6022-15-9E AID:258164 | 28/08/2020

**The calculation of the given problem are as follows:**

1. These question the in tuple Emp and Proj there is id and sal and code are in identical datatype. In these Emp tuple which is fitted for single page and Proj tuple which is large it is fitted on page.

Suppose speeding up the queries for clustered as compared to unclustered index means minimize the number of input output pages. It gives when the Emp tuples are clustered w.r.t. the age index . matching tuples are retrieved in some input output of pages. But in Proj tuples it is unclustered , so matching tuples will require input output of pages. It is same as clustering.

2. As explain above , in unclustered index , the matching tuples retrieve few input output of pages. Means N tuples requires N input output of pages. Let consider, Emp tuple occupy the 200 pages.

The index data entries exceeds the cost of sequential scan these can fetching the Emp tuple of cost when tuples are more than 200. this makes scan better than the index with fewer than 200 matches. And this index are involve the right leaf page and fetching cost of leaf page which qualifies the data.

3. (a) One plan is to use (simple or blocked) NL join with E as the outer. Another plan is SM or Hash join. A third plan is to use D as the outer and to use INL; given the clustered hash index on E, this plan will likely be the cheapest.

(b) The same plans are considered as before, but now, SM join is the best strategy because both relations are sorted on the join column (and all tuples of Emp are likely to join with some tuple of Dept, and must therefore be fetched at least once, even if INL is used).

(c) The same plans are considered as before. As in the previous case, SM join is the best: the clustered B+ tree index on Emp can be used to eﬃciently retrieve Emp tuples in sorted order.

4. (a) BNL with Proj as the outer, followed by sorting on did to implement the aggregation. All attributes except did can be eliminated during the join but duplicates should not be eliminated!

(b) Sort Dept on did ﬁrst (all other attributes except projid can be projected out), then scan while probing Proj and counting tuples in each did group on-the-ﬂy.

(c) INL with Dept as inner, followed by sorting on did to implement the aggregation. Again, all attributes except did can be eliminated during the join but duplicates should not be eliminated!

(d) As in the previous case, INL with Dept as inner, followed by sorting on did to implement the aggregation. Again, all attributes except did can be eliminated during the join but duplicates should not be eliminated!

(e) Scan Dept in did order using the clustered B+ tree index while probing Proj and counting tuples in each did group on-the-ﬂy.

(f) Same as above. Scan Dept in did order using the clustered B+ tree index while probing Proj and counting tuples in each did group on-the-ﬂy.

(g) Scan the clustered B+ tree index using an index-only scan while probing Proj and counting tuples in each did group on-the-ﬂy.

(h) Sort the data entries in the clustered B+ tree index on Dept, then scan while probing Proj and counting tuples in each did group on-the-ﬂy.

5. (a) BNL with Proj as the outer with the selection applied on-the-ﬂy, followed by sorting on did to implement the aggregation. All attributes except did can be eliminated during the join but duplicates should not be eliminated!

(b) Sort Dept on did ﬁrst (while applying the selection and projecting out all other attributes except projid in the initial scan), then scan while probing Proj and counting tuples in each did group on-the-ﬂy.

(c) Select Dept tuples using the index on budget ,join using INL with Proj as inner , projecting out all attributes except did. Then sort to implement the aggregation.

(d) Same as the case with no index; this index does not help.

(e) Retrieve Dept tuples that satisfy the condition on budget in did order by using the clustered B+ tree index while probing Proj and counting tuples in each did group on-the-ﬂy.

(f) Since the condition on budget is very selective, even though the index on budg et is unclustered we retrieve Dept tuples using this index, project outthe did and projid ﬁelds and sort them by did. Then we scan while probing Proj and counting tuple sin each did gorup on-the-ﬂy.

(g) Use an index-only scan on the B+ tree and apply the condition on budget ,while probing Proj and counting tuples in each did group on-the-ﬂy. Notice that this plan is applicable even if the B+ tree index is not clustered. (With in each did group, can optimize search for data entries in the index that satisfy the budget condition, but this is a minor gain.)

(h) Use an index-only scan on the B+ tree and apply the condition on budget, while probing Proj and counting tuples in each did group on-the-ﬂy.

6. (a) 1-relation subplans:

Clustered index on E.sal ; Scan Dept ; and Scan Proj.

2-relation subplans :

(i) unclustered index on E.sal ,probe Dept using the index on did, apply predicate on D. budget and join.

(ii) Scan Dept, apply predicate on D. budget and prob e Proj.

(iii) Scan Proj, probe Dept and apply predicate on D .budget and join.

3-relation subplans:

Join Emp and Dept and probe Proj; Join Dept and Proj and probe Emp.

(b) The least cost plan is to use the index on E.sal to eliminate most tuples, probe Dept using the index on D .did, apply the predicate on D .budget, probe and join on Proj .projid.

(c) Unclustering the index on Proj would increase the number of I/Os but not substantially since the total number of matching Proj tuples to be retrieved is small.