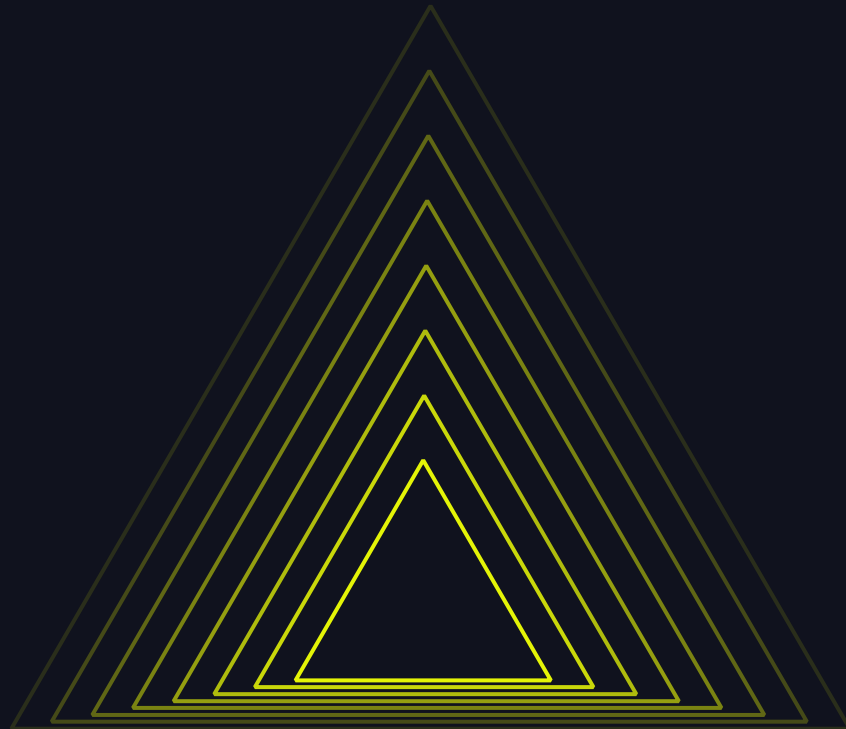


OPTIMIZING MERGE PERFORMANCE

using Liquid Clustering

Bart Samwel
June 11, 2024



ABOUT ME

Look who's talking!

Bart Samwel

Principal Software Engineer @
Databricks Amsterdam



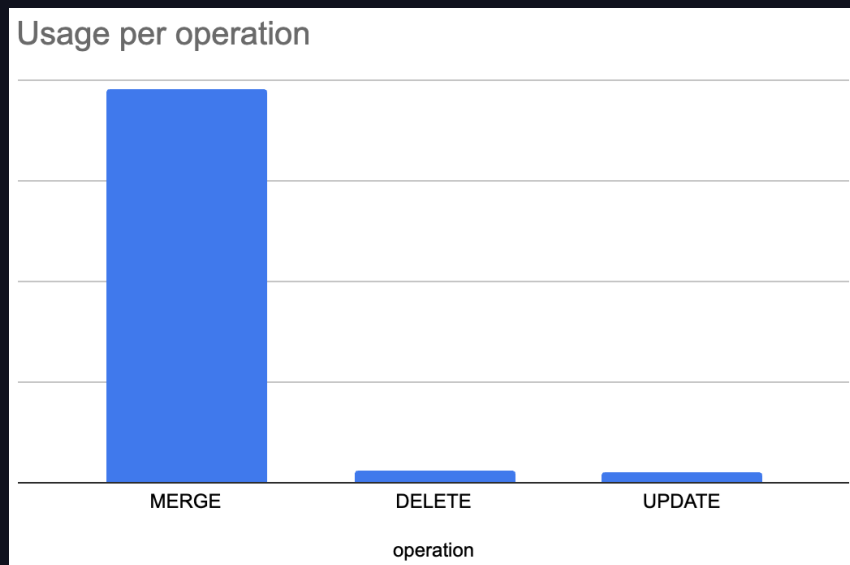
WHAT IS A MERGE?

What is a MERGE?

And why do I care?

MERGE is the workhorse of ETL.

- Incremental ETL (e.g. UPSERTs)
- Change Data Capture (CDC) replay
 - e.g. `APPLY CHANGES INTO`
- Delta Live Tables
- Materialized Views



MERGE Example

```
MERGE INTO MyTargetTable AS t
USING MySourceTable AS s
  ON s.ColX = t.ColX AND s.ColY = t.ColY
WHEN MATCHED AND s.action = 'update'
  THEN UPDATE SET *
WHEN MATCHED AND s.action = 'delete'
  THEN DELETE
WHEN NOT MATCHED
  THEN INSERT *
```

MERGE Example

```
MERGE INTO MyTargetTable AS t
USING MySourceTable AS s
  ON s.ColX = t.ColX AND s.ColY = t.ColY
  WHEN MATCHED AND s.Action = 'update'
    THEN UPDATE SET *
  WHEN MATCHED AND s.Action = 'delete'
    THEN DELETE
  WHEN NOT MATCHED
    THEN INSERT *
```



MyTargetTable

ColX	ColY	ValZ
1	320	"Foo"
4	737	"Bar"
6	380	"Baz"



MySourceTable

ColX	ColY	ValZ	Action
4	737	"Pebbles"	"update"
6	380	"Baz"	"delete"
7	787	"Bambam"	

MERGE Example

```
MERGE INTO MyTargetTable AS t
  USING MySourceTable AS s
    ON s.ColX = t.ColX AND s.ColY = t.ColY
  WHEN MATCHED AND s.Action = 'update'
    THEN UPDATE SET *
  WHEN MATCHED AND s.Action = 'delete'
    THEN DELETE
  WHEN NOT MATCHED
    THEN INSERT *
```



MyTargetTable

ColX	ColY	ValZ
1	320	"Foo"
4	737	"Bar"
6	380	"Baz"



MySourceTable

ColX	ColY	ValZ	Action
4	737	"Pebbles"	"update"
6	380	"Baz"	"delete"
7	787	"Bambam"	



MERGE Example

```
MERGE INTO MyTargetTable AS t
  USING MySourceTable AS s
    ON s.ColX = t.ColX AND s.ColY = t.ColY
  WHEN MATCHED AND s.Action = 'update'
    THEN UPDATE SET *
  WHEN MATCHED AND s.Action = 'delete'
    THEN DELETE
  WHEN NOT MATCHED
    THEN INSERT *
```



MyTargetTable

ColX	ColY	ValZ
1	320	"Foo"
4	737	"Bar"
6	380	"Baz"



MySourceTable

ColX	ColY	ValZ	Action
4	737	"Pebbles"	"update"
6	380	"Baz"	"delete"
7	787	"Bambam"	



MERGE Example

```
MERGE INTO MyTargetTable AS t
  USING MySourceTable AS s
    ON s.ColX = t.ColX AND s.ColY = t.ColY
  WHEN MATCHED AND s.Action = 'update'
    THEN UPDATE SET *
  WHEN MATCHED AND s.Action = 'delete'
    THEN DELETE
  WHEN NOT MATCHED
    THEN INSERT *
```



MyTargetTable

ColX	ColY	ValZ
1	320	"Foo"
4	737	"Pebbles"
6	380	"Baz"



MySourceTable

ColX	ColY	ValZ	Action
4	737	"Pebbles"	"update"
6	380	"Baz"	"delete"
7	787	"Bambam"	



MERGE Example

```
MERGE INTO MyTargetTable AS t
USING MySourceTable AS s
  ON s.ColX = t.ColX AND s.ColY = t.ColY
WHEN MATCHED AND s.Action = 'update'
  THEN UPDATE SET *
WHEN MATCHED AND s.Action = 'delete'
  THEN DELETE
WHEN NOT MATCHED
  THEN INSERT *
```



MyTargetTable

ColX	ColY	ValZ
1	320	"Foo"
4	737	"Pebbles"
6	380	"Baz"



MySourceTable

ColX	ColY	ValZ	Action
4	737	"Pebbles"	"update"
6	380	"Baz"	"delete"
7	787	"Bambam"	



MERGE Example

```
MERGE INTO MyTargetTable AS t
USING MySourceTable AS s
  ON s.ColX = t.ColX AND s.ColY = t.ColY
WHEN MATCHED AND s.Action = 'update'
  THEN UPDATE SET *
WHEN MATCHED AND s.Action = 'delete'
  THEN DELETE
WHEN NOT MATCHED
  THEN INSERT *
```



MyTargetTable

ColX	ColY	ValZ
1	320	"Foo"
4	737	"Pebbles"
6	380	"Baz"



MySourceTable

ColX	ColY	ValZ	Action
4	737	"Pebbles"	"update"
6	380	"Baz"	"delete"
7	787	"Bambam"	



MERGE Example

```
MERGE INTO MyTargetTable AS t
USING MySourceTable AS s
  ON s.ColX = t.ColX AND s.ColY = t.ColY
WHEN MATCHED AND s.Action = 'update'
  THEN UPDATE SET *
WHEN MATCHED AND s.Action = 'delete'
  THEN DELETE
WHEN NOT MATCHED
  THEN INSERT *
```



MyTargetTable

ColX	ColY	ValZ
1	320	"Foo"
4	737	"Pebbles"
6	380	"Baz"



MySourceTable

ColX	ColY	ValZ	Action
4	737	"Pebbles"	"update"
6	380	"Baz"	"delete"
7	787	"Bambam"	



MERGE Example

```
MERGE INTO MyTargetTable AS t
USING MySourceTable AS s
  ON s.ColX = t.ColX AND s.ColY = t.ColY
WHEN MATCHED AND s.Action = 'update'
  THEN UPDATE SET *
WHEN MATCHED AND s.Action = 'delete'
  THEN DELETE
WHEN NOT MATCHED
  THEN INSERT *
```



MyTargetTable

ColX	ColY	ValZ
1	320	"Foo"
4	737	"Pebbles"
6	380	"Baz"
7	787	"Bambam"



MySourceTable

ColX	ColY	ValZ	Action
4	737	"Pebbles"	"update"
6	380	"Baz"	"delete"
7	787	"Bambam"	



MAKING MERGE FAST

How do I optimize MERGE?

And make that SIMPLE?

STEP 1: CLUSTER BY your merge keys using Liquid Clustering

STEP 2: Use meaningful merge keys. Don't use only random GUIDs.

... DONE!

