

CS411 Database Systems

Fall 2006

Department of Computer Science
University of Illinois at Urbana-Champaign

Final Examination

December 15, 2006
Time Limit: 180 minutes

- Print your name and NetID below. In addition, print your NetID in the upper right corner of every page.

Name: _____ **NetID:** _____

- Including this cover page, this exam booklet contains **13** pages. Check if you have missing pages.
- The exam is closed book and closed notes. No calculators or other electronic devices are permitted. Any form of cheating on the examination will result in a zero grade.
- Please write your solutions in the spaces provided on the exam. You may use the blank areas and backs of the exam pages for scratch work. Please do not use any additional scratch paper.
- Please make your answers clear and succinct; you will lose credit for verbose, convoluted, or confusing answers. *Simplicity does count!*

Problem	1	2	3	4	5	6	7	8	Total
Points	10	10	11	16	12	10	16	15	100
Score									
Grader									

Turn over the page when instructed to do so.

Problem 1 (*10 points*) True/False Questions

For each of the following statements, indicate whether it is *TRUE* or *FALSE* by circling your choice. If you change your mind, cross out both responses and write “True” or “False”. You will get *1 point* for each correct answer, *0 points* for each incorrect answer.

- (1) True False
Triggers can operate on insertion, deletion, and updates.
- (2) True False
A user application executes a database Trigger by invoking “RUN TRIGGER *triggername*”
- (3) True False
Secondary storage is volatile.
- (4) True False
Seek time is part of the disk latency time.
- (5) True False
When an index is clustered it requires that the data records are sorted in the search key order of the index.
- (6) True False
B+ trees can include duplicate keys.
- (7) True False
B+ trees can be used to perform ranged queries.
- (8) True False
When a sparse index is used, each record must have an entry in the index.
- (9) True False
In fixed-length records, a character field of type *CHAR* is NOT allowed.
- (10) True False
Spanned records refer to records that are longer than blocks and therefore are broken into fragments.

Problem 2 (10 points) Multiple Choice

- (1) Which of the following is NOT a usual strategy used in swizzling pointers?
 - a) No Swizzling
 - b) Automatic Swizzling
 - c) Manual Swizzling
 - d) On Demand Swizzling

- (2) Which of the following best describes the Nested Block Loop Join method when joining two relations R and S?
 - a) I/O cost of is $3B(R) + 3B(S)$
 - b) Most efficient when the largest relation is used in the outer loop
 - c) Cannot be used unless one of the relations has a sparse index
 - d) I/O cost includes a quadratic term $B(R) B(S)$
 - e) Memory requirements are $B(R) + B(S) \leq M^3$

- (3) What of the following is true about static hash tables?
 - a) Hashing technique should not uniformly distribute the keys.
 - b) Static hashtables don't have any performance issues.
 - c) Performance degrades when many keys are hashed to the same bucket.
 - d) Overflow blocks are not used.

- (4) In an extensible hash indexing scheme, how many bucket array elements will point to a bucket with a local depth (or 'nub') of l ? Assume the variable d is the global depth i.e. there are 2^d bucket array elements.
 - a) 2^l
 - b) 2
 - c) 2^{l-d}
 - d) 2^{d-l}

- (5) Two-phase multi-merge sort (TPMMS) is used to sort a relation with 100 blocks. Which of the following is true?
 - a) I/O Cost is 100 and memory requirement is $M \geq 10$
 - b) I/O Cost is 200 and memory requirement is $M \geq 11$
 - c) I/O Cost is 300 and memory requirement is $M \geq 11$
 - d) I/O Cost is 100 and memory requirement is $M \geq 2$
 - e) I/O Cost is 200 and memory requirement is $M \geq 50$

