# WHY SNOWFLAKE?

## Concepts and Snowflake Fundamentals

# Agenda

Massively Parallel Processing (MPP) for “Big Data” and historical limitations

Snowflake Innovation$_1$ – separation of Storage and Compute

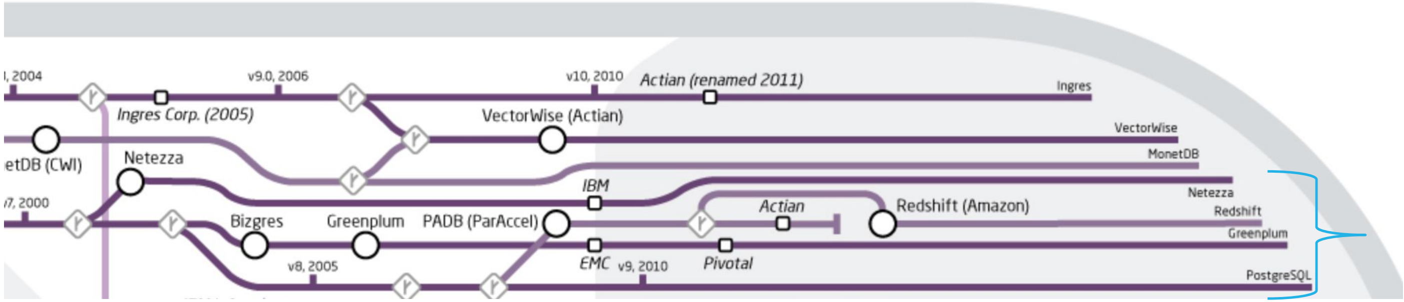
The “secret sauce”$_2$ – micro-partitions, global cloud services

Some feature differentiators enabled by 1 and 2

The Snowflake AI Data Cloud High Level Architecture

Visualization using dashboards, Streamlit in Snowflake

# MPP Architecture Evolution - PostgreSQL genealogy




https://hpi.de/naumann/projects/rdbms-genealogy.html

PostgreSQL descendants:

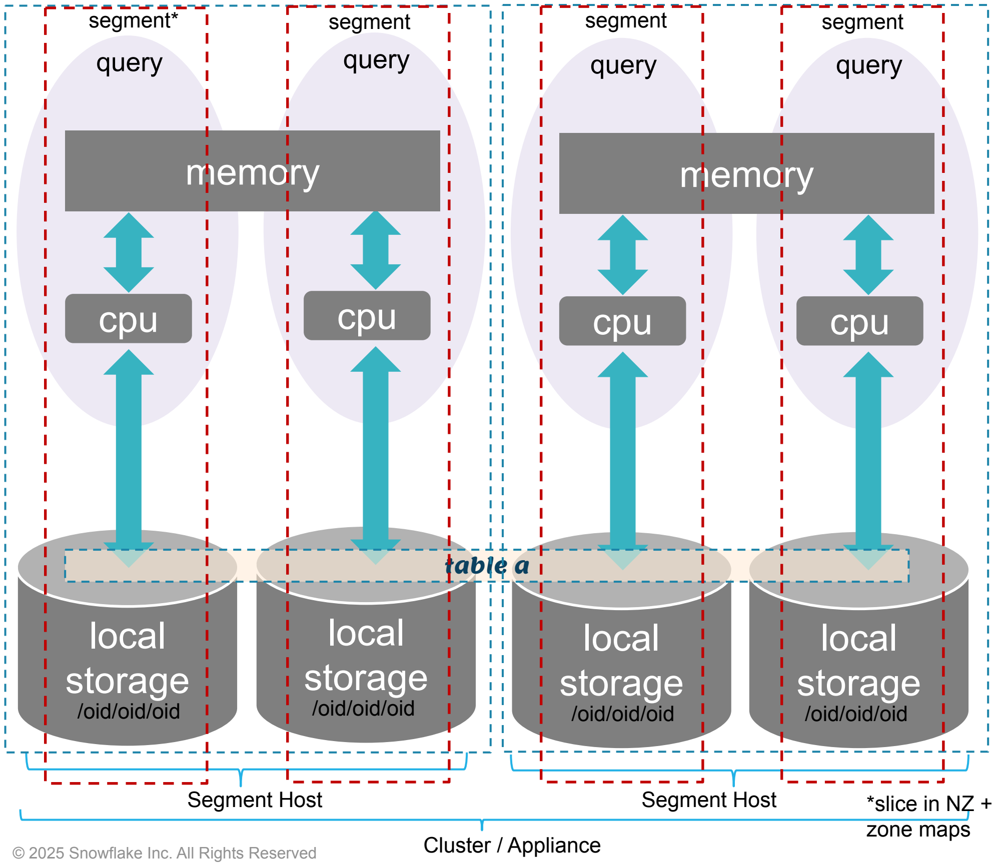
Netezza – forked PostgreSQL 7.2

Greenplum – forked PostgreSQL v8.2

Redshift (as ParAccel) – forked PostgreSQL v8.0.2

*How to build an MPP database from Postgres?*

# MPP Architecture of Greenplum / Netezza




Excellent performance on large table joins… as long as joins are "local"

