CS411: Database Systems **Spring 2016**

Final - Solutions

Objective-Type Questions - 20 points

1. [2] If X and Y are entity sets, then a relationship between them is always a subset of $X \times Y$.

   [✓ True] False

2. [2] The theta join is equivalent to a natural join followed by the selection operation.

   True [✓ False]

3. [2] Since two-attribute relations are always in BCNF, we can break any relation into two-attribute relations to obtain a BCNF decomposition.

   True [✓ False]

4. [2] For any relation, any of its 3NF decompositions will always preserve its functional dependencies whereas some of its BCNF decompositions may not.

   [✓ True] False

5. [2] The projection operator is faster in bag semantics than in set semantics.

   [✓ True] False

6. [2] Extensions in linear hash tables can be costly and disruptive, because after an extension the extended table or the hash table may no longer fit in memory.

   True [✓ False]

7. [2] For large relations, the duplicate elimination operation (DISTINCT) can be processed in one-pass by maintaining the set of unique values in memory.

   True [✓ False]

8. [2] Say a relation R(A, B) has T tuples, and has m distinct values of A. Then the size following a selection with condition (A > 3) on R is between 0 and T - m + 1.

   True [✓ False]

9. [2] In undo-logging, if transaction T modifies X, then the <T,X,V> entry must be written to the log and the log must be flushed to the disk before X is written to disk.

   [✓ True] False

10. [2] If we perform recovery for the second time in undo logging, we should ensure to not perform the same undo command twice since this may lead to inconsistent data.

   True [✓ False]

SQL - 15 points

Consider the following relational schema for a Movies database:

- Actor(actor_id, first_name, last_name, gender)
- Acting(actor,movie)
- Movie(movie_id, name, released_date, length, genre)

You can make the following assumptions:

- Gender in the Actor table can be either 'F' or 'M'
- actor and movie in the Acting table are foreign keys to actor_id in the Actor table and movie_id in the Movie table respectively. The two attributes also form the joint Primary Key of the Acting table.

Specify the following queries using SQL:

1. [4] What is the average length of all the action movies?

   Answer: SELECT AVG(length) FROM Movie WHERE type = 'action'

2. [4] How many actresses does the movie 'The Godfather' have?

   Answer: SELECT COUNT(*) FROM Actor, Acting, Movie WHERE Actor.actor_id = Acting.actor AND Movie.movie_id = Acting.movie AND Movie.name ='The Godfather' AND Actor.gender = 'F'

3. [7] Consider relations R(A,B) and S(B,C). Are the following two SQL queries equivalent on all databases? If not, give a simple counterexample.

```
select R.A
from R,S
where R.B = S.B and S.C = 4

select R.A
from R
where R.B in (select S.B from S where S.C = 4)
```

   Answer: No they are not equivalent.

   R: (1, 0) S: (0, 4; 0, 4)

Indexing - 15 points

Consider the B+ tree shown in Figure 1, and show the effect of the following operations, **in sequence**.

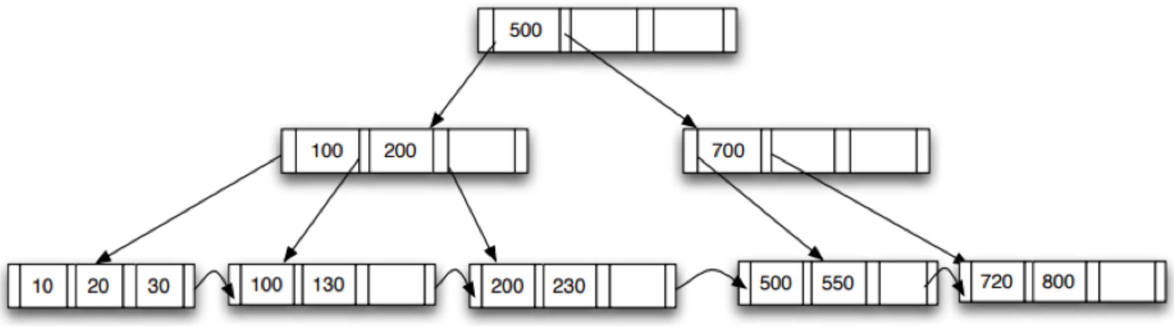

Figure 1: Original B+ tree

1. [5] Insert 80 into the B+ tree shown in Figure 1, and show the updated B+ tree below.

   Answer: Refer Figure 2

2. [5] Insert 250 to the updated B+ tree from question 1, and show the updated B+ tree below.

   Answer: Refer Figure 3

3. [5] Insert 280 to the updated B+ tree from question 2, and show the updated B+ tree below.

