CSE 444: Homework 5

Linxing Preston Jiang   Winter 2018

February 7, 2018

---

1 Query Plan Cost Computation

1. **(a)** $\frac{1000-100}{9000-0} = \frac{1}{10}$

   (b) $\frac{1}{V(S,e)} \cdot \frac{1}{V(S,f)} = \frac{1}{10} \cdot \frac{1}{100} = \frac{1}{1000}$

   (c) $\frac{1}{\max\{V(R,c), V(S,d)\}} = \frac{1}{50}$

   (d) $\frac{1}{V(R,b)} + \frac{1}{V(R,b)} = \frac{1}{50}$

   (e) $\frac{1}{\max\{V(S,g), V(T,h)\}} = \frac{1}{100}$

2.
   - $|R_1| = 10000 \cdot 0.1 = 1000$
   - $|R_2| = 10000 \cdot \frac{1}{1000} = 10$
   - $|R_3| = \frac{1000 \cdot 10}{50} = 200$
   - $|R_4| = \frac{200}{50} = 4$
   - $|R_5| = \frac{4 \cdot 10000}{100} = 400$
   - $|R_6| = |R_5| = 400$

3. **(a)** Because of the clustered index on a, we have $B(R) \cdot 0.10 = 100$ IOs to read from R. And the unclustered index on (e, f) will help reduce the number of IOs to $B(S) \cdot \frac{1}{1000} = 10$ pages. We do not write intermediate results to disk, so the next IO cost comes from reading from T, which will have $T(R_4) \cdot \frac{B(T)}{V(T,h)} = 4 \cdot \frac{1000}{100} = 40$. There the total cost is

$$100 + 10 + 40 = 150$$

2 Query Optimization

1. See table

Subqeury	Cost	Size of ouptut	Plan	P/K	
R	100 page IOs	1K records on 100 pages	Sequential scan of R	K	
S	1K page IOs	10K records on 1K pages	Sequential scan of S	K	
W	10 page IOs	100 records on 10 pages	Sequential scan of W	K	
RS	100 + 100 * 1000 = 100100 page IOs	10K records on 2K pages	Nested loop join of R (outer) S	K	
RW	100 + 100 * 10 = 1100 page IOs	100 records on 20 pages	Nested loop join of R (outer) W	P (WR has lower IO cost)	
SR	1000 + 1000 * 100 = 101000 page IOs	10K records on 2K pages	Nested loop join of S (outer) R	P (RS has lower IO cost)	
WR	10 + 10 * 100 = 1010 page IOs	100 records on 20 pages	Nested loop join of W (outer) R	K	
~~SW~~	~~...~~	~~...~~	~~...~~	~~P~~	Cartesian Product
~~WS~~	~~...~~	~~...~~	~~...~~	~~P~~	
RSW	100100 + 2K + 2K * 10 = 122,100	1K records on 300 pages	Nested loop join of RS (outer) W	P	
WRS	1010 + 20 + 20 * 1K = 21030	1K records on 300 pages	Nested loop join of WR (outer) S	K	Best

2.
   - Clustered index on `R.b`: Because of the selection `R.b > 100`, this index will help reduce the number of reads from `R` by the selectivity of `R.b > 100`.
   - Unclusted index on `S.d`: It depends on the estimated number of tuples from `R` to be read. Because we are joining `R.a` and `S.d`, if the number of tuples from `R` is more than the number of pages from `S`, we should just perform a file scan because every read from an unclustered index can be on a new page. Otherwise, we can choose to use this unclustered index to reduce the number of reads from `S`.

3. An interesting order means an intermediate join which has a higher cost leads to a lower cost in the end. It can either be the order of tuples satisfying the order required by `ORDER BY` or `GROUP BY`, or the order on a equality-join will enable cheaper sort-merge join in the next step. For example, assume we have a hash index on `R.a`, a B+ tree clustered on `W.c`, see the following query:

```
SELECT *
FROM R, S, W
WHERE R.a = S.b
  AND S.b = W.c;
```

For the step of joining `R` and `S`, an interesting order can be obtained by using a sorted-merge join, because for the next step of joining with `W.c`, the tuples of `S.b` is already sorted and so is `W.c` using the B+ tree clustered index. Here's another example:

```
SELECT SUM(R.a)
FROM R, S, W
WHERE R.a = S.b
  AND R.c = W.c
GROUP BY S.b;
```

Now an interesting order is using sorted-merge-join on `R.a` and `S.b` because it will dramatically speed up the `GROUP BY` clause since every tuple is sorted on the `GROUP BY` field S.b.