Week 7 Exercises: Query Processing with Relational Operations

Exercise 14.1 Briefly answer the following questions:

1. Consider the three basic techniques, *iteration*, *indexing*, and *partitioning*, and the relational algebra operators *selection*, *projection*, and *join*. For each technique-operator pair, describe an algorithm based on the technique for evaluating the operator.
2. Define the term *most selective access path for a query*.
3. Describe *conjunctive normal form*, and explain why it is important in the context of relational query evaluation.
4. When does a general selection condition *match* an index? What is a *primary term* in a selection condition with respect to a given index?

Exercise 14.3 Consider processing the following SQL projection query:

SELECT DISTINCT E.title, E.ename FROM Executives E

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 tuples per page.
There are 10 buffer pages.

Consider the optimized version of the sorting-based projection algorithm: The initial sorting pass reads the input relation and creates sorted runs of tuples containing only attributes ename and title. Subsequent merging passes eliminate duplicates while merging the initial runs to obtain a single sorted result (as opposed to doing a separate pass to eliminate duplicates from a sorted result containing duplicates). The cost metric is the number of page I/Os unless otherwise noted, and the cost of writing out the result should be uniformly ignored.

1. How many sorted runs are produced in the first pass? What is the average length of these runs? What is the I/O cost of this sorting pass?
2. How many additional merge passes are required to compute the final result of the projection query? What is the I/O cost of these additional passes?
3. (a) Suppose that a clustered B+ tree index on *title* is available. Is this index likely to offer a cheaper alternative to sorting? Would your answer change if the index were unclustered?
   (b) Suppose that a clustered B+ tree index on ename is available. Is this index likely to offer a cheaper alternative to sorting? Would your answer change if the index were unclustered?
   (c) Suppose that a clustered B+ tree index on <ename, title> is available. Is this index likely to offer a cheaper alternative to sorting? Would your answer change if the index were unclustered?
4. Suppose that the query is as follows:

SELECT E.title, E.ename FROM Executives E

That is, you are not required to do duplicate elimination. How would your answers to the previous questions change?