

# The Duck(DB) Feather in your Parquet Cap

a talk by

**PREQUEL\_**

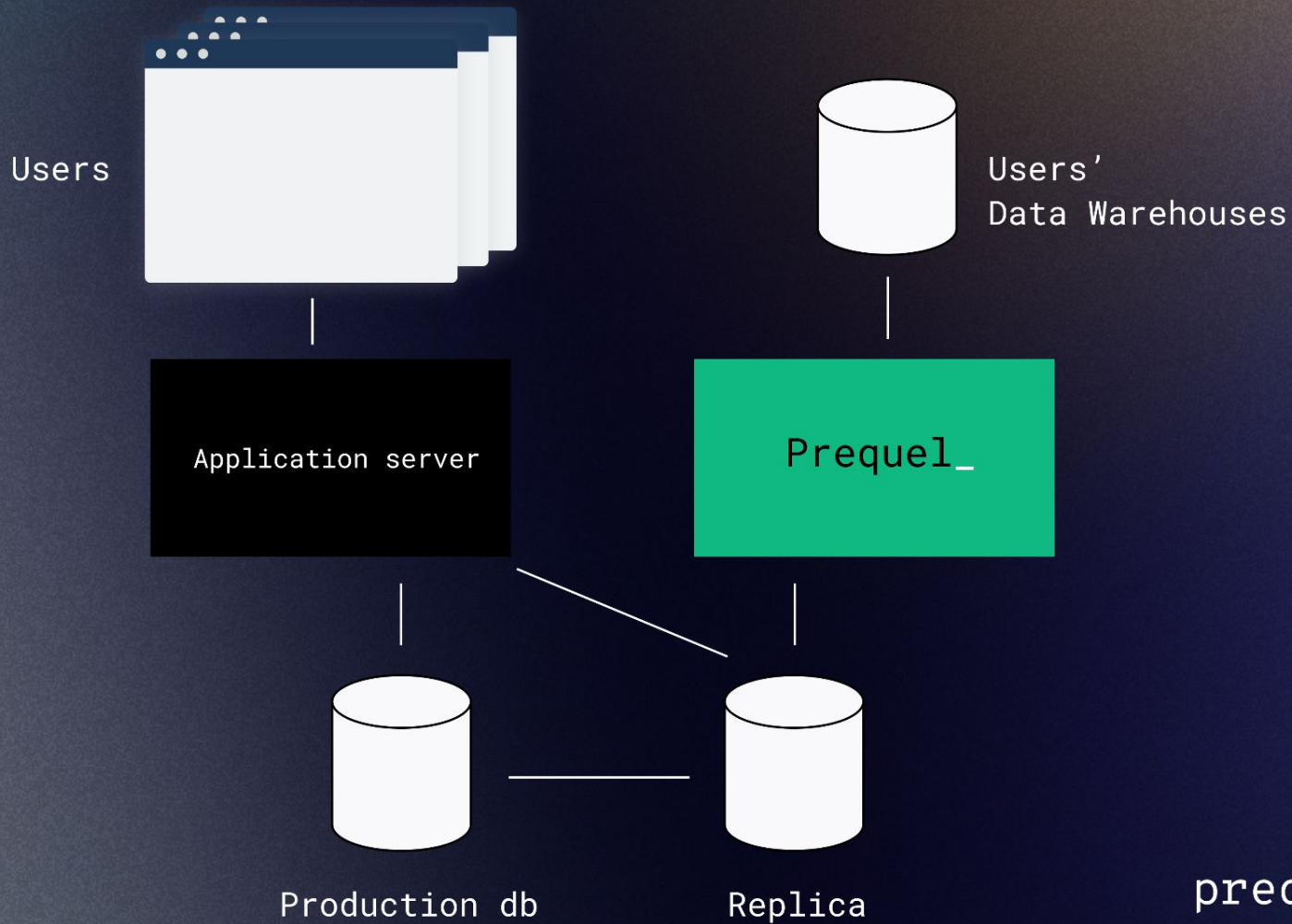
Enterprise-ready Data Sharing

[prequel.co](https://prequel.co)



