Question : 15.6

Exercise 15.6 Briefly answer the following questions:

1. Explain the role of relational algebra equivalences in the System R optimizer.
2. Consider a relational algebra expression of the form $\sigma_c(\pi_l(R \times S))$. Suppose that the equivalent expression with selections and projections pushed as much as possible, taking into accollnt only relational algebra equivalences, is in one of the following forms. In each case give an illustrative example of the selection conditions and the projection lists (c, l, el, 11, etc.).
   (a) *Equivalent maximally pushed form:* $\pi_{l1}(\sigma_{c1}(R) \times S)$,
   (b) *Equivalent maximally pushed form:* $\pi_{l1}(\sigma_{c1}(R) \times U_c 2(S))$.
   (c) *Equivalent maximally pushed form:* $\sigma_c(\pi_{l1}(\pi_{l2}(R) \times 8))$.
   (d) *Equivalent maximally pushed fONT!:* $\sigma_{c1}(\pi_{l1}(\sigma_{c2}(\pi_{l2}(R)) \times 8))$.
   (e) *Equivalent ma:rimally pushed form:* $\sigma_{c1}(\pi_{l1}(\pi_{l2}(\sigma_{c2}(R)) \times S))$.
   (f) *Equivalent maximally pushed form:* $\pi_l(\sigma_c 1(\pi_{l1}(\pi_{l2}(\sigma_{c2}(R)) \times 8)))$.

Soln : 15.6

(a) *Equivalent maximally pushed form:* $\pi_{l1}(\sigma_{c1}(R) \times S)$.

(b) *Equivalent maximally pushed form:* $\pi_{l1}(\sigma_{c1}(R) \times \sigma_{c2}(S))$.

(c) *Equivalent maximally pushed form:* $\sigma_c(\pi_{l1}(\pi_{l2}(R) \times S))$.

(d) *Equivalent maximally pushed form:* $\sigma_{c1}(\pi_{l1}(\sigma_{c2}(\pi_{l2}(R)) \times S))$.

(e) *Equivalent maximally pushed form:* $\sigma_{c1}(\pi_{l1}(\pi c2(\sigma l2(R)) \times S))$.

(f) *Equivalent maximally pushed form:* $\pi_l(\sigma_{c1}(\pi_{l1}(\pi c2(\sigma l2(R)) \times S)))$.

1. Relational algebra equivalences are used to modify the query in hope of finding an optimal plan.

2. (a) $\sigma_{A=1}(\pi_{ABCD}(R \times S))$
$= \pi_{ABCD}(\sigma_{A=1}(R) \times S)$

(b) $\sigma_{A=1,C=2}(\pi_{ABCD}(R \times S))$
$= \pi_{ABCD}(\sigma_{A=1,C>2}(R \times S))$
$= \pi_{ABCD}(\sigma_{A=1}(R) \times \sigma_{C>2}(S))$

(c) $\sigma_{C=5}(\pi_{BC}(R \times S))$
$= \sigma_{C=5}(\pi_C(\pi_B(R) \times S))$

(d) $\sigma_{B=1,C=3}(\pi_{BC}(R \times S))$
$= \sigma_{B=1,C=3}(\pi_C(\pi_B(R) \times S))$
$= \sigma_{B=1}(\sigma_{C=3}(\pi_C(\pi_B(R) \times S)))$
$= \sigma_{B=1}(\pi_C(\sigma_{C=3}(\pi_B(R) \times S)))$

(e) $\sigma_{B=1,C=3}(\pi_{BC}(R \times S))$
$= \sigma_{B=1,C=3}(\pi_C(\pi_B(R) \times S))$
$= \sigma_{B=1}(\sigma_{C=3}(\pi_C(\pi_B(R) \times S)))$
$= \sigma_{C=3}(\pi_C(\pi_B(\sigma_{B=1}(R)) \times S))$

(f) $\sigma_{A=1,B=D}(\pi_{BC}(R \times S))$
$= \pi_{BC}(\sigma_{B=D}(\sigma_{A=1}(R) \times S)) = \pi_{BC}(\sigma_{B=D}(\pi_{BCD}(\sigma_{A=1}(R) \times S))$

Question 15.2

Exercise **15.2** Consider a relation with this schema:

Ernployees(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) $sol > 100$
   (b) $age = 25$
   (c) $age > 20$
   (d) $eid = 1,000$
   (e) $sal > 200 \wedge age > 30$
   (f) $sal > 200 \wedge age = 20$
   (g) $sal > 200 \wedge title = 'CFO'$
   (h) $sal > 200 \wedge age > 30 \wedge title = 'CFO'$
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 sa/level (Le.) 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) $sal > 200 \vee age = 20$
   (b) $sal > 200 \vee title = 'CFO'$
   (c) $title = 'CFO' \wedge ename = 'Joe'$

Soln:

1. For this problem, it will be assumed that each data page contains 20 relations 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 candiate 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) $sal > 200 \wedge age > 30$ This query is similar to the $age > 20$ case if the $age > 30$ clause is examined first. Then, the cost is again 1202.

(f) $sal > 200 \wedge age = 20$ 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) $sal > 200 \wedge title = "CFO"$ In this case, the filescan is the best available method to use, with a cost of 10000. $sal > 200 \wedge age > 30 \wedge title = "CFO"$ 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) = 1002.

(h) $sal > 200 \wedge age > 30 \wedge title = "CFO"$ Similar to the case of $age > 20$; the best access path is again the clustered B+ tree on *age, sal*.

2. (a) $sal > 100$ Since the result desired is only the average salary, an index-only scan can be performed using the unclusterd 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 candiate 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) $sal > 200 \wedge age > 30$ Using the clustered B+ tree again as above is the best option, with a cost of 402.

(f) $sal > 200 \wedge age = 20$ Similarly to the $sal > 200 \wedge age = 20$ 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) $sal > 200 \wedge title = "CFO"$ 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) $sal > 200 \wedge age > 30 \wedge title = "CFO"$ 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 ablility 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) $sal > 200 \wedge age > 30$ 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) $sal > 200 \wedge age = 20$ 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) $sal > 200 \wedge title = "CFO"$ 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) $sal > 200 \wedge age > 30 \wedge title = "CFO"$ Using the clustered B+ tree over $< age, sal >$ one 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. Totalled, 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) $eid = 1000$ Begin 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) $sal > 200 \wedge age > 30$ This case is similar to the $sal > 100$ case above, cost = 16412.

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

(g) $sal > 200 \wedge title = "CFO"$ 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 preformed over the clustered B+ tree in this case, the sort costs 40000 I/Os. Thus, for tuples per page ¡ 40, the B+ index method is superior, otherwise, the sort solution is cheaper.

(h) $sal > 200 \wedge age > 30 \wedge title = "CFO"$ 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) $sal > 200 \vee age = 20$ 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) $sal > 200 \vee title = "CFO"$ Again a filescan is the better alternative here, since no index at all exists for *title*.

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