

Postgres in da ClickHouse

David E. Wheeler
Principal Engineer



Postgres Summit
2026-09-30

 **Hello!**

I'm David Wheeler

Principle Engineer, ClickHouse

PostgreSQL user since 2000

PostgreSQL contributor since 2008

pgTAP: Unit testing for PostgreSQL

Sqitch: Database change management

PGXN: PostgreSQL Extension network

pg_clickhouse: A foreign data wrapper





ClickWHAT?

Open Source

What is ClickHouse?

Development started 2009

Production 2012

Open sourced 2016

#1 open source analytics database on DB-Engines

Column Oriented

What is ClickHouse?

Best for aggregations

Files per column

Sorting and indexing

Background merges

Distributed

What is ClickHouse?

Replication

Sharding

Separation of storage and compute

Bi-directional

Cross-region

OLAP

What is ClickHouse?

Analytics use cases

Aggregations

Visualizations

Mostly immutable data

Company

What is ClickHouse?

Founded 2021

Cloud Service 2022

\$100m ARR 2025

Postgres Service 2026

ClickGres? Nah...



World's Most Popular Relational Database

Transactions

Low-latency CRUD

ACID compliance

Rich SQL interface

Robust Extension Framework

Vast ORM and app ecosystem

Open Source



World's Fastest Analytical Database

Columnar Storage

Skip indexes

Incremental Materialized Views

Customized Table Engines

Lazy Materialization

Native JSON and FTS capabilities

Open Source

Two Great Tastes





Published on: April 29, 2022

8 min read

Two sizes fit most: PostgreSQL and Clickhouse

Relational databases are not in decline. Here's why.



Sid Sijbrandij

Since the introduction of [System R](#) in 1974, relational databases in general, and SQL databases in particular, have risen to become the dominant approach to data persistence in the industry, and have maintained that dominance despite various significant challengers. Though some have rumored the death and decline of traditional relational databases, PostgreSQL has turned out to be an improvement on its predecessors, as well as its supposed

ClickHouse Managed

Postgres

Fast, scalable, enterprise-grade Postgres, natively integrated with ClickHouse.

Built for real-time and AI-driven applications.

A Unified Data Stack that combines Postgres for transactions and ClickHouse for analytics, with no added complexity!

[Start free trial](#)

[Learn more](#)

aws



Available on AWS and in **Private Preview** on GCP

“Combine Postgres for transactions with ClickHouse for analytics to deliver best-in-class performance and scalability with minimal operational overhead.”

ClickHouse Managed Postgres

The unified data stack

OLTP complements ClickHouse OLAP

Fastest Postgres with fastest ClickHouse

Local NVMe for unlimited IOPs & 10x performance



ClickHouse Managed Postgres

Enterprise Postgres

PITR

Automated backups

Read replicas & fast failover

Connection pooling

Major version upgrades

Developer experience

One click-provisioning

ClickHouse Cloud console

Metrics and monitoring

Query Performance Insights

API & Terraform support

Extending the Stack

Extending OLTP + OLAP

Open Source ClickHouse/Postgres Interop

pg_stat_ch

pg_clickhouse

chdb

WalShadow

pg_stat_ch

pg_stat_ch

Export every metric to ClickHouse

Every query execution is an event

Exports duration, rows, buffers, WAL, CPU, errors

Slice by time, application, user, database, or error code

Percentiles and trends are simple ClickHouse SQL

Aggregation by ClickHouse

Materialized views for QPS, latency percentiles, load-by-app, etc.

5µs overhead per query, never in critical path

Background worker ships metrics to ClickHouse

```
shared_preload_libraries = 'pg_stat_ch'  
track_io_timing = on      # Enables I/O timing
```

```
# ClickHouse connection (change for your setup)  
pg_stat_ch.clickhouse_host = 'localhost'  
pg_stat_ch.clickhouse_port = 9000  
pg_stat_ch.clickhouse_database = 'pg_stat_ch'
```

```
# TLS (recommended for production)  
pg_stat_ch.clickhouse_use_tls = on  
pg_stat_ch.clickhouse_skip_tls_verify = off
```

