NetID: _____

CS411 Database Systems

Fall 2014, Prof. Sinha

Department of Computer Science
University of Illinois at Urbana-Champaign

Final Examination

December 17, 2014
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 **16** pages. Check if you have missing pages.
- The exam is closed book and closed notes. You are allowed to use scratch papers. 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!
- Each problem has different weight, as listed below -- So, plan your time accordingly. You should look through the entire exam before getting started, to plan your strategy.

Problem	1	2	3	4	5	6	7		Total
Points	30	20	14	14	18	14	10		120
Score									
Grader									

NetID: _____

Problem 1 (30 points) Misc. Concepts

For each of the following statements, indicate whether the entire statement is TRUE or FALSE by circling your choice. If you feel the statement is partially true and partially false, choose FALSE.

You will get 2 points for each correct answer, and **no penalty** (of negative points) for wrong answers.

1. The may be multiple ways to translate an E-R diagram to a relational schema.

   TRUE FALSE True

2. A relation that satisfies 3rd normal form can never satisfy BCNF.

   TRUE FALSE False

3. For a relation R=(A,B,C,D) with functional dependency AC→BD, we can deduce the functional dependencies A→B and C→D .

   TRUE FALSE False

4. BCNF decomposition is lossless and it preserves all the dependencies.

   TRUE FALSE False

5. Relation R=(A,B), with some unknown functional dependencies, is in BCNF.

   TRUE FALSE True

6. Updates to a schema (e.g., ALTER TABLE) are generally less expensive than updates to the data (e.g., UPDATE).

   TRUE FALSE False

7. In SQL, for set operators (e.g., INTERSECT, UNION), the default semantics is to eliminate duplicates.

   TRUE FALSE True

8. While translating ER-model to a Relational Design, the NULLs approach requires the least space compared to the OO and E/R approach.

   TRUE FALSE False.

9. SELECT * FROM PERSON WHERE AGE = (SELECT AGE FROM PERSON WHERE NAME LIKE “A%”)
   The above SQL statement can sometimes throw a runtime error because the sub-query can return more than one tuple.

   TRUE FALSE True

10. If a SQL function performs grouping by use of “GROUP BY”, then the select clause must have at least one aggregation function.

    TRUE FALSE False

11. Since “Views” do not physically store tuples, a view can never be updated.

    TRUE FALSE False

12. Unclustered indexes are always dense.

    TRUE FALSE True

13. Every B+ tree index is sparse.

    TRUE FALSE False.

14. In a B+ tree index with degree d, every node (except the root) is allowed to have between 1 and 2d keys.

    TRUE FALSE False

15. In the context of two-pass physical join operators, sort based methods have a much lower memory requirement as compared to the hash based joins.

    TRUE FALSE False

16. In the context of query rewriting, duplicate elimination can always be “pushed below” projection.

    TRUE FALSE False

17. Nonquiescent check pointing is a strategy to allow pipelining of operators.

    TRUE FALSE False

18. Since pipelining saves the effort of writing intermediate results to disk, the total cost of executing a query using pipelining is always less than when materializing the intermediate results.

    TRUE FALSE True

19. In a linear hash table, the number of buckets *n* must be a power of 2.

    TRUE FALSE False

20. The “v” of <T,x,v> in Undo logging refers to the new value of element x.

    TRUE FALSE False

NetID: _____

Problem 2 (20 points) Query Languages

Consider a database of students who are registered for courses at UIUC in Fall 2014, with the following schema:

students (UID, gender, year, GPA)
register (UID, courseID, reg_datetime)
courses (courseID, department, courseName)

Part 1:
If we want to find UIDs of all students who do **not** take CS411, does the following relational algebra expression give the correct answer? Why or why not? (4 points)

$$\pi_{UID}(\sigma_{courseID \neq "CS411"}(register))$$

Solution: No. This will select all the students that have at least one course other than CS411. Such students might also take CS411.

