The Bulk Index Join: A Generic Approach to Processing Non-Equijoins
(Extended Abstract)

Jochen van den Bercken[1], Bernhard Seeger[1] and Peter Widmayer[2]
[1]Fachbereich Mathematik und Informatik, Universität Marburg, D-35032 Marburg, Germany
[2]Institut für Theoretische Informatik, ETH-Zentrum, CH-8092 Zürich, Switzerland

Join operations are among the most expensive operations frequently occurring in a database system. During the last two decades, many approaches have been developed for processing join operations efficiently. Due to the dominance of relational systems the focus of most studies has been put on relational joins, in particular equijoins [1]. It is well recognized, however, that non-standard applications require an extensible database system which allows the definition of new data types (e. g. related to spatial entities) and new operations, in particular new join operations (e. g. spatial join, temporal join and band join [2]). Previous proposals of non-equijoins are limited to a special type of join, and a generalization of these proposals to other join types is not obvious. Therefore, such methods, although experimentally shown to be efficient, are not the prime choice for being integrated in an extensible database system.

We present a new algorithm called the bulk index join that can be applied to a broad class of non-equijoins including the types of non-equijoins mentioned above. Similar to the well-known index nested-loops join algorithm, the bulk index join probes the records of the outer relation against the inner relation by using a preexisting index structure. Like the index nested-loops join, our algorithm is generic in that any tree-based index structure supporting the join predicate can be used, including grow and post index trees [3]. Moreover, the designer of a new index structure might use our generic code to extend the functionality of the index structure without any additional effort. For example, GiST [4] would be an ideal candidate that can be merged with our generic join algorithm.

Though the index nested-loops join is a generic approach, it is known to be not very efficient. The reason is that index lookups are processed one by one so that the nodes of the index structure have to be read multiple times. For equijoins this problem can simply be avoided by sorting the outer relation and using a path buffer during probing. In general, sorting is however not applicable to non-equijoins.

In order to visit nodes at most once, index lookups are processed concurrently in our approach. Each of the non-leaf index nodes is associated with an external first-in-first-out buffer. First, all records of the outer relation are inserted as query entries into the buffer of the root node. Then, each of the query entries is read from the buffer and distributed among the buffers of those child nodes whose corresponding subtrees may contain records that fulfill the join predicate. This process continues in a top-down fashion until a query entry reaches a data node or the entire query set of a buffer is small enough to fit into main memory.

The band join [2] serves as an example in our experiments [5]. We found that the I/O cost is reduced by up to two orders of magnitude compared to the original index nested-loops join. The performance of our algorithm can additionally be improved by a factor of up to four by using large multi-page I/Os.

Experiments revealed that even in case of a preexisting index it is often advantageous to build a new index from scratch using our generic bulk loading technique [6]. The reasons are that the page size is tuned with respect to join efficiency and that better clustering is achieved. The gains from these effects outweigh the building cost of the new index.

Our experience indicates that the bulk index join is a generic algorithm that performs real fast, and that its implementation is simple enough to be of immediate practical value.

References

[1] G. Graefe: “Query Evaluation Techniques for Large Databases”, Computing Surveys 25(2): 73-170 (1993).

[2] D. J. DeWitt, J. F. Naughton, D. A. Schneider: “An Evaluation of Non-Equijoin Algorithms”, Proc. VLDB 1991: 443-452.

[3] D. B. Lomet: “Grow and Post Index Trees: Roles, Techniques and Future Potential”, Symp. on Large Spatial Databases 1991: 183-206.

[4] J. Hellerstein, J. Naughton A. Pfeffer, “Generalized Search Trees for Database Systems”, Proc. VLDB 1995: 562-573.

[5] J. van den Bercken, B. Seeger, P. Widmayer: “The Bulk Index Join: A Generic Approach to Processing Non-Equijoins”, Technical Report Nr. 14 (1998), Reihe Informatik, Fachbereich Mathematik und Informatik, Philipps-Universität Marburg.

[6] J. van den Bercken, B. Seeger, P. Widmayer: “A Generic Approach to Bulk Loading Multidimensional Index Structures”. Proc. VLDB 1997: 406-415.