DATA+AI SUMMIT



DEEP DIVE



DEEP DIVE AGENDA

- HOW MERGE WORKS
- KEY OPTIMIZATIONS:
 - #1: LIQUID CLUSTERING
 - #2: DELETION VECTORS
 - #3: DYNAMIC PRUNING
 - #4: BLOOM FILTER JOINS
- MEANINGFUL KEYS

HOW MERGE WORKS

How MERGE Works

STEP 1: Find Files with Matches



MyTargetTable



File1

Key	...
1	
7	
18	



File2

Key	...
11	
12	
6	



File3

Key	...
15	
2	
9	



File4

Key	...
4	
13	
8	



MySourceTable

Key	...
2	D
4	U
15	U
18	U
283	



How MERGE Works

Legend: **Matches**

STEP 1: Find Files with Matches



MyTargetTable



File1

Key	...
1	
7	
18	



File2

Key	...
11	
12	
6	



File3

Key	...
2	
15	
9	



File4

Key	...
4	
13	
8	



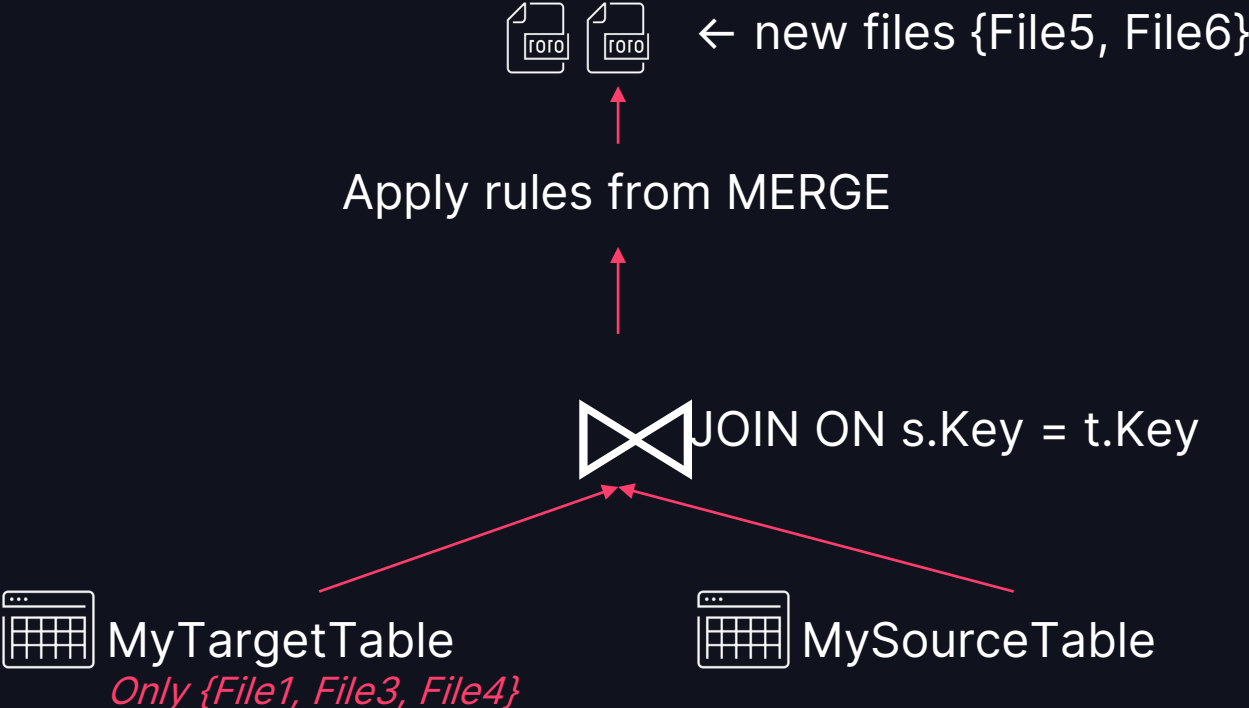
MySourceTable

Key	...
2	D
4	U
15	U
18	U
283	



How MERGE Works

STEP 2: Apply Changes

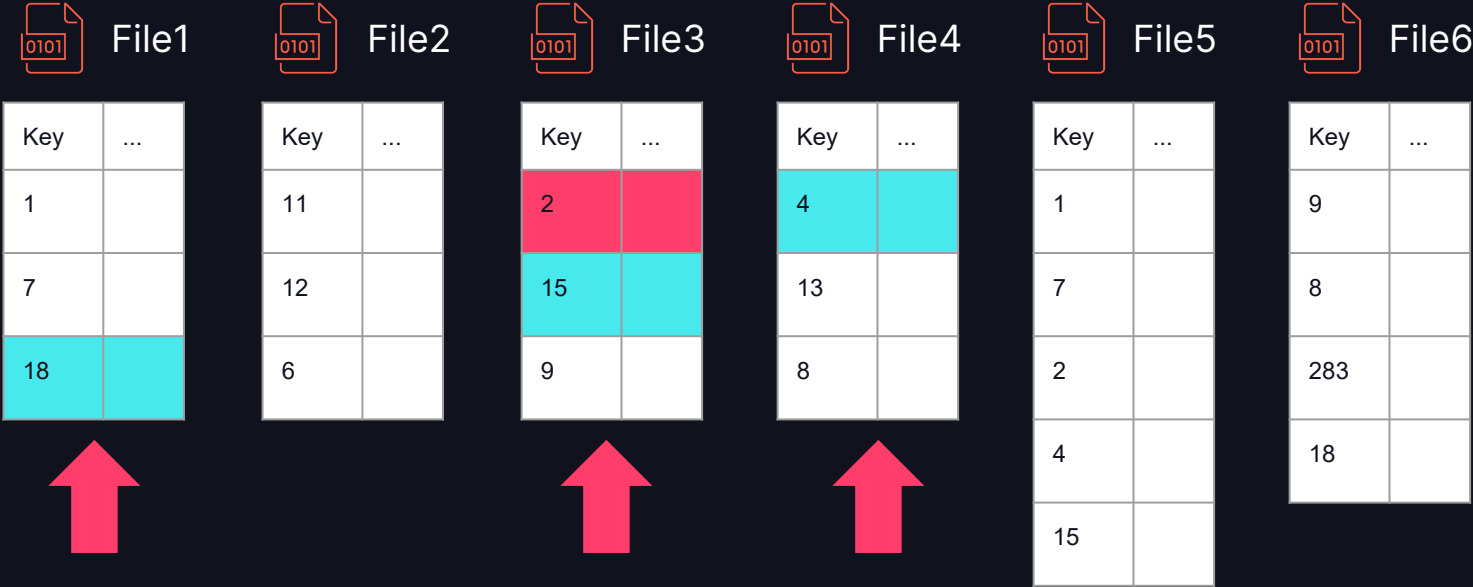


How MERGE Works

Legend: Updates Deletes

STEP 2: Apply Changes

 MyTargetTable



How MERGE Works

Legend: Updates Deletes Inserts

STEP 2: Apply Changes

 MyTargetTable

 File1

Key	...
1	
7	
18	



 File2

Key	...
11	
12	
6	

 File3

Key	...
2	
15	
9	



 File4

Key	...
4	
13	
8	



 File5

Key	...
1	
7	
13	
4	
15	

 File6

Key	...
9	
8	
283	
18	

How MERGE Works

Legend: Updates Deletes Inserts

STEP 3: Commit Changes

 MyTargetTable

 File1

Key	...
1	
1	
18	

 File2

Key	...
11	
12	
6	

 File3

Key	...
2	
1	
9	

 File4

Key	...
4	
1	
8	

 File5

Key	...
1	
7	
13	
4	
15	

 File6

Key	...
9	
8	
283	
18	

How MERGE Works

Legend: Updates Deletes Inserts

STEP 3: Commit Changes

 MyTargetTable (after MERGE)

 File2

Key	...
11	
12	
6	

 File5

Key	...
1	
7	
13	
4	
15	

 File6

Key	...
9	
8	
283	
18	

How MERGE Works

Recap

STEP 1: Find Files with Matches

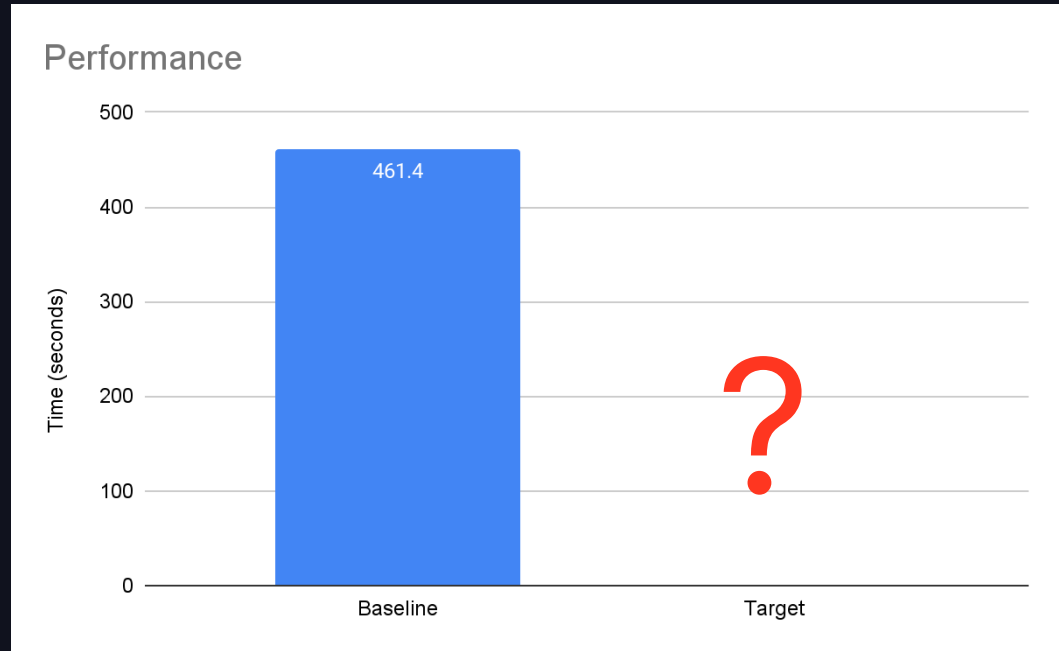
STEP 2: Apply changes to files identified in step 1

STEP 3: Commit Changes (add new files and remove old files)

OPTIMIZATIONS

TRACKING PERFORMANCE

- Orders table
- 100GB, 1 year of data
- MERGE 1% of records of the last week
- Overall change 0.02%.



OPTIMIZATION #1

LIQUID CLUSTERING

LIQUID CLUSTERING

Legend: **Matches**



MyTargetTable



File1

Key	...
1	
7	
18	



File2

Key	...
11	
12	
9	



File3

Key	...
2	
6	
15	



File4

Key	...
4	
13	
8	



MySourceTable

Key	...
2	D
4	U
15	U
18	U
283	



LIQUID CLUSTERING

Legend: **Matches**



MyTargetTable



File1

Key	...
1	
2	
4	



File2

Key	...
6	
7	
8	



File3

Key	...
12	
11	
9	



File4

Key	...
18	
13	
15	



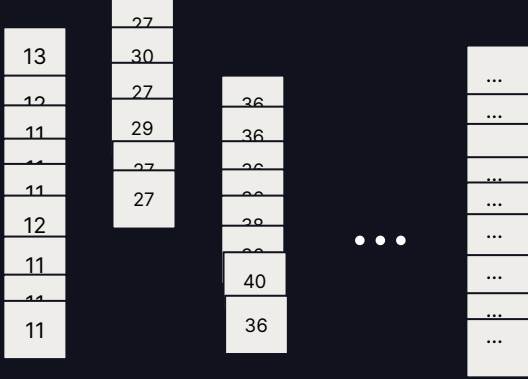
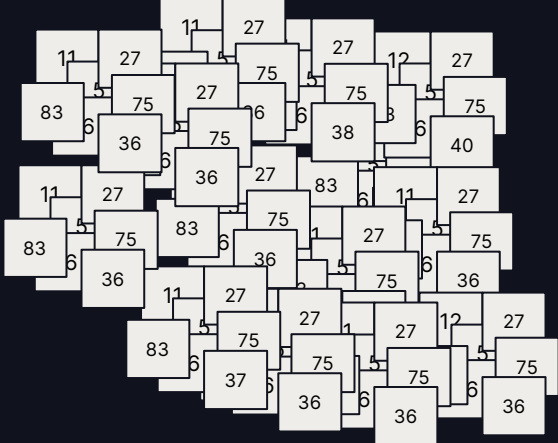
MySourceTable

Key	...
2	D
4	U
15	U
18	U
283	



LIQUID CLUSTERING

WE ORGANIZE THE DATA FOR YOU!



LIQUID CLUSTERING

- To enable Liquid Clustering:

```
CREATE / ALTER TABLE ... CLUSTER BY (Key)
```

- Specify all MERGE keys in **CLUSTER BY**. (Up to 4 keys.)
 - Maybe also add the most common filter key in SELECT queries!
- **OPTIMIZE myTable** automatically organizes your data.

