UCLA
Computer Science Department
Spring 2016

Instructor: C. Zaniolo
TAs: Ariyam Das and Jiaqui Gu

Student Name and ID: Richard Sun 904444918

CS143 Midterm EXAM: Closed Book, 110 Minutes

Attach extra pages as needed. Write your name and ID on any extra page that you attach. Please, write neatly.

Problem	Score	
1	(30%)	13 +13
2	(30%)	30
3	(20%)	12
4	(20%)	20
Total	(100%)	

Extra Credit (5 points): 5

Midterm Score: 93

Problem 1. 30 Points The relation: **warehouse(PartNo, SupplierNo, Price)** describes the suppliers for each part, along with the price they charge.

1.A To cut inventory,the manager of our warehouse wants to eliminate non-competitive suppliers. Competitive suppliers are those that supply at least two parts at a minimum cost (however they might share this minimum with other suppliers). All the others are non-competitive suppliers. Write an SQL query to find all non-competitive suppliers.

1.B Change the definition: competitive suppliers are those who charge a price strictly lower than those of all other suppliers on two or more items. Find the non-competitive suppliers according to the new definition.

A.
```
select w1.SupplierNo
from warehouse as w1
where w1.Price in (select min(w2.Price)
                   from warehouse as w2
                   where w1.PartNo = w2.PartNo)
group by w1.SupplierNo
having count(*) >= 2;
```

B.
```
select w1.SupplierNo
from warehouse as w1
where not exists (select *
                  from warehouse as w2
                  where w1.PartNo = w2.PartNo
                    and w1.SupplierNo <> w2.SupplierNo
                    and w2.Price <= w1.Price)
group by w1.SupplierNo
having count(*) >= 2;
```

Problem 2: 30 points

Suppose that blocks can hold 100 search keys and 101 pointers. We built a *dense* index organized as a B+ tree on a file of 2 millions records, where the records are placed in blocks that hold 10 records each. Say that the search key for the B+ tree is the candidate key for the relation. Answer the following questions assuming the *worst case* scenario:

A. Compute the number of blocks (a.k.a. nodes) used at each level of the B+ tree.
B. How many index blocks and file blocks will we have to access to find a record with a given key value if this is a dense index? (Assume that no block is initially in memory, and make the same assumption for the questions that follow.)
C. Same question as in B, but now assume that our B+ tree is a sparse primary index.
D. Here too assume that our our B+ tree is a sparse primary index. We now have a range query that is satisfied by 1000 records: how many file blocks will we have to access to retrieve those 1000 record?
E. Assume that K was defined as the primary candidate Key in the SQL declarations of R. Can we use a sparse secondary index on the stored R table to speed-up the enforcement of the uniqueness constraint for K?

worst case: $\lceil \frac{101}{2} \rceil - 1 = 50$ leaf node pointers to data
51 internal node pointers

A. leaf nodes: $\frac{2,000,000}{50} = 40,000$ blocks

2nd level: $\lceil \frac{40,000}{51} \rceil = 784$ blocks

3rd level: $\lfloor \frac{784}{51} \rfloor = 15$ blocks

4th level (root): 1 block

B. 4 level index

4 index blocks, 1 file block

C. The index finds which file block the record is in, and you scan that block to find the record, so there are $\frac{2000000 \text{ records}}{10 \text{ records/block}} = 200,000$ file block pointers.

leaf nodes: $\frac{200,000}{50} = 4000$

2nd level: $\lfloor \frac{4000}{51} \rfloor = 78$

3rd level: $\lceil \frac{78}{51} \rceil = 1$

3 level index

3 index blocks, 1 file block

D. sparse index, so records are already clustered in blocks.

10 records / block

$\frac{1000 \text{ records}}{10 \text{ records/block}} = 100$ blocks

E. No. A secondary index is nonclustering, since the tuples not ordered by the search key. Thus, a secondary index must be dense in order to be effective.

Problem 3: 20 points Multiple choice questions. Mark the box of every answer that is true.

A. Relations R and S have attributes a and b. Then, which of the properties below is true for the following RA queries:

Q1: $\pi_a(R) \cap \pi_a(S)$ Q2: $\pi_a(R \cap S)$

- ■ (a) Q1 and Q2 always produce the same answer.
- ■ (b) The answer to Q1 is always contained ($\subseteq$) in the answer to Q2.
- ■ (c) The answer to Q2 is always contained ($\subseteq$) in the answer to Q1.
- □ (d) None of the above is true.

