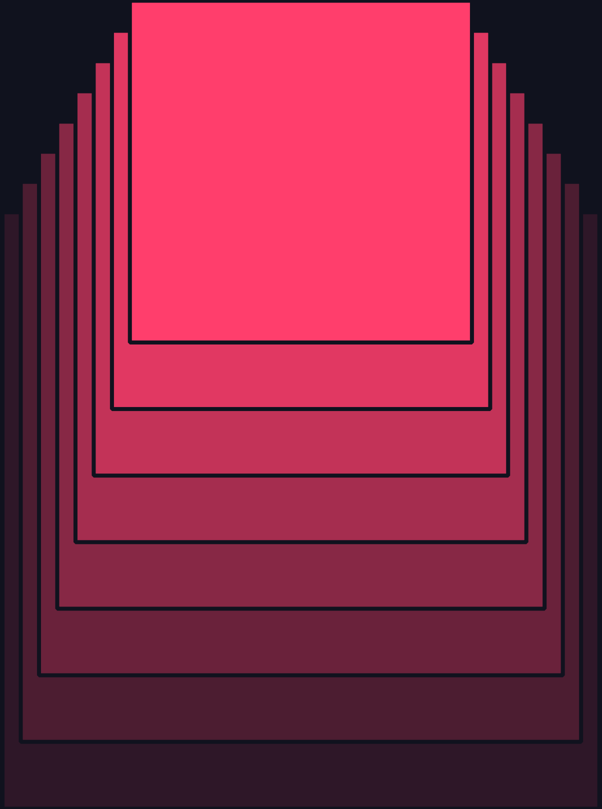
Product safe harbor statement

DATA+AI SUMMIT
BY databricks

INTRODUCING DATA INTELLIGENCE TO DELTA LAKE WITH DatabricksIQ



Sirui Sun – Sr. Staff Product Manager
Terry Kim – Sr. Staff Software Engineer

Sirui Sun

Sr. Staff Product Manager

- Product Lead, Delta Lake
- Previously Google, Microsoft

Based in Seattle

Talk to me about

- All things Delta
- All things storage

Terry Kim

Sr. Staff Software Engineer

- Technical Lead, Delta Lake
- Previously Microsoft, Yahoo

Based in Seattle

Talk to me about

- All things Delta
- All things storage

How to make a Delta table go fast

```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
CREATE TABLE tbl1 PARTITION BY date AS <query>
```

How to make a Delta table go fast

```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
CREATE TABLE tbl1 PARTITION BY date AS <query>

-- decision: what is the best target file size?
ALTER TABLE tbl1 SET TBLPROPERTIES('delta.targetFileSize' = 104857600)

-- decision: do I enable auto-compact and optimize writes?
ALTER TABLE tbl1 SET TBLPROPERTIES (
"delta.autoOptimize.autoCompact" = "true",
"delta.autoOptimize.optimizeWrite" = "true");
```

[#1] File size optimization

How to make a Delta table go fast

```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
CREATE TABLE tbl1 PARTITION BY date AS <query>

-- decision: what is the best target file size?
ALTER TABLE tbl1 SET TBLPROPERTIES('delta.targetFileSize' = 104857600)

-- decision: do I enable auto-compact and optimize writes?
ALTER TABLE tbl1 SET TBLPROPERTIES (
"delta.autoOptimize.autoCompact" = "true",
"delta.autoOptimize.optimizeWrite" = "true");
```

[#1] File size optimization

How to make a Delta table go fast

```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
CREATE TABLE tbl1 PARTITION BY date AS <query>;

-- decision: what is the best target file size?
ALTER TABLE tbl1 SET TBLPROPERTIES('delta.targetFileSize' = 104857600)

-- decision: do I enable auto-compact and optimize writes?
ALTER TABLE tbl1 SET TBLPROPERTIES (
"delta.autoOptimize.autoCompact" = "true",
"delta.autoOptimize.optimizeWrite" = "true");

-- further ZORDER by columns within the partition
-- decision: what do you partition by vs. ZORDER by?
-- run this regularly!
OPTIMIZE tbl1 ZORDER BY customerId;
```

[#2] Data layout

How to make a Delta table go fast

```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
CREATE TABLE tbl1 PARTITION BY date AS <query>

-- decision: what is the best target file size?
ALTER TABLE tbl1 SET TBLPROPERTIES(‘delta.targetFileSize’ = 104857600)

-- decision: do I enable auto-compact and optimize writes?
ALTER TABLE tbl1 SET TBLPROPERTIES (
“delta.autoOptimize.autoCompact” = “true”,
“delta.autoOptimize.optimizeWrite” = “true”);

-- further ZORDER by columns within the partition
-- decision: what do you partition by vs. ZORDER by?
-- run this regularly!
OPTIMIZE tbl1 ZORDER BY customerId;

-- gather statistics
-- ANALYZE regularly!
ALTER TABLE tbl1
SET TBLPROPERTIES (‘delta.dataSkippingNumIndexedCols” = 64)’;
ANALYZE tbl1; VACUUM tbl1;

-- if query patterns change, redo all the above
```

[#3] Continuous maintenance

How to make a Delta table go fast

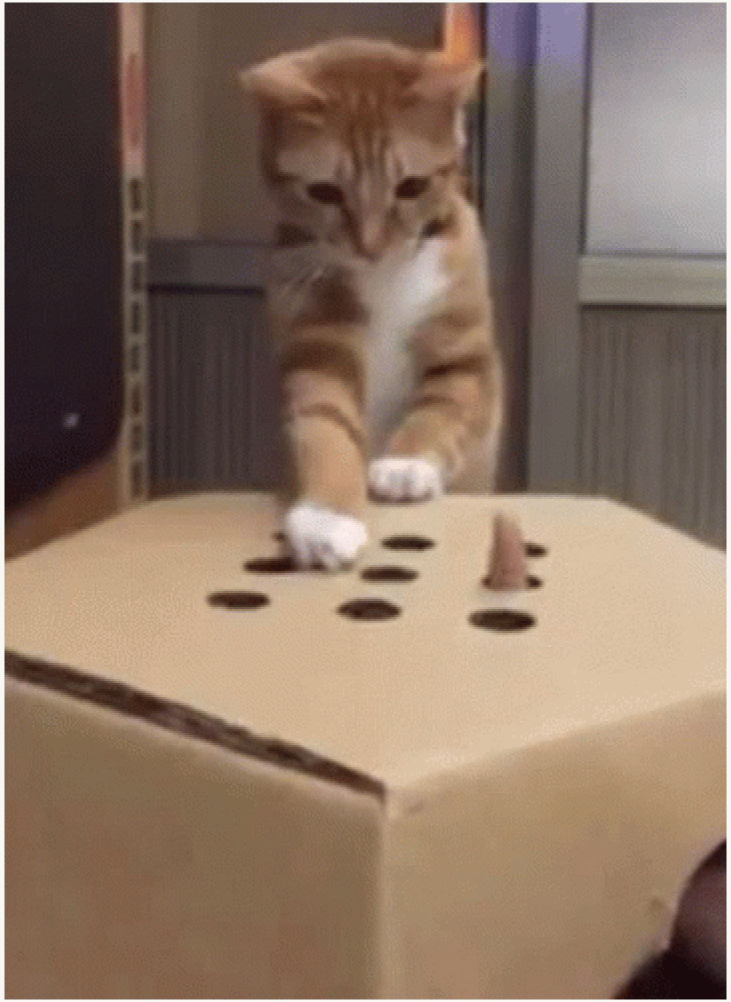
```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
CREATE TABLE tbl1 PARTITION BY date AS <query>

-- decision: what is the best target file size?
ALTER TABLE tbl1 SET TBLPROPERTIES('delta.targetFileSize' = 104857600)

-- decision: do I enable auto-compact and optimize writes?
ALTER TABLE tbl1 SET TBLPROPERTIES (
"delta.autoOptimize.autoCompact" = "true",
"delta.autoOptimize.optimizeWrite" = "true");

-- further ZORDER by columns within the partition
-- decision: what do you partition by vs. ZORDER by?
-- run this regularly!
OPTIMIZE tbl1 ZORDER BY customerId;

-- gather statistics
-- ANALYZE regularly!
ALTER TABLE tbl1
SET TBLPROPERTIES ('delta.dataSkippingNumIndexedCols" = 64)';
ANALYZE tbl1; VACUUM tbl1;

-- if query patterns change, redo all the above
```

[#3] Continuous maintenance

❓ Is your current system doing a good job?

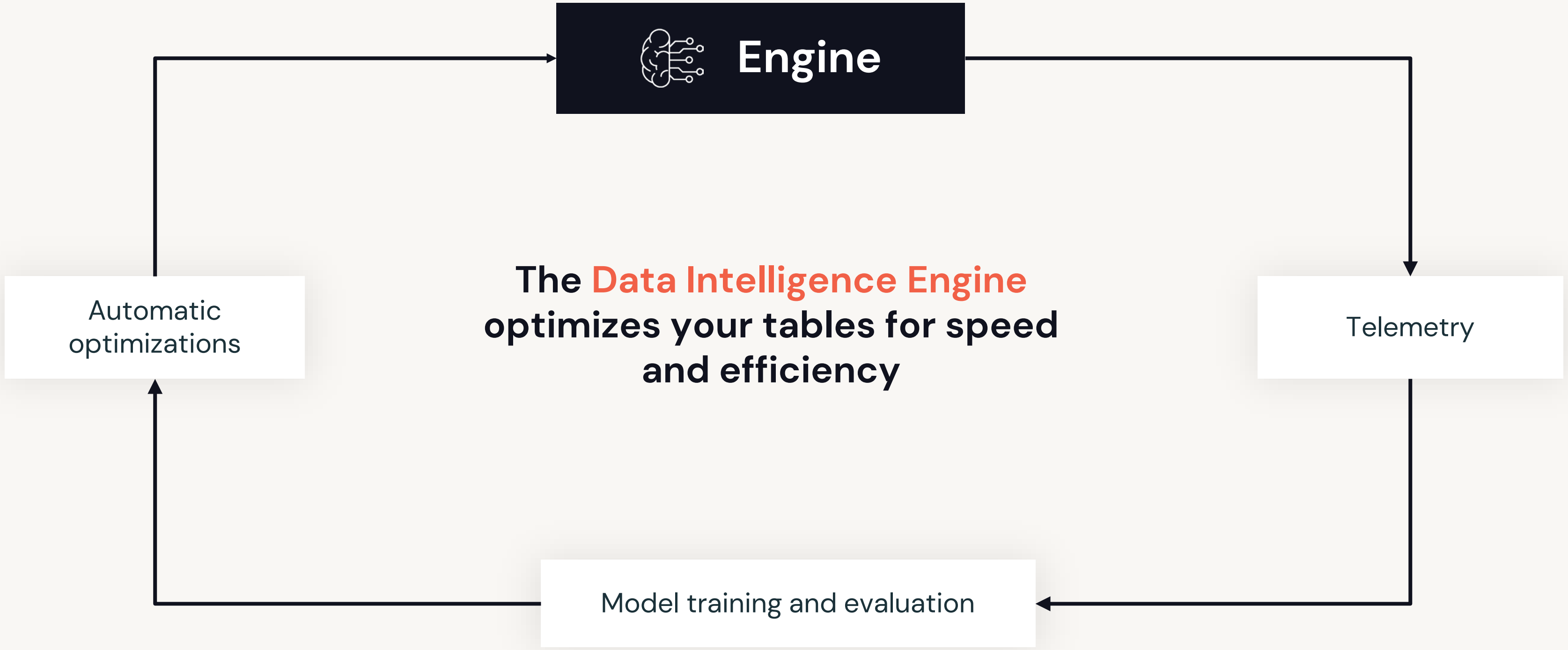
❓ How often should you re-evaluate, as usage patterns change?

❓ How does this work in a decentralized organization?



The Data Intelligence Engine optimizes your tables for speed and efficiency

Solution



The **Data Intelligence Engine** optimizes your tables for speed and efficiency

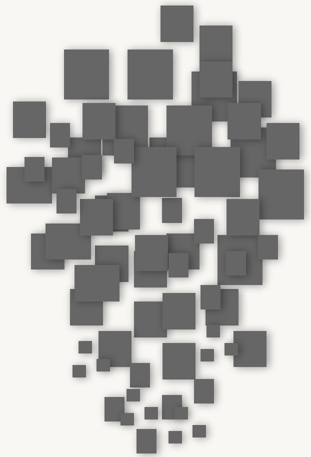
How to make a Delta table go fast

```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
CREATE TABLE tbl1 PARTITION BY date AS <query>

-- decision: what is the best target file size?
ALTER TABLE tbl1 SET TBLPROPERTIES(‘delta.targetFileSize’ = 104857600)

-- decision: do I enable auto-compact and optimize writes?
ALTER TABLE tbl1 SET TBLPROPERTIES (
“delta.autoOptimize.autoCompact” = “true”,
“delta.autoOptimize.optimizeWrite” = “true”);

-- further ZORDER by columns within the partition
-- decision: what do you partition by vs. ZORDER by?
-- run this regularly!
OPTIMIZE tbl1 ZORDER BY customerId;

-- gather statistics
-- ANALYZE regularly!
ALTER TABLE tbl1
SET TBLPROPERTIES (‘delta.dataSkippingNumIndexedCols” = 64)’;
ANALYZE tbl1; VACUUM tbl1;

-- if query patterns change, redo all the above
```

How to make a Delta table go fast

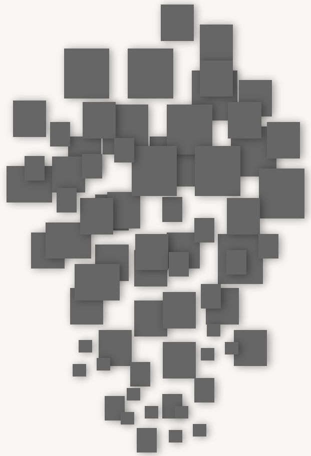
```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
CREATE TABLE tbl1 PARTITION BY date AS <query>

-- decision: what is the best target file size?
ALTER TABLE tbl1 SET TBLPROPERTIES('delta.targetFileSize' = 104857600)

-- decision: do I enable auto-compact and optimize writes?
ALTER TABLE tbl1 SET TBLPROPERTIES (
"delta.autoOptimize.autoCompact" = "true",
"delta.autoOptimize.optimizeWrite" = "true");

-- further ZORDER by columns within the partition
-- decision: what do you partition by vs. ZORDER by?
-- run this regularly!
OPTIMIZE tbl1 ZORDER BY customerId;

-- gather statistics
-- ANALYZE regularly!
ALTER TABLE tbl1
SET TBLPROPERTIES ('delta.dataSkippingNumIndexedCols" = 64)';
ANALYZE tbl1; VACUUM tbl1;

-- if query patterns change, redo all the above
```

File Sizes



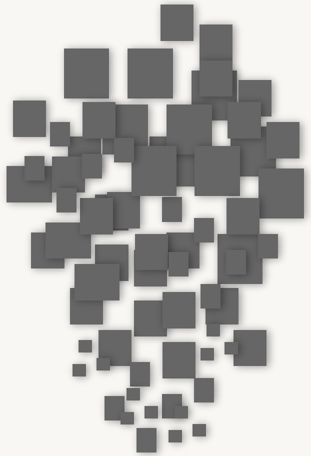
File sizes too small

File Sizes


File sizes too small

File sizes too large

File Sizes



File sizes too small

File sizes too large

Challenge

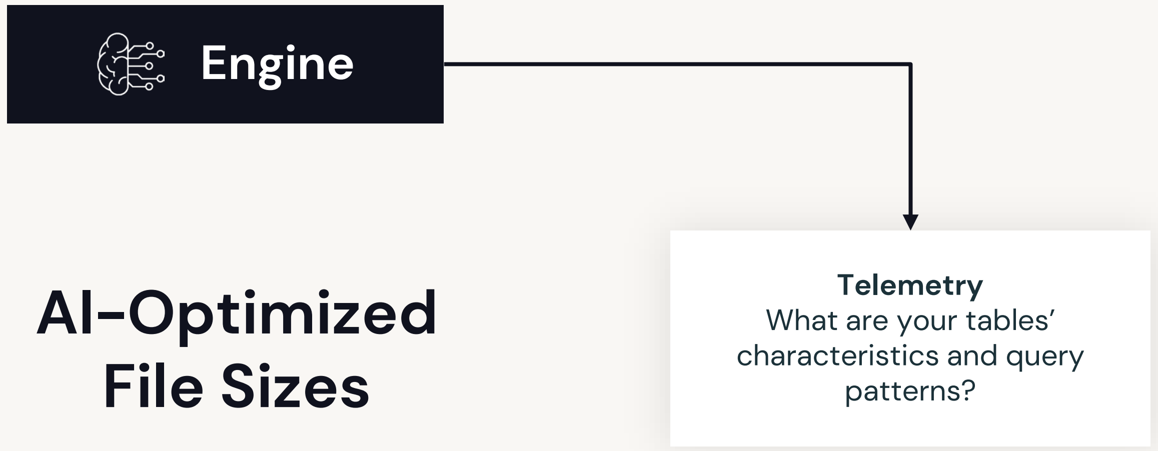
- How to determine the optimal file sizes?
- How to ensure that file sizes align with that optimal

