

Open in app ↗

Medium

 Search Write

DuckDB Database File as a New Standard for Sharing Data?



Josef Machytka

4 min read · 21 hours ago



1



This is not my original idea; I came across it in an excellent article titled “DuckDB Beyond the Hype” by Alireza Sadeghi. However, it immediately caught my attention because sharing data can often be a real pain. While we can use traditional formats like CSV or more modern options like Parquet, these require exporting data from the original source and subsequent import into the target database.

Which can lead to many manual steps and quite often we can during these operations encounter problems with compatibility of different versions of these formats or slightly different behavior of different versions of the same database. But DuckDB can mostly free us from these issues.

Sharing multiple tables adds even more complexity, often necessitating packaging files into tar/zip archives or using modern container formats like HDF5. Here is where DuckDB could also greatly simplify the process. All the

data is encapsulated in a single database file. Internally, the data is stored in a columnar format loosely based on Parquet, and is already compressed.

The idea from the article mentioned above was so inspiring that I decided to conduct some tests and collect a few numbers. Parquet format was originally designed for scientific numerical data. It stores strings as byte arrays, but handling large strings is not its primary use case. Therefore I expected that for strings DuckDB storage will be less efficient, which was conformed.

DuckDB Internals

Before diving into the tests and their results, let's take a closer look at DuckDB's internals. While it's easy to measure the total size of a DuckDB database file, there's currently no straightforward way to determine the precise size of individual tables. Neither the `duckdb_memory()` nor the `duckdb_tables()` internal metadata functions provide this information.

DuckDB's documentation notes that it collects statistics for each table and mentions the automatic creation of some indexes for table columns. These objects, however, are not directly visible to the user, and their sizes remain unknown. Additionally, there's no current option to disable the creation of these objects.

It would be beneficial if the DuckDB development team introduced configuration parameters in future versions to allow users to disable these hidden objects. Such a feature could help reduce database file sizes when the primary purpose is data sharing.

The Experiment

I asked ChatGPT to provide two example tables and Python code to generate CSV data for them with preset target sizes. The first table contains string data, while the second simulates purely numerical content typical for scientific measurements:

```
CREATE TABLE user_data (  
    user_id BIGSERIAL PRIMARY KEY,  
    first_name VARCHAR(50) NOT NULL,  
    last_name VARCHAR(50) NOT NULL,  
    email VARCHAR(100) NOT NULL UNIQUE,  
    signup_date TIMESTAMP NOT NULL,  
    last_login TIMESTAMP,  
    is_active BOOLEAN NOT NULL DEFAULT TRUE,  
    account_balance NUMERIC(12,2) NOT NULL DEFAULT 0.00,  
    country_code CHAR(2) NOT NULL,  
    favorite_number INTEGER,  
    profile_text TEXT,  
    checksum UUID NOT NULL  
);  
  
CREATE TABLE probe_measurements (  
    measurement_id BIGSERIAL PRIMARY KEY,  
    probe_id INTEGER NOT NULL,  
    measurement_time TIMESTAMP NOT NULL,  
    temperature NUMERIC(5,2) NOT NULL,  
    pressure NUMERIC(6,2) NOT NULL,  
    humidity NUMERIC(5,2) NOT NULL,  
    voltage NUMERIC(5,2) NOT NULL,  
    current NUMERIC(5,2) NOT NULL,  
    resistance NUMERIC(10,2) NOT NULL,  
    sample_rate INTEGER NOT NULL  
);
```

The Python code to generate data is straightforward, so I won't include it here to keep this article concise. I generated CSV files of three different sizes for these tables: 10 GB, 50 GB, and 100 GB and imported them first into PostgreSQL, and later used DuckDB to simulate data transfer.

Results

Table 1: "user_data" with strings

For the `user_data` table, which contains a significant amount of string data, DuckDB initially showed inefficiency with smaller datasets. A 10 GB CSV file (40.2 M rows) resulted in 11 GB big PostgreSQL table (size on the disk) and 14 GB big DuckDB database file. However, as the dataset size increased, the DuckDB numbers improved:

- A 50 GB CSV file (200 M rows) was imported to a 60 GB big PostgreSQL table and DuckDB database file of 35 GB.
- A 100 GB CSV file (400 M rows) was imported into a 111 GB big PostgreSQL table and DuckDB database file of 68 GB.

Table 2: "probe_measurements" with numerical data

The `probe_measurements` table, containing only numerical and timestamp data, produced far more impressive results.

- A 10 GB CSV file (143 M rows) was imported into a 15 GB big PostgreSQL table, but DuckDB database file that was only 2.8 GB big.