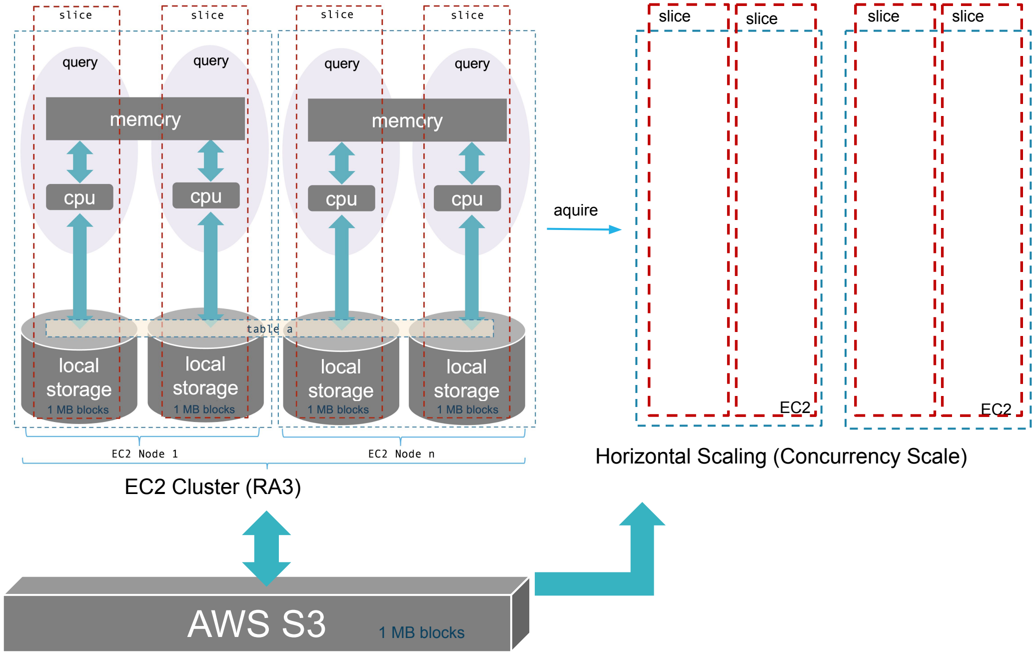
Tightly-coupled Compute and Storage = "secondary" segments enforce data redundancy/durability

Skew! Data/static and processing/dynamic

Vacuum that bloat (table, catalog)

Maintenance Overhead

# MPP Architecture of Redshift




No “idle” secondary slices but degree of parallelism fixed during cluster configuration

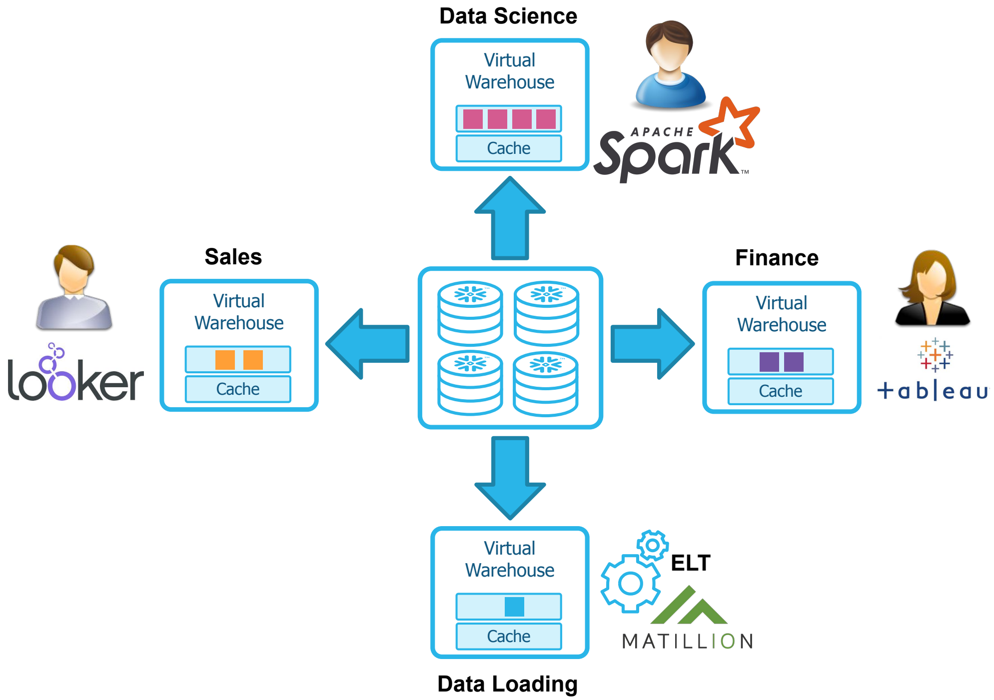
Tightly-coupled Compute and Storage = horizontal slow, vertical scaling worse (elastic vs classic resize)

Skew remains… artifact of data distribution

Redshift Spectrum “extends” Postgres-derived engine but is also dated

Complex Workload Management

# Snowflake separates Storage and Compute




No need to determine and commit size of compute cluster up-front, use t-shirt sized *virtual warehouses*

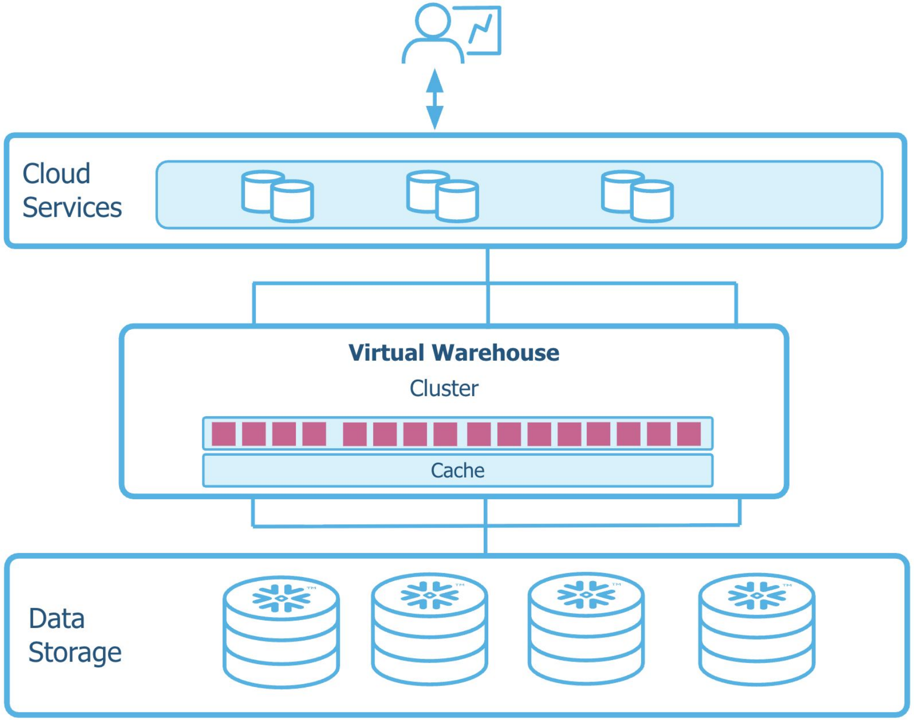
Need more compute power? Resize your virtual warehouse on the fly

