Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.830 Database Systems: Fall 2007

Quiz I

There are 19 questions and 13 pages in this quiz booklet. To receive credit for a question, answer it according to the instructions given. *You can receive partial credit on questions.* You have **80 minutes** to answer the questions.

Write your name on this cover sheet AND at the bottom of each page of this booklet.

Some questions may be harder than others. Attack them in the order that allows you to make the most progress. If you find a question ambiguous, be sure to write down any assumptions you make. Be neat. If we can't understand your answer, we can't give you credit!

THIS IS AN OPEN BOOK, OPEN NOTES QUIZ.
NO PHONES, NO LAPTOPS, NO PDAS, ETC.

Do not write in the boxes below

1-4 (xx/26)	5-10 (xx/26)	11-13 (xx/14)	14-16(xx/20)	17-19 (xx/14)	Total (xx/100)
27 my	14 AT	13 AT	18	14	82 Sm
			Sm	Sm	

Name: DAN PORTS

Good!

I Short Answer Reading Questions

1. [8 points]:

The DBMIN paper ("An Evaluation of Buffer Management Strategies for Relational Database Systems" by DeWitt and Chou) proposes using a different buffer management strategy for each instance of an open database file (table, index, etc.) For each of the following access patterns, table sizes, and amount of free memory, indicate whether an LRU or MRU memory management strategy should be used and indicate the optimal number of pages of memory that should be allocated to the buffer for that file. If the either memory management strategy is equally good, circle, "either".

(Indicate page management strategy buffer size.)

A. A sequential scan of a file with 1000 pages, with 100 pages free memory.
LRU MRU (Either)

Pages of buffer memory for file: 1 ✓

+8

B. A 100 page file being used as the inner loop of a nested loops join with 101 pages free memory.
LRU MRU (Either)

Pages of buffer memory for file: 100 ✓

C. A 100 page file being used as the inner loop of a nested loops join with 99 pages free memory.
LRU (MRU) Either

Pages of buffer memory for file: 98 ✓

D. A 100 page file being used as the outer loop of a nested loops join with 99 pages free memory.
LRU MRU (Either)

Pages of buffer memory for file: 1 ✓

Name: DAN PORTS

2. [6 points]: Assume that a database uses two-phase locking, but releases both read and write locks before end-of-transaction. What implications does this have on the ability of the database to ensure the atomicity of transactions? (Hint: Consider what happens on transaction ABORT.)

(Write your answer below.)

(+3) If a transaction releases its locks before transaction end, other transactions will be able to see its uncommitted changes. If the transaction aborts, these other transactions would now need to be aborted — but they might have already committed. Also cascaded problem

3. [6 points]: A, B and C are relations in a DBMS, each containing fields f1 and f2. Field f2 is the primary key for C. B is many orders of magnitude larger than A, and C is many orders of magnitude larger than B. A page in each of these relations can hold p records, where p is a small integer of the order of 10. The whole of A fits in memory, but B and C do not. Disk seeks in this system are k times slower than sequential page reads, where k is also a small integer of the order of 10. Only I/O costs are significant; CPU costs can be safely ignored.

(+6)

Assume that the database uses a *block* nested loops join: first the system reads a page P of the outer relation, then either scans the entire inner relation (if not using an index) or reads all the index pages matching tuples in the page P.

Which of the following statements are true? For each question, briefly explain your reasoning in the space provided.

(Circle T for true or F for false for each option.)

T / F For block nested loop joins of the form A.f1 = B.f1, a clustered index on B.f1 always improves performance, relative to not using an index.

NL joins can be done w/ A in memory, so need to sequentially read A + B ✓

Index joins require |A| random I/Os. plus sequential reads of all the matching tuples.

So if nearly all tuples match, the NL join is faster

T / F For the join B.f2 = C.f2, a nested loops join using a primary clustered hash index on C.f2 will always be faster than a hash join.

As above, the cost of the NL index join (the worst case (where nearly all tuples match) ✓ is <B> + <C> sequential I/Os plus |B| random I/Os.

The cost of a GRACE join, assuming enough main memory to make GRACE's writes be approximately

Name: DAN PORTS

3b (cont'd).

Is $3(\langle B \rangle + \langle C \rangle)$. So the NL join is faster if $t|B| < 2(\langle B \rangle + \langle C \rangle)$

$\Rightarrow \sim 10 \cdot \sim 10 \cdot \langle B \rangle < 2(\langle B \rangle + \langle C \rangle)$

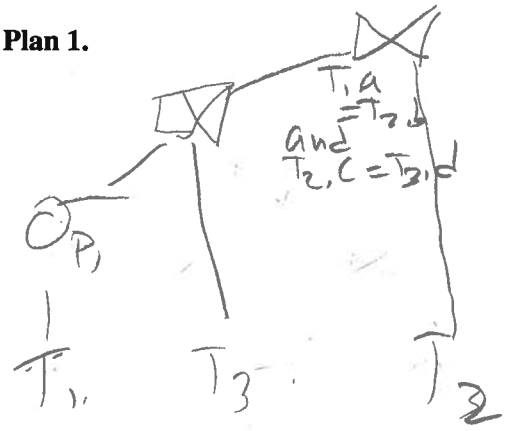
which is true if $\langle C \rangle$ is at least 2 orders of magnitude larger than B.

