NetID: _____

CS411 Database Systems

Fall 2015, Prof. Sinha

Department of Computer Science
University of Illinois at Urbana-Champaign

Final Examination
December 15, 2015
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 **15** pages (last page blank - use as scratch paper if you wish). 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	6	20	16	17	15	8	112
Score								
Grader								

Problem 1 - Misc. Concepts (30 points)

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 1.5 points for each correct answer, –0.5 point for each incorrect answer, and 0 point for each answer left blank.

1. The aggregation method SUM in SQL is applied on a row rather than a column.

False

2. In SQL DISTINCT is used to eliminate duplication.

True

3. MIN() and MAX() SQL operators can only be used together with the GROUP BY statement.

False

4. HAVING <condition> may follow a GROUP BY clause.

True

5. A materialized view is a relation (query result) stored on disk.

True

6. In SQL, suppose an insert or update to relation R leads to a 'value not found in relation S' error. This may be because of a foreign key constraint violation.

True

7. 'Check' constraint is equivalent to foreign key constraint.

False

8. A TRIGGER in SQL may execute an action even when no event (such as insert, delete or update) happened.

False

9. Unclustered indexes must be dense.

True

10. One of the advantages of B+ Tree over Hash Table is that it can be used to quickly perform range queries.

True

11. Extensible Hash Table uses binary hash key.

True

12. We cannot do Selection using a One Pass Algorithm.

False

13. Two-Pass Algorithms are used when relations are too large to fit in memory.

True

14. Tuple-based nested loop join has higher IO cost than block based nested loop join in general.

True (Tuple based nested loop join cost: T(R) T(S). worst case block based nested loop join cost: B(S) + B(S)B(R))

15. The expected cost of $\sigma_{a=m}$(R) for index-based selection, with a clustered index on attribute 'a', is approximately: T(R)/V(R,a). (T(R) = number of tuples in R, V(R,a) = number of distinct values of the attribute a in R)

False (B(R)/V(R,a))

16. Relational algebra operation: $\pi_M(\pi_N(R))$ = $\pi_{M \cap N}$ R

True

17. To materialize an intermediate result means to store the result on disk.

True

18. Given relations R(a,b) and S(a,c), preservation of value sets states that V(R $\bowtie_A$ S, b) = V(R, b)

True

19. In the Nonquiescent checkpoint scheme, the system will wait for all current transactions to complete before logging the 'end checkpoint'.

False

20. In Undo logging, OUTPUTs are done “early”, i.e., before commit.

True

Problem 2 - Query Languages (6 points)

Your SQL query should work in general. Consider the following table schemas:

Salesperson:

Salesperson_id (INT)	Sname (TEXT)	Age (INT)	Salary (INT)

Customer:

Cust_id (INT)	Cname (TEXT)	City (TEXT)	Industry_type (VARCHAR(2))

Orders:

Number (INT)	Order_date (DATE)	Cust_id (INT)	Salesperson_id (INT)	Dollar_amount (INT)

- The second row in the parenthesis is the type of the corresponding attribute

Write a SQL query to find the largest order amount handled by each salesperson, and print the dollar amount of each such order along with the salesperson name and id. (6 points)

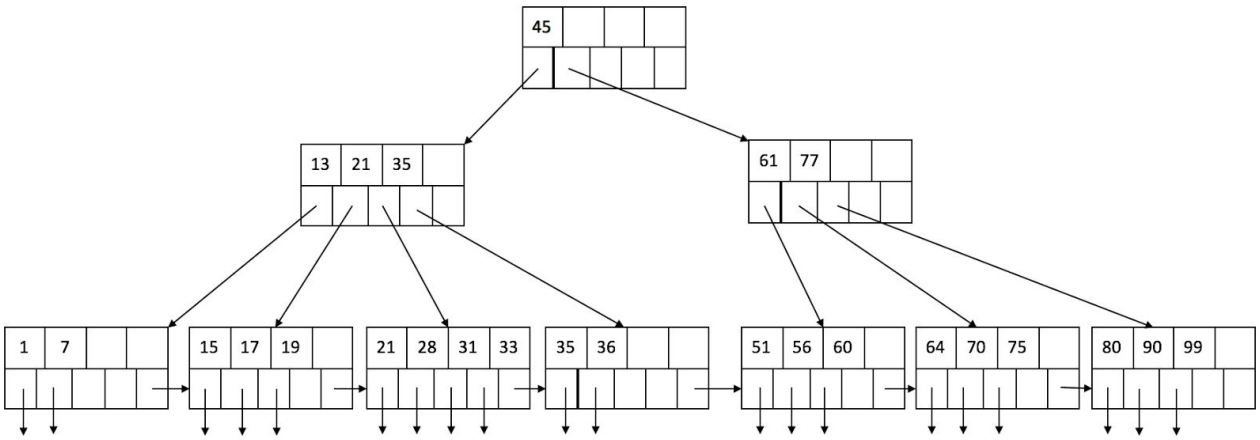
Solution

```
SELECT Salesperson_id, Sname, Dollar_amount
FROM Salesperson JOIN (
        SELECT Salesperson_id, MAX( Dollar_amount )
        FROM Orders
        GROUP BY Salesperson_id)
```

