/

UCLA
Computer Science Department

Instructor: Carlo Zaniolo
Spring 2017

Student Name and ID: ERNEST CHENG , 104659585

CS143 MIDTERM EXAM: Closed Book, 2 Hours

- 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 your 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	
A	(30%)	30
B	(20%)	14
C	(20%)	20
D	(30%)	25
Total	(100%)	89

Extra Credit (6 points): 2

Midterm Score: 91

Problem A: Indexes—30 points

We store the following file in blocks having size 4096 bytes.

customer(id char(20), name char(24))

Our relation has 2 millions of tuples, which are stored unspanned. We create sparse index on id organized as a B+ tree. Pointers take 10 bytes. Please, answer the following questions:

1. How many blocks are needed to store the whole relation?
2. What is the minimum number of nodes needed for the B+ tree?
3. What is the maximum number of nodes needed for the B+ tree, in the worst case scenario?
4. Using the worst-case B+ tree you just constructed, how many blocks must be retrieved to execute the following query:
   SELECT name FROM customer WHERE id =′ 7672945′

1) 44 bytes per tuple

4096 / 44 = 93 tuples / block

2 million / 93 = 21506 blocks

5

2) Sparse index : 21506 blocks.
Pointers take 10 bytes key value takes 20, therefore 30 bytes per key/pointer pair.
Each block can contain (4096 − 10) / 30 = 136 keys and 137 pointer (n = 137)

10

Minimum case:
Max record ptrs at leaf = (137) − 1 = 136

$\lceil \frac{21506}{136} \rceil = 159.$

$\lceil 159/137 \rceil = 2$

$\lceil 2/137 \rceil = 1$

max ptrs at non-leaf

159 + 2 + 1 = 162. min nodes.

next page

3) Maximum case (n = 137).

Min record pointers at leaf = $\left\lceil \frac{137+1}{2} \right\rceil - 1 = 68$

Min pointers at non leaf $\left\lceil \frac{137}{2} \right\rceil = 69$

$\left\lfloor \frac{21506}{68} \right\rfloor = 316$

$\left\lfloor \frac{316}{69} \right\rfloor = 4.$

$320 + 1 = 321$ max nodes

$\left\lfloor \frac{4}{69} \right\rfloor = 1$

10

4) 3 + 1 = 4

index block, relation block

5

Scanned by CamScanner

Problem B: SQL —20 Points

Given the table `taken(StNo, CourseID, Year, Quarter, Sec, Grade, Remarks)`:

write a SQL query to find the students who have taken 4 or more classes and, in every class they took, got a grade that is equal to or lower than the average for that class—a class is identified by `(CourseID, Year, Quarter, Sec)` and `Grade` in `taken` is of type numeric.

```
SELECT StNo
FROM taken as T1
WHERE NOT EXISTS (
    SELECT StNo
    FROM taken as T2
    WHERE T2.grade > ( select AVG(grade)
                       FROM taken as T3
                       WHERE T2.CourseID = T3.CourseID))
GROUP BY StNo
HAVING COUNT(CourseID) >= 4
```

−4

−2

gives average of class in question

gives all rows where a student got above average

taken rows where the StNo doesn't exist in the rows of students that got above average. (∴ equal to lower)

checks that the student has taken more or equal to than 4 classes

Problem C: RA — 20 Points

Given the table `taken(StNo, CourseID, Year, Quarter, Sec, Grade, Remarks)`: write a relational algebra expression to compute all the students who got a grade above 3.0 in every class they took in 2014.

$\pi_{StNo}(\sigma_{year=2014}(taken)) - \pi_{StNo}(\sigma_{year=2014 \wedge grade \leq 3.0}(taken))$

↑ students that took a class in 2014

↑ students that took a class and got below or equal to 3.0 in 2014

Since relational projects unique values, students that took a class in 2014 − students that took a class in 2014 and got below or equal 3.0 will give students who got >3.0 in 2014

25

Problem D: Potpourri — 5 Points each question:

Please answer the following questions as NO or YES. For questions D1–D4, if your answer is YES you must also give the equivalent RA expression.

D1 Can the intersection of relations **R(A,B)** and **S(A,B)** be expressed using only natural joins?

D2 Can the intersection of relations **R(A,B)** and **S(A,B)** be expressed using the set difference operator?

D3 Can the intersection of relations **R(A,B)** and **S(A,B)** be expressed using the cartesian product and projection operators?

D4 Can the intersection of relations **R(A,B)** and **S(A,B)** be expressed using the cartesian product, selection and projection operators?

D5 A relation R is indexed on its candidate key using an extendible hashing: Is this index dense, sparse or it could be either way?

D6 For the extendible hashing on R described in D5: does the structure of its directory and the number of the buckets it uses depend on the order in which the tuples in R have been inserted, or they only depend on the values those tuples ?

D1 Yes $R \bowtie S$ gives the intersection because it naturally joins on both A and B. ✓

D2 Yes $R-(R-S)$ ✓

D3 No. ✓

D4 Yes $\pi_{R.A,R.B}(\sigma_{R.A=S.A \wedge R.B=S.B}(R \times S))$ ✓

D5 Dense. Extendible hashing is always dense ✓

D6 Yes it does, if the candidate key is very similar at the start, buckets will overflow quickly and the number of bits being used will grow faster, but number of buckets used will be less as all buckets are filled completely

✗

Extra Credit Problem: 6 Points

(2)

Our RDBMS has compiled and optimized our relational query into a select-project-join expression consisting of 3 selections, 3 projections and 4 joins. Assume that the relations in our database are in main memory and each contains N tuples or less. Is the worst-case complexity of executing this query $log(N)$, polynomial in N or exponential in N? Justify your answer.

Polynomial N

For each join between N and N, we have to make N^2 comparisons, which is polynomial

selections and projections can be completed in N time ×

∴ polynomial.

Insufficient complexity analysis!