Lecture #21

Carnegie Mellon University

ADVANCED DATABASE SYSTEMS

Snowflake

Andy Pavlo // 15-721 // Spring 2023

ADMINISTRIVIA

Project #2:

→ Final Submission: **Monday May 1st**

→ I will send feedback later this week.

Project #3

→ Final Presentations: **Friday May 5th @ 5:30pm**

DuckDB Guest Lecture (*In-Class*)

→ Wednesday April 19th @ 2:00pm ET

Amazon Redshift Guest Lecture (*In-Class*)

→ Wednesday April 26th @ 2:00pm ET

HISTORICAL CONTEXT

The 2000s saw the rise of several special-purpose relational OLAP engines.
→ Vertica, Greenplum, MonetDB, Vectorwise, ParAccel

There many organizations trying to use SQL on top of Hadoop/HDFS in the early 2010s.
→ Hive, Presto, Impala, Stinger

All these systems were self-managed / on-prem…

HISTORICAL CONTEXT

Google's Dremel paper came out in 2011.

Facebook started building Presto in 2012.

Amazon licensed ParAccel in 2011 and released in on AWS as Redshift in 2013.

SutterHill VCs recruited two Oracle engineers (Dageville, Cruanes) and Vectorwise co-founder (Żukowski) to build Snowflake in 2012.

SNOWFLAKE

Managed OLAP DBMS written in C++.

→ Shared-disk architecture with aggressive compute-side local caching.

→ Written from scratch. Did not borrow components from existing systems.

→ Custom SQL dialect and client-server network protocols.

Disclaimer: Snowflake sponsored this course in Spring 2018. You can watch the guest lecture!

THE SNOWFLAKE ELASTIC DATA WAREHOUSE
SIGMOD 2016

SNOWFLAKE

Shared-Disk / Disaggregated Storage

Push-based Vectorized Query Processing

Precompiled Primitives

Separate Table Data from Meta-Data

No Buffer Pool

PAX Columnar Storage
→ Supports both proprietary + open-source formats

Sort-Merge(?) + Hash Joins

Unified Query Optimizer + Adaptive Optimizations

SNOWFLAKE: ARCHITECTURE

Data Storage: Cloud-hosted object store
→ Amazon S3, MSFT Azure Store, Google Cloud Storage

Virtual Warehouses: Worker Nodes
→ VM instances running Snowflake software with locally attached disks for caching.
→ Customer specifies the compute capacity.
→ Added support for serverless deployments in 2022 (?).

Cloud Services: Coordinator/Scheduler/Catalog
→ Transactional key-value store (FoundationDB)

SNOWFLAKE: EXECUTION ARCHITECTURE

Worker Node (e.g., EC2 Instance)

→ Maintains a local cache of files + columns that previous Worker Processes have retrieved from storage.
→ Simple LRU replacement policy.
→ Optimizer assigns individual table files to worker nodes based on consistent hashing. This ensures that files are only cached in one location.

Worker Process (e.g., Unix Process)

→ Spawned for the duration of a query.
→ Can push intermediate results to other Worker Processes or write to storage.

SNOWFLAKE: VECTORIZED QUERY PROCESSING

Snowflake is a push-based vectorized engine that uses precompiled primitives for operator kernels.

→ Pre-compile variants using C++ templates for different vector data types.

→ Only uses codegen (via LLVM) for tuple serialization/deserialization between workers.

Does not rely on shuffle step between stages

→ Worker processes push data to each other.

Does not support partial query retries

→ If a worker fails, then the entire query has to restart.

SNOWFLAKE: WORK STEALING

Optimizer determines which files workers will retrieve for processing a query before execution.

