OUR MISSION & STORY

Enable every organisation to be data-driven

Founded in 2012 by Oracle industry veterans

$1.4B funding from leading investors on a $12.4B valuation

Gartner

3,500+ active customers, "The Cloud Data Platform" launched and IPO 2020

2015 general availability of "The Data Warehouse Built for the Cloud"

salesforce

Gartner Data Management Solutions for Analytics leader

>600m jobs per day

200+ PB total storage, biggest table 68TN rows

1.2k data providers

NPS - 71, industry average 21

© 2020 Snowflake Inc. All Rights Reserved

ABOUT SNOWFLAKE

COMPANY

2020 Inc. BEST WORK-PLACES

SNOW LISTED NYSE

Forbes

#1 on The Cloud 100 2020 List

2,000+
Employees Worldwide

SOLUTION

Snowflake customers include
7 of the Fortune 10
companies *

71

2020
Net Promoter Score

3,500+
Customers

ECOSYSTEM

salesforce

Deloitte.

aws

Google Cloud

Microsoft

accenture

1,000+
Partners

* As of July 31, 2020

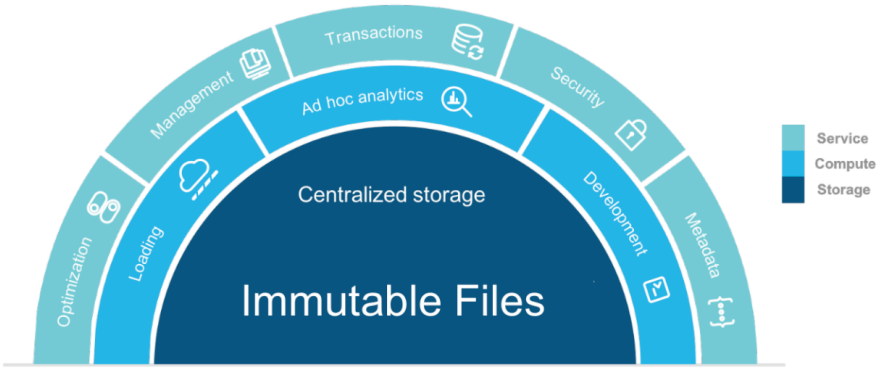
snowflake®

At its core: Data warehouse built for the cloud, focus on elasticity

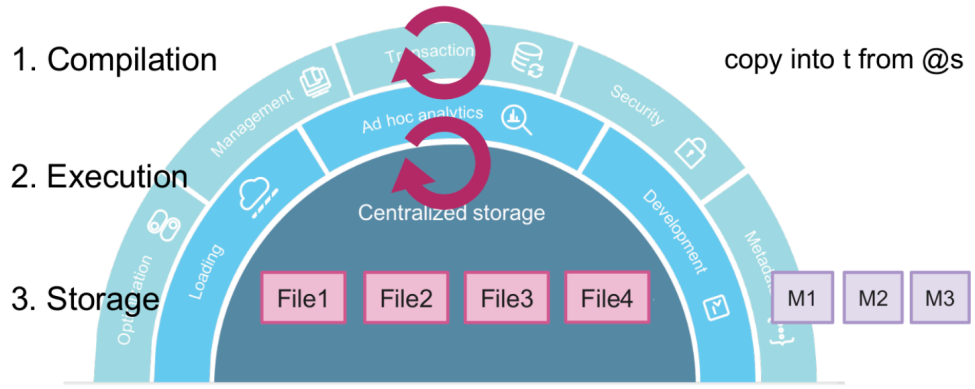
Running on top of AWS, Microsoft Azure, GCP

Use cases: business intelligence, analytics, reporting, dashboarding, application stack, …

