db hw10

3200105872 庄毅非

15.2 Consider the bank database of Figure 15.14, where the primary keys are un- derlined, and the following SQL query:

```
select T.branch\_name
from branch T, branch S
where T.assets > S.assets and S.branch_city = “Brooklyn”
```

Write an efficient relational-algebra expression that is equivalent to this query. Justify your choice.
Answer:

$$\Pi_{T.branch\_name}(\Pi_{(assets,branch\_name)}(\rho_T(branch)) \underset{(T.assets>S.assets)}{\bowtie} (\Pi_{assets}(\sigma_{branch\_city='Brooklyn'}(\rho_S(branch)))))$$

In this way, the two operation sets for performing the join operation are minimized, and the amount of data passed during the operation is reduced, and the simpler selection operation is performed first, reducing the cost of subsequent projection and join operations. In the end, the total cost of all operations is minimized.

15.3 Let relations *r* 1(*A* , *B* , *C*) and *r* 2(*C* , *D* , *E*) have the following properties: *r* 1 has 20,000 tuples, *r* 2 has 45,000 tuples, 25 tuples of *r* 1 fit on one block, and 30 tuples of *r* 2 fit on one block. Estimate the number of block transfers and seeks required using each of the following join strategies for *r* 1 ⋈ *r* 2:

1. Nested-loop join.

Answer: Firstly, the number of disk blocks occupied by r1 is b1 = 20000 / 25 = 800, and the number of disk blocks occupied by r2 is b2 = 45000 / 30 = 1500.

If buffer size M > 800, then we need 800 + 1500 = 2300 disk accesses. Else the size of buffer is less or equal to 800. Assume r1 is the outer relation, then our accesses to the disk is 800 + 20000 * 1500 = 30000800. If r2 is the outer relation, then our accesses to the disk is 1500 + 800 * 45000 = 36001500.

b. Block nested-loop join:
Answer:

If buffer size M > 800, then we need 800 + 1500 = 2300 disk accesses.
Else if r1 is the outer relation, we need $\lceil 800/(M-1) \rceil$ * 1500 + 800 disk accesses.
Else if r2 is the outer relation, we need $\lceil 1500/(M-1) \rceil$ * 800 + 1500 disk accesses.

c. Merge-join:
Answer:

If buffer size M > 800, then we need 800 + 1500 = 2300 disk accesses.
If r1 and r2 are sorted, we need 800 + 1500 disk acccesses.
Else we need firstly sort r1 and r2, and we need write back to disk, whose cost is $2 * 800(1\lceil log_{M-1}(800/M) \rceil + 1) + 2 * 1500(1\lceil log_{M-1}(1500/M) \rceil + 1)$ disk accesses. Then we need another 1500 + 800 = 2300 disk accesses. So our final result is $2 * 800(1\lceil log_{M-1}(800/M) \rceil + 1) + 2 * 1500(1\lceil log_{M-1}(1500/M) \rceil + 1) + 2300$

4. Hash join.

Answer: Assume r1 is the build relaton and r2 is the probe relation. If M > 800 / M + 1, then we don't need to recursively partition. The number of access of disk is 3 * (800 + 1500) = 6900, else number of access of disk is $2 * (800 + 1500) * (\lceil log_{M-1}(800) \rceil - 1) + 800 + 1500 = 4600 * (\lceil log_{M-1}(800) \rceil - 1) + 2300$