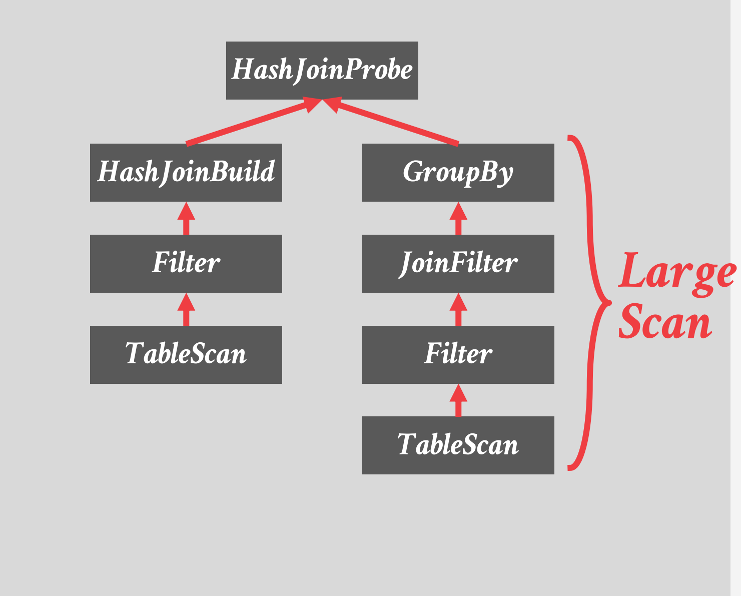
When a worker process completes scanning its input files, it can request from peer worker processes that it scans their files for them.

The requestor always downloads from storage instead of the peer to avoid additional burden.

SNOWFLAKE: FLEXIBLE COMPUTE

If a query plan fragment will process a large amount of data, then the DBMS can temporarily deploy additional worker nodes to accelerate its performance.

Flexible compute worker nodes write results to storage as if it was a table.

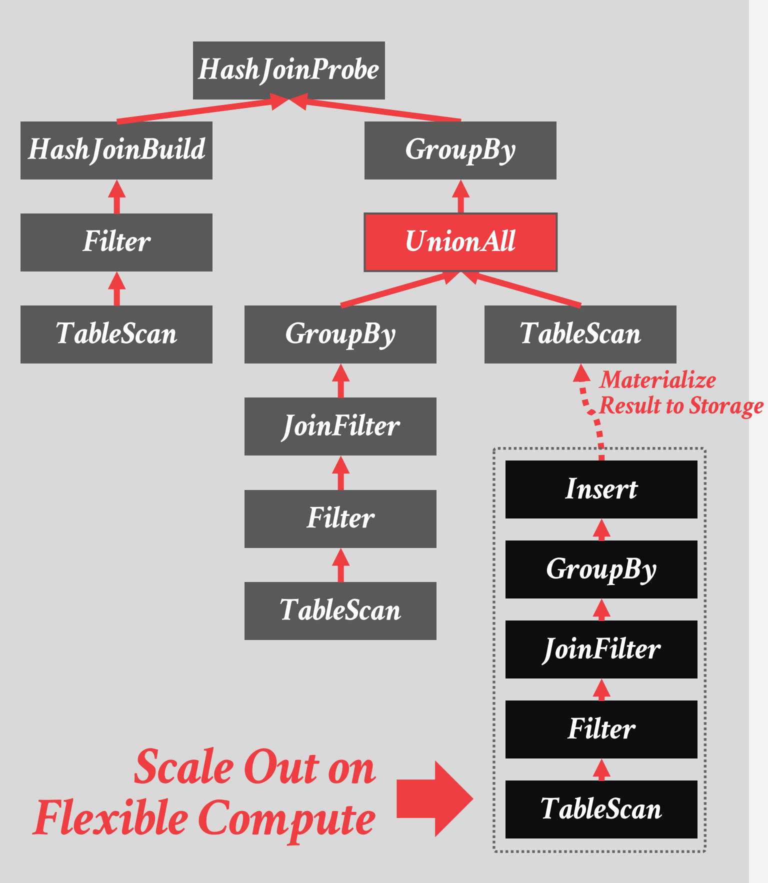


Source: Libo Wang

SNOWFLAKE: FLEXIBLE COMPUTE

If a query plan fragment will process a large amount of data, then the DBMS can temporarily deploy additional worker nodes to accelerate its performance.

Flexible compute worker nodes write results to storage as if it was a table.



Source: Libo Wang

SNOWFLAKE: DATA STORAGE

Cloud object storage (AWS S3) is slower than local disk. And each I/O has higher CPU overhead because of HTTPS API calls.

But cloud storage supports fetching offsets from files. This allows the DBMS to fetch headers and then determine what portions of a file it needs.

