



THE RISE OF POSTGRES AS THE 'EVERYTHING' DATABASE

November 2025

Safe Harbor and Disclaimers

Other than statements of historical fact, all information contained in these materials and any accompanying oral commentary (collectively, the “Materials”), including statements regarding (i) Snowflake’s business strategy, plans or priorities, (ii) Snowflake’s new or enhanced products, services, and technology offerings, including those that are under development or not generally available, (iii) market growth, trends, and competitive considerations, (iv) our vision for Snowpark, the Data Cloud, and industry-specific Data Clouds, including the expected benefits and network effects of the Data Cloud; and (v) the integration, interoperability, and availability of Snowflake’s products, services, or technology offerings with or on third-party platforms or products, are forward-looking statements. These forward-looking statements are subject to a number of risks, uncertainties and assumptions, including those described under the heading “Risk Factors” and elsewhere in the Annual Reports on Form 10-K and the Quarterly Reports on Form 10-Q that Snowflake files with the Securities and Exchange Commission. In light of these risks, uncertainties, and assumptions, the future events and trends discussed in the Materials may not occur, and actual results could differ materially and adversely from those anticipated or implied in the forward-looking statements. As a result, you should not rely on any forwarding-looking statements as predictions of future events. Forward-looking statements speak only as of the date the statements are first made and are based on information available to us at the time those statements are made and/or management’s good faith belief as of that time. Except as required by law, we undertake no obligation, and do not intend, to update the forward-looking statements in these Materials.

Any future product or roadmap information (collectively, the “Roadmap”) is intended to outline general product direction. The Roadmap is not a commitment, promise, or legal obligation for Snowflake to deliver any future products, features, or functionality; and is not intended to be, and shall not be deemed to be, incorporated into any contract. The actual timing of any product, feature, or functionality that is ultimately made available may be different from what is presented in the Roadmap. The Roadmap information should not be used when making a purchasing decision. In case of conflict between the information contained in the Materials and official Snowflake documentation, official Snowflake documentation should take precedence over these Materials. Further, note that Snowflake has made no determination as to whether separate fees will be charged for any future products, features, and/or functionality which may ultimately be made available. Snowflake may, in its own discretion, choose to charge separate fees for the delivery of any future products, features, and/or functionality which are ultimately made available.

The Materials may contain information provided by third-parties. Snowflake has not independently verified this information, and usage of this information does not mean or imply that Snowflake has adopted this information as its own or independently verified its accuracy.



Introducing Today's Speakers



Claire Peracchio
Product Marketing Lead,
Snowflake



Brett Goulder
Senior Software Engineer,
Snowflake



Sho Tanaka
Lead Developer Advocate,
Snowflake



THE COMPLEXITY REBELLION



The Modern Data Stack is Evolving



THEN

The specialized stack



New problem?

(e.g. search, geospatial...)



Let's add **new database**.



NOW

The unified database



New problem?

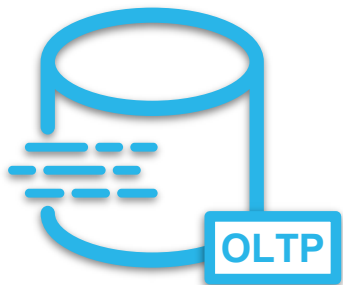
(e.g. search, geospatial...)



Let's use the **same database**.






For Years, Developers Drowned in Complexity



**TRANSACTIONAL
DATABASE**

The ETL Tax

-  Fragile data pipelines
-  Recurring data sync issues
-  Multiple systems to troubleshoot



**ANALYTICAL
DATABASE**

I need a cache...



I need search...



I need streaming...



I need vector search...



Which Interrupted Your Work This Week?

Investigating

- ✓ Which database has the data I need?
- ✓ What's the data architecture?
- ✓ How do I get security approval for this data?

