CS W186 Introduction to Database Systems
Fall 2019 Josh Hug

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
2. SELECT * FROM R WHERE a = 42
3. SELECT * FROM R WHERE b = 42
4. SELECT * FROM R WHERE c = 42
5. SELECT * FROM R WHERE a <= 25
6. SELECT * FROM R WHERE b <= 25
7. SELECT * FROM R WHERE c <= 25
8. SELECT * FROM R WHERE a <= 25 AND b <= 25
9. SELECT * FROM R WHERE a <= 25 AND c <= 25
10. SELECT * FROM R WHERE a <= 25 OR b <= 25
11. SELECT * FROM R WHERE a = b
12. SELECT * FROM R WHERE a = c

2 Single Cost Plans

Consider the relations R(a, b), S(b, c), and T(c, d) with an alt 1 index on R.a, alt 2 clustered indexes on S.b, and T.c, and alt 2 unclustered indexes on S.c and T.d. Assume it takes 2 IOs to reach a leaf node and no index or data pages are ever cached. All indexes have index keys in the range [1, 100].

We want to optimize the following query:

SELECT *
FROM R, S, T
WHERE R.b = S.b AND S.c = T.c
AND R.a <= 50
AND (T.c <= 50 AND T.d <= 20)

In the first pass of the Sellinger query optimization algorithm, we compute the minimum cost access method for every (relation, interesting order) pair. Complete the following table which computes this. Let [R], [S], and [T] and |R|, |S|, and |T| be the number of pages and tuples in R, S, and T, respectively. Let [S.b], [T.c], [S.c], and [T.d] be the number of leaf pages in each index corresponding index.

Relation	Access Method	Interesting Order	I/O Cost	Output Size	Retained
R	Full Table Scan	None	[R]	0.5[R]	No
	Index Scan on a	None	2 + 0.5[R]		Yes
S	Full Table Scan				
	Index Scan on b				
	Index Scan on c				
T	Full Table Scan				
	Index Scan on c				
	Index Scan on d				

3 Multi Table Plans

1. In the second pass of the Sellinger query optimization algorithm, we consider each (relation, interesting order) pair in turn. For each, we compute the cost of joining every other relation into it. In the end, we retain the minimum cost join for each (relations, interesting order) pair. Complete the following table which computes **part** of the second pass. Let B be number of buffer pages, use SC(t) for the sorting cost of table t, assume we take the ceiling of all fractions, and assume that there are not many duplicates in the join columns.

Left Relation	Left Ordering	Right Relation	Right Ordering	Join Type	Interesting Order	IO Cost of the Join	Output Size
R	(a)	S	(b)	BNLJ	None	0.5[R] + $\frac{0.5[R]}{B-2}$[S]	$\frac{0.5[R][S]}{100}$
				SMJ	None	SC(0.5[R]) + 0.5[R] + [S]	
S	None	R	None	BNLJ	None	[S] + $\frac{S}{B-2}$0.5[R]	$\frac{0.5[R][S]}{100}$
				SMJ	None	SC(0.5[R]) + SC([S]) + 0.5[R] + [S]	
S	None	T	(c)	BNLJ			
				SMJ			
S	(b)	R	(a)	BNLJ			
				SMJ			
S	(b)	T	(c)	BNLJ			
				SMJ			

2. Is it possible that the SMJ of the bottom row ever advances to the next pass of the query optimization algorithm?

3. Add a condition to the WHERE clause so that the SMJ of R and S produces an interesting order.

4. Why don't we consider joining R and T in our table?