Topic: Join Operator Implementation

Assigned reading: Section 14.4.1 from Raghu Ramakrishnan's book

For this note we will follow the notation from the assigned section.

Relation R has the size of M pages and relation S has the size of N pages. Also assume that M>N. We have B buffer pages available to carry out the join operation.

Join operation can be implemented as cross product followed by selection and projection. However, efficient implementations of join operator avoid enumeration of cross product. We will limit our discussion to only nested loop join with simple join condition involving single equality check (R.attribute1 = S.attribute2).

Nested loop join is similar to the nested "FOR" loops in the procedural programming paradigm. We take one page from the first or the outer relation and match it with every record from the second or the inner relation. We scan the outer relation only once. Depending on the number of available buffer pages, we might have to read the inner relation multiple times.

Consider the following three cases:

1. B > (M+N): In this case, we can simply read both R and S into the main memory. We will have at least one page to store temporary output. The total I/O cost is M+N pages.
2. B < (M+N+1), B > (N+1): In this case, we can read the smaller relation into the main memory. We will designate the smaller relation as the inner relation. We will scan the outer relation one page at-a-time and match it with all records of the smaller relation. We will keep at least one page reserved to store the temporary output. Total I/O cost is still M+N pages.
3. B<(N+2), B>2: In this case, we will keep one page reserved for temporary output and one page for the inner relation. We will read the outer relation in the block size of B-2 pages. We have to read the inner relation CEILING(M/(B-2)) times into the main memory. The total I/O cost is M + (N. CEILING(M/(B-2)))