UCLA
Computer Science Department
Fall 2003

Instructor: C. Zaniolo
TA: Fusheng Wang

Student Name and ID: _____

CS143 MIDTERM EXAM: Closed Book, 2 Hours

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

Problem	**Score**	
1	(32%)	
2	(20%)	
3	(30%)	
4	(18%)	
Total	(100%)	

Extra Credit:

Midterm Score:

1. Two Queries: one in RA, the other in SQL—32 Points

Consider the relation `Took(StID, Course, Quarter)`
whose tuples record that a student took a given class in a given quarter. You can assume that `StID` is an integer. Also quarters are represented by integers, where later quarters are represented by larger integers.

A. Write a **relational algebra** expression to find all pairs of students who have never taken a class together—i.e., have never taken the same class in the same quarter. Make sure to return each pair of students only once. For example, if your expression returns `(X,Y)` it should not return `(Y, X)`.

B. Write an **SQL query** to find the students who took CS143 without having taken the prerequisite CS31 course *before* they took CS143.

Answer to A: $P - C$ where

Rename: $S = \rho_S(Took), \quad V = \rho_V(Took)$

Class together: $C = \pi_{S.StID,V.StID}(\sigma_{S.Course=V.Course,S.Quarter=V.Quarter}(S \times V))$

Unique Pairs: $P = \pi_{S.StID,V.StID}(\sigma_{S.StID>V.StID}(S \times V))$

B. There are many correct ways to find the CS143 sneakins, i.e., students who took CS143 but never took CS31 or they took it in a later quarter. All correct ways require some nonmonotonic construct such as negation (since the insertion of new CS31 tuples can remove sneakins from the list). I like this SQL query best:

```
select T1.StID from Took as T1 where T1.Course='CS143'
and T1.StID not in (select T2.StID from Took as T2
                    where T2.Course='CS31'
                    and T1.Quarter < T2.Quarter)
```

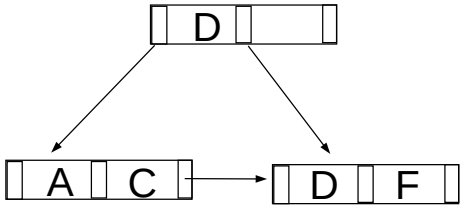
2. B+ Trees—20 Points

Assume B+ trees of order $n = 3$ (n is the maximum number of pointers in a node)

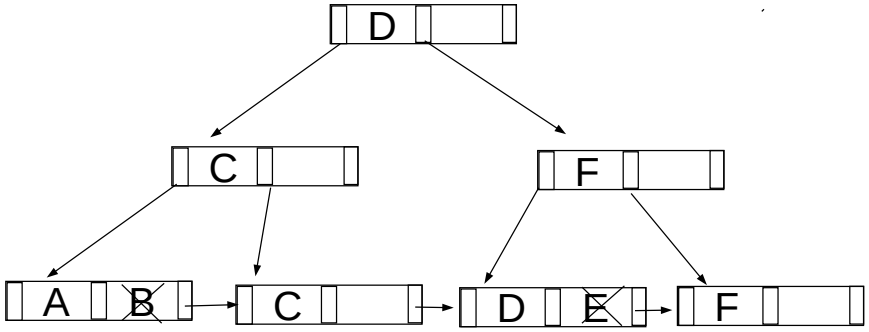
A. Draw a tree of height 2 with the following keys: A, C, D, F (lexicographically ordered).

B. 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.

Two level Tree:



Insert B and E and then delete them



3. Storage Management—30 Points

The relation: `took(StudentID, CourseNo, Quarter, Year, Units, Grade)`

contains the grades for the courses completed by UCLA students during the last 20 years. For simplicity, assume that there are 25,000 students enrolled each quarter, and that each student takes four courses per quarter, and that there are four quarters each year. Then we get a total of 8,000,000 records. If 10,000 new students enter UCLA every year, we can assume that in `took` there are 200,000 different students, each identified by its `StudentID`. On the average, a student took 40 different courses.

A. If the file blocks hold 2048 bytes and each `took` tuple requires 50 bytes, how many blocks will then be needed to store the relation `took`?

B. The table has a primary index on `StudentID`. If the `StudentID` index is a B+ tree with order $n = 101$, how many levels, and nodes, does the B+ use, in the worst case. (n denotes the maximum number of pointers in a node.)

C. How many blocks of the actual file will need to be retrieved to answer the query: find the GPA of a given student?

Answers.

A. Tuples per block $\llcorner 2048/50 \lrcorner = 40$. Number of blocks in the file $\ulcorner 8000000/40 \urcorner = 200000$.

B. For a sparse index, the bottom level of the B+ will have to keep pointers to 200,000 blocks. For a dense index a pointer to each key value, i.e., 200000 pointers, and no additional pointer bucket is needed since this is a primary index. Thus the situation is the same for both cases.

The worst case B+ tree has 50 pointers to the file in the leaf nodes and 51 pointers in the internal nodes:

Leaf: $\llcorner 2000000/50 \lrcorner = 4000$ Middle: $\llcorner 4000/51 \lrcorner = 78$ Root: a node with 78 pointers.

We have three levels.

C. Three blocks to search down the index and one or two blocks to retrieve the data from the file.

4. Potpourri—18 Points

Please indicate if the following statements are TRUE or FALSE. If false, please write one short sentence explaining why. (If true, then just write the word TRUE.)

(a) Sequential IOs are more expensive than random IOs.

FALSE: with sequential I/O the cost for reading the next piece of data is very small; for random I/O the delay is the large delay associated with arm movement and rotational latency.

(b) Generally, when the sizes of join relations are big, nested loop joins perform better than sort-merge joins.

FALSE: nested joins is always the worst kind of join: quadratic in the number of **tuples**. With sort-merge we have log of number of **blocks**.

(c) To modify a block on disk, the block must be read into memory first.

TRUE.

(d) In general, there is at most one sparse index for each relation.

Basically TRUE. But there could be several nested spzrse indices on the same clustering key: e.g., A, AB, ABC ...

(e) For the queries of the form $\sigma_{A=k}(R)$, extensible hashing is better than B+trees (A is an attribute and k a constant).

TRUE.

(f) For the queries of the form $\sigma_{A>k}(R)$, extensible hashing is better than B+tree (A is an attribute and k a constant).

FALSE: Hashing normally does not help with range queries.

Extra Credit—9 Points

`R` is a database table with the following content:

$$\{(1,2),(1,2),(3,4),(5,6),(5,6),(5,6)\}$$

(a) Does `R` have candidate keys, and what are they?

NO: With duplicate tuples there is no key.

(b) We are interested in writing a statement that, when there are multiple occurrences of the same tuple, deletes *exactly one such occurrence* from `R`. In other words, we would like to see the content of our `R` changed to:

$$\{(1,2),(3,4),(5,6),(5,6)\}$$

Is it possible to do that using only SQL (as described in Chapter 4 of the textbook)?

Circle NO. SQL can only distinguish tuples based on their different values. Identical tuples will be all treated the same way—e.g., all deleted.

(c) Is it possible to perform the deletions described in (b) using SQL statements embedded in a programming language that support cursors?

Circle YES. The cursor visit the tuples one-at-a-time. The current tuple can be deleted irrespective of other identical tuples.