CSE 444 Final Examination

June 7, 2012, 8:30am - 10:20am

Name: ________________________________

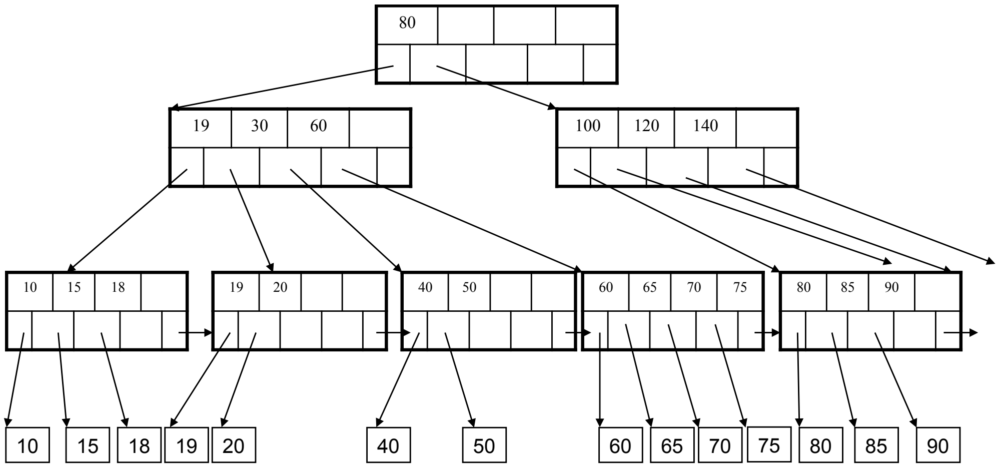
Question	Points	Score
1	20	
2	15	
3	20	
4	15	
5	30	
Total:	100	

- This exam is an open book exam.
- You have 1h:50 minutes; budget time carefully.
- Please read all questions carefully before answering them.
- Some questions are easier, others harder; if a question sounds hard, skip it and return later.
- Relax! You are here to learn.

1 Indexes

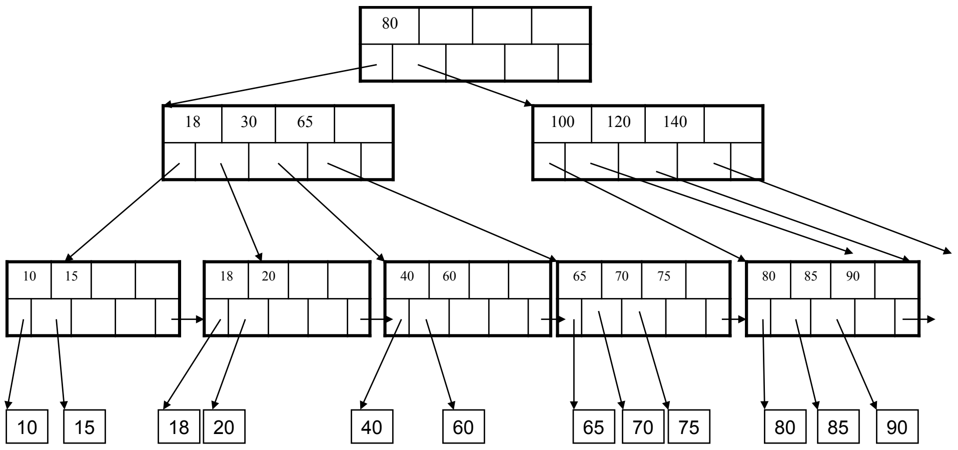
1. (20 points)

   (a) (10 points) Consider the following B+ tree. Draw the modified tree after deleting value 19 and then 50. We only ask you to draw the final tree but you can draw both for partial credit. For the parts of the tree that remain the same, you can simply write "same":



Answer (Draw an updated B+ tree):

Solution:



(b) (10 points) Consider a relation R(a,b,c) with T(R) = 10,000 records, B(R) = 1,000 pages (10 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 selection query $\sigma_{a<2500}(R)$ in each of the following scenarios. Assume that 100 records match the selection predicate.