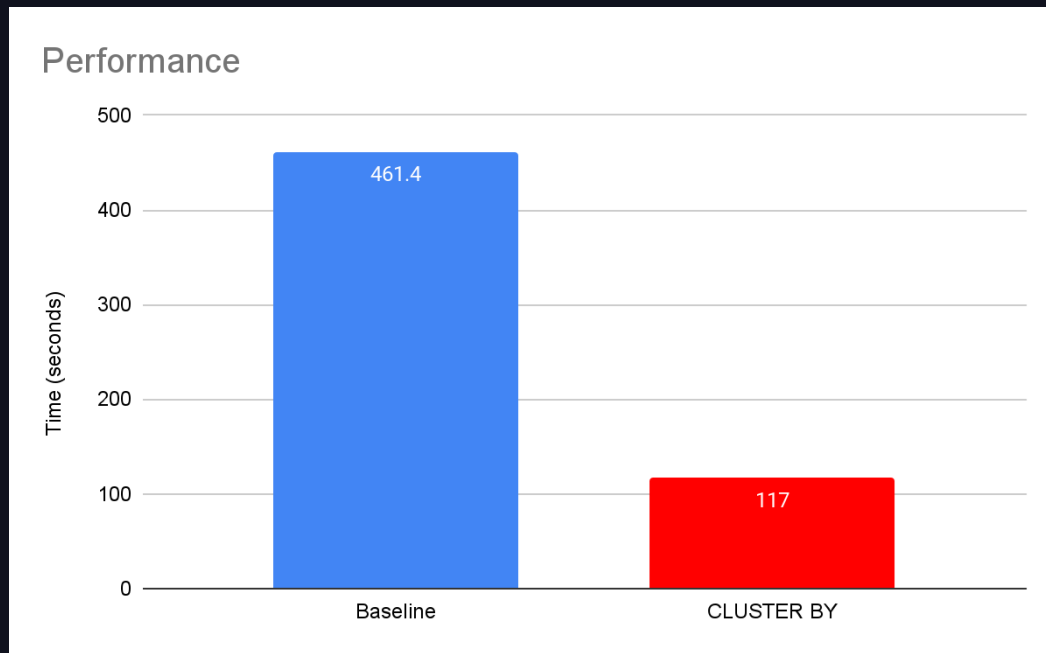
LIQUID CLUSTERING

Databricks Data Intelligence Platform

- Predictive Optimization => no need to run **OPTIMIZE**
 - Automatic table optimization service
 - **OPTIMIZE**, **VACUUM**, and many other optimizations
 - Uses AI to get maximum cost/benefit
- Large ingestions are immediately organized correctly.
- Soon: use **CLUSTER BY AUTO**
 - Databricks picks the clustering keys for you.

TRACKING PERFORMANCE: CLUSTER BY

- Orders table
- 100GB, 1 year of data
- MERGE 1% of records of the last week
- Overall change 0.02%.



DON'T WE KNOW THIS?

- ZORDER?
- Hive-style partitioned tables?
- Liquid Clustering is better!

LIQUID CLUSTERING

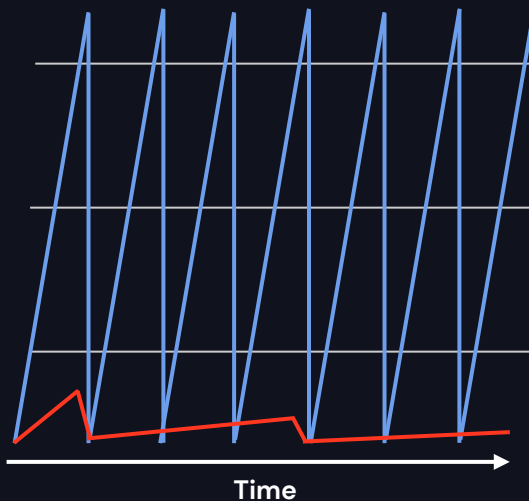
It's better than ZORDER!

- More incremental!
- Allows concurrency
- Automatic Optimization
 - Predictive Optimization.
 - Immediate clustering on write
 - CLUSTER BY AUTO

Read Performance

Bytes scanned for Point Queries
Lower is better

ZORDER Liquid



LIQUID CLUSTERING

It's better than partitioned tables!


Partitioned tables are hard to configure correctly.

Tables with many small files are the #1 performance bottleneck.

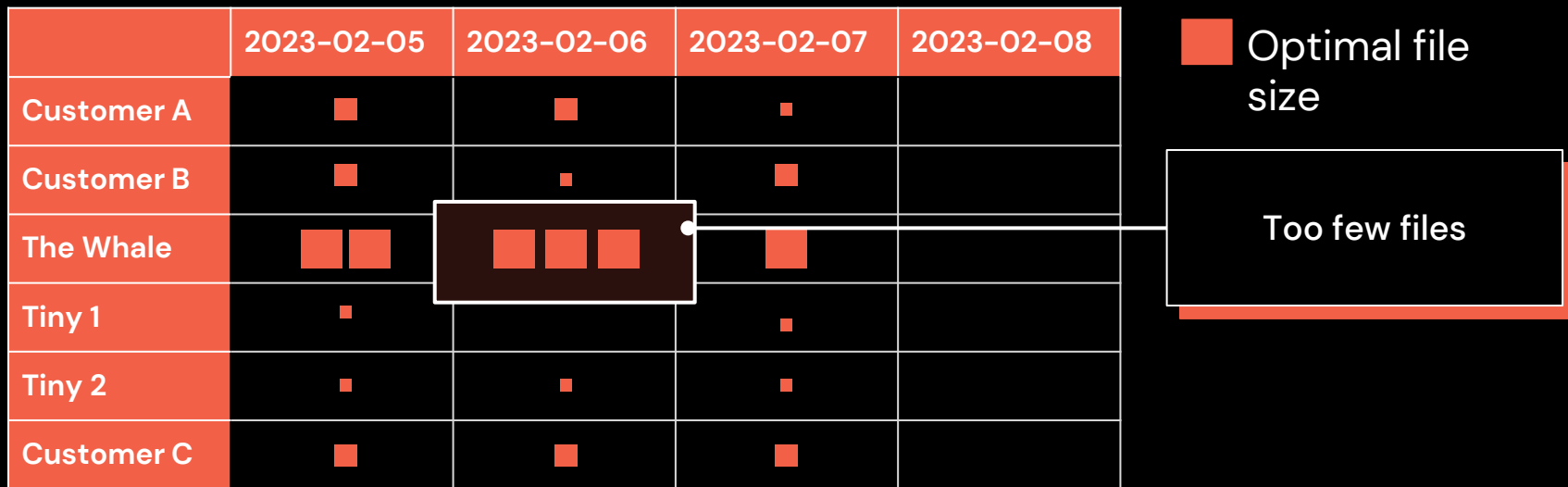
- Over-partitioning: too many small files
- Under-partitioning: no skipping benefits
- Skewed partitioning

Even with a good partitioning strategy...

	2023-02-05	2023-02-06	2023-02-07	2023-02-08
Customer A				
Customer B				
The Whale				
Tiny 1				
Tiny 2				
Customer C				

 Optimal file size

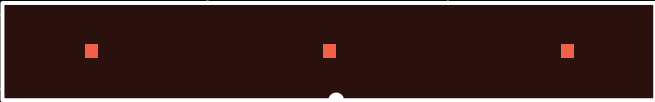
Even with a good partitioning strategy...



Even with a good partitioning strategy...

	2023-02-05	2023-02-06	2023-02-07	2023-02-08
Customer A	■	■	■	
Customer B	■	■	■	
The Whale	■ ■	■ ■ ■	■	
Tiny 1	■		■	
Tiny 2	■	■	■	
Customer C	■	■	■	

■ Optimal file size




Too many files

Liquid Clustering

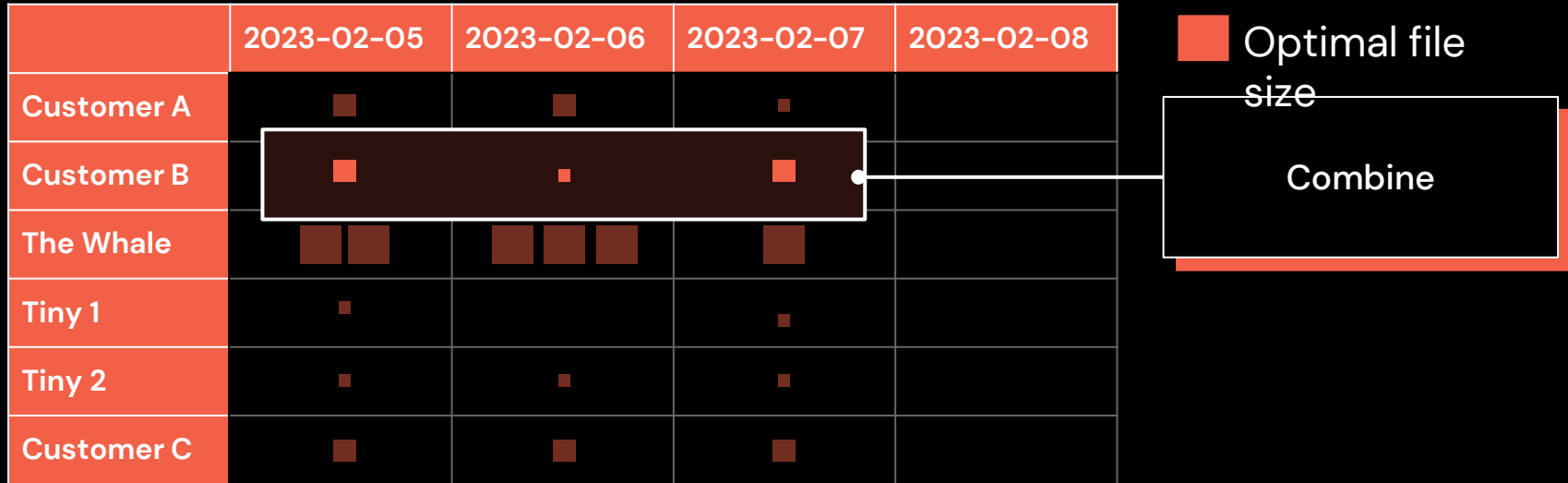
Efficiently balance clustering vs. file size

	2023-02-05	2023-02-06	2023-02-07	2023-02-08
Customer A				
Customer B				
The Whale				
Tiny 1				
Tiny 2				
Customer C				