Part 2:
Among all the students that take CS411 (they may take other courses as well), who registered last (has/have highest reg_datetime in the 'register table)? Write a relational algebra expression to find the UID(s) of such student(s). **Hint:** you may use < or > to compare reg_datetime, where an earlier reg_datetime has a smaller value. In case of ties return the UIDs of all students who were tied. (8 points)

Solution:

$$\pi_{UID}(\sigma_{courseID = 'CS411'}(register)) - \pi_{register1.UID}(register1 \bowtie_C register2),$$

where the join condition C is defined as

$$C = (register1.courseID = 'CS411') \; and \; (register2.courseID = 'CS411') \; and \; (register1.datetime < register2.datetime),$$

And the two relations, register1 and register2 are obtained by a rename operator as

$register1 = \rho_{register1}(register)$, and $register2 = \rho_{register2}(register)$.

Part 3:

Among all the students that take CS department courses, what are the average GPAs for all males (gender = 'M') and for all females (gender = 'F')? Use a single **SQL** query to answer this question. (8 points)

Solution:

```
SELECT gender, AVG(GPA)
FROM students, register, courses
WHERE students.UID = register.UID
AND register.courseID = courses.courseID AND department = 'CS'
GROUP BY gender;
```

NetID: _____

Problem 3 (14 points) Indexing

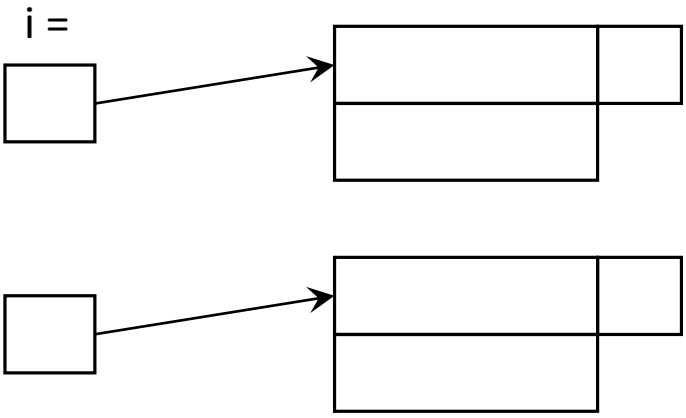
In this problem, we use an extensible hash table to index the following search key values by inserting them one by one:

22, 69, 35

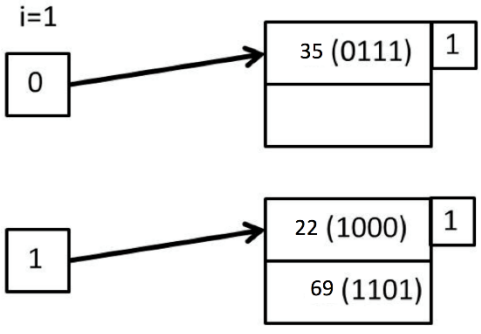
The hash function $h(n)$ for integer valued search key n is $h(n) = n \, mod \, 14$, and each data block can hold 2 data items.

Part 1:

Fill in the blanks of the following empty hash table with both the buckets and the data blocks, after these three key values have been inserted. Show the keys along with their hash values in the data blocks, e.g., 22(1000). Indicate the number of bits in the hash value that are used in the buckets (the value 'i' as discussed in the class). Also, indicate the "nub" value of each data block. Note that, since we only inserted three key values, you do not need to fill in all the blanks in the data block. (8 points)



Solution:

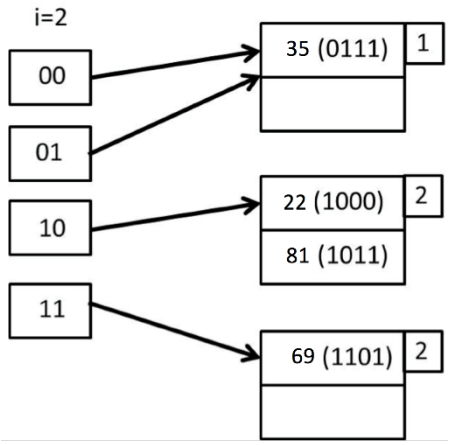


NetID: _____

Part 2:

Consider that we next insert key 81 into the hash table. Redraw the hash table in a similar format of Part 1. Indicate the number of bits in the hash value that are used in the buckets. Also, indicate the "nub" value of each data block. (6 points)

Solution:



NetID: _____

Problem 4 (14 points) Query Processing

Suppose we have two relations, $R(x, y)$ and $S(y, z)$. In this problem, we will explore different ways to join R and S on their common attribute, i.e., y. We know that R fits in 400 blocks of memory, and S fits in 300 blocks. Neither R nor S are sorted on any of their attributes. We also know that the memory buffer fits 51 blocks (M=51).

Part 1:

What is the minimum cost of joining R and S using a block-nested loop join? (6 points)

Answer:

In general, Cost = B(R1) + B(R1)*B(R2) / (M-1), and this is smaller when B(R1) < B(R2) (otherwise, swap R1 and R2)

In this question, B(R) > B(S), so we will use R1=S, R2=R

Thus we have Cost = B(S) + B(S)*B(R) / (M-1) = 300 + 300*400 / 50 = 2700
(Note: answers using M instead of M-1 are also acceptable)

Part 2.1:

Suppose we want to join R and S using a hash-based join. Is this possible? Recall there are constraints that must be met, relating $B(R)$ and $B(S)$ to the amount of available memory blocks, M. What are these constraints, and are they met? (4 points)

Answer:

The constraints are B(R) <= M^2 or B(S) <= M^2 (also acceptable to use (M-1)^2 or M(M-1))

In this case, we verify that 400 < 50^2 and 300 < 50^2, so both constraints can be satisfied.

So yes, it's possible to do a hash-based join.

Part 2.2:

If a hash-based join is possible, what is the cost of that join? If you discovered above that it's not possible, just state that fact. (4 points)

Answer:

A hash-based join is possible. The formula is 3*(B(R) + B(S)) = 3(400+300) = 2100

NetID: _____

Problem 5 (18 points) Query Optimization

Part 1:

Consider three relations $A(w,x)$, $B(x,y)$, and $C(y,z)$. Provide a formula for estimating the number of tuples in $A \bowtie B \bowtie C$ in terms of the number of tuples in each relation ($T(A), T(B), T(C)$) and the sizes of their value sets (e.g., $V(A,x)$ etc.). [**Hint**: a) We have seen in class a formula for estimating the size of the join of two relations. Use it. b) Use the "preservation of value sets" assumption.] (4 points)

Answer:

The order in which the joins are executed doesn't affect the final table's size. So we can compute $T(A \bowtie B \bowtie C)$ as $T((A \bowtie B) \bowtie C)$.

$$T((A \bowtie B) \bowtie C) = T(A \bowtie B) * T(C) / max(V(A \bowtie B, y), V(C, y)) \qquad [1]$$

Here, we know that $T(A \bowtie B) = T(A) * T(B) / max(V(A,x), V(B,x))$

And, from the "preservation of value sets" assumption, we know that $V(A \bowtie B, y)$ can be estimated as $V(B,y)$

Substituting into [1], we get

$$T((A \bowtie B) \bowtie C) = T(A \bowtie B) * T(C) / max(V(A \bowtie B, y), V(C, y))$$

$$= T(C) * [T(A) * T(B) / max(V(A,x), V(B,x))] / max(V(B,y), V(C,y))$$

$$= T(A) * T(B) * T(C) / (max(V(A,x), V(B,x)) * max(V(B,y), V(C,y)))$$

NetID: _____

Part 2:

We want to join the following 4 tables in the most efficient order.