1. Relation R is stored in a heap file.
2. Relation R is stored in a sequential file sorted on a and there is a B+ tree index with search key a. All index pages are already in memory.
3. Relation R is stored in a heap file. There also exists an unclustered B+ tree index with search key a. All index pages are already in memory.
4. Relation R is stored in a heap file. There also exists an unclustered hash-based index with search key a. None of the index pages are in memory.

Answer (Compute the cost of the query in all the cases):

Solution:

1. We need to scan all of R for a cost of 1K page IOs.
2. Since the index is clustered and the 100 matching tuples fit on 10 pages, the cost will be approximately 10 page IOs.
3. Since the index is unclustered, we will potentially make one page IO for each tuple for a cost of 100 page IOs.
4. 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. Assuming that 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 10 fit on a single page. The total cost will be the total number of pages in the index plus 100 page IOs to actually fetch the data.

2 Query Optimization

2. (15 points)

   Consider the following four relations:

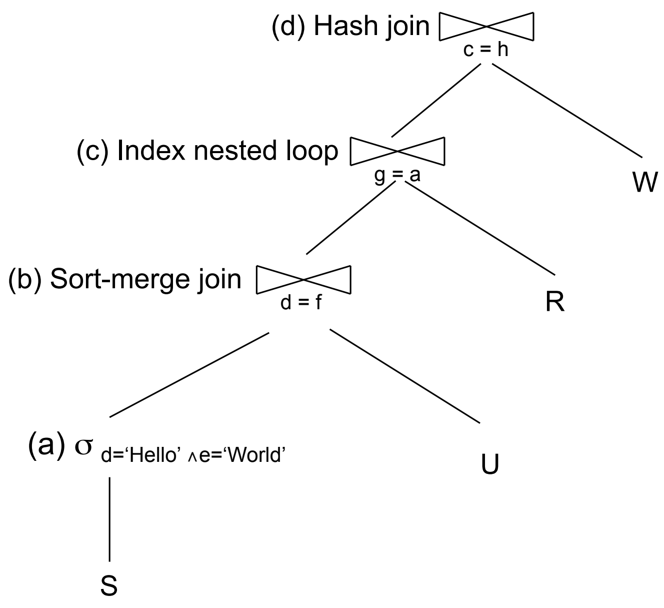
Relation	Records	Pages	Unique Values
R(a,b,c)	T(R) = 10,000	B(R) = 1,000	
S(d,e)	T(S) = 100,000	B(S) = 10,000	V(S,d) = 100 and V(S,e) = 100
U(f,g)	T(U) = 100,000	B(U) = 10,000	V(U,f) = 10,000.
W(h,i,j)	T(W) = 1,000	B(W) = 100	

   Additionally, consider that there exists an unclustered B+ tree index on R.a, which is the primary key of *R*.

   (a) (15 points) What is the cost of the query plan below assuming 2,000 pages of memory? Count only the number of page I/Os.

   Hints:

   - As you compute the cost of the plan in a bottom up fashion, at the output of each operator consider whether the result can fit in memory and be used directly by the next operator or if you need to write anything to disk.
   - Consider whether you need one-pass or two-pass algorithms.



Answer (Compute the cost of the plan):

Solution:
We compute the cost in a bottom-up fashion

- Since there are no indexes on relation S, the cost of operator (a) is 10K page IOs. The cardinality of the output is $\frac{100{,}000}{100*100} = 10$ tuples on 1 page.
- For operator (b), we can sort the output of (a) in memory since it holds in one page. We need a two-pass sort algorithm to sort U for an added cost of $3 * 10K = 30K$ page IOs. The cardinality of the output of (b) is $\frac{10*100{,}000}{10{,}000} = 100$ tuples since each tuple from S has a $\frac{1}{10{,}000}$ probability of joining with a tuple in U.
- For operator (c), for each of the 100 tuples from the outer relation, we need to perform 1 disk IO to retrieve the matching R tuple (R.a is the primary key for R). The total cost is thus 100 page IOs. The cardinality of the result is 100 tuples.
- Finally, for operator (d), we can build a hash-table in memory using the output of (c) directly as it is produced. We then need to scan W for a cost of 100 page IOs.

