UNIVERSITY OF KELANIYA - SRI LANKA
FACULTY OF COMPUTING AND TECHNOLOGY
B.Sc. Honours in Computer Science Degree Examination – February 2022
Academic Year 2019/2020 – First Semester
Computer Science
CSCI 21062 Advanced Database Management Systems

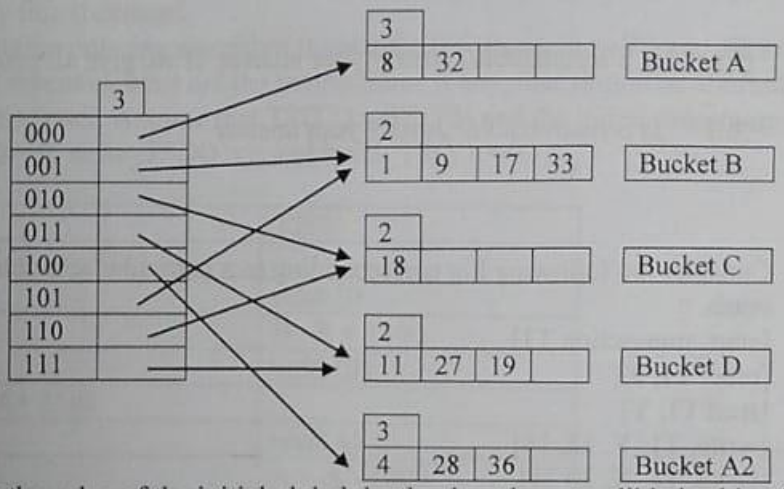
Answer **all** questions

No. of Pages: **Six(06)** No. of Questions: Five(05) Time: **Two and half (2 1/2) hrs.**

1.

A file has 30,000 records of fixed length. The record length is 115 bytes.

(a) This file is stored using the extendible hashing index on a non-key field **dept_no**. The following diagram shows 13 records that are stored using this hashing technique. The entries in the index are the hash values after applying the hashing function on the non-key field **dept_no**.



(i) What was the value of the initial global depth when the extendible hashing index was created?
(ii) Is it possible to say that a split has occurred due to an insertion of any entry? If so, what could be the possible entry/entries that was inserted.
(iii) List the steps to search the hash value 28.
(iv) By using a diagram, show the index after inserting the entry with hash value 69. Will the directory be doubled? Justify your answer.

(b) Assume that this file is stored on a disk with a block size of 512 bytes using an unspanned organisation. A block pointer is 6 bytes long. The file is not ordered by the non-key field **dept_code** which is 9 bytes long and it is required to construct a secondary index on **dept_code**. The secondary index keeps index entry at a fixed length and have a single entry for each index field value but creates an extra level of indirection to handle the multiple pointers. If the record pointers cannot fit in a single block, a linked list of blocks is used. Assume that there are 500 distinct values of dept_code and that the records in the file are distributed evenly among these values.

For the above organization, calculate each of the following:

(i) The blocking factor for the file.
(ii) The blocks needed to store the file.
(iii) The number of average block accesses, if a linear search is done on the data file.
(iv) The blocking factor for the index file.
(v) The number of blocks needed by the level of indirection that stores record pointers.
(vi) The number of first level index entries.
(vii) The number of first level index blocks.
(viii) The number of levels needed to make a multilevel index.

Continued.....

2. (a) (i) Draw the state transition diagram illustrating the states for transaction execution.

(ii) List two possible reasons for a transaction to fail in the middle of execution.

(b) Consider the following schedule S of transactions T1, T2, T3, T4:

S : $R_2(A)$; $W_3(A)$; C_3; $W_1(A)$; C_1; $W_2(B)$; $R_2(C)$; C_2; $R_4(A)$; $R_4(B)$; C_4.

Here, $R_i(A)$ means read the item A of transaction i, $W_i(A)$ means write item A of transaction i means transaction i is committed.

(i) List all conflict operations.

(ii) Is S serializable? Justify your answer. If so, give all possible serial schedules.

(iii) Is S recoverable? Justify your answer

(c) Consider the following log corresponding to a particular schedule at the point of a system crash.

[start_transaction T1]
[read, T1, X]
[read T1, Y]
[write, T1, Y, 12, 18]
[write, T1, X, 20 25]
[commit T1]
[start_transaction T2]
[read, T2, Z]
[read, T2, Y]
[write, T2, Z, 24. 40]

← System crash

Describe the recovery process if the,

(i) deferred update technique is used;

(ii) immediate update technique is used.

Continued.....

(a) Let A and B be two accounts that are accessed by transactions T1 and T2. Transaction T1 updates the balance of account A and transaction T2 gives a bonus for both accounts A and B.

