Delta Lake data layout optimization

Sabir Akhadov

Software Engineer

databricks

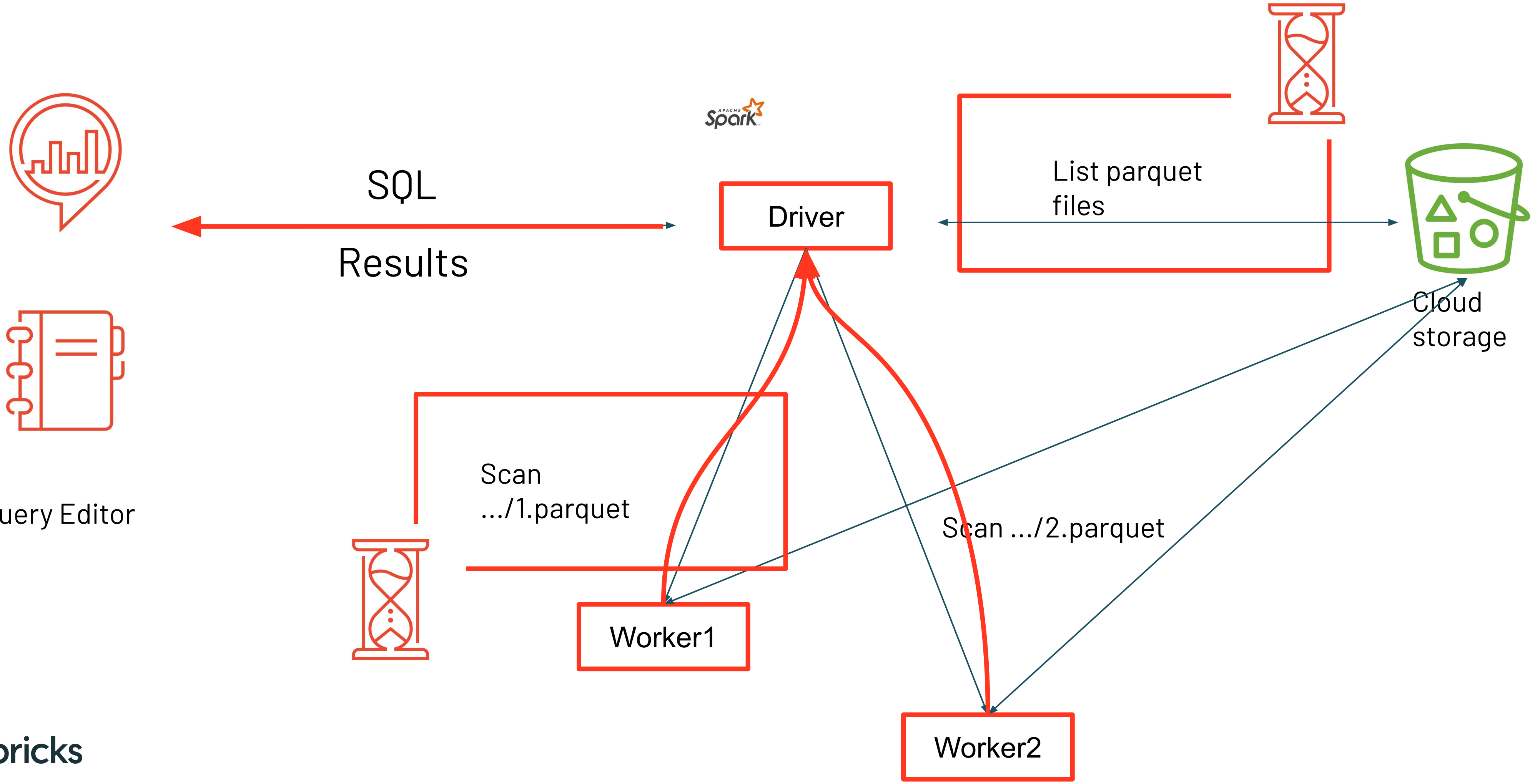
About Databricks and the presenter

Databricks: founded by the original creators of Apache Spark™ with 2000+ employees (engineering in San Francisco, Amsterdam, Toronto, Seattle and Berlin).

Sabir Akhadov: Software Engineer @ Databricks Amsterdam since 2019

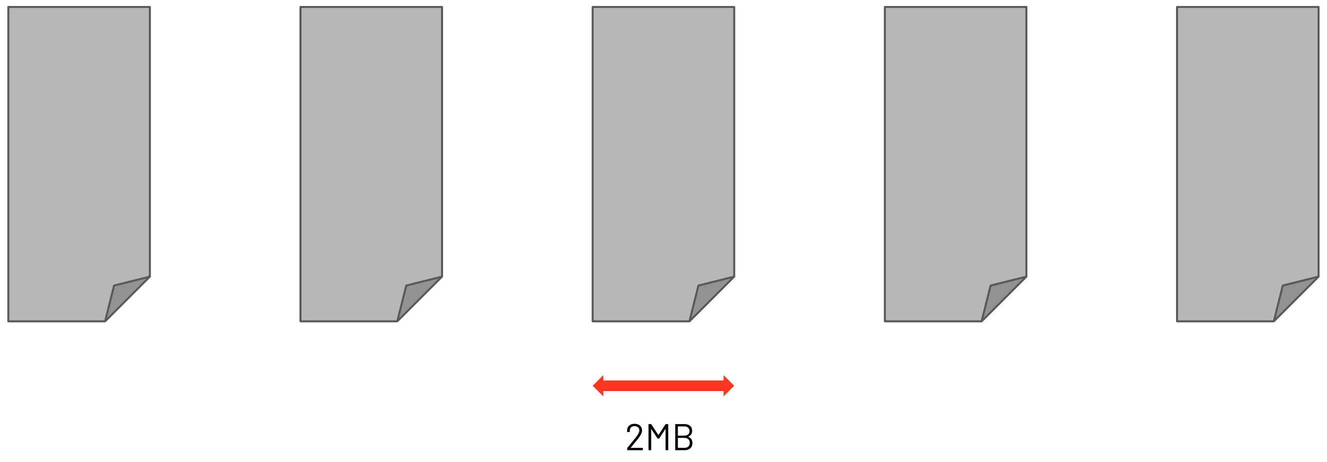
Data layouts team - data organization for the best read/write performance