MULTI-CLUSTER, SHARED DATA ARCHITECTURE



MULTI-CLUSTER, SHARED DATA ARCHITECTURE



OVERVIEW

uid	name
1	Allison
2	Max
3	Benoit
4	Neda
5	Thierry
6	Florian

DMLs

- Micro partitions
- Metadata

Select

- Pruning, Time travel, Cloning

Research areas

- Automatic Clustering

DMLS

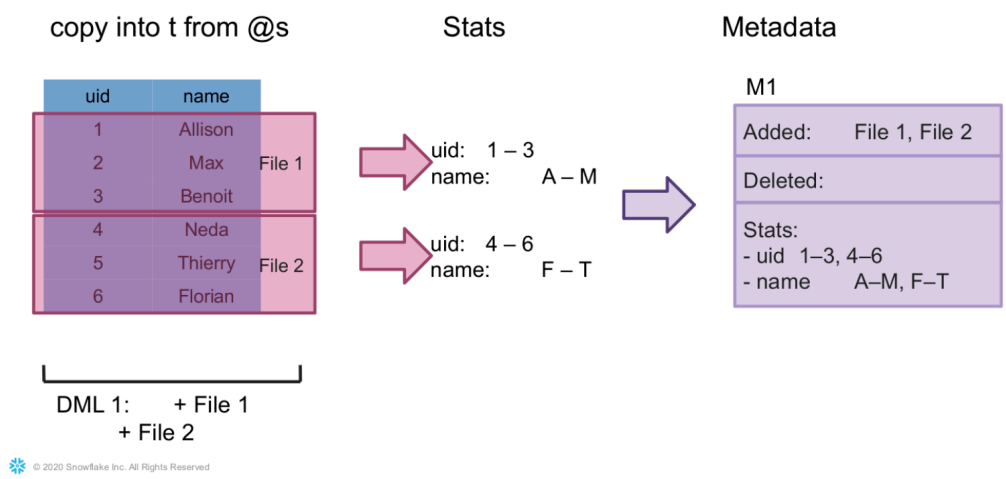
DMLS

uid	name
1	Allison
2	Max
3	Benoit
4	Neda
5	Thierry
6	Florian

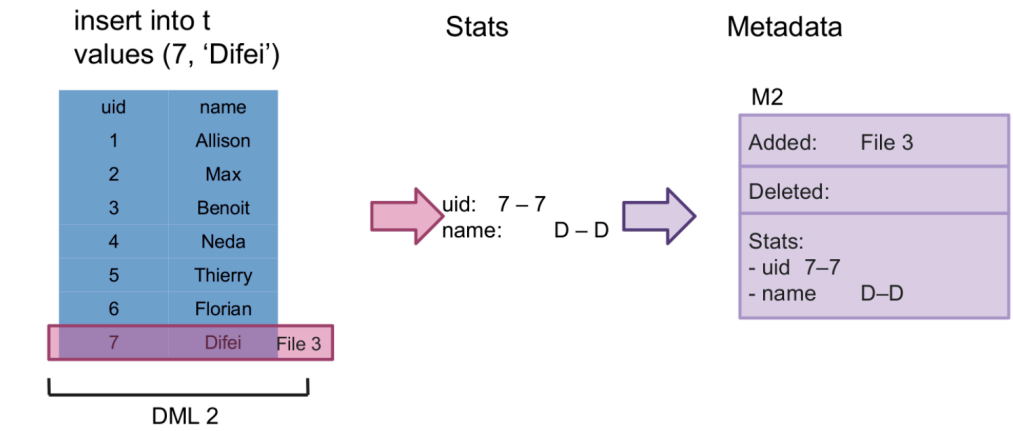
DMLs

- copy into t from @s
- insert into t values (7, ‘Difei’)
- delete from t where name = ‘Neda’