T1:	**T2:**
read(A)	read(B)
A:= A + 1500	B = B + 1100
write(A)	write(B)
	read(A)
	A = A - 1000
	write(A)

(i) Using the 2PL protocol, add the lock and unlock operations to T1 and T2 and rewrite the transactions T1 and T2.

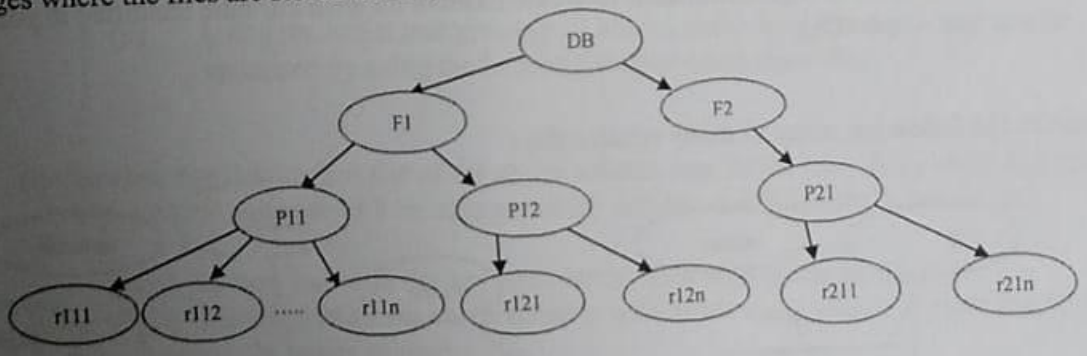
(ii) Give a concurrent execution for the transactions T1 and T2 given in (i) above, such that it prevents the lost update problem.

(iii) A concurrent execution for the transactions T1 and T2 given in (ii) above will not result in a dead lock. Justify this statement.

(iv) If the timestamp ordering algorithm is used for the transactions T1 and T2 and the following schedule is executed, what are the transactions, if any, that should be aborted and rolled back? Justify your answer. Assume that TS(T1) < TS(T2) and the initial timestamps of read_TS(X) = 0, read_TS(Y) = 0, write_TS(X) = 0 and write_TS(Y) = 0.

T1	T2
read(A)	
	read(B)
	B= B + 1100
	write(B)
A = A+ 1500	
	read(A)
	A = A - 1000
write(A)	
	write(A)

(b) Consider the following granularity tree. Here DB is the database, F1, F2 are files, P11, P12 etc. are pages where the files are stored in. r11, r112 etc. are the records.



Consider the following transactions in the given order. Which transactions can be executed in parallel (i.e. Concurrently)? Which transactions must be rolled back? Drawing only the required part of the above diagram is sufficient.

(i) T1 reads r112
(ii) T2 modifies r112
(iii) T3 modifies r11n

Continued.....

4. (a) (i) List two general guidelines for creating and using indexes.

(ii) List two goals of tuning the physical database design.

(iii) Consider the following query written on the relation `Student(S_no, Sname, Degree, Level)`:

```
Select Sname, Degree, Level
From Student
Where level >= 2  OR Degree = "Computer Science";
```

Discuss how this query could be tuned to make it more efficient.

(iv) Consider the following relations which hold information of students who have borrowed library books. The status of the borrower is either 1, 2 or 3 which represents the level of a student.

```
Book (Title, Author, Pname, Acc_no)
Borrowers(Bnum, Name, Address, City, Card_no, Status)
Loans(Card no, Acc no, Date)
```

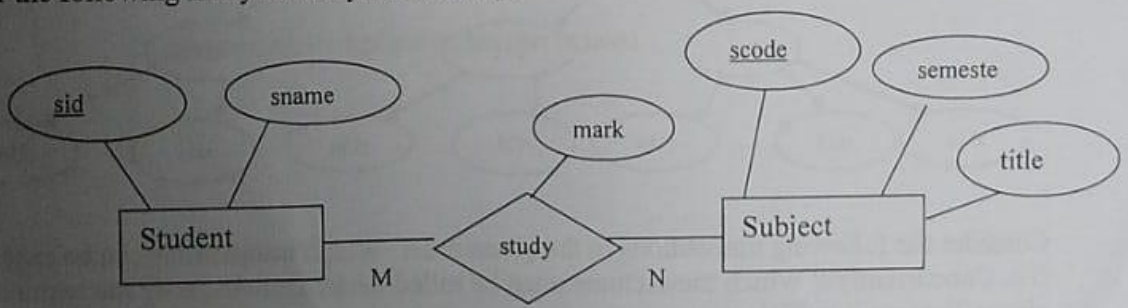
Following is a typical query against these relations:

```
Select B.Name, B.Addres, B.Status, L.Date
From Borrowers  B, Loans L
Where L.Card_no  = B.Card_no and (status = 2 or stauts = 3)
Order by B.Name
```

(A) On what attribute(s) of relation(s) if any, should indexes be defined to speed up the above query? Give reasons for indexing or not indexing an attribute.

