CompSci 516 Fall 2019: Midterm Exam

Solution

Problem 1

1a: T. One is a superkey.

1b: F. Both can be keys.

1c: T. With single attribute, join is intersection.

1d: F. It depends on the query workload, whether the hash function is a good one, whether the data has skew etc.

1e: F. All should be present.

1f: T. Prefix.

Problem 2

2a: no, If there are multiple R rows that join with the same S row, then the first query will return multiple copies of the same S row, while the second query still returns one copy.

2b: no. The join of R1, R2 and projecting to A, C will always be a subset of R3.

2c: 1st: yes, (True OR Unknown) AND True = True AND True = True

2nd: no, (Unknown OR True) AND Unknown = Unknown

2d:

Pass	Runs	Size
0	ceil(100/6) = 17	16 runs will have 6 pages Last run will have 4 pages
1	ceil(17/5) = 4	5 runs at a time are merged 3 runs will have 5*6=30 pages Last run will have (6+4)=10 pages
2	ceil(4/5) = 1	4 runs can be merged with 5 input buffer 100 pages

Cost = 2N(# of passes) = 2*100*3 = 600

If you ignore the cost of final write, cost = 500.

We give points whether or not the cost of final write is ignored.

Problem 3

SELECT drinker

FROM Likes JOIN Beer ON Likes.beer = BeerInfo.beer

GROUP BY drinker

HAVING MAX(price) <= 4 AND COUNT(beer) >= 10

Problem 4.

4a:

FBE

B, E, and F all must belong to every key because they never appear on the right-hand side of any given functional dependency. It turns out that $\{B, E, F\}$/ covers all attributes in the relation, so $\{B, E, F\}$ is the only key.

4b: i

$AB \rightarrow C$ follows from $AB \rightarrow D$ and $BD \rightarrow C$.

4c:

Pick F->A

Get R1(BCDEF) R2(FA)

Pick BD -> C on R1

Get R3(BCD) R4(BDEF)

Final answer: R2, R3, R4.

4d:

no.

Example:

R(A B C D E F)

A1 b1 c1 d1 e1 f1

A1 b2 c2 d2 e2 f2

If you decompose into R1, R2 and join them back on A, four tuples will be generated instead of 2.

Problem 5.

5a:

Step	Cost	Size
(i) Index nested loop join	910	900
(ii) selection on age	0	100

5b:

Step	Cost	Size
(i) Index nested loop join	10+900*1	900
(ii) selection on age	0	900*1/9 = 100

There is no cost to access the index pages as the index pages are in memory

Step	Cost	Size
(i) Index nested loop join	IO = # outer pages + (# outer tuples) * (# inner tuples matched). Since bid is the primary key of B, (# inner tuples matched) is 1. So we have $B_C + T_C * 1 = 10 + 900 * 1 = 910$	Since it's a foreign-key join, the size of the output is the size of input relation that has the foreign key. Here, relation C has a foreign-key bid to B, so the output size is $T_C = 900$
(ii) selection on age	Since we assume the result of (i) is in memory, there is no IO cost to do selection	We assume price is uniformly distributed, so the selectivity is | [20,29] | / | [10, 99] | = 1/9. Since the output size of (i) is 900, here we have 900*1/9 = 100

5c:

No change. B has the key bid for foreign key C.bid. So there will be only one match. Therefore unclustered = clustered

5d:

(1) Each book on average has 9 checkout = 900/100. One C page can hold 900/10 = 90 tuples.

For clustered, 9 tuples will fit in one page. So 1 page access. If you consider page boundaries, 2 pages at most. So total cost is

20 + 100 * 1 = 120

or 20 + 100 * 2 = 220 (we gave points for both).

(2) Yes.

at worst case for one tuple in B it needs to fetch 900/100 = 9 tuples which could be in 9 different pages, compared to only 1 or 2 pages when it is clustered.

Cost = 20 + 100 * 9 = 920.

6a: The drinker table was not given, so the possible drinkers can only be obtained from the Friends relation (used below) or Likes relation.

$$
\begin{aligned}
&\{ D \mid \exists\ F \in \text{Friends}\ (F.d1 = D.drinker) \wedge \\
&\qquad \forall\ L \in \text{Likes}\ ((L.drinker = F.d1) \Rightarrow \\
&\qquad (\exists\ B \in \text{BeerInfo}\ (B.beer = L.beer) \wedge (B.bottle\text{-}color = \text{‘red’})) \\
&\qquad \wedge \\
&\qquad (\forall\ F1 \in \text{Friend}\ (F1.d1 = F.d1) \Rightarrow \\
&\qquad\qquad (\exists\ L1 \in \text{Likes}\ (L1.drinker = F.d2) \wedge (L1.beer = B.beer)) \\
&\qquad) \\
&\qquad) \\
&\}
\end{aligned}
$$

6b.