Moving Data

- ✓ Writing code to join data
- ✓ Waiting for an ETL pipeline to catch up
- ✓ Debugging data sync errors

Testing

- ✓ Schema changes across databases
- ✓ Identifying performance and latency issues
- ✓ Can't test locally since I don't have all the databases running



Fragmented Systems Come at a Cost

74%

Of companies are
using **more than 1**
database platform

Source: [Redgate, 2025](#)

12 hrs

Developers lose half
a day every week
chasing data

Source: [Airtable, 2022](#)

1 in 3

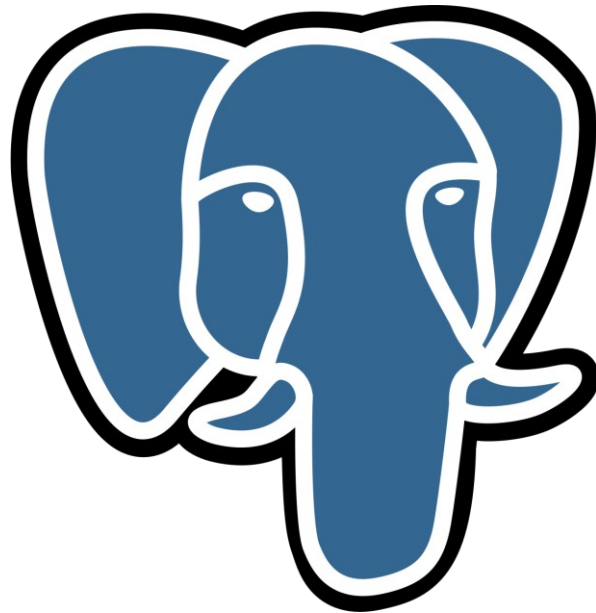
Developers say
complex tech stacks
are their **#1 frustration**

Source: [Stack Overflow, 2024](#)



Now, Developers are Breaking Free with Postgres

- ★ **The #1 Most-Loved Database:** 58% of developers say it's their favorite. Postgres isn't just respected: It's loved.
- ✓ **40-Year Trajectory:** It's a battle-tested database that constantly reinvents itself.
- **Powerful Extensibility:** Extensions allow developers to constantly add new capabilities.



They're Asking, "Why Not Just... Postgres?"

Instead of a separate...	Postgres has the answer
Vector database	pgvector for AI. Your AI agent's memory and system of record.
Search engine (like Elasticsearch)	Full text search and pg_trgm for fast, "fuzzy" string matching.
Document DB (like MongoDB)	Native JSON/JSONB , which is fully indexable and transactional.
Key-value store (like Redis)	hstore (key-value storage) for flexible, high-performance data.
Geospatial DB	PostGIS to store, manage, and query geographic data.
Message queue (like Kafka)	LISTEN/NOTIFY for native, real-time pub/sub and eventing.



Real-World Example: How Instacart Simplified Search

Before

Specialized systems

Struggled with billions of daily search updates.

Systems included:

- **Postgres**: For core transactions.
- **Elasticsearch** : For text search.
- **FAISS Service**: For AI/semantic search.

The Pain

The Pain

- ❌ Fragmented
- ❌ Slow
- ❌ Stale results

After

Unified Postgres

All workloads run smoothly on Postgres.

Postgres now handles:

- **Transactions**: Billions of daily updates.
- **Text Search**: Using built-in **ts_rank**.
- **AI & Semantic Search**: Using **pgvector**.

The Gain

- ✅ Streamlined
- ✅ Fast
- ✅ Relevant results



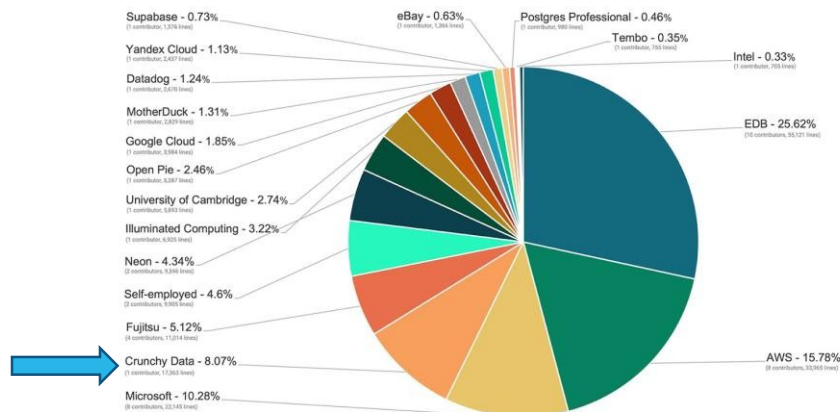
At Snowflake, We're Committed to Postgres



crunchy data

Snowflake acquired Postgres provider
Crunchy Data in June 2025.