Need more concurrency? Configure multi-cluster warehouse

Native support for open table formats

Simple maintenance, it just works!

# Scale “up” for performance



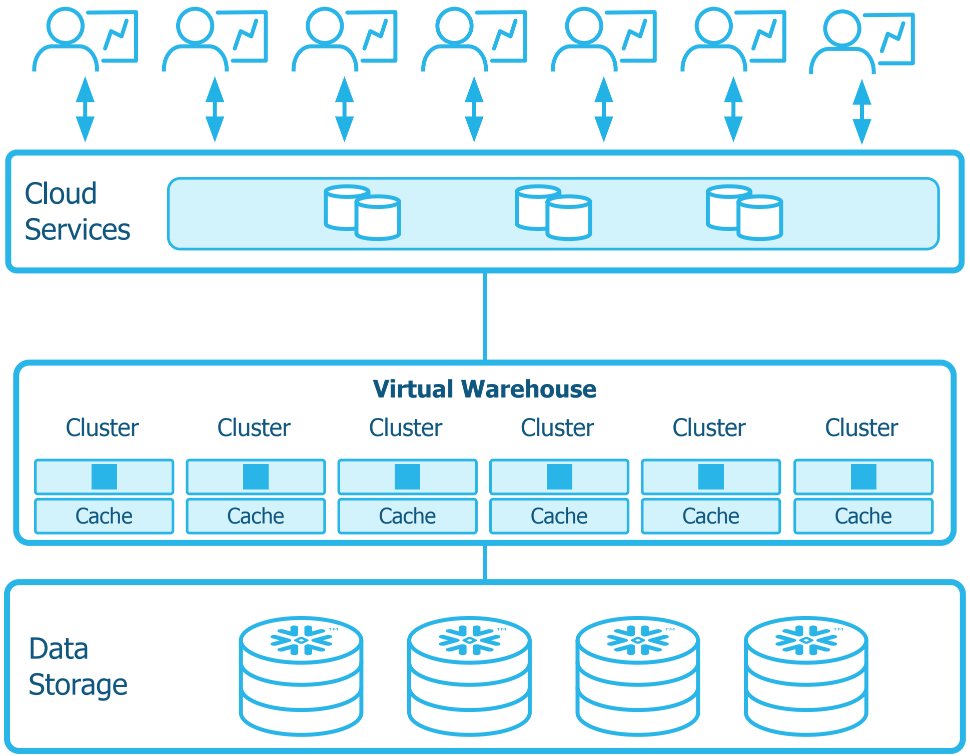

```
alter warehouse PROD_VWH set
    warehouse_size = ‘XLARGE’;
```