 Optimal file size

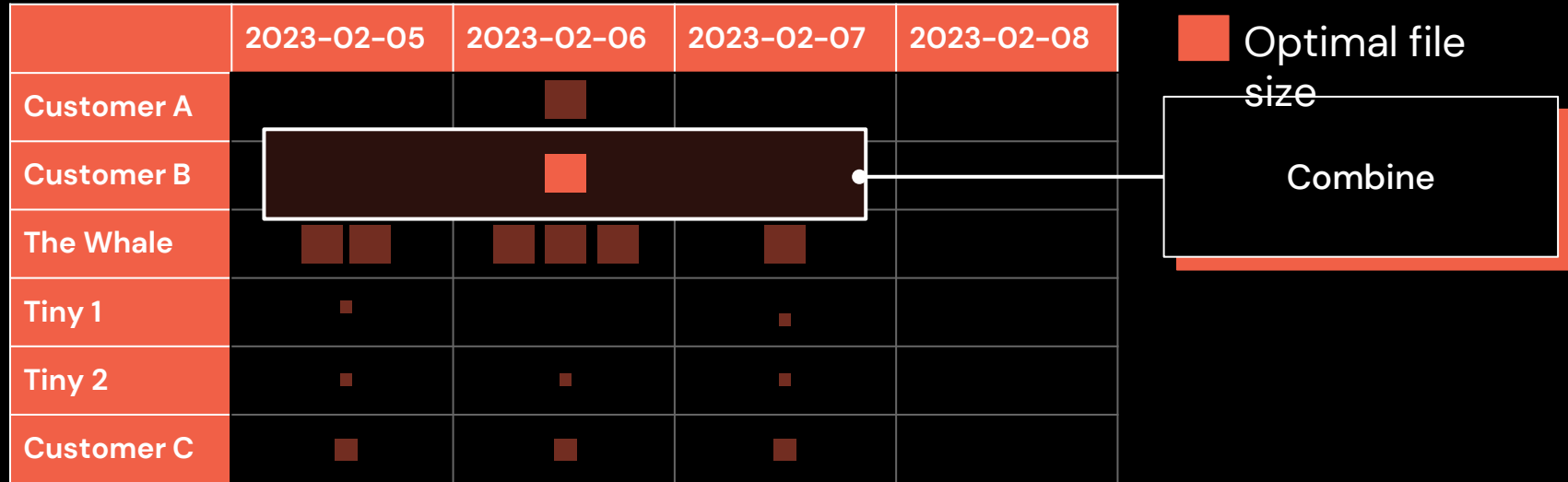
Liquid Clustering

Efficiently balance clustering vs. file size



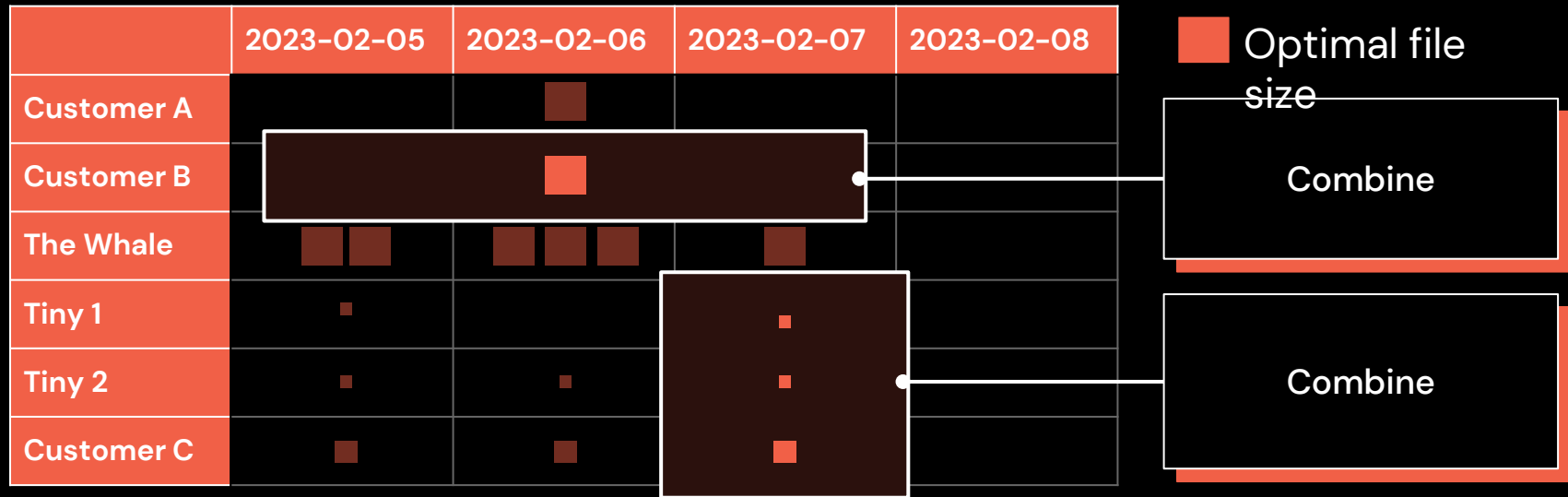
Liquid Clustering

Efficiently balance clustering vs. file size



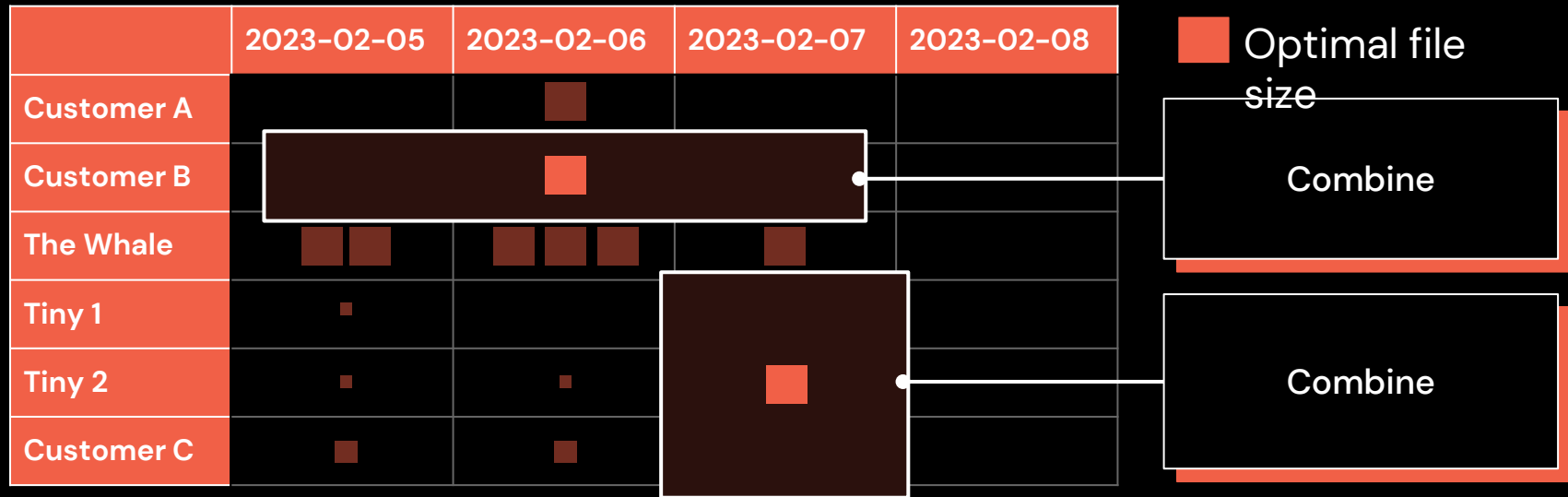
Liquid Clustering

Efficiently balance clustering vs. file size



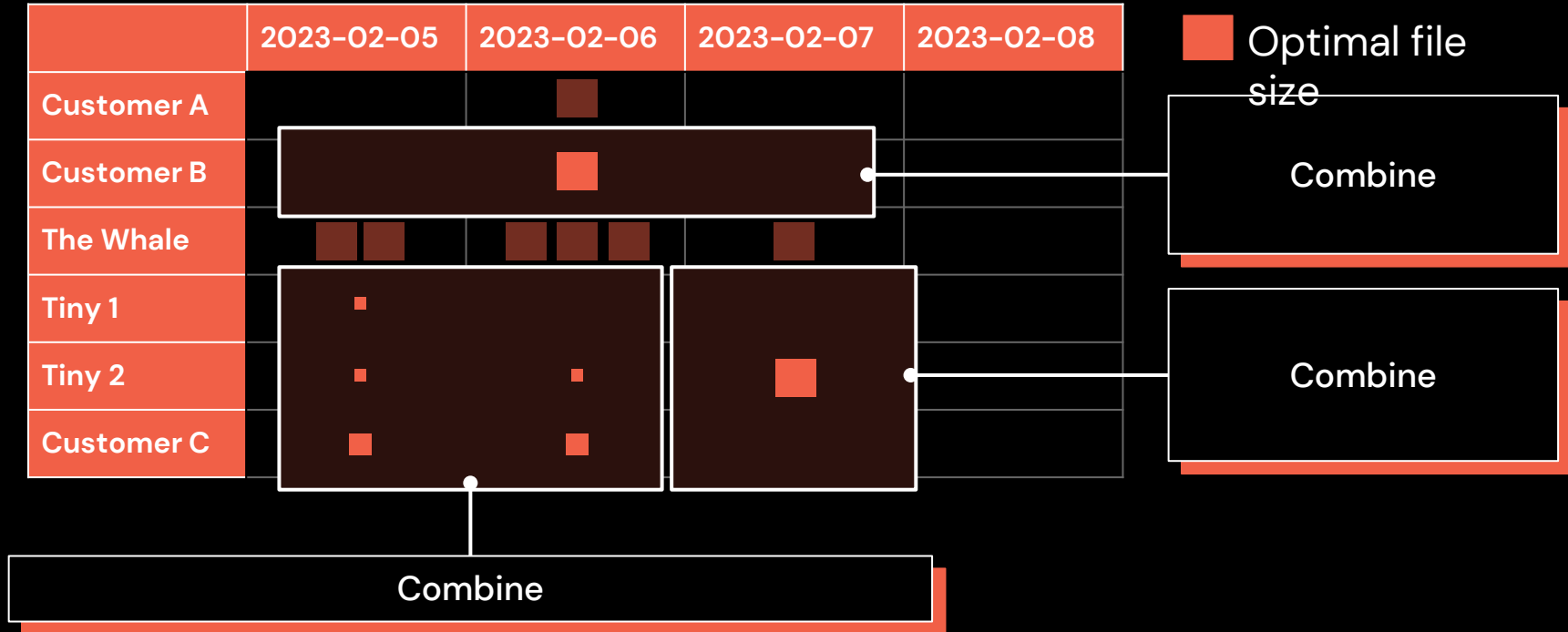
Liquid Clustering

Efficiently balance clustering vs. file size



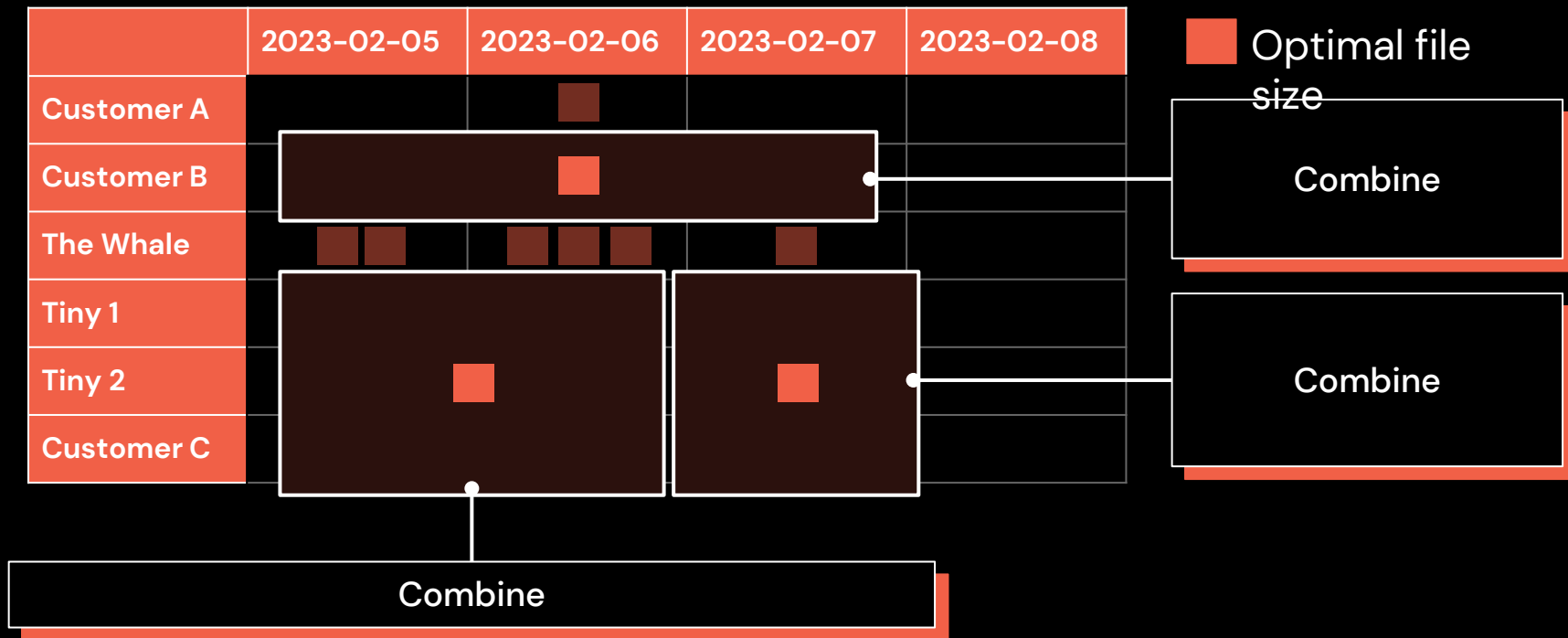
Liquid Clustering

Efficiently balance clustering vs. file size



Liquid Clustering

Efficiently balance clustering vs. file size



Liquid Clustering

Automatically cluster heavy partitions more finely

	2023-02-05	2023-02-06	2023-02-07	2023-02-08
Customer A		■		
Customer B		■		
The Whale	■ ■	■ ■ ■	■	
Tiny 1				
Tiny 2		■	■	
Customer C				

■ Optimal file size

Subdivide

Liquid Clustering

Automatically cluster heavy partitions more finely

	2023-02-05	2023-02-06	2023-02-07	2023-02-08
Customer A		■		
Customer B		■		
The Whale	■ ■	■ ■ ■ ■	■	
Tiny 1				
Tiny 2		■	■	
Customer C				

■ Optimal file size

Subdivide

LIQUID CLUSTERING

It's better than partitioned tables!