B. When null values are allowed in column R.a or column R.b, the value of the following logic expression in SQL

R.a > R.b AND R.a < 0 AND R.b > 0

can be, depending on the tuple considered:

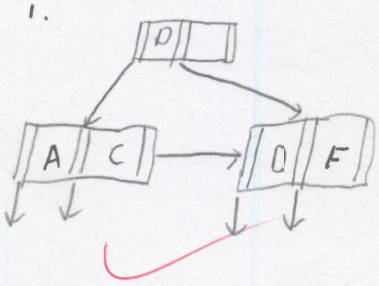
- □(a) only TRUE or FALSE
- ■(b) only FALSE or UNKNOWN
- □(c) only TRUE or UNKNOWN
- □ (d) Any of TRUE, FALSE, or UNKNOWN

Problem 3: 20 points Assume that we use B+ trees of order n = 3 (n is the maximum number of pointers in a node)

(i) Draw a tree of height 2 with the following keys: A, C, D, F (lexicographically ordered).

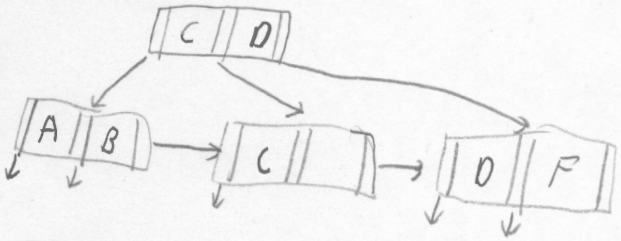
(ii) Show how the insertion of two new keys, followed by their deletions, could change this 2-level tree into a 3-level tree with exactly the same keys (i.e., A, C, D, F). Give an example of such keys, and draw the 3-level tree so generated.

i.

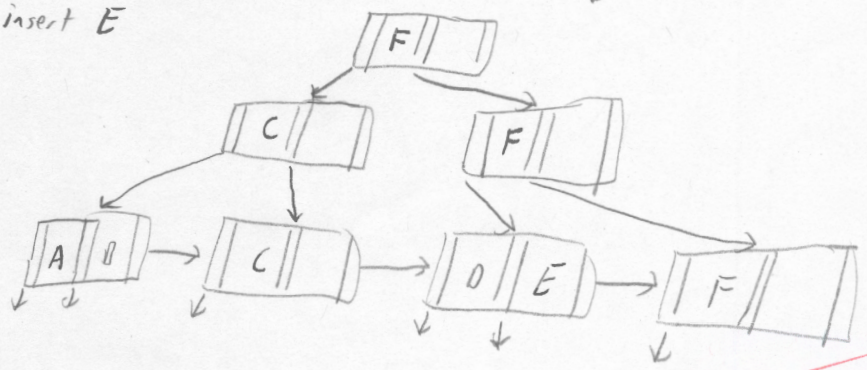


ii. New keys: B and E

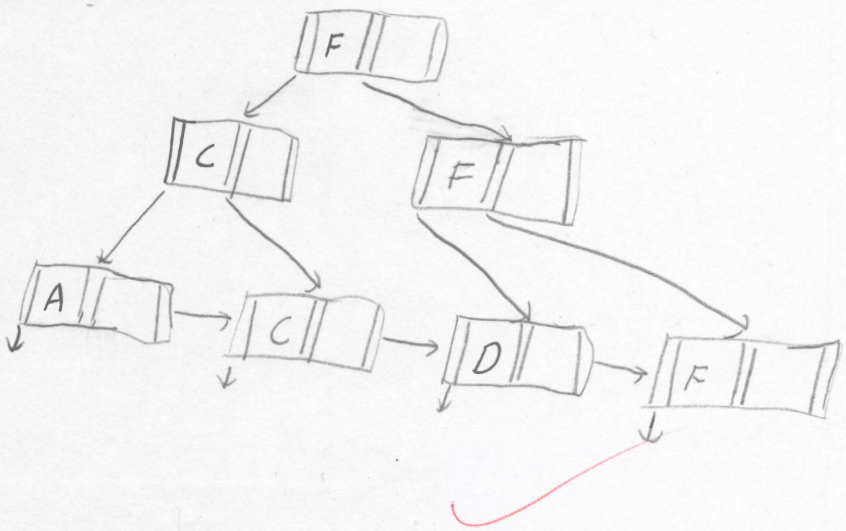
insert B



insert E



delete B and E



5

Extra Credit: 5 points

Overflow buckets are used for both (i) Static Hashing, and (ii) Extensible Hashing, but not in the same way. Explain the kinds of events that cause the creation of new overflow buckets for (i) and (ii): clarify their respective differences and the different actions taken by the hashing module in those cases.

i. In static hashing, overflow buckets are needed when there are many tuples with the same key, or many tuples have similar hashes. Overflow can be caused by inserting items with the same key or inserting many keys at random, since some of those keys will hash to the same bucket, causing overflow. Static hashing creates overflow buckets every time a bucket fills up.

ii. Extensible hashing only needs overflow buckets when there are many tuples with the exact same hash. Unlike static hashes, buckets will first be split into more specific buckets (i.e. based on more digits of the hash value). Overflow buckets are only used when the normal bucket has expanded as much as possible, so it has only records with some exact hash value. Then, if more tuples with this key are inserted, overflow buckets will be created.