Snowflake decided to instead invest heavily on building its own caching layer to hide latencies.

BUILDING AN ELASTIC QUERY ENGINE ON DISAGGREGATED STORAGE
NSDI 2020

SNOWFLAKE: STORAGE FORMAT

Snowflakes (mostly) stores all tables in their internal columnar format by breaking them up into **micropartition** files.

→ Immutable files using PAX storage format
→ Original data for each micropartition is 50-500MB but these get compressed down to ~16MB per file

Snowflake automatically clusters and re-arranges micropartitions in the background based on query access patterns.

SNOWFLAKE: STORAGE FORMAT

Snowflake provides custom data types to store semi-structured data.

- → **VARIANT**, **ARRAY**, **OBJECT** types.

Instead of determining data types of JSON/XML fields during reads, the DBMS automatically infers format and breaks them out into binary columns.

- → Example: Convert string "2023-04-17" into 4-byte **DATE**.
- → Always keep the original unparsed data in case the inference is incorrect.

SNOWFLAKE: CONSISTENT HASHING

DBMS uses consistent hashing to map micropartition files to worker nodes.

→ The mapping is transactional so that all workers are in sync on which node is responsible for which files.

→ Ensures query fragments (tasks) that access the same micropartition are assigned to same worker nodes.

Allows Snowflake to add new compute nodes without changing micropartition assignments

→ Avoid having to wipe all locally cached files.

BUILDING AN ELASTIC QUERY ENGINE
ON DISAGGREGATED STORAGE
NSDI 2020

SNOWFLAKE: QUERY OPTIMIZER

Unified Cascades-style top-down optimization.

→ Snowflake refers to their optimizer as the "compiler".

Optimizer checks catalog to identify what micropartitions it can prune / skip before the query starts executing.

→ Determining how many micropartitions a pipeline will access helps determine the complexity of the query.

DBMS also supports query plan hints and runtime adaptivity.

Source: Jiaqi Yan

SNOWFLAKE: STATISTICS COLLECTION

DBMS maintains statistics for data store in Snowflake's proprietary table format.

→ Only simple zone maps. No histograms/sketches.

→ Statistics are in sync with data when using internal file format (micropartitions).

Table + Micropartitions:

→ # of rows, size in bytes with compression information

Columns:

→ Min/Max, Null/Distinct counts

Source: Jiaqi Yan

SNOWFLAKE: PRUNING

Optimizer uses statistics to determine what micropartitions to skip.

→ Statistics are cached locally to ensure fast evaluation during optimization.

Supports evaluating complex expressions during pruning pass.

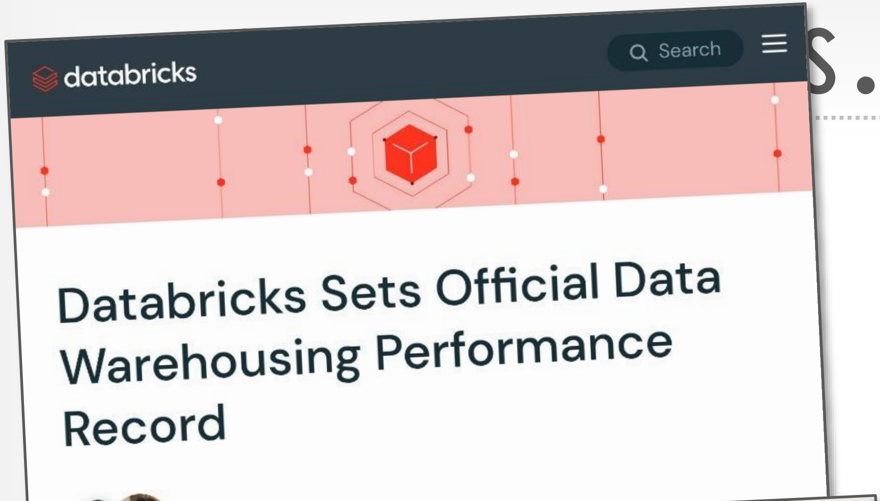
→ Requires specialized expression evaluators that operate on zone map information.

→ Also need to consider null indicators.

