CS 564 Final Exam Fall 2015
Answers

A: STORAGE AND INDEXING **[20pts]**

I. **[10pts]** For the following questions, **clearly circle** True or False.

1. The cost of a file scan is essentially the same for a heap file and a sorted file.

   TRUE

2. It is possible to store fixed-length records in a page format (layout) designed for variable-length records.

   TRUE

3. If a B+ tree index follows the alternative of storing the data records directly in the leaf pages, it is automatically a clustered index.

   TRUE

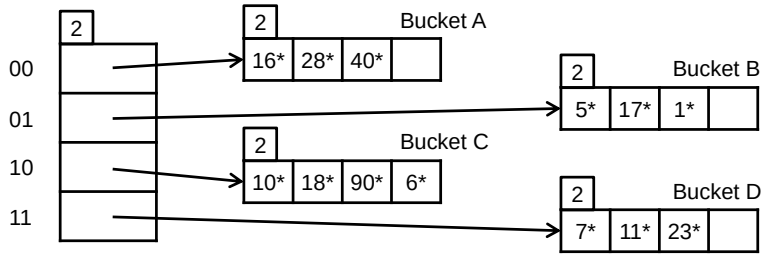
4. Among all the non-leaf nodes of a B+ tree index, duplicate entries of search keys are allowed in the root node.

   FALSE

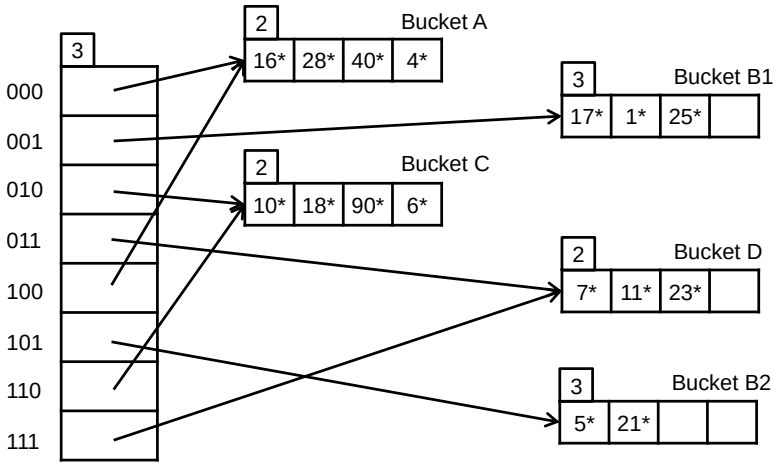
5. A hash index is suitable for range search predicates.

   FALSE

II. **[10pts]** Consider the following extendible hash index.



1. **[8pts]** Draw the index after the following keys have been inserted: 4*, 21*, and 25*.



2. **[2pts]** Having inserted the above three keys, what is the minimum number of delete operations needed for the global depth to decrease? **Clearly circle** the correct answer. (Hint: A bucket is merged with its split image if and only if it becomes empty.)

   (a) 0 (b) 1 (c) 2 (d) 3 (e) 4

   ANSWER: (c)

B: OPERATOR IMPLEMENTATIONS **[30pts]**

I. **[6pts]** For the following questions, **clearly circle** either True or False.

1. It is not possible to speed up the evaluation of the aggregation $\gamma_{COUNT(*)}(R)$ by using a B+ tree index on R.

   FALSE

2. A block nested loop join algorithm can achieve the same number of page I/Os as a hash join algorithm for some database instances.

   TRUE

3. Using the double buffering technique typically reduces the total number of passes needed for an external merge sort.

   FALSE

II. **[10pts]** We are given a relation $R(A, B, C)$ with 1,000 pages. There is a clustered B+ tree index on R on the composite key (A, B). Data records are stored in a file separate from the

index. The index has 2 non-leaf levels and 200 leaf pages. The buffer pool has 200 frames. What is the **lowest** possible I/O cost of computing $\pi_A(R)$? Ignore the cost of writing the output to disk. Describe the optimal algorithm and explain how it achieves that cost.

