# CS3223 Tutorial 4: Projection & Join **Wk 6, Sem 2, 2022/23**

**Questions to be discussed: 1, 2 & 3**.

1. (Adapted from Exercise 14.3, R&G) Consider processing the following SQL projection query:

   SELECT DISTINCT title, dname FROM Executives

   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.

   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 `dname` 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).

   In this question, the cost metric is the number of page I/Os.

   (a) 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?

   (b) 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?

   (c) (i) 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? Would your answer change if the index were a hash index?

   (ii) Suppose that a clustered $B^+$-tree index on `dname` is available. Is this index likely to offer a cheaper alternative to sorting? Would your answer change if the index were unclustered? Would your answer change if the index were a hash index?

   (iii) Suppose that a clustered $B^+$-tree index on `(dname, title)` is available. Is this index likely to offer a cheaper alternative to sorting? Would your answer change if the index were unclustered? Would your answer change if the index were a hash index?

   (d) Suppose that the query is as follows:

   SELECT title, dname FROM Executives

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

2. (Exercise 14.4 R&G) Consider the join $R \bowtie_{R.a=S.b} S$, given the following information about the relations to be joined. 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.

   - Relation R contains 10,000 tuples and has 10 tuples per page.
   - Relation S contains 2000 tuples and also has 10 tuples per page.
   - Attribute b of relation S is the primary key for S.
   - Both relations are stored as simple heap files.
   - Neither relation has any indexes built on it.
   - 52 buffer pages are available.

   (a) What is the cost of joining R and S using a **page-oriented simple nested loops join**? What is the minimum number of buffer pages required for this cost to remain unchanged?

   (b) What is the cost of joining R and S using a **block nested loops join**? What is the minimum number of buffer pages required for this cost to remain unchanged?

   (c) What would be the lowest possible I/O cost for joining R and S using **any join algorithm**, and how much buffer space would be needed to achieve this cost? Explain briefly.

   (d) How many tuples does the join of R and S produce, at most, and how many pages are required to store the result of the join back on disk?

   (e) Would your answers to any of the previous questions in this exercise change if you were told that R.a is a foreign key that refers to S.b?

3. (Exercise 14.5 R&G) Consider the join $R \bowtie_{R.a=S.b} S$, given the following information about the relations to be joined. 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.

   - Relation R contains 10,000 tuples and has 10 tuples per page.
   - Relation S contains 2000 tuples and also has 10 tuples per page.
   - Attribute b of relation S is the primary key for S.
   - Both relations are stored as simple heap files.
   - Neither relation has any indexes built on it.
   - 52 buffer pages are available.
   - Assume that all indexes are format 2, any B$^+$-tree index on $R$ has two levels of internal nodes, and any B$^+$-tree index on $S$ has one level of internal nodes.

   (a) With 52 buffer pages, if unclustered B$^+$-tree indexes existed on R.a and S.b,would either provide a cheaper alternative for performing the join (using an **index nested loops join**) than a **block nested loops join**? Explain.

      1. Would your answer change if only five buffer pages were available?
      2. Would your answer change if S contained only 10 tuples instead of 2000 tuples?

   (b) With 52 buffer pages, if clustered B$^+$-tree indexes existed on R.a and S.b, would either provide a cheaper alternative for performing the join (using the **index nested loops algorithm**) than a **block nested loops join**? Explain.

      1. Would your answer change if only five buffer pages were available?
      2. Would your answer change if S contained only 10 tuples instead of 2000 tuples?