A(w, x)	B(x, y)	C(y, z)	D(z, k)
T(A) = 1000	T(B) = 300	T(C) = 500	T(D) = 3000
V(A, w) = 50	V(B, x) = 20	V(C, y) = 250	V(D, z) = 200
V(A, x) = 100	V(B, y) = 100	V(C, z) = 15	V(D, k) = 150

Some hints:

- T(R) refers to the number of tuples in the relation R
- V(R, s) refers to the number of distinct values of attribute s in R.
- Only attempt to join relations where a natural join is feasible, i.e., if the relations have at least one common attribute.
- The cost of a plan is the total size of all the intermediate tables used in that plan, as discussed in class.

Fill out the dynamic programming table entries in the table below.(14 points)

Sub query	**Size**	**Cost of best plan**	**Best plan**
AB	3,000 (=300*1000/100)		AB
BC	600 (=300*500/250)		BC
CD	7,500 (=500*3000/200)		CD
ABC	6,000 (=1000*600/100) OR (=3000*500/250)	600	A(BC)
BCD	9,000 (=600*3000/200) OR (=300*7500/250)	600	(BC)D
ABCD	90,000 (=6000*3000/200) OR (=3000*7500/250)	6,600 (=6,000+600)	(A(BC))D

NetID: _____

Problem 6 (18 points) Query Optimization

Consider the physical query plans for the following expression:

$$(R(w,x) \bowtie S(x,y)) \bowtie U(y,z)$$

We have the following assumptions:

- B(R) = 5,000, B(S) = B(U) = 12,000
- The intermediate result $R \bowtie S$ occupies k blocks for some k.
- Both joins are implemented as hash-joins, either one-pass or two-pass, depending on k.
- The memory buffer has 101 blocks (M = 101).
- The final disk writes performed by a plan does not count towards its cost.

Consider three cases regarding the range of k, compute the cost of the best physical query plan for each case and fill in the blanks in the table below.

Range of k	**Pipeline/Materialize**	**Algorithm for final join**	**Total Disk IO's**
k <= 50	(A)	(D)	(G)
50 < k <= 5000	(B)	(E)	(H)
K > 5000	(C)	(F)	(I)

Show your calculation for the blanks (G)-(I).

Calculation for (G)

Calculation for (H)

Calculation for (I)

NetID: _____

Problem 7 (10 points) Failure Recovery

Consider the following log sequence.

Log ID	Log
1	<START T1>
2	<T1, A, 10>
3	<START T2>
4	<T2, B, 15>
5	<T1, C, 20>
6	<START T3>
7	<T1, B, 25>
8	<COMMIT T1>
9	<T3, C, 30>
10	<COMMIT T2>
11	<START T4>
12	<T3, D, 35>
13	<T4, A, 40>
14	<COMMIT T3>
15	<T4, C, 45>
16	<COMMIT T4>

NetID: _____

Part 1:

For the questions (a)-(c), assume the given log sequence is a ***UNDO*** log.

(a) Briefly explain the meaning of the record <T2, B, 15> (Log ID 4) (1 point)

(b) When is the latest time for transaction T3 that "dirty data" can be flushed onto the disk (i.e., the time Output(X) for data X can be performed)? (1 point)

Output(s) _____ before Log ID _____

(c) Suppose we insert checkpoint in the following way:

Between Log ID 7 and Log ID 8, < START CKPT (T1, T2, T3) >

If the system crashes right after Log ID 13, show which transaction needs to be undone in correct order. (3 points)

Part 2:

For the questions (d)-(f), assume the given log sequence is a ***REDO*** log.

(d) Please briefly explain the meaning of the record: <T2, B, 15> (Log ID 4) (1 points)

(e) When is the earliest time for transaction T3 that "dirty data" can be flushed onto disk (i.e. the time Output(X) for data X can be performed)? (1 points)

For T3: Output(s) _____ before Log ID _____

(f) Suppose we insert checkpoint in the following way:

Between Log 9 and 10, < START CKP T(T2, T3) >

Between Log 13 and 14, < ENDCKP T >

If the system crashes right after Log ID 14, show which transaction needs to be redone in correct order. (3 points)