Design and Implementation of DuckDB Internals

Welcome & Setup

April 7, 2026

Torsten Grust
Universität Tübingen, Germany

1 Welcome!

Welcome to this course which is all about digging deep into the internals of ▦ **tabular database management systems** (DBMSs).

Our tour through the DBMS kernel will touch on

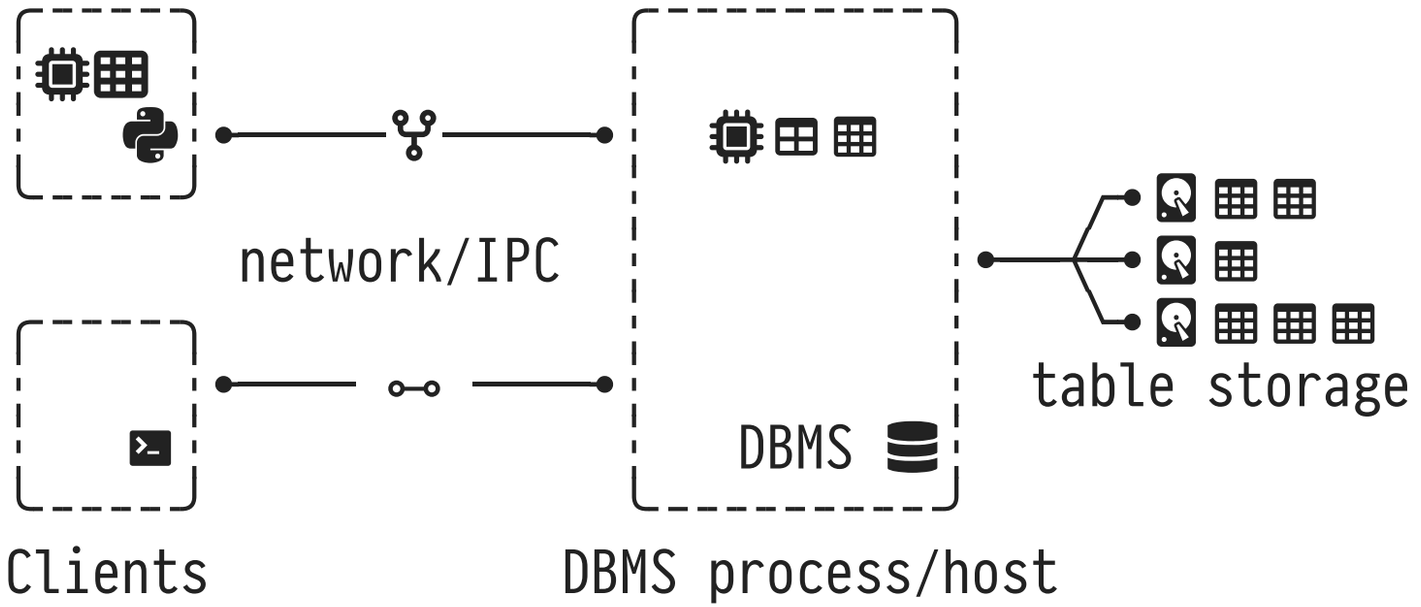
- the efficient representation of data on **secondary** (SSD/HDD 💾) and **primary storage** (RAM 🔲),
- turning declarative SQL queries into efficient **flows of data,**
- a variety of interesting **data structures** 🌳 **for sizable volumes** and associated **algorithms,**
- **modern CPUs** 🔲 and how looping/branching code ↻🔀 executes,
- various forms of **parallelism** 📚 on different levels (from single CPU instructions to threads), or
- ensuring **data integrity** under concurrent access or even if the host machine fails 🔌💥.

Dissecting the Duck's Innards

This course will focus on **DuckDB** , a contemporary tabular DBMS built for high-speed SQL-based data analytics.

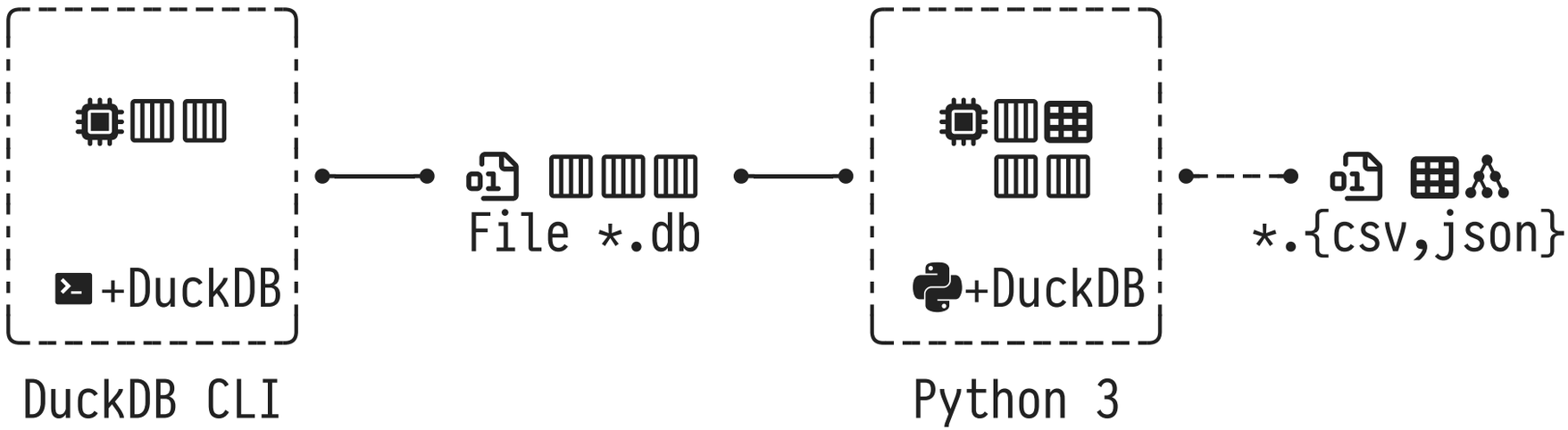
- DuckDB is young, first released in June 2019 (while most DBMSs—like PostgreSQL—originate from the 1980s).
- DuckDB development is moving fast. The system both adopts newest research results and builds on established DB wisdom.
- DuckDB is developed in the open (, MIT license). All code is available for inspection and tinkering.
- DuckDB provides a range of hooks to observe its internals.
- DuckDB comes with a CLI and programming language APIs.
- DuckDB is easy to install and requires no maintenance.