Query execution Spark + Parquet



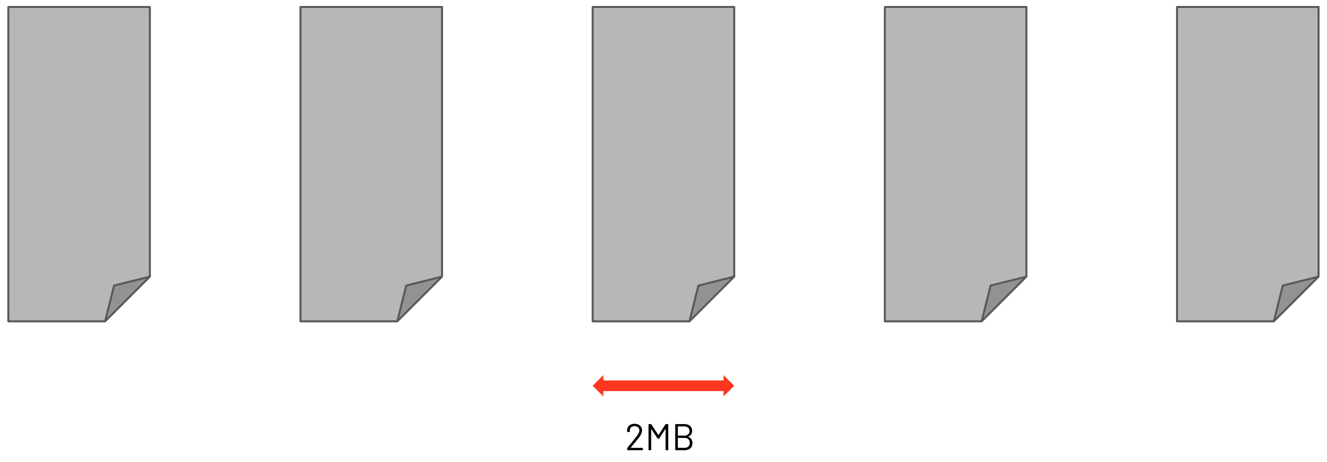
Data layouts

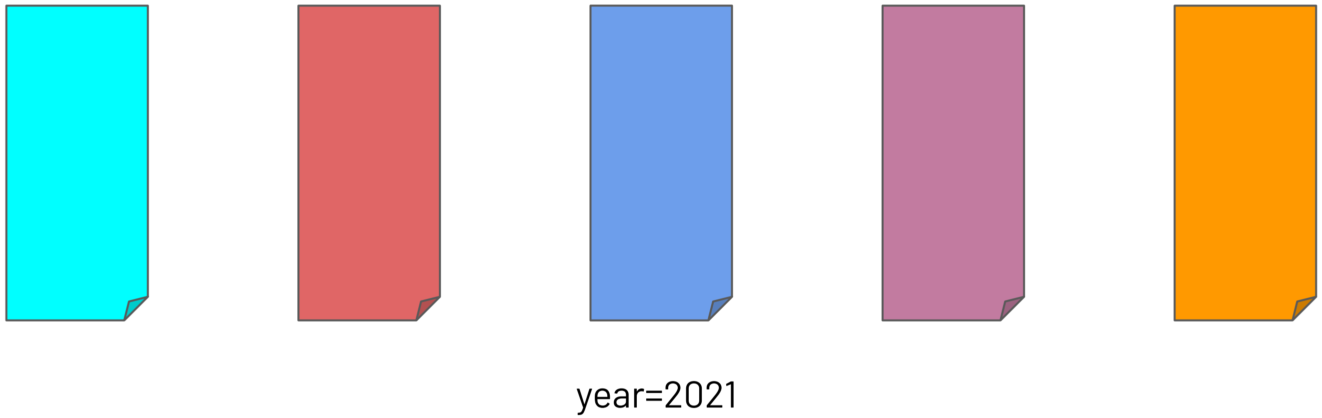
FILE SIZE - I/O overhead



Data layouts

FILE SIZE - I/O overhead

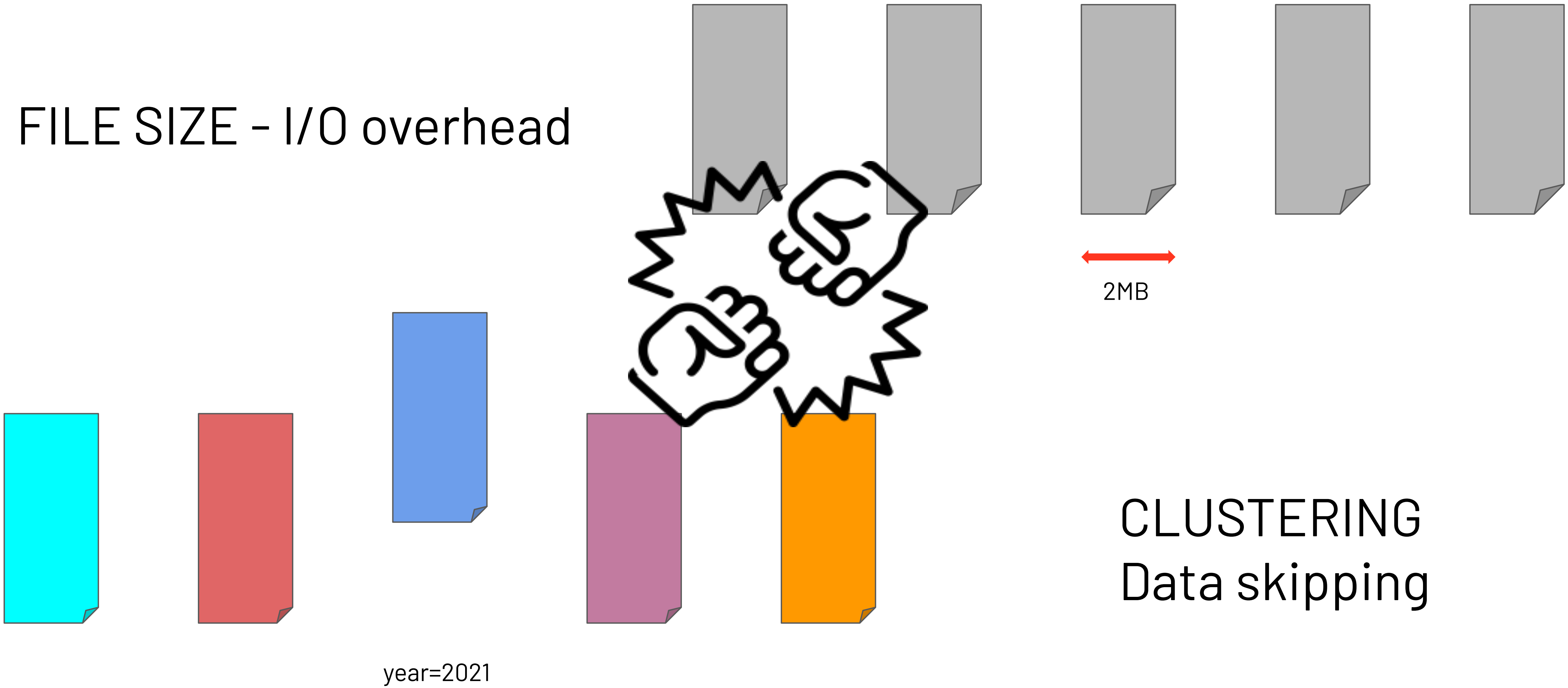




CLUSTERING
Data skipping

Data layouts

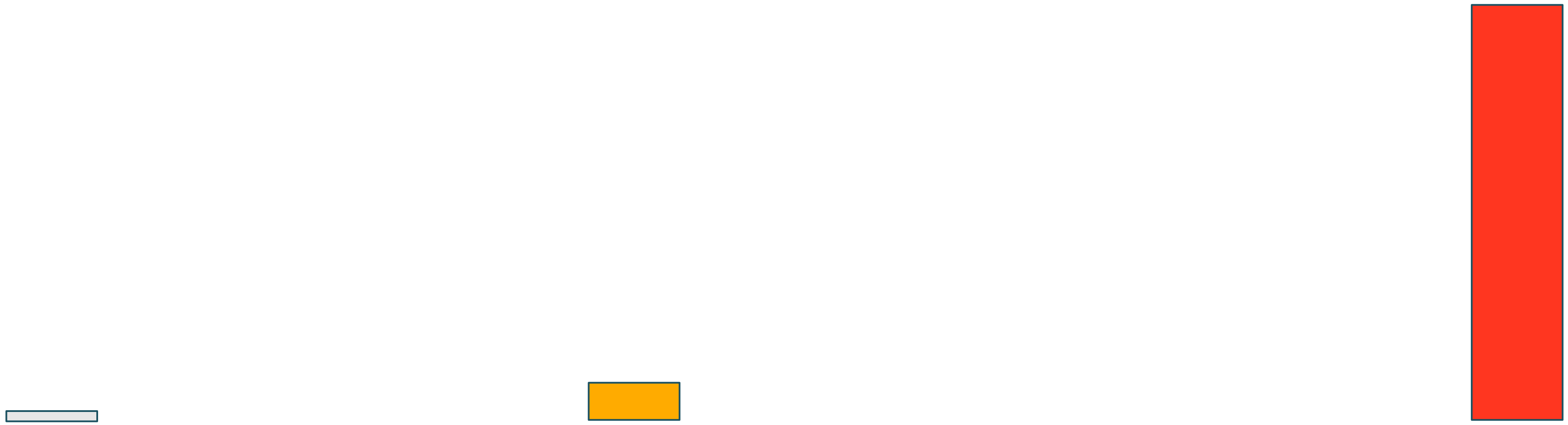
FILE SIZE - I/O overhead



CLUSTERING

Data skipping

Per-file overhead



~40-100ms

S3 fetch request

3-10 seconds

Scan 16 files

x 250MB

2-2.5minutes

Scan 16000

x 250KB

4 workers, 16 cores cluster

This talk

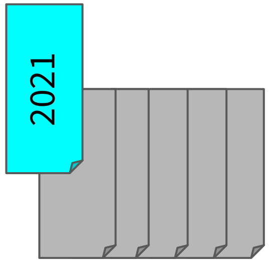
- Parquet tables, data skipping
- Delta lake
- Advisor/Auto optimizations
- Star schema optimizations

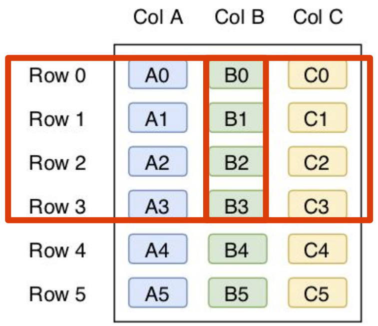
Parquet tables

databricks

Data filtering

- Partitions
- Buckets
- Files
- File “chunks”