✓

4. [6 points]: The Selinger optimizer uses several heuristics to limit the number of plans it considers. Suppose you are joining three tables $T1$, $T2$, and $T3$, with join predicates $T1.a = T2.b$ and $T2.c = T3.d$, and that there is a selection predicate $P1$ over $T1.e$. Describe or show three plans that the Selinger optimizer would not consider.

(Write your answer in the spaces below.)

Plan 1.

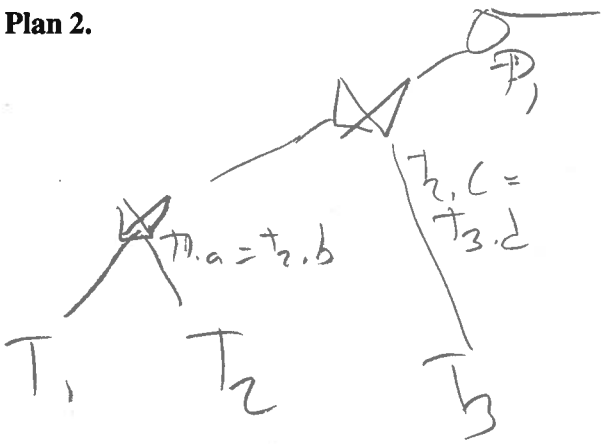


(+6)

(includes Cartesian Join)

✓

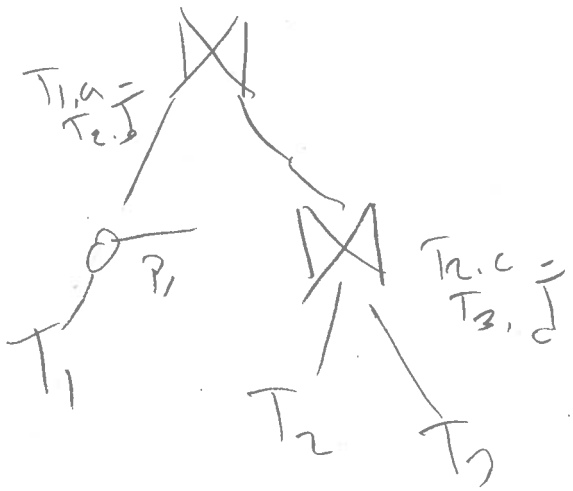
Plan 2.



(selection not pushed down to leaves)

✓

Plan 3.



(not left-deep)

✓

Name: D+N PORTS

II Optimistic Concurrency Control

5. [4 points]: True or False: In the OCC protocol presented in the Kung and Robinson Paper ("On Optimistic Methods for Concurrency Control", Red Book), every successfully committed transaction has three phases. If false, state why.

(Circle TRUE or FALSE, and explain, if needed.)

TRUE (FALSE)

4

R/O transactions have no write phase.

In the OCC protocol presented in the Kung and Robinson Paper (Red Book), transactions are "validated" based on the theory of serial equivalence. The paper states that for a pair of transactions T_i and T_j, T_j is ordered after T_i in the equivalent serial order if one of the following three conditions are satisfied:

A. T_i completes its write phase before T_j starts its read phase.

B. The write set of T_i does not intersect the read set of T_j, and T_i completes its write phase before T_j starts its write phase.

C. The write set of T_i does not intersect the read set or the write set of T_j, and T_i completes its read phase before T_j completes its read phase.

Ben Bitdiddle claims that there is a fourth condition that will also result in a serial equivalent schedule with T_i ordered before T_j:

D. The write set of T_i does not intersect the read set or the write set of T_j, and T_i completes its read phase before T_j begins its write phase.

6. [6 points]: Is Ben correct? If not, provide an example that illustrates why not.

(Circle YES or NO and show an example, if needed.)

YES (NO) Suppose T_2 reads A then begins validation and then, when T_2 has finished validation but before it begins its write phase, T_1 reads A and adds it to its write set. This is not serializable, because T_1 reads the old value of A but might write A after T_2, but Ben's condition permits it. It doesn't because $W(T_1) \cap W(T_2) = \phi$

3+ 1 = 4

Good try

Name: Dan Ports

Suppose we have two transactions here T1 and T2, whose execution is interleaved as follows:

T1	T2
	RA
RA	
RB	
	WB
WB	
	WC
COMMIT	
	COMMIT

7. [4 points]: Is this interleaving serializable? If yes, show the equivalent serial order. If not, state why not.