The following companies, through their team members, contributed 90% of the new lines of code to Postgres 17.



We've brought on the team, including
leading Postgres contributors.



Bringing Postgres to the Snowflake Platform



Simplify your entire data footprint

Consolidate on one powerful platform for all your data workloads.



Connect transactional data effortlessly

Eliminate slow, costly data pipelines to gain agility and innovate faster.



Power mission-critical apps and AI

Get the proven performance, reliability, and security you need to operate at scale.



ENGINEERING SIMPLICITY WITH POSTGRES



Evolving Postgres: Our Strategy

- ▶ Our goal: **Give Postgres the power to work with your data lakehouse...** without sacrificing its transactional strengths.
- ▶ Our approach: **Augment Postgres, don't replace it.**

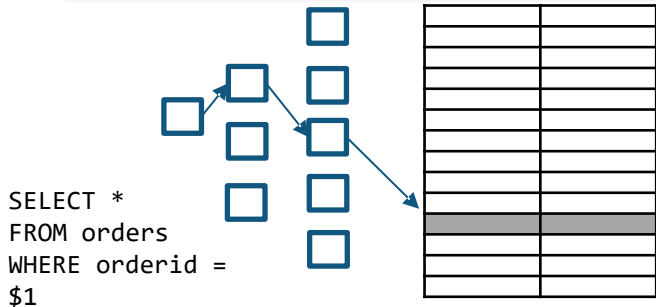


OLTP and OLAP: Where Postgres Hits Its Limits

OLTP: Operational system of record

SQL

- High query rate, small queries
- Low response time
- User-facing applications
- Mission-critical, always on

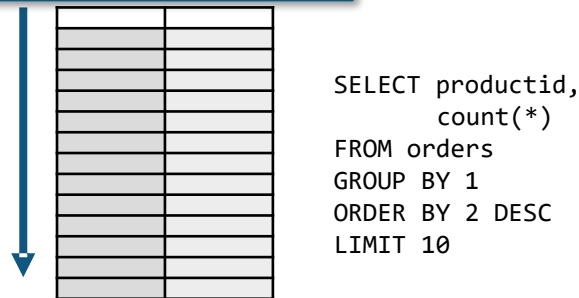


At scale: Fast on OLTP, Slow on OLAP

OLAP: Analytics on a collection of data

SQL

- Low query rate, big queries
- High scan throughput
- Business-facing dashboards
- On demand, business hours



At scale: Slow on OLTP, Fast on OLAP



Why FDWs Aren't Enough

— Unaware of layout

Treat foreign systems like data lakes as black boxes. E.g. planner doesn't know which S3 files contain data for `WHERE date = 'today'`

— Massive over-fetching

It must pull terabytes of data across the network and filter it locally in Postgres.