Refresh

Time period

Last week



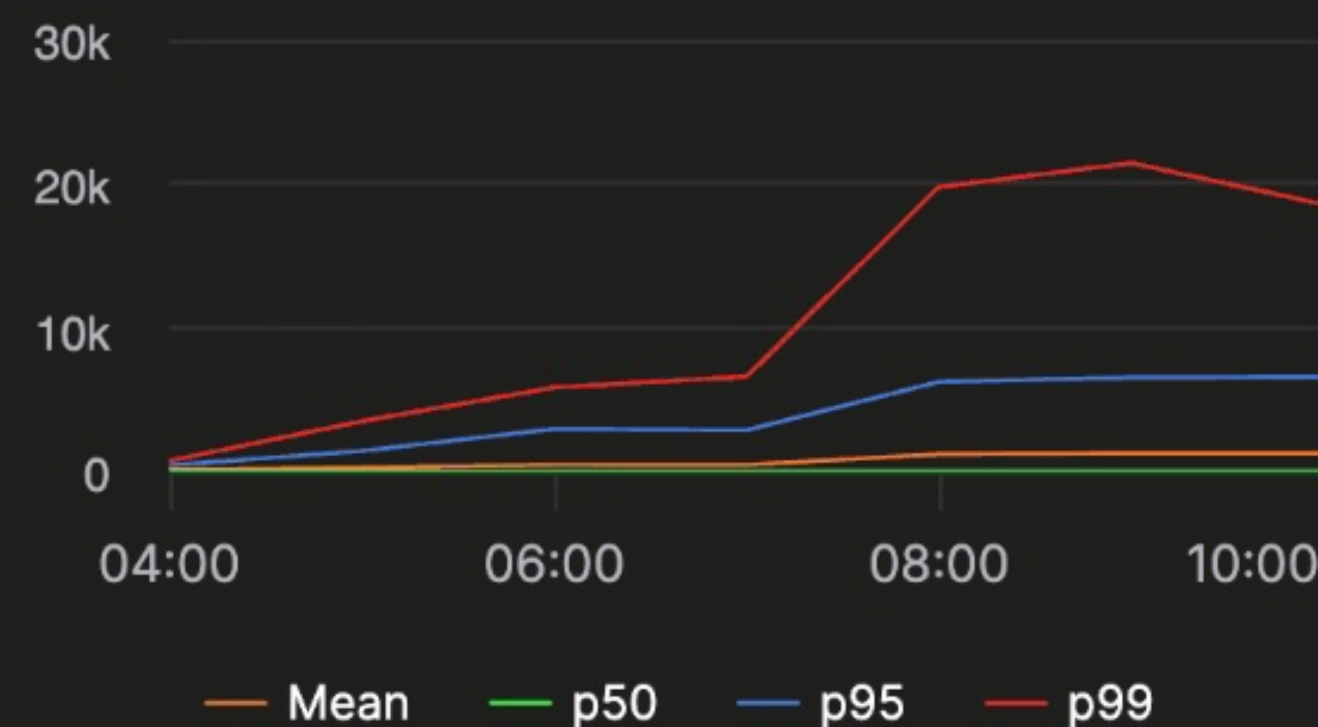
Queries / sec

Normalized from query_count buckets



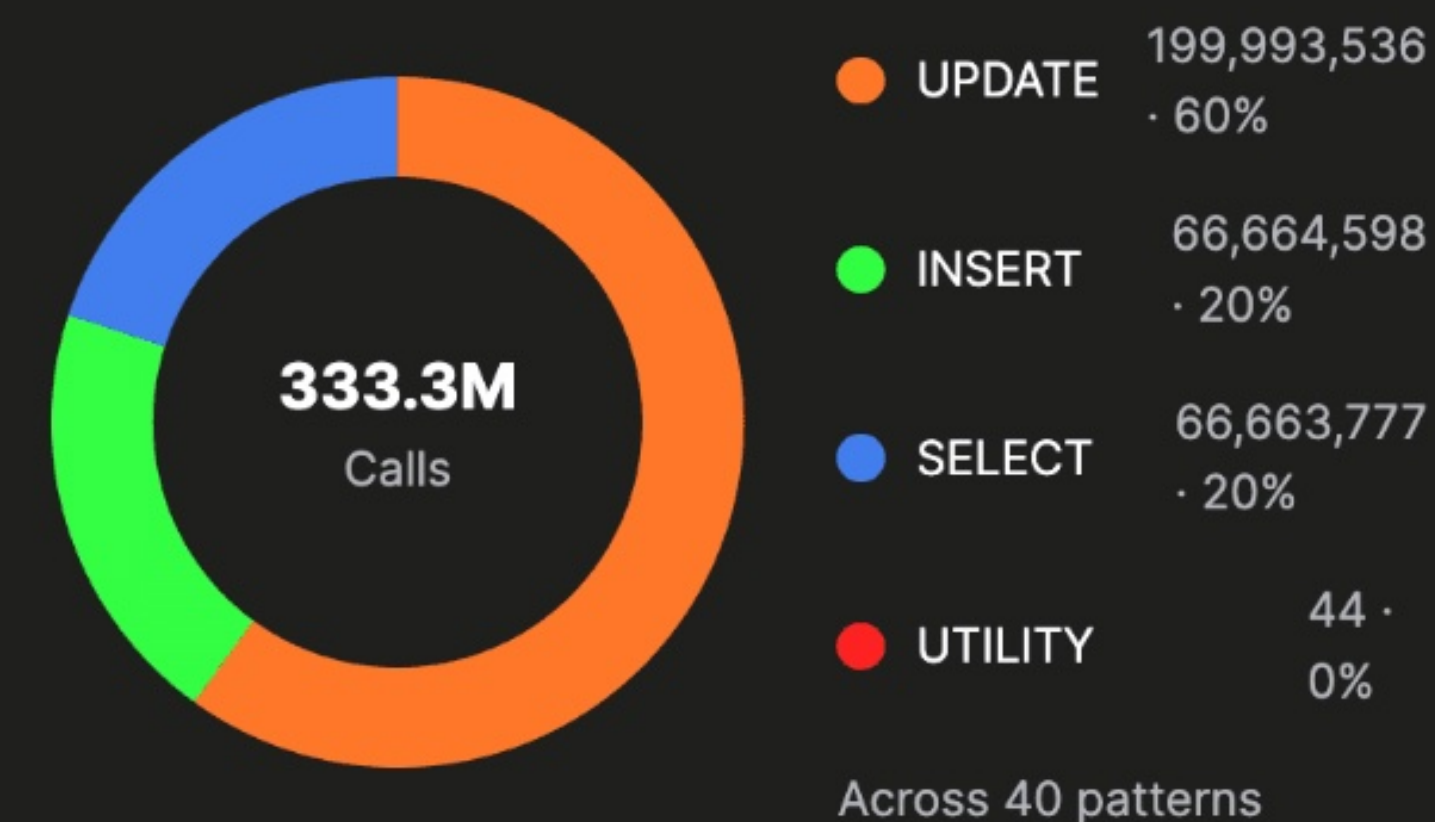
Query latency

Mean, p50, p95, and p99 over time



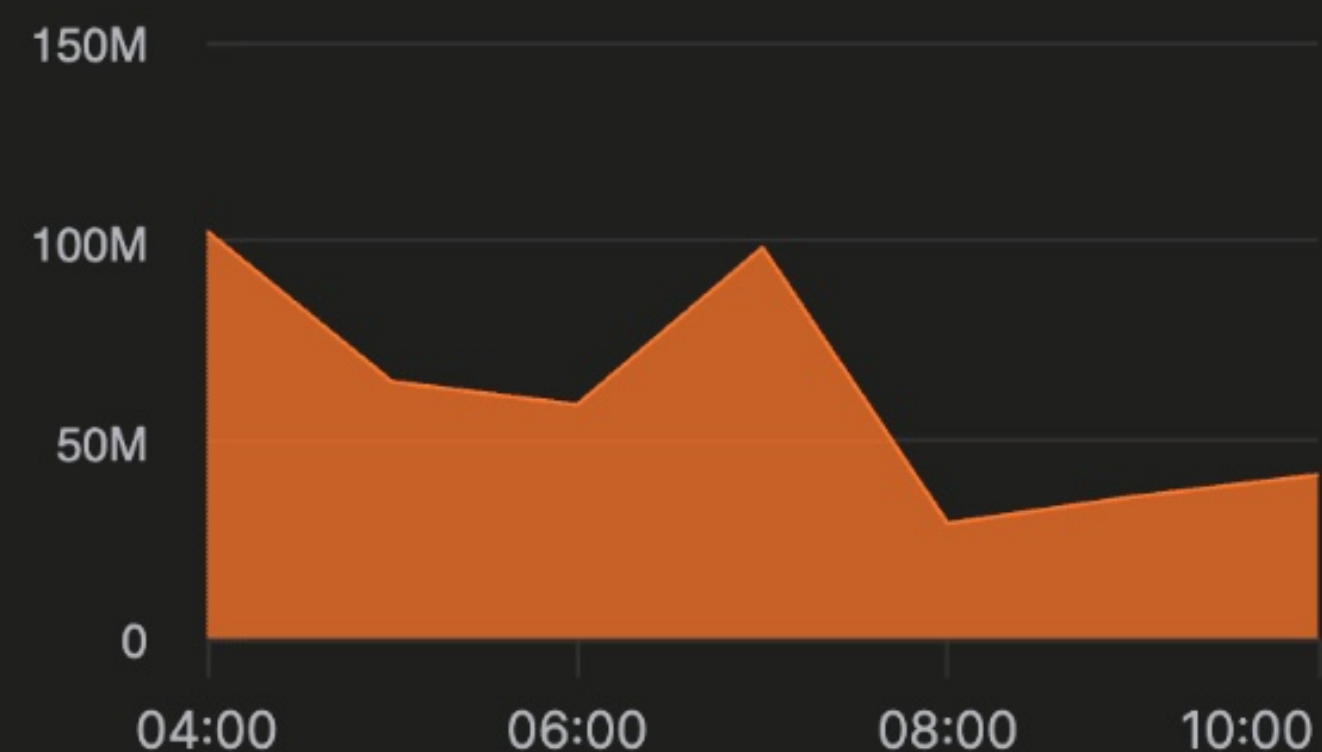
Operations breakdown

Share of calls by SQL operation



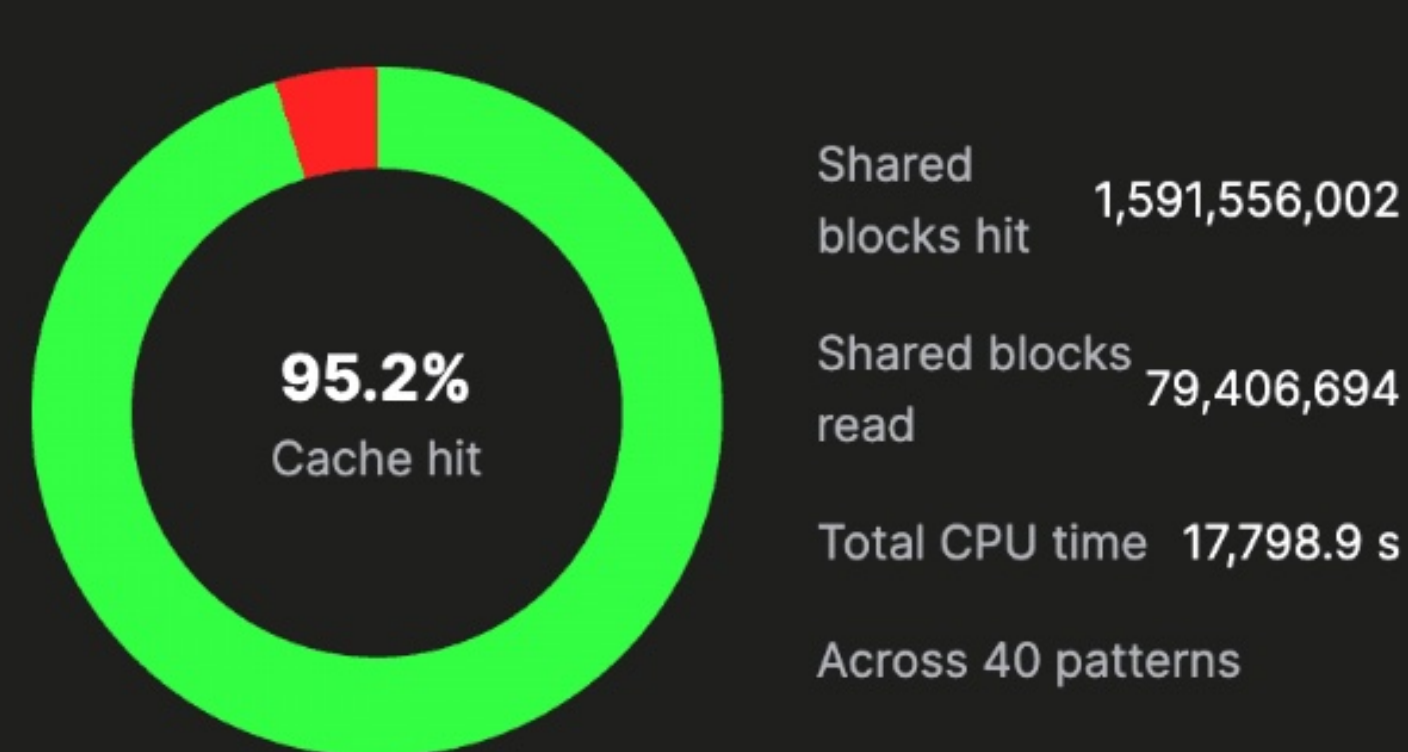
Rows returned / affected

Total 429.3M rows in the selected range



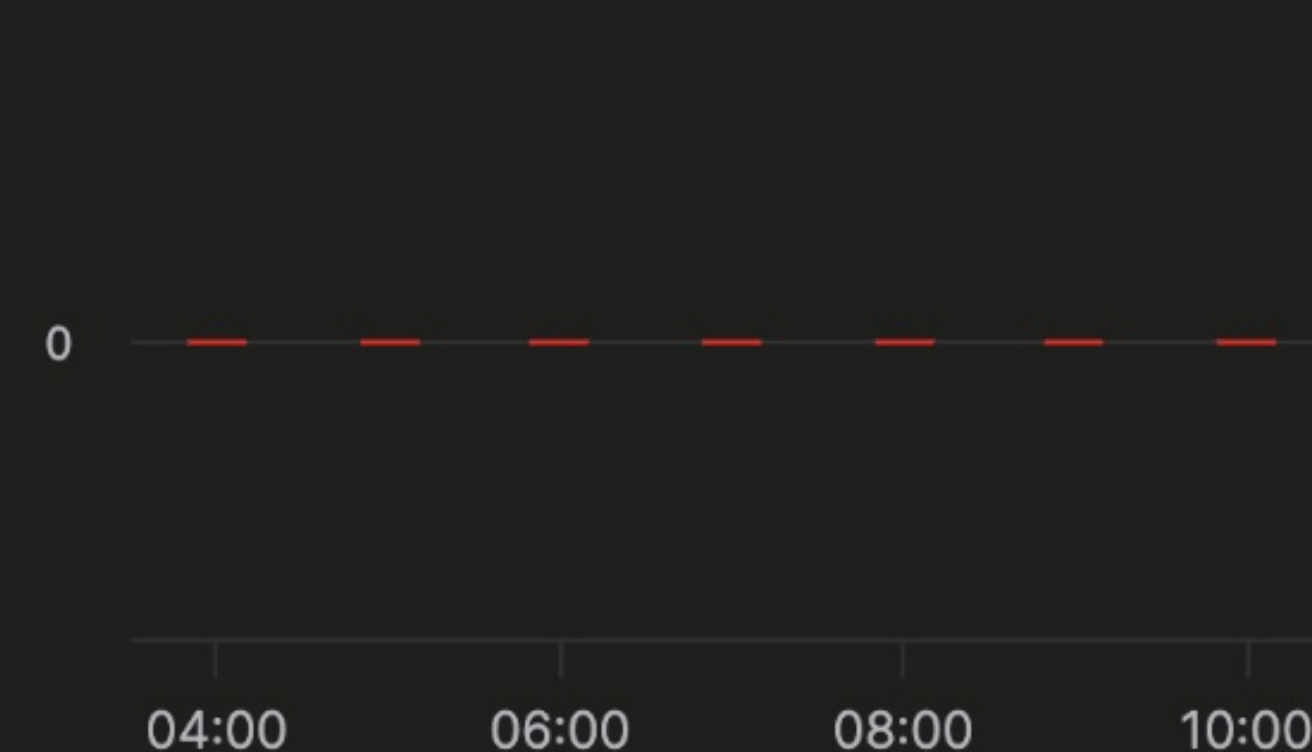
Buffer hit ratio

Shared block hit ratio for aggregated query patterns



Errors