Larger datasets also scaled well:

- A 50 GB CSV file (710 M rows) was imported into a 72 GB big PostgreSQL table, but DuckDB database file was only 14 GB.
- A 100 GB CSV file (1,420 M rows) was imported into a 144 GB big PostgreSQL table, but DuckDB database file was only 27 GB.

These results nicely demonstrate effectiveness of DuckDB storage format for numerical data.

Summary

While CSV and Parquet formats are well-known and reliable, they require additional manual steps to be effective mediums for sharing data between systems or clients. DuckDB, with its direct interface into PostgreSQL and MySQL databases and its straightforward Python integration for other databases, could significantly simplify the process. Sharing a single DuckDB database file containing already imported data and using DuckDB engine itself for copying data makes it remarkably easy to transfer and load data into target systems.



Etl



Written by Josef Machytka

Edit profile

43 Followers · 10 Following

I work as Professional Service Consultant - PostgreSQL specialist in NetApp Deutschland GmbH, Open Source Services division.

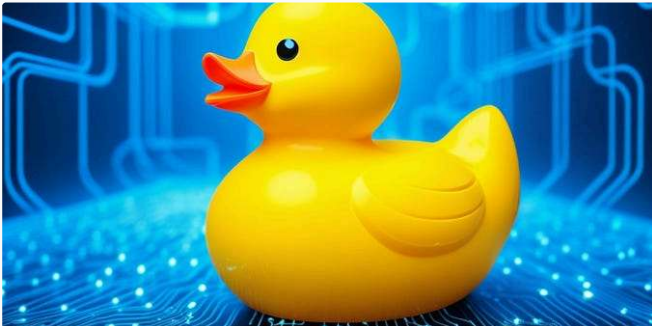
No responses yet




What are your thoughts?

Respond

More from Josef Machytka




 Josef Machytka

Extending DuckDB ETL Capabilities with Python

```

-- Create the table
CREATE TABLE special_data_types (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  status ENUM('active', 'inactive', 'pending') NOT NULL,
  permissions SET('read', 'write', 'execute') NOT NULL,
  small_number TINYINT NOT NULL,
  medium_number MEDIUMINT NOT NULL,
  description TEXT,
  data BLOB,
  created_at DATE NOT NULL
);

-- Insert 10 rows of data
INSERT INTO special_data_types (name, status, permissions, small_number, medium_number, description, data, created_at)
VALUES ('Alice', 'active', 'read,write', 5, 1000, 'Alice description', 'Alice data', '2023-01-01'),
('Bob', 'inactive', 'read', 10, 2000, 'Bob description', 'Bob data', '2023-02-01'),
('Charlie', 'pending', 'write,execute', 15, 3000, 'Charlie description', 'Charlie data', '2023-03-01'),
('David', 'active', 'read,write,execute', 20, 4000, 'David description', 'David data', '2023-04-01'),
('Eve', 'inactive', 'execute', 25, 5000, 'Eve description', 'Eve data', '2023-05-01'),
('Frank', 'pending', 'read,write', 30, 6000, 'Frank description', 'Frank data', '2023-06-01'),
('Grace', 'active', 'read', 35, 7000, 'Grace description', 'Grace data', '2023-07-01'),
('Hank', 'inactive', 'write,execute', 40, 8000, 'Hank description', 'Hank data', '2023-08-01'),
('Ivy', 'pending', 'read,write,execute', 45, 9000, 'Ivy description', 'Ivy data', '2023-09-01'),
('Jack', 'active', 'execute', 50, 10000, 'Jack description', 'Jack data', '2023-10-01');
```

 Josef Machytka

DuckDB as a Rudimentary Data Migration Tool

DuckDB has recently become my go-to solution for small ETL tasks. It is an...


After exploring how to use DuckDB as an intelligent ETL tool for PostgreSQL, and how...

Nov 24👤 26🔖⋮

Nov 30👤 24💬 1🔖⋮

```
D SELECT
  u.username,
  u.email,
  o.order_date,
  o.total_amount,
  p.product_name,
  od.quantity,
  p.price,
  (od.quantity * p.price) AS total_price
FROM
  pg13.users u
JOIN
  pg15.orders o ON u.user_id = o.user_id
JOIN
  pg15.order_details od ON o.order_id = od.order_id
JOIN
  pg14.products p ON od.product_id = p.product_id
ORDER BY
  u.username, o.order_date;
```

username	email	order_date	total_amount	product_name	quantity	price	total_price
varchar	varchar	date	decimal(10,2)	varchar	int32	decimal(10,2)	decimal(10,2)
Alice	alice@example.com	2024-11-20	150.00	Mouse	2	20.00	40.00
Alice	alice@example.com	2024-11-20	150.00	Keyboard	1	50.00	50.00


 Josef Machytka

Easy Cross-Database Selects with DuckDB

DuckDB was created with simplicity and ease of use in mind. In my previous article, I...

