Name	NetID

CS411: Database Systems **Spring 2017**

Midterm 2, May 1

- READ THESE INSTRUCTIONS CAREFULLY BEFORE YOU START. DO NOT turn this page UNTIL the proctor instructs you to.
- First: write your name and NetID at the top of all the sheets.
- The exam lasts for 90 minutes, i.e., from 8–9.30am.
- We will not answer any questions during the exam. If you need to make any assumptions for any of the questions, please feel free to do so and then clarify the assumption in your answer.
- The examination contains both objective type (True/False) and long answer questions. There are 10 objective type questions, and 4 long answer questions. All questions are compulsory.
- For the objective type questions, **please circle the right answer, and provide a short description justifying your choice**. If you need extra space for any calculations, feel free to use the back side of each page.
- For the long answer questions, please answer in the space provided; if you need more space, feel free to use the back side of each page. You should not need more space, and we will not provide more space, so please use your space wisely. Show all necessary steps as part of your calculation to get partial credit.
- The maximum score you can obtain is $25 + 15 + 15 + 30 + 15 = 100$.
- You must stop writing when time is called by the proctors.
- **Cheating: No.**

Question	1	2	3	4	5	Total
Points						

Objective-Type Questions - 25 points

Please **circle** the right answer, and provide a short 1–2 line description justifying your choice.

1. [1] You know the answer to this question.

   Answer: True. Marking False would result in a paradox.

2. [3] Consider the following three join mechanisms taught in class: 1) block-based nested loop join, 2) one-pass hash join, and 3) two-pass hash join. The memory requirements of these mechanisms are in the following order: `requirement`$(1) \leq$ `requirement`$(3) \leq$ `requirement`(2).

   Answer: True

3. [3] For undo logging, if `<ABORT T>` is seen in the log on disk, then transaction `T` has no dirty data on disk, i.e., all of the changes made by transaction `T` have been written to disk.

   Answer: False

4. [3] Consider a relation $R(A, B, C)$, with two functional dependencies $A \rightarrow BC$, $B \rightarrow C$. Suppose we decompose R into $R_1(A, B)$, and $R_2(B, C)$, we can no longer determine if the functional dependency $A \rightarrow C$ holds in R, using R_1 and R_2, and therefore the dependency $A \rightarrow C$ is not preserved.

   Answer: False.

5. [3] Consider the following algebraic law which we know is valid for set semantics: $\sigma_C(R \cup S) = \sigma_C(R) \cup \sigma_C(S)$. This law remains valid for bag semantics as well.

   Answer: True

6. [3] We always prefer bushy join trees over left/right-deep join trees. This is because in left/right-deep join trees, we incur more cost because we have more intermediate join operations to compute.

   Answer: False

7. [2] While estimating the size of a join operation between two relations R and S, with the join attribute as A, we implicitly assume that all of the values of A in one of the two relations, is present in the other.

   Answer: True

8. [2] Unclustered indices are always dense, in the sense that every unique search key has an entry in the index.

   Answer: True

9. [3] B-trees provide better response time for range queries than hashing-based indexing schemes.

   Answer: True

10. [2] For a given relation that is not in BCNF, there is always a decomposition of the given relation into two relations which are in BCNF—this decomposition may not be unique.

    Answer: False

SQL - 15 points

You are the owner of a pet hotel. Since you are so smart and hard working, the size of your hotel has grown so large that it is impossible to keep track of pets' information with handwritten logs anymore and therefore you decide to use a relational database to help you manage your hotel data.

Joey, the almighty SQL madman, has designed the following relational schema for your database:

Pet (*pid, name, type, owner*)

Contract (*cid, pid, startDate, duration*)

PricePerDay (*type, price*)

You can make the following assumptions:

- The attributes *startDate* and *duration* are both expressed in days, an integer. For example, startDate=400 means the 400th day, while duration=10 means the duration is 10 days.
- The attribute *type* in the Pet table is a foreign key to the attribute *type* in the PricePerDay table.
- The attribute *pid* in the Contract table is a foreign key to the attribute *pid* in the Pet table.