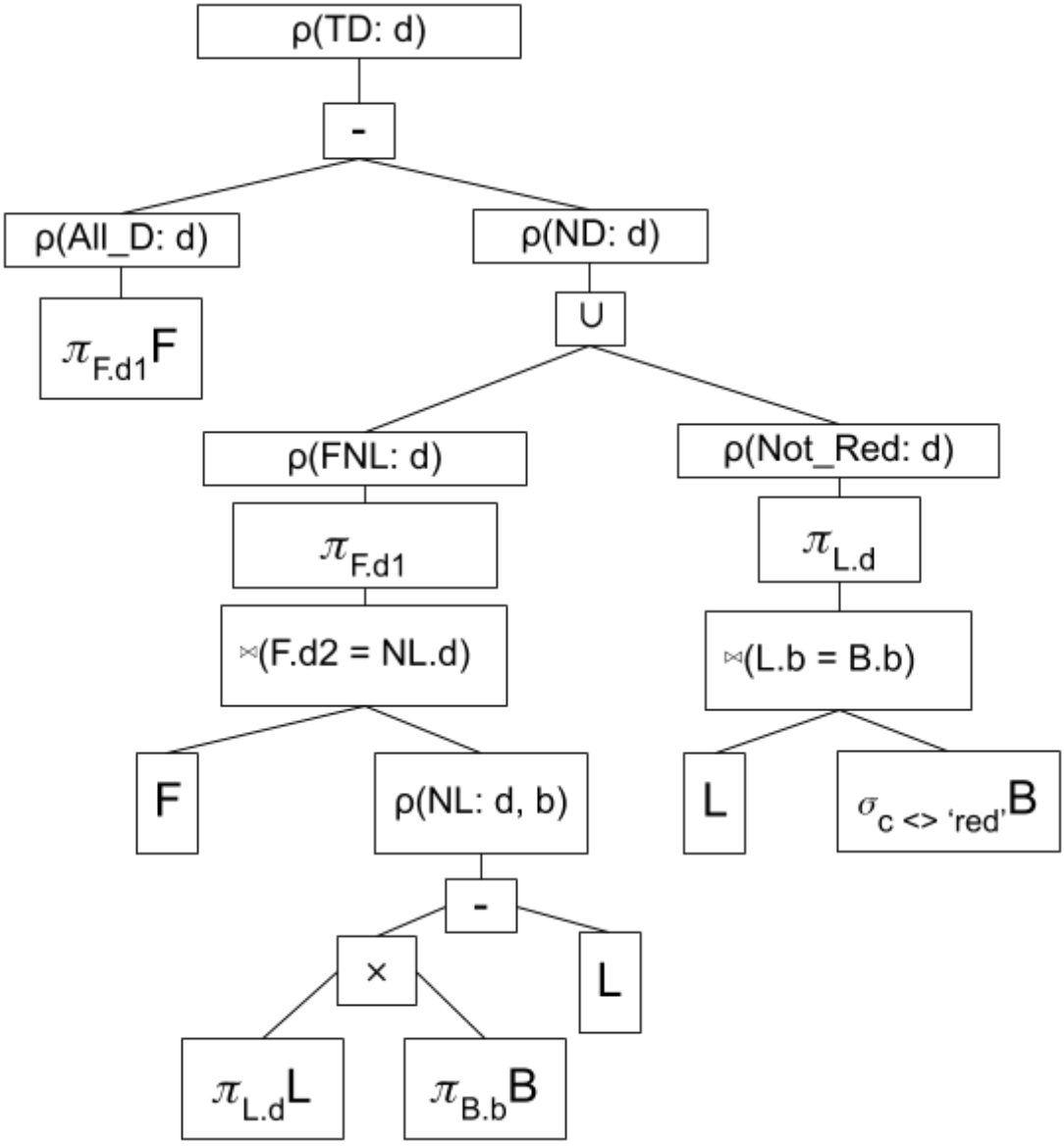
Select all drinkers d1 from friends such that there are no Likes tuple for this drinker (1) that is not in red bottle, **OR** (2) there is a friend who does not like that beer

```
SELECT F.d1
FROM Friends F
WHERE NOT EXISTS  /* F.d1 should not be selected if such a Likes tuple exists */
        (SELECT *
        FROM LIKES L1
         WHERE L1.drinker = F.d1 AND
            (L1.bottle-color <> 'red'  /* condition (1), F.d1 should not be selected if the beer in
                                          L1 is not in red bottle */

               OR
             EXISTS              /* condition (2), F.d1 should not be selected if there is a friend who
                                    does not like L1.beer */

                (SELECT *
                FROM FRIENDS F1
                WHERE F1.d1 = F.d1
                        AND NOT EXISTS
                        (SELECT *
                        FROM LIKES L2
                        WHERE L2.beer = L1.beer
                                AND L2.drinker = F1.d2
                        )
                    )
                )
        }
```

Another solution OF 6b from the RA plan in 6c:

```
SELECT drinker FROM Likes WHERE drinker not in
        (
                (SELECT F.d1 as drinker
                 FROM Friends F JOIN
                                ( SELECT drinker FROM
                                                (SELECT drinker FROM L) as Ld,
                                                (SELECT beer from B) as Bb
                                EXCEPT Likes.drinker) AS NL
                                ON F.d2 = NL.drinker
                ) AS FNL
                UNION
                ( SELECT drinker
                  FROM Likes L join Beer B on L.beer = B.beer
                  WHERE bottle-color <> 'red'
                ) AS Not_Red
        )
```

6C. RA


ρ(TD: d)
-
ρ(All_D: d)
ρ(ND: d)
$\pi_{F.d1}$F
U
ρ(FNL: d)
ρ(Not_Red: d)
$\pi_{F.d1}$
$\pi_{L.d}$
⋈(F.d2 = NL.d)
⋈(L.b = B.b)
F
ρ(NL: d, b)
L
$\sigma_{c <> 'red'}$B
-
×
L
$\pi_{L.d}$L
$\pi_{B.b}$B

(All drinkers that like some beer x beer) - L

= (drinker, beer) pairs such that drinker does not like beer. = NL

F Join NL on (d2=d) = (d1, d2 = d, b) pairs such that (d1, d2) are friends and d2 does not like b = FNL say

Project to d -- drinker with at least one friend who does not like a beer that d likes

Union with drinkers d who like at list one beer bottled in non-red bottles

Subtract the union from F.d1, so that drinkers satisfying either non-red or friend-doesnotlike-a-liked-beer are eliminated.