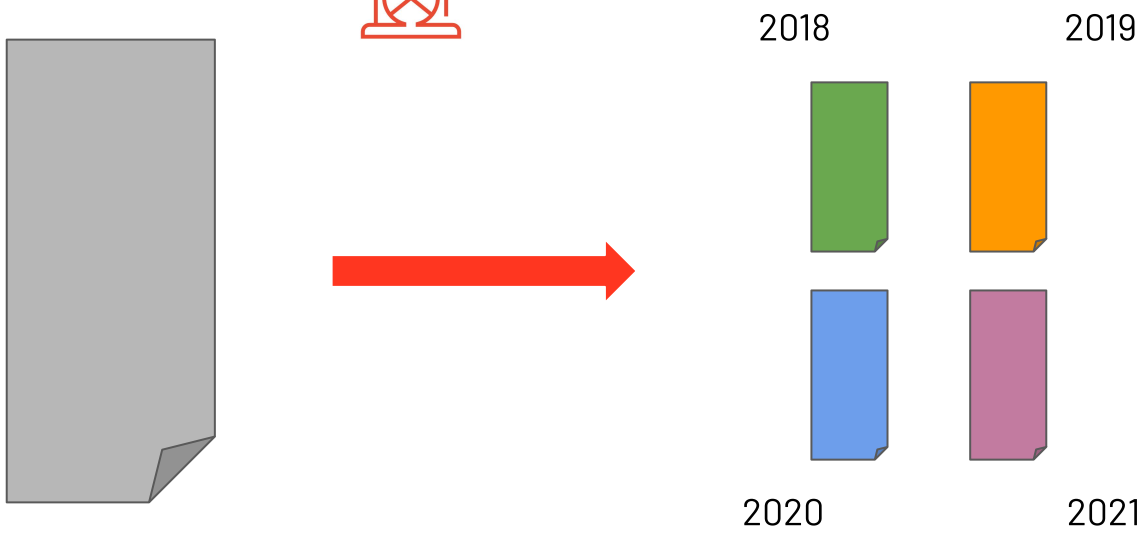




Partitioning

LIST FILES IN /

LIST FILES IN /2021/



year >= 2018 and year <= 2021

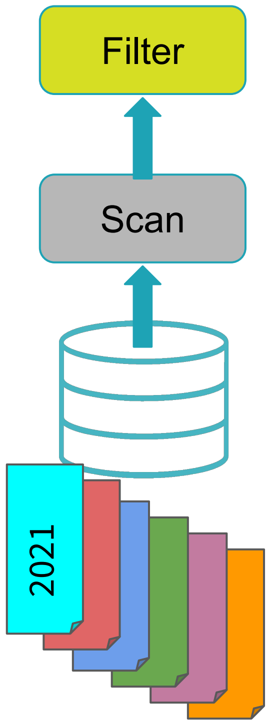
```
year=2021/
    file1.parquet
    file2.parquet
year=2020/
    file3.parquet
year=2019/
    ...etc...
```

```sql
SELECT * FROM table WHERE year = 2021
```

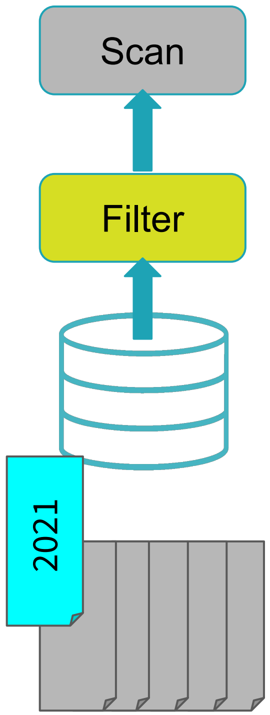
databricks

Static partition pruning

```
SELECT * FROM table WHERE year = 2021
```



Partition files



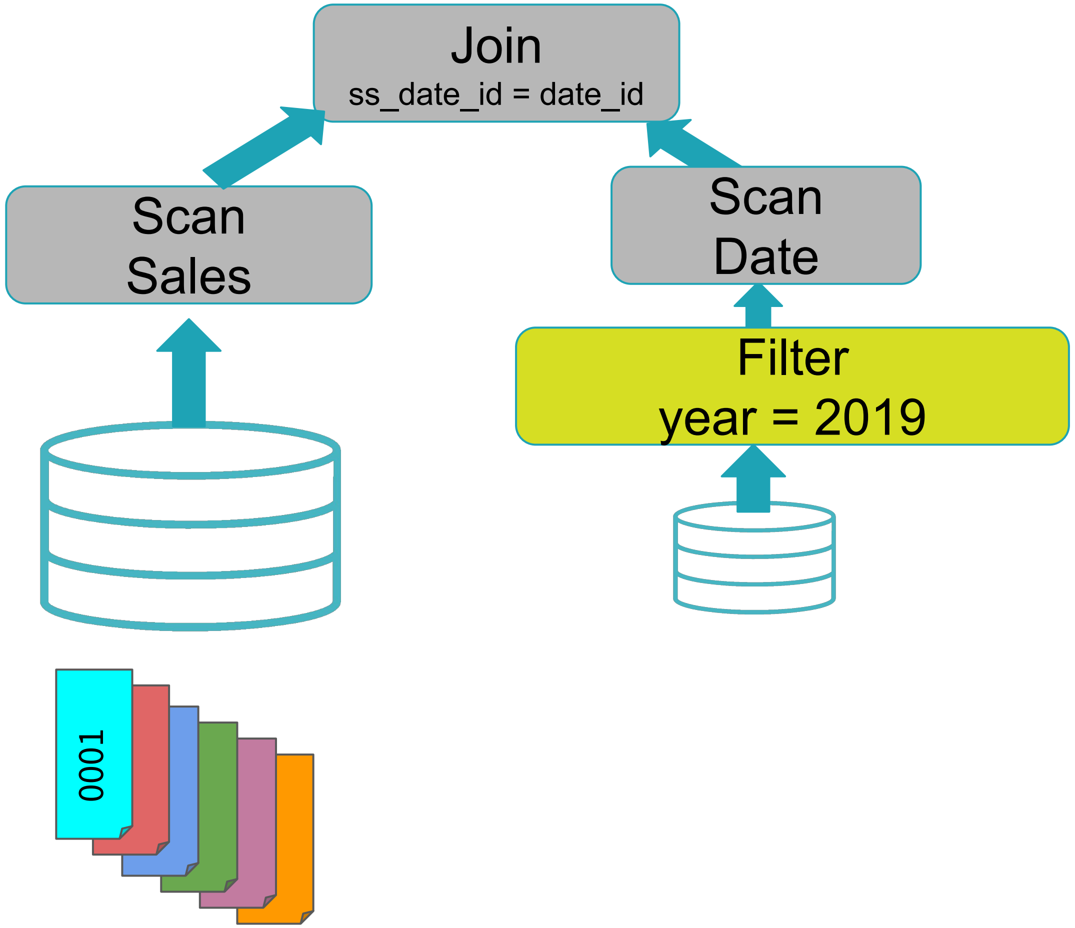
Partition files

year=2021/
  file1.parquet
  file2.parquet
year=2020/
  file3.parquet
year=2019/
  ...etc...