**Scale Up**

- X-Small – 1 Node
- X-Large – 16 Nodes

**Small VWH is suspended when all queries complete.**

# Scale “out” for concurrency



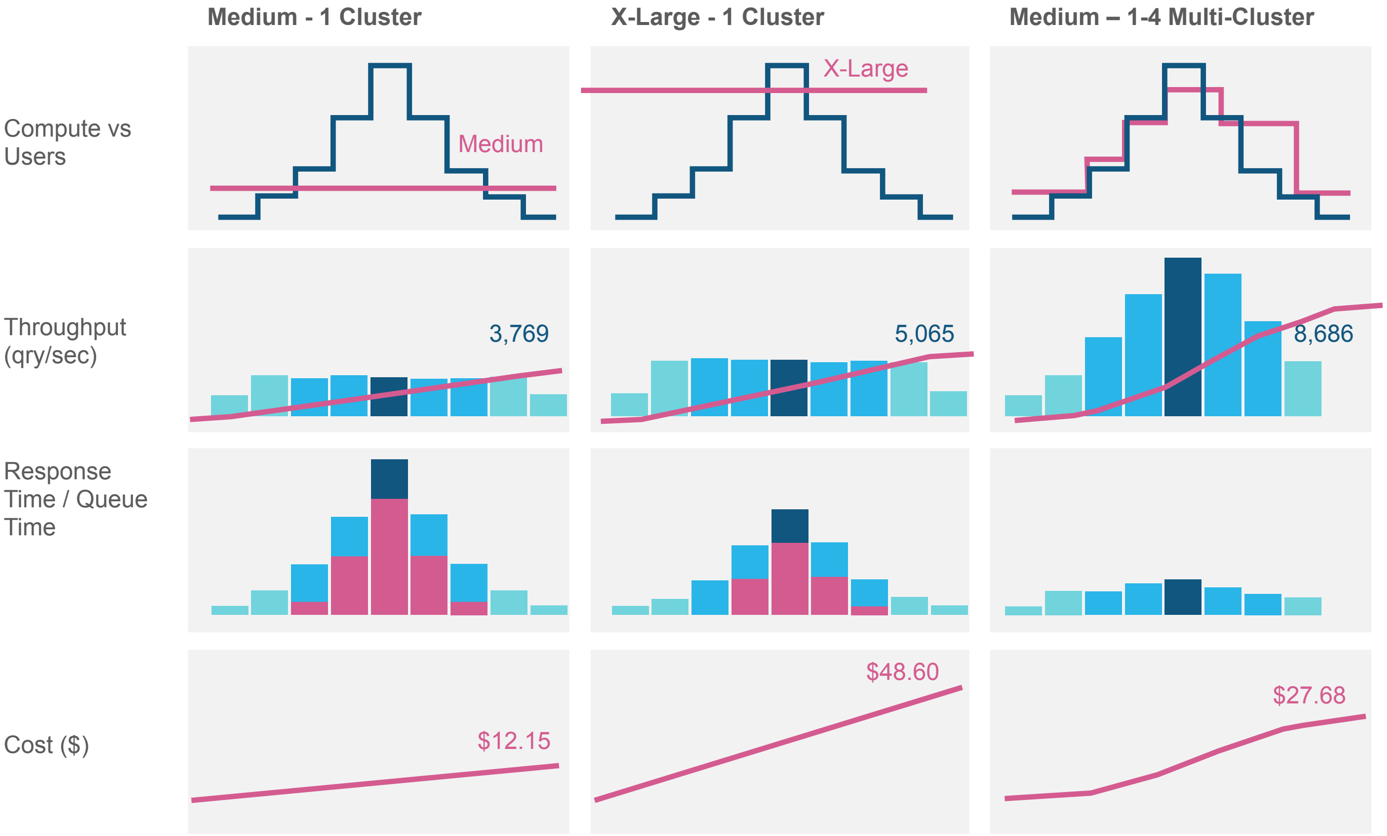

```
ALTER WAREHOUSE prod_wh SET
   MIN_CLUSTER_COUNT = 1
   MAX_CLUSTER_COUNT = 5;

Time Elapsed: 68ms
```

**Automatically Scale Out**

1 – 10 <u>same size </u>Clusters

# Scale out optimization pattern



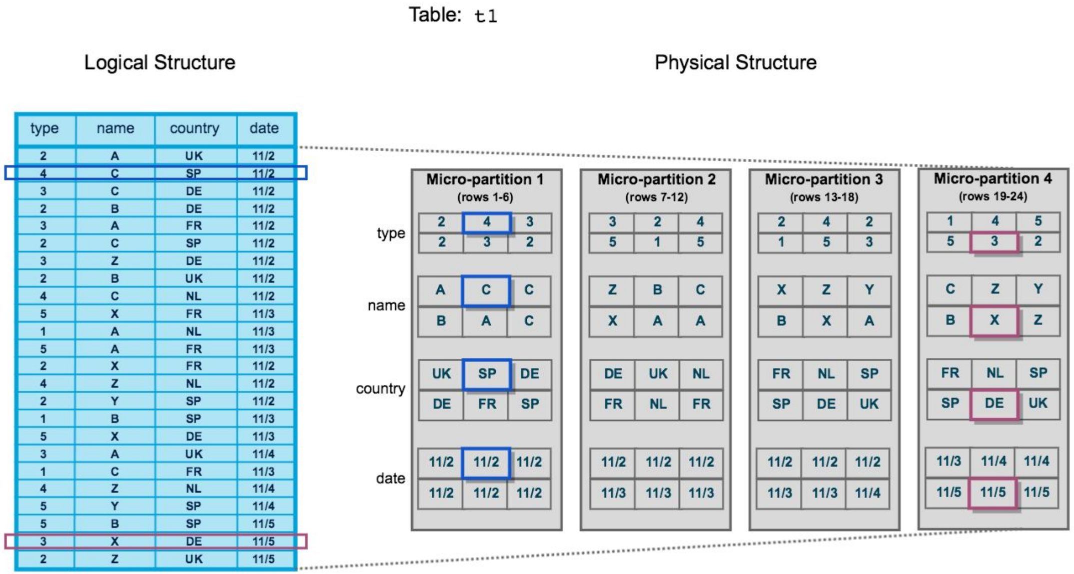

Not over or under provisioning – match resources to demand

Provides greatest throughput

Provides best response time through peak

Provides the best bang for buck

# Secret sauce: Micro-partitions




Contains 50 MB and 500 MB of uncompressed data

Groups of rows in tables are mapped into individual micro-partitions, organized in a columnar fashion

Unlike traditional static partitioning, Snowflake micro-partitions are derived automatically

Size and structure allows for extremely granular pruning of very large tables

Micropartitions are immutable

Snowflake stores metadata about all rows stored in a micro-partition, including

- The range of values for each of the columns in the micro-partition.
- The number of distinct values.
- Additional properties used for both optimization and efficient query processing.

# Micro-partitions – academic papers

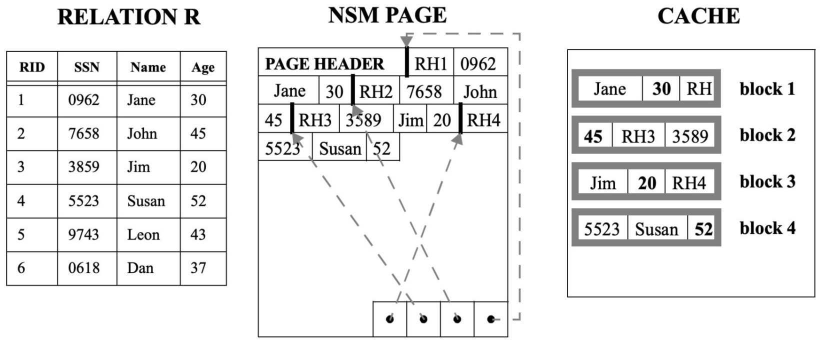



**FIGURE 1: The N-ary Storage Model (NSM) and its cache behavior.** *Records in R (left) are stored contiguously into disk pages (middle), with offsets to their starts stored in slots at the end of the page. While scanning* age, *NSM typically incurs one cache miss per record and brings useless data into the cache (right).*

### 2.1 The N-ary Storage Model

Traditionally, the records of a relation are stored in slotted disk pages [29] obeying an n-ary storage model (NSM). NSM stores records sequentially on data pages.