0 errors in the selected range



Query Insights Demo

pg_clickhouse

pg_clickhouse

Query ClickHouse from Postgres

Provide ClickHouse query execution from PostgreSQL

Minimize query migration while adopting ClickHouse for analytics

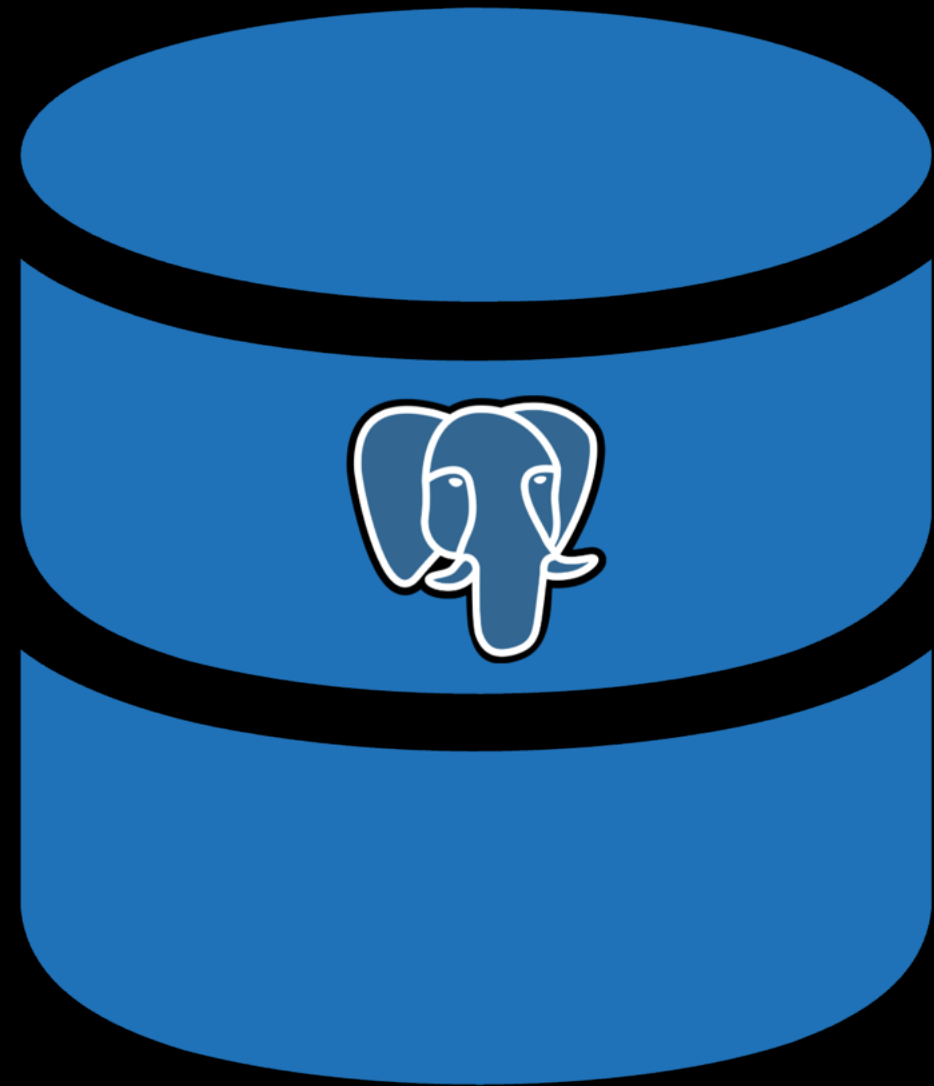
Push down query execution to ClickHouse

INNER, LEFT, RIGHT, FULL, SEMI, ANTI JOIN pushdown

Correlated/uncorrelated, EXISTS, ANY/IN subquery pushdown

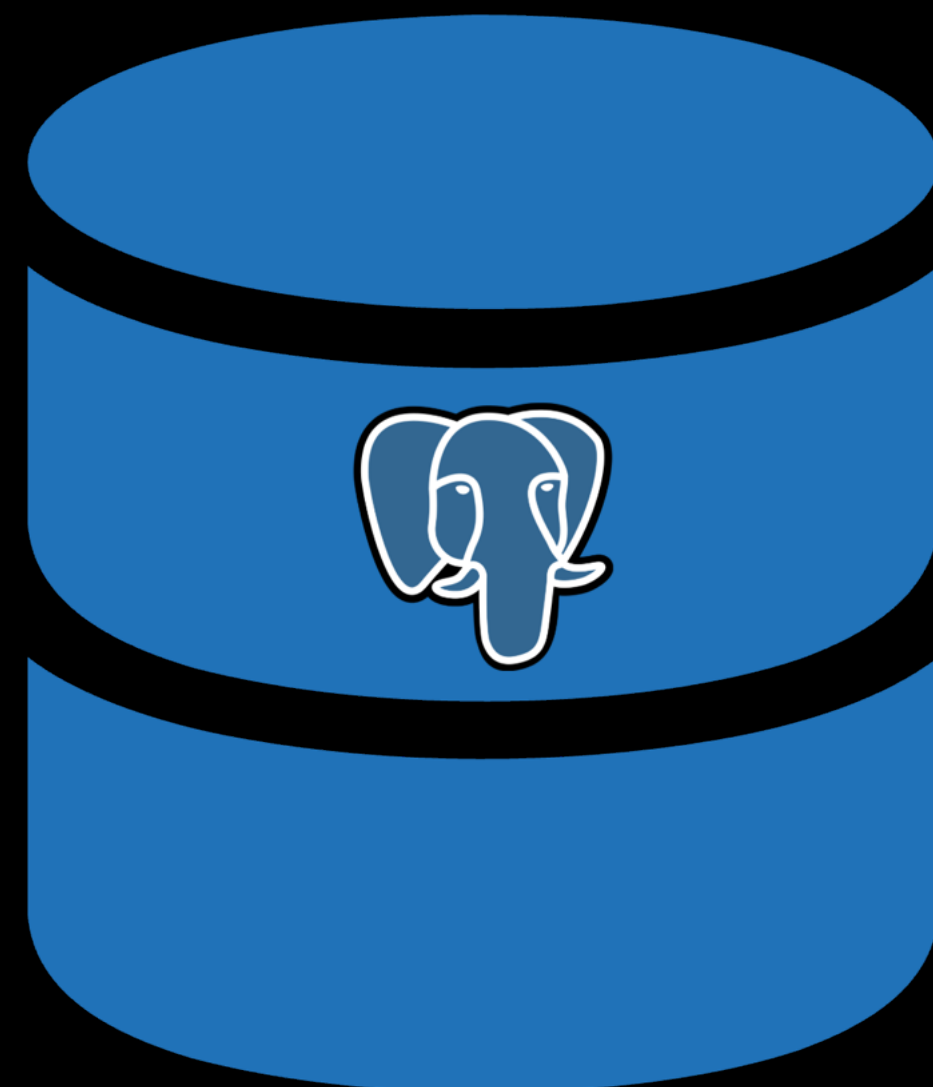
Aggregate function pushdown

Foundation for continual query and pushdown evolution



