Query Evaluation & Optimization

Exercise 15.2 Consider a relation with this schema:

Employees(*eid:* **integer**, *ename:* **string**, *sal:* **integer**, *title:* **string**, *age:* **integer**)

Suppose that the following indexes, all using Alternative (2) for data entries, exist: a hash index on *eid*, a B+ tree index on *sal*, a hash index on *age*, and a clustered B+ tree index on <*age, sal*>. Each Employees record is 100 bytes long, and you can assume that each index data entry is 20 bytes long. The Employees relation contains 10,000 pages.

1. Consider each of the following selection conditions and, assuming that the reduction factor (RF) for each term that matches an index is 0.1, compute the cost of the most selective access path for retrieving all Employees tuples that satisfy the condition:
   (a) *sal* > 100
   (b) *age* = 25
   (c) *age* > 20
   (d) *eid* = 1, 000
   (e) *sal* > 200 ∧ *age* > 30
   (f) *sal* > 200 ∧ *age* = 20
   (g) *sal* > 200 ∧ *title* ='CFO'
   (h) *sal* > 200 ∧ *age* > 30 ∧ *title* ='CFO'
2. Suppose that, for each of the preceding selection conditions, you want to retrieve the average salary of qualifying tuples. For each selection condition, describe the least expensive evaluation method and state its cost.
3. Suppose that, for each of the preceding selection conditions, you want to compute the average salary for each *age* group. For each selection condition, describe the least expensive evaluation method and state its cost.
4. Suppose that, for each of the preceding selection conditions, you want to compute the average age for each *sal* level (i.e., group by *sal*). For each selection condition, describe the least expensive evaluation method and state its cost.
5. For each of the following selection conditions, describe the best evaluation method:
   (a) *sal* > 200 ∨ *age* = 20
   (b) *sal* > 200 ∨ *title* ='CFO'
   (c) *title* ='CFO' ∧ *ename* ='Joe'

Exercise 15.3 For each of the following SQL queries, for each relation involved, list the attributes that must be examined to compute the answer. All queries refer to the following relations:

Emp(*eid:* **integer**, *did:* **integer**, *sal:* **integer**, *hobby:* **char(20)**)
Dept(*did:* **integer**, *dname:* **char(20)**, *floor:* **integer**, *budget:* **real**)

1. SELECT COUNT(*) FROM Emp E, Dept D WHERE E.did = D.did
2. SELECT MAX(E.sal) FROM Emp E, Dept D WHERE E.did = D.did
3. SELECT MAX(E.sal) FROM Emp E, Dept D WHERE E.did = D.did AND D.floor = 5
4. SELECT E.did, COUNT(*) FROM Emp E, Dept D WHERE E.did = D.did GROUP BY D.did
5. SELECT D.floor, AVG(D.budget) FROM Dept D GROUP BY D.floor HAVING COUNT(*) > 2
6. SELECT D.floor, AVG(D.budget) FROM Dept D GROUP BY D.floor ORDER BY D.floor

Exercise 15.4 You are given the following information:

Executives has attributes *ename*, *title*, *dname*, and *address*; all are string fields of the same length.
The *ename* attribute is a candidate key.
The relation contains 10,000 pages.
There are 10 buffer pages.

1. Consider the following query:

SELECT E.title, E.ename FROM Executives E WHERE E.title='CFO'

Assume that only 10% of Executives tuples meet the selection condition.
(a) Suppose that a clustered B+ tree index on *title* is (the only index) available. What is the cost of the best plan? (In this and subsequent questions, be sure to describe the plan you have in mind.)
(b) Suppose that an unclustered B+ tree index on *title* is (the only index) available. What is the cost of the best plan?
(c) Suppose that a clustered B+ tree index on *ename* is (the only index) available. What is the cost of the best plan?
(d) Suppose that a clustered B+ tree index on *address* is (the only index) available. What is the cost of the best plan?
(e) Suppose that a clustered B+ tree index on *<ename, title>* is (the only index) available. What is the cost of the best plan?