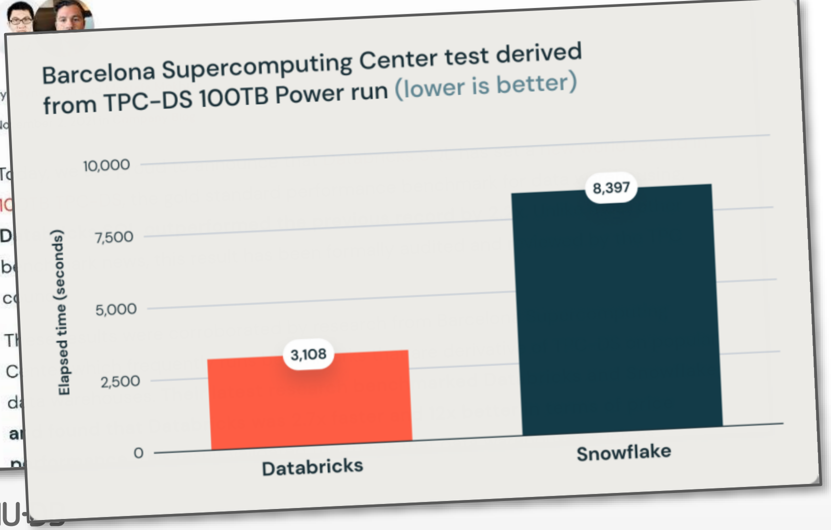
```
SELECT * FROM xxx
 WHERE col1 + col2 > 1234;
```

```
SELECT * FROM xxx WHERE
 cdate BETWEEN '2023-01-01'
           AND '2023-12-31';
```

