UCLA
Computer Science Department

Instr: C. Zaniolo
TAs: Jiaqi Gu, Jia Teoh, Zijun Xue

Your Name and ID: Elan Markowitz [redacted]

CS143, Fall 2017: MIDTERM EXAM: Closed Book, 110 minutes.

Please Read:

- Attach extra pages as needed. Write your name and ID on the extra pages.
- If you need to make any assumptions to solve a problem, please write you assumptions clearly in your answer.
- Simplicity and clarity of your solutions will count. You may get as few as 0 point for a problem if your solution is far more complicated than necessary, or if we cannot understand your solution.
- Please write neatly.

Problem	Score	
1	(40%)	40
2	(40%)	40
3	(20%)	20
Total	(100%)	

Extra Credit (9 Points) 3

Midterm Score: 100 + 3

Problem 1: 40 points–all questions have the same weight

The relation `warehouse(PartNo, SupplierNo, Price)` describes the suppliers for each part, along with the price they charge for the part. To cut inventory, the manager of our warehouse wants to eliminate non-competitive suppliers. A supplier is competitive if either (i) he is the only supplier of some part, or (ii) he supplies at least two parts at a minimum cost (however the supplier could share this minimum cost for the with other suppliers). All the others are non-competitive suppliers.

A1: Write an SQL query to find all non-competitive suppliers.

A2: Let us now modify the definition of competitive supplier as follows: A supplier is competitive if either (i) he is the only supplier of some part, or (ii) he supplies at least two parts at a strictly minimum cost (i.e., ties are not viewed as min). All the others are non-competitive suppliers. Please write an SQL query to find the non-competitive suppliers according to this modified definition

A3: Write an SQL statement to delete from the warehouse all the part-suppler tuples where the supplier is non-competitive (as defined in A2):

A4: Is query A2 expressible in basic RA, which does not have aggregates? (Hint: re-express "minimum cost" and "least cost" without using aggregates.)

non-competitive iff not solesupplier and not cheapest for two parts

A1:
```
SELECT SupplierNO FROM warehouse w1
WHERE NOT EXISTS (SELECT * FROM warehouse w2
    WHERE w1.SupplierNo != w2.SupplierNO
    AND w1.PartNO = w2.PartNO)
```

A1: Non competitive if sells ≤2 competitively priced parts and does not sell any unique parts.

```
1  SELECT SupplierNo FROM warehouse w1
2  WHERE NOT EXISTS (SELECT * FROM warehouse w2
3      WHERE w1.SupplierNo != w2.SupplierNO AND w2.PartNo = w2.PartNo AND
4          w2.Price < w1.Price)
5  GROUP BY SupplierNo
6  HAVING COUNT(*) <= 2
7  Except
8  SELECT SupplierNO FROM warehouse w1
9  WHERE NOT EXISTS (SELECT * FROM warehouse w2
10                   WHERE w1.SupplierNo != w2.SupplierNo
11                   AND w1.PartNo = w2.PartNo);
```

A2: Same as A1 but change line 4 from "w2.Price < w1.Price)" to "w2.Price <= w1.Price)"

To be minimally priced there cannot be another supplier at the same or lower price.

A3:

```
DELETE FROM warehouse WHERE SupplierNo NOT IN
(SELECT SupplierNo FROM warehouse w1
 WHERE NOT EXISTS
    (SELECT * FROM warehouse w2
     WHERE w1.SupplierNO != w2.SupplierNO
       AND w1.PartNO = w2.PartNO)
 OR 2 <= (COUNT
    (SELECT * FROM warehouse w3
     WHERE NOT EXISTS (SELECT * FROM warehouse w4
        WHERE w3.SupplierNO != w4.SupplierNO
          AND w3.PartNO = w4.PartNo
          AND w4.Price <= w3.Price)
    AND w1.SupplierNo != w3.supplierNO)));
```

A4: Yes

$w1 \leftarrow \rho_{w1}(warehouse)$

$w2 \leftarrow \rho_{w2}(warehouse)$

$w3 \leftarrow \rho_{w3}(warehouse)$

$w4 \leftarrow \rho_{w3}(warehouse)$

~~$w5 \leftarrow \rho_{w5}(warehouse)$~~

~~$w6 \leftarrow \rho_{w6}(warehouse)$~~

$SellsCheapPart \leftarrow \Pi_{SupplierNo, PartNo}(warehouse) - \Pi_{w1.SupplierNo, w1.PartNo}\left(\sigma_{w1.Price >= w2.Price}(w1 \times_{PartNo} w2)\right)$

$Sells2cheap \leftarrow \Pi_{S1.SupplierNo}\left(\sigma_{S1.supplierNo = S2.SupplierNo, S1.PartNo != S2.PartNo}\left(\rho_{S1}(SellsCheapPart) \times \rho_{S2}(SellsCheapPart)\right)\right)$

$SoleProvider \leftarrow \Pi_{SupplierNo}\left(\Pi_{SupplierNo, PartNo}(warehouse) - \Pi\left(w_{3.SupplierNo, w_{3.PartNo}}\sigma_{w3.PartNo = w4.PartNo}(w_3 \times w_4)\right)\right)$

replace set names with associated query

$$\left(\Pi_{SupplierNo}(warehouse) - Sells2(cheap) - SoleProvider\right)$$

Problem 2, 40 Points. All questions have the same weight.