	Partitioned Tables	Liquid Clustering
Bad file sizes?	✗ Yes	✓ No
Key changes?	✗ CREATE TABLE only	✓ CREATE + ALTER TABLE
Key types?	✗ Low cardinality columns only	✓ Any column (incl TIMESTAMP, IDs)

LIQUID CLUSTERING

It's better than partitioned tables!

	Partitioned Tables	Liquid Clustering
Bad file sizes?	✗ Yes	✓ No
Key changes?	✗ CREATE TABLE only	✓ CREATE + ALTER TABLE
Key types?	✗ Low cardinality columns only	✓ Any column (incl TIMESTAMP, IDs)
Concurrency	✓ Supported ✗ Have to add filters	✓ Supported ✓ Works out of the box

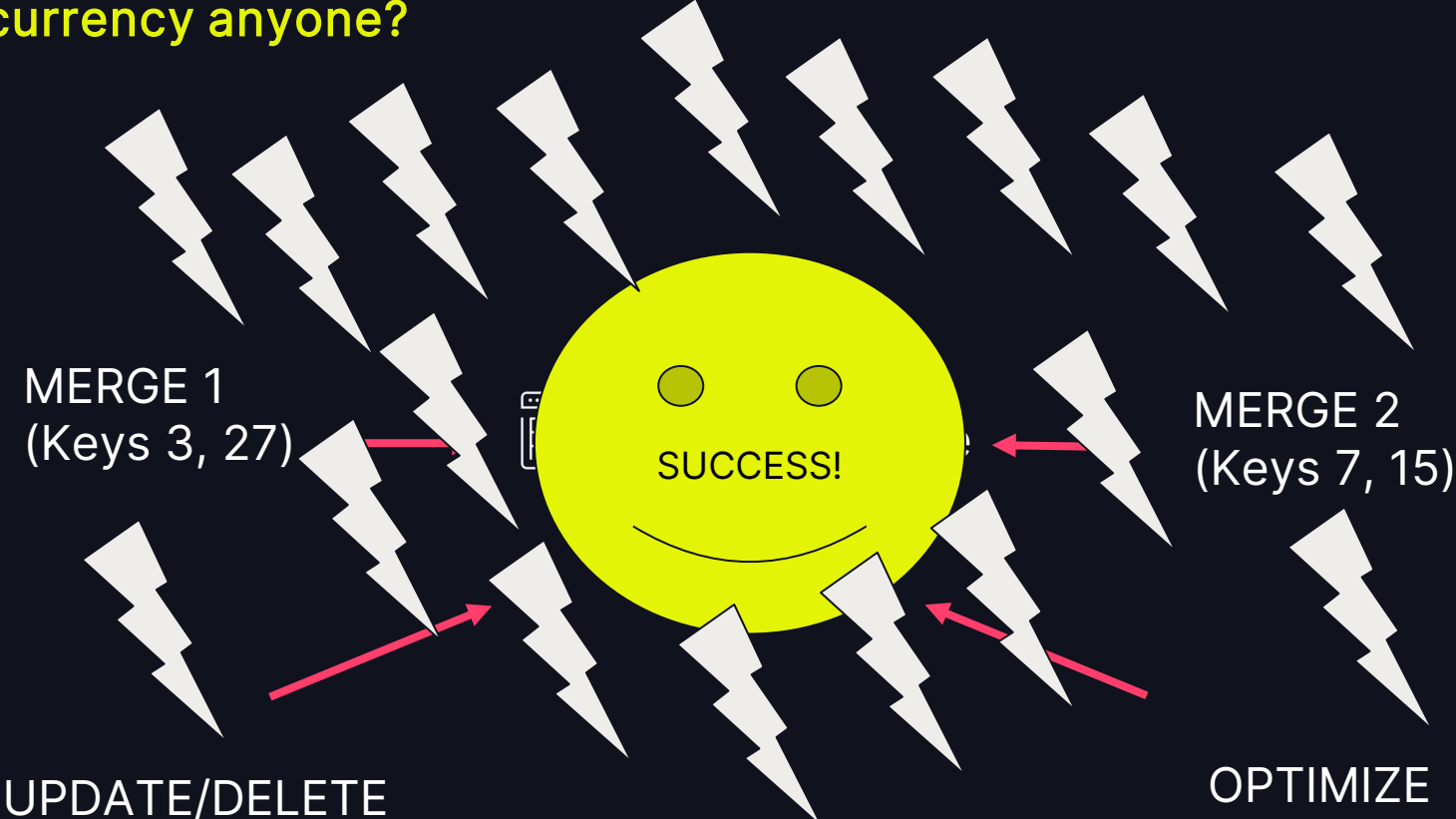
LIQUID CLUSTERING

Concurrency anyone?



LIQUID CLUSTERING

Concurrency anyone?



OPTIMIZATION #2

DELETION VECTORS

LOTS OF COPIES

Legend: Updates Deletes Inserts



File1 File2 File3 File4 File5 File6

Key	...
1	
7	
18	

Key	...
11	
12	
6	

Key	...
2	
15	
9	

Key	...
4	
13	
8	

Key	...
1	
7	
13	
4	
15	

Key	...
283	
18	



DELETION VECTORS

Legend: Deletes Inserts

Mark rows as deleted!

 MyTargetTable

 File1
+ DV1

Key	...
1	
7	
18	

 File2

Key	...
11	
12	
6	

 File3
+ DV2

Key	...
2	
45	
9	

 File4
+ DV3

Key	...
4	
13	
8	

 File5

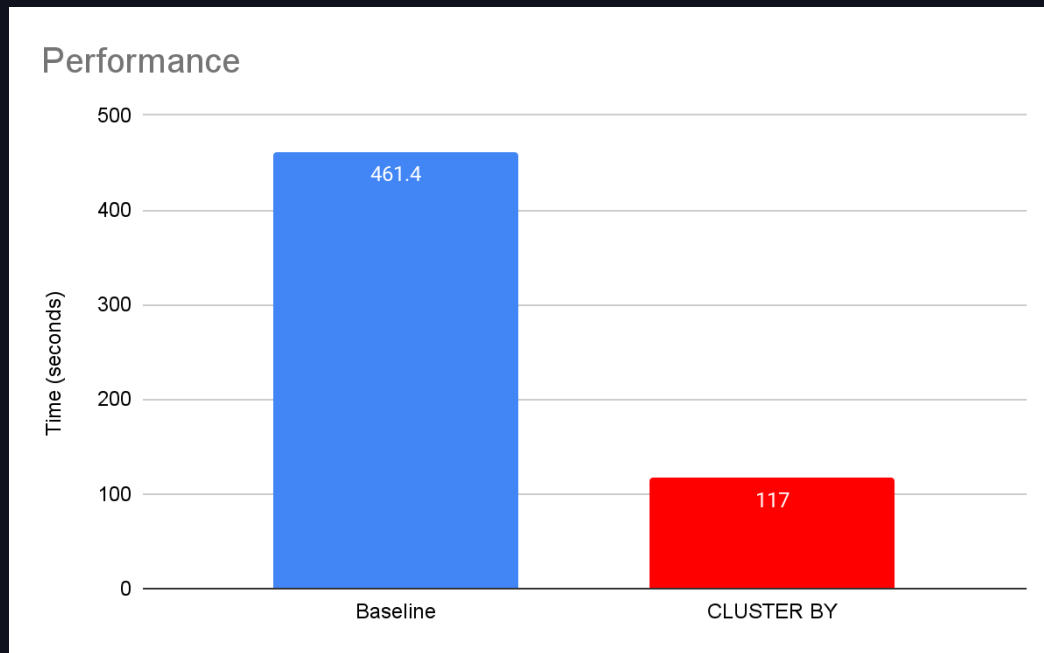
Key	...
4	
15	
283	
18	

NO COPIES!



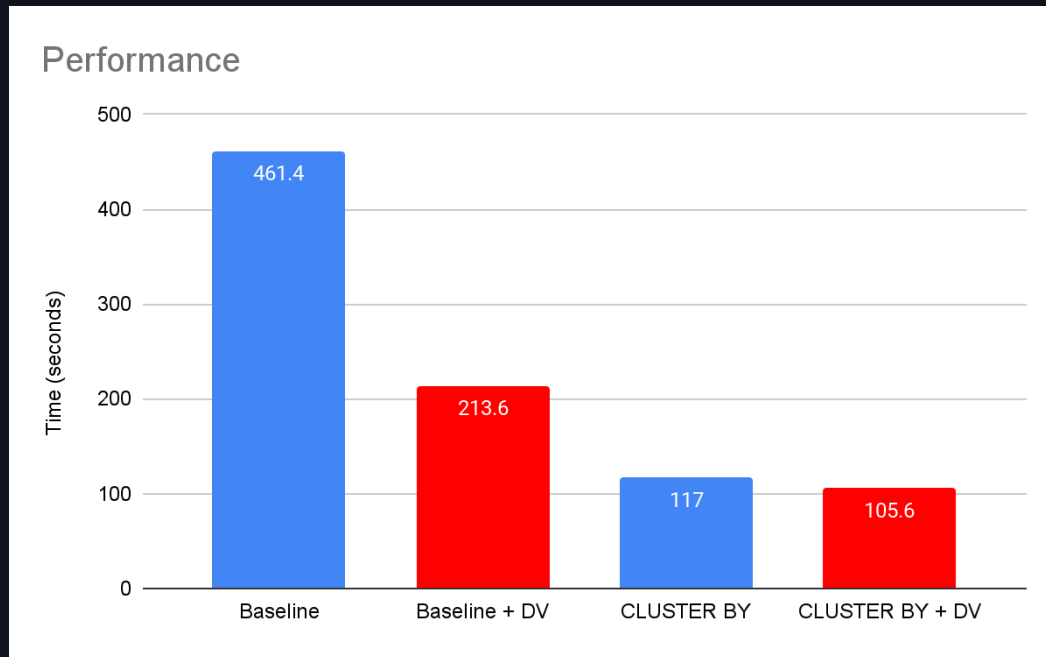
TRACKING PERFORMANCE: CLUSTER BY

- Orders table
- 100GB, 1 year of data
- MERGE 1% of records of the last week
- Overall change 0.02%.



TRACKING PERFORMANCE: DELETION VECTORS

- Orders table
- 100GB, 1 year of data
- MERGE 1% of records of the last week
- Overall change 0.02%.