Nov 26👤 14🔖⋮



 Josef Machytka

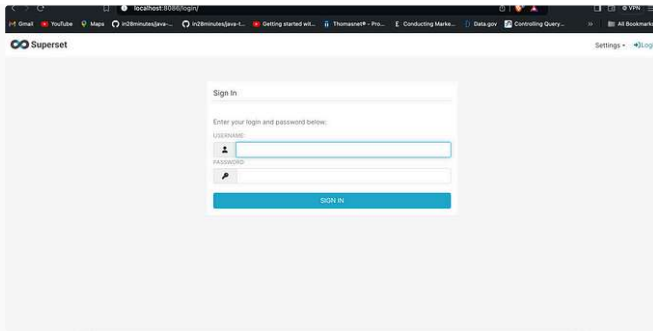
Quick and Easy Data Exports to Parquet Format Using DuckDB

The Parquet format has become almost an industry standard for Data Lakes and Data...

Dec 5👤 8🔖⋮

See all from Josef Machytka

Recommended from Medium

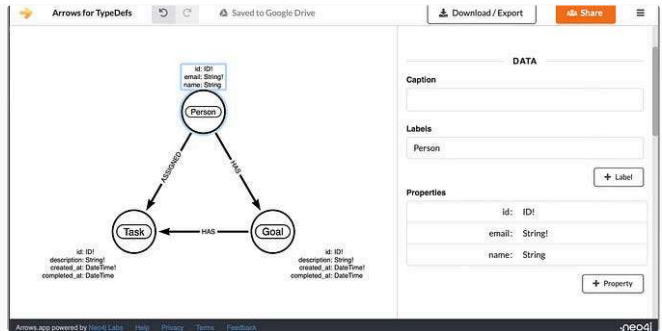


 Vivek Suresh

DuckDB and Superset Combination to Power Analytics

Since DuckDB has been all the rage in the past 1 year in data engineering circles and...

Sep 17  13



 Jason Koo

Quickly Create GraphQL TypeDefs with Arrows

Arrows.app is a free graph data modeling app originally created to swiftly define property...

Dec 9  4

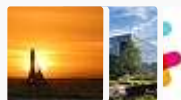


Lists



Staff picks

791 stories · 1534 saves



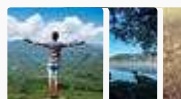
Stories to Help You Level-Up at Work

19 stories · 904 saves



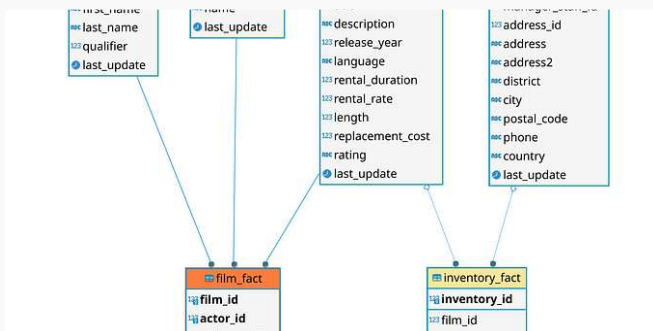
Self-Improvement 101


20 stories · 3173 saves



Productivity 101

20 stories · 2685 saves



 In Dev Genius by sefedo



 Kemal Öz

End to end ELT process example for DW using Postgres.

The purpose of this paper is mostly educational. It targets developers that are...

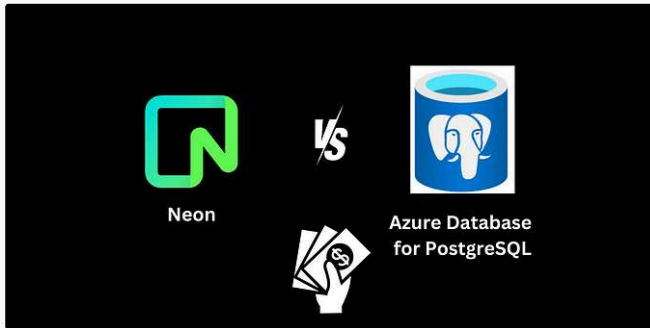
Dec 21 🖱 156




SQL Query Journey in Postgres

In PostgreSQL, when a client sends an SQL query, it undergoes a complex journey from...

5d ago 🖱 24




 In Data Engineer Things by Bobur

Cost Comparison: Neon vs. Azure Database for PostgreSQL Flexible...

Neon offers a serverless architecture which makes it more cost-effective and easy to...

4d ago 🖱 5



 In Towards Data Engineering by Burak Uğur

Create Data Lakehouse Using Spark+Iceberg+Nessie+Dremio

Hi everyone, in this article I will talk about the concept of data lakehouse and develop...

Sep 19 🖱 77



See more recommendations