(Circle YES or NO and write an answer.)

YES NO

T_2, T_1

8. [4 points]: Can this interleaving be generated by a database that uses strict two-phase locking for concurrency control? If not, state why not.

(Circle YES or NO and write an answer, if needed.)

YES (NO)

When T_2 writes B; it acquires an exclusive lock on B, which it does not release until COMMIT. So T_1 would be unable to acquire an exclusive lock to write B.

4

Name: DAN PORTS

Now, suppose we run this same transaction schedule using optimistic concurrency control, as described in the Kung and Robinson paper. In this case, writes in the transaction schedule are performed on local copies of the data items and are not made globally visible until a transaction performs its write phase. Assume that T1 begins its validation phase before T2 finishes its read phase phase, and that T2 begins its validation phase before T1 completes its validation phase.

9. [4 points]: If we use the serial validation strategy (Section 4 of the paper), is the above execution interleaving permissible (i.e., allowed to continue without aborting)? If yes, indicate the equivalent serial order. If not, state the reason the interleaving is not allowed.

(Circle YES or NO and write an answer.)

YES (NO)

With serial validation, each transaction's write phase is in a critical section, so no other transactions can write at the same time. In particular, T_1 could not write B between T_2's two writes.

10. [4 points]: If we use the parallel validation strategy (Section 5 of the paper), is the above execution interleaving permissible (i.e., allowed to continue without aborting)? If yes, indicate the equivalent serial order. If not, state the reason the interleaving is not allowed.

(Circle YES or NO and write an answer.)

YES (NO)

T_2 must validate and begin its write phase first (otherwise it would abort, because T_1 writes B, which is in T_2's write set). Then when T_1 validates, it will abort, because T_2 is active and T_2 writes B, which is in T_1's read set.

W-W conflict

2

Name: DAN PORTS

III SpamDB

Simon A. Spammer has spent months assiduously harvesting people's email addresses and online purchasing habits, and correlating them. He has been storing his treasure trove of information in a relational database with two relations.

The first is a `persons` relation, where each record contains a person's name, age, unique SSN, and email address. The second is a huge `purchases` relation, which contains information about every purchase he knows about. Each purchase is made by a person with a particular SSN, and includes a product category as classified by the store, product name (a short string), and purchase price in dollars. The number of distinct product names is considerably larger than the number of distinct categories. The same product can be sold under different categories by different stores.

Simon has been using his database without any indices whatsoever. Unfortunately the database has grown huge and rather unwieldy of late, and neither relation in Simon's database fits in memory now. Naturally, Simon wants to add indices to improve performance. He is interested in the following classes of queries:

A: Find the email addresses of all people who made a purchase that cost more than some large dollar amount D, a rather rare occurrence.

B: Find the total dollar amount of purchases of a particular product.

C: Find the total dollar amount of purchases of a particular product category.

Answer the following questions to help Simon with his task. Write your answers in the space provided. For each index, indicate the type of index (e.g. hash or B-Tree) and whether it is clustered or unclustered. Assume you may only have one clustered index per relation.

11. [4 points]: To answer queries of type **A** efficiently, what index(es) should Simon add to his database? Why?

Clustered B+ tree on purchases.price to allow quick range queries on price ~~to support~~ X. Needs to be a B+-tree to support range queries.

4

clustered hash on persons.ssn to efficiently join the resulting SSNs from the products scan.

Name: DAN PORTS

12. [4 points]: If Simon wants to answer queries of type **B** and **C** efficiently, what index(es) should he add to his database? Why?

Hash indexes on purchases: product and purchases: category. Should be hash files instead of trees because range queries are not necessary. One of them should be clustered - whichever query occurs most frequently (# matches/cat > # matches/pdct) so CL on cat. better

3

13. [6 points]: If a particular product is always classified under a unique category irrespective of the store it is bought from, can Simon exploit this to speed up queries of type **B** or **C**? If yes, how, and why does your solution provide a performance boost?