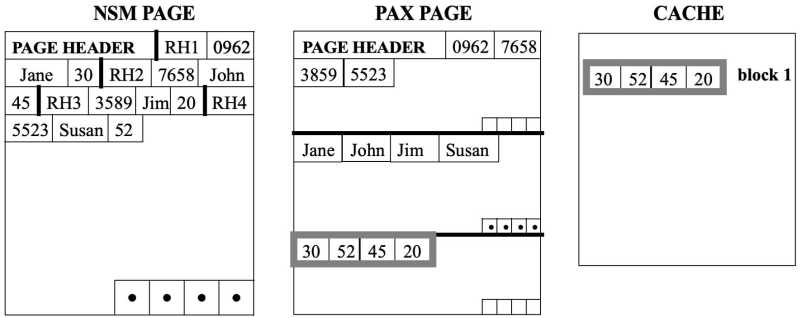



**FIGURE 3: Partition Attributes Across (PAX), and its cache behavior.** *PAX partitions records into minipages within each page. As we scan R to read attribute* age,*values are much more efficiently mapped onto cache blocks, and the cache space is now fully utilized.*

https://www.pdl.cmu.edu/PDL-FTP/Database/pax.pdf
https://www.vldb.org/conf/2001/P169.pdf

# Micro-partitions

| Date | Product | Customer | Amount |
|---|---|---|---|
| Feb 14 | Boots | Frank | $150 |
| Feb 14 | Boots | Benoit | $150 |
| Feb 14 | Skis | Thierry | $300 |
| Feb 14 | Snowboard | Mike | $250 |
| Feb 15 | Boots | Chris D | $150 |
| Feb 15 | Skis | Denise | $600 |
| Feb 15 | Snowboard | Shelly | $250 |
| Feb 16 | Boots | Rob | $150 |
| Feb 16 | Skis | Sunny | $600 |
| Feb 16 | Snowboard | Chris K | $250 |
| Feb 16 | Snowboard | Greg | $750 |
| Feb 16 | Snowboard | Matt | $750 |

## **Profile Overview** (Finished)

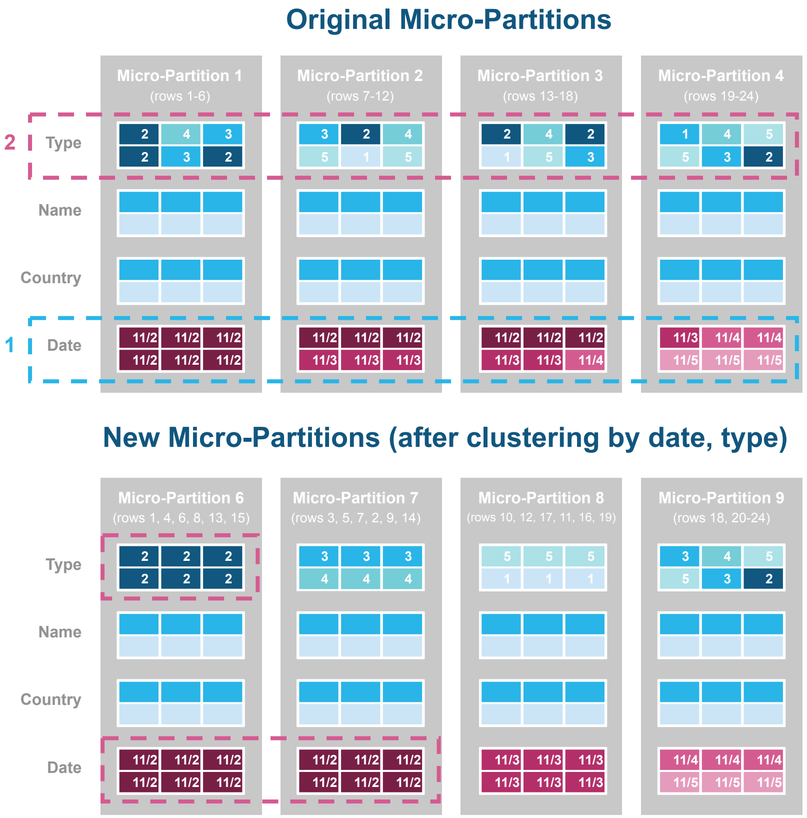
(100%)