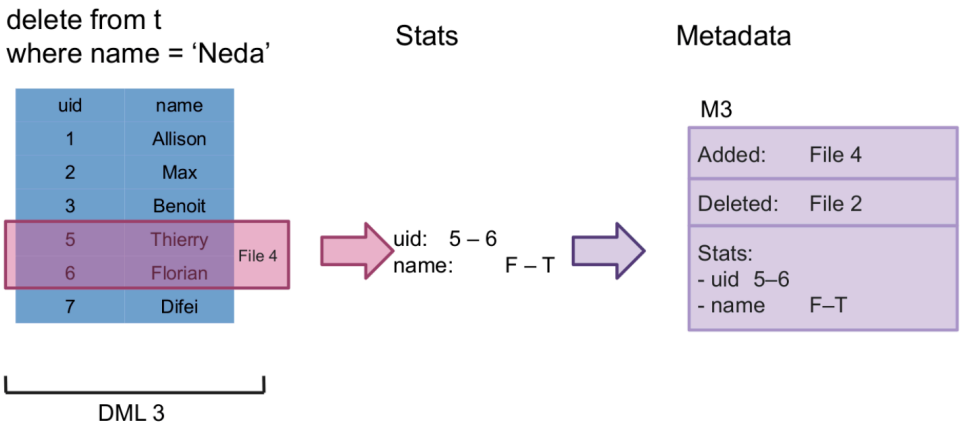
MICRO PARTITIONS FOR DML 1



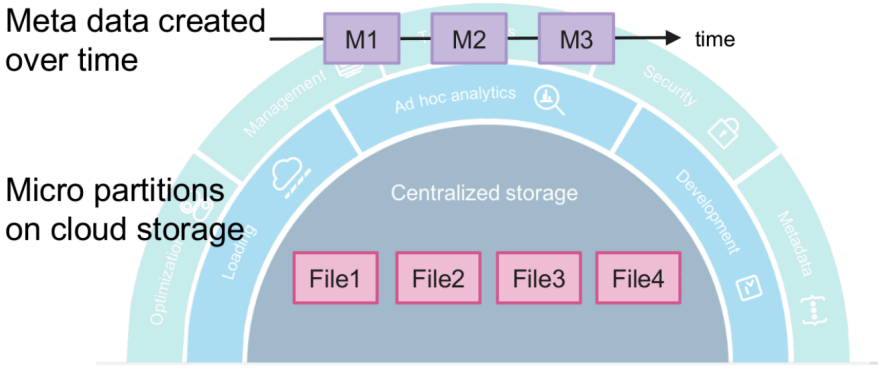
MICRO PARTITIONS FOR DML 2



MICRO PARTITIONS FOR DML 3



SUMMARY



SELECT AND PRUNING, TIME TRAVEL, CLONING

SELECT AND PRUNING

```
select * from t
```

Load metadata	M1, M2, M3
Compute scan set	Files 1, 3, 4
No pruning	

```
select * from t where uid = 5
```

Load metadata	M1, M2, M3
Compute scan set	Files 1, 3, 4
After pruning	File 4

Meta data M1

Added:	Files 1, 2
Deleted:	
Stats: uid	1–3, 4–6
name	A–M, F–T

Meta data M2

Added:	File 3
Deleted:	
Stats: uid	7–7
name	D–D

Meta data M3

Added:	File 4
Deleted:	File 2
Stats: uid	5–6
name	F–T

TIME TRAVEL

```
select * from t at (timestamp => '3pm')
```

Load metadata	M1, M2
Compute scan set	Files 1, 2, 3
No pruning	

```
select * from t  at (timestamp => '3pm')
where uid = 5
```

Load metadata	M1, M2
Compute scan set	Files 1, 2, 3
After pruning	File 2

Meta data M1 (2pm)

Added:	Files 1, 2
Deleted:	
Stats: uid	1–3, 4–6
name	A–M, F–T

Meta data M2 (3pm)

Added:	File 3
Deleted:	
Stats: uid	7–7
name	D–D

Meta data M3 (4pm)

