CS W186 Introduction to Database Systems
Spring 2020 Josh Hug, Michael Ball

DIS 7

1 Selectivity Estimation

Consider a relation R(a, b, c) with 1000 tuples. We have an index on a with 50 unique values in the range [1, 50] and an index on b with 100 unique values in the range [1, 100]. We do not have an index on c.

Use selectivity estimation to estimate the number of tuples produced by the following queries.

1. SELECT * FROM R

   1000 (no predicates, so select all tuples)

2. SELECT * FROM R WHERE a = 42

   Sel = 1 / unique values of a = 1/50
   1000 * (1/50) = 20

3. SELECT * FROM R WHERE b = 42

   Sel = 1 / unique values of b = 1/100.
   1000 * (1/100) = 10

4. SELECT * FROM R WHERE c = 42

   Sel = 1/10 because we do not have any information on the relation we use 1/10 as default.
   1000 * (1/10) = 100

5. SELECT * FROM R WHERE a <= 25

   Sel = (v – low) / (high – low + 1) + 1 / distinct values = (25 – 1) / (50 – 1 + 1) + 1/ 50 = 1/2
   1000 * (1/2) = 500

6. SELECT * FROM R WHERE b <= 25

   Same formula as part 5, or just look and see that the first 25 values are 1/4 of the values between 1 and 100: Sel = 1/4
   1000 * (1/4) = 250

7. SELECT * FROM R WHERE c <= 25

   Sel = 1/10
   1000 * (1/10) = 100

8. SELECT * FROM R WHERE a <= 25 AND b <= 25

   Sel = Sel(a <= 25) * Sel(b <= 25) = 1/2 * 1/4 = 1/8
   1000 * (1/8) = 125

9. SELECT * FROM R WHERE a <= 25 AND c <= 25

   Sel = Sel(a <= 25) * Sel(c <= 25) = 1/2 * 1/10 = 1/20
   1000 * (1/20) = 50

10. SELECT * FROM R WHERE a <= 25 OR b <= 25

    Sel = Sel(a <= 25) + Sel(b <= 25) – Sel(a <= 25) * Sel(b <= 25) = 1/2 + 1/4 – 1/2 * 1/4 = 5/8
    1000 * (5/8) = 625

11. SELECT * FROM R WHERE a = b

    Sel = 1/max(distinct values of a, distinct values of b) = 1/max(50, 100) = 1/100
    1000 * (1/100) = 10

12. SELECT * FROM R WHERE a = c

    We don't have information on c, so we just use the number of distinct values in a: Sel = 1/50
    1000 * (1/50) = 20

For the rest of this worksheet we will try to optimize the following query:

```
SELECT *
FROM R, S, T
WHERE R.b = S.b AND S.c = T.c
AND R.a <= 50;
```

We have 3 relations: R(a,b), S(b,c), and T(c,d).

- R has 1,000 data pages and 10,000 records
- S has 2,000 data pages and 40,000 records
- T has 3,000 data pages and 30,000 records

2 Single Table Access Plans

Assume we have the following indexes:

- Alt 2 unclustered index on R.a with 50 leaf pages
- Alt 2 clustered index on R.b with 100 leaf pages
- Alt 2 clustered indexes on S.b, T.c, and T.d (leaf page counts aren't relevant)

Also assume that it takes 2 IOs to reach the level above a leaf node and that no index or data pages are ever cached. All indexes have keys in the range [1, 100] with 100 distinct values.

1. How many IOs does a full scan on R take?
   1000 IOs, need to read every data page.

2. How many IOs does an index scan on R.a take?
   2 + 0.5(50) + 0.5(10000) = **5,027 IOs**. The selectivity of the R.a $\leq$ 50 condition is 0.5 so we multiply the leaf/data pages by 0.5. The index is unclustered so you have to do an IO for every record.

3. How many IOs does an index scan on R.b take?
   2 + 100 + 1000 = **1102 IOs**. This index is clustered so we only have to do an IO for every page. We can't apply the selectivity of the R.a $\leq$ 50 condition to the leaf/data pages because the index is built on a different column, so we're going to need to read all the pages.

4. How many pages from R will advance to the next stage for all of these access plans?
   500. The selectivity is 1/2 so only half the pages will advance.

Now assume that the other potential single table access plans have the following IO costs:

- Full scan on S: 2000 IOs
- Index scan on S.b: 1500 IOs
- Full scan on T: 3000 IOs
- Index scan on T.c: 3500 IOs
- Index scan on T.d: 3500 IOs

5. What single table access plans advance to the next stage?
- Full scan on R (best overall plan for R so it must advance)
- Index scan on R.b (interesting order because R.b is used in a join)
- Index scan on S.b (best overall plan for S so it must advance)
- Full scan on T (best overall plan for T so it must advance)
- Index scan on T.c (produces an interesting order on T.c so it must advance)

3 Multi-table Plans

1. Assume we have 52 buffer pages. Calculate the IO cost for joining the following 2 joins. Assume the access plan for R is the index scan on R.b (S only had one access plan that advanced):

a. R $BNLJ_{R.b=S.b}$ S
   500 + (500/50)(2000) = **20,500**. Only 500 pages advanced from R on the first pass so we use that value in the formula.

b. $RSMJ_{R.b=S.b}$ S
   500 + 2000 = **2500**. We don't need to sort either table because the access plans from pass 1 produced the output in sorted order for both of these.

Assume all of the joins our database could do are as follows:

1. R BNLJ S: 1.a
2. R SMJ S: 1.b
3. S BNLJ R: 18,000 IOs
4. S SMJ R: 3,000 IOs
5. R BNLJ T: 30,000 IOs
6. R SMJ T: 40,000 IOs
7. T BNLJ R: 35,000 IOs
8. T SMJ R: 20,000 IOs
9. S BNLJ T: 15,000 IOs
10. S SMJ T: 10,000 IOs
11. T BNLJ S: 25,000 IOs
12. T SMJ S: 30,000 IOs

2. Which of these joins will actually be considered by the query optimizer on pass 2?
1, 2, 3, 4, 9, 10, 11, 12. We don't consider joins requiring a cross join, but every other join will be considered. R and T require a cross join because there is no join condition, so we will not consider any join involving R and T.

3. Which of these joins will advance to the next pass of the query optimizer?
2, 10. None of these joins produce an interesting order. This is because a Sort Merge Join is the only join that produces data in sorted order and it is sorted on the columns in the join condition. If we SMJ R and S the data will be sorted on b, but b isn't used in another join condition, ORDER BY, or GROUP BY clause. The same idea applies for SMJ S and T which produces the data in sorted order on c. Therefore only the best join for each considered set of tables is advanced which is 2 and 10.

4. Will any of these joins produce an interesting order?
No (see explanation above)

5. How could we modify the query so that the R SMJ S produces an interesting order?
R SMJ S will be sorted on column b so we need b to be interesting. We could add ORDER BY b, GROUP BY b, or another join condition involving R.b or S.b to the query.

6. Will the query plan: T BNLJ (R SMJ S) be considered by the final pass of the query optimizer?
No, this query plan is not left deep (all joined tables must be on the left side of all joins for it to be left deep) so it is not considered.