UCLA
Computer Science Department
Fall 2010

Instructor: Carlo Zaniolo
TAs: Suming Chen & Hamid Mousavi

Student Name and ID: _____

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 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**	
A	(32%)	
B	(40%)	
C	(28%)	
Total	(100%)	

Extra Credit (6 points):

Midterm Score:

A: Queries—32 Points

The relation: `warehouse(PartNo, SupplierNo, Price)`
describes the suppliers for each part, along with the price they charge. To cut inventory, the manager of our warehouse wants to eliminate noncompetitive 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.

A1 Write an SQL query to find all non-competitive suppliers. Answer: there are many ways to skin the cat. One is this:

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

Another is this:

```
select SupplierNo
from  warehouse  as w1
where w1.Price <= ALL (select  min(w2.Price)
                              from warehouse as w2
                              where  w2.PartNo=w1.PartNO)
group by SupplierNo having count(w1.PartNo) >=2
```

and we could use the NOT EXIST instead of `ALL`

A2 Modify the previous query by defining competitive suppliers as those that supply at least two parts at the strict minimum price (ties now are not allowed!).

Answer: It is much easier to work on the second solution, which we can modify as follows:

```
select SupplierNo
from  warehouse  as w1
where w1.Price <ALL (select  w2.Price
                                 from warehouse as w2
                                 where  w2.PartNo=w1.PartNO
                                 AND    w1.SupNO <> W2.SupNO)
group by SupplierNo having count(w1.PartNo) >=2
```

This basically assures that any other another supplier who supplies the same part charges more than this supplier. If we try a similar modification on the min query we get the following WRONG query!

```
select SupplierNo
from  warehouse  as w1
where w1.Price < min (select  min(w2.Price)
                              from warehouse as w2
                              where  w2.PartNo=w1.PartNO
   AND    w1.SupNO <> W2.SupNO)
group by SupplierNo having count(w1.PartNo) >=2
```

This query says that this supplier charges less than other suppliers supplying this part. But if there no other supplier the min is undefined and we will eliminate the only supplier for that part! If we want to use the min, we must instead write:

```
select SupplierNo
from  warehouse  as w1
where w1.Price < min (select  min(w2.Price)
                              from warehouse as w2
                              where  w2.PartNo=w1.PartNO)
           OR  not exists (select * from warehouse as w3 where  w3.PartNo=w1.Part
group by SupplierNo having count(w1.PartNo) >=2
```

A3 Write an SQL statement to delete from the warehouse all the tuples where the suppliers are non-competitive (as defined in A2) and also they are not the sole supplier of that part.

Answer:

```
delete from warehouse AS NC
  where NC.SupplierNo IN % copy SQL from A2
  AND exists (select * from warehouse as w3 where  w3.PartNo=NC.PartNO AND w3.SupN
```

A4 Is query A2 expressible in basic RA, which does not have aggregates. You do not need to write the query, just explain how you will express the conditions involving "minimum cost" and "at least two"

Answer: `having count(w1.PartNo) >=2` can be expressed using a `<>`-join, whereas the min can be expressed using negation.

B–Indexes. 40 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 a StudentID. On average, a student took 40 different courses.

B1 If the file blocks hold 4096 bytes and each tuple in `took` requires 200 bytes. How many blocks will then be needed to store the unspanned tuples of this relation ?

Answer: 4096/200 =20.48: thus 20 unspanned tuples per block. Now: 8,000,000/20= 400,000 blocks.

There is a **sparse** index on `StudentID, CourseNo`, where `StudentID, CourseNo` and the pointer to the file take 10 bytes each, and the index is implemented as a B+ tree.

B2 Compute the levels and the blocks at each level of the B+ tree, assuming a best-case scenario.

Best Case: (4096-10)/30= 136.2; Thus, N=137. Now 400,000/136 = 2,941.17, i.e., 2942 at the bottom level. 2942/137= 21.47, i.e. 22 blocks at first level. Then the root.