S. SNOWFLAKE (2021)

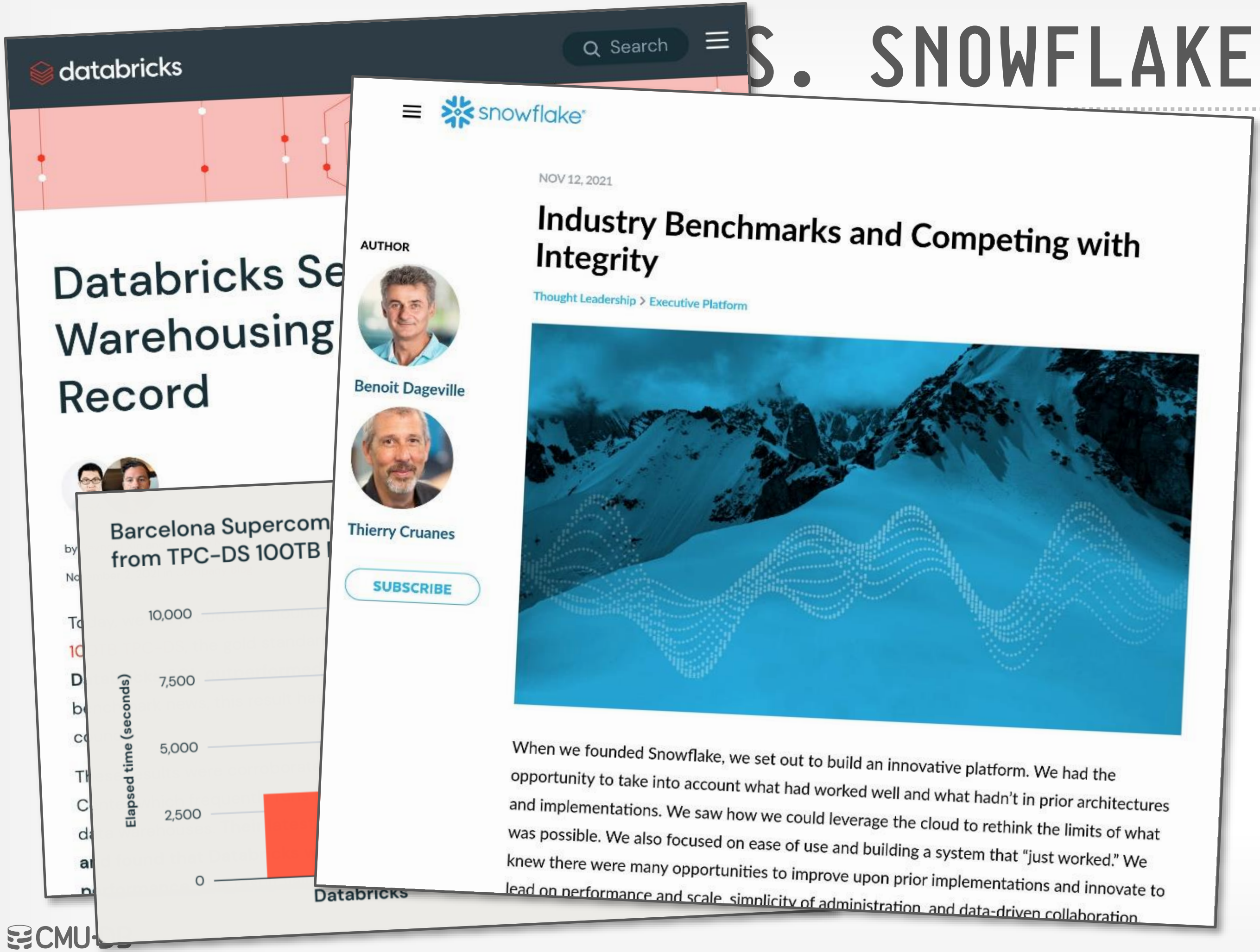


Databricks Sets Official Data Warehousing Performance Record



Barcelona Supercomputing Center test derived from TPC-DS 100TB Power run (lower is better)