NetID: _____

Problem 3 - Indexing (20 points)

Consider the following B+Tree with degree d=2, which means each node except the root has between 2 and 4 keys, as shown in Figure below:



B+ Tree

(a) There are currently 20 records in this tree (each leaf node can point to up to 4 records, and its neighboring leaf node). What is the maximum number of records that could be removed so that the resulting B+ tree is still valid and the height does not change? (6 points)

Solution:
Minimum nodes in pointed by root is 2, each must have 2 keys, meaning 3 pointers to leaf level; each leaf node must have 2 records.
Thus total records after: 2*3*2 = 12 records. 20 - 12 = 8 records.

(b) What will happen when we insert value 37, 38, 40, in that order? Explain the intermediate steps (i.e., result of each of the three insertions) and show the part of the **final** tree (after inserting key 40) that has changed from the above form. Assume rebalance may only happen at leaf nodes. (8 points)

Solution:
Insert 37: leaf node [35, 36] becomes [35, 36, 37]
Insert 38: leaf node [35, 36, 37] becomes [35, 36, 37, 38]
Insert 40: [35, 36, 37, 38, 40], need to split [35, 36, 37, 38, 40], two cases:
case1: [35, 36] and [37, 38, 40],
case2: [35, 36, 37] and [38, 40]
And accordingly we need to add a key in the above node: for case 1: 37, for case 2: 38

(c) Consider the original tree, what happens when you delete 36? show the part of the tree that has changed from the original tree. (6 points)

Solution:
rebalance result: [21,28,31], [33, 35] OR [21,28], [31, 33, 35],
the resulting parent node is 33 OR 35 respectively .

Problem 4 - Query Processing (16 points)

Consider two relations, R(w,x) and S(x,y). We want to join them on their common attribute. R fits into 200 blocks and S fits into 1500 blocks. Neither R nor S are sorted on any of their attributes.

You do not need to describe the steps of the algorithm (unless explicitly told to do so), just writing down the formula used and plugging in correct values is sufficient.

(a) Assume the memory buffer has 51 blocks available (M=51). What is the cost of joining R and S on x using a block-based nested loop join. Use the relation with less blocks in outer loop. Write the formula of your cost computation, and plug in relevant values in the formula. You do not need to compute the result. (4 points)

Solution: B(R) + B(R) *B(S) / (M-1) = 200 + 200 * 1500 / (51 - 1)

(b) Assume again the memory buffer has 51 blocks available (M=51). We would like to join R and S on x using a two-pass sort-based join. Is it possible? Do we meet the memory constraints for a two-pass sort-based join here? Show formula and the calculation to support your claim. (4 points)

Solution: B(S) > B(R), so we use B(S) to see if we meet constraint: B(S) <= M^2
We meet the constraint since 1500 < 50^2 = 2500. So we can do two-pass sorted join.

(c) When a two-pass sorted join is possible for arbitrary relations R and S, we know that the total IO cost for it is 5 * (B(R) + B(S)). Briefly explain how this formula is derived. (5 points)

Solution: The cost in sort phase is 4(B(R) + B(S)) since we need a **2 pass sort**:

in the first pass we sort the B(R) / (M - 1) partitions and B(S) / (M - 1) partitions, this costs 2*(B(R) + B(S)) (read and write).

Second pass sort all the partitions of R and S (by incrementing pointers) and this costs 2*(B(R) + B(S)) (read and write).

Sort phase costs 4(B(R) + B(S)) total.

Now we have sorted R and S, increment pointers based common attribute will give use the desired join result. B(R) + B(S) cost is required to read.

Total: 5 * (B(R) + B(S)) cost

(d) Suppose a new relation P does not have sorted index, what is the cost of doing a sorted table scan? Write the result in terms of B(P). (3 points)

Solution: 3B(P) cost.

Problem 5 – Query Optimization (17 points)

(a) Suppose we have to execute the query R1 ⋈ R2 ⋈ .... ⋈ Rn , and we plan to do so using a left deep join tree. Show what this means by either indicating the join order by placement of parentheses or by drawing the join tree. (3 points)

Solution: (((R1 ⋈ R2) ⋈ R3) .... ⋈ Rn)

(b) Given relations R(A, B, C, D) and S(E, F, G), then, $\sigma_{G=10}(R \bowtie_{D=E} S)$ = ? Choose the correct equivalent expression that has been optimized using the 'push selections down' heuristic from the following: (2 points)

(i) $\sigma_{G=10}$ (R) $\bowtie_{D=E}$ $\sigma_{G=10}$ (S)
(ii) R $\bowtie_{D=E}$ $\sigma_{G=10}$ (S)
(iii) $\sigma_{G=10}$ (R) $\bowtie_{D=E}$ S

Solution:
(ii) is the answer (since S contains attribute G.)