B3 Compute the levels and the blocks at each level of the B+ tree, assuming a worst-case scenario.

Worst case: 68 pointers down. 400000/68 = 5 882.35 i.e. 5822 in the middle level: 5,883/69 = 85.246 i.e. 85 (with 86 less than half full). Then the root.

B4 How many blocks from the B+ tree and file will the DBMS retrieve from disk to answer the following query: *Find the GPA (average grade weighted by units) for a given student.* Report your results first assuming the best-case and then assuming the worst-case scenario; also assume that the buffers are initially empty.

Best case: 3 blocks from the B+ tree and then one file block (possibly two if a student's courses are spread over two blocks) Worst case: the same.

B5 We now have that computes the average grade that all students who took CS144 got in CS144. How many blocks from the B+ tree and file will the DBMS retrieve from disk to answer this query? Report your results first assuming the best-case and then assuming the worst-case scenario; also assume that the buffers are initially empty.

The DBMS will follow the leftmost link in the B+ tree and then the chain at the bottom level, and from there, jump to all the blocks in the file .

So 2 blocks down the B+ tree, then all 2920 blocks at the bottom level of the B+ tree, and finally 400,000 blocks of file for best case B+ tree. For the worst case, replace 2920 with 5420.

(For a dense index we could try to find all the occurrences of "CS144" in the index and only fetch those blocks.)

C: Potpourri—28 Points

Please indicate if the following statements are TRUE or FALSE. You must write a short sentence explaining your answer:

C1 The index of problem B is a clustered index.

*TRUE, Since the index is sparse the records in disk must be ordered. This implies that the index is of clustered type.

C2 Sequential I/O is more expensive than random I/O.

*FALSE. The opposite holds: random access incur delay of the arm movement (before the rotational delay of the sequential access)

C3 To modify a block on disk, the block must be read into memory first.

*TRUE, unless we just overwrite it with completely new content.

C4 For queries of the form $\sigma_{A>k}(R)$ hashing is better than B+ trees.

*FALSE. Hashing works great for equality-based searches, but does not help at all on order-based searches since it does not preserve the order of recordes.

C5 With extensible hashing we never have to use overflow buckets.

*FALSE. When many records share the same key value, and they are hashed into the same bucket which may cause the overflow into supplementary buffers.

C6 The intersection of relations R(A,B) and S(A,B) can be expressed using the set set difference operator.

*TRUE: $R \cup S = R - (R - S)$

C7 The symmetric outer join of R1 and R2 can have fewer tuples than R1.

*FALSE. Every time that a tuple in R1 is not matched by a tuple in R2, we preserve it by padding it with null values.

Extra Credit [6 points]

Say that we execute an SQL query Q on a database containing N tuples that, because of previous queries, are already in main memory buffers. Our DBMS process Q by first translating it into an RA expression E, and then letting the DBMS query optimizer optimize E. Say that E contains M operators. Please, answer the following questions and explain your answers:

- Is the worst-case optimization of E exponential or polynomial in M?

  It is exponential on the number of tables in the join. The optimizer will enumerate all their permutations to find the one that has the least cost! The number of permutations of M elements grows exponentially with M (actually faster than that).

- Is the worst case execution of Q (i.e., the execution of E after optimization exponential or polynomial in N?

  All the relational operators are polynomial time on N, the size of the table. M of those is still polynomial considering M is independent from N. (This even holds for the case of having no optimization)–particularly after the smart optimizer has figured out how to minimize the exponents and coefficient of that polynomial.

- Is the worst-case execution of Q exponential or polynomial in M?

  If we take the cartesian product of M tables of size K we obtain K^M tuples. So, it is exponential.

  Conclusion: Queries that use joins of more than 10 tables are too expensive to optimize and execute! Of course, users do not write such complex queries– unfortunately poorly designed software might end up doing that!