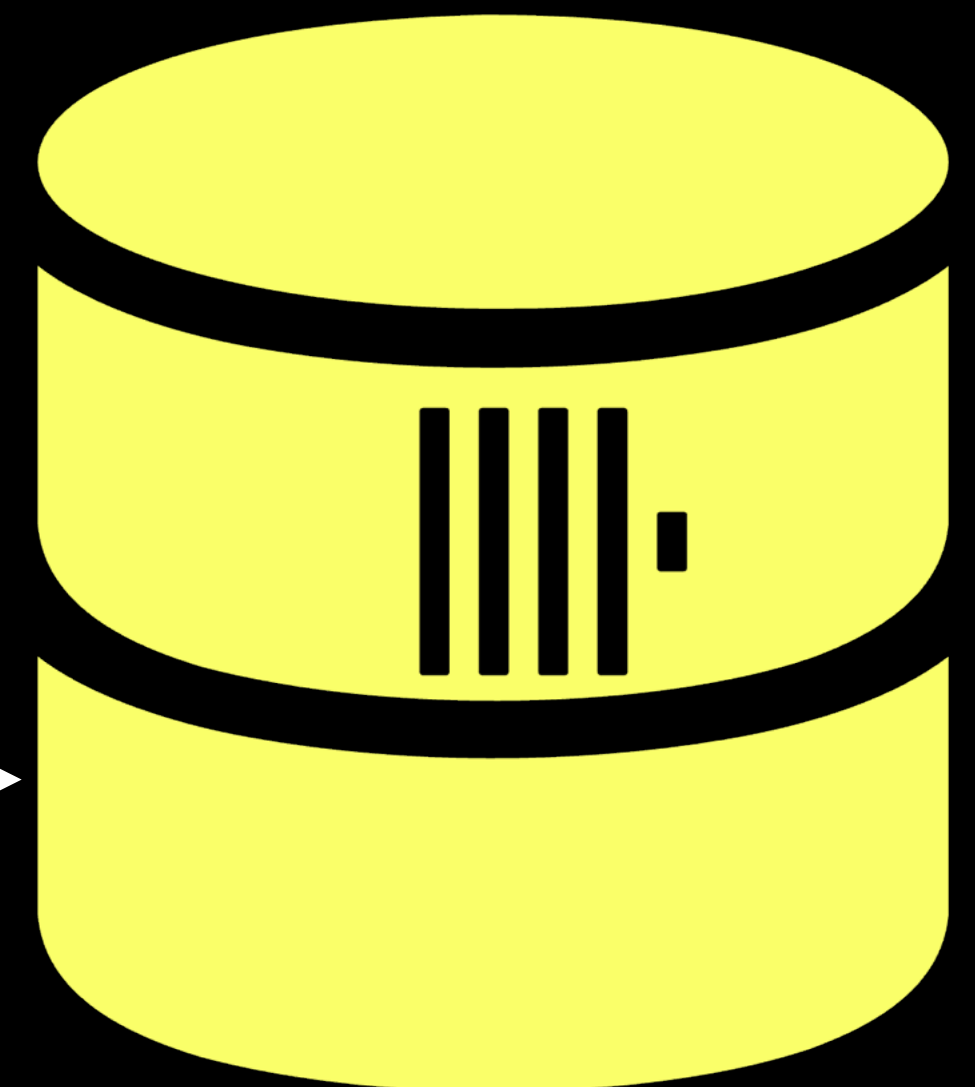
```
SELECT AdvEngineID, COUNT(*)  
FROM hits  
WHERE AdvEngineID <> 0  
GROUP BY AdvEngineID  
ORDER BY COUNT(*) DESC;
```

Time: 25838.272 ms



```
SELECT AdvEngineID, COUNT(*)  
FROM hits  
WHERE AdvEngineID <> 0  
GROUP BY AdvEngineID  
ORDER BY COUNT(*) DESC;
```

Time: 430.377 ms



TPC-H Pushdown

Query	PostgreSQL	pg_clickhouse	Pushdown
Query 1	4483 ms	59 ms	✓
Query 2	588 ms	24 ms	✓
Query 3	786 ms	62 ms	✓
Query 4	550 ms	38 ms	✓
Query 5	721 ms	1439 ms	✓
Query 6	592 ms	17 ms	✓
Query 7	639 ms	29 ms	✓
Query 8	398 ms	383 ms	✓
Query 9	2842 ms	162 ms	✓
Query 10	860 ms	125 ms	✓
Query 11	276 ms	21 ms	*

Query	PostgreSQL	pg_clickhouse	Pushdown
Query 12	963 ms	26 ms	✓
Query 13	1037 ms	1354 ms	
Query 14	675 ms	30 ms	✓
Query 15	2520 ms	387 ms	
Query 16	539 ms	823 ms	
Query 17	2107 ms	37 ms	✓
Query 18	5230 ms	7228 ms	
Query 19	68 ms	47 ms	✓
Query 20	473 ms	28 ms	
Query 21	1145 ms	4470 ms	
Query 22	270 ms	45 ms	*

ClickBench

pg_clickhouse is the fastest PostgreSQL extension, outperforming all other Postgres analytics extensions, performing only slightly slower than native ClickHouse itself.