Yes - can have a clustered hash index on (category, product), since for every product the category is known. This index can be clustered, eliminating random I/Os, and can be used by both queries B & C. Good

6

Name: DAN BARTS

V Dana's Concurrency Control

Dana Bass is designing a database with a new form of concurrency control. It works as follows:

- Every transaction T receives a timestamp, TS_T, when it begins. Timestamps are unique and are monotonically increasing.
- Every data object O has a read-timestamp, RT_O, containing the timestamp of the last transaction to read the object and a write timestamp, WT_O, containing the timestamp of the last transaction to write the object. Every object has a primary copy that is the value written by the last committed transaction.
- When a transaction T reads an object O, it sets RT_O to TS_T if $TS_T \geq RT_O$. The first time T reads O, it reads the primary copy of O and saves it in a *side-file* as $copy_{T,O}$. Successive reads of O by T read $copy_{T,O}$.
- When a transaction T writes an object O, it sets WT_O to TS_T if $TS_T \geq WT_O$ and $TS_T \geq RT_O$. Otherwise T aborts. When a write succeeds, the primary copy of O is not overwritten, but is stored as $copy_{T,O}$ in the side-file used by T on reads. Hence, other transactions continue to read the primary copy of O or their own side copies of O.
- At commit time, a transaction T installs any updates from its side-file, overwriting the primary copy of all objects it updated.

15. [6 points]: Suppose only one transaction is allowed to commit at a time. Does Dana's concurrency control protocol guarantee serializability? Why or why not?

(Write your answer in the space below.)

No. Consider:

T_1	T_2
	WA
RA	
WB	
Commit	
	RA
	Commit

(6) The scheme allows this, but it is not serializable,

16. [6 points]: Now suppose multiple transactions can commit concurrently. Does Dana's concurrency control protocol guarantee serializability? Why or why not?

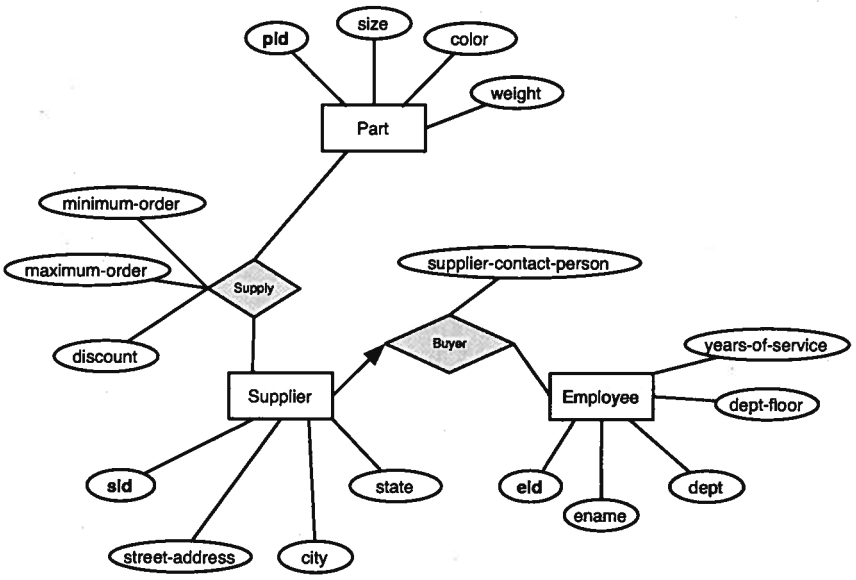
(Write your answer in the space below.)

(6) No. Same as above

Name: DAN PORTS

VI Schema Design

Consider the following entity-relationship diagram (here, primary keys are shown in **bold**):



Every employee can be a buyer for multiple suppliers, but each supplier has only one buyer employee.

Every supplier supplies multiple parts, and every part is supplied by multiple suppliers.

17. [6 points]: A hierarchical, or tree-structured schema is one in which each table has exactly one parent, unless it is a root table in which case it has zero parents. Recall that for many-to-many relationships, a hierarchical schema results in some redundancy, as data from tables at the bottom of the hierarchy is replicated multiple times. Assuming that there are 100 suppliers and 2000 parts and that each supplier selects the parts it supplies uniformly and at random, devise a hierarchical schema with minimum redundancy that corresponds to this ER diagram. You do not need to write out all of the attributes of the tables and relationships – just show the hierarchical relationship of the tables.

(Show your hierarchy in the space below.)

(6)

Employee
|
Supplier
|
Part

Name: DAN PORTS

18. [4 points]: Assuming that every attribute is functionally dependent on the key of the entity in which it appears and that `dept-floor` is also functionally dependent on `dept`, write a collection of 3NF relations corresponding to this diagram.

(Show your relational schema in the space below.)

Part: {pid, size, color, weight}
supplier: {sid, street, city, state, supplier-rating, buyerid}
employee: {eid, ename, deptid, years-of-service}
dept: {deptid, dept name, dept-floor}
supply: {pad, sid, min, max, discount}

(4)

19. [4 points]: What is the disadvantage of dropping `eid` from the schema and making `ename` the key for `employee`?

(Write your answer in the space below.)

Can't represent multiple employees with the same name, Even if using the employee's other fields (dept, ...) to disambiguate, uniqueness is not guaranteed

(4)

End of Quiz I

Name: DAN PORTS