RollNo: _____ Name: _____

National University of Computer and Emerging Sciences, Lahore Campus

Course:	Advanced Database Concepts	Course Code:	CS451
Program:	BS(Computer Science)	Semester:	Spring 2018
Duration:	3 hours	Total Marks:	50
Paper Date:	Tue 22-May-2017	Weight	40%
Section:	CS	Page(s):	8
Exam:	Final		

Instruction/Notes: Scratch sheet can be used for rough work however, all the questions and steps are to be shown on this question paper. No extra/rough sheets should be submitted with question paper.
You will not get any credit if you do not show proper working, reasoning and steps as asked in question statements. Calculators are allowed.

Q1. *(8 points)* Consider a relation *R(a,b,c)* with r= 100,000 records, b= 5,000 pages (20 records fit on each page), and where *a* is a non-negative integer primary key. How many pages will be read from disk to answer the Query: SELECT * FROM R WHERE a<2500, in each of the following scenarios. Assume that 400 records match the selection predicate and total number of index pages are 100.

a. Relation R is stored in a heap (unordered) file.
b. Relation R is stored in a sequential (ordered) file sorted on a and there is a B+ tree index with search key *a*. All index pages are already in memory.
c. Relation R is stored in a heap (unordered) file. There also exists an unclustered B+ tree index with search key *a*. All index pages are already in memory.
d. Relation R is stored in a heap (unordered) file. There also exists an unclustered hash-based index with search key *a*. None of the index pages are in memory.

ANSWER:
a. We need to scan all of R for a cost of **5000 page IOs**.
b. Since the index is clustered and the 400 matching tuples fit on 20 pages, the cost will be approximately **20 page IOs**.
c. Since the index is unclustered, we will potentially make one page IO for each tuple for a cost of **400 page IOs**.
d. Since the index is a hash-based index and the predicate is an inequaility predicate, the best that we can do is scan the entire index. As the number of index pages is much less than the size of the relation, this will be more efficient than scanning the heap file. We can make this assumption because we know that the R records are very large. Only 20 fit on a single page. The total cost will be the total number of pages in the index (i.e. 100) plus 400 page IOs to actually fetch the data = **500 pages IOs**.

RollNo: _____ Name: _____

Q2. *(6 points)* For this question, consider hash indexes. Each index has a number of buckets consisting, each bucket consisting of many disk blocks. The indexes have the following characteristics:
A block can hold 50 value/pointer pairs. (The hash structure does not contain the records themselves, only pointers to them.) The file being indexed has 2000 records. The indexed field can contain duplicates.

a. For a linear hash index, how many total buckets require in the worst-case and also how many blocks will the buckets require in the worst-case (including blocks in an overflow chain)?
Hint: *The worst case is when all records have the same key value.*
b. In the above case, what is the worst-case number of I/Os to look up a value (including the record itself)?
c. For an extendible hash index, what is the best-case (smallest possible) size of the directory?

ANSWER:
a. The worst case is when all records have the same key value. They will placed in one block, plus overflow blocks, a total of 2000 / 50 = 40 blocks. The hash table must have 40 buckets, 39 empty and one full (with 39 blocks in an overflow chain).
b. 41 I/Os: 1 to get to the bucket, plus 39 to get to last block in the overflow chain, plus 2000 to get to the record (i.e. all records with same key).

c. In the best case, the directory has 64 entries (i.e. 2^6): 40 blocks are needed to hold data, so 32 entries are not enough to point to these 40 blocks.