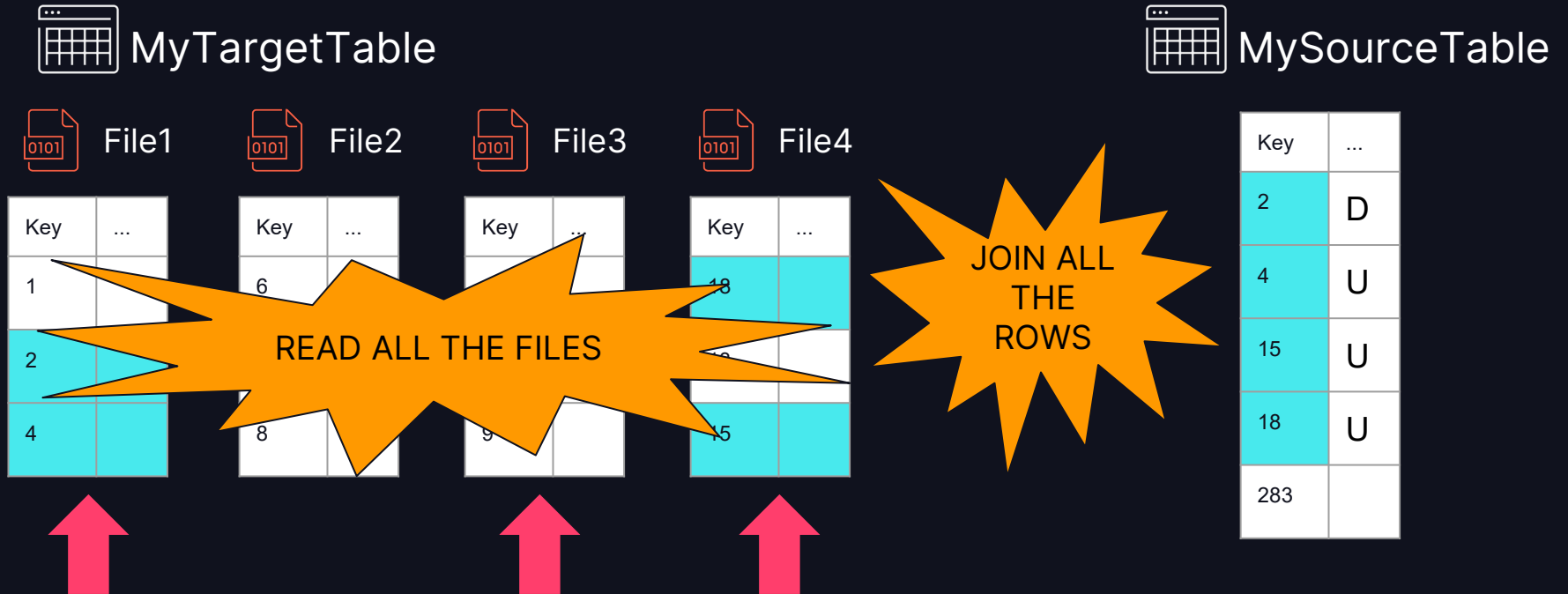
OPTIMIZATION #3

DYNAMIC FILE PRUNING

DYNAMIC FILE PRUNING

Legend: **Matches**

[Recap] STEP 1: Find Files with Matches



DYNAMIC FILE PRUNING

MIN/MAX Data Skipping



MyTargetTable



File1

Key	...
1	
2	
4	

Key:
1..4



File2

Key	...
6	
7	
8	

Key:
6..8



File3

Key	...
12	
11	
9	

Key:
9..12



File4

Key	...
18	
13	
15	

Key:
15..18



MySourceTable

Key	...
2	D
4	U
15	U
18	U
283	



DYNAMIC FILE PRUNING

Legend: **Matches**

MIN/MAX Data Skipping



MyTargetTable



File1



File2



File3



File4

Key	...
1	
2	
4	

Key ...

MIN(Key) = 1
MAX(Key) = 4

MAY CONTAIN
2, 4

Key	...
2	
1	

Key	...
18	
13	
15	



Key:
1..4

Key:
6..8

Key:
9..12

Key:
15..1



MySourceTable

Key	...
2	D
4	U
15	U
18	U
283	



DYNAMIC FILE PRUNING

Legend: **Matches**

MIN/MAX Data Skipping



MyTargetTable



File1

Key	...
1	
2	
4	

✓ Key:
1..4



File2

Key	...
6	
7	
8	

✗ Key:
6..8



File3

Key	...
18	
13	
15	

Key:
9..12



File4

Key	...
18	
13	
15	

Key:
15..1

MIN(Key) = 6
MAX(Key) = 8

MAY CONTAIN
Nothing!



MySourceTable

Key	...
2	D
4	U
15	U
18	U
283	



DYNAMIC FILE PRUNING

Legend: **Matches**

MIN/MAX Data Skipping



MyTargetTable



File1

Key	...
1	
2	
4	

✓ Key:
1..4



File2

Key	...
6	
7	
8	

✗ Key:
6..8



File3

Key	...
12	
11	
9	

✗ Key:
9..12



File4

Key	...
-----	-----

Key:
15..1

MIN(Key) = 9
MAX(Key) = 12

MAY CONTAIN
Nothing!



MySourceTable

Key	...
2	D
4	U
15	U
18	U
283	



DYNAMIC FILE PRUNING

Legend: Matches

MIN/MAX Data Skipping



MyTargetTable



File1

Key	...
1	
2	
4	

✓ Key:
1..4



File2

Key	...
6	
7	
8	

✗ Key:
6..8



File3

Key	...
12	
11	
9	

✗ Key:
9..12



File4

Key	...
18	
13	
15	

✓ Key:
15..18

MIN(Key) = 15
MAX(Key) = 18

MAY CONTAIN
15, 18



MySourceTable

Key	...
2	D
4	U
15	U
18	U
283	



DYNAMIC FILE PRUNING

Legend: Matches

MIN/MAX Data Skipping



MyTargetTable



File1



File2



File3



File4

Key	...
1	
2	
4	

Key	...
6	
8	

Key	...
9	

Key	...
18	
15	

READ FEWER FILES



Key:
1..4



Key:
6..8



Key:
9..12



Key:
15..1



MySourceTable

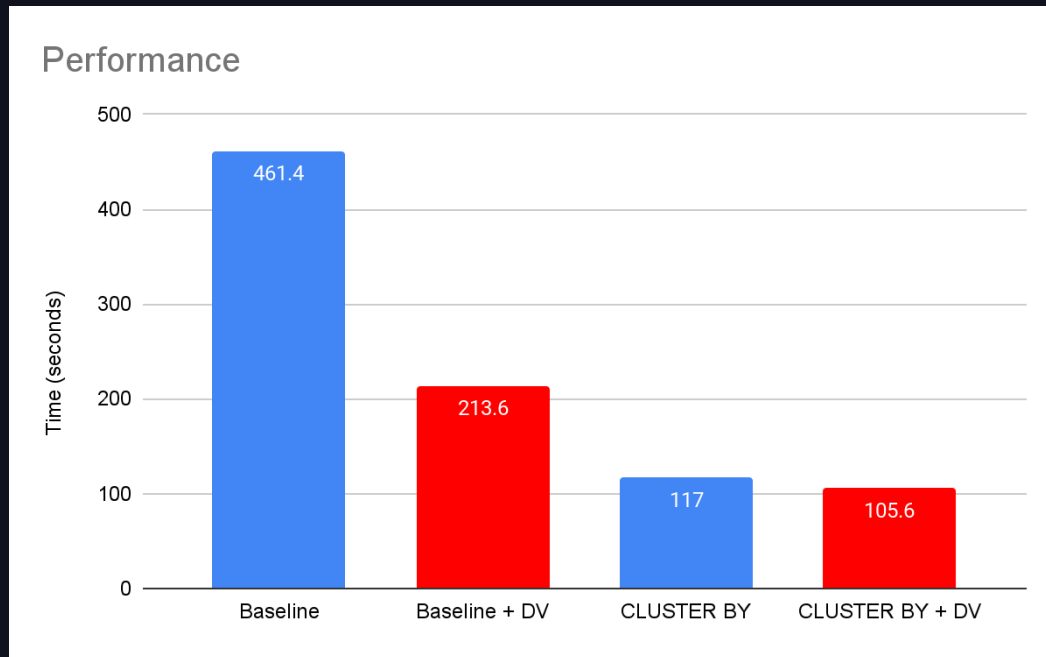
Key	...
2	D
4	U
15	U
18	U
283	

JOIN
FEWER
ROWS



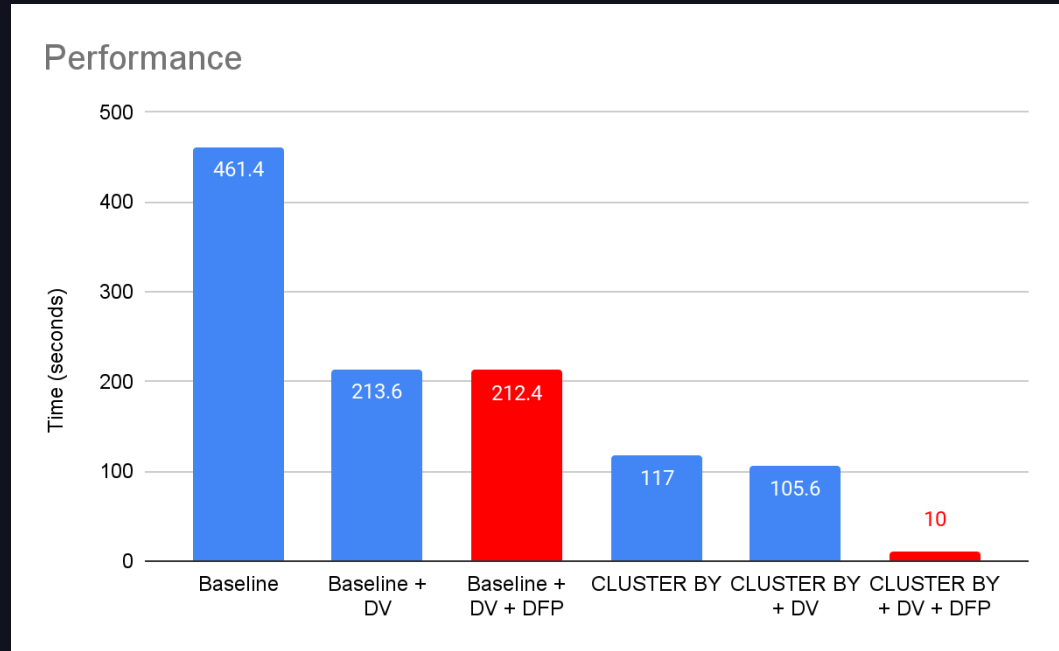
TRACKING PERFORMANCE: DELETION VECTORS

- Orders table
- 100GB, 1 year of data
- MERGE 1% of records of the last week
- Overall change 0.02%.



TRACKING PERFORMANCE: FILE PRUNING

- Orders table
- 100GB, 1 year of data
- MERGE 1% of records of the last week
- Overall change 0.02%.