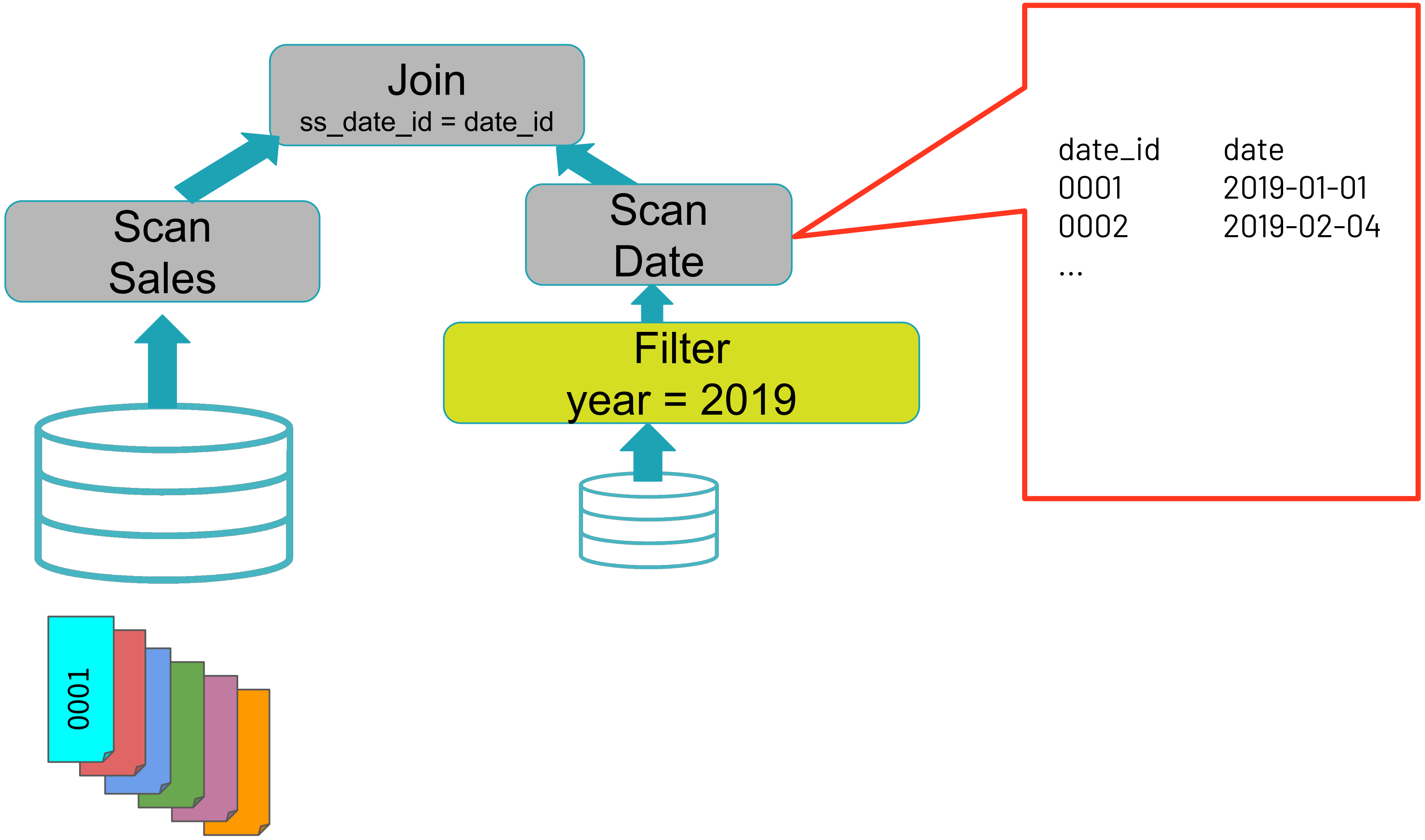
databricks

Dynamic partition pruning

ss_date_id=0001/
  file1.parquet
  file2.parquet
ss_date_id=0002/
  file3.parquet
ss_date_id=0003/
  ...etc...