RollNo: _____ Name: _____

Q3. *(4+3= 7 points)* Consider the following database of the "BLOGs" website. The website keeps tracks of the different users and blog written by them on different topics. Each user is identified by a unique username. The field Bwriter in Blog table is a foreign Key from user table and it gives the unique username of the Blog-writer

USER

Uname	**Age**	**Gender**
Sara	25	F
Zara	42	F
Ali	15	M
Ahmad	19	M
Aliya	27	F
Tania	29	F
Hamza	34	M

BLOG

BId	**Bname**	**Bwriter**	**Topic**
10	BigData Frameworks	Ahmad	Big Data
20	Generation Gap	Sara	Human Interactions
100	Map Reduce	Hamza	Big Data
30	The world of CNN	Ali	Deep Learning
50	Cassandra	Ali	Databases
70	Neural Nets	Tania	Deep Learning
60	MongoDB	Tania	Databases
120	Emerging trends	Sara	Big Data
80	Hbase	Ali	Databases

a. Suppose that following are the most often used queries on the large BLOGs database. You are required to improve the performance of all these queries. On which column(s) would you create an index? Write down the column name(s) and type of index (i.e. B^+-tree, Hash, Bitmap). Also one sentence why you choose the column(s).

Query1: SELECT Uname, Bname, Bid FROM user JOIN blog ON Uname=Bwriter WHERE Age< 25;
Query2: SELECT DISTINCT Uname, Age FROM user JOIN blog ON Uname=Bwriter WHERE Gender= 'F';

b. Consider the current state of BLOG table given above, find the selectivity (*sl*) of the condition to retrieve:

i. Bname= 'MongoDB' **ii.** Bwriter= 'Ali **iii.** Topic= 'Databases'

ANSWER:
a. Filter columns: Age (B+-tree), Gender (Bitmap), Joining Columns: Uname (Hash/B+-tree), Bwriter (Hash/B+-tree)

b. i. 1/9= 0.11 (i.e. 11%) ii. 3/9= 0.33 (i.e. 33%) iii. 3/9= 0.33 (i.e. 33%).

RollNo: _____ Name: _____

Q4. *(4 points)* Consider the following classes of schedules: conflict-serializable, view-serializable, strict, cascadeless, recoverable and non-recoverable. For a schedule ***S****: r2(X); w3(X); w1(Y); r2(Y); r2(Z); r3(Y); c3; c2; r1(Z); c1*, state which of the preceding classes it belongs to. Give proper reason. The actions are listed in the order they are scheduled. Also draw the serializability (precedence) graph for this schedule. If the schedule is conflict-serializable or view-serializable, write down the equivalent serial schedule(s) otherwise explain why it is not.

ANSWER:

S: r2(X), w3(X), w1(Y), r2(Y), r2(Z), r3(Y), c3, c2, r1(Z), c1.

It is conflict-serializable, view serializable, not strict , not cascadeless, non-recoverable as T2/T3 read the value of Y which is updated by T1 and T3/T2 commit before T1.

Equivalent serial schedule is T1→T2→T3. Edges in graph: T1--Y-->T2, T1--Y-->T3, & T2--X-->T3.

RollNo: _____ Name: _____

Q5. *(6 points)* Show that the above schedule **S** will be accepted/rejected by the below protocols. Provide proper reason and show your working.

a. The strict two-phase locking protocol (add locks to the transactions)

b. The strict timestamp-ordering protocol (you have T1 < T2 < T3)

c. Optimistic Concurrency Control (Use defer the validation until a later time when the conflicting transactions have finished.)

ANSWER:

a. Accepted:

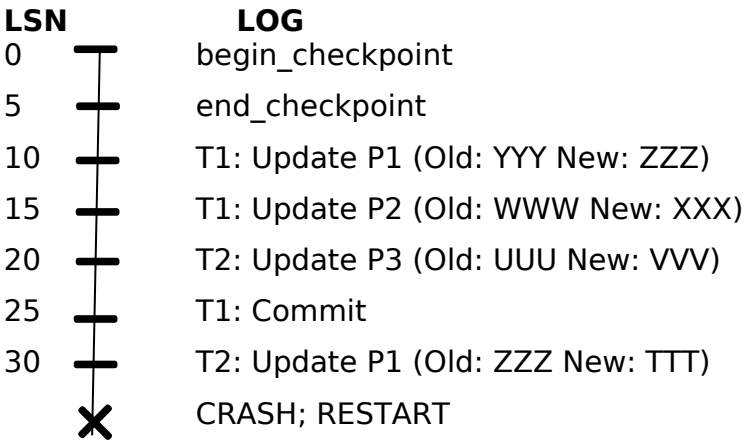
b. Accepted:

c. Accepted:

RollNo: _____ Name: _____

Q6. *(1+2+1+1=5 points)* Consider the execution shown in below Figure. Assume that the Dirty Page Table and Transaction Table were empty before the start of the log.

a. What is the value of the LSN stored in the master log record?
b. What is done during Analysis? (Be precise about the points at which Analysis begins and ends and show the contents of Dirty Page Table and Transaction Table constructed in this phase.)
c. What is done during Redo? (Be precise about the points at which Redo begins and ends.)
d. What is done during Undo? (Be precise about the points at which Undo begins and ends.)



ANSWER:
a) LSN = 1
b) Analysis phase starts from LSN=5 record until it reaches the end i.e. LSN=30. The end checkpoint contains the empty transaction table and dirty page table. The analysis phase will further reconstruct these tables as follows:

Transaction Table (TT)

TID	Last_LSN	Status
T1	~~10~~ ~~15~~ 25	~~inProcess~~ commit
T2	~~20~~ 30	inProcess

Dirty Page Table (DPT)

Page_ID	LSN
P1	10
P2	15
P3	20

c) REDO begins from LSN=10 (smallest LSN in DPT) and proceed with the REDO of updates. The LSNs {10, 15, 20, 30} corresponding to the updates for pages P1, P2, P3, and P1, respectively, are not less than the LSNs of pages in initial DPT. So those data pages will be read again and all the updates reapplied from the LOG. At this point REDO phase is finished and the UNDO phase starts.
d) UNDO is applied only to the active transaction T2. UNDO phase starts at LSN=30 (the last update of active transaction T2) and proceeds backward in the LOG. The backward chain of updates for T2 (LSN=30 and LSN=20 records) is followed and undone.

RollNo: _____ Name: _____

Q7. *(8 points)* Consider the above BLOGs database, and the following SQL query:

SELECT Uname, Bname
FROM user JOIN blog ON Uname=Bwriter
WHERE Age< 20 AND (Topic= 'Deep Learning' OR Topic= 'Big Data');

Write an efficient relational-algebra expression that is equivalent to this query and draw query tree of the optimal query execution plan for this query.

ANSWER:
RA:
PROJECT Uname, Bname ((PROJECT Uname (SELECT Age<20 (USER))) JOIN Uname=Bwriter (PROJECT Bname, Bwriter (SELECT *Topic= 'Deep Learning' OR Topic= 'Big Data'* (BLOG)**)))**

Opr1. Select: Age<20; opr2. Project: Uname; opr3. Select: *Topic= 'Deep Learning' OR Topic= 'Big Data'*; opr4. Project: Bwriter, Bname; opr5. JOIN: on Uname=Bwriter; opr6. Project: Uname, Bname.

RollNo: _____ Name: _____

Q8. *(6 points)*
a. Name the cost components for a cost function that is used to estimate query execution cost. Which cost components are used most often as the basis for cost functions?
b. What are the three types of parallel architectures applicable to database systems? Which one is most commonly used?
c. What are intra-query and inter-query parallelisms? Which one is harder to achieve in the shared-nothing architecture?

ANSWER:
a. Disk access for Read/Write i.e. I/O, Storage cost for intermediate result, communication, ...
b. share-everything, shared-disk, and shared-nothing architectures (commonly used).
c. Intra-query: Execute independent operations in parallel
Inter-query: Execution of multiple queries in parallel. Goal: scale up
Inter-query is difficult to achieve on shared-disk or shared-nothing architectures