Added:	File 4
Deleted:	File 2
Stats: uid	5–6
name	F–T

ZERO-COPY CLONING

```
create table t2 clone t
```

Consolidate metadata M1, M2, M3
Associate with table t2

New metadata M4

Added:	Files 1, 3, 4
Deleted:	
Stats: uid	1–3, 7–7, 5–6
name	A–M, D–D, F–T

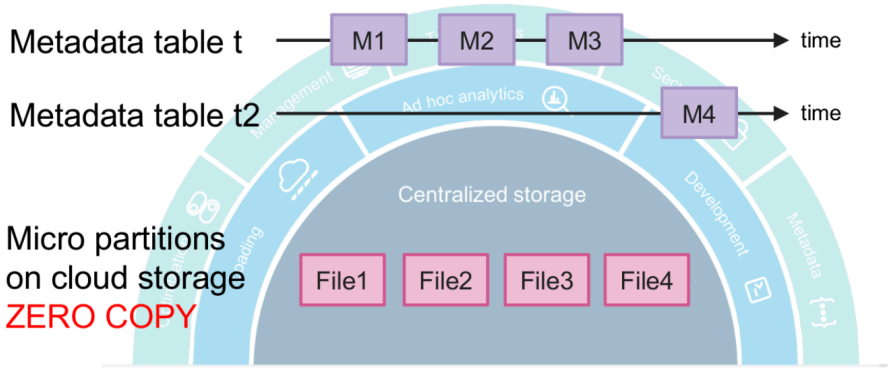
```
create table t2 clone t
at (timestamp => '3pm')
```

Consolidate metadata M1, M2
Associate with table t2

New metadata

Added:	Files 1, 2, 3
Deleted:	
Stats: uid	1–3, 4–6, 7–7
name	A–M, F–T, D–D

CLONING SUMMARY



AUTOMATIC CLUSTERING

Recall: Data Pruning

- **Compile-time** pruning
  - Min/max metadata checked against predicates
  - Even works for complex predicates e.g. **MONTH(date)-1=7**
  - Tricky for some functions (e.g. **DAYOFWEEK(date)=3**), impossible for others
- **Run-time** pruning
  - Based on metadata from query operations, e.g. probe-side join pruning, scalar subqueries
- Problem: Pruning performance & impact depends heavily on data organization.
  - ⇒ **"Clustering"**

Example: Natural Data Clustering

	uid	name
File 1	1	Allison
	2	Max
File 2	3	Benoit
	5	Thierry
File 3	6	Florian
	7	Difei

Naturally, the data layout ("clustering") follows the order of DML operations.

```
SELECT name FROM t1
WHERE id = 3;
```
☾ *Scans only F2*

```
SELECT count(*) FROM t1
WHERE name like 'D%';
```
☾ *Scans the whole table*

Example: Desired Data Clustering

	uid	name
File 1	1	Allison
	3	Benoit
File 2	7	Difei
	6	Florian
File 3	2	Max
	5	Thierry

Ideally, the data layout should follow the most ”relevant” attribute.

```
SELECT name FROM t1
WHERE id = 3;
```
☾ *Scans F1 and F3*

```
SELECT count(*) FROM t1
WHERE name like 'D%';
```
☾ *Scans only F2*

Ⓟ Optimize the data layout to guarantee consistent query performance!