- Processing 63%
- Local Disk IO 3%
- Synchronization 3%
- Initialization 31%

| | |
|---|---|
| Scan progress | 100.00% |
| Bytes scanned | 179.58 MB |
| Percentage scanned from cache | 100.00% |
| Bytes written to result | 25.33 MB |

| | |
|---|---|
| Partitions scanned | 13 |
| Partitions total | 73 |

# Auto Clustering




## What is it?

Reorganize table data to align with query patterns

## Value

Process only the relevant data from large tables

Faster queries and fewer compute credits

No blocking / impact on DML

## Cost

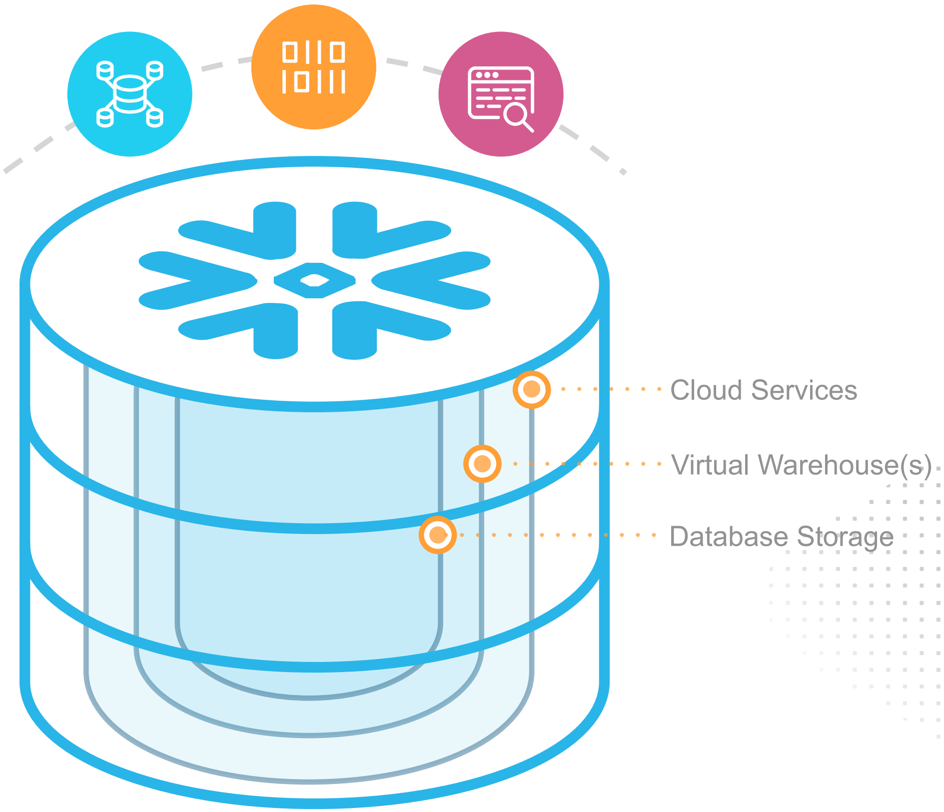
Credits for background clustering maintenance

# Adaptive Caching

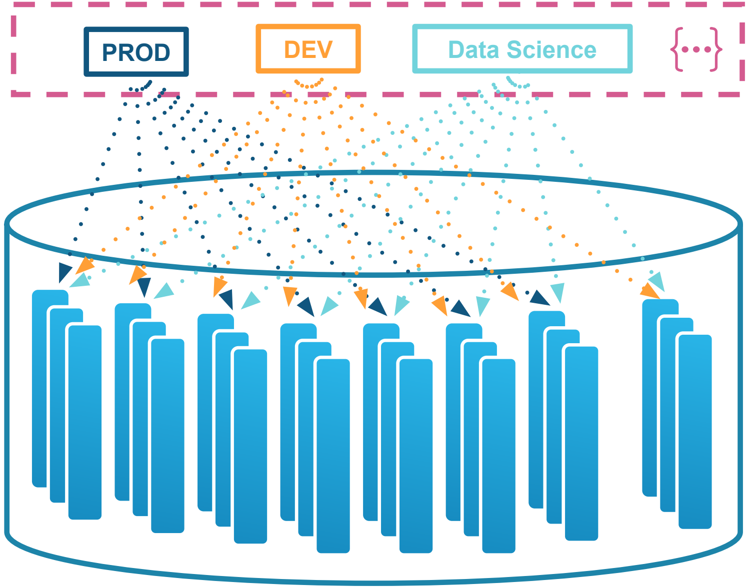
### Metadata
Metadata cached for fast access during query planning

### Data
Active working set transparently cached on virtual warehouse SSD

### Query results
Results sets cached for reuse without requiring compute
(e.g., static dashboard queries)



# Zero-Copy Cloning




**The Metadata layer keeps track of every micro-partition file in every customer database.**

**Creating a DEV environment usually means copying the PROD database**

Limited to subset of full Prod

Up to 2x storage requirement

Periodic refreshes

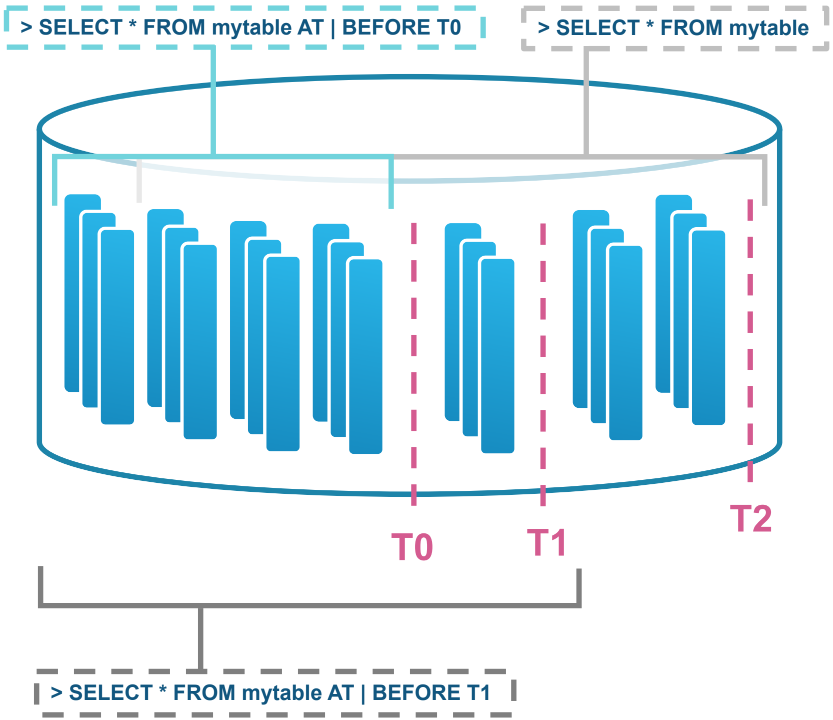
**Snowflake Zero-Copy Clones**

Simply “point” to the same files

Consumes zero additional storage

Changes to either DB are isolated

# Time Travel




**T0 – Initial state of database**

**T1 –** `update myTable set colX = Y where…`

**T2 – ELT job loads new data**

## Previous versions of data automatically retained

AT | BEFORE [ timestamp | statement | offset ]