Problem 4 (*16 points*) Triggers

Consider the following tables with the following inserts and triggers:

```
CREATE TABLE UserBucket (firstname VARCHAR(20), lastname VARCHAR(20),
bucket INT DEFAULT NULL);
```

```
CREATE TRIGGER UserBeforeTrigger BEFORE INSERT ON UserBucket
FOR EACH ROW
REFERENCING NEW ROW AS NewTuple
WHEN (NewTuple.firstname LIKE 'E%')
SET NewTuple.bucket=1;
```

```
CREATE TABLE BucketCount (bucketid INT, count INT);
INSERT INTO BucketCount(bucketid, count) VALUES(0, 0);
INSERT INTO BucketCount(bucketid, count) VALUES(1, 0);
```

```
CREATE TRIGGER UserAfterTrigger AFTER INSERT ON UserBucket
FOR EACH ROW
REFERENCING NEW ROW AS NewTuple
UPDATE BucketCount SET count=count+1 WHERE bucketid=NewTuple.bucket;
```

```
CREATE TRIGGER UserTableTrigger AFTER UPDATE
OF lastname ON UserBucket
REFERENCING
OLD TABLE As OldStuff
NEW TABLE AS NewStuff
BEGIN
UPDATE BucketCount SET count=(SELECT count(*) FROM OldStuff)
END
```

- (a) Show the contents of the *UserBucket* table after executing the following query: [3 points]

```
INSERT INTO UserBucket(firstname, lastname) VALUES('Evan', 'Smith');
```

- (b) Starting with the initial problem (*ie*, ignoring the queries in subproblem *a*), show the contents of the *UserBucket* table after executing the following query: [3 points]

```
INSERT INTO UserBucket(firstname, lastname) VALUES('Angela', 'Smith');
```

- (c) With the initial problem (*ie*, ignoring the queries in the previous subproblems), show the contents of the *BucketCount* table after executing the following queries: [3 points]

```
START TRANSACTION;  
INSERT INTO UserBucket(firstname, lastname) VALUES('Laura', 'Stuart');  
INSERT INTO UserBucket(firstname, lastname) VALUES('Evonne', 'Chu');  
ROLLBACK;
```

- (d) Starting with original problem (*ie*, ignoring the queries in the previous subproblems), show the contents of the *BucketCount* table after executing the following queries: [3 points]

```
INSERT INTO UserBucket(firstname, lastname) VALUES('Larry', 'Smith');  
INSERT INTO UserBucket(firstname, lastname) VALUES('Edward', 'Johnson');  
UPDATE UserBucket SET lastname='Jones' WHERE firstname='Edward' AND lastname='Johnson';
```

- (e) Explain one advantage and one disadvantage of using a trigger to enforce a constraint compared to using a CHECK constraint [4 points]

Problem 5 (12 points) B+ Tree Consider the B+ tree index of degree 3 shown in the figure below. For each problem below, you only need to show the final resulting B+ Tree.

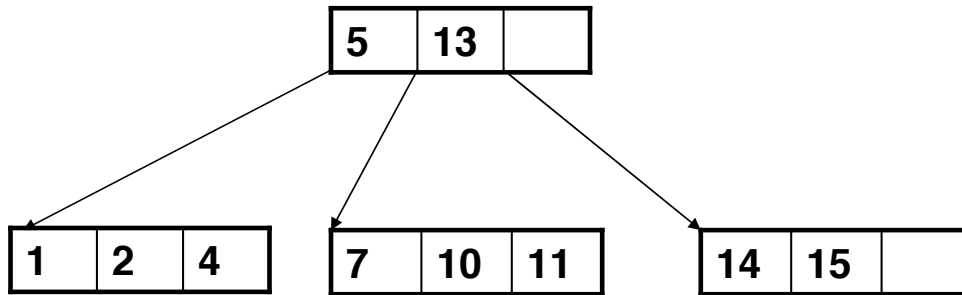


Figure 1: Problem 5

a) Show the tree that would result from inserting a data entry with key 19 into this tree. [2 points]

b) Show the B+ tree that would result from inserting a data entry with key 3 into the *original* tree. [3 points]

c) Show the B+ tree that would result from deleting the data entry with key 10 from the *original* tree. [2 points]

d) Show the B+ tree that would result from deleting the data entry with key 14 from the original tree. [3 points]

e) Explain why in a real database most relations only require B+ trees with two or three levels. [2 points]

Problem 6 (*10 points*) **Indexing** Consider indexing the following key values using a linear hash table. Here is the specification of these index structures.