— Performance killer

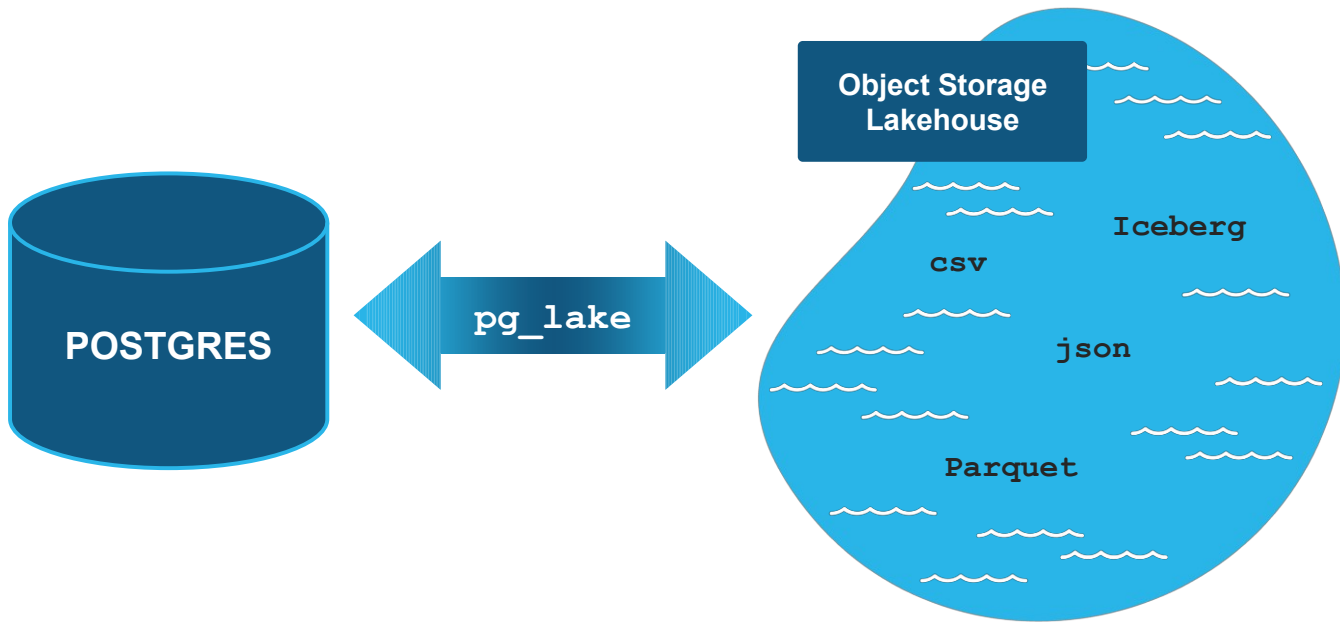
This approach is incredibly slow and expensive.



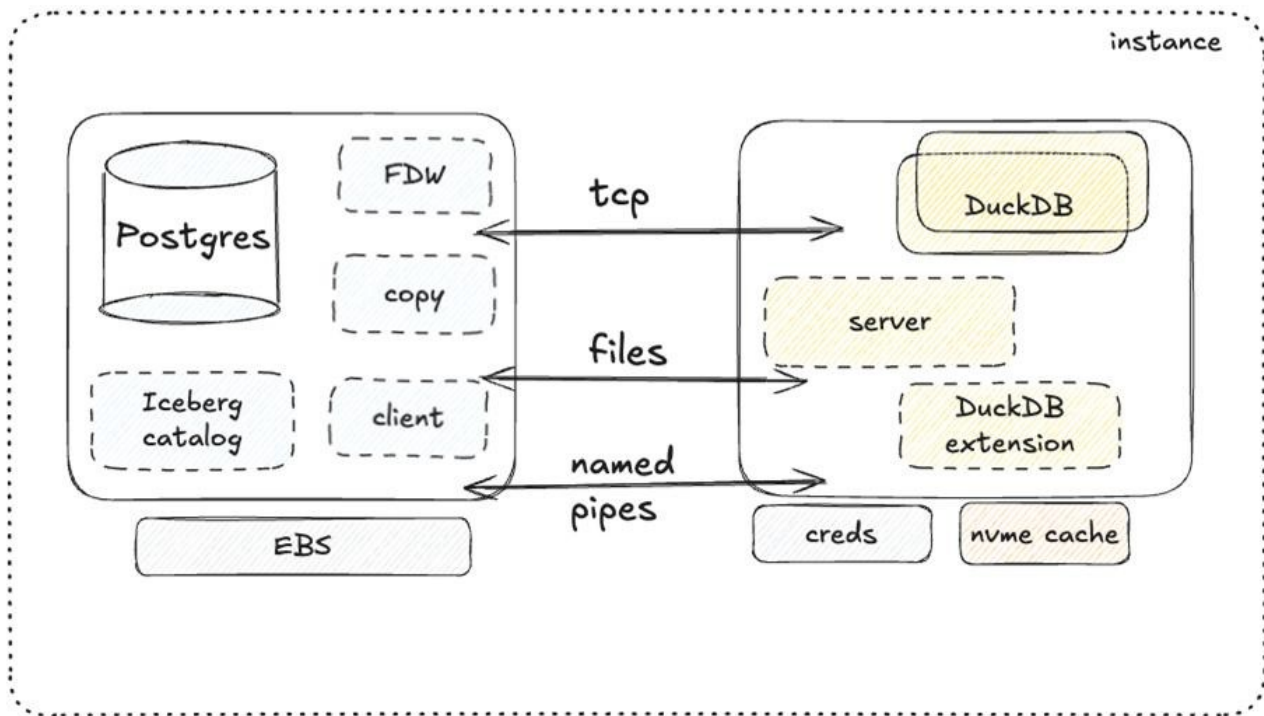
Our Answer: Connect Postgres to Your Lakehouse with **pg_lake**



