Data Blocks: Hybrid OLTP and OLAP on Compressed Storage using both Vectorization and Compilation †

[EXTENDED ABSTRACT]

Harald Lang[1,‡], Tobias Mühlbauer[1], Florian Funke[2,*],
Peter Boncz[3,*], Thomas Neumann[1], Alfons Kemper[1]

[1]Technical University Munich, [2]Snowflake Computing, [3]Centrum Wiskunde & Informatica
{harald.lang,muehlbau,neumann,kemper}@in.tum.de, florian.funke@snowflake.net, boncz@cwi.nl

This work aims at reducing the main-memory footprint in high performance hybrid OLTP & OLAP databases, while retaining high query performance and transactional throughput. The basic assumptions for this work are (i) that the data set can be separated into hot (mostly written) and cold (mostly read) data, (ii) that OLTP mostly affects the hot data set but may also access (read/update) cold data. For this purpose an innovative compressed columnar storage format for cold data, called *Data Block*s is introduced. The problem of hot/cold clustering in general has been solved in the previous work of Funke et al. in [2], whereas this work presents the data structures to efficiently manage cold data.

To achieve highest OLTP performance the compression schemes of Data Blocks are very light-weight, such that OLTP transactions can still quickly access individual tuples. This sets our storage scheme apart from those used in specialized analytical databases, where data must usually be bit-unpacked. Data Blocks employ compression techniques like *ordered dictionary compression*, *truncation* and *single value compression*. In any case, the compressed representation of an attribute value remains byte-addressable. Typically, these values are represented as 1-, 2- or 4-byte integers. This also applies for dictionary encoded string-like attributes. Light-weight compression allows efficient predicate evaluation and cheap access to each individual tuple, as no other tuples are touched during decompression.

In contrast to the SAP HANA [1] system, which divides relations into a read- and a write-optimized partition to accelerate hybrid workloads, our HyPer system uses a different approach: Relations are divided into fixed-size chunks, e.g., 2^{16} tuples, which are individually compressed into read-optimized immutable Data Blocks when they are identified as cold. Freezing chunks individually into Data Blocks avoids costly merge phases of the read-optimized (compressed) and the write-optimized (uncompressed) partitions. Further, all existing index structures are unaffected as all tuples can still be directly accessed by their tuple identifiers.

To speed up scans on Data Blocks, we introduce a novel "positional" type of Small Materialized Aggregates [5] called *Positional SMA* (PSMA). PSMAs are small indexes that narrow the scan range within a block even if the block cannot be skipped based on materialized minimum and maximum values. Internally, PSMAs consist of a concise lookup table (typically 2 KB, 4 KB or 8 KB) that is computed when a cold chunk is "frozen" into a Data Block. The table entries contain scan ranges that point to the compressed data inside a Data Block with potential matching tuples. PSMAs are designed to support scan restrictions like $=, <, \leq, >, \geq$ and as well as `between` predicates.

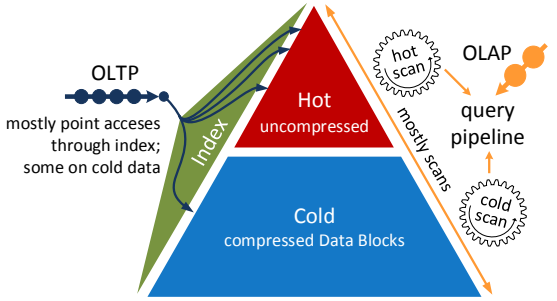

Figure 1: We propose the novel Data Block format that allows efficient scans and point accesses on compressed data and address the challenge of integrating multiple storage layout combinations in a compiling tuple-at-a-time query engine by using vectorization.

Compared to sub-byte encodings like BitWeaving [4], the compression schemes in Data Block naturally offer lower compression ratios. However, due to the chunked relation approach we can choose the optimal compression method based on the actual value distribution of an attribute within each chunk. This further reduces the memory footprint while remaining byte-addressable. Nevertheless, the resulting variety of physical data representations constitutes a challenge for JIT-compiling tuple-at-a-time query engines: Different storage layout combinations and extraction routines require either the generation of multiple code paths or to accept runtime overhead incurred by interpretation. So far high-performance analytical systems use either vectorized query execution or "just-in-time" (JIT) query compilation. The fine-grained adaptivity of Data Blocks necessitates the integration of the best features of each approach by an interpreted vectorized scan subsystem feeding into JIT-compiled query pipelines.

Our thorough experimental evaluation of Data Blocks integrated into HyPer [3], our full-fledged hybrid OLTP & OLAP database system, shows that Data Blocks accelerate performance on a variety of query workloads while retaining high transaction throughput. We further compare the performance of predicate evaluation and unpacking with a horizontally bit-packed storage and show that Data Blocks outperform bit-packing in almost all cases by factors.

†To appear at *SIGMOD 2016*
‡Speaker
*Work done while at Technical University Munich.

References

[1] F. Färber, S. K. Cha, J. Primsch, C. Bornhövd, S. Sigg, and W. Lehner. SAP HANA Database: Data Management for Modern Business Applications. *SIGMOD Record*, 40(4), 2011.

[2] F. Funke, A. Kemper, and T. Neumann. Compacting Transactional Data in Hybrid OLTP&OLAP Databases. *PVLDB*, 5(11), 2012.

[3] A. Kemper and T. Neumann. HyPer: A Hybrid OLTP&OLAP Main Memory Database System based on Virtual Memory Snapshots. In *ICDE*, 2011.

[4] Y. Li and J. M. Patel. BitWeaving: Fast Scans for Main Memory Data Processing. In *SIGMOD*, 2013.

[5] G. Moerkotte. Small Materialized Aggregates: A Light Weight Index Structure for Data Warehousing. In *VLDB*, 1998.