The hash function $h(n)$ for key n is $h(n) = n \bmod 16$; i.e., the hash function is the remainder after the key value is divided by 16. Thus, the hash value is 4 bits. Assume that each bucket can hold 2 data items. We adopt the policy that the average occupancy of a bucket cannot exceed 85%.

Suppose that we insert the keys in the order of: 45, 36, 31, 56, 34.

(a) Draw the linear hash index after all the keys are inserted. Show how the keys and their hash values are distributed within the buckets. [6 points]

(b) Briefly explain the advantages and disadvantages of linear hash tables, in terms of the following aspects, compared to B+ Tree? [4 points]

- Insertion:

- Querying:

Problem 7 (*16 points*) Transaction Logs

A database has four elements, A, B, C, and D. Assume that the following is a normal sequence of *undo* log records, using non-quiescent checkpointing:

- 1 <start T1>
- 2 <T1,B,40>
- 3 <start T2>
- 4 <T2,A,56>
- 5 <T2,C,34>
- 6 <start T3>
- 7 <commit T1>
- 8 <T3,B,12>
- 9 <commit T2>
- 10 <T3,D,89>
- 11 <start T4>
- 12 <T4,C,7>
- 13 <T3,A,22>
- 14 <commit T4>
- 15 <T3,A,99>
- 16 <commit T3>

- (a) When is the *latest time* for transaction T1, T2 that "dirty data" can be flushed onto disk (ie, the time Output(X) for data X can be performed)? [2 points]

For T1: Output(s) _____ before Log Line ____.

For T2: Output(s) _____ before Log Line ____.

- (b) Suppose we start checkpointing right after Log 5, indicate where and what the start checkpointing record would look like. Then, indicate where and what the *earliest* end checkpoint record would look like. [2 points]

Between Log Line 5 and 6, <START CKPT(____, ____)>

Between Log Line ____ and ____, <END CKPT>

- (c) Continue from (b). Suppose the system crashes right after Log 14 and the end checkpoint has been written out to disk. What is the contents of the earliest log line we must examine? And which transaction records do we need to undo in sequence? [4 points]

Earliest Log Line Contents: <_____>, Transaction Sequence:

- (d) Now, suppose that the log is a *redo* log. When is the *earliest time* for transactions T3 and T4 that "dirty data" can be flushed onto disk? [4 points]

For T3: Output(s) _____ after Log Line ____.

For T4: Output(s) _____ after Log Line ____.

- (e) Show the contents of log line 13 if this was extended to be *undo-redo* log. Assume the values shown are the undo log entries. [2 points]

- (f) Explain one advantage of using a redo log as opposed to an undo log. [2 points]

Problem 8 (*15 points*) Query Optimization

Consider a distributed system with two sites X and Y connected through a network. Relation R(A, B, C) resides at site X and relation S(C, D) resides at site Y. The statistics of the relations are as follows:

$T(R) = 2,000$ (number of tuples in R)

$T(S) = 100,000$ (number of tuples in S)

$S_A = S_B = S_C = S_D = 10$ bytes (size of each attribute)

$V(C, R) = 200$ (number of distinct values of attribute C in R)

$V(C, S) = 2,000$ (number of distinct values of attribute C in S)

We want to compute the natural join of the two relations: $T = R \bowtie S$, and we want the resulting relation T to be stored at site X. We are given the two plans as follows:

Plan A: Copy relation S from site Y to X, compute the join at site X, and store the result at site X.

Plan B: Copy relation R from site X to Y, compute the join at site Y, and move the resulting relation to site X for storage.

a) Calculate the total number of bytes transferred between site X and Y for plan A. [4 points]

b) Calculate the total number of bytes transferred between site X and Y for plan B. [6 points]

c) Grand Challenge: Design another plan that can further reduce the amount of data transferred between the two sites than plans A and B. Briefly describe your plan and calculate the total number of bytes transferred between sites X and Y for your plan. [Hint: Try to avoid transferring the whole relation.] [5 points]

END OF CS411 FINAL EXAM