**OPEN
SOURCE**



pg_lake Core Architecture



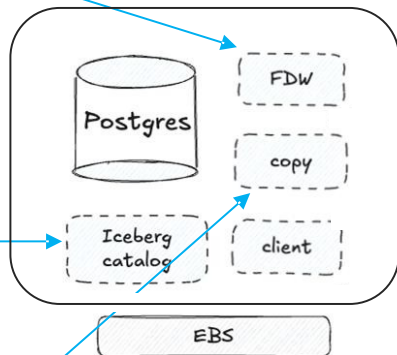
pg_lake Core Architecture: The Brain

Full SQL on Iceberg and other data lake tables using FDW API with planner hooks

External tools can read and write to Iceberg via transactions on catalog

Easy import and export to and from Parquet, CSV, and JSON

The Brain



Linux user: postgres

Postgres Acts as the Control Plane

- **Single source of truth:** Connects with standard SQL tools (psql, JDBC). Handles all security and SQL parsing.
- **Atomic Iceberg catalog:** Manages Iceberg metadata directly in Postgres tables.
`CREATE TABLE ... USING iceberg`
- **Operates as the brain:** Decides where to route queries based on workload type.

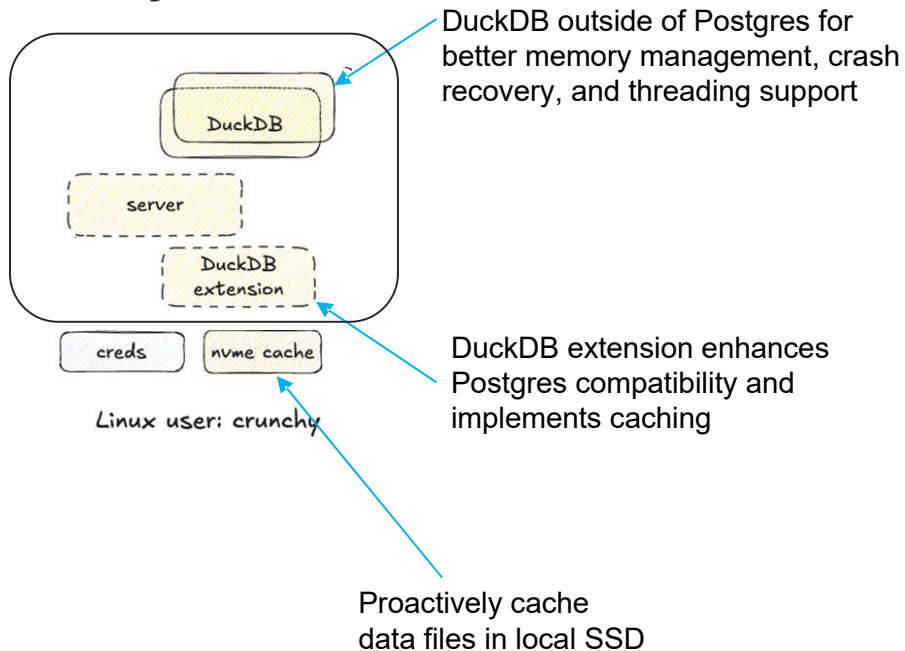


pg_lake Core Architecture: The Analytical Muscle

Running DuckDB for Analytics

- **Dedicated data plane:** A separate server process built purely for speed.
- **Vectorized execution:** Scans terabytes of data lake files with extreme efficiency.
- **Workload isolation:** Runs outside Postgres so a heavy analytical query won't make it crash.