Specify the following queries using SQL:

1. [4] How much time does each type of pet spend in the hotel? Returned tuples should be formatted as (*type, totalDuration*)

   Answer: SELECT type, sum(duration) as totalDuration from Pet, Contract WHERE Pet.pid=Contract.pid group by type;

2. [5] How much money will you make if all of your contracts i.e., those in the Contract table, are fulfilled? (Note: total amount = days spent × pricePerday)

   Answer: SELECT sum(duration*price) FROM Pet, Contract, PricePerDay WHERE Pet.pid=Contract.pid AND PricePerDay.type=Pet.type;

3. [6] Who is the most frequent pet owner that comes to your hotel, i.e., who is the owner who provides the most contracts?

   Answer: SELECT owner FROM Pet, Contract WHERE Pet.pid=Contract.pid GROUP BY owner HAVING COUNT(*)>= ALL (SELECT COUNT (*) FROM Pet, Contract WHERE Pet.pid=Contract.pid GROUP BY owner)

B-Trees and Indexing- 25 points

1. Consider a partially filled B-tree of order 2 as shown in Figure 1.

   (a) [4] In the figure given below, fill in the values present in the root node of the tree. Use the figure itself to save time on re-drawing the B-tree.

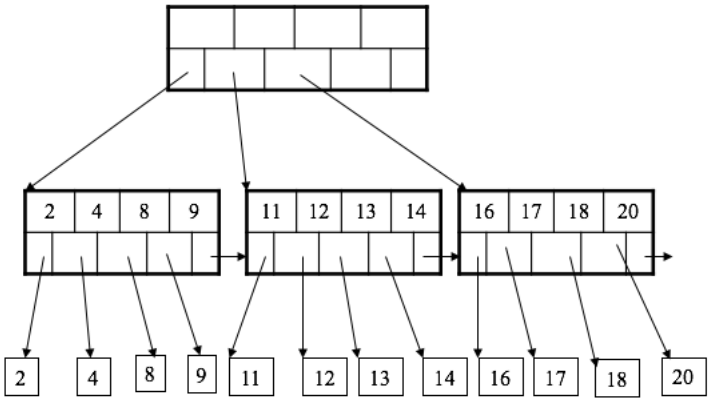
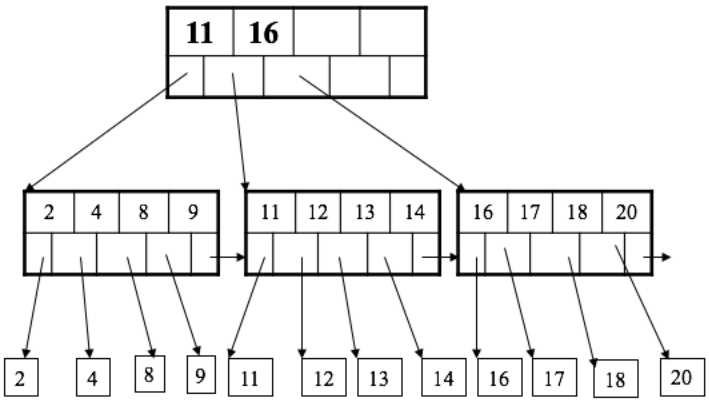


Figure 1: B-tree with root values missing

Answer:



   (b) [6] There are many sequences of exactly 3 insertions that will cause the root node in the B-tree shown in Figure 1 to split. List one such sequence of three insertions, and justify your choice (1–2 lines are sufficient)

   Answer: 10, 15, 21 (this is not the only correct answer).

2. [5] Draw an extensible hash table with a block capacity of 2, i.e., each block holds two records, after the insertion of the following search keys (in the same order as provided):

   1000, 1101, 0111, 1011, 1111

   You only need to draw the final table. Indicate the bit counter for each bucket as well as the array of pointers.

   Answer: See Figure 2.

