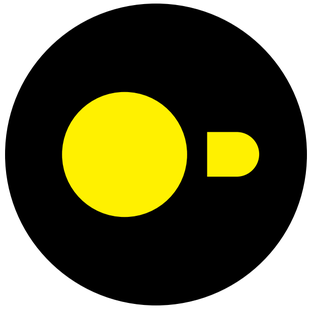


Hands-on: a PhD Centered Around DuckDB

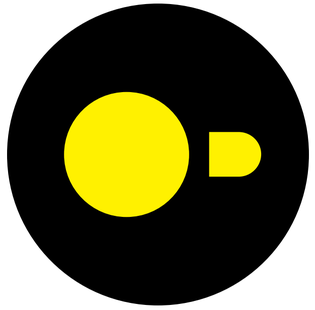
Laurens Kuiper (ex-CWI, now at DuckDB Labs Amsterdam)

DuckDB

Overview

1. About Me
2. DuckDB: a Real (Research) System
3. The Story of Sorting in DuckDB
4. Conclusion

1. About Me



Computer Science Career

- 2014 – 2017: CS Bachelor in Nijmegen (some SQL)
- 2017 – 2020: Data Science Master in Nijmegen (no DB stuff)
- 2020 – 2024: PhD at CWI in Amsterdam (only DB stuff)
- 2021 – Now: Software Developer at DuckDB Labs Amsterdam

DuckDB Contributions

- Core:
  - Larger-than-memory query processing:
    - Sorting
    - Hash Aggregation
    - Hash Join
    - Memory management

- Extensions:
  - fts
  - json

2. DuckDB: a Real (Research) System

Database Systems Research

- During my time in university, I:
  - Worked with existing code ❌
  - Started with a clean slate ✅

- Lots of database systems research is done in the same way

- Advantage: unrestricted innovation 🚀
  - Not constrained by design choices made prior

- Disadvantage: unrestricted innovation 💥
  - Not constrained by what is realistic in a system

Which System?

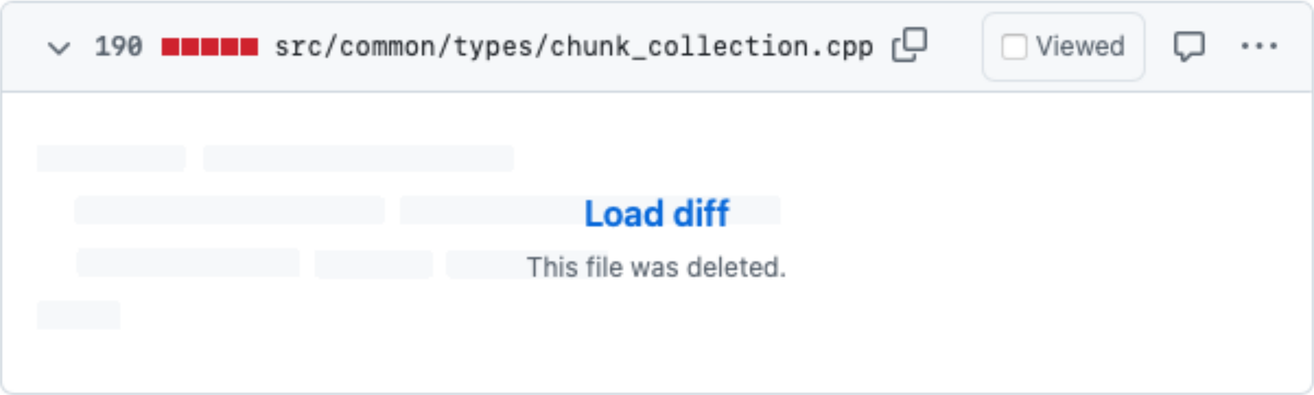
- If you're a researcher in 2015 who wants to work with existing code, where do you start?
  - Only a select few universities have a good in-house system
  - Open-source options are all but modern:
    - OLAP: MonetDB
    - OLTP: PostgreSQL

- Since 2019:
  - OLAP: DuckDB
  - OLTP: ???

3. The Story of Sorting in DuckDB

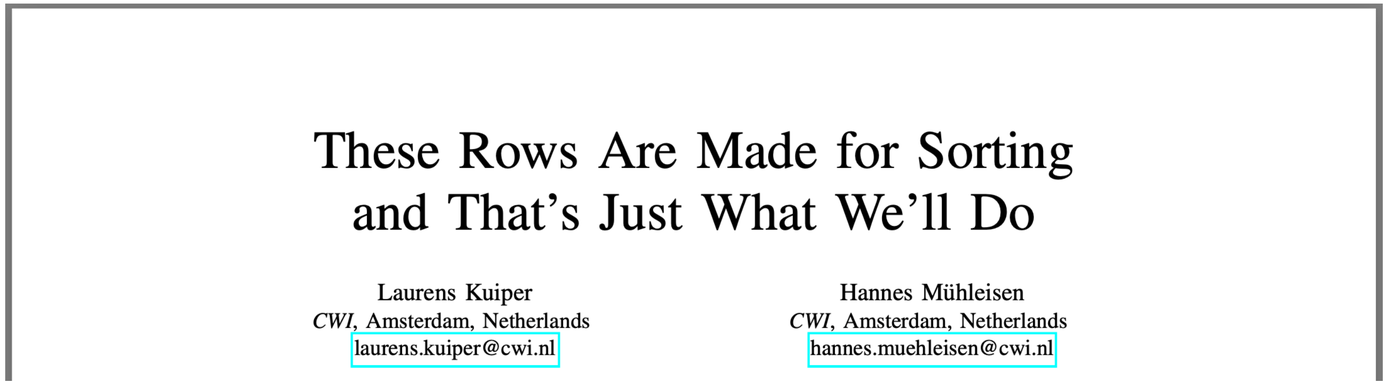
Humble Beginnings

- Implementation #1:
  - Very inefficient comparisons (bad cache locality, lots of branches)
  - Single-threaded
  - In-memory only
  - Used this data structure:



Second Chance

- Implementation #2 (ICDE ’23):
  - Efficient comparisons (good cache locality, few branches)
  - Fully parallel
  - Handles larger-than-memory data


These Rows Are Made for Sorting
and That’s Just What We’ll Do

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

Hannes Mühleisen
CWI, Amsterdam, Netherlands
hannes.muehleisen@cwi.nl

- Many problems with larger-than-memory sorting
- Horrible API

Second Chance

- Why didn't we find these problems earlier?
  - Implemented with one research objective in mind: performance
- How did these problems get found?
  - My code was being used after the research was done
- Problems with larger-than-memory sorting were found by users
- API problems encountered by Richard:
  - When integrating the sort into Window Operator and Range Joins

Third Time's the Charm

- Implementation #3:
  - Takes lessons from *research* and *practical experience*
  - Claims to tackle *all of the downsides* of implementation #2

New Sorting Implementation #17584

Merged **Mytherin** merged 77 commits into `duckdb:main` from `lnkuiper:sorting` on May 28

Conversation 3 | Commits 77 | Checks 52 | Files changed 61

lnkuiper commented on May 21 — Member

This PR proudly presents a full rewrite of DuckDB's sorting code. It is currently integrated into the `ORDER BY` operator, and should be integrated into other operators over a series of PRs, such that the current sort code can be removed from the code base at some point.

Current Implementation

DuckDB's current sorting implementation was written a few years ago by an inexperienced 1st year PhD student (me).

...

New Implementation

The new sorting code is written by a much more experienced software developer (me again).
It tackles all of the downsides of the current implementation:

...

Third Time's the Charm

- Implementation #3:
  - Improved parallel scaling
  - Less I/O for larger-than-memory processing
  - Highly adaptive to pre-sorted data
  - In summary: better performance

  - Same API as DuckDB's query operators:
    - Sink → Combine → Finalize → GetData
    - Easier integration in other operators (already in Window!)

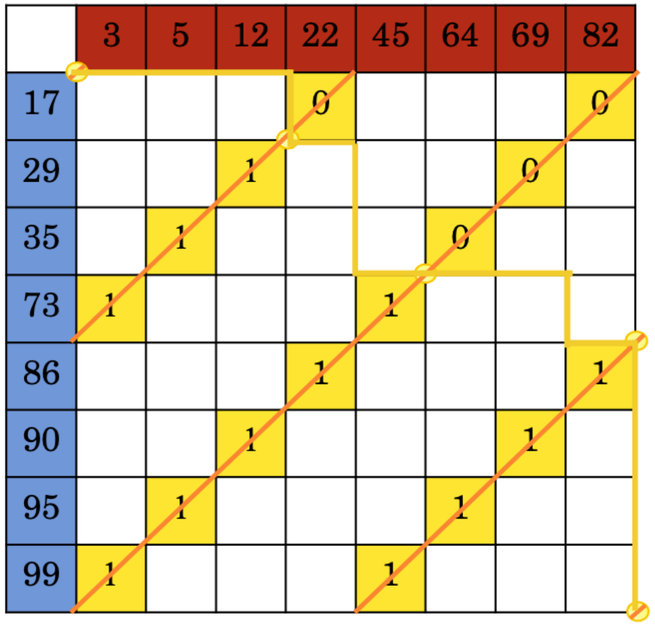
Third Time's the Charm

- Implementation #3:
  - **Improved parallel scaling**
  - Less I/O for larger-than-memory processing
  - Highly adaptive to pre-sorted data
  - In summary: better performance

  - Same API as DuckDB's query operators:
    - Sink → Combine → Finalize → GetData
    - Easier integration in other operators (already in Window!)

Third Time's the Charm

- Novel merge sort implementation:
  - Streaming
  - K-way
  - Parallel
  - External



- *Merge Path* by Oded Green et al.
  - Precompute where sorted runs *intersect* (boundaries)
  - Merge ranges between boundaries independently, in parallel
  - Generalized to K sorted runs in DuckDB v1.4.0

Third Time's the Charm

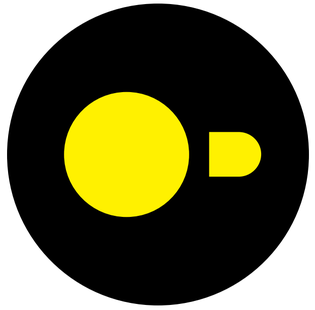
- M1 Max (10 threads, 30 GB memory limit)

```
SELECT *
FROM lineitem
ORDER BY l_shipdate;
```

- v1.4.0 Performance preview:

SF	v1.3.2 [s]	v1.4.0 [s]	Speedup [x]
1	0.328	**0.189**	1,735
10	3,353	**1,520**	2,205
100	273,982	**80,919**	3,385

4. Conclusion



Summary

- Database systems research:
  - Has more credibility when implemented in a real system
  - Can now use a modern open-source OLAP system: DuckDB

  - Becomes better the more it is used in practice:
    - Maintaining DuckDB's sort implementation made it more robust
    - Usage identified pain points that were missed in the paper

- Combination of research and implementation in a real system produced a better sorting implementation than either could have produced alone