AI-Optimized File Sizes

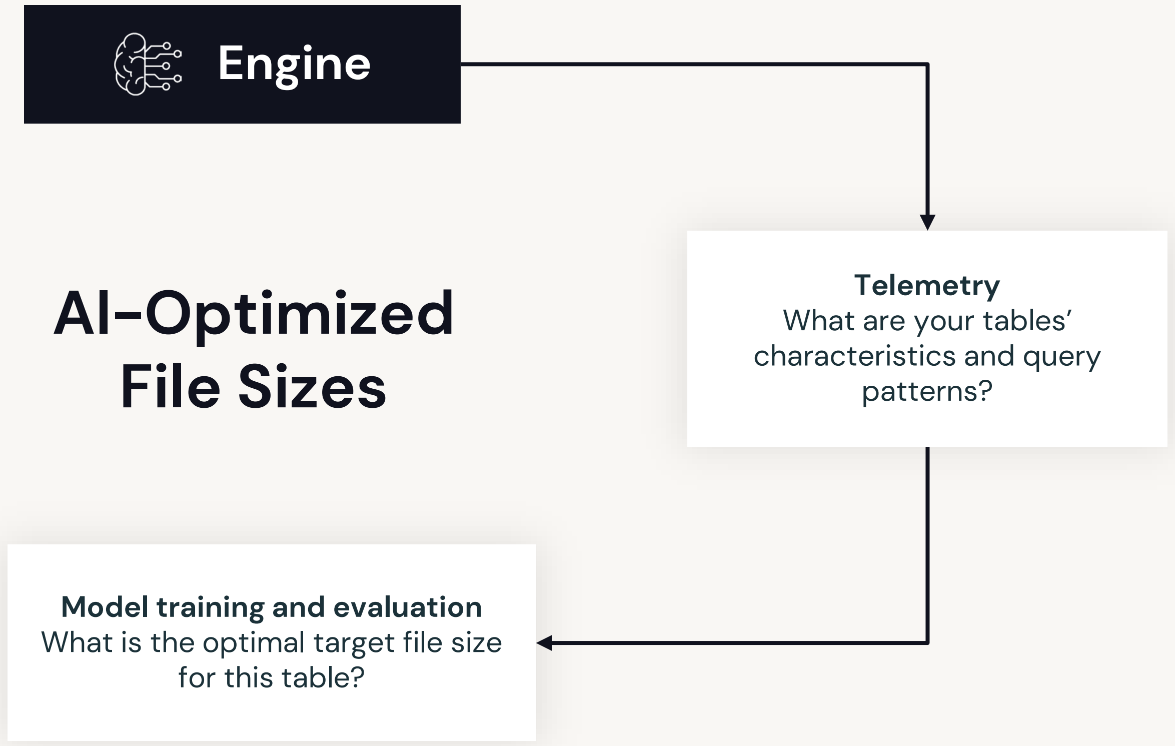
Engine

AI-Optimized
File Sizes

AI-Optimized File Sizes



AI-Optimized File Sizes



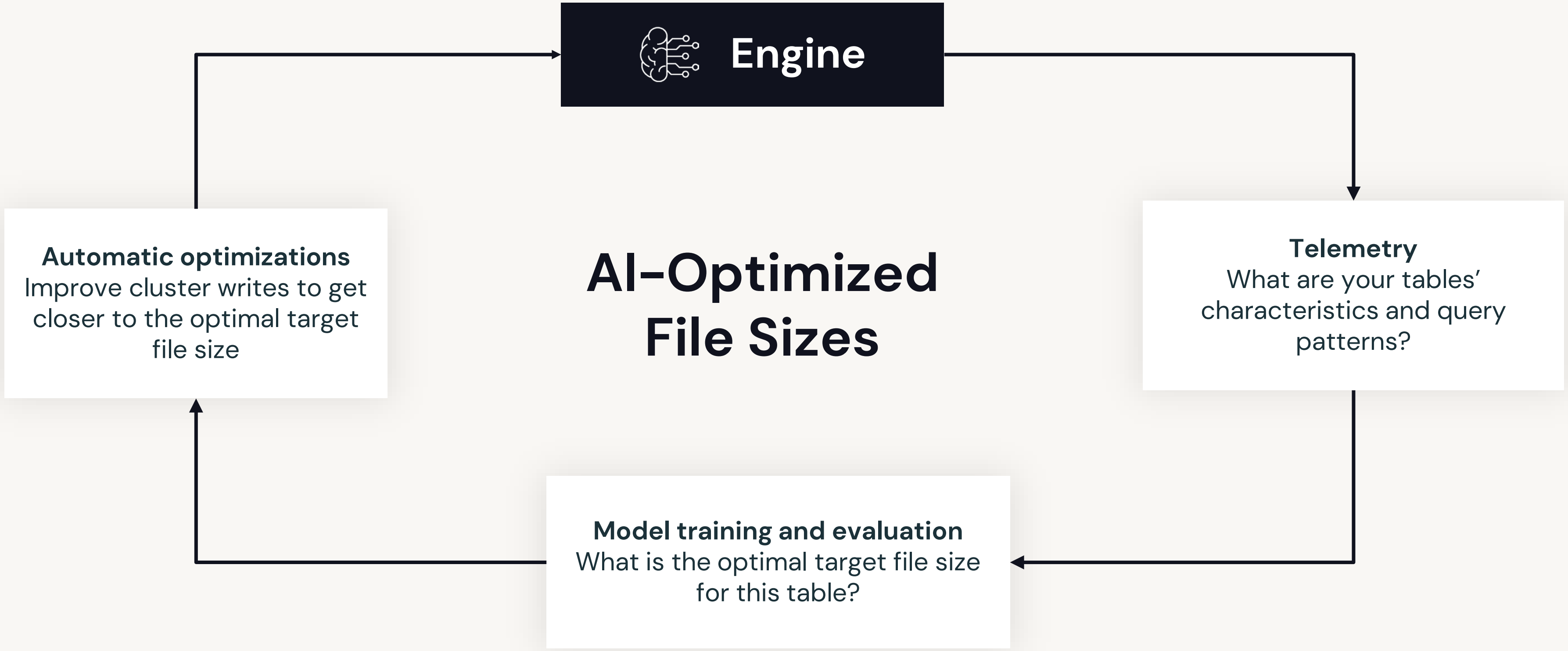
Engine

AI-Optimized File Sizes

Telemetry
What are your tables' characteristics and query patterns?

Model training and evaluation
What is the optimal target file size for this table?

AI-Optimized File Sizes


Engine
AI-Optimized File Sizes
Telemetry
What are your tables' characteristics and query patterns?
Model training and evaluation
What is the optimal target file size for this table?
Automatic optimizations
Improve cluster writes to get closer to the optimal target file size

AI-Optimized File Sizes

Results:

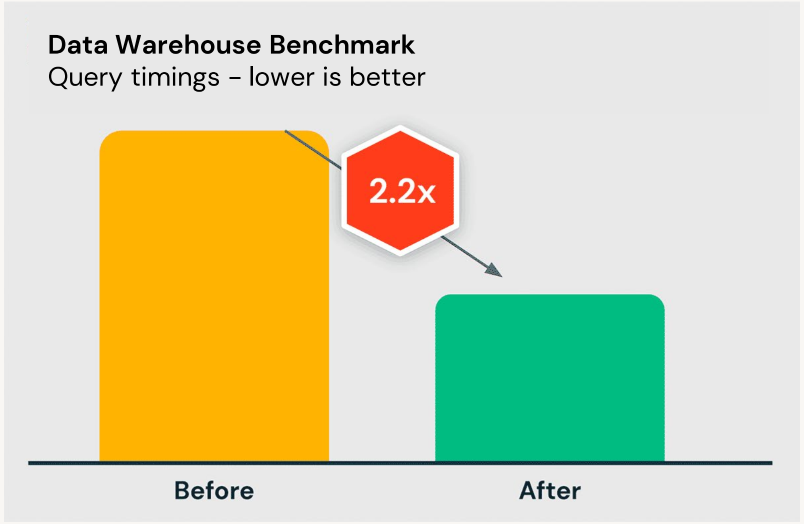
Faster, Cheaper:

- 6x improvement in average ingested file size
- Background compactions: 30x file size improvements

AI-Optimized File Sizes

Results:

30X file size improvements across the fleet



How to make a Delta table go fast

```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
CREATE TABLE tbl1 PARTITION BY date AS <query>
-- decision: what is the best target file size?
ALTER TABLE tbl1 SET TBLPROPERTIES(‘delta.targetFileSize’ = 104857600)

-- decision: do I enable auto-compact and optimize writes?
ALTER TABLE tbl1 SET TBLPROPERTIES (
“delta.autoOptimize.autoCompact” = “true”,
“delta.autoOptimize.optimizeWrite” = “true”);
-- further ZORDER by columns within the partition
-- decision: what do you partition by vs. ZORDER by?
-- run this regularly!
OPTIMIZE tbl1 ZORDER BY customerId;

-- gather statistics
-- ANALYZE regularly!
ALTER TABLE tbl1
SET TBLPROPERTIES (‘delta.dataSkippingNumIndexedCols” = 64)’;
ANALYZE tbl1; VACUUM tbl1;

-- if query patterns change, redo all the above
```

How to make a Delta table go fast

```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
CREATE TABLE tbl1 PARTITION BY date AS <query>

-- further ZORDER by columns within the partition
-- decision: what do you partition by vs. ZORDER by?
-- run this regularly!
OPTIMIZE tbl1 ZORDER BY customerId;

-- gather statistics
-- ANALYZE regularly!
ALTER TABLE tbl1
SET TBLPROPERTIES (‘delta.dataSkippingNumIndexedCols” = 64)’;
ANALYZE tbl1; VACUUM tbl1;

-- if query patterns change, redo all the above
```

How to make a Delta table go fast

```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
CREATE TABLE tbl1 PARTITION BY date AS <query>

-- further ZORDER by columns within the partition
-- decision: what do you partition by vs. ZORDER by?
-- run this regularly!
OPTIMIZE tbl1 ZORDER BY customerId;

-- gather statistics
-- ANALYZE regularly!
ALTER TABLE tbl1
SET TBLPROPERTIES (‘delta.dataSkippingNumIndexedCols” = 64)’;
ANALYZE tbl1; VACUUM tbl1;

-- if query patterns change, redo all the above
```

[#2] Data layout

Liquid Clustering (GA)

Challenge

Partitioning and ZORDERing: good for performance, but complicated:

- Which columns should be partitioned?
- Which columns should be ZORDER’ed
- What if the column is high cardinality?
- What if things change over time?

Hive-style partitioning

PARTITIONED BY (date, customerId)

	2024-06-11	2024-06-12	2024-06-13
Customer A			
Customer B			
Customer C			
Customer D			
Customer E			
Customer F			

Hive-style partitioning

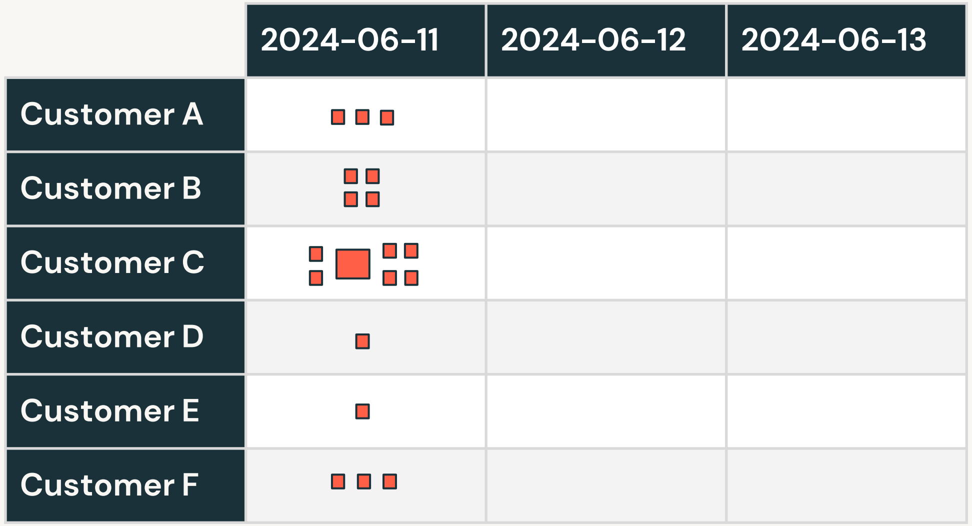
PARTITIONED BY (date, customerId)

	2024-06-11	2024-06-12	2024-06-13
Customer A			
Customer B			
Customer C			
Customer D			
Customer E			
Customer F			

Incrementally ingested hourly

Hive-style partitioning

PARTITIONED BY (date, customerId)



🟥 Target file size

Incrementally ingested hourly

Hive-style partitioning

PARTITIONED BY (date, customerId)

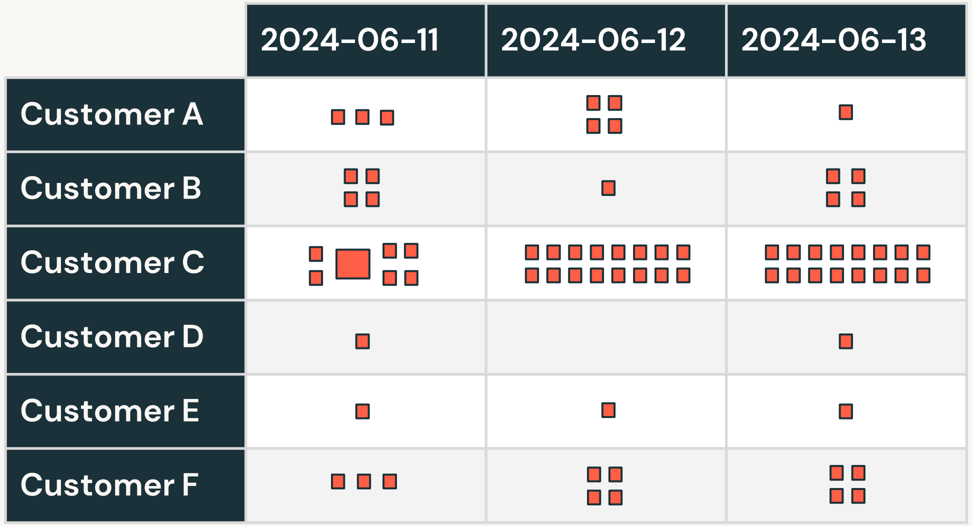
	2024-06-11	2024-06-12	2024-06-13
Customer A	🟥🟥🟥	🟥🟥🟥🟥	
Customer B	🟥🟥🟥🟥	🟥	
Customer C	🟥🟥 🟥 🟥🟥🟥🟥	🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥	
Customer D	🟥		
Customer E	🟥	🟥	
Customer F	🟥🟥🟥	🟥🟥🟥🟥	

🟥 Target file size

Incrementally ingested hourly

Hive-style partitioning

PARTITIONED BY (date, customerId)



