

Ducks to the rescue

ETL using Python and DuckDB

PyCon Italia 2026 – 30.05.2026

Marc-André Lemburg :: eGenix.com GmbH

Speaker Introduction

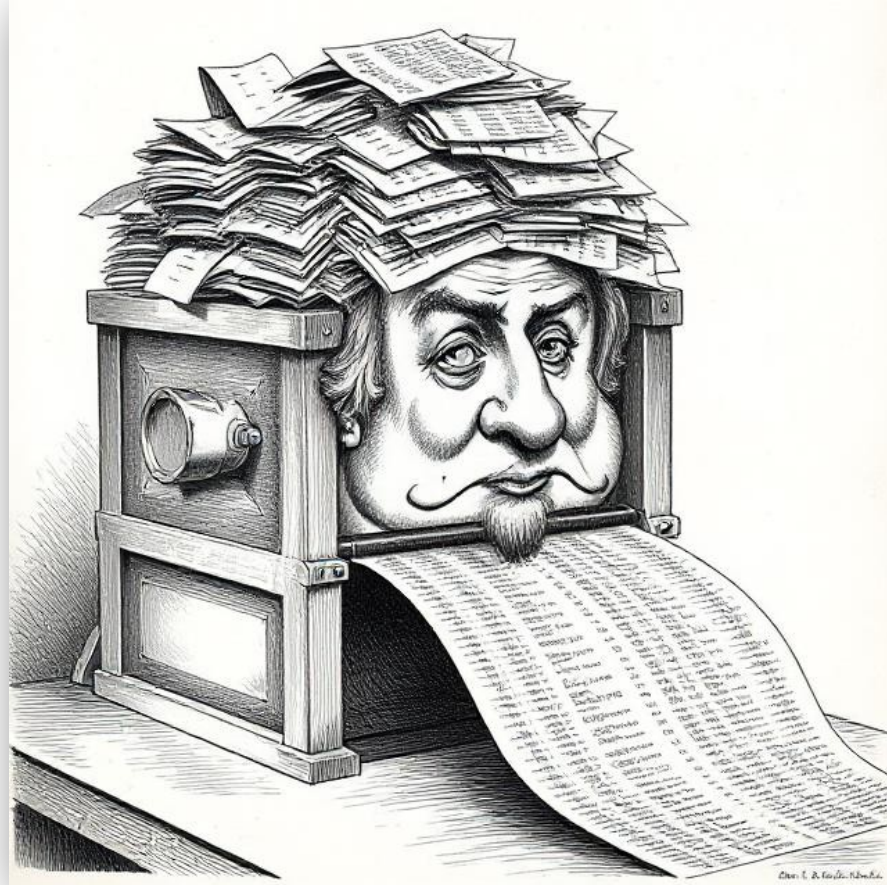
Marc-André Lemburg

- Studied Mathematics, Computer Science and Physics
- CEO eGenix.com GmbH
- **Senior Solution Architect, Consulting CTO and Coach**
- **EuroPython Society Fellow** and former Chair
- **Python Software Foundation Fellow** and former director
- **Python Core Developer** (Unicode, DB-API, platform module, ...)
- **Co-founder Python Meeting Düsseldorf**
- Based in Düsseldorf, Germany

