Efficient External Sorting in DuckDB

Laurens Kuiper
CWI
Amsterdam, Netherlands
laurens.kuiper@cwi.nl

ABSTRACT

It is not uncommon for database systems to have hundreds or even thousands of gigabytes of RAM at their disposal. High-performance systems such as HyPer [3], and ClickHouse [1] fully utilize the available memory and perform much better on analytical workloads than their traditional disk-based counterparts. Because these systems usually run on machines with such large memory capacities, the assumption is often that the workload fits in memory.

While laptops have also enjoyed increased memory capacity, their physical design has limited space. Therefore they typically have only 16GB of memory. Laptops are often used in interactive data analysis, with tools like Pandas [5] and dplyr [8], showing that there is a need for analytical data management technology that runs on a laptop. However, these tools operate only in memory. As a result, users cannot process datasets that are slightly larger than memory, on their own machine.

Disk-based database systems, on the other hand, have long solved the problem of processing larger-than-memory datasets. These systems are generally much slower than in-memory systems on analytical workloads. When a user wants to process a larger-than-memory dataset using an in-memory system, usually one of two things happens 1) The system throws an error stating it is out of memory, 2) The system switches to an external strategy that is much less efficient than the in-memory strategy, which results in a slow execution time, even when, for example, the input is only 10% larger than memory. Fast queries may become slow or run into an error when a table grows in size, creating a frustrating experience for users.

We can mitigate this problem by implementing operators such that they optimally use the amount of available memory and only write data to disk when this is necessary. I/O quickly becomes the bottleneck on machines with low-bandwidth storage devices. However, most modern laptops have nVME storage with high write speeds, making I/O less of a limiting factor.

We have implemented a parallel, external sorting operator in DuckDB [7] that demonstrates this. Our implementation seamlessly transitions from in-memory to external sorting by storing data in buffer-managed blocks that are offloaded to disk using a least-recently-used queue, similar to LeanStore [4].

Transitioning from memory to disk is made possible by DuckDB's buffer manager and unified internal row layout, shown in Figure 1, which can be spilled to disk using *pointer swizzling* [2]. When, swizzled, pointers are replaced by relative offsets, that can easily be restored, shown in Figure 2.

We compare our implementation against four other systems using an improvised relational sorting benchmark on two tables from TPC-DS [6]. Our implementation achieves excellent performance when data fits in memory and shows a graceful degradation in performance as we go over the limit of available memory.

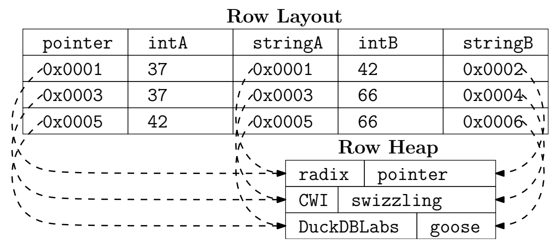

Figure 1: DuckDB's row layout and row heap.

Row Layout

offset	intA	stringA	intB	stringB
0	37	0	42	5
12	37	0	66	3
24	42	0	66	10

Row Heap

radix	pointer
CWI	swizzling
DuckDBLabs	goose

Figure 2: DuckDB's swizzled row layout and row heap.

REFERENCES

[1] Baktagul Imasheva, Azamat Nakispekov, Andrey Sidelkovskaya, and Ainur Sidelkovskiy. 2019. The Practice of Moving to Big Data on the Case of the NoSQL Database, ClickHouse. In *Optimization of Complex Systems: Theory, Models, Algorithms and Applications, WCGO 2019, World Congress on Global Optimization, Metz, France, 8-10 July, 2019 (Advances in Intelligent Systems and Computing)*, Hoai An Le Thi, Hoai Minh Le, and Tao Pham Dinh (Eds.), Vol. 991. Springer, 820–828. https://doi.org/10.1007/978-3-030-21803-4_82

[2] Alfons Kemper and Donald Kossmann. 1995. Adaptable Pointer Swizzling Strategies in Object Bases: Design, Realization, and Quantitative Analysis. *VLDB J.* 4, 3 (1995), 519–566. http://www.vldb.org/journal/VLDBJ4/P519.pdf

[3] Alfons Kemper and Thomas Neumann. 2011. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, Serge Abiteboul, Klemens Böhm, Christoph Koch, and Kian-Lee Tan (Eds.). IEEE Computer Society, 195–206. https://doi.org/10.1109/ICDE.2011.5767867

[4] Viktor Leis, Michael Haubenschild, Alfons Kemper, and Thomas Neumann. 2018. LeanStore: In-Memory Data Management beyond Main Memory. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*. IEEE Computer Society, 185–196. https://doi.org/10.1109/ICDE.2018.00026

[5] Wes McKinney et al. 2010. Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference*, Vol. 445. Austin, TX, 51–56.

[6] Meikel Poess, Bryan Smith, Lubor Kollar, and Paul Larson. 2002. TPC-DS, Taking Decision Support Benchmarking to the next Level. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data* (Madison, Wisconsin) *(SIGMOD '02)*. Association for Computing Machinery, New York, NY, USA, 582–587. https://doi.org/10.1145/564691.564759

[7] Mark Raasveldt and Hannes Mühleisen. 2019. DuckDB: An Embeddable Analytical Database. In *Proceedings of the 2019 International Conference on Management of Data* (Amsterdam, Netherlands) *(SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 1981–1984. https://doi.org/10.1145/3299869.3320212

[8] Hadley Wickham, Romain François, Lionel Henry, and Kirill Müller. 2021. *dplyr: A Grammar of Data Manipulation.* https://CRAN.R-project.org/package=dplyr R package version 1.0.7.