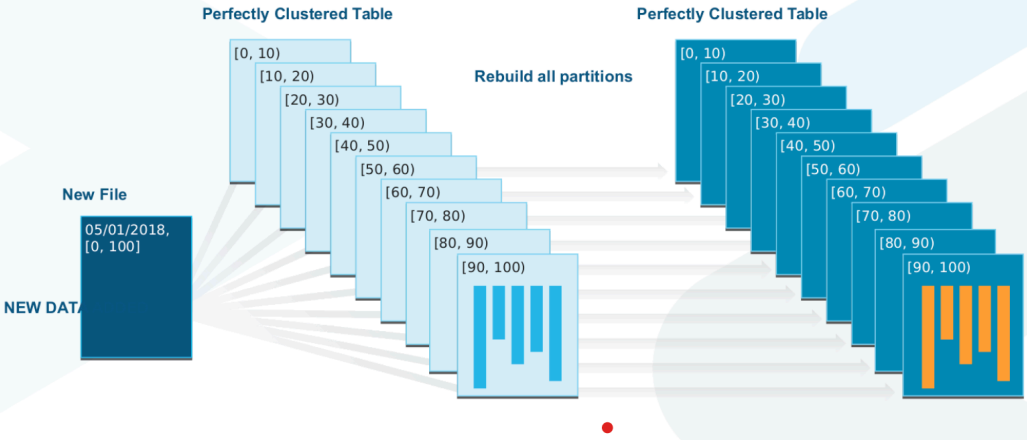
Clustering a Table

- Customer provides **Clustering Keys**
  - Indicates intention that the table will primarily be queried by those keys
  - Should specify “interesting columns”, i.e. columns on which the workload frequently performs selective filters. Examples: Customer ID, Geolocation, LINEARIZE(app_id, date), …
- Snowflake automatically maintains the table organization
  - ⇒ Example of using cloud resources to optimize query performance
- In Snowflake: **Clustering = Maintaining an (approximate) sort order**
  - Partitions are re-organized to contain similar ranges of clustering keys
  - Goal is not to sort, but to guarantee good pruning performance based on min/max
  - Automatic Clustering: **Continuous Approximate Sorting at Petabyte Scale**

Strategies

- Periodic batch reclustering
  - Extremely expensive, blocks other changes (full table rewrite), not suitable for background
  - Query performance degrades until next recluster starts
  - Not practical on PB-size tables
- Recluster tables inline with incoming changes
  - Puts all of the load on the DML operations
  - Direct impact on customer performance (DMLs get more expensive)
  - Suffers from write amplification

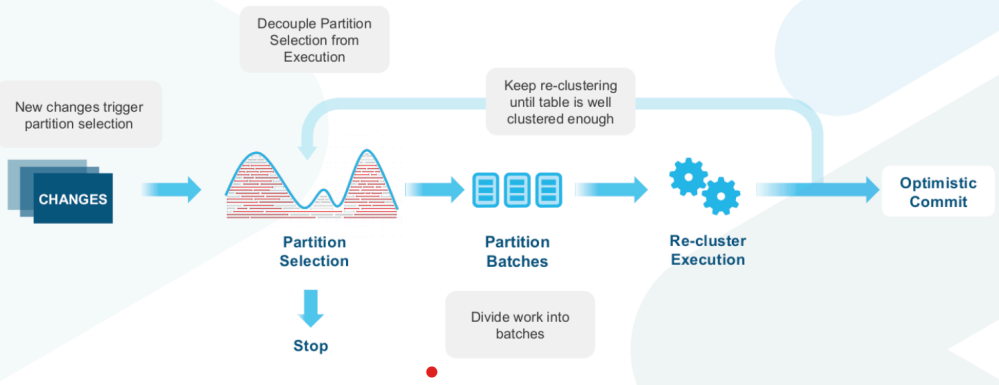
Write Amplification



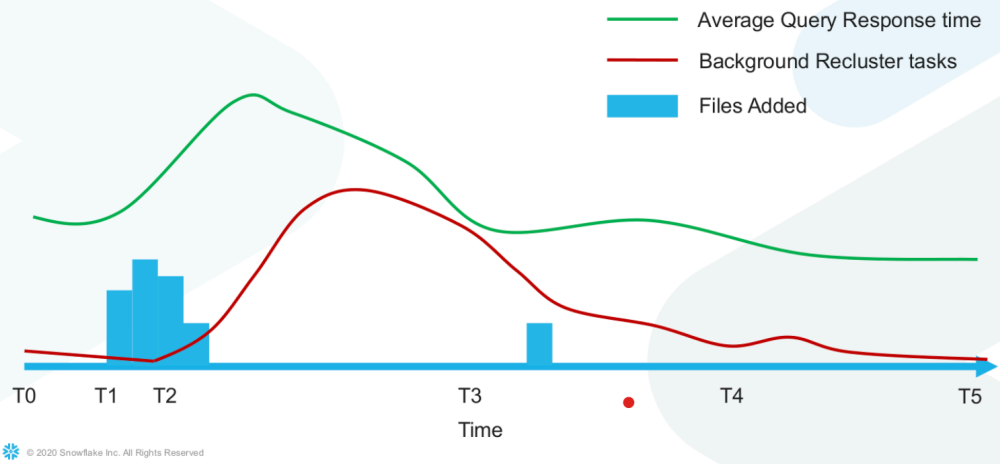
Strategies

- Periodic batch reclustering
  - Extremely expensive, blocks other changes (full table rewrite), not suitable for background
  - Query performance degrades until next recluster starts
  - Not practical on PB-size tables
- Recluster tables inline with incoming changes
  - Puts all of the load on the DML operations
  - Direct impact on customer performance (DMLs get more expensive)
  - Suffers from write amplification
- Our approach: **Incrementally (and approximately) recluster tables in background**
  - Background task periodically selects a (small) set of candidate files that are reclustered to maintain optimal table layout.