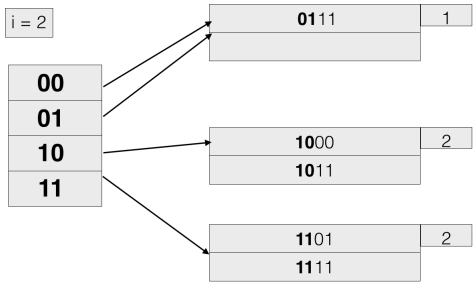


Figure 2: Extensible Hash Table

Query Execution and Optimization - 30 points

1. [10] Consider four relations with the following statistics:

A(w,x)	B(x,y)	C(y,z)	D(w,z)
T(A) = 500	T(B) = 200	T(C) = 400	T(D) = 100

- R(a,b) means relation R has attributes a and b. T(R) refers to the number of tuples in relation R.
- Use formula $T(R1 \bowtie R2) = T(R1) \times T(R2) \times 0.01$ for size estimation, where R1 and R2 are any two relations.
- Join the relations only where a natural join is feasible.
- You may assume that the join operation is symmetric, i.e., the plan (R1R2) is the same as the plan (R2R1).

Use the Dynamic Programming algorithm to find the optimal plan to join the above relations. Complete the following table:

Subset	Size	Lowest cost	Lowest cost plan
AB	1000	0	AB
BC	800	0	BC
CD	400	0	CD
AD	500	0	AD
ABC	4000	800	A(BC)
BCD			
ABD			
ACD			
ABCD			

Answer:

Subset	Size	Lowest cost	Lowest cost plan
AB	1000		AB
BC	800		BC
CD	400		CD
AD	500		AD
ABC	4000	800	A(BC)
BCD	800	400	B(CD)
ABD	1000	500	B(AD)
ACD	2000	400	A(CD)
ABCD	4000	1200	A(BCD)

2. Use Figure 3 to answer the following questions. Use B(R), B(S), B(T), B(U) to denote the number of blocks in each of those relations. For convenience, use K1= B($R \bowtie S$) and K2=B($R \bowtie S \bowtie T$). Let M denote the number of blocks in the buffer. Consider the following physical execution plans; **for each plan, estimate the cost and memory requirements for that plan**. You can skip the cost of the final write of ($R \bowtie S \bowtie T \bowtie U$). You should include cost of reading and writing the initial relations and any intermediates. Use the formulae discussed in class; if you wish to use other formulae, then clarify the assumptions you are making.

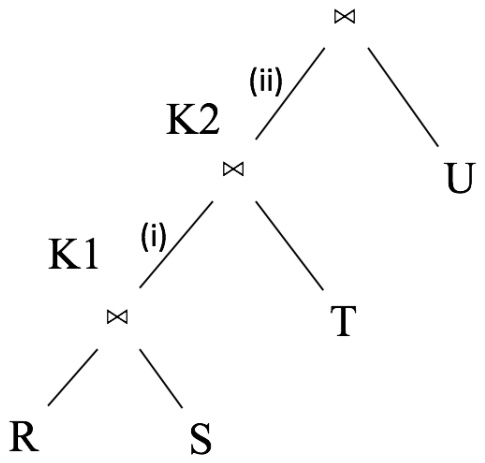


Figure 3: Figure for Question 2

(a) [6] Construct hash tables of S, T, U, all of which fit in memory simultaneously. Use a **one-pass hash join** for $R \bowtie S$, $(R \bowtie S) \bowtie T$ and $(R \bowtie S \bowtie T) \bowtie U$, while pipelining the output of $R \bowtie S$ and $(R \bowtie S) \bowtie T$ without materialization i.e., you don't materialize at (i) and (ii) (see Figure 3).

Answer:

Memory requirement: $M \geq 3 + B(S) + B(T) + B(U)$. This is because we need $B(S) + B(T) + B(U)$ blocks to store the three hash tables. In addition to storing the hash tables, we need one block to iterate over blocks of R, one block to pipeline the intermediate results and finally, one block for the output buffer.

IO cost: $B(R) + B(S) + B(T) + B(U)$. This is straightforward.