- More details and contact: <https://malemburg.com/>

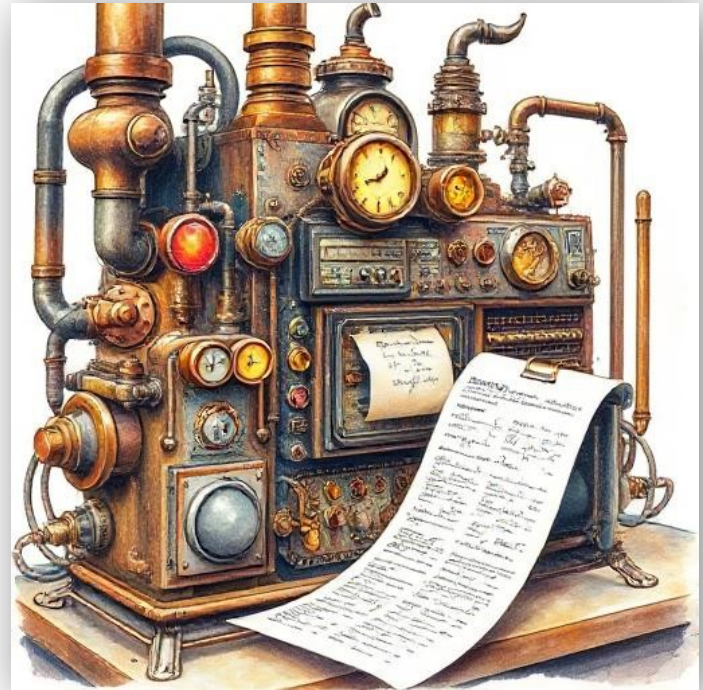


Motivation



ETL – Extract, Transform, Load

- Move data from a source to a target, transforming it along the way
 - In business, a synonym for “[data conversion](#)”
- **Extract**
 - Pull data from a data source and make it available to the ETL process
- **Transform**
 - Take the data and adapt it for loading
- **Load**
 - Load the data into the target database
- **Process**
 - Run your application against the target database



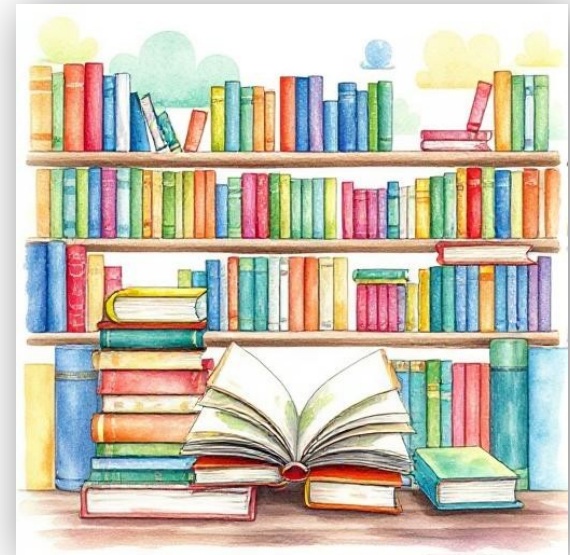
What's DuckDB ?

- Open source, in-process analytics data storage (OLAP)
- Column based (Apache Arrow)
- Persistent and in-memory
- SQL as standard query language
- Single writer / multiple readers
- *Similar to SQLite, but for OLAP workloads*
- Great Python support



Typical ETL steps (generic)

- **Read the data**
 - Download data from FTP, an object store, or the web
 - Fetch an RSS feed and read the listed web resources
 - Read a CSV or Excel file
 - Read tables from a webpage or PDF
 - Talk to a database and query data or run reports
 - Get data from an email box
 - Listen on a stream input for data, e.g. MQTT, Kafka queues, etc.
- **Save data** locally or in an object store



Typical ETL steps (generic)

- **Massage the data for easier loading**
 - Adjust encoding (to UTF-8)
 - Remove whitespace
 - Remove extra reporting lines
 - Trim down the data to what you are really interested in
 - Extract meta data
 - Reformat input data into CSV or Parquet files
- Should be done using a **streaming approach** to reduce memory load
 - A line based approach is typically good enough
 - Polars streaming can help with this



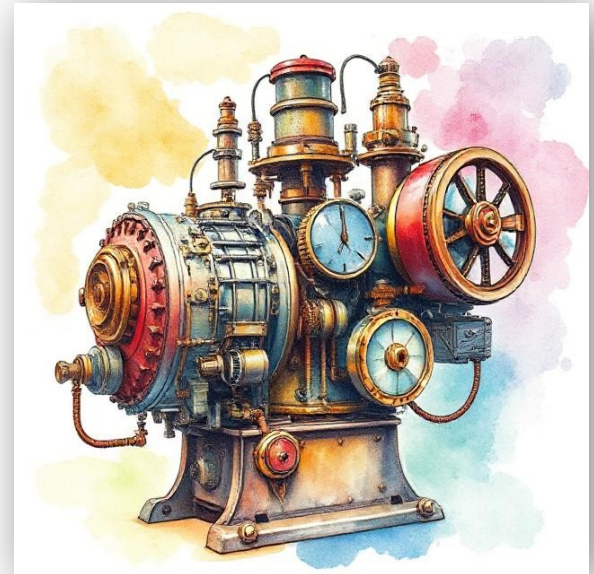
Typical ETL steps (using DuckDB)