Automatic Incremental Re-Clustering



Effect on Query Performance



Clustering Metrics: Depth

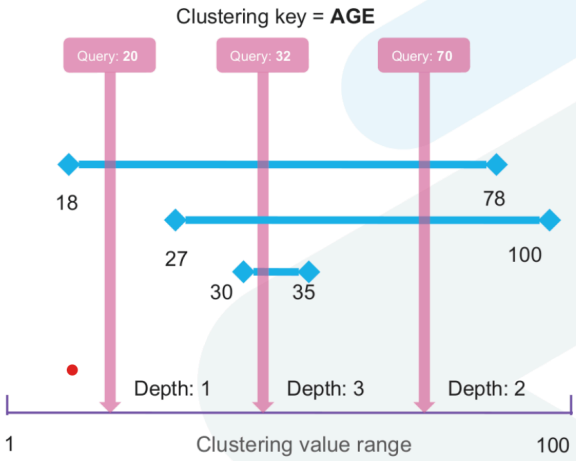
- **Depth at a single value (or range):**
  Number of partitions overlapping at a certain value in the clustering key domain value range

Partition 1

Sam	18	USA
Trevor	78	Canada

Partition 2

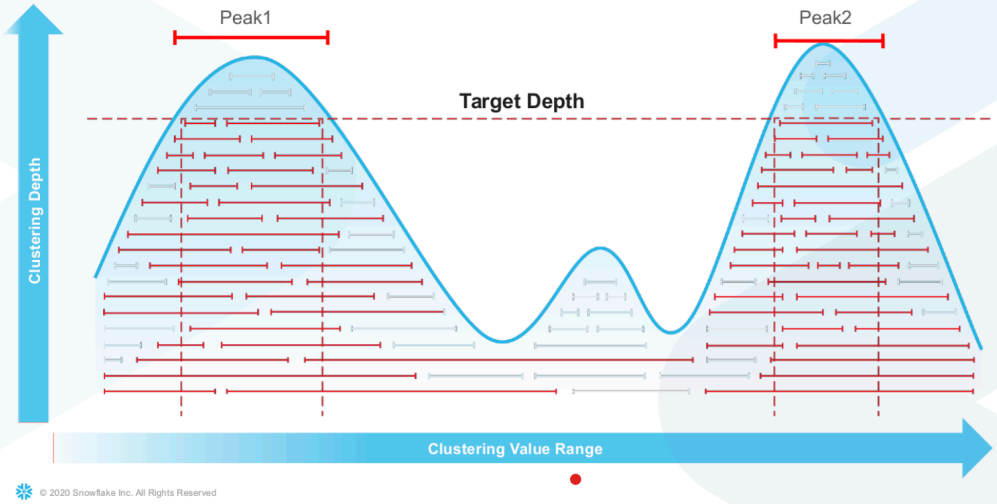
Anna	30	England
Raj	35	India



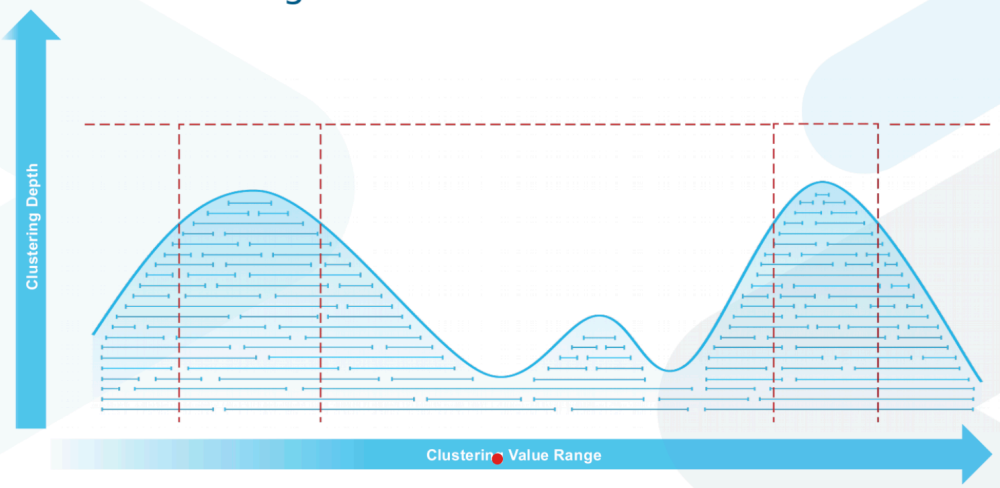
Partition Selection: Intuition

Reduce **Worst** Clustering Depth
(below an acceptable threshold)
to get
Predictable Query Performance

Partition Selection: Intuition



After Clustering



Partition Selection: Algorithm Internals

- Intuition: Work on the peaks
  - Sort the micro-partition endpoints
  - First pass: Compute peak ranges and calculate the number of overlapping micro-partitions
    - Stabbing count array (Segment Array)
  - Second pass: For the peak ranges – compute list of partitions ordered by depth
    - MinMaxPriorityQueue

Clustering Metrics: Width

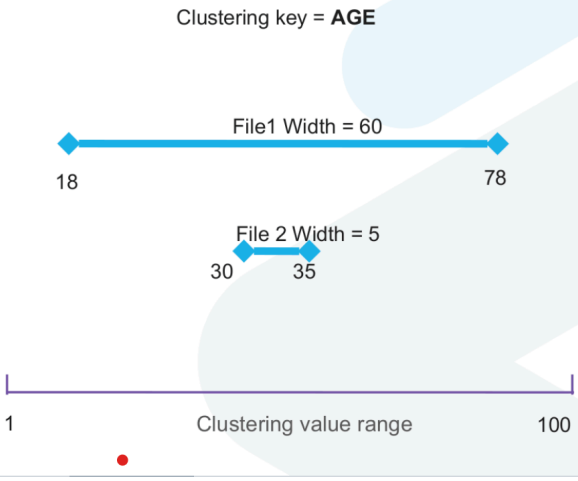
- **Width of a partition:**
  Width of the line connecting the min and max value in the clustering values domain range

