ABSTRACT

Enhancing the performance of large database systems depends heavily on the cost of performing join operations. Optimizing the join operation of two very large tables is considered as one of the interesting research topics to many researchers, especially when both tables, to be joined, are very large to fit in the main memory. In such cases, join is usually performed using hashing or sorting technique. In hash based join, the database needs to hash and partition the build input, and with a sort-merge join, the database needs to sort the input before it can produce the join results. However, both hashing and sorting are considered as blocking operations since they block the progress of join operation until they are completed. In recent years there are a large number of join queries which are being executed by the interactive users and applications. In all the interactive applications the time to produce the first few results are very crucial. The state-of-art join algorithms are not ideal for this setting as most of the algorithms are hash/sort based algorithms, which require some pre-work before it can produce the join results. Hence, four new join algorithms are proposed in the present research, which can produce join results at higher rates during the early stages of the join operation. These proposed join algorithms are implemented without hashing and sorting techniques.

The proposed MRR join algorithm can be used to produce early and maximized join results during the earlier stages of the join operation. This is achieved by exploiting the distribution of the data in the join attribute column which is available in the histogram. The rows which will produce the
maximum join results are identified with the help of histogram and are joined during the earlier stages of the join operation. All the recent adaptive join algorithms that produce the result earlier use the concept of hashing and sorting. Even though they claim to produce the resultant tuples early, the hashing technique used in these algorithms induces a delay in the production of resultant tuples. This delay is due to the construction of hash table. Further, the hash table has to be accommodated in the memory which leads to memory overhead. The proposed MRR join algorithm does not require hashing or sorting. This reduces the memory and I/O overhead. It applies a concept of LCM to find the matching tuples instead of the Bit Array and Binary Search method used in existing algorithms. Thus the MRR join algorithm avoids sorting and hashing, which results in lower system resources and early result generation. In MRR join algorithm the join operation can be terminated when the required matching tuples are found. This can further reduce the number of comparisons and the delay in producing the join results, which in turn reduces the time and I/O overhead.

In the proposed Bucket Join algorithm the tuples are divided into buckets without the concept of hashing. The matched tuples and the tuples which do not produce the join results are eliminated during each phase of the join operation, thus reducing the number of comparisons required to produce the join results. Thus the bucket join algorithm can replace the other early join algorithms in any situation where a fast initial response time is required without any penalty in the memory usage and I/O operations.

The proposed path matrix join algorithm is used to increase the performance of the repeated join queries. Repetitive join queries are usually improved by using the join index or materialized views. Join index pre-computes the join and projects the ROW_ID’s of the joined tuples in the
join index. Join index once computed can be reused many times. Join query can also be improved by using materialized views. When a join query is executed, the results are immediately sent out as the pre-computed join result is already stored in the materialized view. Both join index and materialized view requires more CPU time and memory. To overcome the limitations of join index and materialized view a new join technique called Path Matrix join algorithm is proposed. The path matrix join algorithm uses a simple data structure called a Path Matrix, to record the location of the matching tuple for each distinct join attribute value. The proposed Path Matrix join algorithm uses a new Random Record join algorithm to find the matching rows and record the ROW_ID. The proposed path matrix join algorithm does not build intermediate join tables, but works directly on the table and builds the data structure as it performs the join operation. Path matrix join algorithm is designed to avoid complete relation scan and maximizes the rate of the join tuple generation.

The proposed Trace Back join algorithm is capable of producing the join results much earlier than the existing adaptive join algorithms. All the existing adaptive join algorithms concentrate only on the adaptive policy of the algorithm like flushing policy and reading policy, but fails to consider the join algorithm used in the adaptive join. This proposed method concentrates on reducing the number of comparisons required to produce the join query results. Here, the role of inner and outer table is continuously exchanged during the join operation. This type of join is more suitable for online shopping or order line applications where, similar types of orders will be placed together. In such cases, each tuples is likely to take the same join attribute value as the previous tuples.