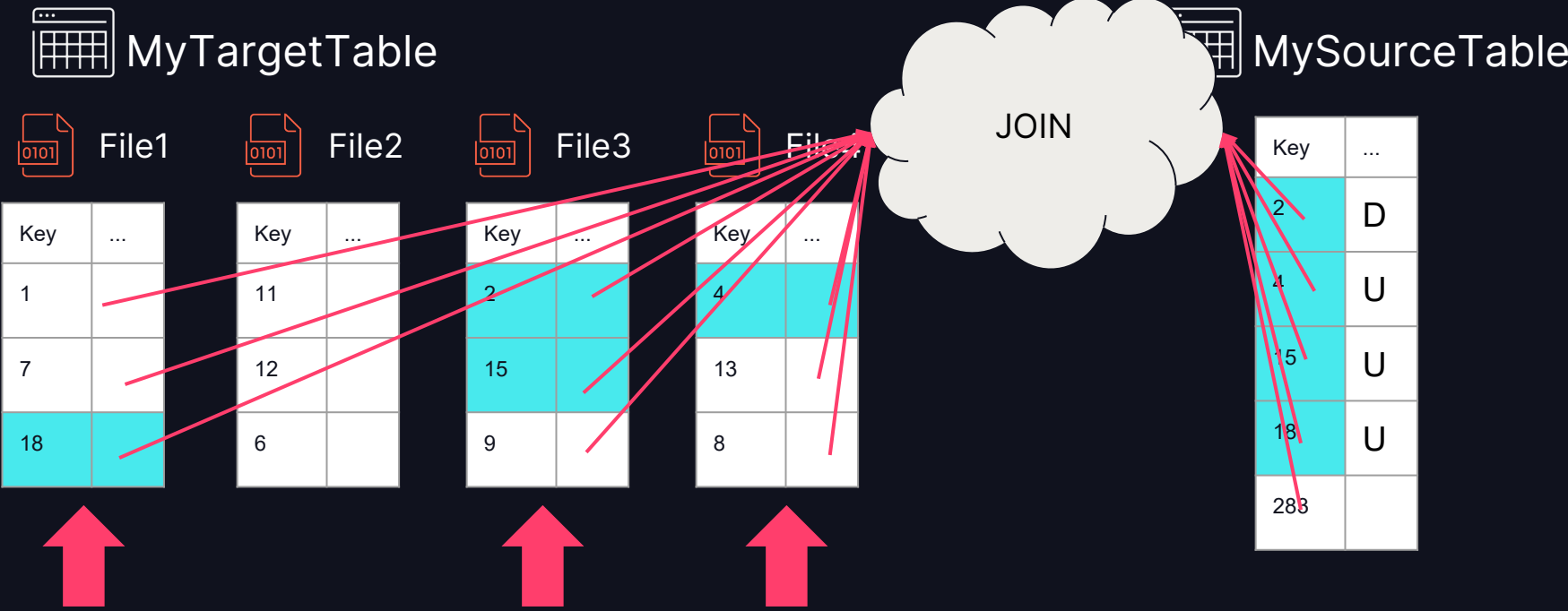
OPTIMIZATION #4

BLOOM FILTER JOINS

BLOOM FILTER JOINS

Legend: Matches

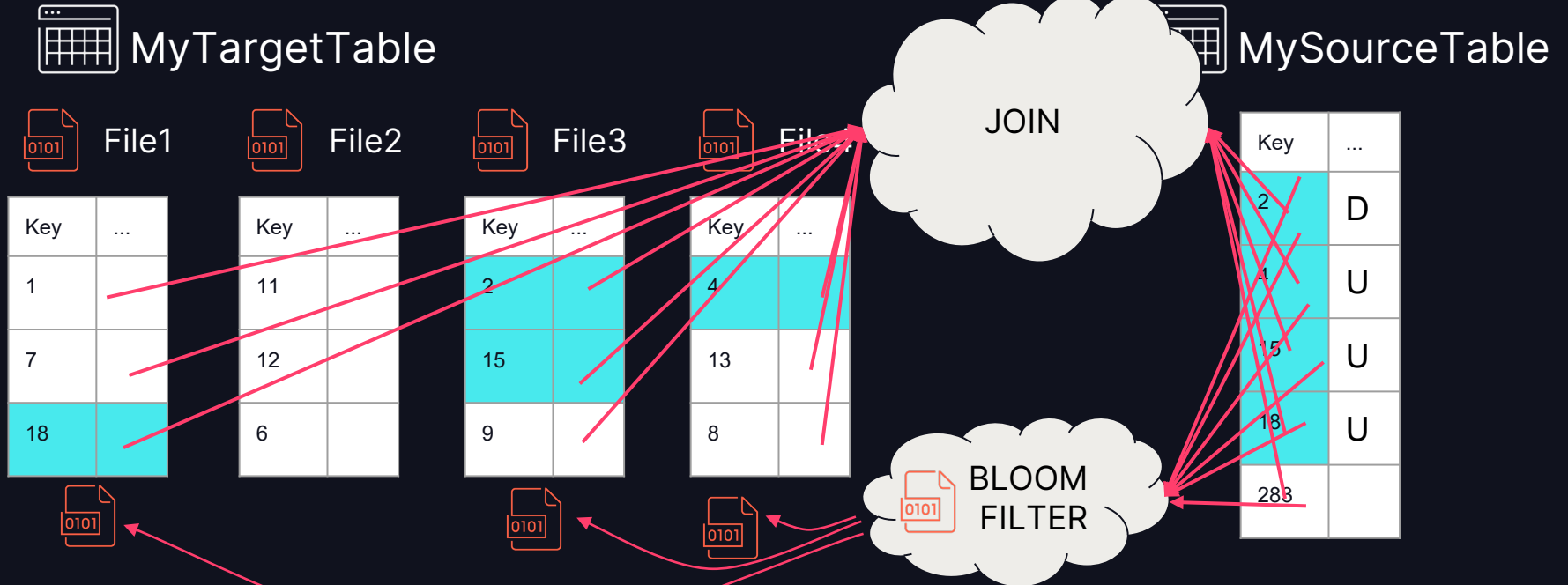
[Recap] STEP 1: Find Files with Matches



BLOOM FILTER JOINS

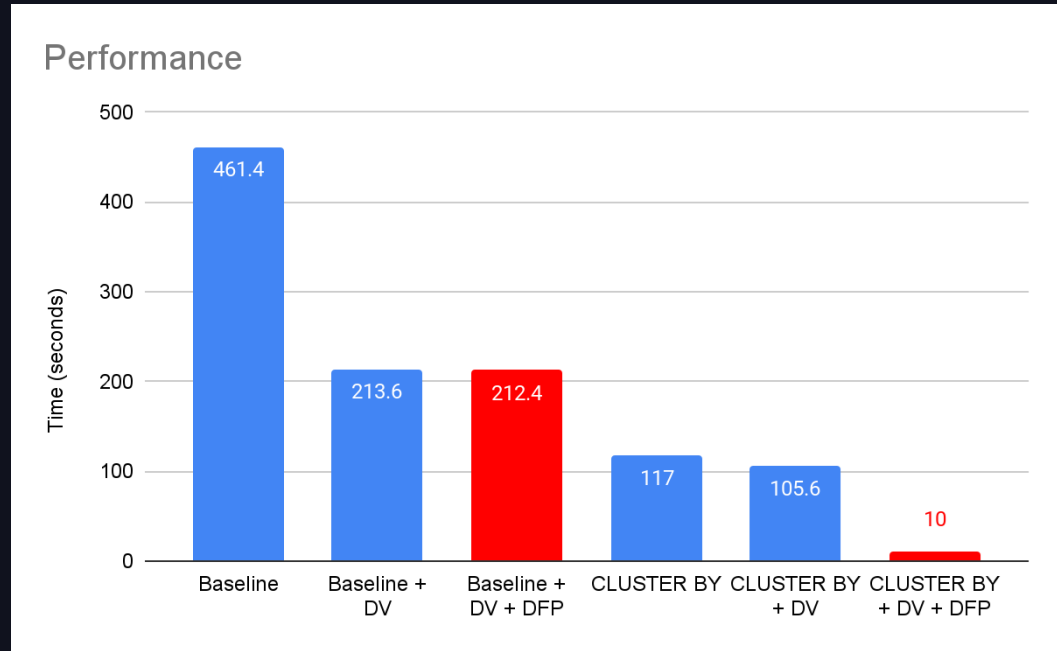
Legend: Matches

Filter early and skip things!



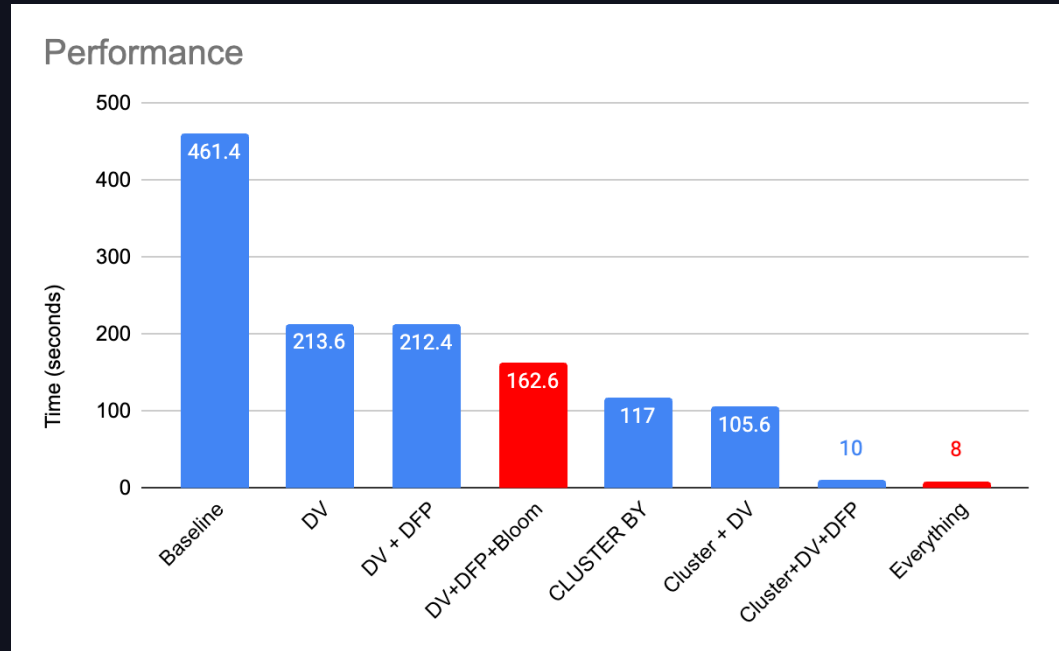
TRACKING PERFORMANCE: FILE PRUNING

- Orders table
- 100GB, 1 year of data
- MERGE 1% of records of the last week
- Overall change 0.02%.



TRACKING PERFORMANCE: FILE PRUNING

- Orders table
- 100GB, 1 year of data
- MERGE 1% of records of the last week
- Overall change 0.02%.



How do I optimize MERGE?

And make that SIMPLE?

STEP 1: **CLUSTER BY** your merge keys using Liquid Clustering

- This enables Dynamic File Pruning and turns on Deletion Vectors.

STEP 2: Use meaningful merge keys. Don't use only random GUIDs.

How do I optimize MERGE?

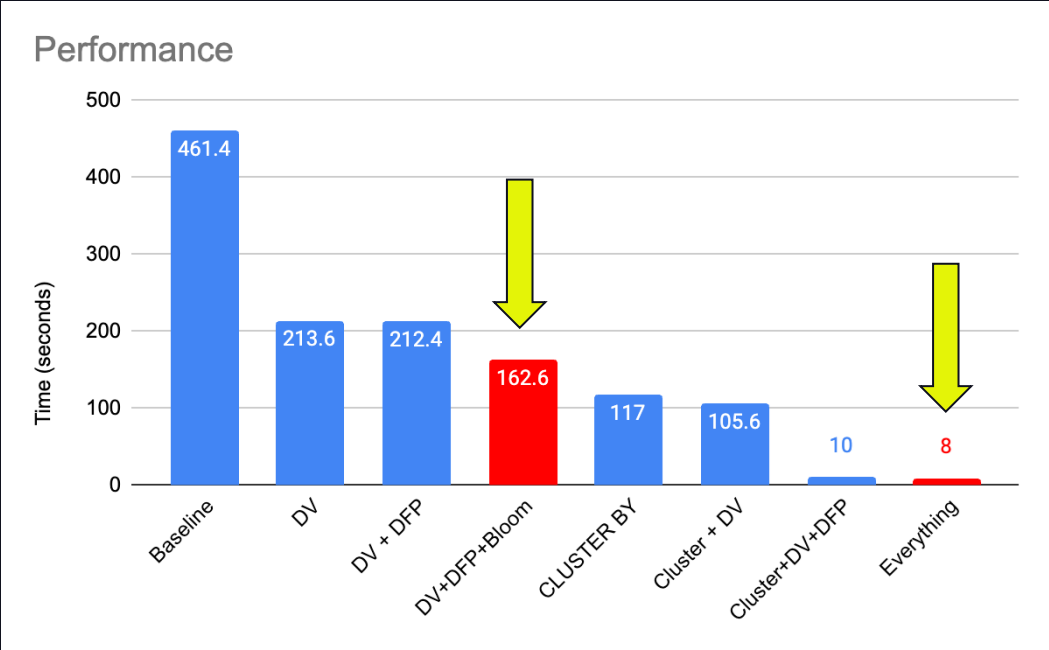
And make that SIMPLE?