The Analytical Muscle



Breaking Down Data Silos with Postgres

Key Benefit #1

Query data in place

- ! **The benefit:** Break down silos. Query CSV, JSON, and Parquet data where it lives without deciding on a schema, writing ETL, or loading it into Postgres first.
- ✓ **How it works:** Point Postgres directly at files. Pg_lake uses DuckDB to inspect files and infer columns automatically. Join external files, Iceberg tables, and regular Postgres queries in one table.

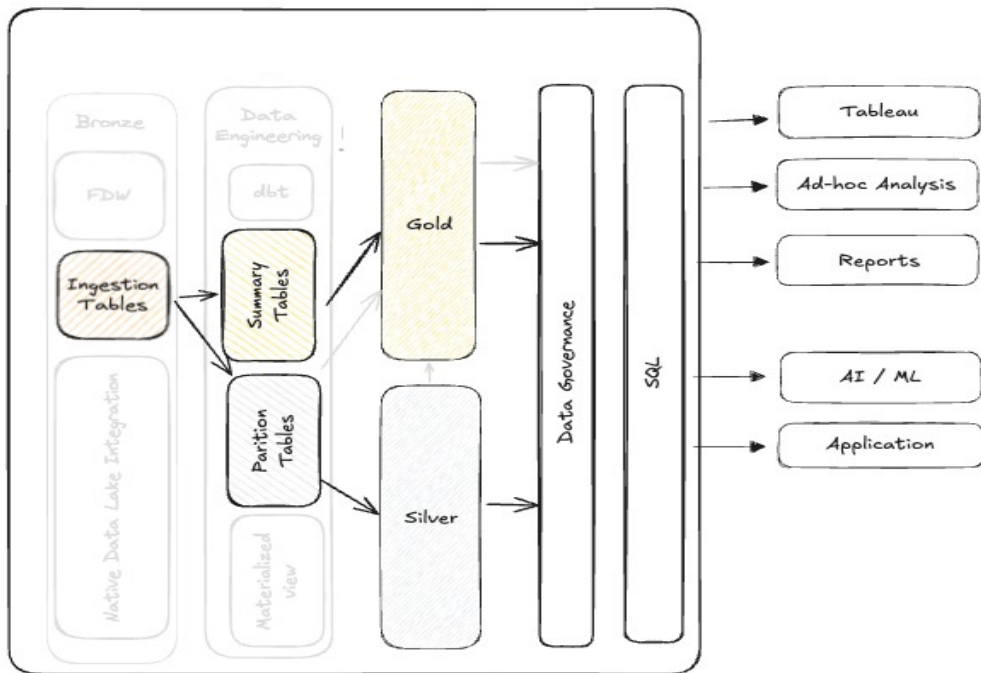
Key Benefit #2

Simplify Data Pipelines

- ! **The benefit** Collapse the complex “modern data stack” (ingestion, warehouse, transformation) into a single Postgres-centric architecture. Eliminate fragile synchronization between systems.
- ✓ **How it works:** Bronze, Silver, and Gold layers live in one logical system. Run SQL transformations across layers with Postgres transactions. No separate ingestion or warehouse engines needed.



Powering Real-Time Analytics Inside Postgres



New use cases pg_lake enables

- Power user-facing apps and dashboards with live data.
- Join streaming and static data in a single SQL interface.
- Visualize massive datasets instantly using your favorite tools.

