An evaluation of packed memory arrays for columnar engines.

The Packed Memory Array (PMA) is an existing data structure to represent a dynamic set of elements according to a specified sorted order. As the elements are sorted, this allows to quickly identify the satisfying values in a range query, once the start and end interval of the range are found in the PMA. Furthermore, the elements are always stored contiguously in an array. Thus, scans become particularly efficient, as they feature sequential accesses to memory. From a theoretical point of view, an insertion or removal of an element in the data structure features a cost $O(\log_2^2(N))$ in the RAM model for the amortised worst case analysis, and $O(\frac{\log_2^2(N)}{B})$ in the I/O model. Whereas finding the successor/predecessor of a given element is $O(1)$ in the worst case.

In a DBMS environment, range queries has been traditionally solved with some variant of a B-Tree. Nevertheless, there are some key points on why PMAs could provide a competitive alternative. First, because data is contiguously stored in an array, we expect PMAs to provide faster access to its elements in a scan. Second, PMAs are more tractable from the perspective of a DBMS memory manager. Indeed, they avoid the problem of the segmentation of the memory space, whereas for a B-tree the system needs to keep track to the multiple allocations associated its nodes. Third, typically B-Trees can age quite badly, causing the nodes being spread randomly all over the memory space. Last, in a columnar engine, data is stored in columns, usually represented as some form of array over typed values. PMAs naturally fit this model. Of course, the drawback of a PMA is a higher cost in terms of updates.
In this work we analyse and evaluate the performance of the PMA in the context of columnar engines. We optimise the data structure for the cache-memory hierarchy. We investigate both the choice of a proper index to speed the search of a key and the alternative layouts that can be used to implement the PMA. We compare the data structure against regular AB-trees (memory B-Trees). Finally, as future work, we aim to integrate the data structure in MonetDB, and provide additional evaluations in a more realistic scenario.