Not DuckDB: Dedicated DBMS Process/Host Isolated From Users



- DBMS controls family of disks holding table data, maps relevant table fragments into RAM buffer.
  - On-disk data organized in directories or in raw blocks.
- Client processes and DBMS process isolated, connected via network or inter-process communication.
  - Data needs to be de-/serialized after/before wire transfer.
- Data structures in clients are inaccessible by the DBMS.
- DBMS archetypes: PostgreSQL, MySQL, SQL Server™, Oracle®.

DuckDB: A Tabular DBMS Inside Your Own Process



- DuckDB kernel and client share a **single process** ⬚.
  - Python: `import duckdb`, C/C++: link with `libduckdb`.
- Table data resides in a **single database file** *.db, native DuckDB data formats in file and in RAM are similar.
- DuckDB sees in-process client data and can **directly read/write client data structures** using SQL (“**zero copy**”, replacement scans).
- DuckDB allocates sizable RAM buffers (but can use disk for temporary storage if required).

2 This Course (*Dissecting the Duck's Innards*, short: *DiDi*)

- We will focus on DuckDB as a **tabular** SQL-based DBMS ▦. There are other kinds of DBMSs (for graphs, key/value pairs, vectors, …), but we will not discuss those here.
- We will get our hands dirty using **DuckDB** and its extensive **SQL** dialect. Lots of SQL will be read and written.
- Whenever possible we try to observe DuckDB under load or use hooks to inspect its operation while SQL queries are processed.
- We thus assume basic familiarity with the tabular data model and SQL, e.g., as discussed in *Tabular Database Systems* (*TaDa*).
- We will draw data and queries from a variety of sources and have **fun** along the way! 😎

Torsten Grust?

Time Frame	Affiliation/Position
1989–1994	Diploma in Computer Science, TU Clausthal
1994–1999	Promotion (PhD), U Konstanz
2000	*Visiting Researcher*, IBM (USA)
2000–2004	Habilitation, U Konstanz
2004–2005	Professor Database Systems, TU Clausthal
2005–2008	Professor Database Systems, TU München
since 2008	Professor Database Systems, U Tübingen

- Web: https://db.cs.uni-tuebingen.de/grust
- Bluesky 🦋: @teggy.org
- E-mail: torsten.grust@uni-tuebingen.de

- Feel free to reach out with criticism, bug reports, suggestions for improvement, pats on the back, or simply to say “*Hi!*” 👋

Slides and Further Lecture Material

These **slides** (PDF), **code fragments** (SQL, Python, C), and **sample data** will be uploaded to a GitHub repository:

github.com/DBatUTuebingen/DiDi

- Slides point to relevant code files or extra material using tags like `#001`:
  - Refers to a file named `001-*` on the GitHub repository (e.g., `001-sum-quantity.awk`).

- **NB.** Code and extra material provide essential content (e.g., details on SQL-based experiments or sample data).
  - + = **:** Only slides + code provide a complete picture.

Material

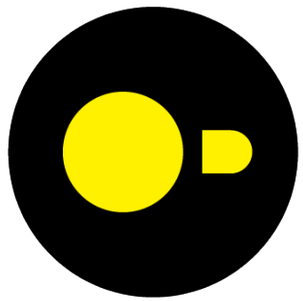
This course is *not* based on a single textbook. Rather, we build on

- a variety of scientific papers,
- textbook excerpts (few),
- the DuckDB documentation at https://duckdb.org/docs/,
- Python/C/C++ code snippets (our own and from inside the),
- blog posts from a range of authors,
- SQL references/standards,
- experience, and best practices.

There is a plethora of books on tabular DBMSs (both usage and internals), sample SQL snippets (experiments, benchmarks, and idioms), or performance tweaks. If we will use such sources, we will provide pointers.

Get Your Hands Dirty: Install DuckDB!

The tabular DBMS **DuckDB** will be the primary tool in this course:



DuckDB

https://duckdb.org, version 1.5 (March 2026: 1.5.0)

- Implements an extensive SQL dialect, is highly performant, open to contributions, and generally awesome.
- Straightforward to install and use on macOS , Windows , Linux (x86 + ARM).

  No DuckDB CLI (SQL prompt/REPL) on iOS or Android.[2]

[2] Run the DuckDB CLI in the web browser: https://shell.duckdb.org. Suffices for quick SQL experiments.

3 ¦ *Duck*DB? 🦆?

In case you were wondering:

DuckDB has been named after *Wilbur, the Duck*, which has been living as a pet with Hannes Mühleisen[3]—co-inventor of DuckDB with Mark Raasveldt—on Hannes' houseboat in Amsterdam.

Hannes (CEO) and Mark (CTO) run DuckDB Labs, a company that provides support and consultancy services around DuckDB. The labs are located in Amsterdam, The Netherlands.

Hannes and Wilbur (© Hannes Mühleisen)

[3] Hannes originally is from the Stuttgart area. Back then his car had the license plate D S QL 1337.