The total cost is thus $10K\ (a)\ +\ 30K\ (b)\ +\ 100\ (c)\ +100\ (d)\ =\ 40{,}200$ page IOs.

3 Transactions Concurrency Control

3. (20 points)

   (a) (10 points) Consider the following schedule. Explain what happens when transactions try to execute as per this schedule and the DBMS uses timestamp-based concurrency control.

   $ST_1 \rightarrow ST_2 \rightarrow ST_3 \rightarrow ST_4 \rightarrow R_1(X) \rightarrow R_2(X) \rightarrow W_2(X) \rightarrow W_1(X) \rightarrow W_3(Y) \rightarrow W_2(Y) \rightarrow C_3 \rightarrow W_4(Z) \rightarrow C_4 \rightarrow R_2(Z)$

   Answer (Fill in the table below showing what happens as the transactions execute):

T_1	T_2	T_3	T_4	X	Y	Z
1	2	3	4	RT=0 WT=0 C=true	RT=0 WT=0 C=true	RT=0 WT=0 C=true
$R_1(X)$						
...						

Solution:

T_1	T_2	T_3	T_4	X	Y	Z
1	2	3	4	RT=0 WT=0 C=true	RT=0 WT=0 C=true	RT=0 WT=0 C=true
$R_1(X)$				RT=1		
	$R_2(X)$			RT=2		
	$W_2(X)$			WT=2 C=false		
$W_1(X)$ **abort**						
		$W_3(Y)$			WT=3 C=false	
	$W_2(Y)$ **delay**					
		C_3			C=true	
	ignore $W_2(Y)$ **and proceed**					
			$W_4(Z)$			WT=4 C=false
			C_4			C=true
	$R_2(Z)$ **abort**					

(b) (10 points) Consider the following schedule. Explain what happens when transactions try to execute as per this schedule and the DBMS uses **multiversion** concurrency control:

$ST_1 \rightarrow ST_2 \rightarrow ST_4 \rightarrow R_1(X) \rightarrow R_2(X) \rightarrow W_2(X) \rightarrow W_1(X) \rightarrow W_4(Z) \rightarrow R_2(Z)$

Answer

(Fill in the table below showing what happens as the transactions execute):

T_1	T_2	T_4	X_0 ...
1	2	4	
$R_1(X)$			RT=1
...			

Solution:

T_1	T_2	T_4	X_0	X_2	Z_0	Z_4
1	2	4				
$R_1(X)$			RT=1			
	$R_2(X)$		RT=2			
	$W_2(X)$			Create		
$W_1(X)$						
abort						
		$W_4(Z)$				Create
	$R_2(Z)$				RT=2	

Notice that transaction 1 aborts when it tries to write because the value that it wrote should have been read by T_2. This is the main reason why we need to keep track of the read timestamps for all the versions.

4 Transactions Recovery

4. (15 points)

A DBMS uses the ARIES recovery algorithm. The DBMS just crashed. Upon restart, the state of the database on disk is the following:

Log on disk:

LSN	1	2	3
Transaction ID	T1	T2	T2
Previous LSN	-	-	2
Type	Update	Update	Commit
Page ID	P1	P2	-
Description	Updated A...	Updated B...	-

Pages on disk

Page 1	PageLSN=1
A_1	

Page 2	PageLSN=0
B_0	

(a) (5 points) Show the transactions table and the dirty pages table at the end of the analysis phase:

Solution:

TransactionID	LastLSN
T1	1

Note: To simplify our discussion of the ARIES protocol, we ignored END log records. These log records are written once all the actions associated with either committing or aborting a transaction are completed. So here we remove T2 as soon as we see the commit log record.