   Answer: Refer Figure 3

Query Execution - 15 points

1. Consider three relations R(w,x) , S(x,y) and U(y,z). It is given that B(R) = 1000, B(S) = 2000, and B(U) = 3000. Assume that all the relations are clustered. Imagine that you

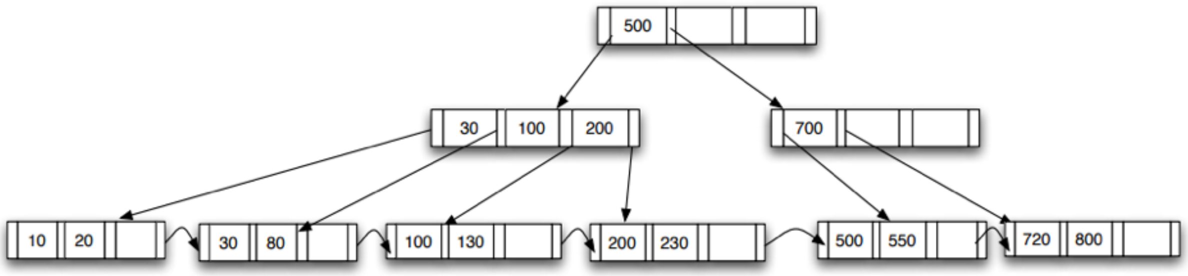

Figure 2: B+ tree after the insertion of 80

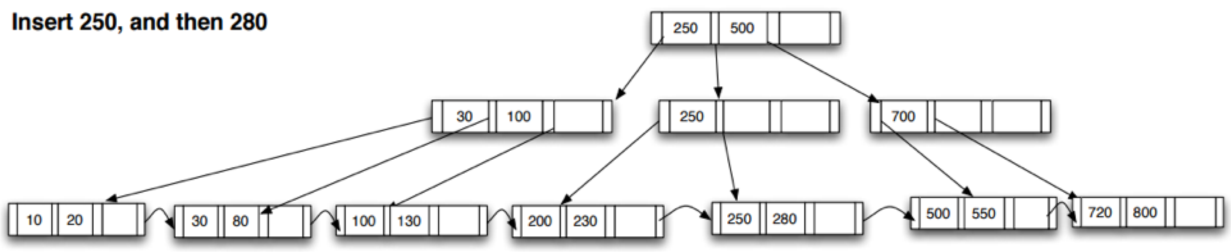

Figure 3: B+ tree after the insertion of 250, 280

have completed the Dynamic Programming algorithm for query optimization, and you have identified that the best logical plan is $(R \bowtie S) \bowtie U$. Now, you need to determine the best physical plan: choice of join operators, and pipelining/materialization, taking memory considerations into account.

For each of the following scenarios, compute the lowest cost (in terms of I/O operations) required to compute $(R \bowtie S) \bowtie U$. In each case, provide a succinct justification by providing details of the physical plan (join algorithm, pipelining/materialization) you would use.

(a) [5] Infinite memory availability. That is, assume you have no constraints on M at all.

Answer: First load all relations to memory, and then perform the join.

(b) [5] Highly constrained memory. Specifically, assume that M = 3, so you can hold only 3 blocks in memory at a time.

Answer: Perform block-based nested loop join twice. First for the inner join and then for the outer join.

Note: All reasonable answers with clear assumptions were accepted.

2. [5] Consider two relations r and s. It is given that the relation r has **more** number of tuples than relation s. For each of the following questions, **circle** all the correct options. Note: more than one may be correct.

(a) If relation s is treated as the outer table in the block nested loop join, which of the following are true:

i. cost of the join operation is increased
ii. number of iterations is reduced
iii. this approach is more optimal than having relation r as the outer table
iv. relation s needs to fit in memory

(b) If relation s is treated as the build relation in hash join, which of the following are true:

i. cost of the join operation is reduced

ii. cost of the join operation remains unchanged if both relations can simultaneously fit in memory

iii. relation r needs to fit in memory

iv. relation s needs to fit in memory

Answer: (a): ii, iii; (b): ii, iv

Query Optimization - 20 points

1. Assume five relations A, B, C, D, E, having only one common attribute l; note that they may have other attributes themselves. Answer the questions below under the following assumptions: "containment of values" and "preservation of value sets".

T(A) = 2000	T(B) = 1000	T(C) = 6000	T(D) = 5000	T(E) = 3000
V(A, l) = 500	V(B, l) = 400	V(C, l) = 3000	V(D, l) = 2500	V(E, l) = 1500

(a) [5] Estimate the number of tuples returned as a result of the following expression:

$$((A \bowtie B) \bowtie (C \bowtie D)) \bowtie E$$

Provide details of your calculation. (That is, start with the innermost parenthesis, and work your way out, arguing how you obtained each step.