STEP 1: **CLUSTER BY** your merge keys using Liquid Clustering

- This enables Dynamic File Pruning and turns on Deletion Vectors.

STEP 2: Use meaningful merge keys. Don't use only random GUIDs.

MEANINGFUL KEYS



MEANINGFUL KEYS

Legend: Matches



MyTargetTable



File1

Key	...
1	
2	
4	

✓ Key:
1..4



File2

Key	...
6	
7	
8	

✗ Key:
6..8



File3

Key	...
12	
11	
9	

✗ Key:
9..12



File4

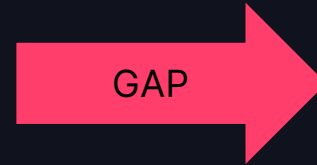
Key	...
18	
13	
15	

✓ Key:
13..15



MySourceTable

Key	...
2	D
4	U
15	U
18	U
283	



MEANINGFUL KEYS

Legend: **Matches**



MyTargetTable



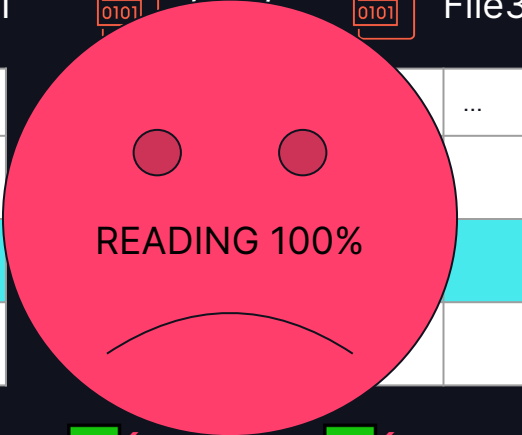
File1

Key	...
1	
2	
4	

✓ Key:
1..4



File2



✓ Key:
6..8



File3

Key	...
9	
12	

✓ Key:
9..12



File4

Key	...
18	
13	
15	

✓ Key:
13..1



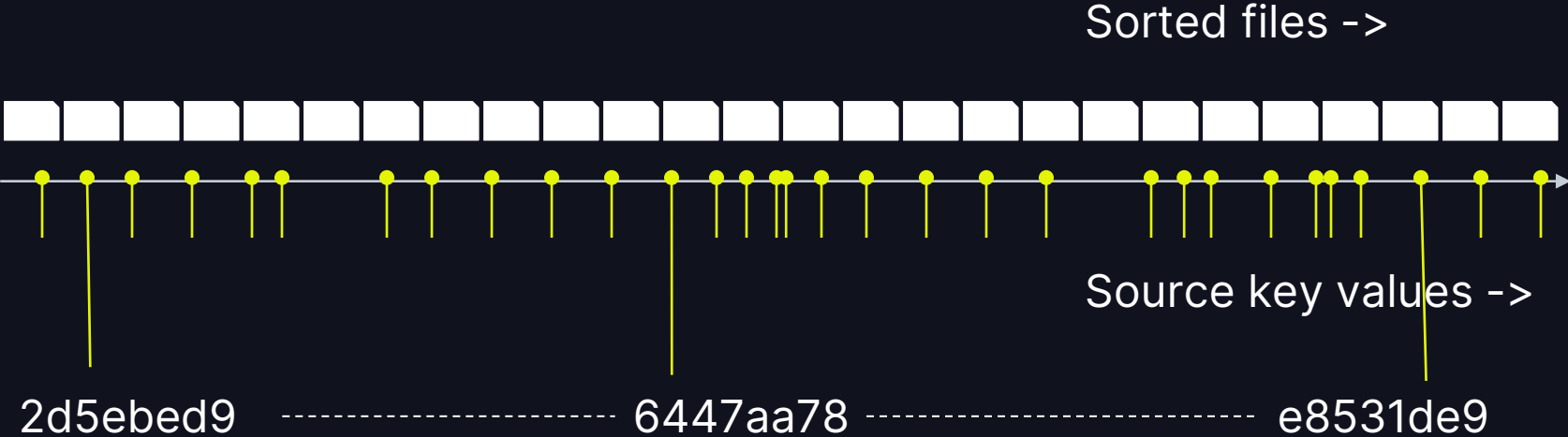
MySourceTable

Key	...
2	D
7	U
11	U
15	U
17	



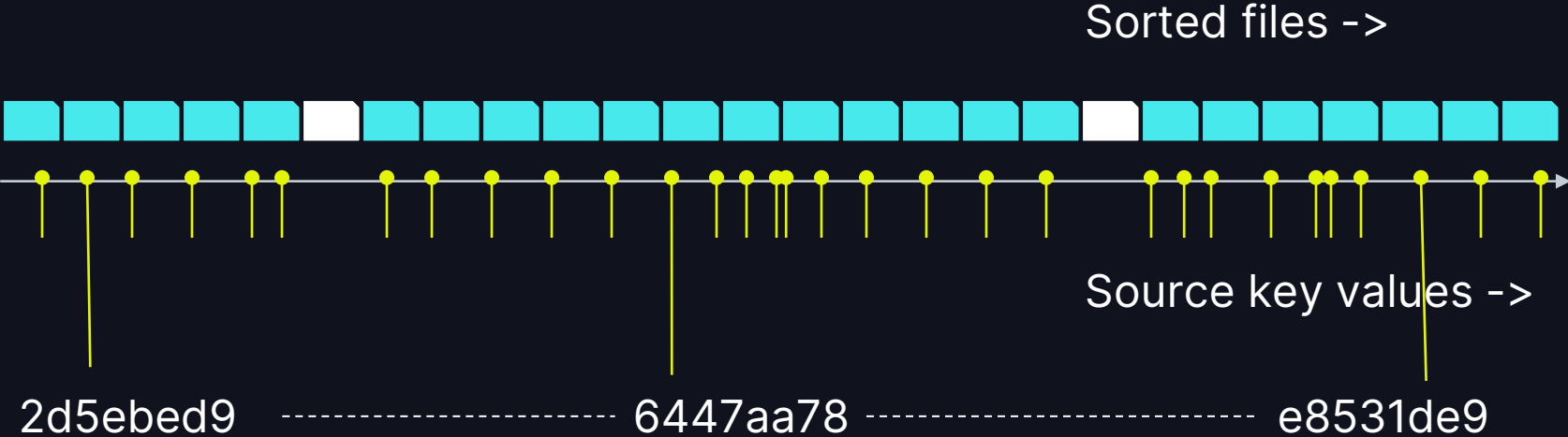
MEANINGFUL KEYS

GUIDs have no meaningful clustering



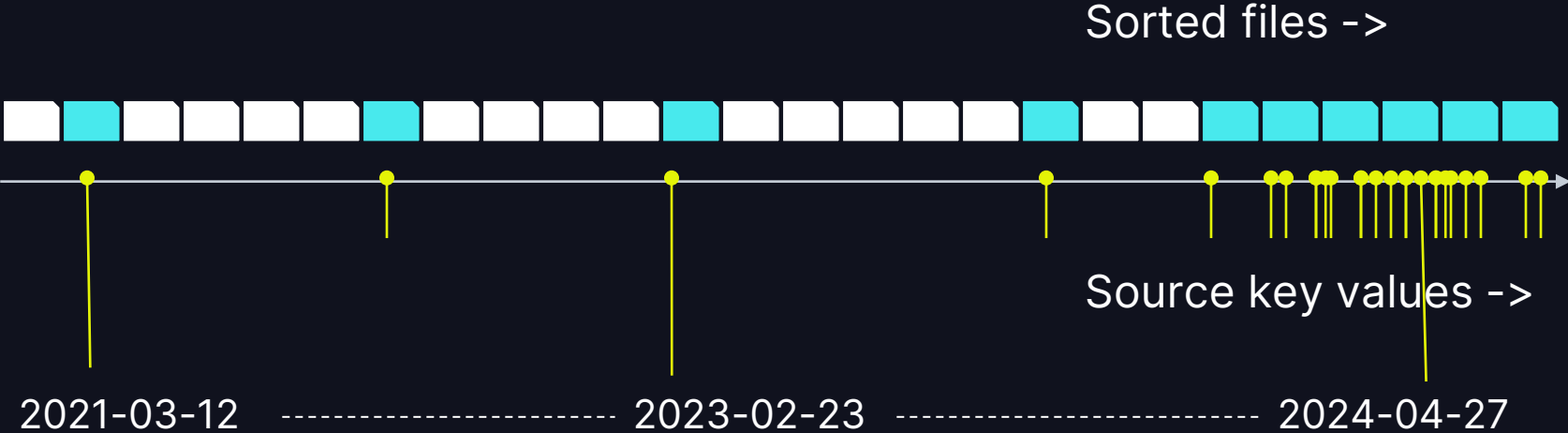
MEANINGFUL KEYS

GUIDs have no meaningful clustering



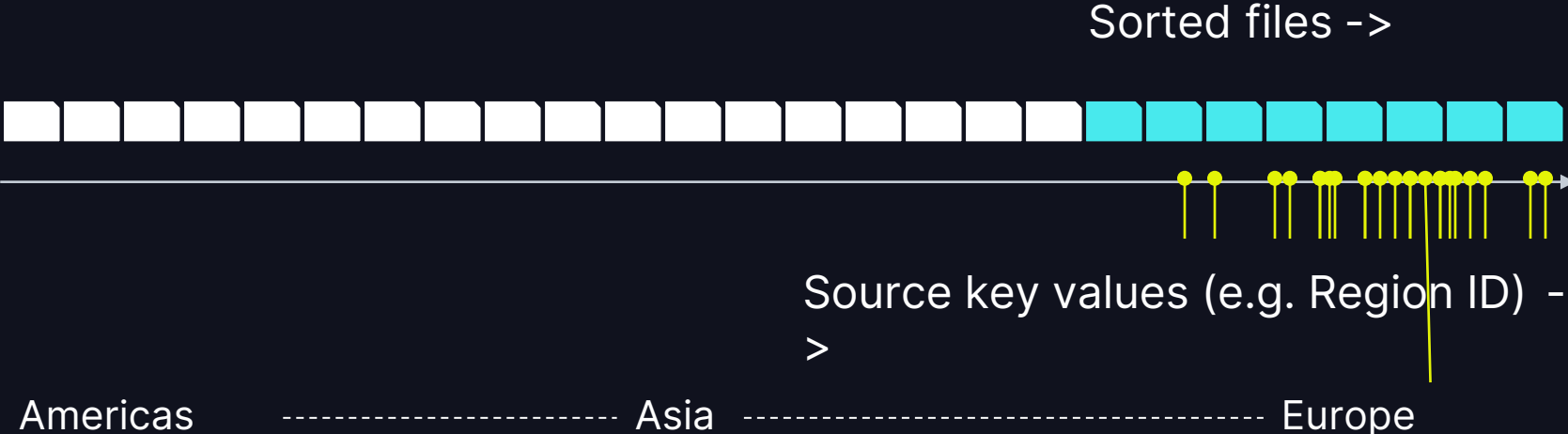
MEANINGFUL KEYS

How about DATES / TIMESTAMPS?



MEANINGFUL KEYS

How about the source of the data?



MEANINGFUL KEYS

Rules of thumb

- Find merge keys with meaningful grouping patterns / clusters of updates
 - Date / Timestamp (updates focus on recent data)
 - Sequential IDs (same: correlates with time)
 - Region / country / branch / source (if ingestion comes in clustered by source)
- Add these keys to your MERGE condition
 - Even if they're not needed for uniqueness. E.g. merge by GUID + region is OK.
- **CLUSTER BY** only on the meaningful keys with update locality.
 - You can leave out the GUIDs and other random IDs. They don't help.
 - Put GUIDs in **CLUSTER BY** only if you filter by them in **SELECT** queries.
- Use **CLUSTER BY AUTO** (when it is released)!

SUMMARY

STEP 1: **CLUSTER BY** your merge keys using Liquid Clustering

STEP 2: Use meaningful merge keys. Don't use only random GUIDs.

DATA+AI SUMMIT