We have 1 million employee tuples with unique key Eno. The length of each tuple is 50 bytes. These are stored as fixed-length unspanned records in a file consisting of blocks of size 2048 bytes.

On this file, we build a **sparse** index on Eno. The index is is organized as a B+ tree, where each key takes 18 bytes and each pointer takes 22 bytes (the leaf nodes are chained together as in the textbook). The B+ tree blocks contain 2048 bytes.

A. How many blocks are needed to store all the records in the file.

B. Compute the blocks used at each level of the B+ tree, for the best case and the worst case.

C. Is this index a primary index or a secondary one?

D. A query asking for the records of all employees whose Eno falls in a certain range returns 800 records. Estimate the number of pages accessed to retrieve this query in the worst case situation.

A: Tuples per block = $\lfloor$ Block size / Tuple size $\rfloor$ = $\lfloor 40.96 \rfloor$ = 40 ✓

Blocks needed = $\lceil$ number of tuples / Tuples per block $\rceil$ = $\lceil 1{,}000{,}000 / 40 \rceil$

= **25,000 blocks** ✓

B: Best case, each block is filled as much as possible

$n = \left\lfloor \frac{B + key\_size}{ptr\_size + key\_size} \right\rfloor = \left\lfloor \frac{2048 + 18}{22 + 18} \right\rfloor = \lfloor 51.65 \rfloor = 51$ ✓

number of leaf nodes = $\lceil$ number of blocks / n-1 $\rceil$ = $\lceil 25000/50 \rceil$ = **500**

(sparse tree => 1 ptr per block) and (n-1 Data pointers per leaf)

L1 nodes = $\lceil$ leaf nodes / n $\rceil$ = $\lceil 500/51 \rceil$ = **10** ✓

L1 nodes $\leq$ n so next layer is the root node. **1** ✓

Worst case, nodes are minimally filled (min pointers)

n is the same: 51

minimum keys for a non-root leaf node is $\lceil (n-1)/2 \rceil = 25$

= num data pointers

Leaf nodes = $\lfloor$ Num blocks / 25 $\rfloor$ = **1000** ✓

minimum pointers for non-leaf non-root node is $\lceil n/2 \rceil = 26$

L1 nodes = $\lfloor 1000/26 \rfloor$ = **38** ✓

38 < min pointers*2 so there is only one layer left

Root nodes = **1** ✓

C: This is a primary index since it is sparse. Data must be clustered by Eno.

D: Worst case for 800 sequential records

1 ROOT

38 L1

1000 LEAFS (25 data pointers)

Number of data blocks = $\lceil (800-) / \text{tuples per block} \rceil + 1 = 21$ blocks

recordcount

21 blocks => 21 data pointers

21 data pointers could be stored on $\lceil 21/25 \rceil + 1$ leaf nodes

= 2 Leaf nodes

+ Root node

+ 1 L1 node

Total 21+2+1+1 = 25 blocks accessed

Nice!

Problem 3, 20 Points. All questions have the same weight.

A file is indexed using an extensible hash table which applies the function h(n) = n mod 256 to the search keys of the first seven records in the file. The records are stored in buckets that contain 3 records.

The keys of the first seven records are: {106, 115, 916, 126, 16, 15, 31} which produce the following hash values:

```
h(106) = 106  = 01101010
h(115) = 115  = 01110011
h(916) = 148  = 10010100
h(126) = 126  = 01111110
h(16) =  16   = 00010000
h(15) =  15   = 00001111
h(31) =  31   = 00011111
```

Assuming that that the leftmost i bits of the hash key are used for a bucket address table of size 2^i, please answer the following questions:

A. What is the size of the hash table after all the seven items above are inserted? (Explain how that can be determined without drawing the actual table.)

B. As more records in the file are are indexed using this extensible hash function, could there be situations where overflow buckets must be used? If your answer is yes, give the search keys of three records, that once they are added to those first seven, will cause the creation of an overflow bucket.

A. i=2, bucket address = 4, max records = 12
There are only three hashes that begin with 00 and three that begin with 01 and 1 that begins 10. Since none are greater than 3 we do not need to split any buckets, and 2 leftmost bits are enough for the hash key. **Directory size = 4, number buckets = 3**

B. Yes. 16 + 256, 16 + 2·256, 16 + 3·256
272, 528, 784
These numbers all hash to 16. Since 16 is already a record, there are 4 records that hash to 16. Looking at the binary it is clear that no matter how many bits you look at, there will never be any differences between these hashes. Thus we must use overflow buckets.

n	h(n)
16	16 = 00010000
272	16 = 00010000
528	16 = 00010000
784	16 = 00010000

Extra Credit. Each question 3 Points

Consider the relations `R(A, B)` and `S(B, C)` `R` contains $n > 0$ tuples and `S` contains $m > 0$ tuples; NULLs are not allowed in either relation.

- What is the minimum number of tuples that may result from the natural join of these two relations:? 0 ✓
- What is the maximum? minimum(|R|, |S|) ✗
- How will your previous answers change if instead of natural join we take the full outer join?

It will always have |R|·|S| tuples ✗.