- **Load data into DuckDB**
 - Use very fast DuckDB tooling for this
 - `SELECT * FROM read_csv(...)`
 - Avoid reading data into Python first
 - takes much longer
 - uses more memory
- **Load data into staging tables**
 - or even a separate database



Typical ETL steps (using DuckDB)

- **Transform data in DuckDB**
 - Filter out unneeded and duplicate data
 - Validate data
 - Fill in missing data (NULLs and N/A values)
 - Correct data/time formatting
 - Adjust int/float data to fit the target database
 - Convert data types where needed
 - Rename columns to match target database
- **Use DuckDB SQL for this**
 - DuckDB uses the PostgreSQL dialect, with extensions
 - **Hint: LLMs are great at writing SQL for you**



Typical ETL steps (using DuckDB)

- **Merge data into target database**
 - Use DuckDB SQL for preparing / processing the merge
 - Fall back to Python if SQL gives you problems
 - Polars can help with this, since it is integrated with DuckDB
 - **Use native Duck integrations** for writing to targets
 - PostgreSQL
 - MySQL
 - SQLite
 - Iceberg and Delta Lakesor stream via Python using the DB API
- **Remove or clear the staging tables again**
 - or move the data to archive tables



Advanced ETL

Guidelines

- **Know your queries:**
You should have good knowledge of how the target data is going to be used
- **Use the Pareto principle (80/20 rule):**
Optimize for often used queries
- **Keep a healthy balance:**
Performance and space requirements are often trade-offs



Advanced ETL

- **Optimize the data for querying** in your target database
 - Goal: avoid query time joins
 - **Move joins to insert time**
 - Denormalize your data
 - It's ok to keep multiple copies around
 - Just make sure data is updated as part of the ETL process
 - **Use materialized views**
 - Update materialized views at low query times
 - (DuckDB does not support these at the moment)



Advanced ETL

- **Add / update indexes carefully**
 - Only add them based on query statistics
 - Index updates should be deferred until after the loading has finished
- **Partition the data**
 - This is only needed for really large datasets
 - Can be done based on index columns
- **Deduplicate data**
 - Sometimes needed to remove duplicate records
 - Often needed when dealing with streamed data, e.g. IoT data, sensor data, mobile data, etc.



Advanced ETL

- **Prefilter data**
 - Your queries may not need all the data
 - Leave the full set in DuckDB
- **Preaggregate data**
 - Often enough you only need aggregates over certain windows
 - Leave the full set in DuckDB
- **Precompute data sketches**
 - Data sketches produce results which are only accurate to a certain percentage
 - They run a lot faster than full queries



Demo... ETL with DuckDB

- **Task:**
Load **German weather data** (DWD) into PostgreSQL
 - Daily measurements for all measurement points
 - Some going back to 1781
 - 2.4GB raw CSV data
- **Extract:**
Read data from DWD into DuckDB
 - 435MB on disk
- **Transform:**
Produce monthly aggregates
- **Load:**
Write to a DuckLake