Answer:

$T(A \bowtie B) = T(A) * T(B)/max(V(A,l), V(B,l)) = 2000 * 1000/500 = 4000$

$V(A \bowtie B, l) = min(V(A,l), V(B,l)) = 400$

$T(C \bowtie D) = T(C) * T(D)/max(V(C,l), V(D,l)) = 6000 * 5000/3000 = 10000$

$V(C \bowtie D, l) = min(V(C,l), V(D,l)) = 2500$

$T((A \bowtie B) \bowtie (C \bowtie D)) = 4000 * 10000/2500 = 16000$

$V((A \bowtie B) \bowtie (C \bowtie D), l) = min(V(A \bowtie B, l), V(C \bowtie D, l)) = 400$

$T(((A \bowtie B) \bowtie (C \bowtie D)) \bowtie E) = 16000 * 3000/1500 = 32000$

(b) [5] Estimate the number of tuples returned as a result of the following alternative expression:

$$(A \bowtie (B \bowtie (C \bowtie (D \bowtie E))))$$

Provide details of your calculation. Using the results of (a) and (b), what conclusions can we draw?

Answer:

$T(D \bowtie E) = 5000 * 3000/2500 = 6000$

$V(D \bowtie E, l) = 1500$

$T(C \bowtie (D \bowtie E)) = 6000 * 6000/3000 = 12000$

$V(C \bowtie (D \bowtie E), l) = 1500$

$T(B \bowtie (C \bowtie (D \bowtie E))) = 1000 * 12000/1500 = 8000$

$V(B \bowtie (C \bowtie (D \bowtie E)), l) = 400$
$T(A \bowtie (B \bowtie (C \bowtie (D \bowtie E)))) = 2000 * 8000/500 = 32000$
Conclusion: No matter which order you join the relations, the containment/preservation assumptions lead to the same answer.

2. [10] Consider the following relations and assumptions about the number of blocks in each relation:

   B(A) = 300

   B(B) = 400

   B(C) = 25

   B(D) = 500

   Use the dynamic programing algorithm as taught in the class to determine the optimal plan to join the above relations efficiently. Show your work by completing the following table (each step in the dynamic programming algorithm should be one row):

   Details: when doing the calculations, you should

   (a) Use the following formula of size estimation as discussed in the class: $B(R1 \bowtie R2) = B(R1) * B(R2) * 0.01$, where R1 and R2 are any two relations.

   (b) Assume that the join operation is not symmetric, i.e., the plan (R1 R2) is not the same as the plan (R2 R1). When choosing between these two plans, select the one that has the smaller table on the left.

Subset	Size	Lowest Cost	Lowest-cost Plan
AB	1200	0	AB
AC	75	0	CA
AD	1500	0	AD
BC	100	0	CB
BD	2000	0	BD
CD	125	0	CD
ABC	300	75	(CA)B
ABD	6000	1200	D(AB)
ACD	375	75	(CA)D
BCD	500	100	(CB)D
ABCD	1500	375	((CA)B)D

Transaction Management - 15 points

Consider the transaction log provided below in Table 1, and answer the questions that follow.

Note: For questions 1–4, assume the given log sequence is an UNDO log.

LogID	Log
1	<START T1>
2	<T1, A, 4>
3	<START T2>
4	<START T3>
5	<T3, B, 7>
6	<T2, C, 9>
7	<T3, G, 15>
8	<T2, H, 13>
9	<ABORT T3>
10	<T2, D, 9>
11	<START T4>
12	<T4, J, 13>
13	<T1, K, 12>
14	<COMMIT T1>
15	<T4, E, 3>
16	<T4, F, 9>
17	<COMMIT T4>

Table 1: Transaction log

1. [2] What is the latest time—i.e., before which LogID—for transaction T1 to output variable A onto disk? Give the LogID for your answer.

   Answer: Right before LogID 14

2. [2] At the end of the log, what is the value of Variable B?

   Answer: It is 7 since T3 was aborted

3. [3] Suppose we want to start check-pointing after LogID 9. Between which two log records would we start check-pointing, and what should the log entry (for check-pointing) look like? At the earliest time possible, between which two log records should we stop check pointing, and what should the log entry look like?

   Answer: Between LogID 9 and 10, run <STARTCKPT(T1,T2)>. After T2 commits, run <ENDCKPT>

4. [3] Given the check points you have created in Question 3, suppose the system crashes right after LogID 13. Show which transactions/actions (e.g.: <T1, A, 15>) need to be undone and in what order.

   Answer: We need to undo in this sequence: <T1, K, 12>, <T4, J, 13>, <T2, D, 9>, <T2, H, 13>, <T2, C, 9>, <T1, A, 4>

Note: For questions 5–6, assume the given log sequence is a REDO log.

5. [2] What does the record, <T4, J, 13> (which is at LogID 12), indicate? Consider the original log from Table 1.

Answer: Transaction T4 changes the value of J to 13.

6. [3] Suppose the system crashes right after LogID 15. Show which transactions/actions (e.g.: <T4, J, 13>) need to be redone and in what order. Consider the original log from Table 1.

   Answer: We need to redo in this sequence: <T1, A, 4>, <T1, K, 12>.