Using the medallion architecture New use cases pg_lake enables

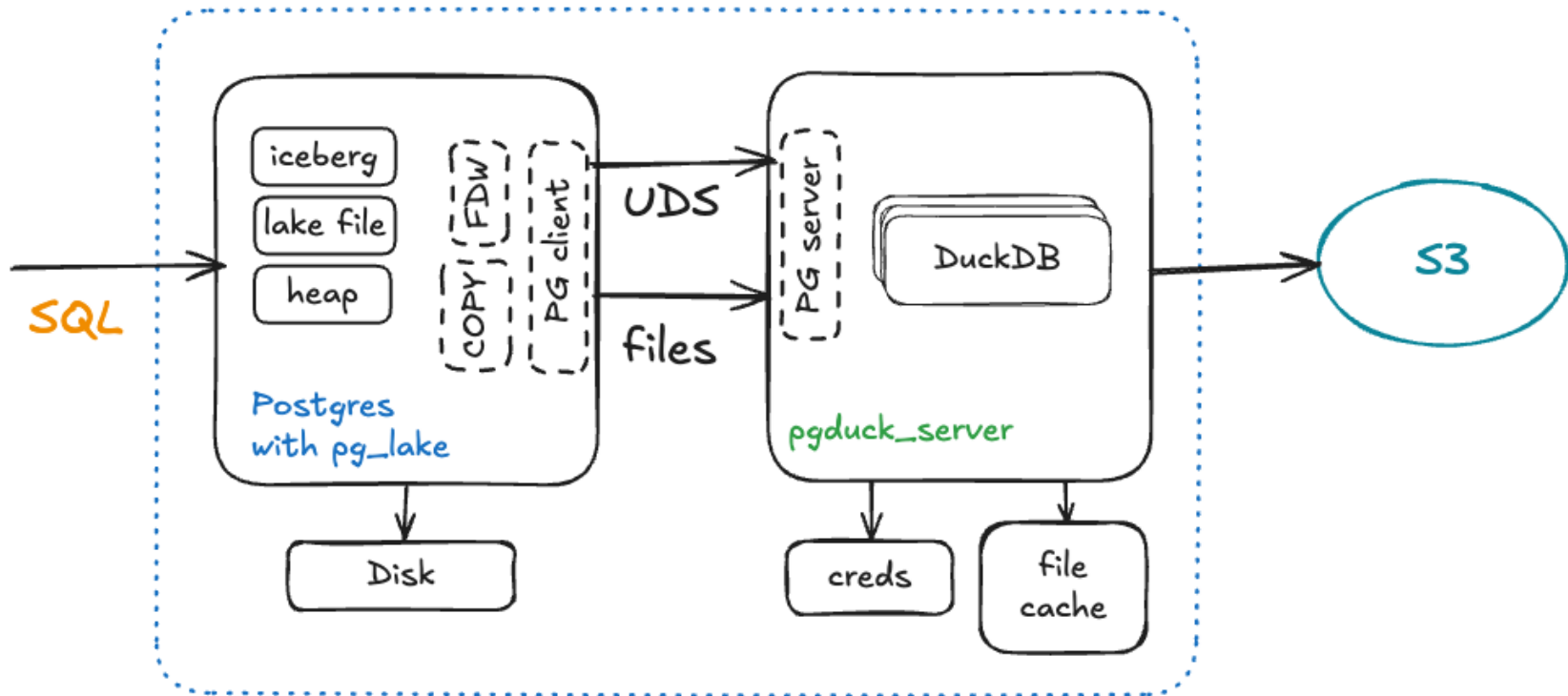
- **Bronze (Ingest):** Query raw files instantly.
- **Silver (Refine):** Clean and structure data incrementally with SQL.
- **Gold (Serve):** Materialize aggregates to ensure sub-second performance for reporting.



PG_LAKE IN ACTION



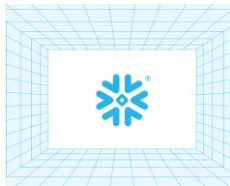
Demo 1



Demo 2



Learn More About What We're Doing with Postgres



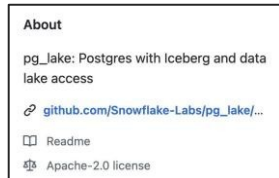
Introducing pg_lake

Read why [we open sourced pg_lake](#).