PageID	RecoveryLSN
P1	1
P2	2

(b) (5 points) Explain what will happen during the Redo phase: Where will Redo begin? What updates will be performed? What changes will be made to the pages?

Solution:

- Redo will begin at the earliest RecoveryLSN, which is LSN=1 in our example.
- For the log entry at LSN=1, since P1 is in the Dirty Pages table and its RecoveryLSN is no greater than 1, then we will read P1 from disk. However, the pageLSN for P1 is already 1, so the update will NOT be redone.
- For the log entry at LSN=2, since P2 is in the Dirty Pages table and its RecoveryLSN is no greater than 2, then we will read P2 from disk. In this case the pageLSN is 0, so the update will be performed and the pageLSN for P2 will be updated to 2.

(c) (5 points) Explain what will happen during the Undo phase: What updates will be undone? What log entries will be written? What changes will be made to the pages?

Solution:

- The only loser transaction is T1. All its updates will be undone. In this case, there is only one update at LSN=1. That update will be undone.
- A CLR will be written to record the undo action for update at LSN=1.
- P1 will be updated to undo the change to A made by transaction T1. Additionally, the pageLSN will be updated to the LSN of the CLR.

Note: To simplify our discussion of the ARIES protocol, we ignored END log records. These log records are written once all the actions associated with either committing or aborting a transaction are completed.

5 Distributed and Parallel DBMSs

5. (30 points)

   (a) (10 points) In the presumed-abort two-phase commit protocol, what happens if the coordinator sends PREPARE messages and all but one subordinate vote to commit the transaction. The last subordinate wants to commit the transaction also but it crashes before receiving the PREPARE message from the coordinator.

   Answer (Describe the sequence of operations at the coordinator, at the subordinates that did not crash, and at the crashed subordinate after it recovers):

Solution:

- The coordinator will time-out waiting for the reply from the failed subordinate and it will decide to abort the transaction. It will then write an abort log record and will send abort messages to the subordinates.
- The subordinates that did not crash will receive the abort message from the coordinator. They will write an abort log record, will abort the transaction, and will "forget" it.
- At the crashed subordinate, after restarting, the recovery process will find that a transaction was executing at the time of the crash and that no commit protocol log record had been written, then the recovery process will abort the transaction, write an abort record, and forget the transaction.

(b) (10 points) Consider the following relations:

```
Purchases(cid, pid, time)
Products(pid,description,price)
```

where cid is the unique identifier of each customer; pid is the unique identifier of each product; time is the time when a customer bought a product; description is the description of the product, and price is its price.

Assuming that relation Purchases is stored at site A and that Products is stored at site B, describe how a semijoin can serve to efficiently answer the following query:

```
SELECT *
FROM Purchases, Products
WHERE Purchases.pid = Products.pid
AND Products.price > 50
```

Answer (Describe how the query will be answered using a semijoin):

Solution:
The semijoin can be used as follows:

- Site B computes $T = \pi_{pid}(\sigma_{price>50}(Products))$.
- Site B sends T to site A.
- Site A computes a semijoin between Purchases and T: $S = \pi_{cid,pid,time}(A \bowtie T)$
- Site A ships S back to site B.
- Site B computes the final answer as $S \bowtie Products$.

(c) (10 points) What does MapReduce do when some tasks take significantly longer to complete their executions than others. When is this approach helpful? When is it NOT helpful.

<u>**Answer**</u> (All three subquestions):

Solution:

When a MapReduce operation is close to completion, the master schedules backup executions of the remaining in-progress tasks. The task is marked as completed whenever either the primary or the backup execution completes.

This approach can be helpful when the extra latency is caused by hardware or system failures at the machine that executes the primary task. It can also help if that machine is not failing but is simply heavily loaded.

This approach does not help when the extra latency comes from an uneven distribution of the processing load to the different tasks. In that case, the replicated tasks will take just as long to complete as the primary ones.