Partition 1: Wide Partition

Sam	18	USA
Trevor	78	Canada

Partition 2: Narrow Partitiom

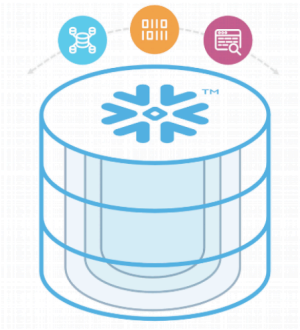
Anna	30	England
Raj	35	India



Clustering Levels

Review

- Wake up when data arrives
- Split the table files into levels (clustering state)
- Run clustering partition selection algorithm on each level
- Queue up “Recluster Tasks” for execution
- Scale up and execute re-cluster tasks
  - Sized just right – optimized not to spill
  - Non-blocking – optimistic commit
- If clustering depth is not “good enough” repeat
- Else sleep



SUMMARY

Summary

Snowflake Cloud Data Platform

Overview:

- Data Warehouse as a Service, Architecture focus on Elasticity
- Data stored on cloud storage, partitioned into (immutable) micro partitions
- File metadata helps with pruning. Allows time travel, cloning

Automatic Clustering:

- Technique to optimize data layout in the background
- Maintain an approximate sort order of the data
- Incrementally re-cluster batches of files, selected to minimize depth (partition overlap)
- ☾ Example of how to utilize cloud resources to improve DB performance.