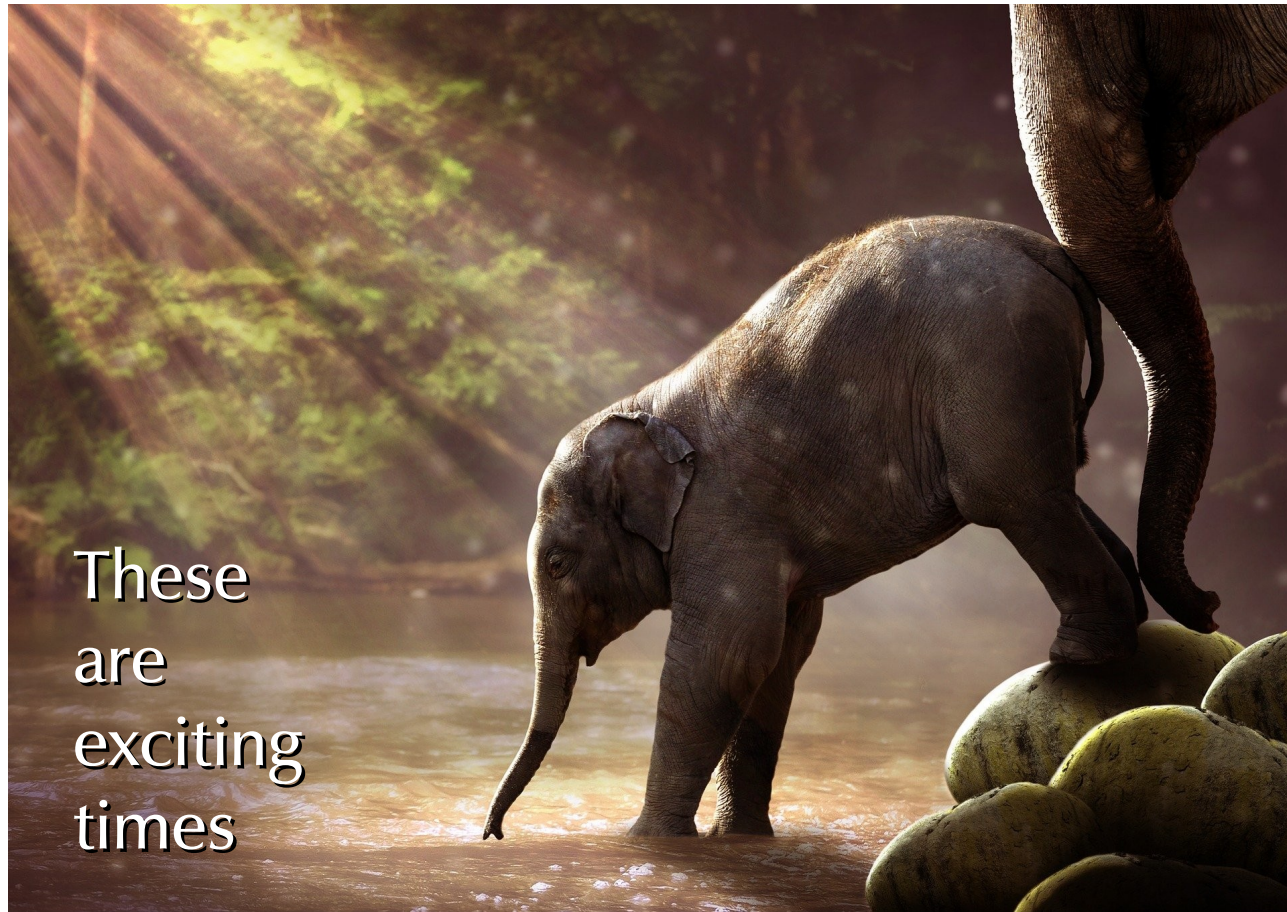


Conclusion

- **DuckDB makes ETL a whole lot easier**
 - Most ETL steps can be handled in a single process
 - No need for complex workflow setups using e.g. Airflow
 - Fast
 - Easy to debug
 - Work directly on the data
- Just a “`uv add duckdb`” away...



Main takeaway: Never stop **learning** and **trying out new things**



Thank you for your attention !



Time for discussion

Contact

eGenix.com Software, Skills and Services GmbH

Marc-André Lemburg

Pastor-Löh-Str. 48

D-40764 Langenfeld

Germany

eMail: mal@egenix.com

Phone: +49 211 9304112

Fax: +49 211 3005250

Web: <https://www.egenix.com/>

LinkedIn:



References

- Several photos taken from Pixabay and Unsplash
- Many images were generated using DeepAI text2img
- Some screenshots taken from the mentioned websites
- All other graphics and photos are (c) eGenix.com or used with permission
- Details are available on request
- Logos are trademarks of their respective trademark holders