ANSWER: The optimal cost is 202 I/Os by using only the index file (2 non-leaf pages to get to the first leaf page and 200 leaf pages). This is possible because the projection list attributes (A) form a prefix of the search key (A,B). Thus, the leaf pages are already sorted on the projection list, which enables a sort-based project using a simple sequential scan of the leaf pages. Notice that we do not have to scan the full data records, because we do not need to access attribute C to project on A.

III. **[14pts]** We are given two relations: R with 1,000 pages and S with 40,000 pages. The buffer pool has 102 frames. We are performing a key-foreign key join of R and S wherein S has the foreign key attribute. Which of the following algorithms has the lowest I/O cost: **block nested loop join, hash join, or sort-merge join**? Ignore the cost of writing the output to disk. Assume that the fudge factor for constructing a hash table is $f = 1.4$. Explain your answer in detail.
(Hint: It is possible to obtain and justify the answer fully without even computing all the I/O costs exactly.)

ANSWER: Hash join. Since $102 > \sqrt{1.4 \times 1,000}$**, hash join will have an I/O cost of** $3 \times (|R| + |S|)$**. But since** $102 < \sqrt{40,000}$**, sort-merge will not finish in 2 passes, and will need to do a full external merge sort, which raises the I/O cost compared to the hash join. Block nested loop join will obviously be much slower than hash join, since there will be** $\lceil 1.4 \times 1,000/100 \rceil = 14$ **blocks of R, and hence, 14 passes over S.**

C: QUERY OPTIMIZATION **[20pts]**

I. **[14pts]** Consider the following database schema:

Product (PID, PName, Price)

Company (CID, CName, Location, PID)
Company.PID is a foreign key referring to Product.PID.

Product has 20,000 tuples, and each record is 20 bytes long. Company has 1,000 tuples, and each record is 25 bytes long. Each page can hold 5,000 bytes. The buffer pool has 102 frames. Assume that there are 500 distinct values for the attribute Location and that its distribution is uniform. As always, the fudge factor for a hash table is $f = 1.4$.
Consider the following SQL query:

```
SELECT  PName, CName
FROM    Product, Company
WHERE   Product.PID = Company.PID
```

```
AND     Location = "Madison" ;
```

Propose a physical plan (does not need to be optimal) that evaluates the above query. Compute its total I/O cost and briefly explain each component of the total cost. Ignore the cost of writing the output to disk.

ANSWER: The number of pages for Product is $(20,000 \times 20)/5,000 = 80$**, and for Company is** $(1,000 \times 25)/5,000 = 5$**. Thus, both tables easily fit together in the buffer pool. Thus, a simple physical plan is to first compute the join. Any join algorithm can be used; the I/O cost is just one read of each table, i.e.,** $80 + 5 = 85$**. After the join, we simply pipeline the result to an in-memory selection on Location and then an in-memory projection without duplicate elimination. Thus, we have no more I/O costs.**

II. **[6pts]** Consider the following database schema:

$$R(A,B), \quad S(A,B,C), \quad T(B,D,E)$$

For the following questions, **clearly circle** either True or False.

1. The following two relational algebra queries are equivalent:

$$Q_1 = \sigma_{A=1,B>2}((R \bowtie S) \bowtie T)$$
$$Q_2 = (\sigma_{A=1,B>2}(R \bowtie S)) \bowtie T$$

   TRUE

2. The following two relational algebra queries are equivalent:

$$Q_3 = \pi_E(\sigma_{D=1}(T \bowtie S))$$
$$Q_4 = \pi_B(S) \bowtie \pi_{B,E}(\sigma_{D=1}(T))$$

   FALSE

D: TRANSACTION MANAGEMENT **[10pts]**

I. **[4pts]** For the following questions, **clearly circle** either True or False.

1. Using 2PL guarantees that there will be no deadlocks during a concurrent execution of transactions.

   FALSE

2. All four SQL isolation levels guarantee that there will be no dirty reads during a concurrent execution of transactions.

   FALSE

II. **[6pts]** Consider the following two transactions:

$$T1 : R(A), W(A), R(B), W(B), Commit$$
$$T2 : R(B), R(C), W(C), W(B), Commit$$

Consider the following interleaved schedule of the two transactions:

$$R_{T1}(A), R_{T2}(B), R_{T2}(C), W_{T1}(A), R_{T1}(B), W_{T1}(B), W_{T2}(C), W_{T2}(B), Commit_{T1}, Commit_{T2}$$

Is the schedule serializable? If you claim yes, write an equivalent serial (non-interleaved) execution of the two transactions. If you claim no, explain why it is not serializable.