2. Suppose that the query is as follows:

SELECT E.ename FROM Executives E WHERE E.title='CFO' AND E.dname='Toy'

Assume that only 10% of Executives tuples meet the condition *E.title ='CFO'*, only 10% meet *E.dname ='Toy'*, and that only 5% meet both conditions.
(a) Suppose that a clustered B+ tree index on *title* is (the only index) available. What is the cost of the best plan?
(b) Suppose that a clustered B+ tree index on *dname* is (the only index) available. What is the cost of the best plan?
(c) Suppose that a clustered B+ tree index on *<title, dname>* is (the only index) available. What is the cost of the best plan?
(d) Suppose that a clustered B+ tree index on *<title, ename>* is (the only index) available. What is the cost of the best plan?
(e) Suppose that a clustered B+ tree index on *<dname, title, ename>* is (the only index) available. What is the cost of the best plan?
(f) Suppose that a clustered B+ tree index on *<ename, title, dname>* is (the only index) available. What is the cost of the best plan?

3. Suppose that the query is as follows:

SELECT E.title, COUNT(*) FROM Executives E GROUP BY E.title

(a) Suppose that a clustered B+ tree index on *title* is (the only index) available. What is the cost of the best plan?

(b) Suppose that an unclustered B+ tree index on *title* is (the only index) available. What is the cost of the best plan?
(c) Suppose that a clustered B+ tree index on *ename* is (the only index) available. What is the cost of the best plan?
(d) Suppose that a clustered B+ tree index on *<ename, title>* is (the only index) available. What is the cost of the best plan?
(e) Suppose that a clustered B+ tree index on *<title, ename>* is (the only index) available. What is the cost of the best plan?

4. Suppose that the query is as follows:

SELECT E.title, COUNT(*) FROM Executives E WHERE E.dname > 'W%' GROUP BY E.title

Assume that only 10% of Executives tuples meet the selection condition.
(a) Suppose that a clustered B+ tree index on *title* is (the only index) available. What is the cost of the best plan? If an additional index (on any search key you want) is available, would it help produce a better plan?
(b) Suppose that an unclustered B+ tree index on *title* is (the only index) available. What is the cost of the best plan?
(c) Suppose that a clustered B+ tree index on *dname* is (the only index) available. What is the cost of the best plan? If an additional index (on any search key you want) is available, would it help to produce a better plan?
(d) Suppose that a clustered B+ tree index on *<dname, title>* is (the only index) available. What is the cost of the best plan?
(e) Suppose that a clustered B+ tree index on *<title, dname>* is (the only index) available. What is the cost of the best plan?

Solutions

Answer 15.2 The answer to each question is given below.

1. For this problem, it will be assumed that each data page contains 20 tuples per page.

(a) *sal* > 100 For this condition, a filescan would probably be best, since a clustered index does not exist on *sal*. Using the unclustered index would accrue a cost of $10{,}000\ pages * \frac{20bytes}{100bytes} * 0.1$ for the B+ index scan plus $10{,}000\ pages * 20\ tuples\ per\ page * 0.1$ for the lookup = 22000, and would be inferior to the filescan cost of 10000.

(b) *age* = 25 The clustered B+ tree index would be the best option here, with a cost of $2\ (lookup) + 10000\ pages * 0.1\ (selectivity) + 10{,}000 * 0.2\ (\ reduction\ ) * 0.1 = 1202.$ Although the hash index has a lesser lookup time, the potential number of record lookups $(10000\ pages * 0.1 * 20\ tuples\ per\ page = 20000)$ renders the clustered index more efficient.

(c) *age* > 20 Again the clustered B+ tree index is the best of the options presented; the cost of this is $2\ (lookup) + 10000\ pages * 0.1\ (selectivity) + 200 = 1202.$

(d) *eid* = 1000 Since *eid* is a candidate key, one can assume that only one record will be in each bucket. Thus, the total cost is roughly 1.2 (lookup) + 1 (record access) which is 2 or 3.

(e) *sal* > 200 ∧ *age* > 30 This query is similar to the *age* > 20 case if the *age* > 30 clause is examined first. Then, the cost is again 1202.

(f) *sal* > 200 ∧ *age* = 20 Similar to the previous part, the cost for this case using the clustered B+ index on < *age,sal* > is smaller, since only 10 % of all relations fulfill *sal* > 200. Assuming a linear distribution of values for *sal* for *age*, one can assume a cost of $2\ (lookup) + 10000\ pages * 0.1\ (selectivity\ for\ age) * 0.1\ (selectivity\ for\ sal) + 10{,}000 * 0.4 * 0.1 * 0.1 = 142.$

(g) *sal* > 200 ∧ *title* = '*CFO*' In this case, the filescan is the best available method to use, with a cost of 10000.