		ClickHouse (c8g.4xlarge)	pg_clickhouse (c8g.4xlarge)	ClickHouse (c6a.4xlarge)	pg_clickhouse (c6a.4xlarge)
✓					
Load time:		237s (×1.13)	209s (×1.00)	237s (×1.13)	243s (×1.16)
Data size:		14.22 GiB (×1.00)	14.21 GiB (×1.00)	14.21 GiB (×1.00)	14.21 GiB (×1.00)
✓	Q0.	0.002s (×1.04)	0.012s (×1.96)	0.002s (×1.07)	0.013s (×2.02)
✓	Q1.	0.008s (×1.01)	0.018s (×1.61)	0.008s (×1.04)	0.020s (×1.68)
✓	Q2.	0.021s (×1.00)	0.035s (×1.45)	0.026s (×1.15)	0.053s (×2.03)
✓	Q3.	0.174s (×1.56)	0.214s (×1.90)	0.111s (×1.03)	0.179s (×1.60)
✓	Q4.	0.312s (×1.00)	0.330s (×1.06)	0.471s (×1.50)	0.481s (×1.53)
✓	Q5.	0.504s (×1.26)	0.524s (×1.31)	0.542s (×1.35)	0.544s (×1.35)
✓	Q6.	0.009s (×1.00)	0.024s (×1.83)	0.010s (×1.05)	0.022s (×1.71)
✓	Q7.	0.008s (×1.00)	0.022s (×1.74)	0.011s (×1.14)	0.027s (×2.01)
✓	Q8.	0.325s (×1.10)	0.382s (×1.29)	0.467s (×1.57)	0.470s (×1.58)
✓	Q9.	0.460s (×1.28)	0.492s (×1.37)	0.560s (×1.55)	0.549s (×1.52)
✓	Q10.	0.163s (×1.40)	0.205s (×1.74)	0.151s (×1.30)	0.167s (×1.43)
✓	Q11.	0.258s (×1.41)	0.284s (×1.55)	0.217s (×1.20)	0.224s (×1.23)
✓	Q12.	0.496s (×1.00)	0.548s (×1.11)	0.749s (×1.51)	0.789s (×1.59)
✓	Q13.	0.740s (×1.10)	0.799s (×1.19)	1.020s (×1.51)	1.003s (×1.49)
✓	Q14.	0.457s (×1.06)	0.520s (×1.21)	0.675s (×1.56)	0.661s (×1.53)
✓	Q15.	0.222s (×1.00)	0.248s (×1.11)	0.407s (×1.79)	0.413s (×1.82)
✓	Q16.	1.007s (×1.04)	1.072s (×1.10)	1.774s (×1.82)	1.803s (×1.85)
✓	Q17.	0.686s (×1.00)	0.749s (×1.09)	1.202s (×1.74)	1.253s (×1.81)
✓	Q18.	1.694s (×1.00)	1.801s (×1.06)	3.368s (×1.98)	3.361s (×1.98)
✓	Q19.	0.016s (×1.00)	0.030s (×1.53)	0.065s (×2.82)	0.045s (×2.07)
✓	Q20.	2.671s (×1.00)	2.819s (×1.06)	2.825s (×1.06)	2.805s (×1.05)
✓	Q21.	2.951s (×1.03)	3.046s (×1.07)	2.896s (×1.02)	2.878s (×1.01)
✓	Q22.	0.000s (×0.00)	0.000s (×0.00)	0.000s (×0.00)	0.000s (×0.00)

The background of the slide features a series of stylized, overlapping mountain peaks. The peaks are rendered in various shades of purple and blue, with the foreground peaks being a darker, more saturated blue and the background peaks fading into a lighter, hazy purple. The overall effect is a soft, atmospheric landscape.

pg_clickhouse Demo

chdb

chdb

Import/Export URLs using chDB

COPY data to/from URLs

Links libchdb, in-process ClickHouse

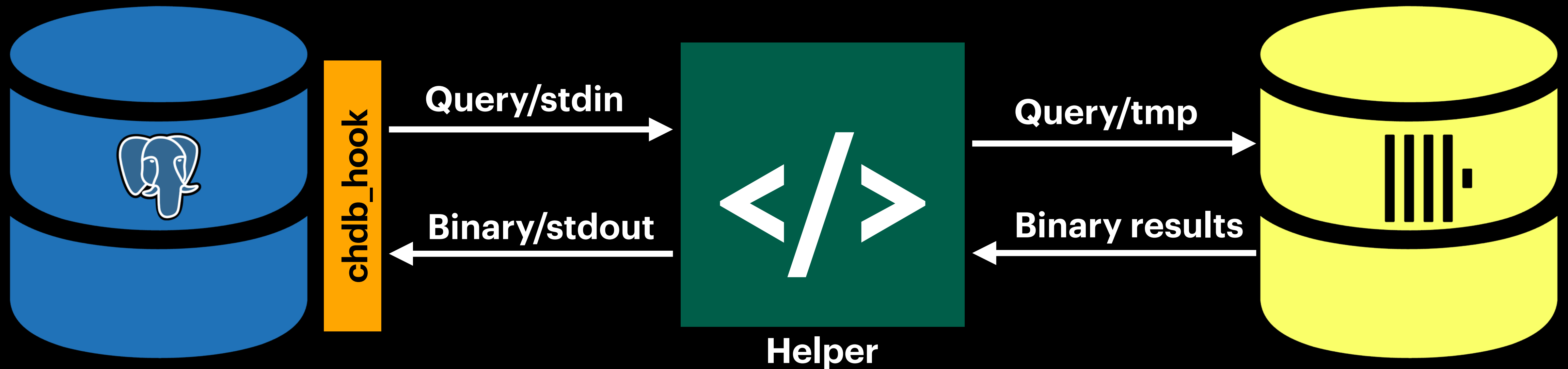
S3, GCP, Azure Blob, files, etc.

TSV, CSV, JSON, BSON, Prometheus, Protobuf, Avro, Parquet, Arrow, XML, CapnProto, Markdown, MsgPack, ORC...