	2024-06-11	2024-06-12	2024-06-13
Customer A	🟥🟥🟥	🟥🟥🟥🟥	🟥
Customer B	🟥🟥🟥🟥	🟥	🟥🟥🟥🟥
Customer C	🟥🟥🟥🟥🟥🟥🟥	🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥	🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥
Customer D	🟥		🟥
Customer E	🟥	🟥	🟥
Customer F	🟥🟥🟥	🟥🟥🟥🟥	🟥🟥🟥🟥

🟥 Target file size

Incrementally ingested hourly

Hive-style partitioning

PARTITIONED BY (date, customerId)

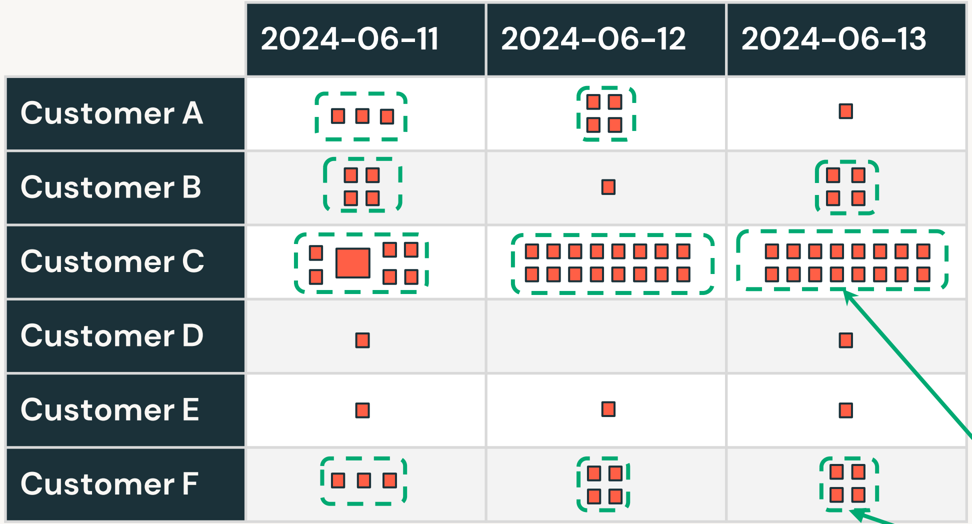
	2024-06-11	2024-06-12	2024-06-13
Customer A	▪▪▪	▪▪▪▪	▪
Customer B	▪▪▪▪	▪	▪▪▪▪
Customer C	▪▪ 🟥 ▪▪▪▪	▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪	▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪
Customer D	▪		▪
Customer E	▪	▪	▪
Customer F	▪▪▪	▪▪▪▪	▪▪▪▪

🟥 Target file size

OPTIMIZE to compact small files

Hive-style partitioning

PARTITIONED BY (date, customerId)



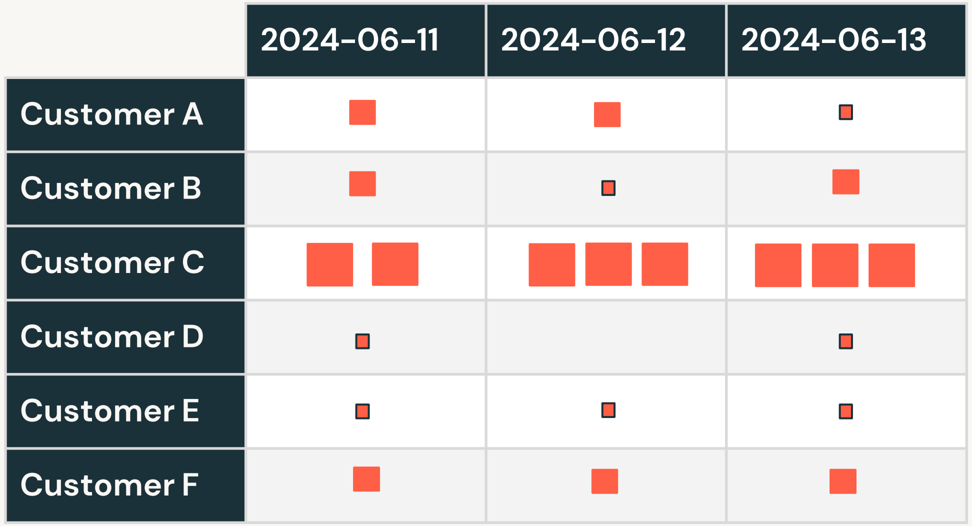
🟥 Target file size

OPTIMIZE to compact small files

Files can be compacted within partition boundaries

Hive-style partitioning

PARTITIONED BY (date, customerId)



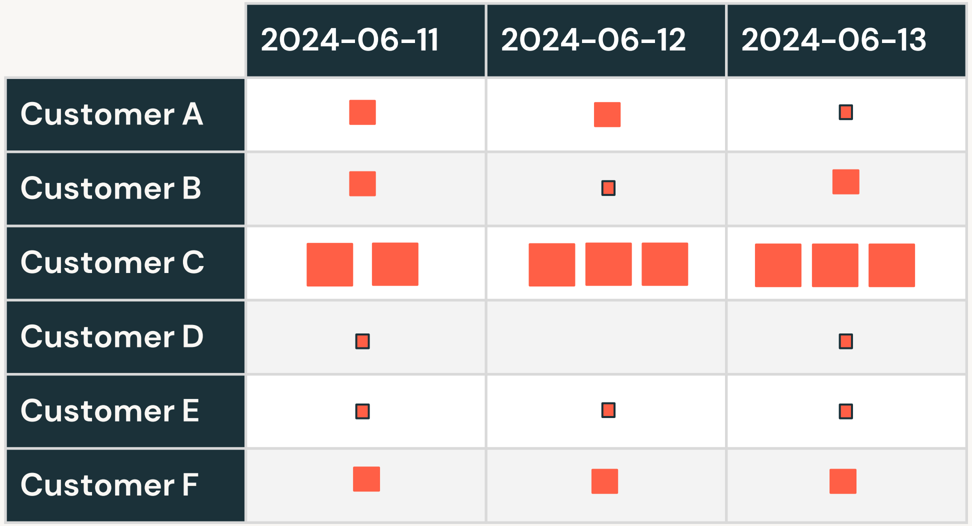
	2024-06-11	2024-06-12	2024-06-13
Customer A	🟥	🟥	▪
Customer B	🟥	▪	🟥
Customer C	🟥 🟥	🟥 🟥 🟥	🟥 🟥 🟥
Customer D	▪		▪
Customer E	▪	▪	▪
Customer F	🟥	🟥	🟥

🟥 Target file size

Small-file problem still persists -> slower read

Hive-style partitioning

PARTITIONED BY (date, customerId)



🟥 Target file size

Small-file problem still persists -> slower read

Want to partition by week? Have to rewrite the whole table!

Introducing...

Liquid Clustering

- **Fast**
  - Faster writes and similar reads vs. well-tuned partitioned tables
- **Incremental**
  - Low write amplification
- **Self-tuning / skew-resistant**
  - Avoids over- and under-partitioning
  - Produces consistent file sizes
- **Flexible**
  - Want to change the clustering columns? No problem!

Liquid Clustering

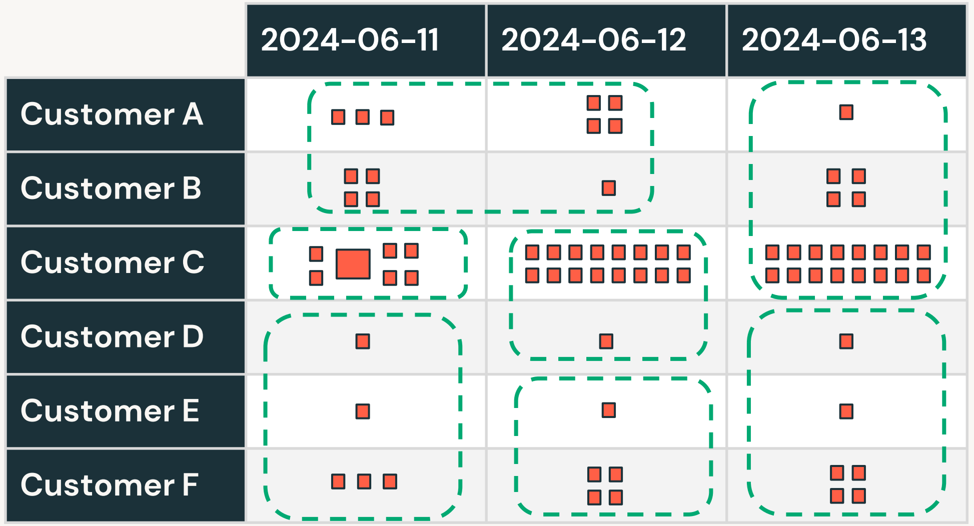
CLUSTER BY (date, customerId)

	2024-06-11	2024-06-12	2024-06-13
Customer A	▪▪▪	▪▪▪▪	▪
Customer B	▪▪▪▪	▪	▪▪▪▪
Customer C	▪▪ 🟥 ▪▪▪▪	▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪	▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪
Customer D	▪		▪
Customer E	▪	▪	▪
Customer F	▪▪▪	▪▪▪▪	▪▪▪▪

🟥 Target file size

Liquid Clustering

CLUSTER BY (date, customerId)

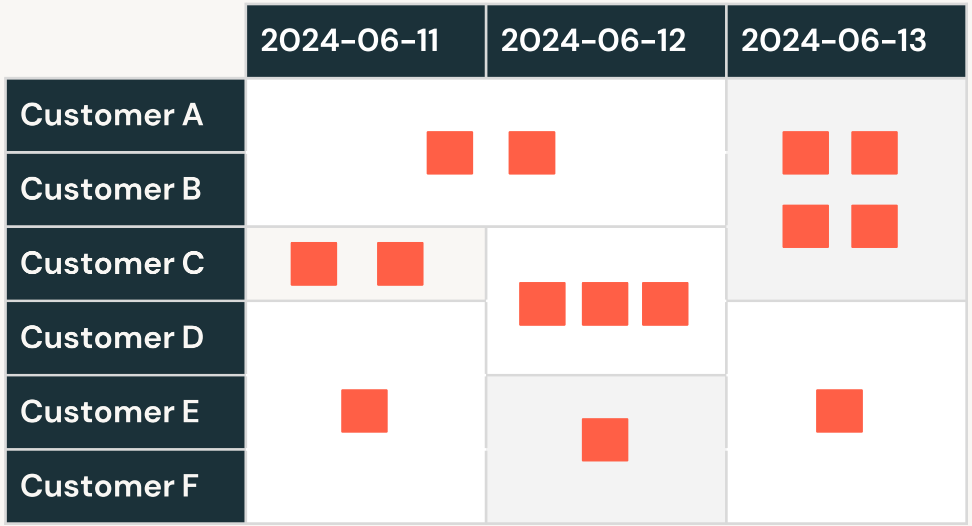


🟥 Target file size

Intelligently balance clustering vs. file size

Liquid Clustering

CLUSTER BY (date, customerId)