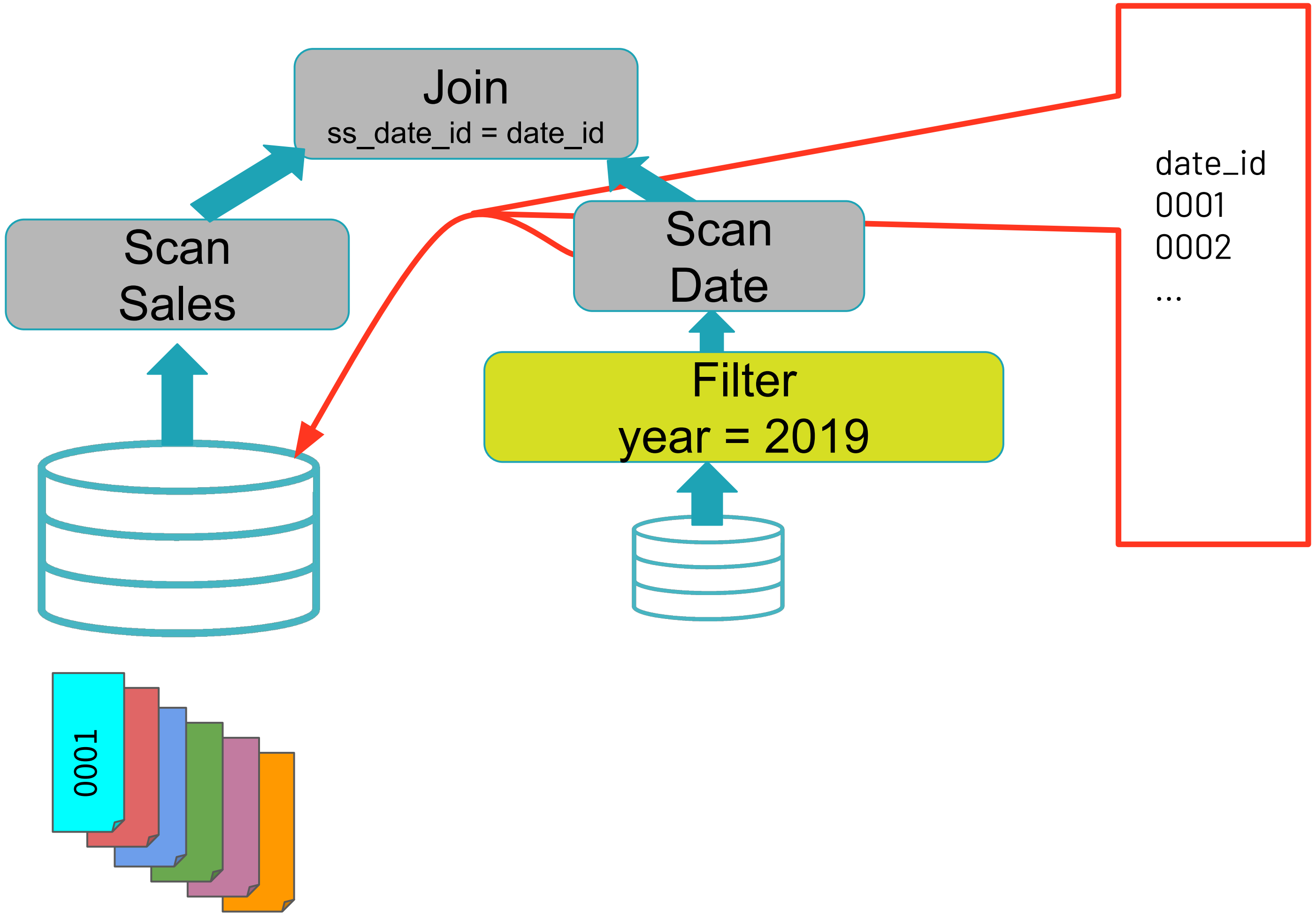


Dynamic partition pruning



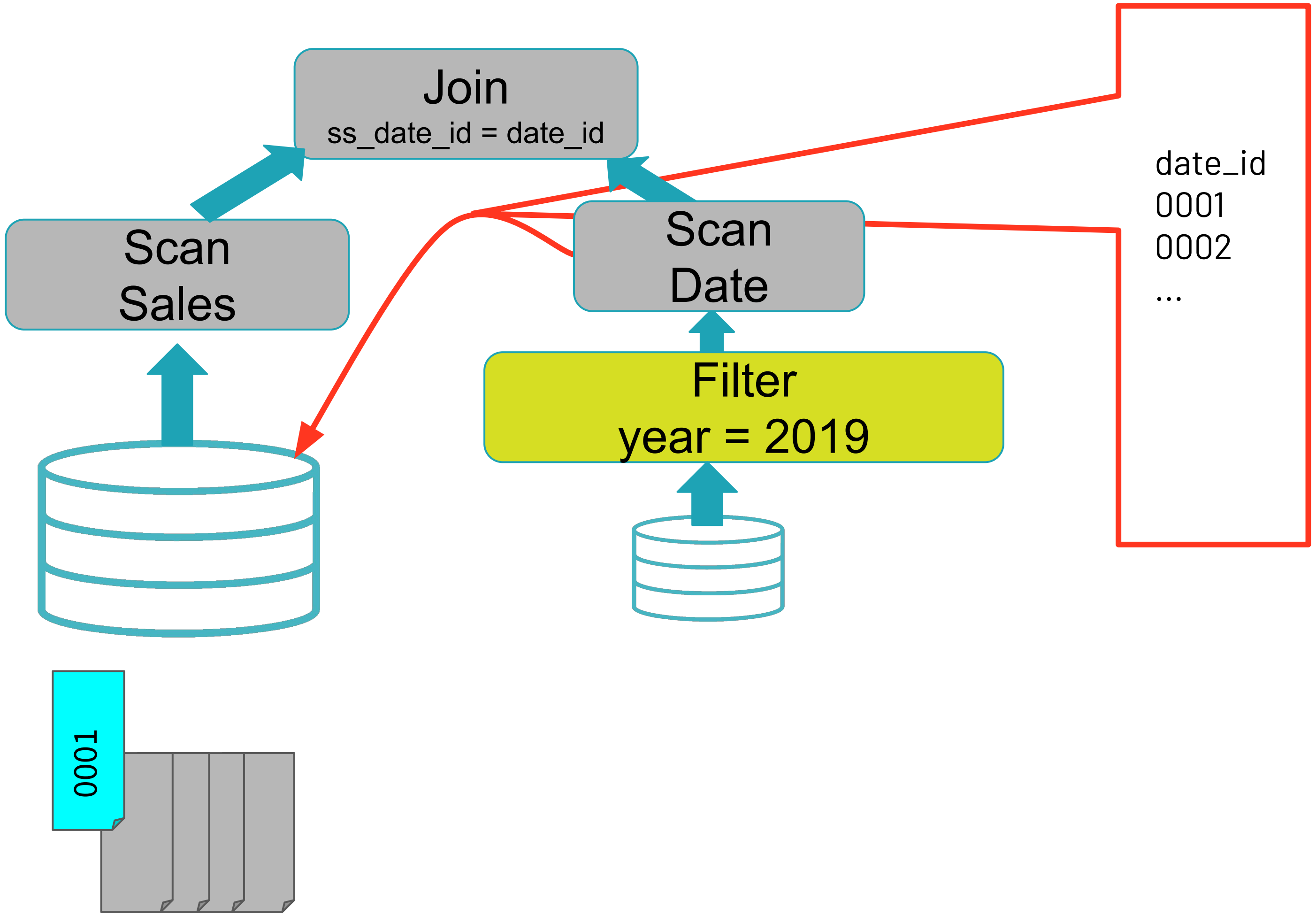
databricks

Dynamic partition pruning



databricks

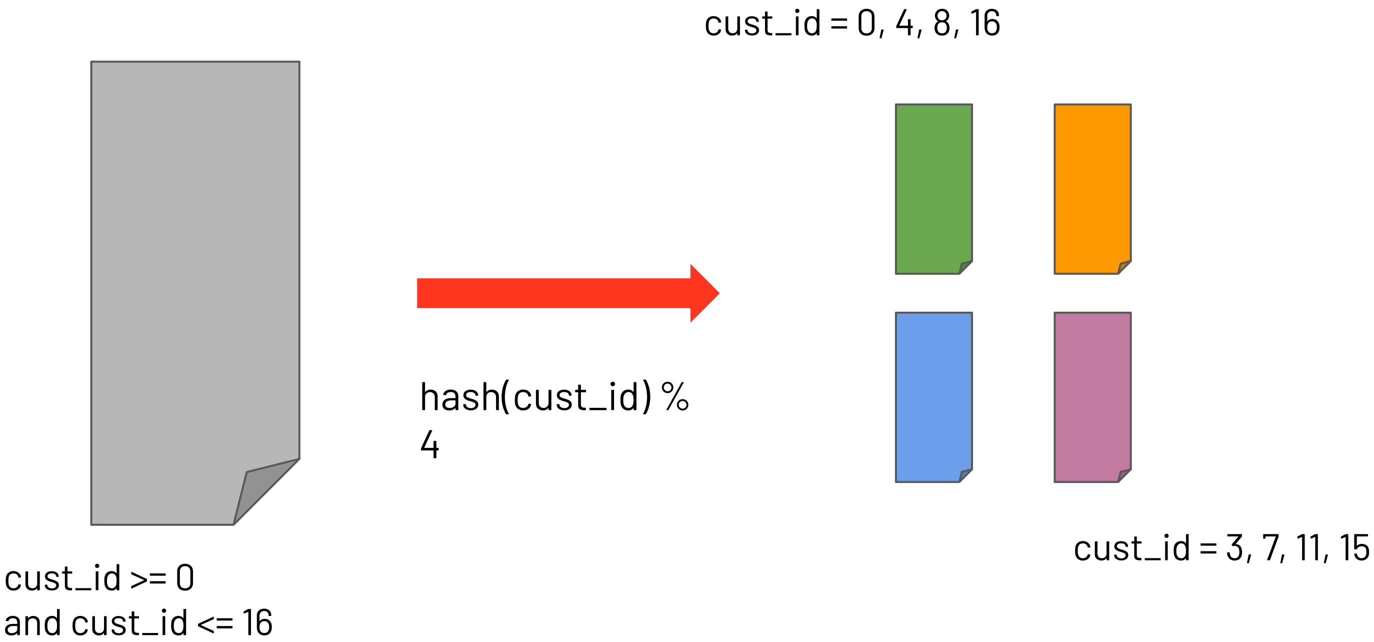
Dynamic partition pruning



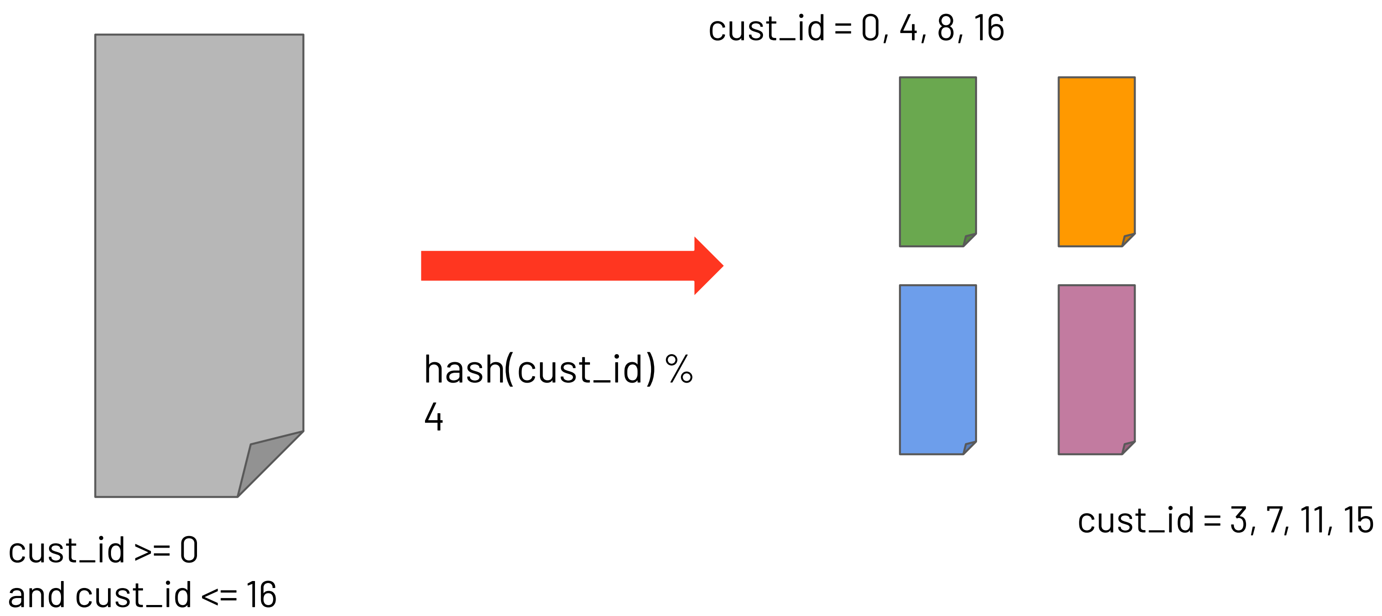
Partitioning disadvantages

- Low cardinality columns only - overpartitioning
- Data skew
- Underpartitioning - large files
- Static - data evolves, know your workloads at table creation