(h) *sal* > 200 ∧ *age* > 30 ∧ *title* = '*CFO*' Here an age condition is present, so the clustered B+ tree index on < *age,sal* > can be used. Here, the cost is $2\ (lookup) + 10000\ pages * 0.1\ (selectivity) + 200 = 1202.$

2. (a) *sal* > 100 Since the desired result is only the average salary, an index-only scan can be performed using the unclustered B+ tree on *sal* for a cost of $2\ (lookup) + 10000 * 0.1 * 0.2\ (\ due\ to\ smaller\ index\ tuples\ ) = 202.$

(b) *age* = 25 For this case, the best option is to use the clustered index on < *age,sal* >, since it will avoid a relational lookup. The cost of this operation is $2\ (B + tree\ lookup) + 10000 * 0.1 * 0.4\ (due\ to\ smaller\ index\ tuple\ sizes) = 402.$

(c) *age* > 20 Similar to the *age* = 25 case, this will cost 402 using the clustered index.

(d) *eid* = 1000 Being a candidate key, only one relation matching this should exist. Thus, using the hash index again is the best option, for a cost of 1.2 (hash lookup) + 1 (relation retrieval) = 2.2.

(e) *sal* > 200 ∧ *age* > 30 Using the clustered B+ tree again as above is the best option, with a cost of 402.

(f) *sal* > 200 ∧ *age* = 20 Similarly to the *sal* > 200 ∧ *age* = 20 case in the previous problem, this selection should use the clustered B+ index for an index only scan, costing $2\ (B +$

$lookup) + 10000 * 0.1\ (selectivity\ for\ age) * 0.1\ (selectivity\ for\ sal) * 0.4\ (smaller\ tuple\ sizes, index - only\ scan) = 42.$

(g) *sal* > 200∧*title* = '*CFO*' In this case, an index-only scan may not be used, and individual relations must be retrieved from the data pages. The cheapest method available is a simple filescan, with a cost of 10000 I/Os.

(h) *sal* > 200 ∧ *age* > 30 ∧ *title* = '*CFO*' Since this query includes an age restriction, the clustered B+ index over < *age,sal* > can be used; however, the inclusion of the title field precludes an index-only query. Thus, the cost will be $2\ (B + tree\ lookup) + 10000 * 0.1\ (selectivity\ on\ age) + 10{,}000 * 0.1 * 0.4 = 1402\ I/Os.$

3. (a) *sal* > 100 The best method in terms of I/O cost requires usage of the clustered B+ index over < *age,sal* > in an index-only scan. Also, this assumes the ability to keep a running average for each age category. The total cost of this plan is $2\ (lookup\ on\ B + tree, find\ min\ entry) + 10000 * 0.4\ (index - only\ scan) = 4002$. Note that although *sal* is part of the key, since it is not a *prefix* of the key, the entire list of pages must be scanned.

(b) *age* = 25 Again, the best method is to use the clustered B+ index in an index-only scan. For this selection condition, this will cost $2\ (age\ lookup\ in\ B + tree) + 10000\ pages * 0.1\ (selectivity\ on\ age) * 0.4\ (index - only\ scan, smaller\ tuples, more\ per\ page, etc.) = 2 + 400 = 402.$

(c) *age* > 20 This selection uses the same method as the previous condition, the clustered B+ tree index over <*age,sal*> in an index-only scan, for a total cost of 402.

(d) *eid* = 1000 As in previous questions, *eid* is a candidate field, and as such should have only one match for each equality condition. Thus, the hash index over *eid* should be the most cost effective method for selecting over this condition, costing 1.2 (hash lookup) + 1 (relation retrieval) = 2.2.

(e) *sal* > 200 ∧ *age* > 30 This can be done with the clustered B+ index and an index-only scan over the < *age,sal* > fields. The total estimated cost is $2\ (B + lookup) + 10000\ pages * 0.1\ (selectivity\ on\ age) * 0.4\ (index - only\ scan) = 402.$

(f) *sal* > 200 ∧ *age* = 20 This is similar to the previous selection conditions, but even cheaper. Using the same index-only scan as before (the clustered B+ index over < *age,sal* >), the cost should be $2 + 10000 * 0.4 * 0.1\ (age\ selectivity) * 0.1\ (sal\ selectivity) = 42.$

(g) *sal* > 200∧ *title* = '*CFO*' Since the results must be grouped by age, a scan of the clustered < *age,sal* > index, getting each result from the relation pages, should be the cheapest. This should cost $2 + 10000 * .4 + 10000 * tuples\ per\ page * 0.1 + 5000 * 0.1\ (index\ scan\ cost) = 2 + 1000(4 + tuples\ per\ page)$. Assuming the previous number of tuples per page (20), the total cost would be 24002. Sorting the filescan alone, would cost 40000 I/Os. However, if the tuples per page is greater than 36, then sorting the filescan would be the best, with a cost of 40000 + 6000 (secondary scan, with the assumption that unneeded attributes of the relation have been discarded).