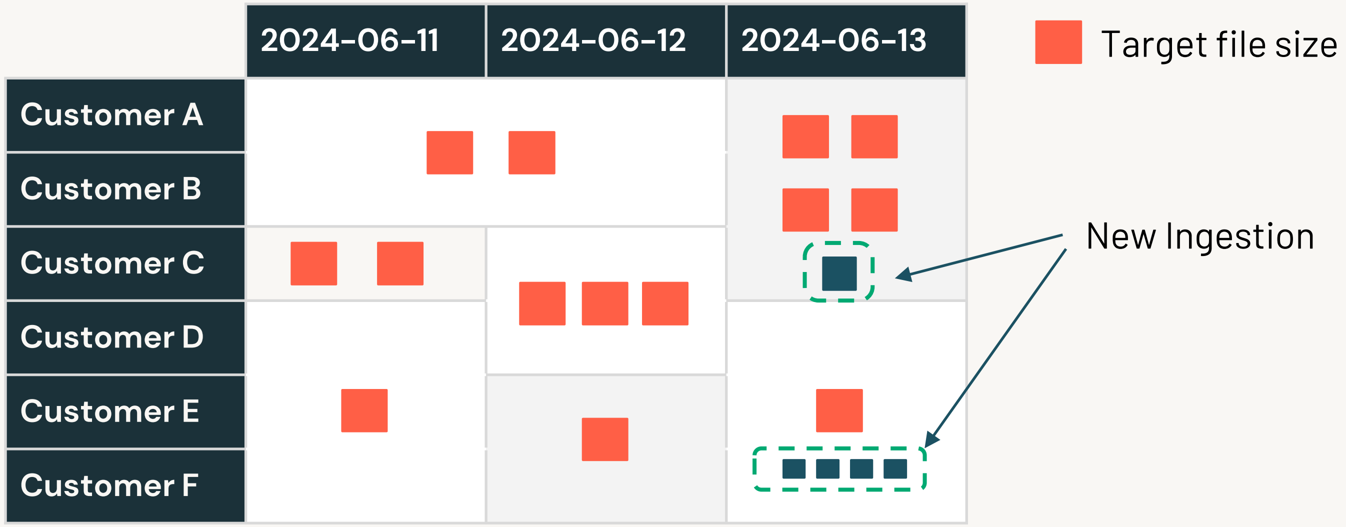


🟥 Target file size

Intelligently balance clustering vs. file size

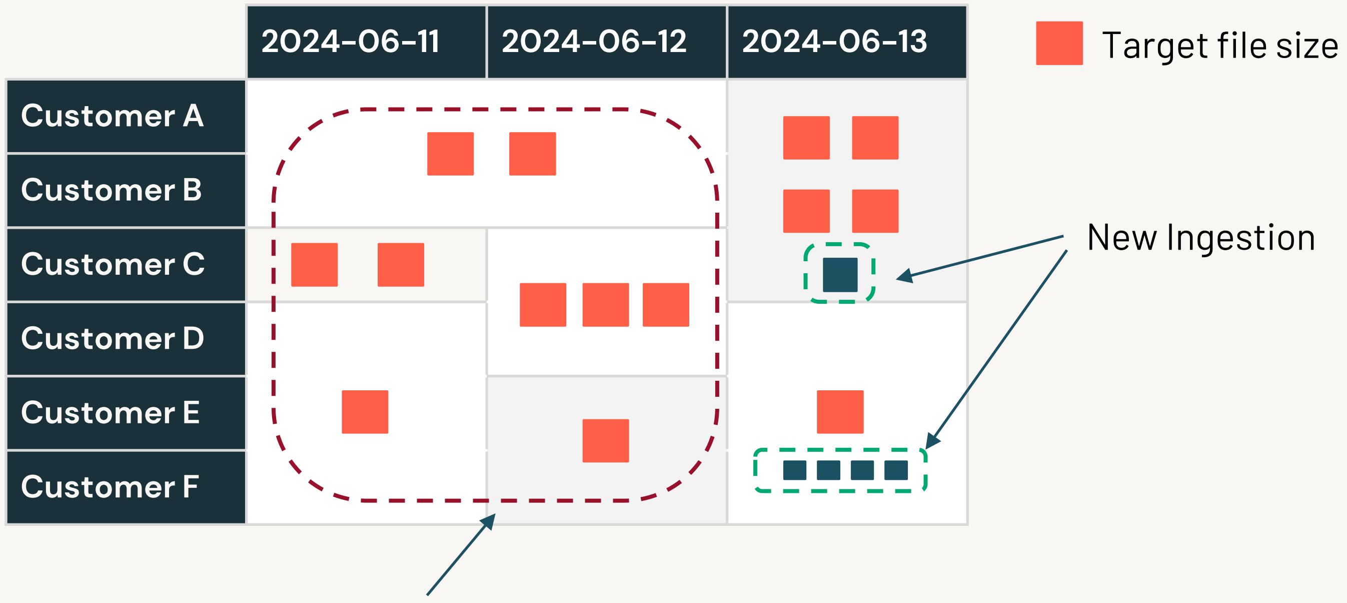
Liquid Clustering

Incremental clustering



Liquid Clustering

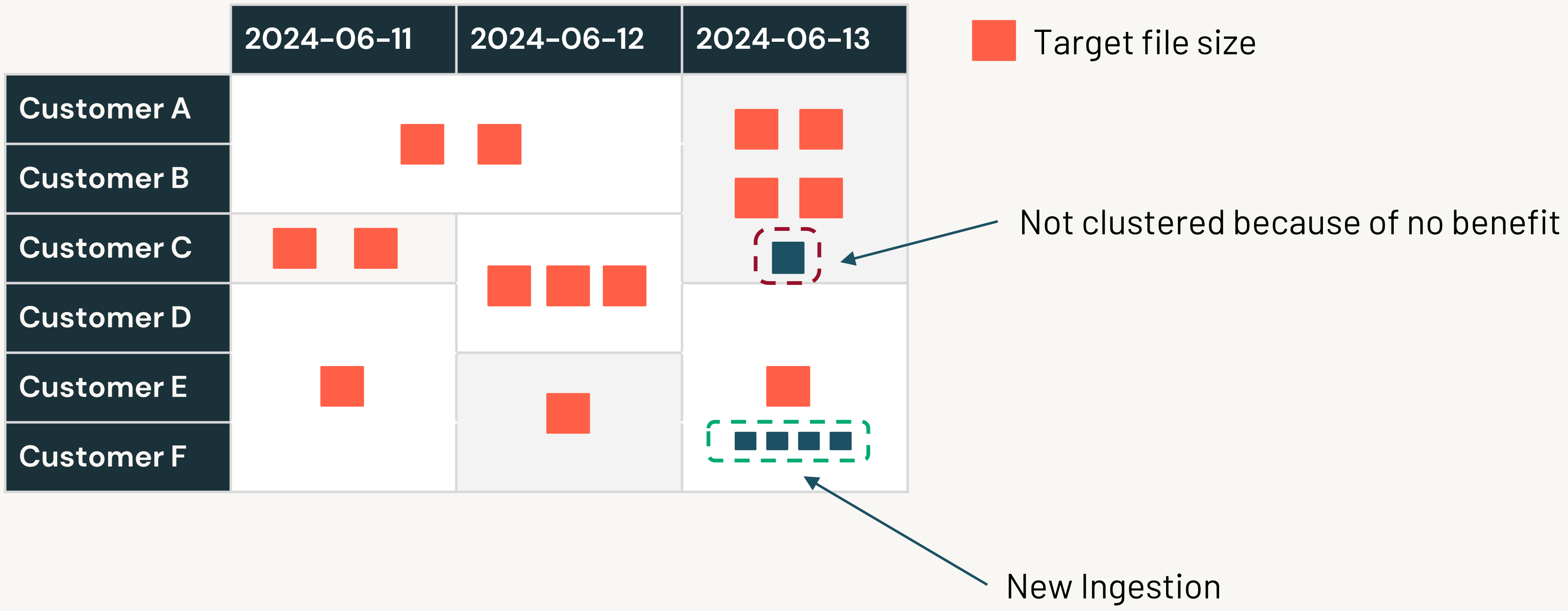
Incremental clustering



Already well clustered files are not touched

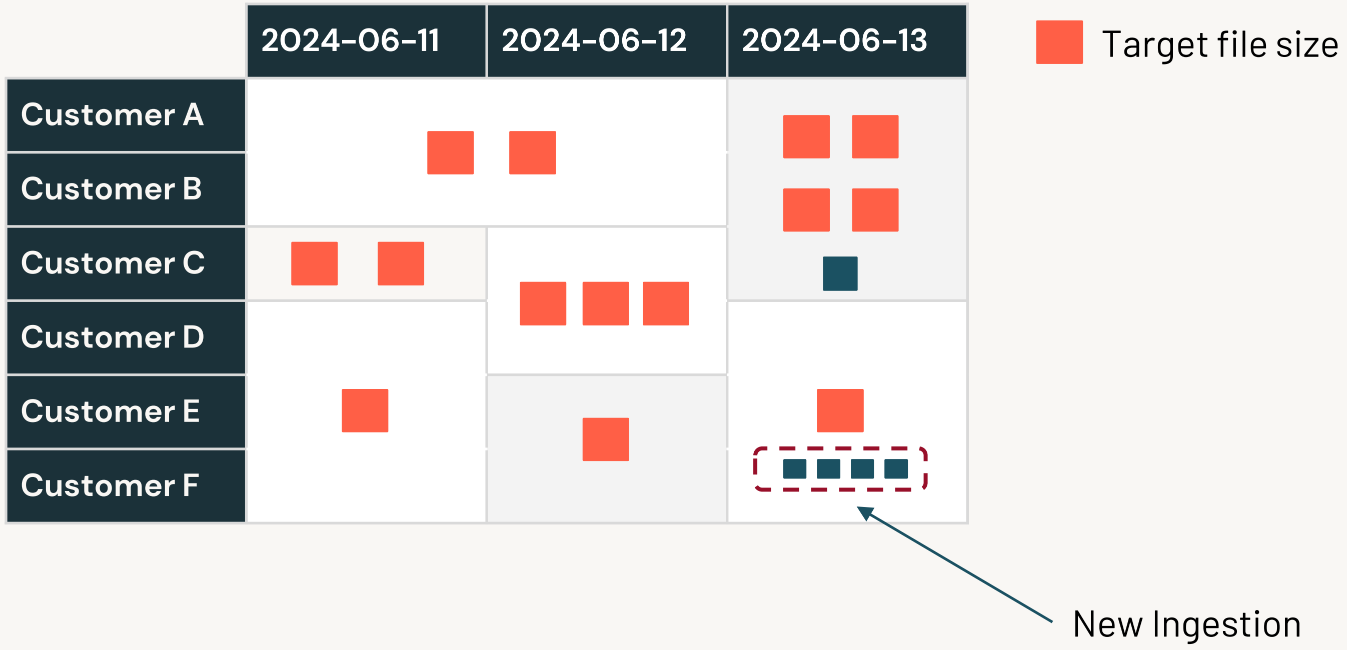
Liquid Clustering

Incremental clustering



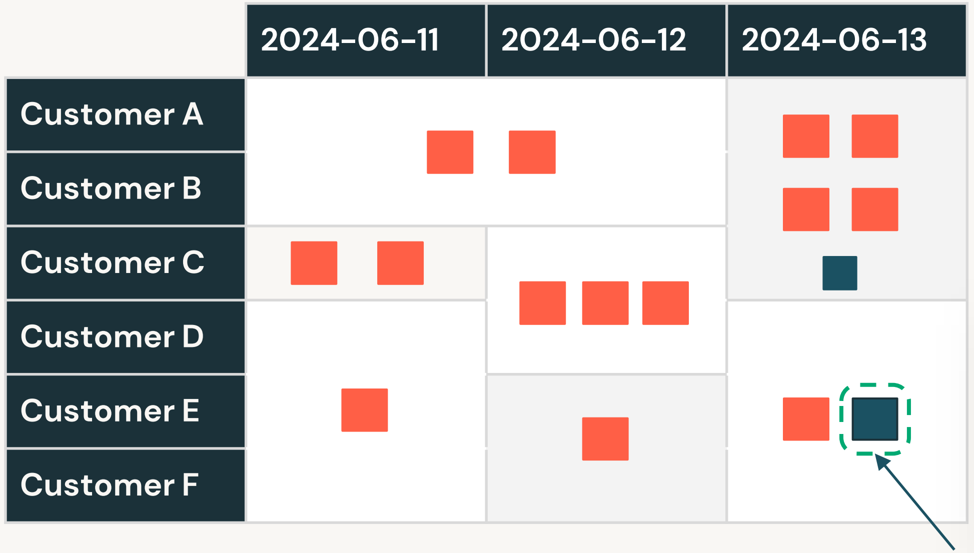
Liquid Clustering

Incremental clustering



Liquid Clustering

Incremental clustering



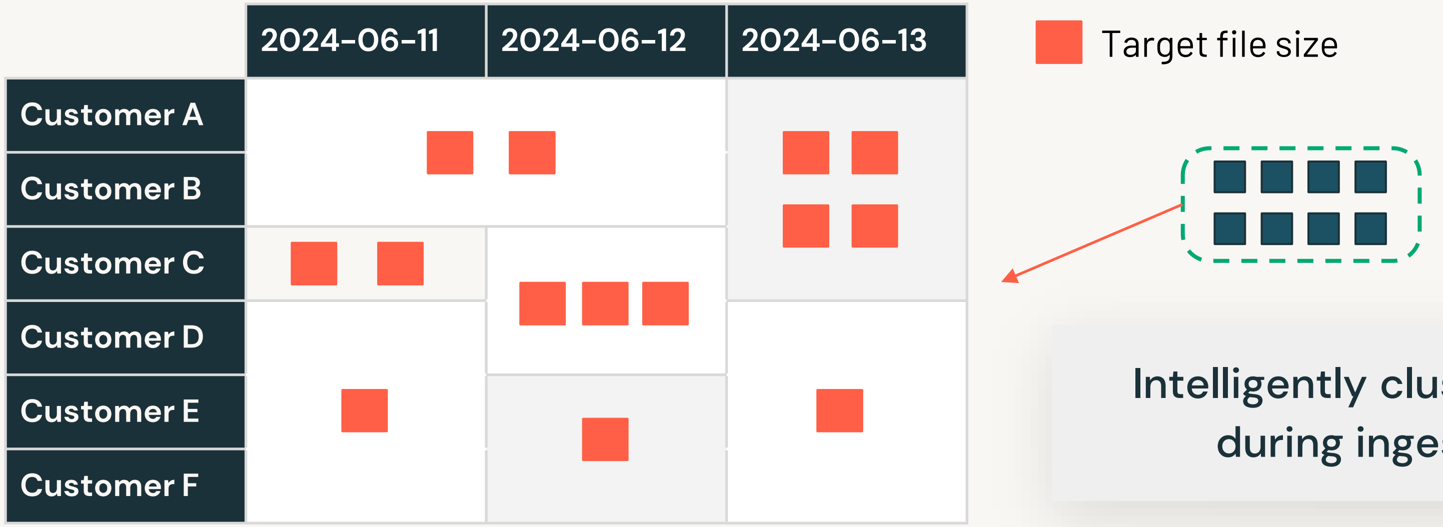
🟥 Target file size

Intelligently selects files for incremental clustering:

- Fast: cluster only necessary data

Liquid Clustering

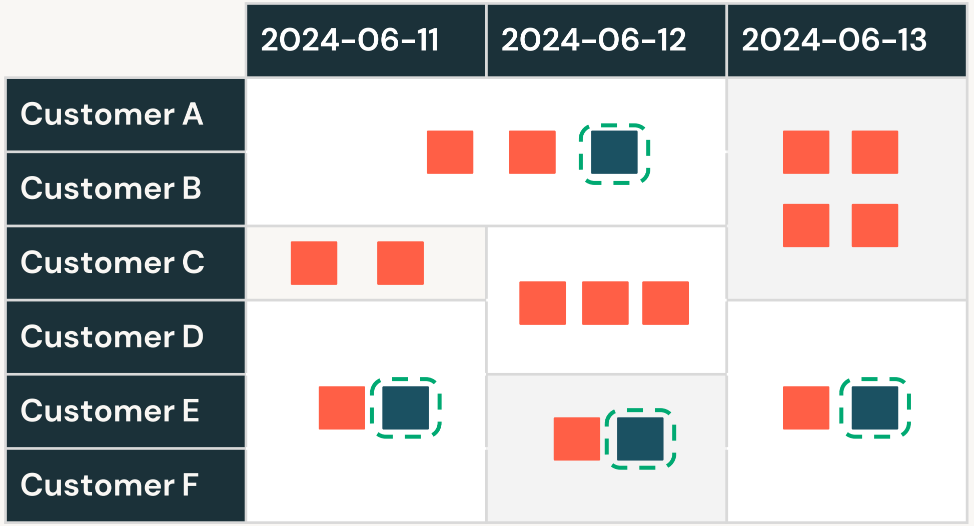
Clustering on write



Intelligently cluster data during ingestion

Liquid Clustering

Clustering on write



🟥 Target file size

Intelligently cluster data during ingestion

- No write amplification
- Good clustering right after ingestion

How to make a Delta table go fast

```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
CREATE TABLE tbl1 PARTITION BY date AS <query>

-- further ZORDER by columns within the partition
-- decision: what do you partition by vs. ZORDER by?
-- run this regularly!
OPTIMIZE tbl1 ZORDER BY customerId;

-- gather statistics
-- ANALYZE regularly!
ALTER TABLE tbl1
SET TBLPROPERTIES (‘delta.dataSkippingNumIndexedCols” = 64)’;
ANALYZE tbl1; VACUUM tbl1;

-- if query patterns change, redo all the above
```

Partitioning: used to avoid concurrent write conflicts.

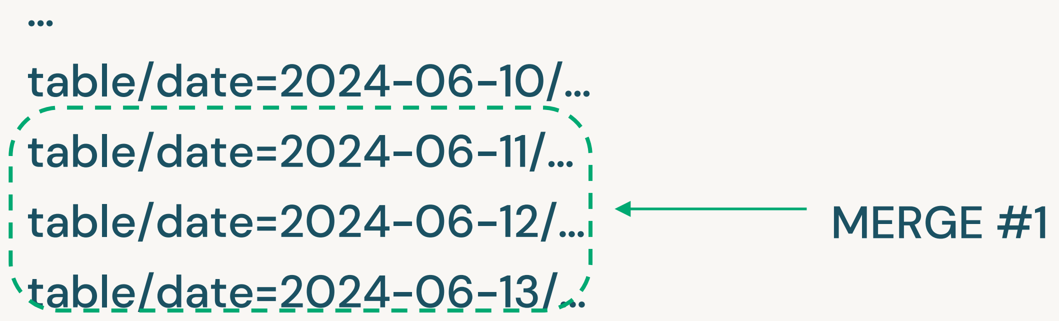
Hive-style partitioning

Conflict Resolution

```
...
table/date=2024-06-10/...
table/date=2024-06-11/...
table/date=2024-06-12/...
table/date=2024-06-13/...
```

Hive-style partitioning

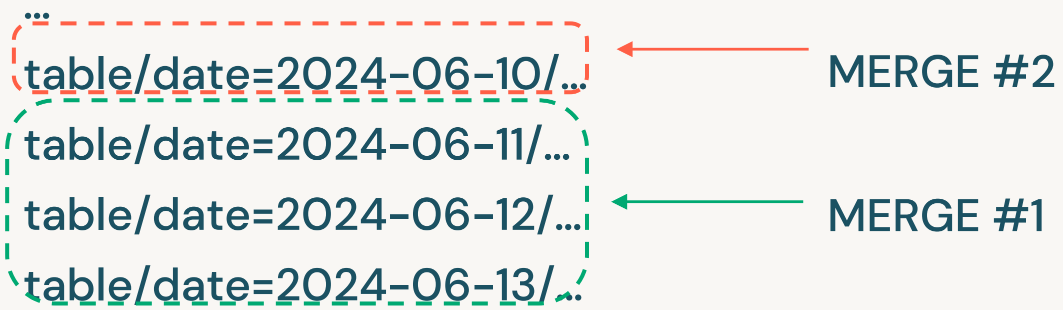
Conflict Resolution



```
...
table/date=2024-06-10/...
table/date=2024-06-11/...
table/date=2024-06-12/...
table/date=2024-06-13/...
```

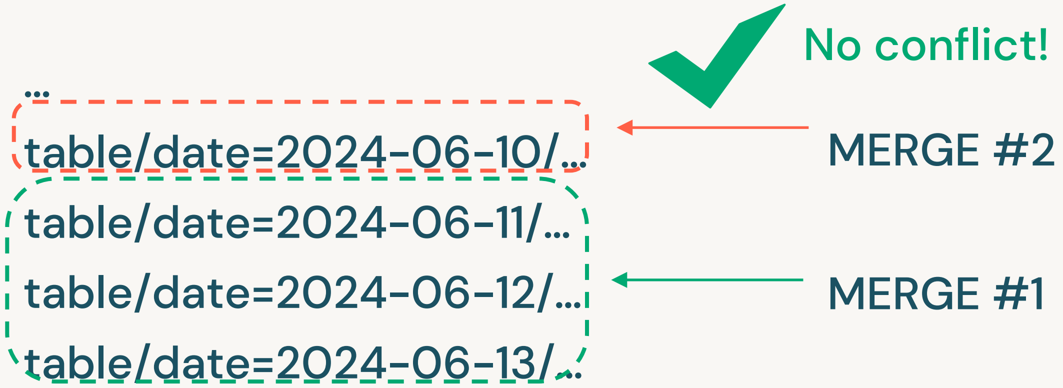
Hive-style partitioning

Conflict Resolution


...
table/date=2024-06-10/...
table/date=2024-06-11/...
table/date=2024-06-12/...
table/date=2024-06-13/...
MERGE #2
MERGE #1