(c) Consider four relations A(i, j), B(k, l), C(j, m), D(k, m) with the following statistics:

A(i, j)	B(k, l)	C(j, m)	D(k, m)
T(A) = 200	T(B) = 100	T(C) = 400	T(D) = 500

-R(x, y) means relation R has attribute x and y. T(R) refers to number of tuples in relation R.

Use the dynamic programming algorithm to find the optimal cost of $A \bowtie B \bowtie C \bowtie D$, recording all intermediate steps in the table below. Do not consider joins that amount to performing a Cartesian product. (12 points)

– Use formula $T(R1 \bowtie R2) = T(R1) * T(R2)*0.01$ for size estimation:, where R1 and R2 are any two relations.
– Stick to joining only those relations where a natural join is feasible.
– The cost of a plan is the total size of intermediate tables used by the plan.
– Assume that the join operation is symmetric, i.e., the plan (R1 R2) is the same as the plan (R2 R1).

Solution:

Subset	Size	Best Plan Cost	Best Plan
AC	800	0	AC
BD	500	0	BD
CD	2000	0	CD
ACD	4000	800	(AC)D
BCD	2000	500	(BD)C
ABCD	4000	2000 OR 1300	A(BCD) OR (AC)(DB)

Problem 6 - Query Optimization: pipelining vs. materializing (15 points)

Consider the physical query plans for the following logical plan:
(R(a, b) ⋈ S(b, c)) ⋈ U(c, d)

We have the following assumptions:

1. B(R) = 400, B(S) = 8000, B(U) = 10000, where B(R) means number of blocks needed for relation R

2. The intermediate results of A ⋈ B occupies k blocks, for some value of k

3. Both joins will be implemented as hash joins (either one-pass or two-pass, depending on k)

4. The memory has 41 blocks (M = 41)

5. We have a perfect hash function, which can create k buckets of size N/k each for any table of N tuples and for any k.

6. As per usual convention (followed in class), writes of the final result to the disk are not counted in the "total disk I/O" calculations.

Consider three cases regarding the range of k and compute the cost of the best physical query plan for each case, to fill in the blanks (A)-(I) in the following table. If you choose the two-pass algorithm for the final join, specify the number of buckets in the fields of the third column.

Range of k	Pipeline/Materialize	Algorithm for final join (one-pass or two-pass)	Total disk I/O's (may include variable 'k' in your answer)
k < 30	(A)	(D)	(G)
30 ≤ k ≤ 1200	(B)	(E)	(H)
1200 < k	(C)	(F)	(I)

Also, in next page:

1. show calculations of (G), (H) and (I). Providing the formula with values substituted is sufficient for (G), (H) and (I).
2. Explain how you arrive at formulas in (G) and (H).

(Workspace for Problem 6)

Solution:

Range of k	Pipeline/Materialize	Algorithm for final join	Total disk I/O's
k < 30	Pipeline	one-pass	(G) 3 * (400 + 8000) + 10000
30 <= k <= 1200	Pipeline OR Materialize	30 buckets, two pass	(H) 3 * (400 + 8000 + 10000) + 2k
1200 < k	Materialize	40 buckets, two pass	(I) 3 * (400 + 8000 + 10000) + 4k

(G)
1. **two pass hash joins**: read relation L, M and write to hash buffers, finally read again, each costs B(R) + B(S)
2. B(U) comes from one pass read of relation U.

(H)
1. 3 * (B(R) + B(S)) for **2 pass hash joins**, k cost for joined result goes to disk.
2. then 3B(U) for 2 pass hash for P (read, write, read)
3. another k cost for final join.

Problem 7 - Failure Recovery (10 points)

Consider the following log sequence for an UNDO logging scheme.

Log ID	Log
1	<START T1>
2	<START T2>
3	<T2, A, 1>
4	<COMMIT T2>
5	<T1, A, 2>
6	<START T3>
7	<START T4>
8	<T1, B, 3>
9	<T1, B, 4>
10	<T3, C, 5>
11	<COMMIT T1>
12	<T3, A, 6>
13	<T4, B, 7>
14	<COMMIT T3>
15	<COMMIT T4>
16	<START T5>
17	<T5, D, 8>
18	<T5, D, 9>
19	<T5, E, 10>

(a) Suppose the system crashes right after logID 11, i.e., logID 12 onwards is not present in the log file that the recovery manager finds. In the space below, indicate which transactions need to be undone. (just indicate their log ids). (2 points)

log 10

(b) Suppose we want to start non-quiescent checkpointing right after logID 13. In the space below, indicate where the start checkpointing record would be (i.e., between which two consecutive logIDs), and what it would look like (i.e., what information the log entry would include). Then, indicate where the earliest end checkpoint record (for this checkpoint) could be. (3 points)

Between logID 13 and 14, ⟨START CKPT(T3, T4)⟩
Between logID 15 and 16, ⟨END CKPT⟩

(c) Continue from (b). Suppose the system crashes right after logID 19. What is the portion of the log the recovery manager would need to inspect and which transactions would need to be undone? (3 points)

1. Log portion needs to be inspected: from crash point until ⟨START CKPT⟩
2. Transactions need to be undone: T5

(intentionally blank - scratch paper)