Bucketing

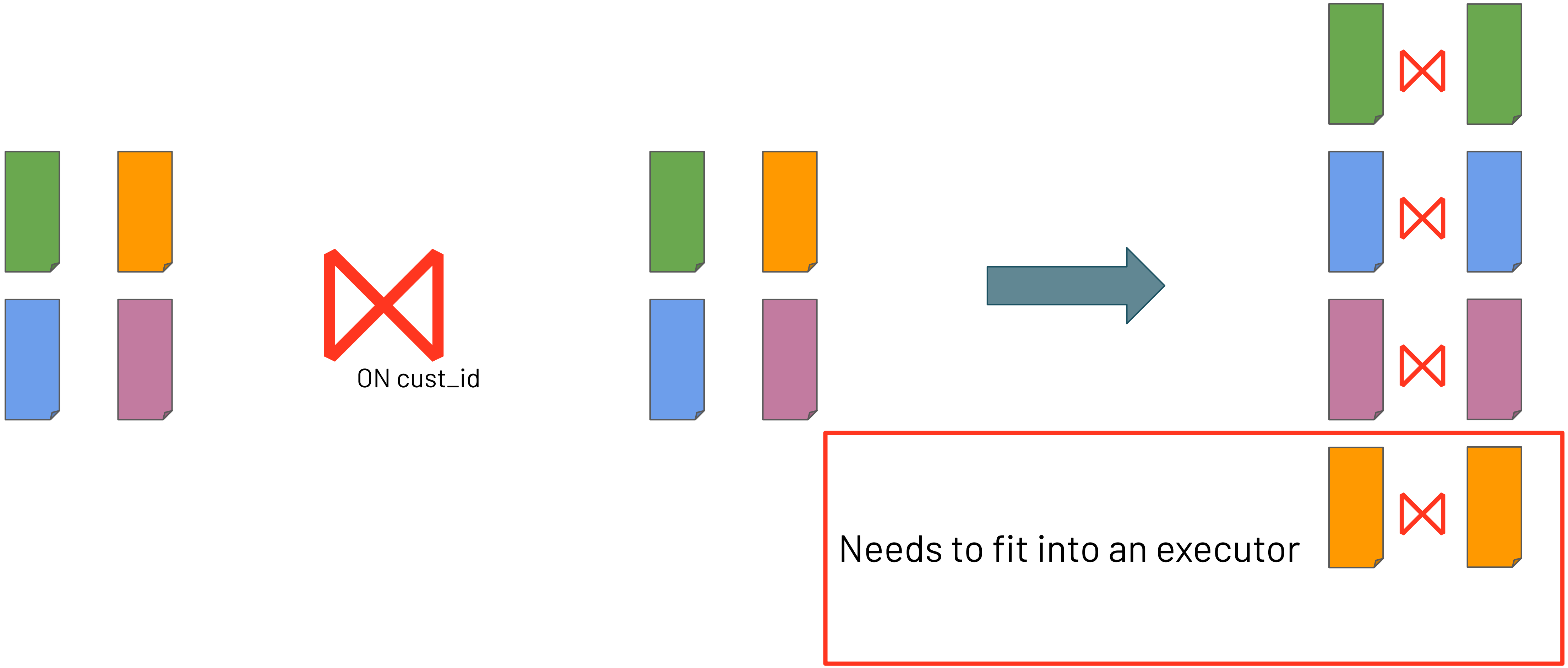

cust_id = 0, 4, 8, 16
hash(cust_id) %
4
cust_id >= 0
and cust_id <= 16
cust_id = 3, 7, 11, 15

Bucketing



- Good for high cardinality columns - every write produces exactly the same number of files
- Pinpoint queries cust_id = 1

Bucketed join



Bucketing disadvantages

- Static, hard to evolve
  - How many buckets?
  - Adding new bucket key requires rewrite
- Data skew
- No range scans