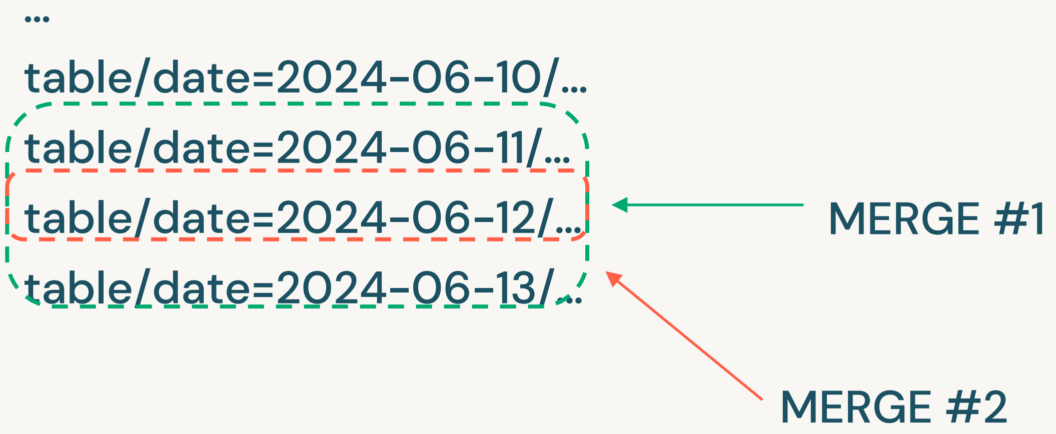
Hive-style partitioning

Conflict Resolution


...
table/date=2024-06-10/...
table/date=2024-06-11/...
table/date=2024-06-12/...
table/date=2024-06-13/...
No conflict!
MERGE #2
MERGE #1

Hive-style partitioning

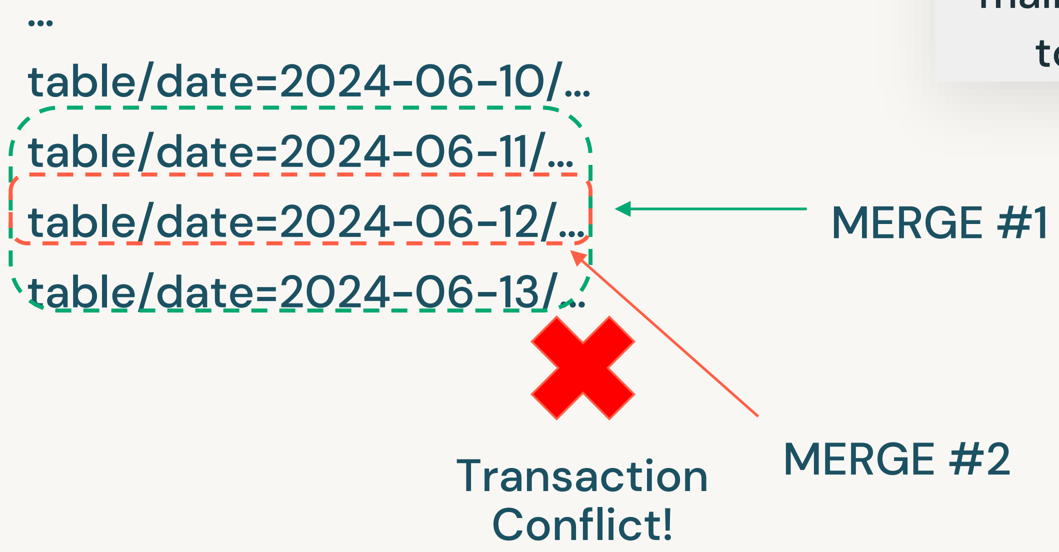
Conflict Resolution



...
table/date=2024-06-10/...
table/date=2024-06-11/...
table/date=2024-06-12/...
table/date=2024-06-13/...

Hive-style partitioning

Conflict Resolution


...
table/date=2024-06-10/...
table/date=2024-06-11/...
table/date=2024-06-12/...
table/date=2024-06-13/...
MERGE #1
MERGE #2
Transaction Conflict!

Need to orchestrate business and/or maintenance transactions carefully to avoid transaction conflicts!

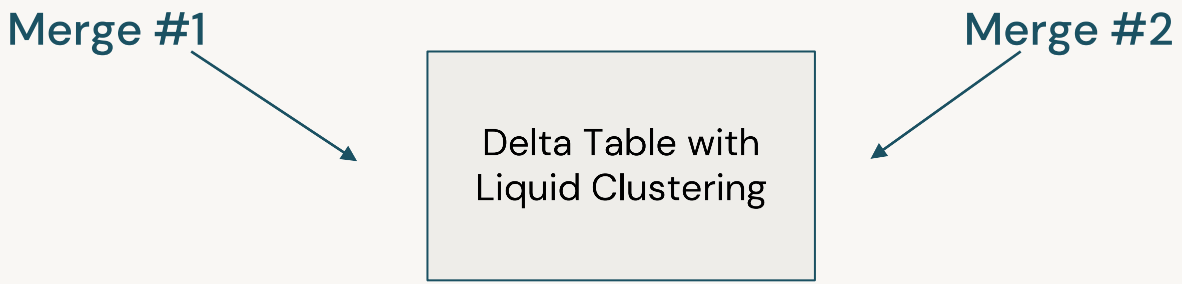
Liquid Clustering

Conflict Resolution

Delta Table with Liquid Clustering

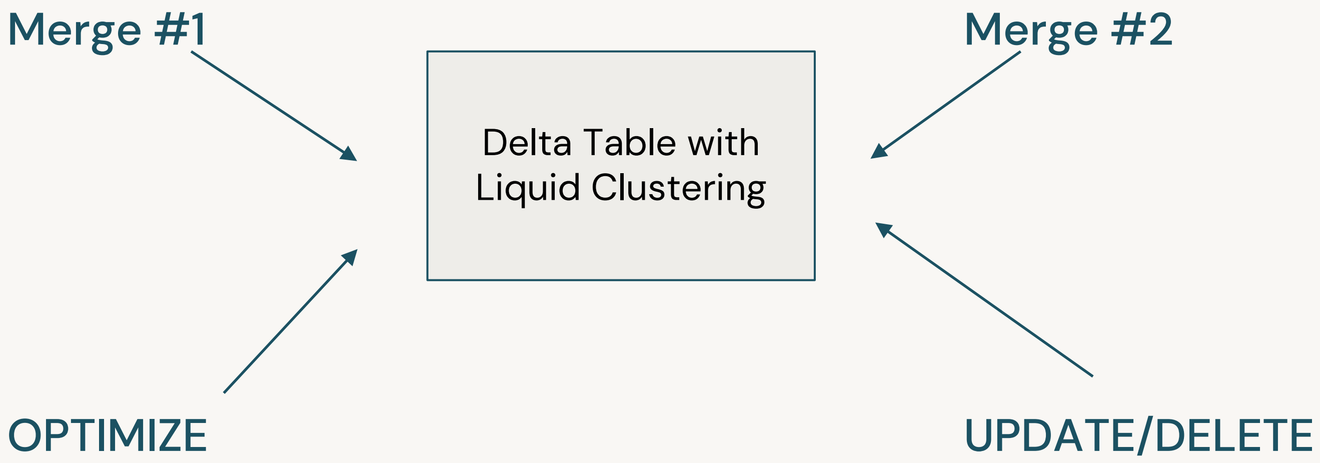
Liquid Clustering

Conflict Resolution



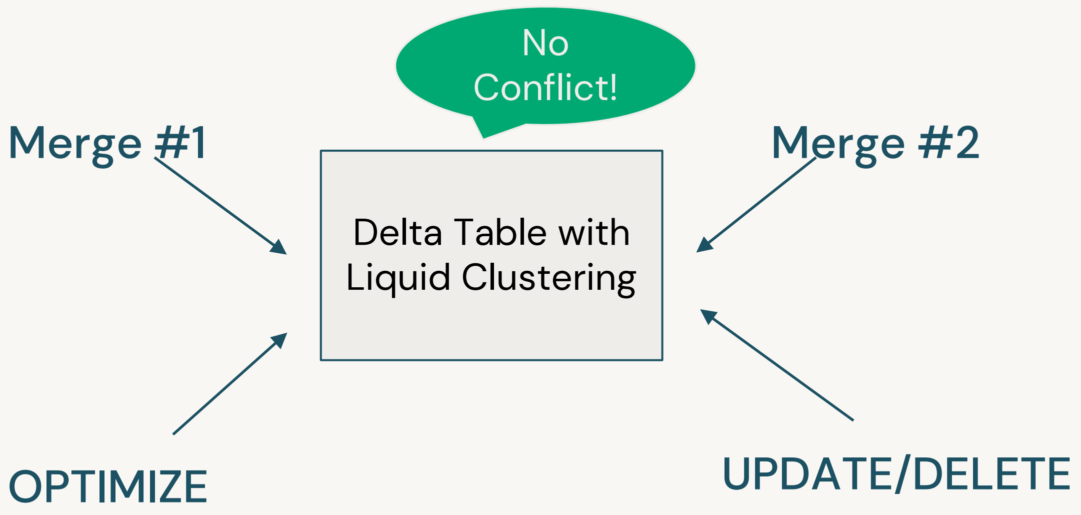
Liquid Clustering

Conflict Resolution



Liquid Clustering

Conflict Resolution



Row Level Concurrency powered by Data Intelligence

- **No conflict** as long as different rows are updated.
- **No more hassle** for orchestrating business and/or maintenance transactions

Liquid is easy to use

```
CREATE TABLE prod.sales_schema.sales
CLUSTER BY timestamp, customer_id
AS ...
```

- Choose clustering columns regardless of **cardinality** or **skew.**

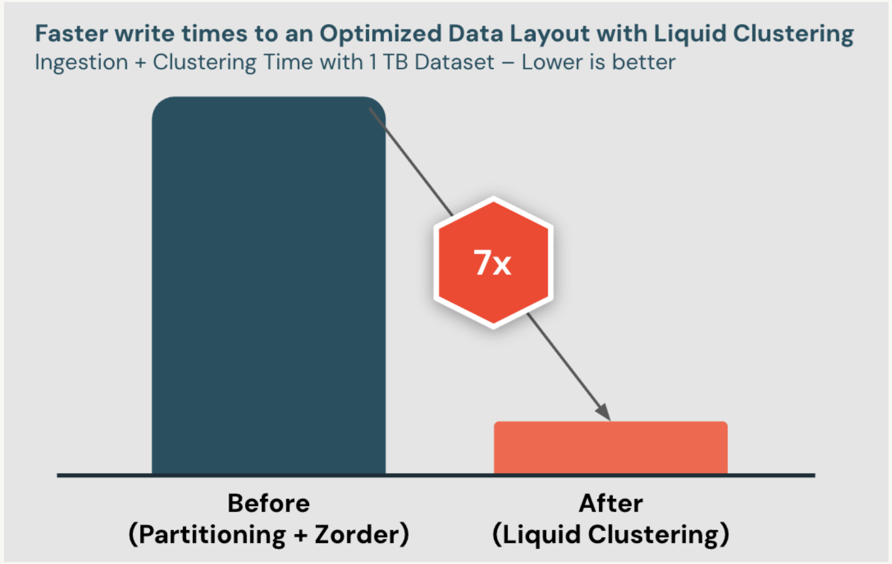
Liquid is easy to use

```
CREATE TABLE prod.sales_schema.sales
CLUSTER BY timestamp, customer_id
AS ...
```

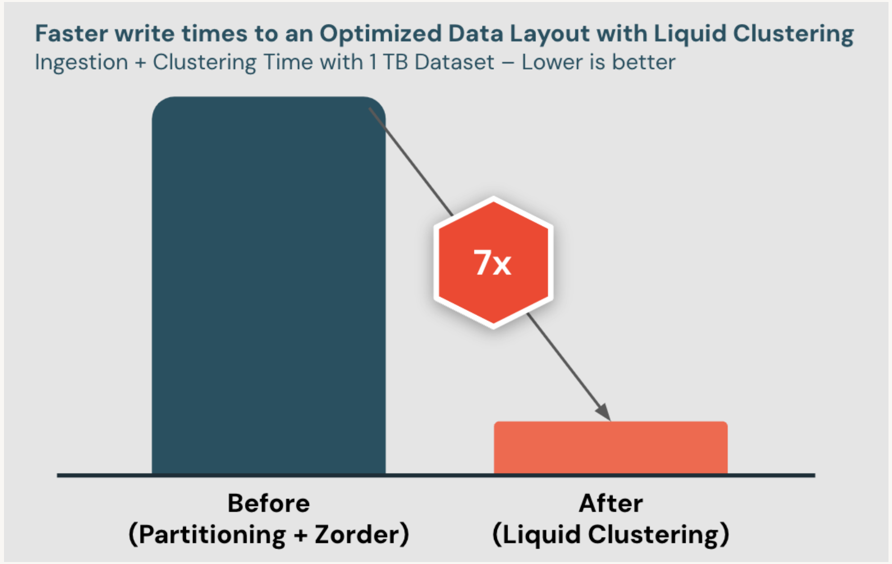
- Choose clustering columns regardless of **cardinality** or **skew.**
- If query patterns change, **easily change clustering columns:**

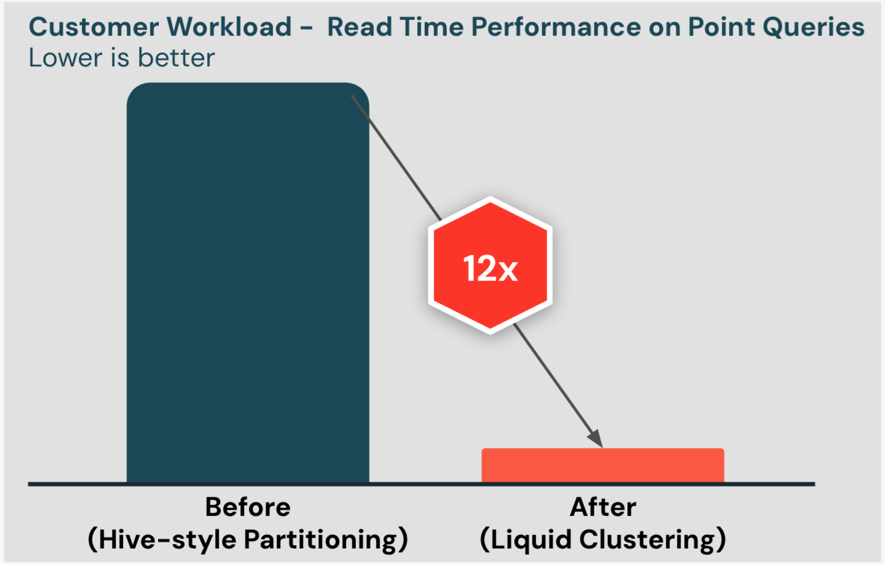
```
ALTER TABLE prod.sales_schema.sales
CLUSTER BY timestamp, sales_territory, account_id
```

Liquid is fast



Liquid is fast





Liquid is widely adopted

Liquid is widely adopted

1200+

Customers using Liquid clustering **weekly**

Liquid is widely adopted

1200+

Customers using Liquid clustering **weekly**

600 TB+

Largest Liquid table

Liquid is widely adopted

1200+	600 TB+	100 PB+	20 EB+
Customers using Liquid clustering **weekly**	Largest Liquid table	Data written to Liquid tables	Data scanned from Liquid tables

Liquid is widely adopted

1200+

Customers using Liquid clustering **weekly**

600 TB+

Largest Liquid table

100 PB+

Data written to Liquid tables

20 EB+

Data scanned from Liquid tables

1.1 ZB+

Data **pruned** by Liquid tables

Liquid is widely adopted

1200+

Customers using Liquid clustering **weekly**

600 TB+

Largest Liquid table

100 PB+

Data written to Liquid tables

20 EB+

Data scanned from Liquid tables

1.1 ZB+ (1100 EB+)

Data **pruned** by Liquid tables

Liquid is widely adopted

"Delta Lake Liquid Clustering **improved our time series queries up to 10x and was remarkably simple to implement** on our Lakehouse. It allows us to cluster on columns **without worrying about cardinality or file size and significantly reduces the amount of data it needs to read** – something we have always had to manage ourselves with Delta partitioning and z-order fine-tuning."

