University of Waterloo
Midterm Examination

Winter, 2011

Student Name: _____

Student ID Number: _____

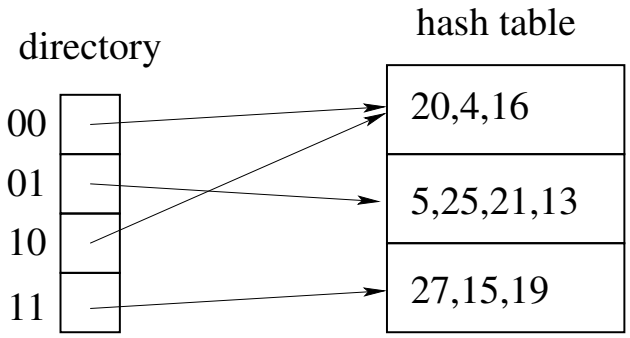
Course Abbreviation and Number	CS448
Course Title	Database Systems Implementation
Sections	01 (13:00)
Held With Courses	CS 648
Sections of Held With Courses	01
Instructor	K. Salem

Date of Exam	February 14, 2011
Time Period	19:00-21:00
Duration of Exam	120 minutes
Number of Exam Pages (including this cover sheet)	8 pages
Exam Type	Closed Book
Additional Materials Allowed	None

Question 1: (6 marks)	Question 2: (6 marks)	Question 3: (12 marks)	Question 4: (8 marks)
Question 5: (8 marks)	Question 6: (10 marks)		
Total: (50 marks)			

1. (6 total marks)

The diagram below shows an extensible hash table with four hash buckets. Each number x in the buckets represents an entry for a record for which the hashed key value is x. For example, the number 20 in the first hash bucket represents a record for which the hashed key value is 20. The maximum number of records per bucket is four.



a. (2 marks) Give a sequence of two insertions that will cause exactly one new hash bucket to be added without changing the size of the directory. An example of an insertion is ‘‘`insert 9`’’, which refers to insertion of a tuple with a hashed key value of `9`. The two hashed key values in your sequence should be distinct, and should be distinct from the values already shown in the hash table.

b. (2 marks) Give a sequence of two insertions that will not cause the hash table's directory to grow and that will not cause any new hash buckets to be added. Start from the original hash table shown in the figure above, not from the hash table that would result from the inserts from part (a). The two hashed key values in your sequence should be distinct, and should be distinct from the values already shown in the hash table.

c. (2 marks) Give a sequence of two insertions that will cause the hash table's directory to double twice. If this is not possible, write `NOT POSSIBLE`. Start from the original hash table shown in the figure above. The two hashed key values in your sequence should be distinct, and should be distinct from the values already shown in the hash table.

2. (6 total marks)

Suppose that a B+Tree index has been created for some attribute $R.x$ of a relation R.

a. (1 mark)

Assume that the B+Tree is clustered and of Type I (index leaves contain tuples of R). Consier the two rightmost leaf blocks of the B+Tree. Is it necessarily true that the these two blocks are sequential in the file or disk that contains the B+Tree? Answer `YES` or `NO`.

b. (2 marks)

Again, assume that the B+Tree is clustered and of Type I. Suppose that x_1, x_2, and x_3 are values of $R.x$, and $x_1 < x_2 < x_3$. Suppose also that the tuples corresponding to x_1 and x_3 are located on the same page, p. Is it necessarily true that the tuple corresponding to x_2 is also located on page p? Answer `YES` or `NO`.

c. (2 marks)

Repeat question (b), but this time under the assumption that the index is unclustered and of Type II (index leaves contain tuple IDs).

d. (1 mark) Which of these two types (Type I clustered, Type II unclustered) of indexes will allow the tuples of R to be retrieved in order of their $R.x$ values? Answer "`Type I clustered`", "`Type II unclustered`", "`both`", or "`neither`".

3. (12 total marks)

For this question, consider a query that performs a join of tables R and S with the join condition $R.a = S.b$. $R.a$ is the primary key of R, and there is a clustered index on $R.a$. $S.b$ is a candidate key of S, but there is no index on $S.b$. Suppose that the size of R is B blocks, that the size of S is also B blocks, and that the size of the index on $R.a$ is negligible (much less than B).

a. (4 marks)

Suppose the plan for the query is a block nested loop join of R and S, with S as the outer. M blocks of memory are available for the join operation, where $M < B$. Give an expression (in terms of B and M) for the I/O cost of this plan. Assume that the plan output is not materialized.

b. (2 marks)

Suppose instead that the plan for the query is a merge join of R and S. Tuples of R are read in $R.a$ order using the clustered index on $R.a$. S is sorted using an external merge sort, the output of which is pipelined into the join. A total of M blocks of memory are available, where $\sqrt{B} < M < B$. How much memory should be used for the merge join operator, and how much should be used for the sort operator? Briefly justify your answer.

3. (cont'd)

c. (4 marks)

Estimate the I/O cost of the plan from part (b), assuming that the M blocks of memory are allocated as you specified in your answer to that part. Assume that block writes and block reads have the same cost and that the plan output is not materialized.

d. (2 marks)

Assuming that $\sqrt{B} < M < B$, under what circumstances is the block nested loop join plan preferable to the merge join plan? Express your answer in terms of B and M, and briefly justify it.

4 (8 marks)

The pseudo-code below implements the `GetNext` method of an iterator version of tuple-oriented nested-loop join, as discussed in class.

```
Iterator state:
Iterator R;  // right (inner) child
Iterator L;  // left (outer) child
Tuple t;     // current outer tuple

GetNext() {
  tuple t2;  // current inner tuple
  if (t == NULL) t = L.GetNext();
  while (t <> NULL) {
   while((t2 = R.GetNext()) <> NULL) {
     if ( t matches t2 ) return ( t join t2 )
   }
   R.Close();
   R.Open();
   t = L.GetNext();
  }
  return(NULL);
}
```

Implement the `GetNext` method of an iterator for merge join, assuming that both the left and right inputs are already sorted on their join keys. Declare the iterator state and write your code in the style of the example above. To simplify this problem, you may assume that the *left input* (only) does not contain duplicate join keys. The right input may contain duplicates.

5 (8 total marks) Consider the hybrid hash join algorithm presented in the notes and discussed class. Assume that a total of M blocks of memory are available for the join, of which k blocks are used to build the hash table for the first partition and the remaining $M - k$ blocks are used to stage the remaining partitions to temporary files. For the purposes of this question, make the simplifying assumption that the hash table is perfectly space-efficient, i.e., k blocks worth of tuples can be stored in a hash table of size k blocks.

a. (4 marks)

What is the size of the largest build relation that can be handled by this join operator without having to recursively re-partition the build relation? Express your answer in terms of M and k, and justify your answer.

b. (2 marks)

What is the best value for k, if the goal is to maximize the size of build table that can be handled by this operator without re-partitioning? Justify your answer.

c. (2 marks)

Does memory size impose a limit on the maximum size of the probe input that can be handled (without re-partitioning) by this operator? If so, what is the maximum probe table input size (in terms of M and k) that can be handled by this operator? If not, write `NO LIMIT`.

6 (10 total marks) Provide *brief* answers to the each of the following questions. "Brief" means a sentence or two.

a. (2 marks)
What does *pipelining* mean, in the context of relational query processing?

b. (2 marks)
What is the *database physical design* problem?

c. (2 marks)
An external merge sort operates in two phases. What are the two phases, and what occurs in each phase?

d. (2 marks)
How does the *generalized clock* replacement algorithm differ from the basic clock algorithm?

e. (2 marks)
What is the *cache inclusion* problem?