

What The Heck Is DuckDB?

July 27th 2022 | 6 min | by @ProgRockRec ★ 2,289 reads

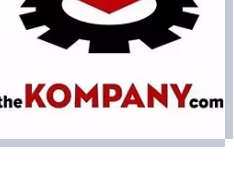


#duckdb #olap #column-database #sqlite #amazon-redshift #database #database-design #sql-database

1x

Read by Dr. One (en-US)

Audio Presented by Kion



Introduction

I like to spend time on LinkedIn reading through posts from companies about news, new releases, funding, new products, etc.. During a recent perusal, I saw something about "DuckDB", and as much as I pay attention to the data space, I hadn't heard of this one, so I thought I'd investigate and answer the question "What the heck is DuckDB?".

Overview

DuckDB itself is an MIT-licensed open source project whose code is hosted on [GitHub](#). DuckDB Labs is a commercial company formed by the creators of DuckDB in July 2021 to provide support, custom extensions, and even custom versions of the product as a way to monetize it. This model reminds me of the early days of open source monetization and is one I like.

DuckDB is briefly described as SQLite for analytic workloads. While SQLite is an embeddable, row-based, and b-tree indexed data store well suited for OLTP workloads, DuckDB is an embeddable column-based data store that uses vectorized processing to optimize OLAP workloads, you could also think of it as an embedded [Amazon Redshift](#) or a mutant offspring of SQLite and Redshift. Some of the features are:

- Simple installation
- Single-File storage format
- No server
- Fast processing
- Language library integrations
- Not reliant on any external config files or settings
- Programmatic SQL API
- Fully ACID compliant
- WASM (web assembly) version available

OLTP	OLAP
For your operation workloads	For your analytic workloads
Shorter queries	Longer queries for complex questions
Tables are more highly normalized	Tables are de-normalized
Typically implemented as row-oriented data stores	Typically implemented as column-oriented data stores

Testing

I decided to use their very clever WASM [web-based shell](#) to try out querying some Parquet files, of which I grabbed some to play with from [here](#). I started with the ".files add" command to load up the parquet file:

```
DuckDB Web Shell
Database: v0.4.0
Package: duckdb/duckdb-wasm@0.1.16.1-dev12.0
Connected to a local transient in-memory database.
Enter .help for usage hints.

duckdb> .help

Commands:
.clear          Clear the shell.
.example       Example queries.
.features      Shell features.
.files list    List all files.
.files add     Add files.
.files download $FILE Download a file.
.files drop   Drop all files.
.files drop $FILE Drop a single file.
.files track $FILE Collect file statistics.
.files paging $FILE Show file paging.
.files read $FILE Show file ready.
.open $FILE   Open database file.
.reset       Reset the shell.
.timer on/off Turn query timer on or off.
.output on/off Print results on or off.

Repositories:
https://github.com/duckdb/duckdb
https://github.com/duckdb/duckdb-wasm

Feedback:
https://github.com/duckdb/duckdb-wasm/discussions
```

Then I did some basic SQL to check it out:

```
duckdb> select count(*) from userdata1.parquet;

 count_star()
-----
         1000

Elapsed: 1 ms

duckdb>
```

```
duckdb> select first_name, last_name, email from userdata1.parquet
where country = 'Nigeria';

first_name | last_name | email
-----
Emilly     | Stewart  | estewart@opnsources.org
Annie      | Torres  | atorrest@hng.com
William    | Green   | wgreen@hphb.com
Jack       | Medina  | jmedina7@firo.gov
Jeremy     | Bennett | jbennett@wikipedia.org
Carlos     | Day     | cday@gravatar.com
Ryan       | Mills   | rmills@hngofire.com
Betty      | Gibson  | bgibsonka@asu.edu
Wanda     | Stanley | wstanley@sourceforge.net
Evelyn     | Spencer | espencer@ltd.com
George     | Howard  | ghoward@hquest.com

Elapsed: 2 ms
```

You can even do an 'explain':

```
duckdb> explain select first_name, last_name, email from
userdata1.parquet where country = 'Nigeria';

-- PROJECTION --
-- first_name
-- last_name
-- email
--
-- PARQUET_SCAN
-- id
-- country
-- first_name
-- last_name
-- email
--
-- Filters: country=Nigeria
-- AND country IS NOT NULL
```