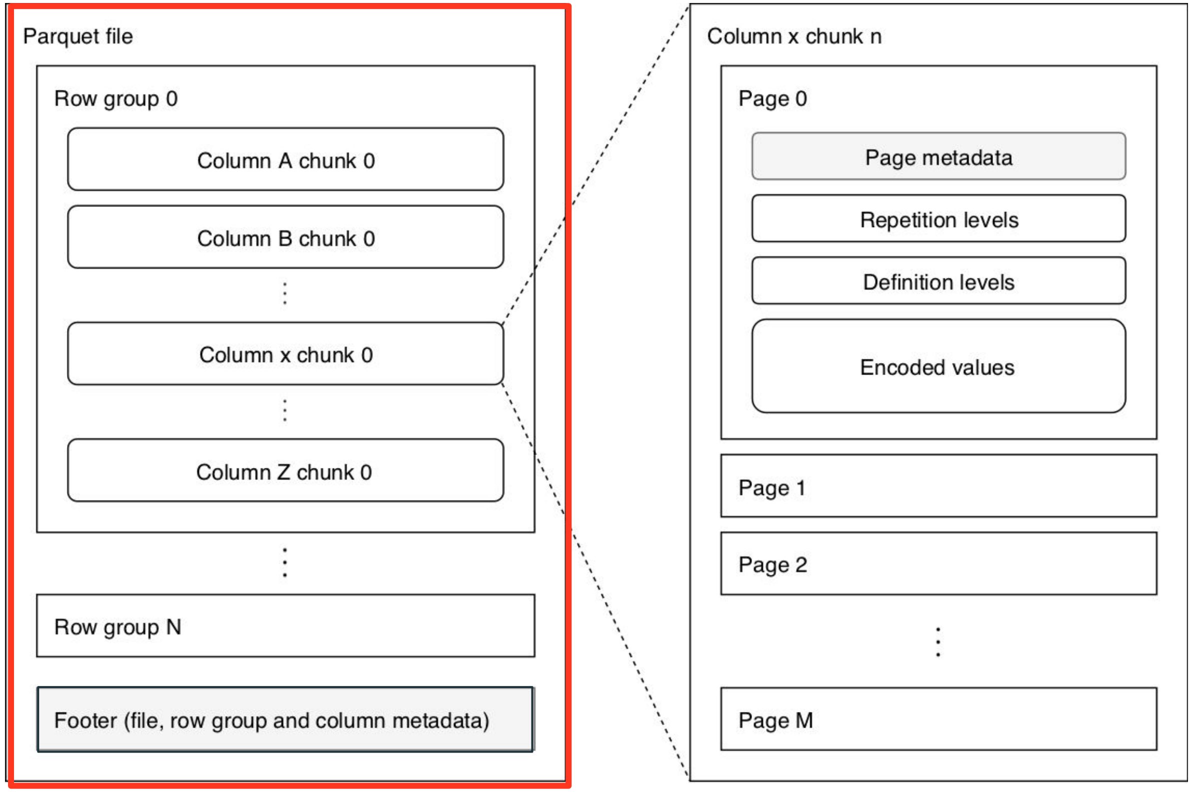
File skipping

- Determine from parquet footer if a file ***could contain*** values

databricks

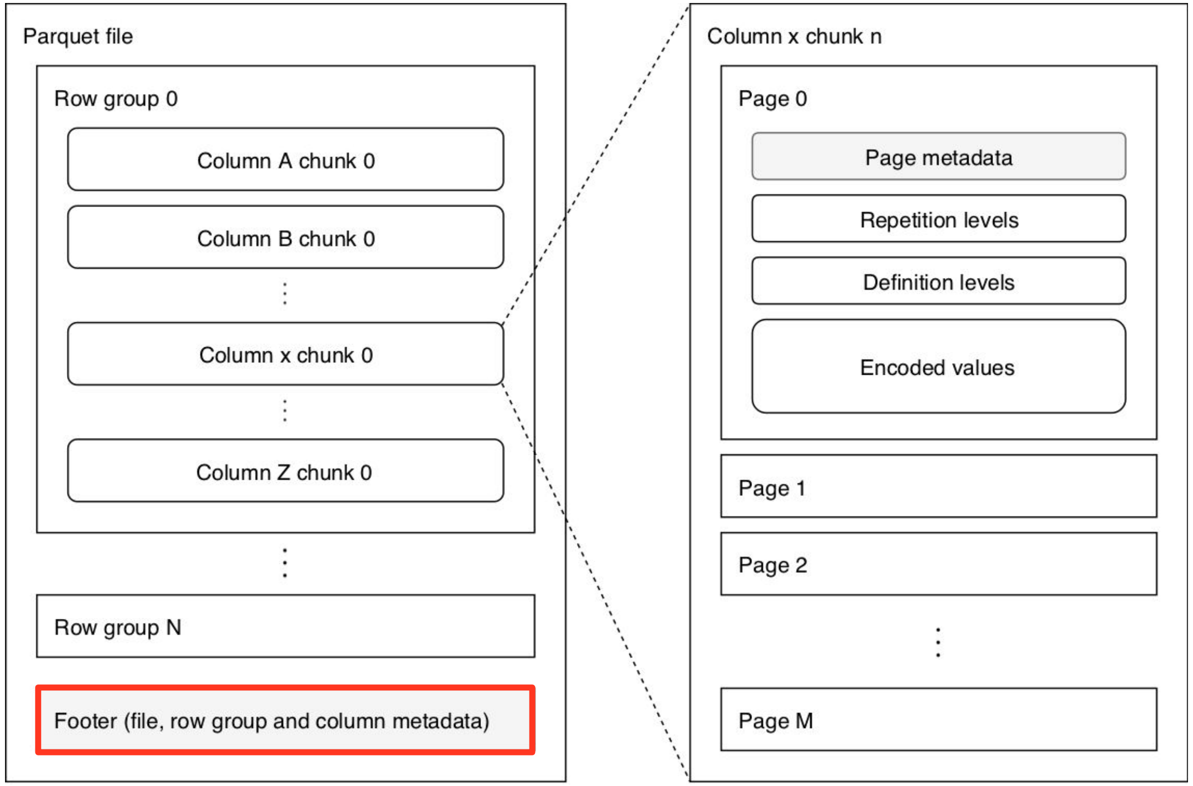
Parquet internals

- File



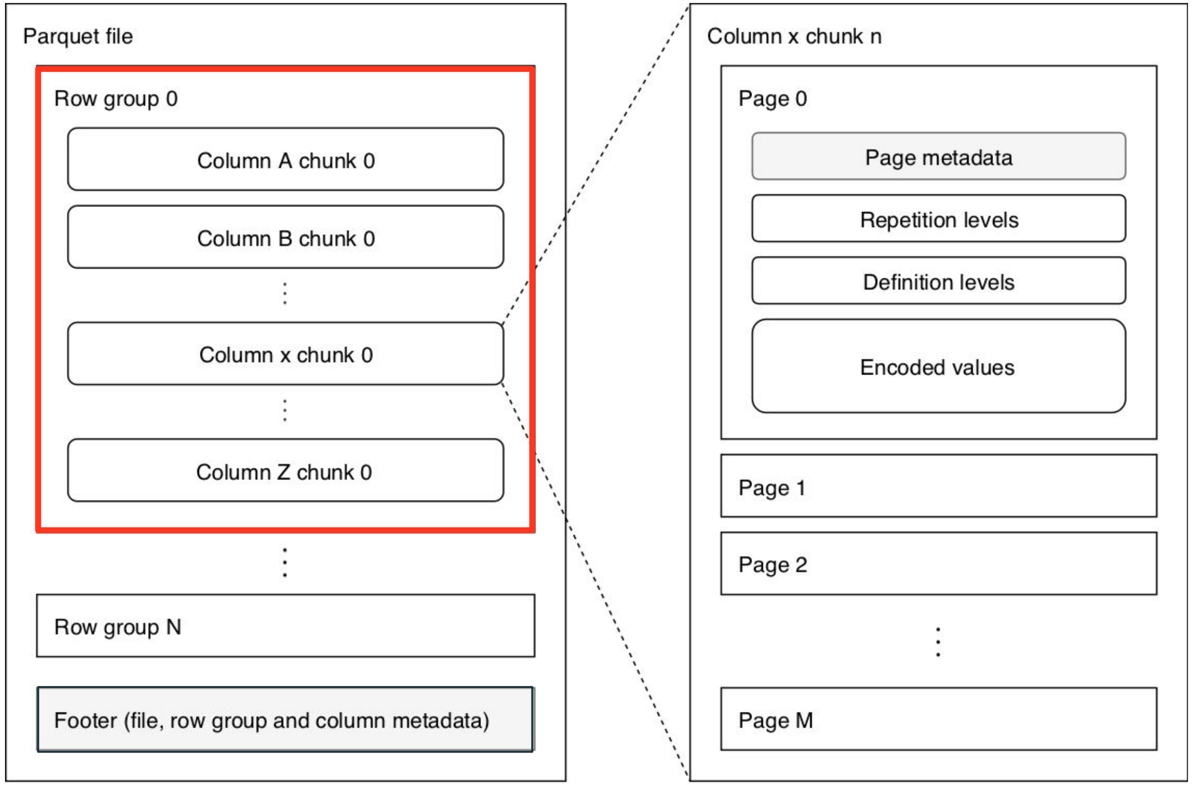
Parquet internals

- File
  - Footer
    - row-group min max stats



Parquet internals

- File
  - Footer
    - row-group min max stats
  - Row group
    - Column chunks
      - Pages - stats



databricks

File skipping

- Min-Max row-group statistics
- Determine from footer if a row group ***could contain*** values

```
SELECT * FROM table WHERE x > 5

Row-group 0: x: [min: 0, max: 9]
Row-group 1: x: [min: 3, max: 7]
Row-group 2: x: [min: 1, max: 4]
```

File skipping

- Min-Max row-group statistics
- Determine from footer if a row group ***could contain*** values

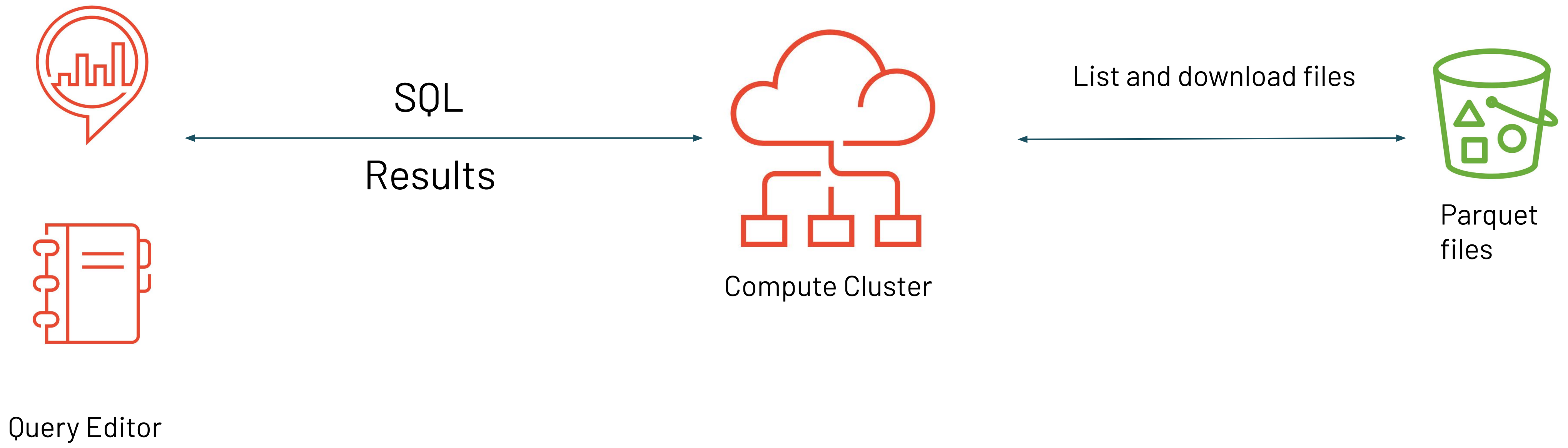
I have to read a file in order to skip it?!

Need to schedule Spark tasks to read files.