(h) *sal* > 200 ∧ *age* > 30 ∧ *title* = '*CFO*' Using the clustered B+ tree over < *age,sal* > would accrue a cost of $2 + 10000 * 0.1\ (selectivity\ of\ age) + 5000 * 0.1 = 1502$ lookups.

4. (a) *sal* > 100 The best operation involves an external merge sort over< *sal,age*>, discarding unimportant attributes, followed by a binary search to locate minimum *sal* < 100 and a scan of the remainder of the sort. This costs a total of $16000\ (sort) + 12\ (binary\ search) +$

$10000 * 0.4\,(smaller\ tuples) * 0.1\,(selectivity\ of\ sal) + 2 = 16000 + 4000 + 12 + 400 + 2 = 16414.$

(b) *age* = 25 The most cost effective technique here employs sorting the clustered B+ index over < *age,sal* >, as the grouping requires that the output be sorted. An external merge sort with 11 buffer pages would require 16000. In total, the cost equals $16000\,(sort) + 10000 * 0.4 = 20000.$

(c) *age* > 20 This selection criterion works similarly to the previous one, in that an external merge over < *age,sal* > is required, using the clustered index provided as the pages to sort. The final cost is the same, 20000.

(d) *eid* = 1000 Being a candidate key, only one relation should match with a given *eid* value. Thus, the estimated cost should be 1.2 (hash lookup) + 1 (relation retrieval).

(e) *sal* > 200 ∧ *age* > 30 This case is similar to the *sal* > 100 case above, cost = 16412.

(f) *sal* > 200 ∧ *age* = 20 Again, this case is also similar to the *sal* > 100 case, cost = 16412.

(g) *sal* > 200 ∧ *title* = '*CFO*' The solution to this case greatly depends of the number of tuples per page. Assuming a small number of tuples per page, the cheapest route is to use the B+ tree index over *sal*, getting each index. The total cost for this is $2\,(lookup,sal > 200) + 10000 * .2\,(smaller\ size) * .1\,(selectivity) + 10000 * .1\,(selectivity) * tuples\ per\ page$. The solution to this case is similar to that of the other requiring sorts, but at a higher cost. Since the sort can't be performed over the clustered B+ tree in this case, the sort costs 40000 I/Os. Thus, for the number of tuples per page equal to 40, the B+ index method is superior, otherwise, the sort solution is cheaper.

(h) *sal* > 200 ∧ *age* > 30 ∧ *title* = '*CFO*' This solution is the same as the previous, since either the index over *sal* or an external sort must be used. The cost is the cheaper of $2 + 1000 * (.2 + tuples\ per\ page)$ [index method] and 40000 [sort method].

5. (a) *sal* > 200 ∨ *age* = 20 In this case, a filescan would be the most cost effective, because the most cost effective method for satisfying *sal* > 200 alone is a filescan.

(b) *sal* > 200 ∨ *title* = '*CFO*' Again a filescan is the better alternative here, since no index at all exists for *title*.

(c) *title* = '*CFO*' ∧ *ename* = '*Joe*' Even though this condition is a conjunction, the filescan is still the best method, since no indexes exist on either *title* or *ename*.

Answer 15.3 The answer to each question is given below.

1. E.did, D.did
2. E.sal, E.did, D.did
3. E.sal, E.did, D.did, D.floor
4. E.did, D.did
5. D.floor, D.budget
6. D.floor, D.budget

Answer 15.4 1. (a) The best plan, a B+ tree search, would involve using the B+ tree to find the first *title* index such that *title*='CFO', cost = 2. Then, due to the clustering of the index, the relation pages can be scanned from that index's reference $cost = 10000 * 10\% + 2500 * 10\%\,(Scanning\ the\ index) = 1000 + 250 + 2 = 1252\,(total\ cost)$.

(b) An unclustered index would preclude the low cost of the previous plan and necessitate the choice of a simple filescan, cost = 10000, as the best.