Same data type mapping as pg_clickhouse

Derive from table or define for table

chdb_hook Architecture



```
CREATE TABLE times (  
    id      INT NOT NULL,  
    months  INT NOT NULL,  
    days    INT NOT NULL  
);
```

```
LOAD 'chdb_hook';
```

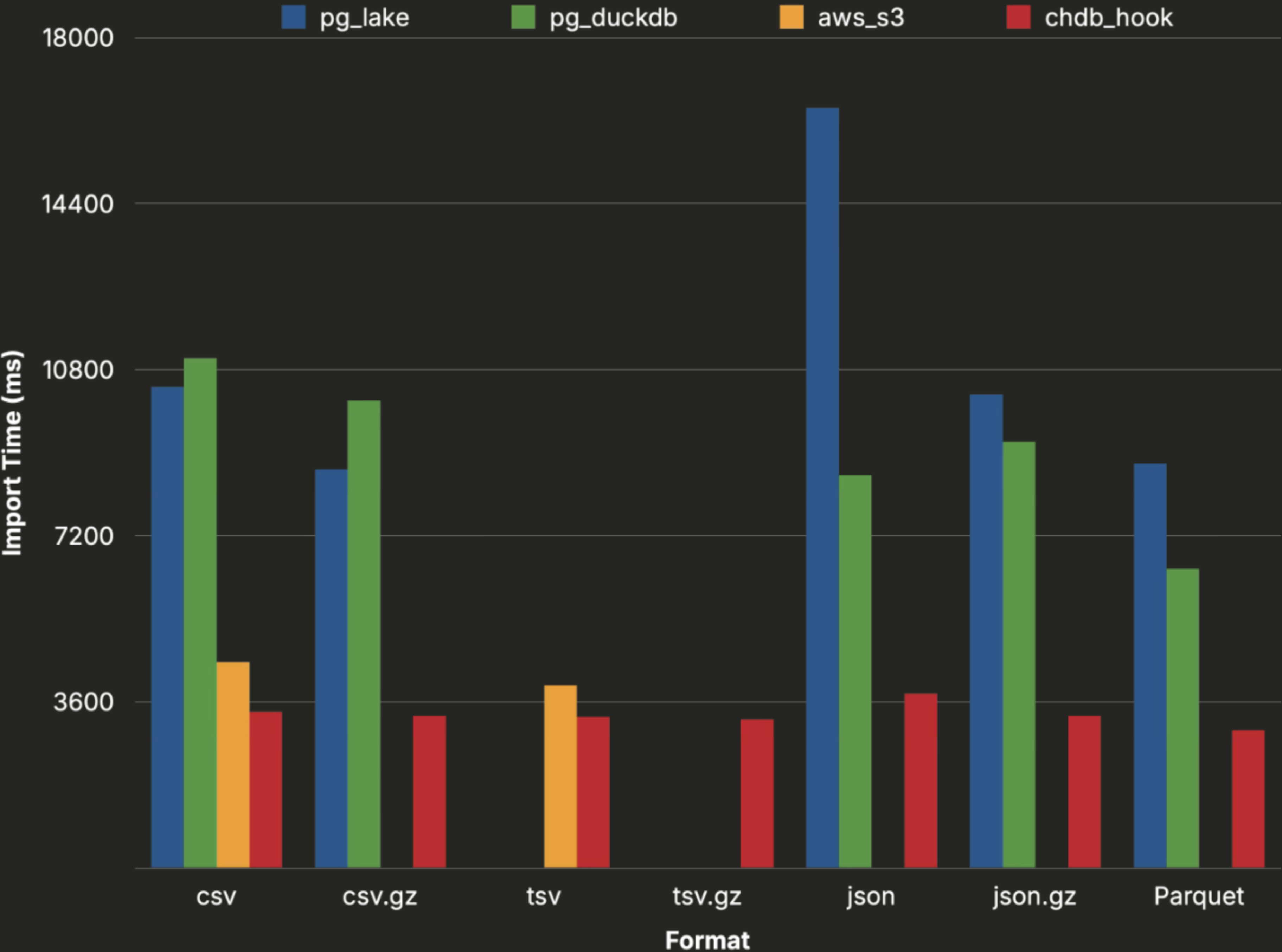
```
COPY times FROM s3://datasets-documentation/my-test-  
bucket-768/{some,another}_prefix/  
some_file_{1..3}.csv';
```

```
CREATE TABLE reviews () WITH (  
    encoding_check = 'replace',  
    copy_from = 's3://datasets-documentation/  
amazon_reviews/  
amazon_reviews_2015.snappy.parquet'  
);
```