(B) Write SQL commands to create indexes for each attribute identified in your answer to part (A).

(b) Consider the following many to many relationship:



(i) Give the normalised relations for the above ER diagram.

(ii) What opportunities might exist for denormalising the normalised relations given in part (b) (i) ? Give the suitable denormalised relations.

Continued.....

(c) (i) Consider each of the following relation schemas, and the set of functional dependencies. Identify whether the functional dependencies given for each relation schema are equivalent or not.

(A) R(A,C,D,E,H)
Functional dependency F = {A → C, AC → D, E → AD, E → H}
Functional dependency G = {A → CD, E → AH}

(B) R(A,B,C,D,E,F)
Functional dependency F1={A->BC, B->CDE, AE->F}
Functional dependency F2={A->BCF, B->DE, E->AB}

(ii) Consider each of the following relation schemas and the given functional dependencies. For each of the functional dependencies identify the minimal functional dependency.

(A) R(A, B, C)
F: { AB -> C A ->B}

(B) R(A,B,C,D)
F : {A -> B , C->B, D ->ABC, AC ->D}

5. (a) The following are some relations that could be used for a student database.

Student(st_num, name, major_subject, year, age)
Class(classname, dept_name, meets_at, room,)
Registered (st_num, classname)
Faculty(faculty_id, dept_name)

The underlined attribute(s) is (are) the primary key. Following query is executed on the above schema:

Retrieve the names of students who are aged above 22 and enrolled in the Department of Software Engineering.

(i) Draw a possible initial query tree for the above.
(ii) Using the initial tree given in (i) above, show how the query tree can be optimised by using the heuristic optimisation algorithm.

(b) (i) Suppose that the Student file of the above schema has 20000 records in 4000 disks blocks with a blocking factor of 5 records per block and has the following access paths:

(A) A primary index on the key attribute *st_num* with levels $x_{st\_num} = 4$.
(B) A clustering index on major subject, with levels $x_{major\_subject} = 3$ there are 50 distinct values for major subject.
(C) A secondary index on *year* with levels $x_{year} = 2$. There are 4 distinct values for *year*.

Using the cost functions identify whether any of the access paths given above or the linear search approach is suitable for each of the following operations.

Op1 : $\sigma_{st\_num = 'S001'}$(Student)
Op2: $\sigma_{major\_subject = 'AD' \text{ and } year = '3'}$(Student)

Continued.....

(ii) In addition to the above access path, suppose that the Registered file has 40000 records stored in 125 disk block and has the secondary index on *st_num* with $x_{st\_num} = 4$.

Using the cost functions for the join operations, identify the best way to perform the following join. Use the blocking factor of the resulting join file as 4 (i.e. $bfr_{SR} = 4$).

Student $\bowtie_{st\_num = st\_num}$ Registered

Note: You may consider the following cost functions for your calculation.

Cost function for SELECT

S1: using a primary index to retrieve a single record

$C = x + 1$

S2: Using an ordering index to retrieve multiple records

$C = x + (b/2)$ (if comparison on a key field with an ordering index)

S3: Using a clustering index to retrieve multiple records

$C = x + \lceil (s/bfr) \rceil$

S4: Using a secondary index

$C = x + s$ (for equality)

$C = x + (b_{I1}/2) + (r/2)$ - for comparison condition

Here x – number of levels , s – selection cardinality, b_{I1} – number of first level index blocks, r – number of records in the file

Cost function for JOIN

Assume that the relations R and S are stored in b_R and b_S blocks respectively and r_R and r_S are the number of records of relations R and S respectively.

J1 - Using Nested-loop join

If R is used for the outer loop the cost function:

$C_J = b_R + (b_R * b_S) + (js * r_R * r_S)/brf_{RS}$

J2 : Using Single-loop join and an access structure

Suppose an index exists for the joining attribute B of S with index level x_B:

If the type of index is a primary index then the cost function:

$$C_J = b_R + (r_R*(x_B+1)) + (js * r_R * r_S)/brf_{RS}$$

If the type of index is a secondary index where s_B is the selection cardinality for the join attribute B of S then the cost function:

$$C_J = b_R + (r_R*(x_B+ s_B)) + (js * r_R * r_S)/brf_{RS}$$

Here js is the join selectivity.

.....//////.....