- **Bryce Bartmann** (Chief Digital Technology Advisor, Shell)

"Using Databricks innovative Liquid Clustering, we have **observed remarkable improvements in query performance** compared to the traditional z-order methods. Additionally, Liquid clustered tables have **streamlined our data processing by eliminating partitioning bottlenecks**, improving scanning, and **reducing data skews**."

- **Edward Goo** (Director of ETL Engineering, YipitData)

"Liquid clustering has greatly improved the ability of our researchers to investigate complex datasets for specific trends and events. We look forward to watching this feature grow and be adopted as a key feature of the Delta ecosystem."

- **Robert Batts** (Big Data Lead, Cisco)

Liquid is widely adopted

"Delta Lake Liquid Clustering **improved our time series queries up to 10x and was remarkably simple to implement** on our Lakehouse. It allows us to cluster on columns **without worrying about cardinality or file size and significantly reduces the amount of data it needs to read** – something we have always had to manage ourselves with Delta partitioning and z-order fine-tuning."

- **Bryce Bartmann** (Chief Digital Technology Advisor, Shell)

"Using Databricks innovative Liquid Clustering, we have **observed remarkable improvements in query performance** compared to the traditional z-order methods. Additionally, Liquid clustered tables have **streamlined our data processing by eliminating partitioning bottlenecks**, improving scanning, and **reducing data skews**."

- **Edward Goo** (Director of ETL Engineering, YipitData)

"**Liquid clustering has greatly improved the ability of our researchers to investigate complex datasets for specific trends and events**. We look forward to watching this feature grow and be adopted as a key feature of the Delta ecosystem."

- **Robert Batts** (Big Data Lead, Cisco)

Liquid is widely adopted

"Delta Lake Liquid Clustering **improved our time series queries up to 10x and was remarkably simple to implement** on our Lakehouse. It allows us to cluster on columns **without worrying about cardinality or file size and significantly reduces the amount of data it needs to read** – something we have always had to manage ourselves with Delta partitioning and z-order fine-tuning."

- **Bryce Bartmann** (Chief Digital Technology Advisor, Shell)

"Using Databricks innovative Liquid Clustering, we have **observed remarkable improvements in query performance** compared to the traditional z-order methods. Additionally, Liquid clustered tables have **streamlined our data processing by eliminating partitioning bottlenecks**, improving scanning, and **reducing data skews.**"

- **Edward Goo** (Director of ETL Engineering, YipitData)

"**Liquid clustering has greatly improved the ability of our researchers to investigate complex datasets for specific trends and events**. We look forward to watching this feature grow and be adopted as a key feature of the Delta ecosystem."

- **Robert Batts** (Big Data Lead, Cisco)

How to make a Delta table go fast

```
-- decision: what columns do I pick?
-- decision: is the cardinality correct?
-- Liquid!
CREATE TABLE tbl1 CLUSTER BY date, customerId AS <query>;

-- further ZORDER by columns within the partition
-- decision: what do you partition by vs. ZORDER by?
-- run this regularly!
OPTIMIZE tbl1 ZORDER BY customerId;

-- gather statistics
-- ANALYZE regularly!
ALTER TABLE tbl1
SET TBLPROPERTIES (‘delta.dataSkippingNumIndexedCols” = 64)’;
ANALYZE tbl1;

-- don’t forget to OPTIMIZE and VACUUM
OPTIMIZE tbl1; VACUUM tbl1;

-- if query patterns change, redo all the above
```

How to make a Delta table go fast

```
-- Liquid!
CREATE TABLE tbl1 CLUSTER BY date, customerId AS <query>;

-- gather statistics
-- ANALYZE regularly!
ALTER TABLE tbl1
SET TBLPROPERTIES (‘delta.dataSkippingNumIndexedCols” = 64)’;
ANALYZE tbl1;

-- don’t forget to OPTIMIZE and VACUUM
OPTIMIZE tbl1; VACUUM tbl1;

-- if query patterns change, redo all the above
```

How to make a Delta table go fast

```
-- Liquid!
CREATE TABLE tbl1 CLUSTER BY date, customerId AS <query>;

-- gather statistics
-- ANALYZE regularly!
ALTER TABLE tbl1
SET TBLPROPERTIES ('delta.dataSkippingNumIndexedCols" = 64)';
ANALYZE tbl1;

-- don't forget to OPTIMIZE and VACUUM
OPTIMIZE tbl1; VACUUM tbl1;

-- if query patterns change, redo all the above
```

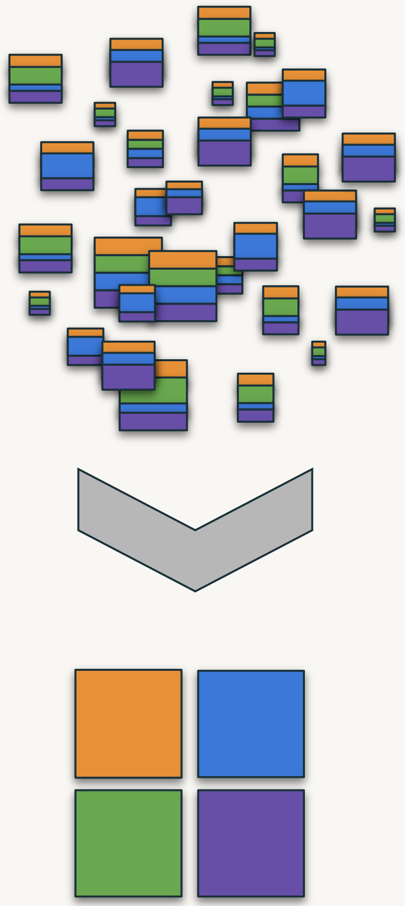
[#3] Continuous maintenance

Predictive Optimization (GA)

Challenge

Tables can be optimized for better price-performance, but...

- Which optimizations?
- Which tables?
- How often?



Introducing...

Predictive Optimization

Solution

- Intelligence engine determines which tables to optimize
- Databricks automatically performs optimizations

Optimizations

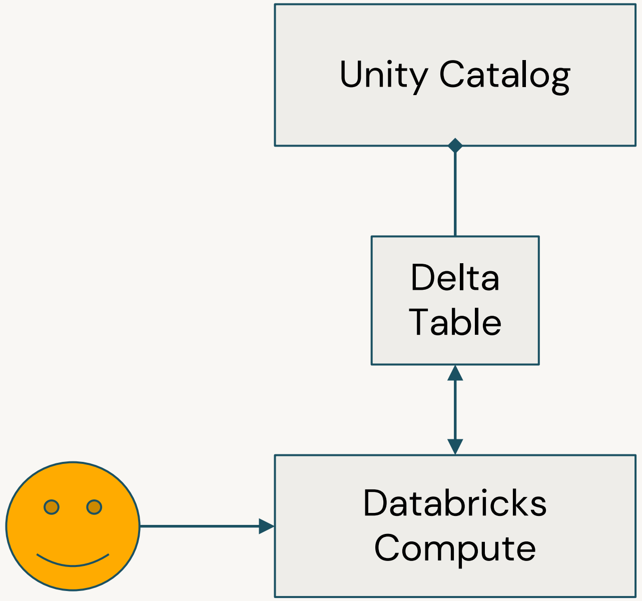
- Compaction
- Liquid clustering
- Garbage collection (VACUUM)