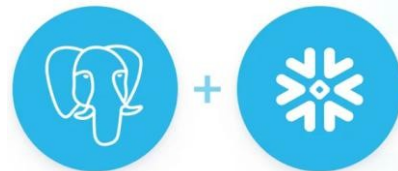
pg_lake Demo

Watch a [step-by-step demo](#).



GitHub Repo for pg_lake

Check out the pg_lake [GitHub repo](#).



Snowflake Postgres

See more about [Snowflake Postgres](#).



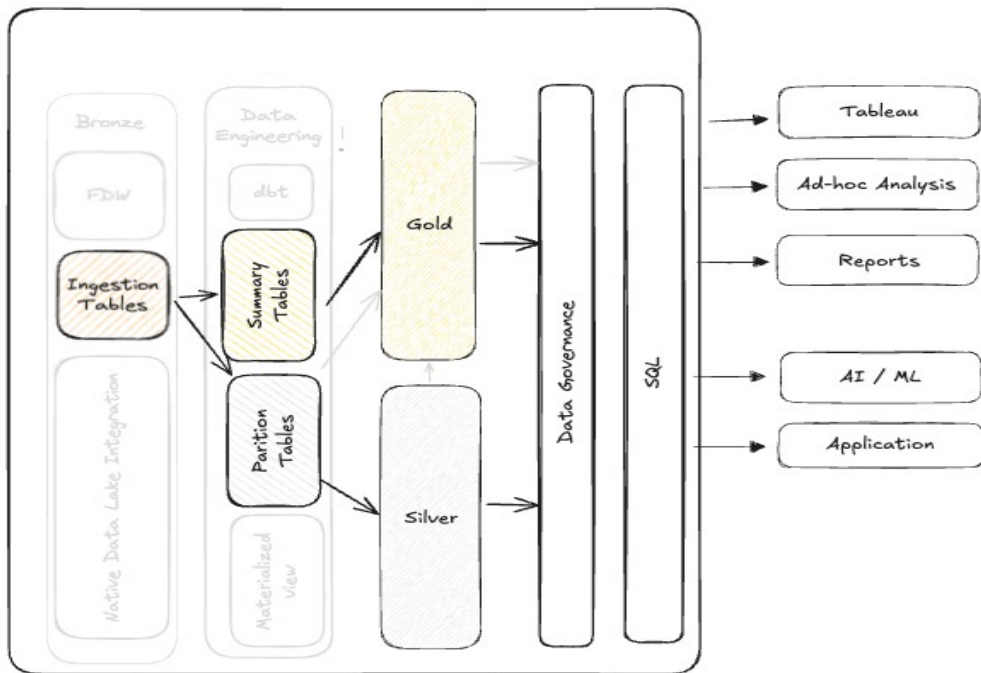
► **The future is
unified and open**



THANK YOU



Powering Real-Time Analytics with pg_lake



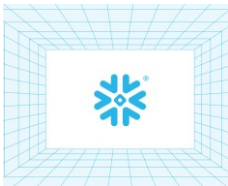
- User facing dashboards and apps with real-time responsiveness
- Up to date data to power your business
- Efficient SQL on top of large datasets
- Integrate both streaming and static data
- Visualize with your favorite tools - Grafana, Tableau, Superset

Medallion architecture

- Bronze: Raw ingest of up to millions of records per second
- Silver: In database incremental processing to clean data
- Gold: Aggregated data for efficient querying, tailored for business insights and reporting



Learn More About What We're Doing with Postgres



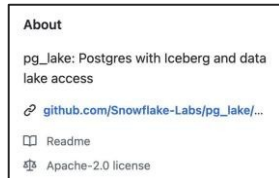
Introducing pg_lake

Read why [we open sourced pg_lake](#).



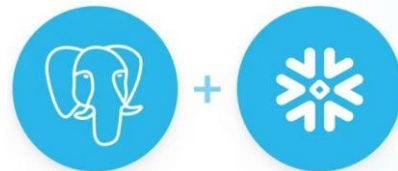
pg_lake Demo

Watch a [step-by-step demonstration](#).



GitHub Repo for pg_lake

Check out the pg_lake [GitHub repo](#).



Snowflake Postgres

See more about [Snowflake Postgres](#).