(c) Due to the WHERE clause, the clustered B+ index on ename doesn't help at all. The best alternative is to use a filescan, cost = 10000.

(d) Again, as in the previous answer, the best choice is a filescan, cost = 10000.
(e) Although the order of the B+ index key makes the tree much less useful, the leaves can still be scanned in an index-only scan, and the increased number of tuples per page lowers the I/O cost. Cost = 10000 * .5 = 5000.

2. (a) A clustered index on *title* would allow scanning of only the 10% of the desired tuples. Thus the total cost is $2\ (lookup) + 10000 * 10\% + 2500 * 10\% = 1252$.
(b) A clustered index on *dname* works functionally in the same manner as that in the previous question, for a cost 1002 + 250 = 1252. The *ename* field still must be retrieved from the relation data pages.
(c) In this case, using the index lowers the cost of the query slightly, due to the greater selectivity of the combined query and to the search key taking advantage of it. The total cost = $2\ (look\ up) + 10000 * 5\% + 5000 * 5\% = 752$.
(d) Although this index does contain the output field, the *dname* still must be retrieved from the relational data pages, for a cost of $2\ (lookup) + 10000 * 10\% + 5000 * 10\% = 1502$.
(e) Since this index contains all three indexes needed for an index-only scan, the cost drops to $2\ (look\ up) + 10000 * 5\% * .75\ (smaller\ size) = 402$.
(f) Finally, in this case, the prefix cannot be matched with the equality information in the WHERE clause, and thus a scan would be the superior method of retrieval. However, as the clustered B+ tree's index contains all the indexes needed for the query and has a smaller tuple, scanning the leaves of the B+ tree is the best plan, costing $10000 * .75 = 7500$ I/Os.

3. (a) Since *title* is the only attribute required, an index-only scan could be performed, with a running counter. This would cost $10000 * .25\ (index - only\ scan, smaller\ tuples) = 2500$.
(b) Again, as the index contains the only attribute of import, an index-only scan could again be performed, for a cost of 2500.
(c) This index is useless for the given query, and thus requires a sorting of the file, costing $10000 + 2500 + 2 * 4 * 2500$ = 32500. First, we write out only the necessary attribute, requiring reading the file. Then we need 1+$\log_9$(2500/10) = 4 passes. Finally, a scan of this sorted result will allow us to answer the query, for a total cost of 35000.
(d) This is similar to the previous part, except that the initial scan requires fewer I/Os if the leaves of the B+ tree are scanned instead of the data file. $Cost = 5000 + 2500 + 2 * 4 * (2500) = 27500$. Finally, a scan of this sorted result will allow us to answer the query, for a cost of 30000.
(e) The clustered B+ index given contains all the information required to perform an index-only scan, at a cost of $10000 * .5\ (tuple\ size) = 5000$.

4. (a) Using a clustered B+ tree index on *title*, the cost of the given query is 10000 I/Os. The addition of another index would not lower the cost of any evaluation strategy that also utilizes the given index. However, the cost of the query is significantly cheaper if a clustered index on *dname, title* is available and is used by itself, and if added would reduce the cost of the best plan to 2250. (See below.)
(b) The cheapest plan here involves simply sorting the file. First we traverse the file and write out the matching records at a cost of at a cost of $10000 + 10000 * .25 * 10\% = 10250$ pages. Then, we sort the file for a cost of $2 * 3 * 250$ = 1500 pages. Lastly we need to read the sorted file for 250 pages. Total cost $= 12000$.
(c) The optimal plan with the indexes given involves scanning the *dname* index and sorting the (records consisting of the) *title* field of records that satisfy the WHERE condition. This

would cost $2500 * 10\%$ [scanning relevant portion of index] + $10000 * 10\%$ [retrieving qualifying records] + $10000 * 10\% * .25$ (reduction in size) [writing out *title* records] + $2 * 3 * 250$ [sorting *title* records would require $1+\log_9 25 = 3$ passes]. This is a total of 3000.

(d) We can simply scan the relevant portion of the index; discard tuples that don't satisfy the WHERE condition, and write out the *title* fields of qualifying records, in total 250 pages. The *title* records must then be sorted requiring $1+\log_9(250/10)=3$ passes. $Cost = 5000 * 10\% + 10000 * 10\% * .25 + 2 * 3 * 250 = 2250$.

(e) A clustered index on *title, dname* supports an index-only scan costing $10000 * .5 = 5000$.