Predictive Optimization

High-level Architecture

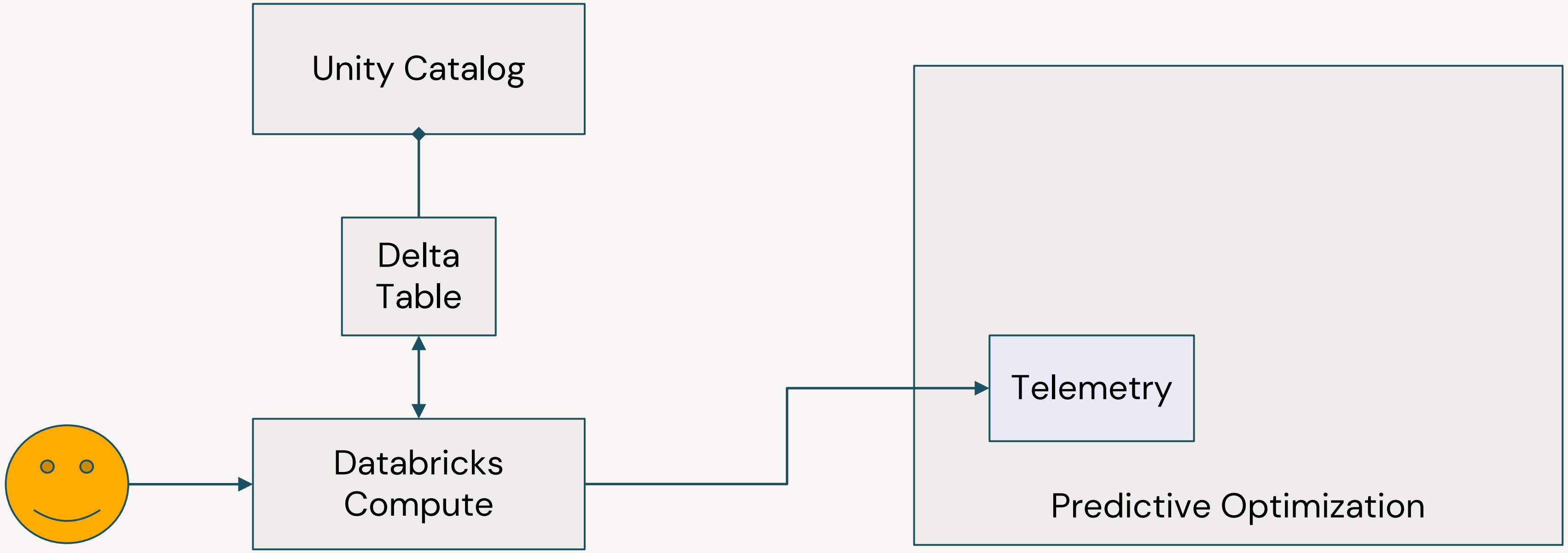
Predictive Optimization

High-level Architecture



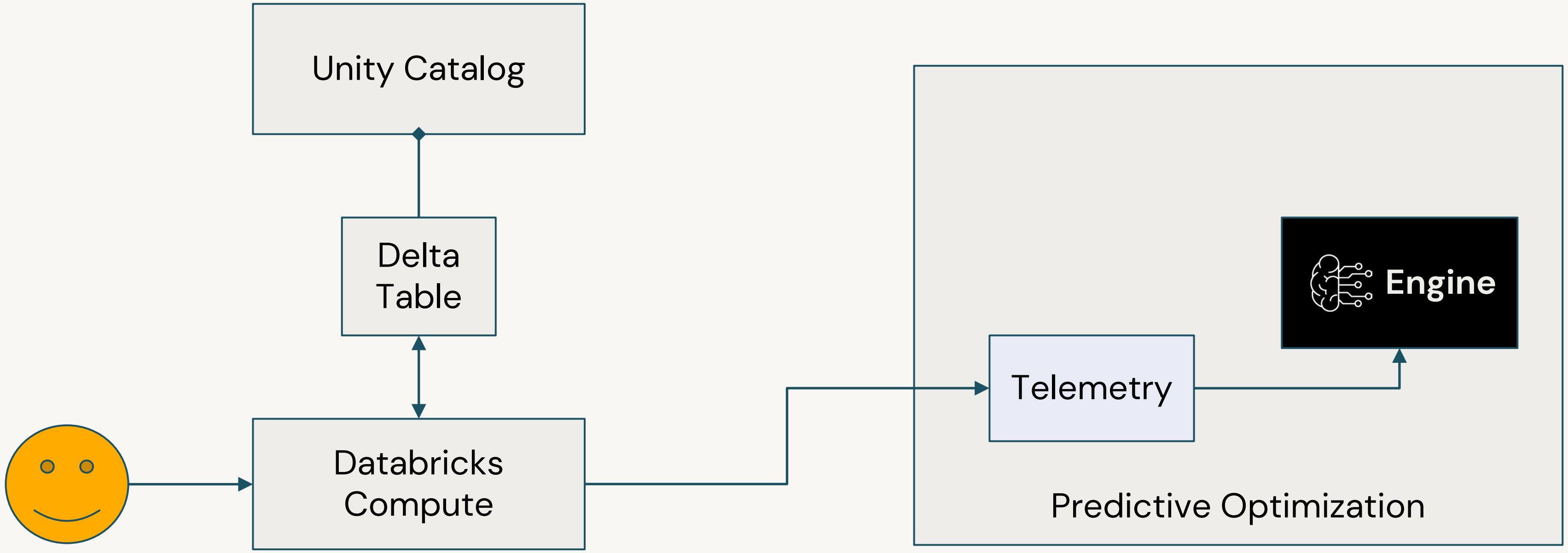
Predictive Optimization

High-level Architecture



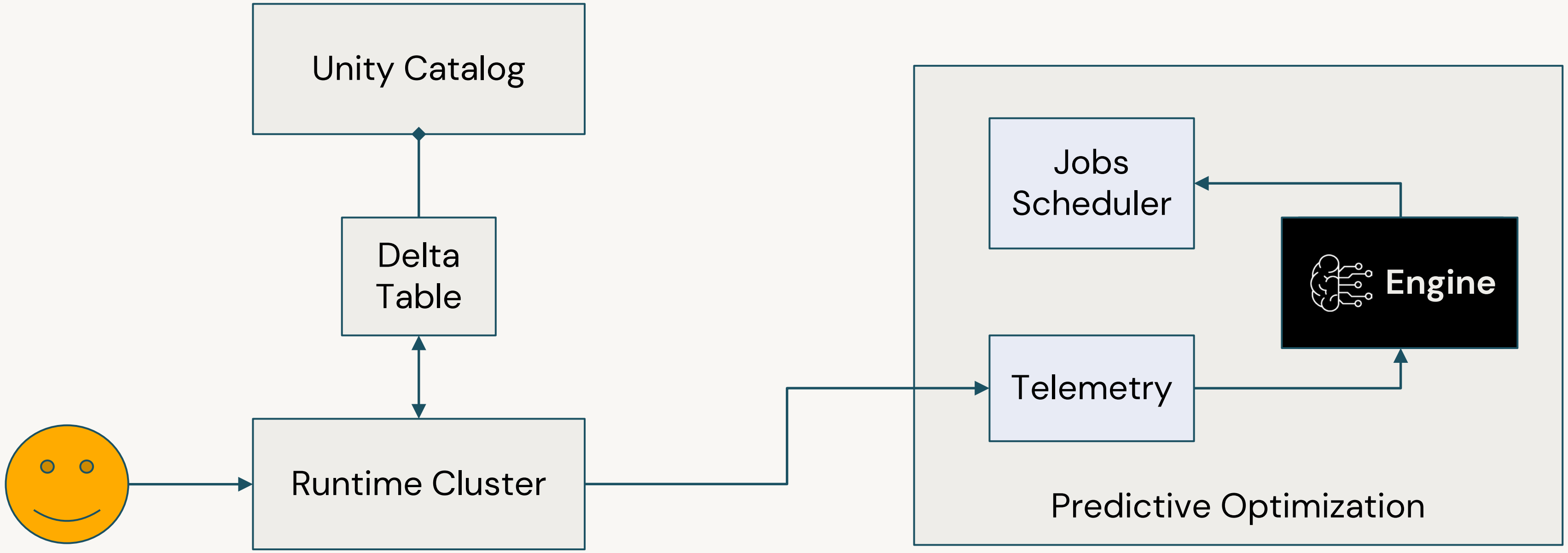
Predictive Optimization

High-level Architecture



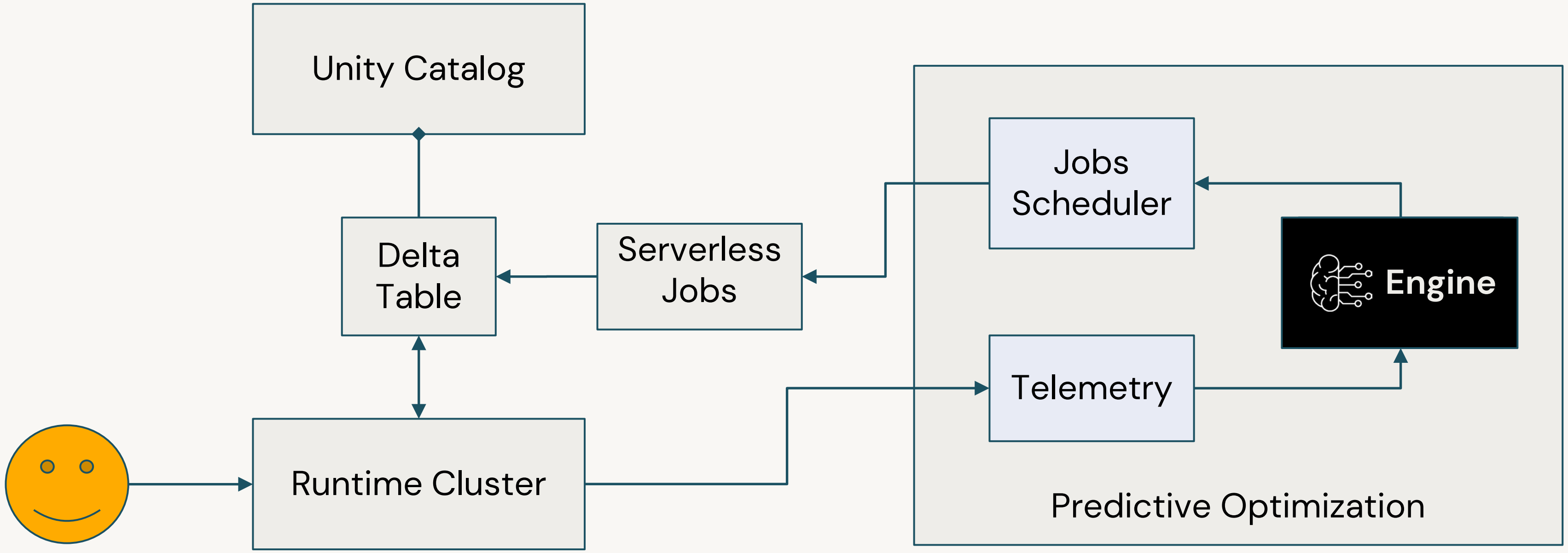
Predictive Optimization

High-level Architecture



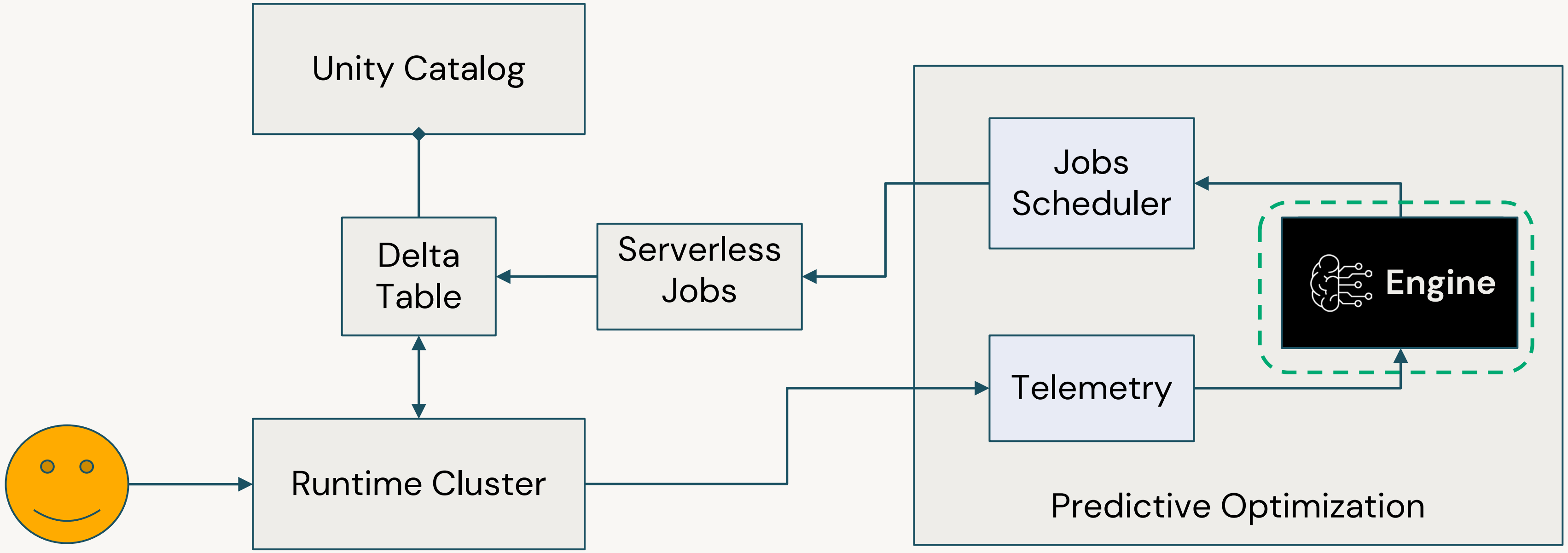
Predictive Optimization

High-level Architecture



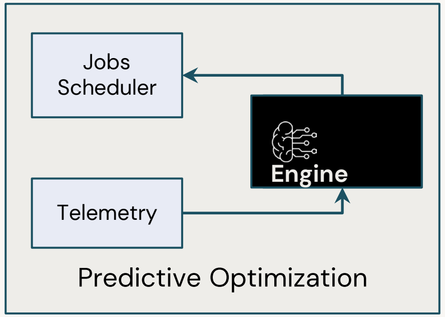
Predictive Optimization

High-level Architecture



Predictive Optimization

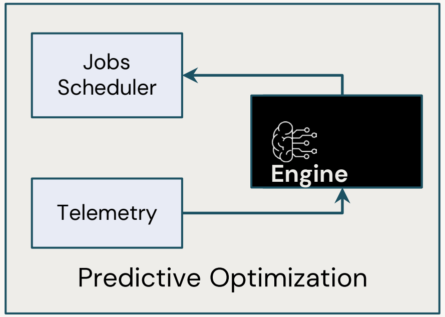
Intelligence Engine in Action (Liquid Clustering)



Determine <u>clustering</u> return-on-investment (ROI)

Predictive Optimization

Intelligence Engine in Action (Liquid Clustering)



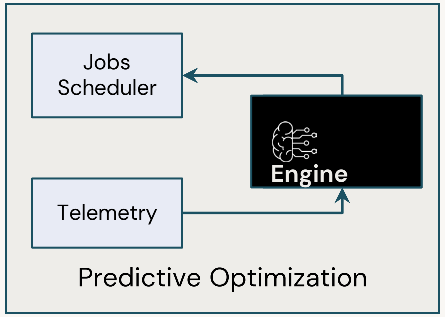
Determine clustering return-on-investment (ROI)

Telemetry

- Table usage / query patterns
- Clustering Quality
- Clustered / Nonclustered Bytes

Predictive Optimization

Intelligence Engine in Action (Liquid Clustering)



Determine clustering return-on-investment (ROI)

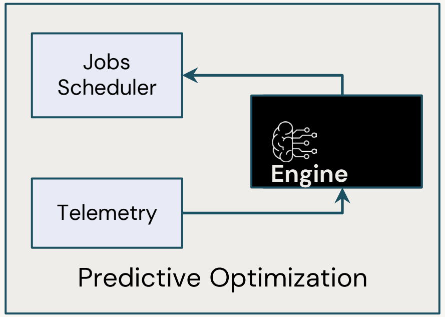
Telemetry

- Table usage / query patterns
- Clustering Quality
- Clustered / Nonclustered Bytes

Table	Query speedup	Clustering cost	Table usage
1	HIGH	LOW	LOW
2	HIGH	LOW	HIGH
3	LOW	LOW	HIGH
4	HIGH	HIGH	HIGH

Predictive Optimization

Intelligence Engine in Action (Liquid Clustering)



Determine clustering return-on-investment (ROI)

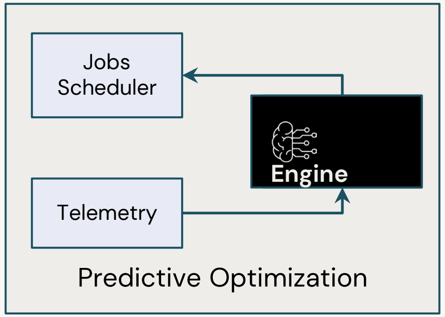
Telemetry

- Table usage / query patterns
- Clustering Quality
- Clustered / Nonclustered Bytes

Table	Query speedup	Clustering cost	Table usage	Expected ROI
1	HIGH	LOW	LOW	LOW
2	HIGH	LOW	HIGH	HIGH
3	LOW	LOW	HIGH	LOW
4	HIGH	HIGH	HIGH	MEDIUM

Predictive Optimization

Intelligence Engine in Action (Liquid Clustering)



Determine clustering return-on-investment (ROI)

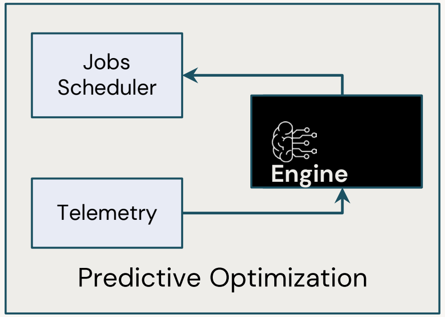
Telemetry

- Table usage / query patterns
- Clustering Quality
- Clustered / Nonclustered Bytes

Table	Query speedup	Clustering cost	Table usage	Expected ROI
2	HIGH	LOW	HIGH	HIGH
4	HIGH	HIGH	HIGH	MEDIUM
Skip everything below this!				
1	HIGH	LOW	LOW	LOW
3	LOW	LOW	HIGH	LOW

Predictive Optimization

Intelligence Engine in Action (VACUUM)



Determine VACUUM return-on-investment (ROI)

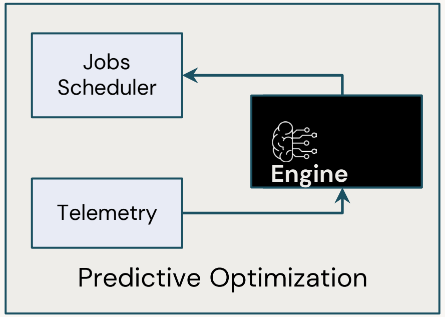
Telemetry

- Commit patterns (bytes added/ removed)
- Retention window
- Table metadata

Tabl e	VACUUMable data	VACUUM cost
1	HIGH	LOW
2	LOW	LOW
3	HIGH	HIGH
4	LOW	HIGH

Predictive Optimization

Intelligence Engine in Action (VACUUM)



Determine VACUUM return-on-investment (ROI)

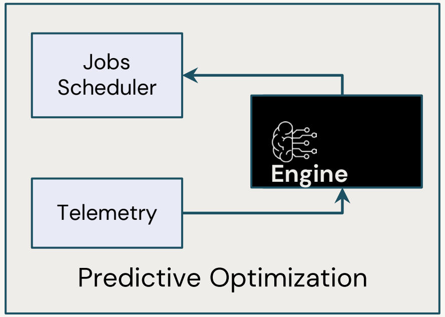
Telemetry

- Commit patterns (bytes added/ removed)
- Retention window
- Table metadata

Table	VACUUMable data	VACUUM cost	Expected ROI
1	HIGH	LOW	HIGH
2	LOW	LOW	LOW
3	HIGH	HIGH	MEDIUM
4	LOW	HIGH	LOW

Predictive Optimization

Intelligence Engine in Action (VACUUM)



Determine VACUUM return-on-investment (ROI)

Telemetry

- Commit patterns (bytes added/ removed)
- Retention window
- Table metadata

Table	VACUUMable data	VACUUM cost	Expected ROI
1	HIGH	LOW	HIGH
3	HIGH	HIGH	MEDIUM
Skip everything below this!			
3	HIGH	HIGH	MEDIUM
4	LOW	HIGH	LOW

Predictive Optimization

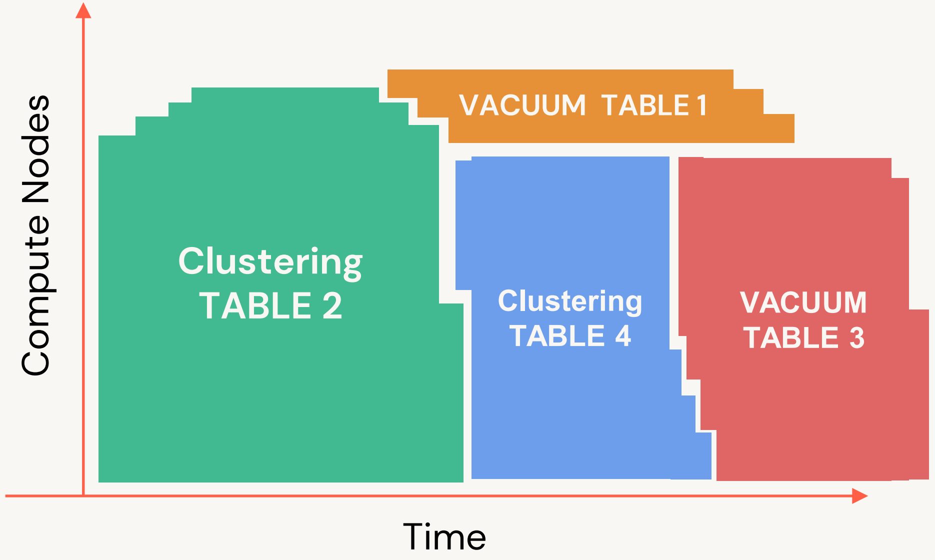
Intelligence Engine in Action (Scheduling)

Table	Operation	Expected ROI
2	Clustering	High
1	VACUUM	High
4	Clustering	Medium
3	VACUUM	Medium

Predictive Optimization

Intelligence Engine in Action (Scheduling)

Table	Operation	Expected ROI
2	Clustering	High
1	VACUUM	High
4	Clustering	Medium
3	VACUUM	Medium



Predictive Optimization is widely adopted

1.5K+

Customers using Predictive Optimization

5 EB+

Data under management

2 PB+

Data optimized per day

Predictive Optimization is widely adopted

"Databricks' Predictive Optimizations intelligently optimized our Unity Catalog storage, which **saved us 50% in annual storage costs while speeding up our queries by >2x.** It learned to prioritize our largest and most-accessed tables. And, it did all of this automatically, saving our team valuable time."

- **Shu Li** (Data Engineering Lead, Anker)

Predictive Optimization is widely adopted

"Databricks' Predictive Optimizations intelligently optimized our Unity Catalog storage, which **saved us 50% in annual storage costs while speeding up our queries by >2x.** It learned to prioritize our largest and most-accessed tables. And, it did all of this automatically, saving our team valuable time."

- **Shu Li** (Data Engineering Lead, Anker)

ANKER

"Databricks Predictive Optimization consistently helps the FinOps group minimize storage costs. **We've immediately seen a 26% drop in storage costs,** and we expect additional incremental savings going forward. The capability has enabled us to retire procedures, scripts, and manual maintenance operations, **allowing us to achieve greater out-of-the-box scalability.**"

- – **Alessandro Caronia,** Infrastructure Operations Manager and **Simona Fiazza**, End to End Operations Manager

plenitude

Predictive Optimization is widely adopted

"Databricks' Predictive Optimizations intelligently optimized our Unity Catalog storage, which **saved us 50% in annual storage costs while speeding up our queries by >2x.** It learned to prioritize our largest and most-accessed tables. And, it did all of this automatically, saving our team valuable time."