**Niger Little-Poole**  
Lead Data Architect

PREQUEL\_



# Data delivery is a transportation planning problem



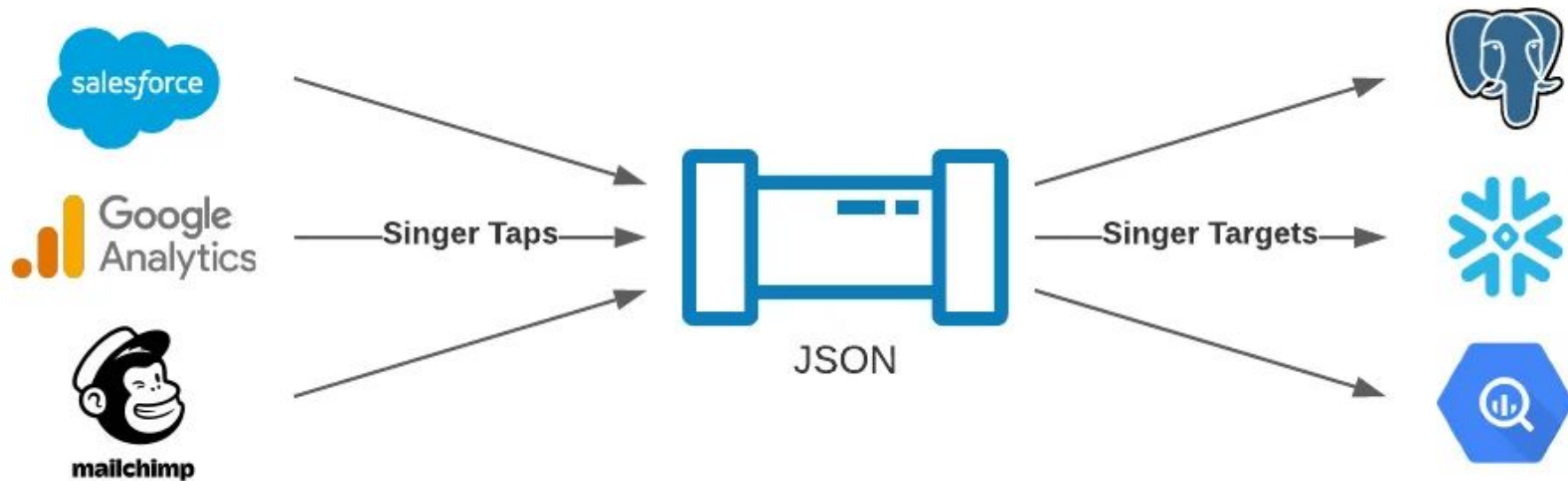


# Planner

- Every time we move a batch of data, we call that a “Transfer”
- Each transfer has a manifest
- Each manifest lists all the steps to execute the transfer
- Our planner generates the most efficient manifest, constrained by data integrity
- DuckDB is a core component of our planner

# Type Integrity

# Worse Case Data Movement



# Parquet is the Gold Standard

## Columnar Binary file format for relational data

Space efficient, fast for column operations, and fully structured.

## Robust Type support

Universal primitive types that can be easily extended via logical typing standards

## Many systems can read/write Parquet in Parallel

Most OLAP warehouses can load/unload Parquet files to object storage with higher throughput than via queries.



# XKCD 927

HOW STANDARDS PROLIFERATE:  
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)

SITUATION:  
THERE ARE  
14 COMPETING  
STANDARDS.

14?! RIDICULOUS!  
WE NEED TO DEVELOP  
ONE UNIVERSAL STANDARD  
THAT COVERS EVERYONE'S  
USE CASES.



SOON:

SITUATION:  
THERE ARE  
15 COMPETING  
STANDARDS.

## **Rule 927.b**

**“If a standard can vary, it will infinitely”**

# Our Planner has to account for type mismatches

## Examples of type variation between Parquet writers

1. **Strings:** encodings
2. **Boolean:** typed boolean? Bit? uint?
3. **Decimals:** byte arrays or integers? What size integers?
4. **Timestamps:** Strings? Seconds since epoch? Milliseconds since epoch? time zones?
5. **Complex objects:** arrays? maps ? structs?

# DuckDB to the Rescue

Enter ".help" for usage hints.

Connected to a transient in-memory database.

Use ".open FILENAME" to reopen on a persistent database.

D `SELECT * FROM parquet_metadata('~/.Downloads/mock.parquet') LIMIT 10;`

type	file_name	stats_min	row_group_id	row_group_num_rows	row_group_num_columns	row_group_bytes	column_id	file_offset	num_values	path_in_schema	stat
s_max_value	compression	encodings	index_page_offset	dictionary_page_offset	data_page_offset	total_compressed_size	total_uncompressed_size				
BYTE_ARRAY	/Users/nlp/Downloads/mock.parquet	0	0	1000	11	0	0	0	1000	id	
00411460f7c92d2124a67ea0f4cb5f85	SNAPPY	PLAIN	0	0	0	4	32800	00411460f7c92d2124a67ea0f4cb5f85	0	ffeabd223de0	
d4eacb9a3e6e53e5448d	0	0	0	1000	11	0	1	0	1000	updated_at	
1970-01-01 00:00:00	SNAPPY	PLAIN	0	0	0	32804	6072	1970-01-01 00:00:00	0	1970-01-01 0	
0:16:39	0	0	0	1000	11	0	2	0	1000	test_smallint	
INT32	0	999	0	0	0	38876	4030	0	0	999	
SNAPPY	PLAIN	0	0	1000	11	0	3	0	1000	test_integer	
INT32	0	999	0	0	0	42906	4030	0	0	999	
SNAPPY	PLAIN	0	0	1000	11	0	4	0	1000	test_bigint	
INT64	0	999	0	0	0	46936	4788	0	0	999	
SNAPPY	PLAIN	0	0	1000	11	0	5	0	1000	test_real	
FLOAT	0.0	320.4968	0	0	0	51724	4032	0.0	0	320.4968	
SNAPPY	PLAIN	0	0	1000	11	0	6	0	1000	test_double	
DOUBLE	0.0	986.7618700491175	0	0	0	55756	8028	0.0	0	986.76187004	
91175	SNAPPY	PLAIN	0	0	0						

# Demo

**Clickhouse:** [Parquet Documentation](#)

**DuckDB:** [Environment](#)

**Gist:** [Some Example Queries](#)

# Why DuckDB?

## **Parquet.**

Not only metadata scanning, but also normalization.

## **SQL.**

SQL is our preferred interface for data manipulation

## **Embedded.**

Allows our planner to be a library, not a service





# Questions