S. SNOWFLAKE (2021)



Databricks Se
Warehousing
Record

NOV 12, 2021

Industry Benchmarks and Competing with Integrity

AUTHOR

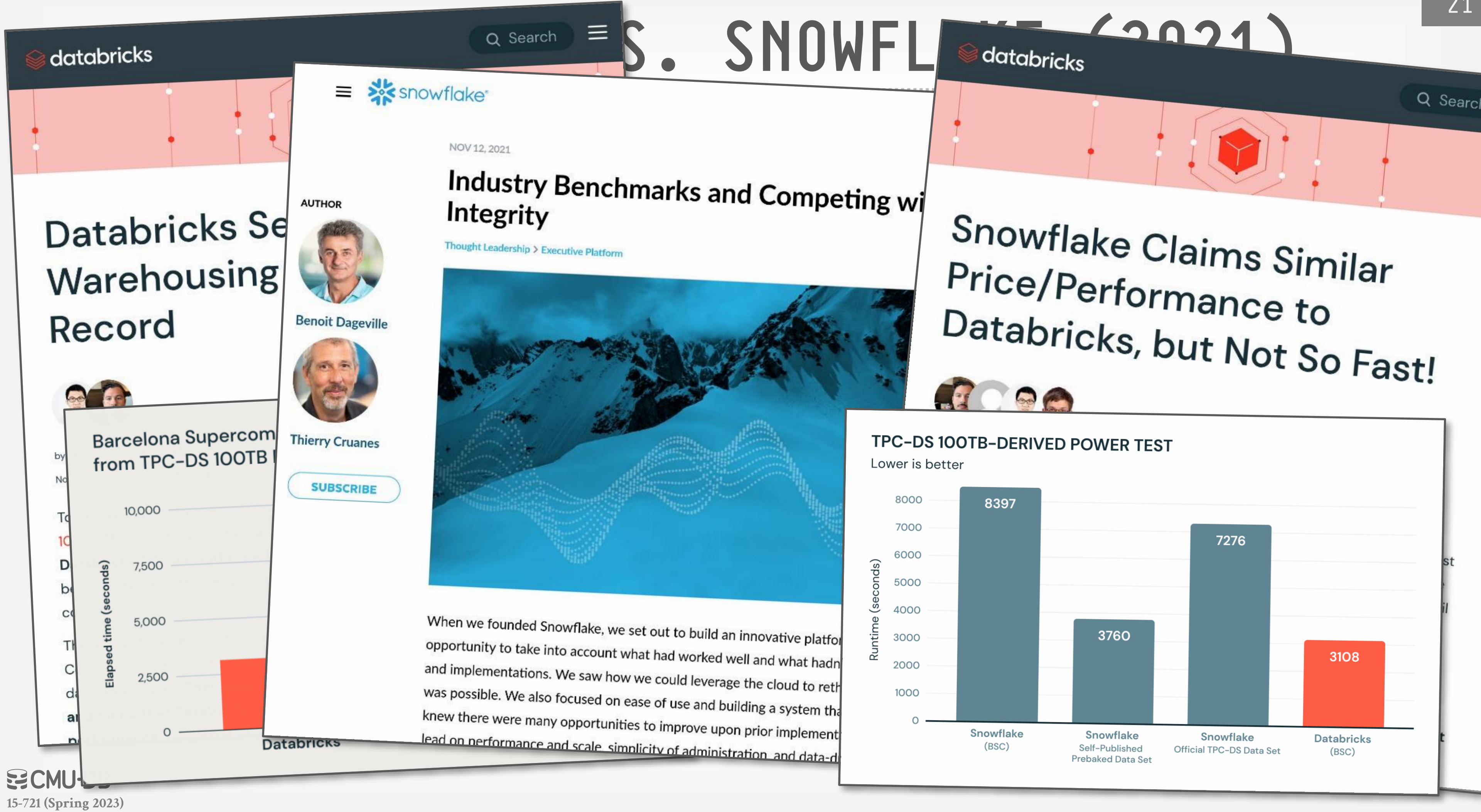
Benoit Dageville

Thierry Cruanes

Thought Leadership > Executive Platform

When we founded Snowflake, we set out to build an innovative platform. We had the opportunity to take into account what had worked well and what hadn't in prior architectures and implementations. We saw how we could leverage the cloud to rethink the limits of what was possible. We also focused on ease of use and building a system that "just worked." We knew there were many opportunities to improve upon prior implementations and innovate to lead on performance and scale, simplicity of administration, and data-driven collaboration

S. SNOWFL (2021)



databricks

Search

Databricks Se
Warehousing
Record

by

No

Barcelona Supercom
from TPC-DS 100TB

snowflake

NOV 12, 2021

AUTHOR

Benoit Dageville

Thierry Cruanes

SUBSCRIBE

Industry Benchmarks and Competing wi Integrity

Thought Leadership > Executive Platform

When we founded Snowflake, we set out to build an innovative platfo opportunity to take into account what had worked well and what hadn and implementations. We saw how we could leverage the cloud to reth was possible. We also focused on ease of use and building a system tha knew there were many opportunities to improve upon prior implement lead on performance and scale, simplicity of administration, and data-d

databricks

Search

Snowflake Claims Similar Price/Performance to Databricks, but Not So Fast!

TPC-DS 100TB-DERIVED POWER TEST

Lower is better

OBSERVATION

Like Dremel and Databricks, Snowflake has the problem that the DBMS does not have statistics if data files are created outside of the DBMS.

→ Snowflake originally required users to load all data files into the DBMS before they can be queried.

Snowflake expanded its architecture to support additional methods for ingesting data.

→ Snowpipe (via Apache Arrow)

→ External Tables (2019)

→ Hybrid Tables (2022)

APACHE ICEBERG (2017)

Infrastructure and file format extension to Parquet for maintaining catalog about data files in an object store.

→ Keeps track of partitioning, versioning, and schema changes.

→ Provides catalog service for runtime lookups and pruning of meta-data.

Snowflake added support for ingesting, creating, and querying Iceberg files in 2021.

SNOWFLAKE HYBRID TABLES (2022)

New service called Unistore to support OLTP workloads directly in Snowflake

→ Customer declares a table as "hybrid" (row + columnar)
→ Write updates to row-based storage with strong transactional guarantees.
→ Background jobs merge them into micropartition files.

OLAP queries retrieve data from row-based and columnar storage and then merges the results.

FOUNDATIONDB

Transactional key-value store used by Snowflake for its catalog service early in its design.

When Apple bought FoundationDB in 2015, Snowflake maintained their own fork.

Apple then open-sourced FoundationDB in 2018 and works closely with Snowflake dev team.

PARTING THOUGHTS

Snowflake created the roadmap on how to build a scalable cloud-based OLAP DBMS as a service.

Andy still considers it a state-of-the-art system but there is a lot of things about how it is implemented that is not public.

NEXT CLASS

DuckDB Guest Lecture