CLONE AT | BEFORE to recreate a prior version

UNDROP recovers from accidental deletion

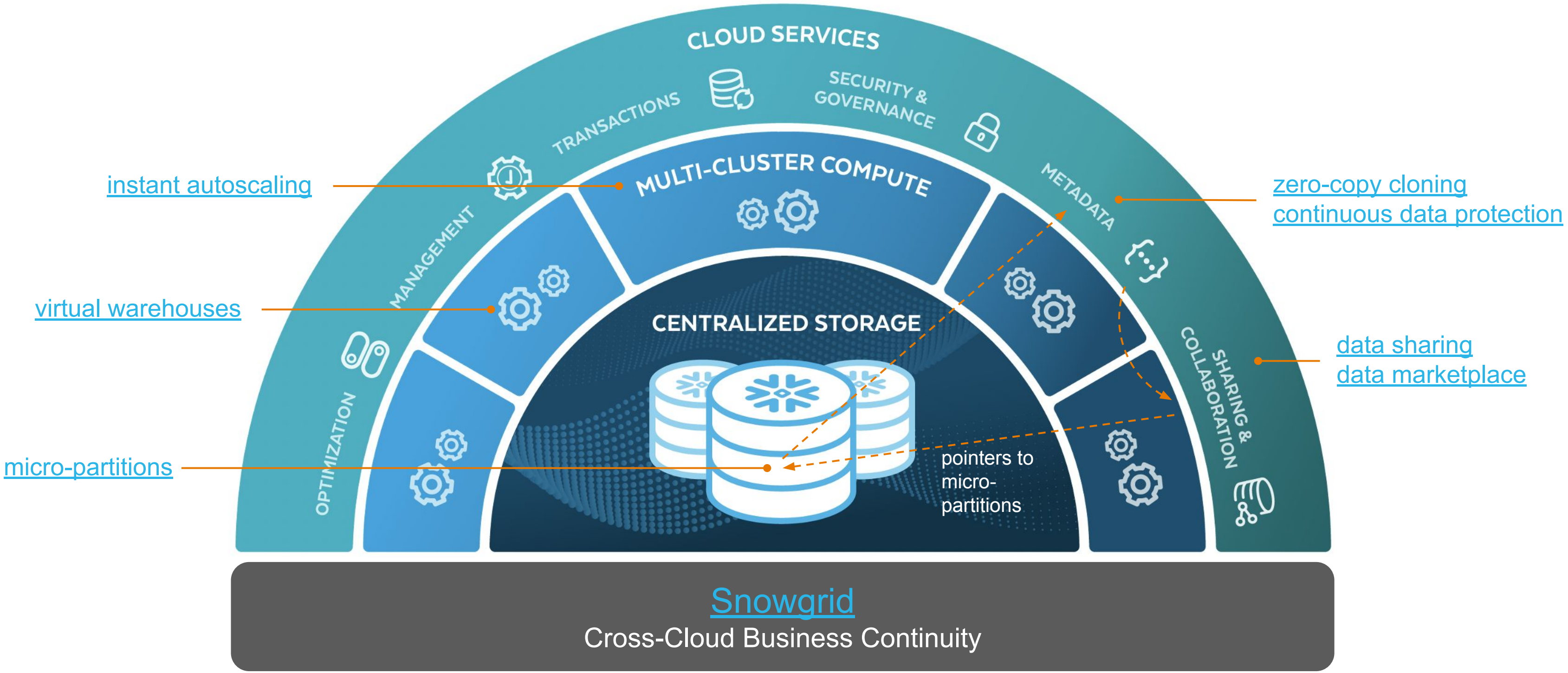
## Accessed via SQL extensions


AT | BEFORE [ timestamp | statement | offset ]