- **Shu Li** (Data Engineering Lead, Anker)

"Databricks Predictive Optimization consistently helps the FinOps group minimize storage costs. **We've immediately seen a 26% drop in storage costs,** and we expect additional incremental savings going forward. The capability has enabled us to retire procedures, scripts, and manual maintenance operations, **allowing us to achieve greater out-of-the-box scalability.**"

- – **Alessandro Caronia,** Infrastructure Operations Manager and **Simona Fiazza**, End to End Operations Manager

"Thanks to Predictive Optimization (PO), we were able to **decommission our DIY solution for table maintenance.** PO is **more efficient and cost-effective, as it optimizes only the tables that benefit from maintenance operations.** PO simplifies our data platform, allowing for better allocation of resources and a more streamlined data management process."

- **Nikita Bochkarev**, Senior Data Engineer at Toloka AI

Automatic Statistics

With Predictive Optimization

Private Preview

Challenges

- For Query Optimization stats, need to run ANALYZE
- For Delta stats, first 32-columns, are they the right ones?

Automatic Statistics

With Predictive Optimization

Private Preview

Challenges

- For Query Optimization stats, need to run ANALYZE
- For Delta stats, first 32-columns, are they the right ones?

With Automatic Statistics

- Intelligently determine which columns to collect Delta stats for
- QO stats collected and maintained automatically

Automatic Statistics

With Predictive Optimization

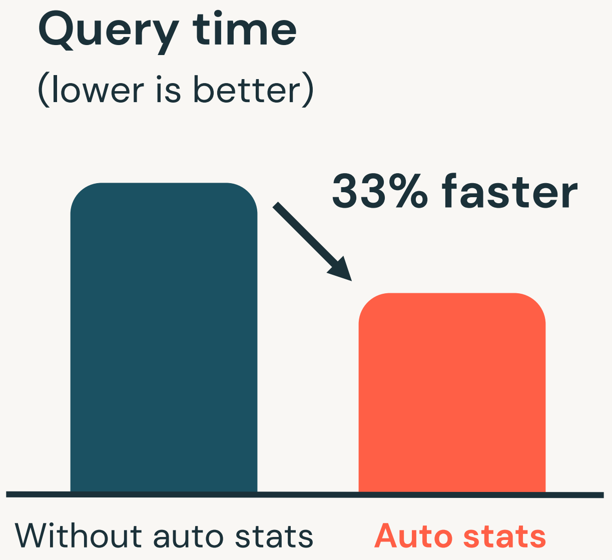
Private Preview

Challenges

- For Query Optimization stats, need to run ANALYZE
- For Delta stats, first 32-columns, are they the right ones?

With **Automatic Statistics**

- Intelligently determine which columns to collect Delta stats for
- QO stats collected and maintained automatically



Contact your Account Team for Preview and early on-boarding!

How to make a Delta table go fast

```
-- Liquid!
CREATE TABLE tbl1 CLUSTER BY date, clusterId AS <query>;

-- gather statistics
-- ANALYZE regularly!
ALTER TABLE tbl1
SET TBLPROPERTIES (‘delta.dataSkippingNumIndexedCols” = 64)’;
ANALYZE tbl1;

-- don’t forget to OPTIMIZE and VACUUM
OPTIMIZE tbl1; VACUUM tbl1;

-- easily update clustering keys if query patterns change
```

How to make a Delta table go fast

```
-- Create Liquid table...
CREATE TABLE tbl1 CLUSTER BY date, clusterId AS <query>;

-- easily update clustering keys if query patterns change
```

Automatic Liquid Clustering Key Selection

```
> CREATE TABLE tbl CLUSTER BY AUTO
```

Fully automated:

- Clustering key selection
- Clustering on write
- Background clustering

Automatic Liquid Clustering Key Selection

```
> CREATE TABLE tbl CLUSTER BY AUTO
```

Liquid clustering powered by the Data Intelligence Engine

Telemetry
What are your workloads' query patterns?

Fully automated:

- Clustering key selection
- Clustering on write
- Background clustering

Automatic Liquid Clustering Key Selection

```
> CREATE TABLE tbl CLUSTER BY AUTO
```

Liquid clustering powered by the Data Intelligence Engine

Telemetry
What are your workloads' query patterns?

↓

Model evaluation
How should Liquid clustering configuration be updated?

Fully automated:

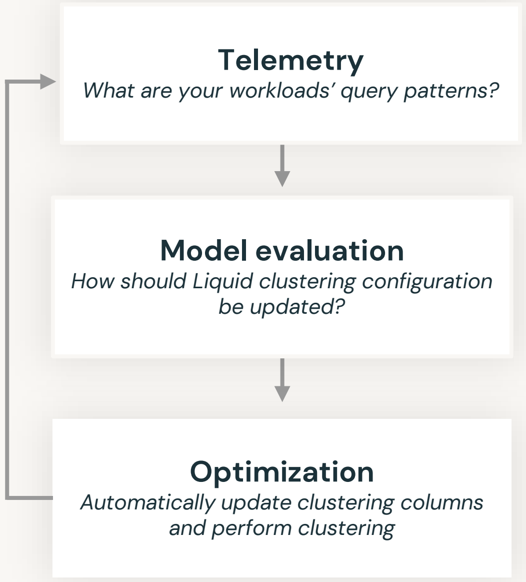
- Clustering key selection
- Clustering on write
- Background clustering

©2024 Databricks Inc. — All rights reserved

Automatic Liquid Clustering Key Selection

```
> CREATE TABLE tbl CLUSTER BY AUTO
```

Liquid clustering powered by the Data Intelligence Engine


Telemetry
What are your workloads' query patterns?
Model evaluation
How should Liquid clustering configuration be updated?
Optimization
Automatically update clustering columns and perform clustering

Fully automated:

- Clustering key selection
- Clustering on write
- Background clustering

Automatic Liquid Clustering Key Selection

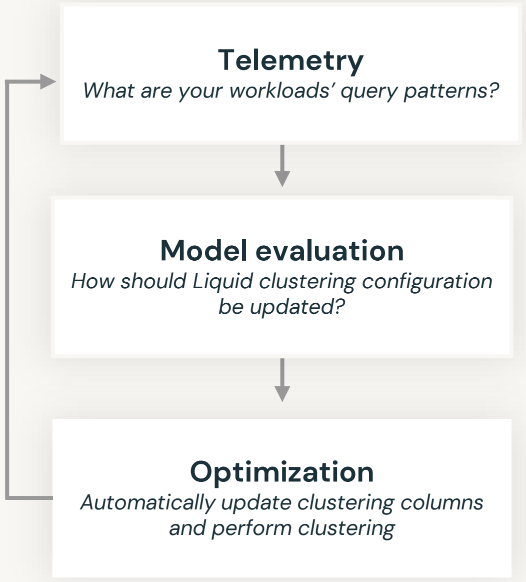
```
> CREATE TABLE tbl CLUSTER BY AUTO
```

Fully automated:

- Clustering key selection
- Clustering on write
- Background clustering

Contact your account team for the Private Preview

Liquid clustering powered by the Data Intelligence Engine



Telemetry

What are your workloads' query patterns?

Model evaluation

How should Liquid clustering configuration be updated?

Optimization

Automatically update clustering columns and perform clustering

©2024 Databricks Inc. — All rights reserved

How to make a Delta table go **fast**

```
-- Simply create a table!
CREATE TABLE tbl1 AS <query>;

-- Create liquid table
CREATE TABLE tbl1 CLUSTER BY AUTO AS <query>;

-- easily update clustering keys if query patterns change
```

Observability

What benefits is the Data Intelligence Engine providing me?

- Predictive Optimization system table

PO system table

What tables is PO performing the most compactions?

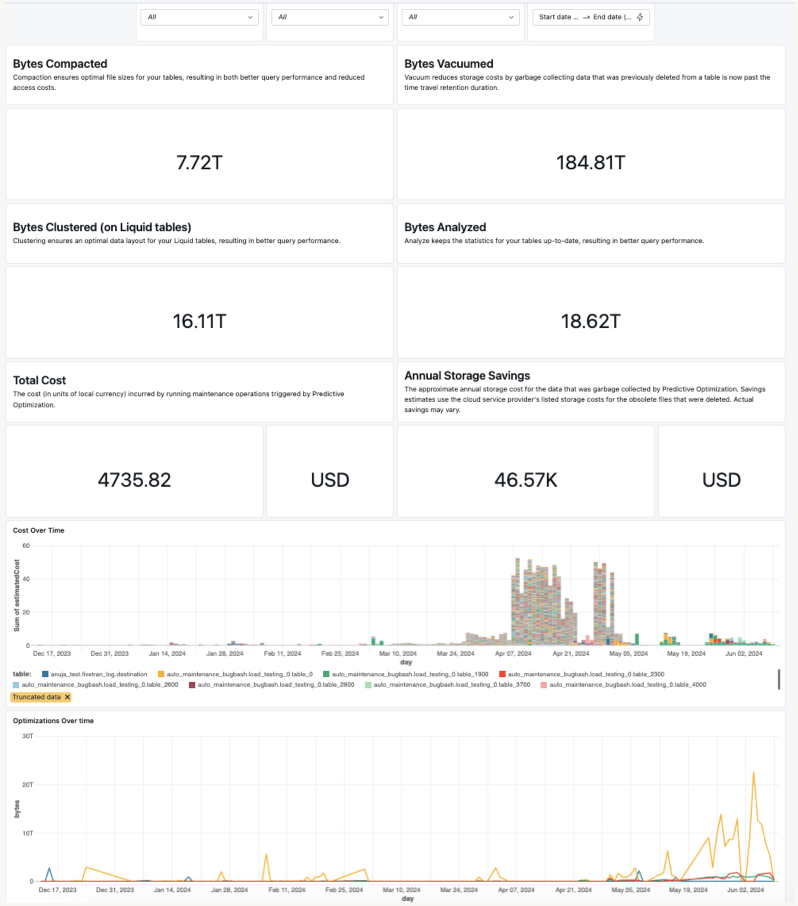
```
SELECT
     schema_name,
     table_name,
SUM(operation_metrics["amount_of_data_compacted_byt
es"]) as bytesCompacted
FROM
system.storage.predictive_optimization_operations_h
istory
WHERE
    metastore_name = {{metastore_name}}
    AND catalog_name = {{catalog_name}}
    AND operation_type = "COMPACTION"
GROUP BY ALL
 ORDER BY bytesCompacted DESC
```

Observability

What benefits is the Data Intelligence Engine providing me?

- Predictive Optimization system table
- **Out-of-the-box Predictive Optimization dashboard**

Coming soon


All
All
All
Start date ... → End date (...

Bytes Compacted
Compaction ensures optimal file sizes for your tables, resulting in both better query performance and reduced access costs.

7.72T

Bytes Vacuumed
Vacuum reduces storage costs by garbage collecting data that was previously deleted from a table is now past the time travel retention duration.

184.81T

Bytes Clustered (on Liquid tables)
Clustering ensures an optimal data layout for your Liquid tables, resulting in better query performance.

16.11T

Bytes Analyzed
Analyze keeps the statistics for your tables up-to-date, resulting in better query performance.

18.62T

Total Cost
The cost (in units of local currency) incurred by running maintenance operations triggered by Predictive Optimization.

4735.82 USD

Annual Storage Savings
The approximate annual storage cost for the data that was garbage collected by Predictive Optimization. Savings estimates use the cloud service provider's listed storage costs for the obsolete files that were deleted. Actual savings may vary.

46.57K USD

Cost Over Time

table: 🟦 anuja_test.fivetran_log.destination 🟧 auto_maintenance_bugbash.load_testing_0.table_0 🟩 auto_maintenance_bugbash.load_testing_0.table_1900 🟥 auto_maintenance_bugbash.load_testing_0.table_2300 🟦 auto_maintenance_bugbash.load_testing_0.table_2600 🟥 auto_maintenance_bugbash.load_testing_0.table_2800 🟩 auto_maintenance_bugbash.load_testing_0.table_3700 🟥 auto_maintenance_bugbash.load_testing_0.table_4000

Truncated data ✕

Optimizations Over time

Observability

What benefits is the Data Intelligence Engine providing me?

- Predictive Optimization system table
- Out-of-the-box Predictive Optimization dashboard
- **Out-of-the-box Delta table system table**

  Coming soon

Delta table system table

For UC managed tables

What are my largest Delta tables as of today?

```
SELECT
        tableName,
        tableSize
FROM
    system.storage.managed_tables
WHERE
        date = current_date()
```

```sql
-- Create Liquid table...
CREATE TABLE tbl1 CLUSTER BY AUTO;

-- easily update clustering keys if query
patterns change
```

With the Databricks Data Intelligence Engine…

Easy

Hands off; highly observable

With the Databricks Data Intelligence Engine…

Easy
Hands off; highly observable

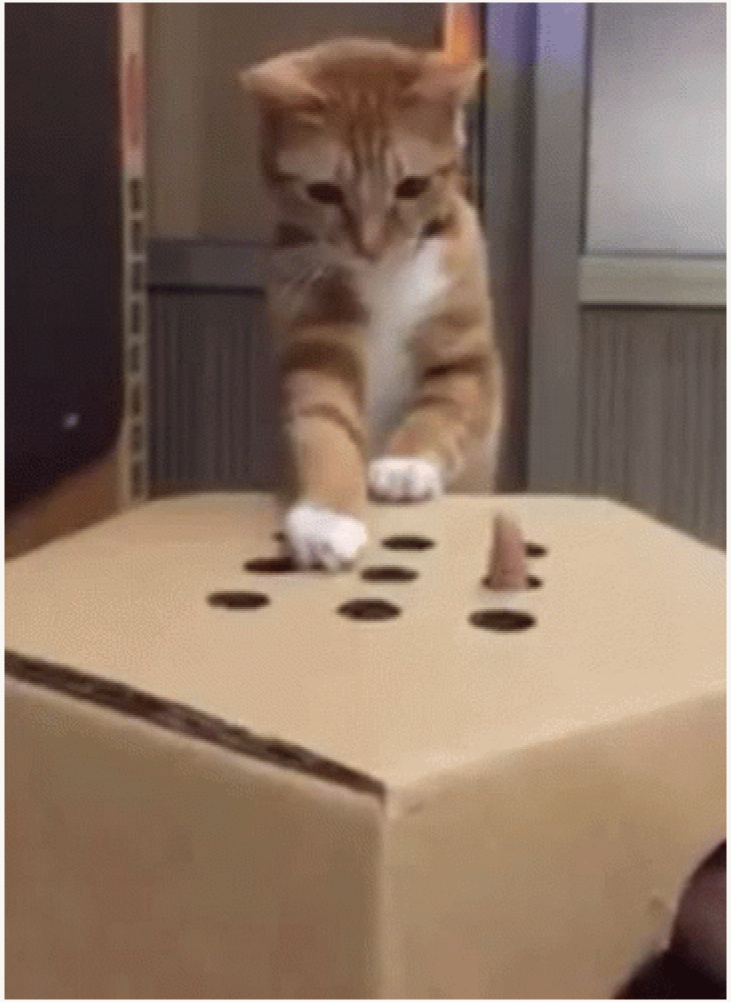
Fast
Optimal Liquid clustering, file sizing based on usage patterns

With the Databricks Data Intelligence Engine…

Easy
Hands off; highly observable

Fast
Optimal Liquid clustering, file sizing based on usage patterns

Efficient
Automatic garbage collection; optimizations only performed



Learn more at the summit!

Databricks Events App

Tells us what you think

- We kindly request your valuable feedback on this session.
- Please take a moment to rate and share your thoughts about it.
- You can conveniently provide your feedback and rating through the **Mobile App**.

What to do next?

- Discover more related sessions in the mobile app!
- Visit the Demo Booth: Experience innovation firsthand!
- More Activities: Engage and connect further at the Databricks Zone!

Get trained and certified

- Visit the Learning Hub Experience at Moscone West, 2nd Floor!
- Take complimentary certification at the event; come by the Certified Lounge
- Visit our Databricks Learning website for more training, courses and workshops! databricks.com/learn

Thank you.