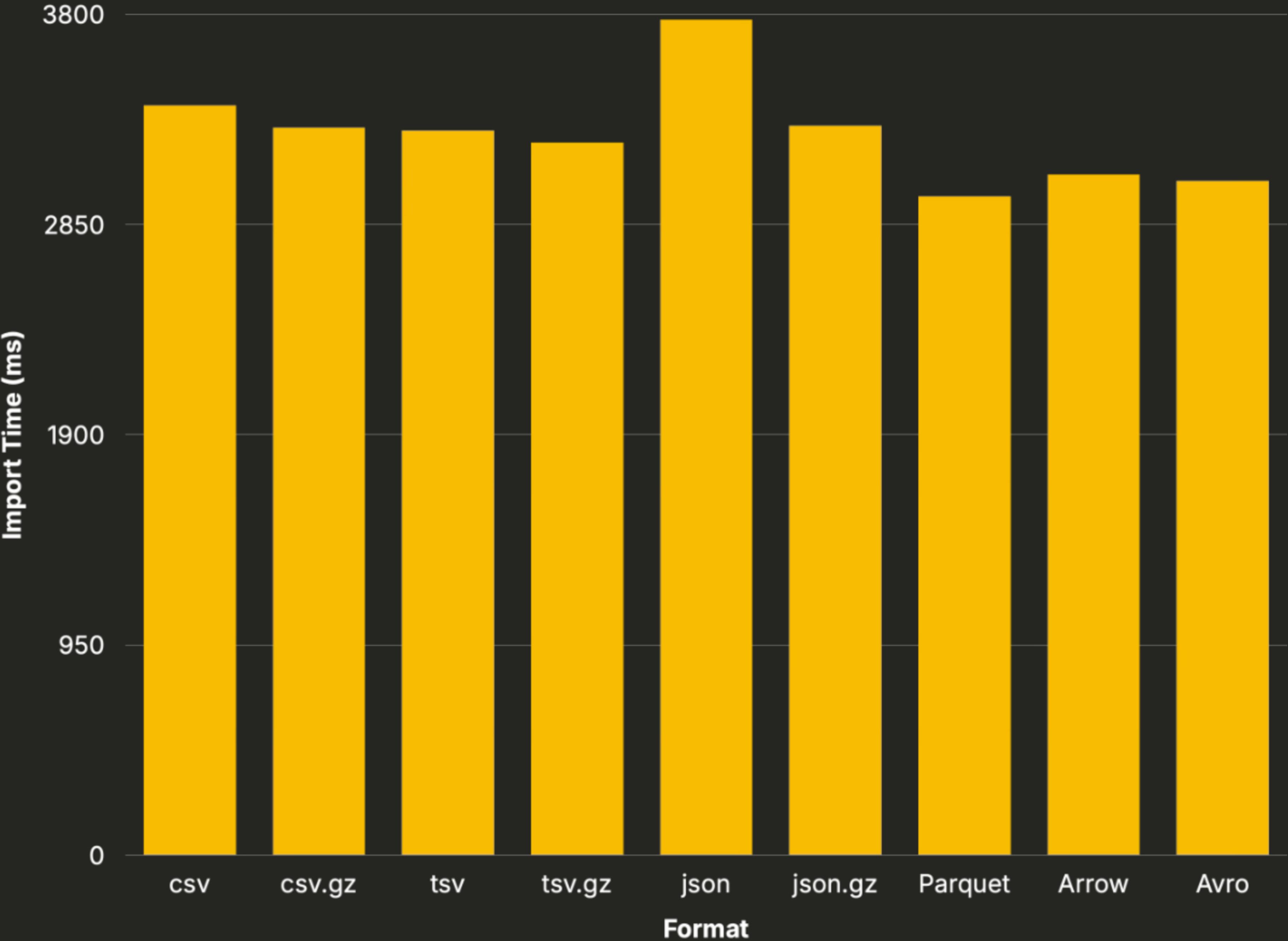
try=# \d reviews

Table "public.reviews"				
Column	Type	Collation	Nullable	Default
review_date	integer			
marketplace	text			
customer_id	numeric(20,0)			
review_id	text			
product_id	text			
product_parent	numeric(20,0)			
product_title	text			
product_category	text			
star_rating	smallint			
helpful_votes	bigint			
total_votes	bigint			
vine	boolean			
verified_purchase	boolean			
review_headline	text			
review_body	text			

COPY FROM 1m Taxi Trips



COPY FROM
1m Taxi Trips



The logo for WalShadow, featuring the text "WalShadow" in a bold, white, sans-serif font. The text is positioned on the left side of the image, set against a dark background with stylized, layered mountain ranges in various shades of gray and black. The mountains are depicted with smooth, undulating peaks, creating a sense of depth and a minimalist landscape aesthetic.

WalShadow

WalShadow

Sub-second physical replication to ClickHouse

Replicate from Postgres to ClickHouse from physical WAL

Postgres transactions visible in ClickHouse ~200ms

Consumes same WAL as Postgres physical replication

Decodes outside the database

Writes ClickHouse-native blocks directly into ClickHouse

Close to Postgres physical standby latency

Data immediately available in ClickHouse

WalShadow

Sub-second physical replication to ClickHouse

Full replication cycle

Not logical replication

Schema changes

Initial load

No replication slot

Create Table

Continuous replication

None of its overhead

Add Column

Schema evolution

**Significantly less
resource overhead**

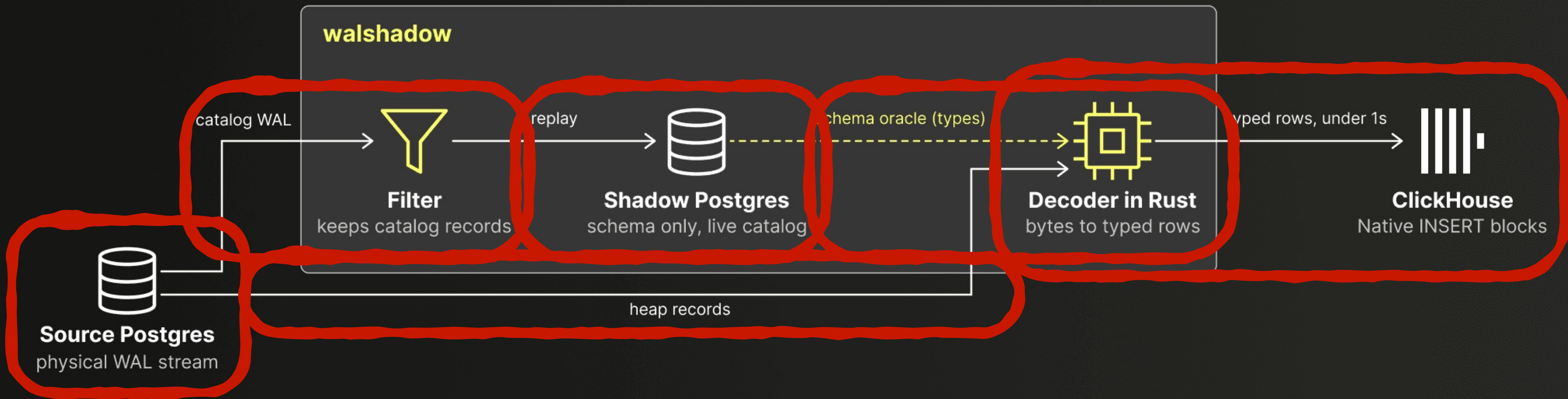
Drop Column

Restart recover

Rename Column

Planned switchovers

WalShadow Architecture



WalShadow Commit-to-visible latency

One linear scale; shorter is faster

Postgres

native replica floor



50 ms

Walshadow

near the replica floor



200 ms

PeerDB/ClickPipes

logical replication baseline



~10 s

About 50x lower latency than PeerDB, close to a raw Postgres replica.

WalShadow Throughput

End-to-end rows per second; longer is faster

Postgres insert rate

the source ceiling



290K rows/s

Walshadow

keeps pace with source



289K rows/s

PeerDB

about 40% of source



120K rows/s

“WalShadow takes a new approach to Postgres-to-ClickHouse replication, effectively turning ClickHouse into an analytical physical standby.”

WalShadow Demo

Open Source

All four projects open source

pg_stat_ch, pg_clickhouse, chdb: Apache 2

WalShadow: AGPL

Optimal UX on ClickHouse Cloud

Use them for your own databases

Contributing to Postgres

Contributed a few patches:

Make XactLockTableWait() and ConditionalXactLockTableWait() interruptable more

Fix integer-overflow problem in scram_SaltedPassword()

Allow ALTER COLUMN TYPE for columns in publication column lists

PGConf.eu: A Postgres Ecosystem Foundation?

Will continue to contribute to Postgres

Yes, We're Hiring

clickhouse.com/careers

Job openings

 postgres 

Senior Infrastructure Engineer - Postgres
United States • Engineering

Senior Infrastructure Engineer - Postgres
United States • Engineering

Principal Software Engineer - Postgres
AMER • Engineering

Principal Software Engineer - Postgres
AMER • Engineering

Principal Software Engineer - Postgres
AMER • Engineering

Senior Software Engineer - Postgres
APJ • Engineering

Thank You!

Postgres in da ClickHouse

David E. Wheeler
Principal Engineer



Postgres Summit
2026-09-30