CLONE AT | BEFORE to recreate a prior version

UNDROP recovers from accidental deletion

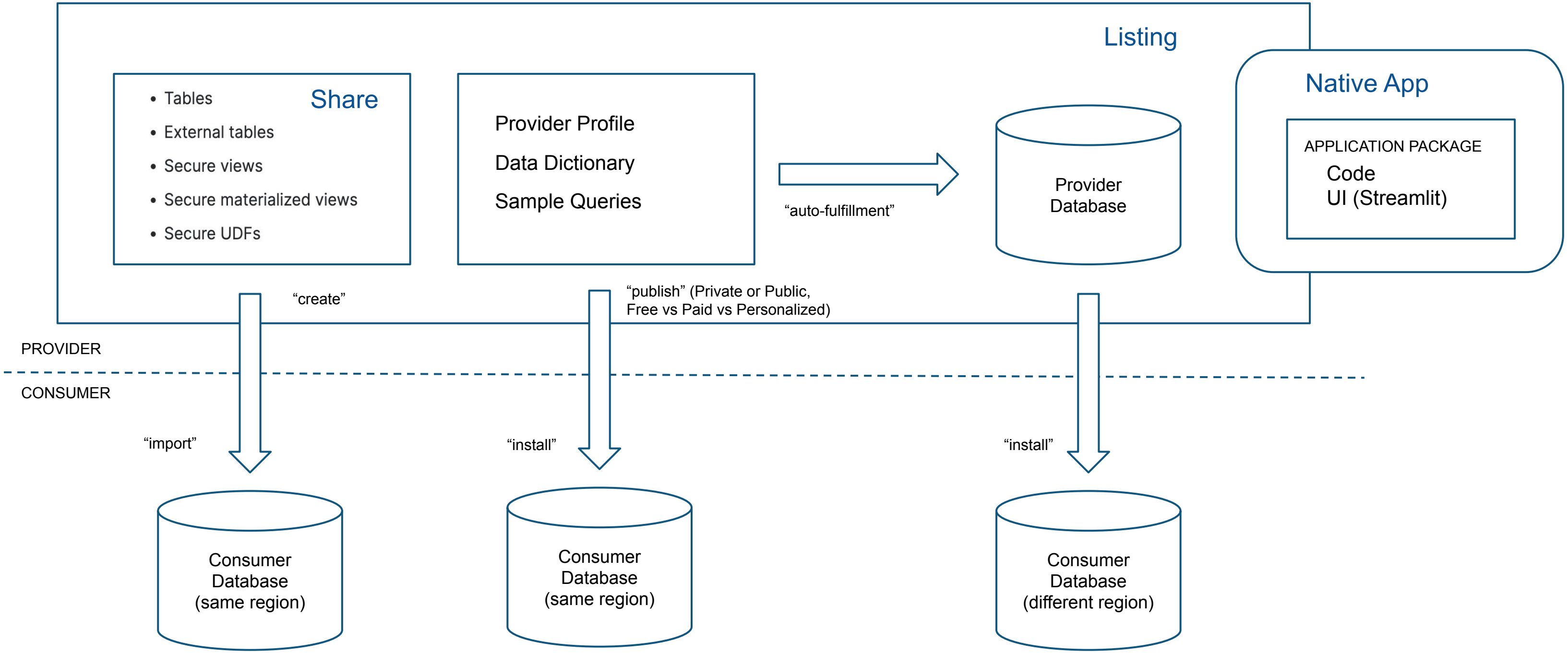
# Snowflake Architecture



# Data Collaboration Concepts

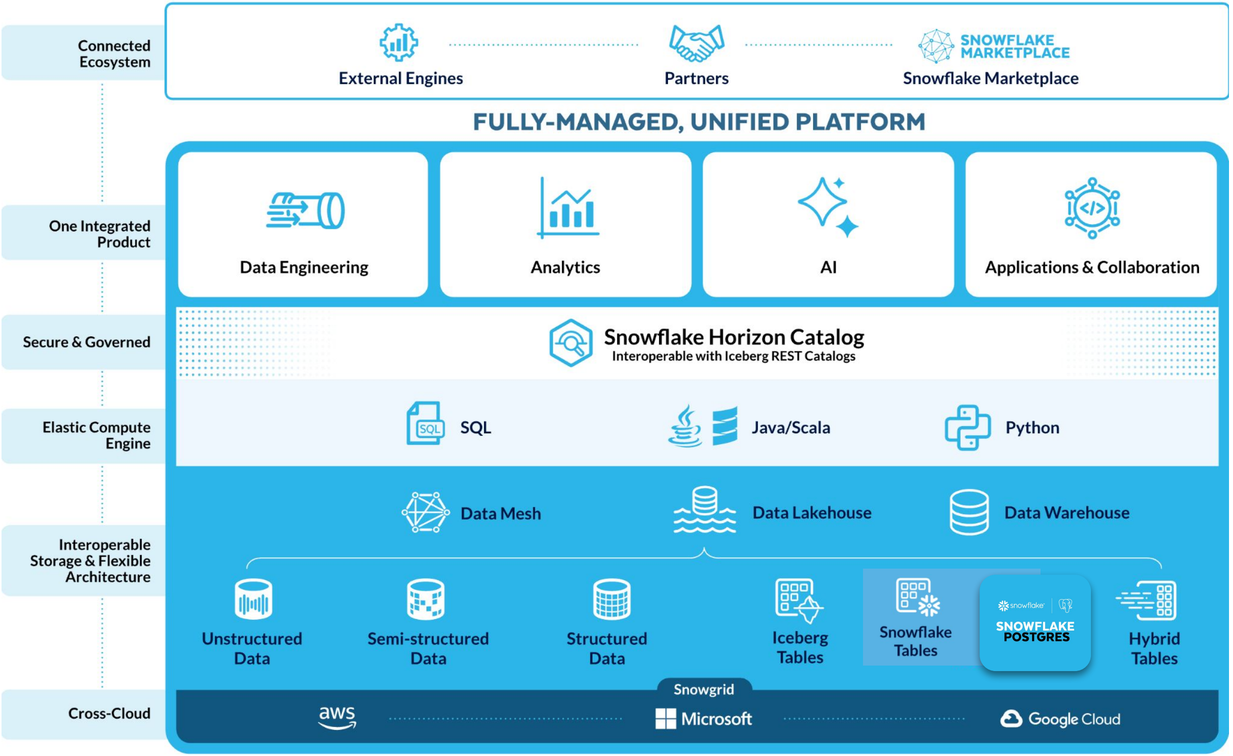
*Shares are named Snowflake objects that encapsulate all of the information required to share a database.*

*A listing is an enhanced method of [Secure Data Sharing](#) and uses the same [provider and consumer model](#).*




https://other-docs.snowflake.com/en/collaboration/provider-listings-preparing

# Unified Platform for All Enterprise Data and AI



# Snowsight UI
# & visualization tools
# Quick Tour