A very nice feature with DuckDB is if you are working with Python for example, you can just add it as a library by adding "import duckdb" to your python script, and then it is in your python process, so it then feels very integrated into your program, unlike working with something like MySQL or Postgres. So, using our userdata1.parquet file, we could do something like this:

```
import duckdb
myconnector = duckdb.connect('myduckdb.duckdb')
cursor = myconnector.cursor()
cursor.execute("""
CREATE TABLE userdata(
  registration_date date,
  id int,
  first_name varchar,
  last_name varchar,
  email varchar,
  gender varchar,
  ip_address varchar,
  cc varchar,
  country varchar,
  birthdate varchar,
  salary float,
  title varchar,
  comments
)
""")

cursor.execute("COPY userdata FROM 'userdata1.parquet' (HEADER)");
print(cursor.execute("select count(*) from userdata").fetchall())
cursor.close()
conn.close()
```

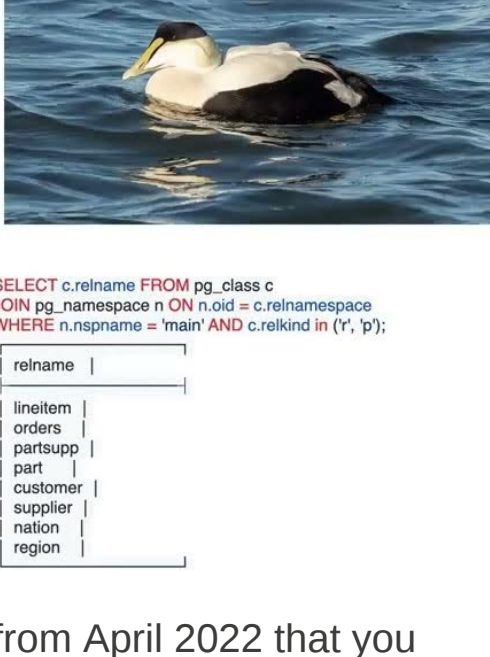
In the code snippet shown above, we connect to the 'myduckdb.duckdb' database, create a table that matches our parquet file, copy the data into it and then perform a simple count query.

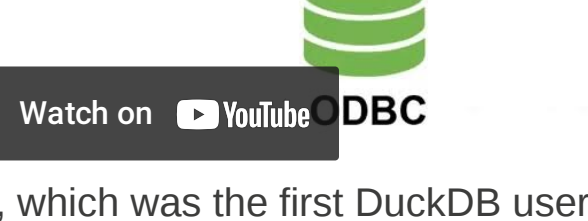
Summary

This is a really cool project. While I've been aware of the advantages of columnar data stores for about eight years because of Sisense initially, I only started working with them more extensively in the past year. I'm especially excited by their WASM implementation and the clever things they did with [Arrow](#) as a data protocol for the data import and query results. Not only is it a great technical example of WASM, but it has some great utility as well. Then, as I was wrapping this up, I ran across a

[The DuckDB User Group Meeting - April 14 2022](#)

- Python-Style List/Struct syntax
- Compatibility views for Postgres Catalog
- ODBC Driver
- Row-Group Based Storage





, which was the first DuckDB user group meeting from April 2022 that you might want to check out. Lots of great things on the roadmap. As a final note, I have nothing to do with this project or the people behind it, they don't even know I wrote this, so if I seem like a bit of a fanboy, it's because I'm legitimately very keen on the project.

by Shawn Gordon @ProgRockRec.
Technology and blockchain developer and enthusiast as well as a prolific musician.

[Check out my library](#)



Comments

[Signup](#) or [Login](#) to Join the Discussion

TAGS

#duckdb #olap #column-database #sqlite #amazon-redshift #database #database-design #sql-database

RELATED STORIES

- Blockchain and the Insufferable Millennial
Published at Jan 25, 2018 by [EzraBachus](#) [@EzraBachus](#)
- My Personal Guide to SQL Window Functions (Part 1)
Published at Nov 26, 2022 by [SILVIO](#) [@SILVIO](#)
- Scanning 2.6 Million Domains for Exposed .Env Files
Published at Nov 19, 2022 by [SALAL](#) [@SALAL](#)
- How to Install MySQL 8 on macOS Using Homebrew
Published at Oct 31, 2022 by [SALAL](#) [@SALAL](#)
- How to Build a Data-Driven Product Using Metabase
Published at Oct 17, 2022 by [SALAL](#) [@SALAL](#)
- How to Create World Leading Databases
Published at Oct 13, 2022 by [SALAL](#) [@SALAL](#)

THIS ARTICLE WAS FEATURED IN...

- TheCompany
- Buzzsumo
- DB-engines
- Newsbreak
- Perennet on Arweave

MENTIONED IN THIS STORY

COMPANIES

- Amazon
- Apache
- Assembly
- Funding
- YouTube

[Join HackerNoon](#)

Customized |