable hash index is created on the *title* for the relation *module* as shown below. Based on the hash values on the titles in the table, draw the hash index after insertion of the following tuples (tuple is of the form "*sID*, *title*"): (1) "16, Machine Leaning"; (2) "20, Machine Leaning".

Search key	Hash value
Programming	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
Software engineering	1010 0101 1111 0010



[6/25]

Question 2. Consider the following three relations in a banking system:

i. *account(account_Number, customer_Name, balance, branch_Name)*
ii. *branch(branch_Name, branch_City, branch_postcode, magagerID)*
iii. *manager (managerID, managerName, salary, managerPhone)*

"*account_Number*", "*branch_Name*", and "*managerID*" are the keys for relations *account*, *branch*, and *manager*, respectively. Relation *account* has 500,000 tuples stored on 50,000 blocks. Relation *branch* has 1,000 tuples stored on 100 blocks. Relation manager has 500 tuples stored on 50 blocks. Suppose that it takes $t_T=0.1ms$ for one block transfer and $t_S=4ms$ for one seek. Answer the following questions.

[25 marks]

a) Describe how the selection $\delta_{salary<10,000}$ on relation *manager* can be evaluated in the most cost effective way by using a primary B+tree index and a secondary B+tree index on *salary*, respectively.

[4/25]

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

[4/25]

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

[4/25]

d) Assume that relations *account* and *branch* have been sorted and the merge join algorithm is used to evaluate "*account* $\bowtie$ *branch*". The buffer for reading and writing, $b_b=2$. How many block transfers are needed? How many seeks are needed?

[4/25]

e) Suppose that the hash join algorithm is applied to evaluate "*account* $\bowtie$ *branch*", the number of partitions, $n_h=50$, and the size of the buffer for reading and writing, $b_b=2$. How many block transfers are needed? How many seeks are needed?

[4/25]

f) With the results from Question 2.b), 2.c), 2.d) and 2.e), which algorithm is the most cost effective in terms of total time for block transfers and seeks? Justify your answer.

[5/25]

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

staff(ID, name, email, department)
teaches(ID, module_Code)
module(module_Code, module_Title, level)

where *staff.ID* is the key for *staff*, and *module_Code* is the key for *module; teaches.ID* and *teaches.module_Code* are the foreign keys referencing *staff* and *module*, respectively.

- the number of records in *staff*, $n_{staff} = 1{,}000$;
- the number of blocks in *staff*, $b_{staff} = 200$
- the number of distinct values for the attribute *department* in the *staff* relation, *V(department, staff) = 50*
- index: a three-level primary B^+-tree index (height=3) on the *ID* attribute of *teaches* relation.
- the number of records in *teaches*, $n_{teaches} = 3{,}000$
- the number of blocks in *teaches*, $b_{teaches} = 100$
- the number of records in *module*, $n_{module} = 1{,}500$
- the number of blocks in *module*, $b_{module} = 150$

Consider the following relational algebra expression and answer the questions below.

$\Pi_{name,\ module\_Title}(\sigma_{department="Industrial\ Design"\ \wedge\ level="4"}(staff \bowtie teaches \bowtie module)$

[25 marks]

a) One of the heuristic rules for query optimisation is to perform selection operations as early as possible. Write the equivalent algebra expression for the given expression based on this heuristic rule and the equivalence rules.

[4/25]

b) Suppose that the join between relations *staff* and *teaches* is evaluated using the indexed nested loop join algorithm; the join between the result and the relation *module* is evaluated using the nested loop join algorithm. Also, assume that pipelining is used for selection, projection and nested loop join. Draw an annotated evaluation tree for the relational algebra expression obtained from Question 3.a).

[4/25]

c) Based on the given catalog information and query evaluation tree, what is the estimated size of the selection $\sigma_{department="Industrial\ Design"}(staff)$? How many blocks are needed to store the results?

[4/25]

d) Based on the given catalog information and query evaluation tree, what is the estimated size of the join "*staff* $\bowtie$ *teaches*" using the indexed nested loop join algorithm?

[4/25]

e) Based on the results from Question 3.d), what is the estimated size of the nested loop join with the relation *module*? Justify your answer.

[4/25]

f) Suppose that the linear scan algorithm is used to evaluate all the selection operations. What is the total number of block transfers for the evaluation plan in Question 3.b)? Note that no intermediate relation needs to be stored as the result of using pipelining.

[5/25]

Question 4. Answer the following questions.

[25 marks]

a) Is the following schedule conflict serialisable? If yes, to which serial schedule is it equivalent? Justify your answer.
Schedule: T1:read(A); T1:write(A); T1:read(B); T2:read(A); T3:write(B); T2:read(Z); T4:read(B); T4:read(W); T4:write(W); T2:read(W).

[4/25]

b) Is the following schedule recoverable? Justify your answer.
Schedule: T1:write(X); T1:write(Y); T4:read(Y); T4:write(A); T2:read(X); T2:write(Y); T2:abort; T4:commit; T1:write(Z); T1:commit; T3:read(Y); T3:write(Z); T3:commit.

[3/25]

c) Consider the following transaction logs and answer questions: (1) At the checkpoint, which transactions should be in the list L? (2) When recovering from the system crash, in the redo pass, which operations need to be redone? (3) In the undo pass, which transactions need to be undone? (4) What transaction logs need to be added after the successful recovery?

Start of the logs
<T101 start>
<T101, A, 5896, 7006>
<T104 start>
<T104, B, 1065, 1732>
<T105 start>
<checkpoint {L}>
<T105, C, 35, 190>
<T105 commit>
<T101 commit>
<T107 start>
<T107, A, 377, 773>
<T104, B, 1000>
<T104 abort>
←System crash, start recovery

[4/25]

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

[6/25]

e) Assume there are two relations, *club* stored at Liverpool, and *customer* stored at Suzhou. The join attribute is *customer_ID*. Assume that a query is initiated at Liverpool site. Describe how the bloom-join can be used to reduce the cost of the join of the two

relations stored at different geographical locations.

[4/25]

f) State two key characteristics of object-relational models, compared to traditional relational databases.

[4/25]

END OF EXAM PAPER