What The Heck Is DuckDB?

July 7th 2022 | 6 min | by @YogiRajSdec ★ 2,200 reads

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

Read by Dr. One (en-US)

Audio Presented by Tom

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

Check out my Stories

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:

Then I did some basic SQL to check it out:

```
duckDB> select count(*) from usersdata1.parquet;
┌──────────────┐
│ count_star() │
│    int64     │
├──────────────┤
│         1000 │
└──────────────┘
Elapsed: 1 ms
duckDB>
```

```
duckDB> select first_name, last_name, email from usersdata1.parquet
where country = 'Nigeria';
┌────────────┬────────────┬──────────────────────────────────────┐
│ first_name │ last_name  │                email                 │
│   varchar  │   varchar  │                varchar               │
├────────────┼────────────┼──────────────────────────────────────┤
│ Felix      │ Stewart    │ estewart8@opensource.org             │
│ Annie      │ Torres     │ atorres2@blog.com                    │
│ William    │ Ortiz      │ wortiz0@google.com                   │
│ Jack       │ Moore      │ jmoore7@flickr.com                   │
│ Jeremy     │ Bennett    │ jbennett6@wikipedia.org              │
│ Carlos     │ Day        │ cday@businessinsider.com             │
│ Ryan       │ Mills      │ rmills@nytimes.com                   │
│ Marie      │ Stone      │ bgiddins@ebay.com                    │
│ Bonnie     │ Giddins    │ bgiddins@ebay.com                    │
│ Evelyn     │ Spencer    │ espencer@ted.com                     │
│ George     │ Howard     │ ghoward3@bloomberg.com               │
└────────────┴────────────┴──────────────────────────────────────┘
Elapsed: 2 ms
```

You can even do an 'explain':

```
duckDB> explain select first_name, last_name, email from
usersdata1.parquet where country = 'Nigeria';
┌───────────────────────────┐
│┌─────────────────────────┐│
││       PROJECTION        ││
││    ────────────────     ││
││       first_name        ││
││        last_name        ││
││          email          ││
│└─────────────────────────┘│
│┌─────────────────────────┐│
││      PARQUET_SCAN       ││
││    ────────────────     ││
││         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 usersdata1.parquet file, we could do something like this:

```
import duckdb
myconnector = duckdb.connect('mydbs/duckdb.duckdb')
cursor = myconnector.cursor()
cursor.execute("""
CREATE TABLE usersdata(
    registration_dtim date,
    id                int
    first_name        varchar
    last_name         varchar
    email             varchar
    gender            varchar
    ip_address        varchar
    city              varchar
    country           varchar
    birthdate         varchar
    salary            float
    title             varchar
    comments
)
;;
"""
)

cursor.execute("""COPY usersdata FROM 'usersdata1.parquet' (HEADER)""")

print(cursor.execute('select count(*) from usersdata;').fetchall())
cursor.close()
conn.close()
```

In the code snippet shown above, we connect to the 'mydbs/duckdb.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 input 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

, 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.

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 on Jan 26, 2018 by @yogirajsdec, #business

My Personal Guide to SQL Window Functions (Part 1)
Published on Nov 3rd, 2021 by @yogi, #news

Scanning 2.6 Million Domains for Exposed .Env Files
Published on Nov 4th, 2021 by @yogi, #security#cybersecurity#cyber#cybers

How to Install MySQL 8 on macOS Using Homebrew
Published on Oct 31, 2021 by @yogi, #mysql

How to Build a Data-Driven Product Using Metabase
Published on Oct 27, 2021 by @yogirajsdec, #dataanalysis

How to Create World Leading Databases
Published on Oct 13, 2021 by @yogirajsdec#business#economics

THIS ARTICLE WAS FEATURED IN...

Thehackernoon | Noscrum | DS-engineer | Noodersuk | Permanent for browsers

MENTIONED IN THIS STORY

COMPANIES

Amazon | Apache | Assembly | Funding | YouTube