ANSWER: No, it is not serializable. There is a write-write conflict on B; the update by T1 is lost, since T2 overwrites it after reading an older value of B. Thus, it is not equivalent to either $T1 \rightarrow T2$ **or** $T2 \rightarrow T1$**.**

E: RELATIONAL ALGEBRA AND SQL **[20pts]**

I. **[10pts]** For the following two questions, **clearly circle** all the correct options and only the correct options. If you circle only some of the correct options and no wrong options, you will get proportional points. But if you circle even one wrong option, you will get zero for that question.
(Hint: If you are not sure about an option, it is probably better not to circle it.)

1. **[4pts]** Which of the following symbols do **not** represent relational operators from Codd's original relational algebra?

   (a) γ (b) θ (c) δ (d) $+$ (e) $\times$

   ANSWER: (a), (b), (c), (d)

2. **[6pts]** This question is directly from Homework 1. Consider the following database schema:

```
Likes (drinker, beer)
Frequents (drinker, bar)
Serves (bar, beer)
```

   Which of the following relational algebra queries answer the following question: which drinkers frequent *only* bars that serve *only* beers they like?

(a) $\pi_{drinker}(Likes \bowtie Frequents \bowtie Serves)$

(b) $\pi_{drinker}(Frequents) -$
$\pi_{drinker}(((\pi_{drinker}(Frequents) \times \pi_{beer}(Serves)) - Likes) \bowtie Frequents \bowtie Serves)$

(c) $\pi_{drinker}(Frequents) - \pi_{drinker}(Frequents - \pi_{drinker,bar}(Likes \bowtie Serves))$

(d) $\pi_{drinker}(Frequents) - \pi_{drinker}(Likes \bowtie Frequents \bowtie Serves)$

(e) $\pi_{drinker}(Frequents) - \pi_{drinker}(\pi_{drinker,beer}(Frequents \bowtie Serves) - Likes)$

ANSWER: (b), (e)

II. **[10pts]** Consider the following database schema (the Expedia database):

Locations (Zipcode, City, State)

Hotels (HotelID, Name, Zipcode, NumRooms)
Hotels.Zipcode is a foreign key referring to Locations.Zipcode.

Searches (SearchID, Zipcode, StartDate, Duration, NumPeople)
Searches.Zipcode is a foreign key referring to Locations.Zipcode.

Results (HotelID, SearchID, Rank, Price)
Results.HotelID is a foreign key referring to Hotels.HotelID.
Results.SearchID is a foreign key referring to Searches.SearchID.

Write a **single** SQL query (it should be one statement) to answer the following:

Get the names and cities of the top 10 hotels (in increasing order of their rank) for the search request with SearchID 564 such that all the hotels output are in the same state as the state corresponding to the zipcode from the given search request.

ANSWER:

```
SELECT    H.Name, H.City
FROM      Results R, Hotels H, Searches S, Locations LH, Locations LS
WHERE     S.SearchID = 564
          AND R.HotelID = H.HotelID
          AND R.SearchID = S.SearchID
          AND LH.Zipcode = H.Zipcode
          AND LS.Zipcode = S.Zipcode
          AND LH.State = LS.State
ORDER BY  R.Rank
LIMIT     10;
```

EXTRA CREDIT (OPTIONAL) **[+5pts]**

Consider the relation $R(\underline{K}, A_1, \ldots, A_n)$, where n is a positive integer, and K is the key. What is the number of non-trivial functional dependencies on R? Explain your answer.

ANSWER: K is the key of R. Thus, any non-trivial FD has to be of the form $XK \rightarrow Y$. X can be any subset of $A_1 \ldots A_n$. Fix some X of size k ($= 0$ to n). The number of choices for Y is $2^{n+1} - 2^{k+1}$, since it could be any set of attributes, except a subset of XK. Thus, the total number of non-trivial FDs is $\sum_{k=0}^{n} \binom{n}{k}(2^{n+1} - 2^{k+1}) = 2^n \cdot 2^{n+1} - 2 \cdot 3^n = \mathbf{2(4^n - 3^n)}$.