(b) [6] Construct hash tables for S and T, both of which fit in memory simultaneously. Use a **one-pass hash join** for $R \bowtie S$ and $(R \bowtie S) \bowtie T$ while pipelining the output of $R \bowtie S$ at (i). Then, materialize $(R \bowtie S) \bowtie T$ at (ii), construct a hash table for U, and use a one-pass hash join for $(R \bowtie S) \bowtie T$ with U.

Answer:

Memory requirement: $M \geq \max(B(S) + B(T) + 3, B(U) + 2)$. This is because in the first stage we need $B(S) + B(T)$ blocks to store the two hash tables. In addition to storing the hash tables, we need one block to iterate over blocks of R,

one block to pipeline the intermediate results and finally, one block for the output buffer.

In the second stage, we need $B(U)$ blocks to store the single hash table. In addition to this, we need one block to iterate over the materialized result, and one for the output buffer.

IO cost: $B(R) + B(S) + B(T) + B(U)$ + 2K2. We bring each block for each of the 4 relations once, in addition to writing and reading the materialized result at (ii).

(c) [8] Use a block nested loop join for $R \bowtie S$ (with R as the outer relation), pipeline $R \bowtie S$ at (i), while maintaining a hash table for T, join $R \bowtie S$ with T using a **one-pass hash join**. Then, materialize $(R \bowtie S) \bowtie T$ at (ii), construct a hash table for U, and use a one-pass hash join for joining $(R \bowtie S) \bowtie T$ with U.

Answer:

Memory requirement: $M \geq \max(B(T)+4, B(U)+2)$. This is because in the first stage we need $B(T)$ blocks to store the hash table for T. In addition to storing the hash table, we need one block to iterate over blocks of R as the outer relation, one block to iterate over blocks of S, one block to pipeline the intermediate results and finally, one block for the output buffer.

In the second stage, we need $B(U)$ blocks to store the single hash table. In addition to this, we need one block to iterate over the materialized result, and one for the output buffer.

IO cost: $B(R) + \frac{B(R)}{M-B(T)-3} \cdot B(S) + B(T) + B(U)$ + 2K2. We need $B(T) + 3$ blocks to store the hash table, pipeline block, output block and the block for the inner relation. This leaves $M - B(T) - 3$ for the outer relation R, resulting in $\frac{B(R)}{M-B(T)-3}$ iterations for the block-based nested loop join. For each iteration, we bring all blocks of S into memory once. In addition to this we bring all blocks of R, T, U into memory once. We need 2K2 IO cost for writing and reading the materialized result at (ii).

Transaction Management - 15 points

Consider the following **UNDO** log, and use it to answer the following questions.

LogID	Log
1	<START T1>
2	<T1, A, 20>
3	<START T2>
4	<START T3>
5	<T2, C, 7>
6	<T1, B, 15>
7	<T2, D, 9>
8	<COMMIT T1>
9	<START T4>
10	<T3, E, 2>
11	<T4, A, 6>
12	<ABORT T2>
13	<T3, B, 9>
14	<COMMIT T3>
15	<T4, B, 21>
16	<START T5>
17	<START T6>
18	<T6, D, 8>
19	<COMMIT T6>
20	<T5, E, 6>

1. [4] At the end of the log, what is the value of Variable C? Explain why (1-2 lines are sufficient).

   Answer: Because Transaction T2 was the only transaction to touch C, and was aborted during its execution, all of the variables touched by T2 will be rolled back. So because of entry 5, Variable C will be rolled back to a value of 7.

2. [5] Suppose we want to start nonquiescent checkpointing after LogID 8.

   Between which two LogID lines would we start checkpointing, and what should the log entry (for checkpointing) look like?

   Then, between which two LogID lines would we stop checkpointing, and what should the log entry look like?

   Answer: Between Log 8 and 9, $< STARTCKPT(T2, T3) >$
   Between Log 14 and 15, $< ENDCKPT >$

3. [6] Continue from (2) after starting checkpointing. Suppose the system crashes right after LogID 13. Show which actions (e.g.: <T1, A, 15>) need to be undone and in what order.

   Answer: The undo transactions in sequence: < T3, B, 9 >, <T4, A, 6 >, <T3, E, 2>