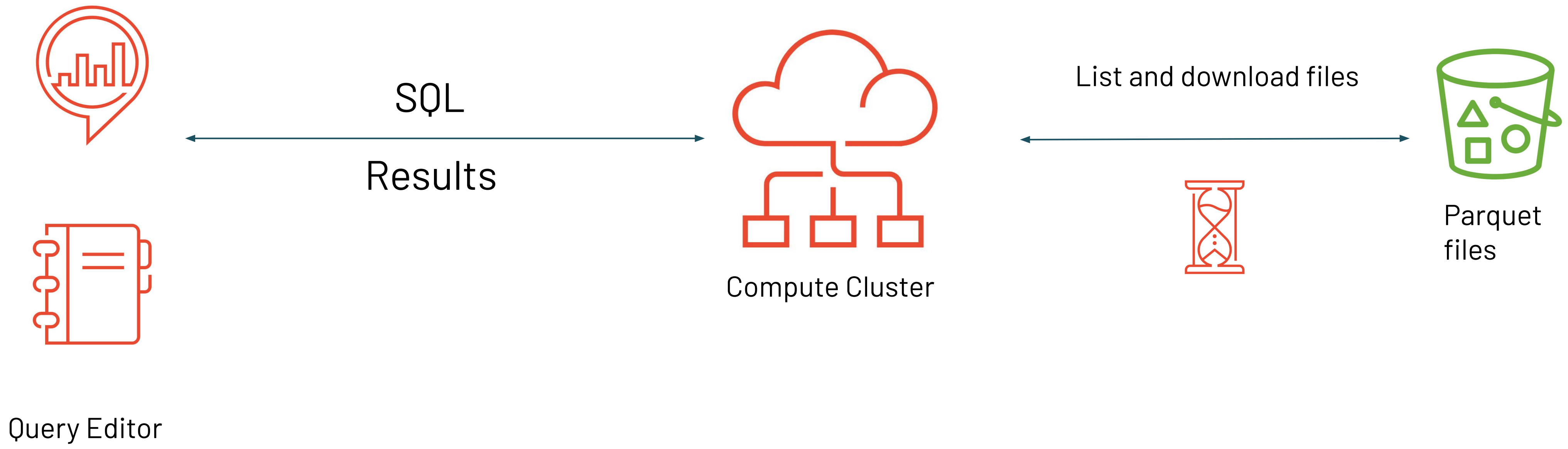
Delta Tables

databricks

File listing is slow

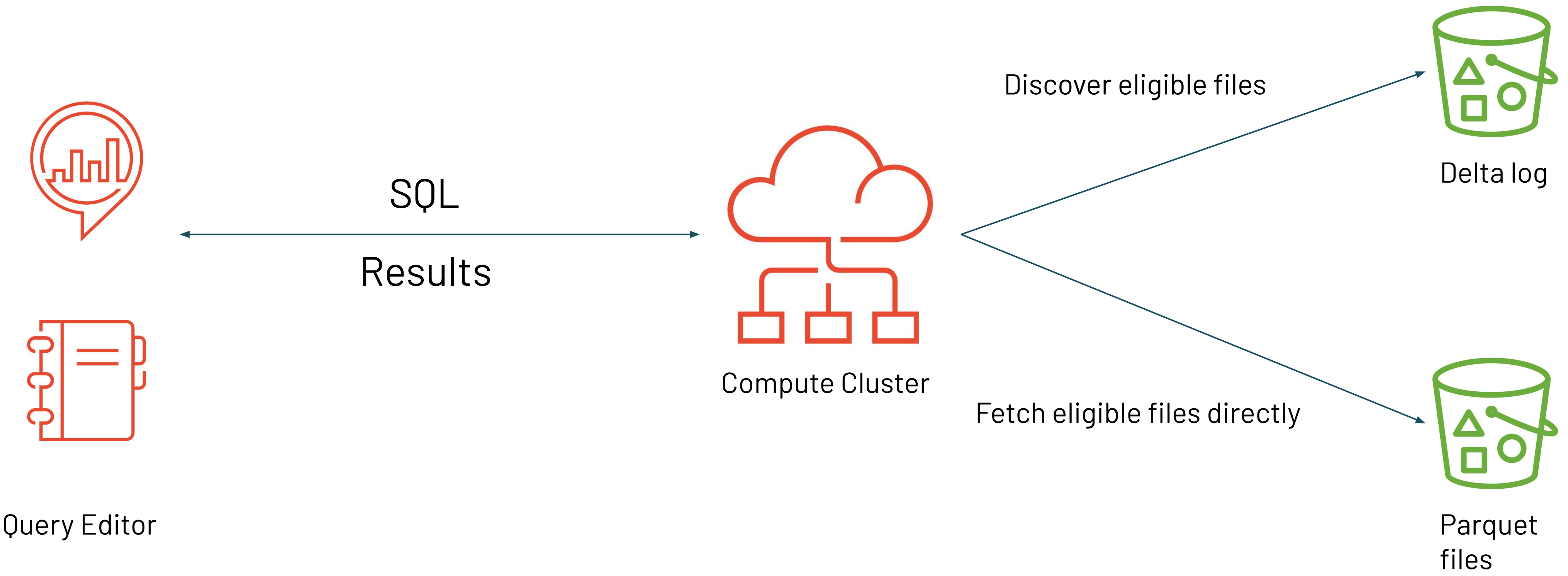


File listing is slow



databricks

Delta Log


SQL
Results
Query Editor
Compute Cluster
Discover eligible files
Fetch eligible files directly
Delta log
Parquet files

Delta Table

DELTA LAKE

Transaction log

Partition directories(optional)

Data files

- my_table/
  - _delta_log/
    - 000000.json
    - 000001.json
    - ....
    - 000010.checkpoint.parquet
  - year=2021
    - file-01.parquet
    - file-02.parquet
    - ...
  - year=2020
    - file-00.parquet

    ...

databricks

Delta log functions

- ACID transactions
- History - time travel
- **Files present**
- **File statistics**

Inspecting delta log - added files

```
import com.databricks.sql.transaction.tahoe.DeltaLog
import org.apache.spark.sql.catalyst.TableIdentifier

val db = "tpcds_sf100_delta"
val table = "store_sales"
val deltaLog = DeltaLog.forTable(spark,TableIdentifier(table, Some(db)))
deltaLog.snapshot.allFiles.show(3)
```

```
+--------------------+--------------------+-------+----------------+----------+--------------------+----+
|                path|     partitionValues|   size|modificationTime|dataChange|               stats|tags|
+--------------------+--------------------+-------+----------------+----------+--------------------+----+
|ss_sold_date_sk=2...|{ss_sold_date_sk ...|4234199|   1610537322000|     false|{"numRecords":880...|null|
|ss_sold_date_sk=2...|{ss_sold_date_sk ...|4162372|   1610537322000|     false|{"numRecords":866...|null|
|ss_sold_date_sk=2...|{ss_sold_date_sk ...|4157255|   1610537323000|     false|{"numRecords":865...|null|
+--------------------+--------------------+-------+----------------+----------+--------------------+----+
```

Inspecting delta log - added files

```scala
import com.databricks.sql.transaction.tahoe.DeltaLog
import org.apache.spark.sql.catalyst.TableIdentifier

val db = "tpcds_sf100_delta"
val table = "store_sales"
val deltaLog = DeltaLog.forTable(spark,TableIdentifier(table, Some(db)))
deltaLog.snapshot.allFiles.show(3)
```

```
+--------------------+--------------------+-------+----------------+----------+--------------------+----+
|                path|     partitionValues|   size|modificationTime|dataChange|               stats|tags|
+--------------------+--------------------+-------+----------------+----------+--------------------+----+
|ss_sold_date_sk=2...|{ss_sold_date_sk ...|4234199|   1610537322000|     false|{"numRecords":880...|null|
|ss_sold_date_sk=2...|{ss_sold_date_sk ...|4162372|   1610537322000|     false|{"numRecords":866...|null|
|ss_sold_date_sk=2...|{ss_sold_date_sk ...|4157255|   1610537323000|     false|{"numRecords":865...|null|
+--------------------+--------------------+-------+----------------+----------+--------------------+----+
```

Added file stats

```
{
 "numRecords": 88099,
  "minValues": {

...
    "ss_net_profit": -9592
  },
  "maxValues": {
...
    "ss_net_profit": 8301
  },
...
  }
}
```

databricks

Added file stats

```
{
 "numRecords": 88099,
  "minValues": {

...
    "ss_net_profit": -9592
  },
  "maxValues": {
...
    "ss_net_profit": 8301
  },
...
  }
}
```

```
SELECT * FROM store_sales WHERE ss_net_profit < 0
```

Improving data layout for skipping

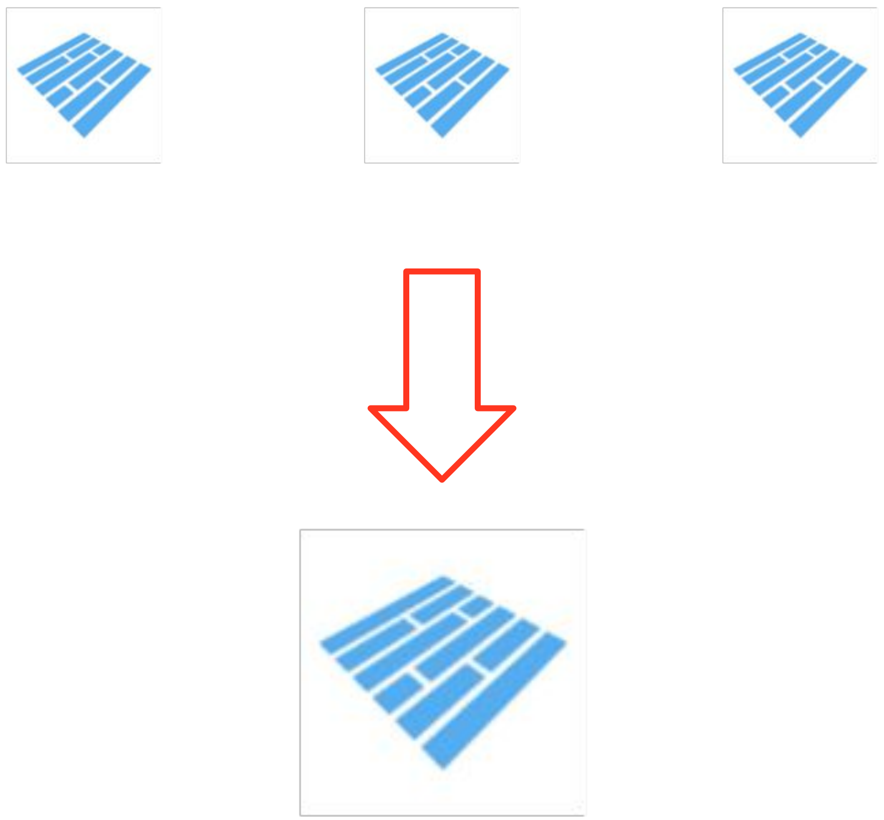
databricks

File size

- Files too small -> more IO, delta log processing slower
- Files too large -> cannot skip file, large UPDATE overhead
- 32MB - 128MB and 1GB for large tables > 10TB

Compaction

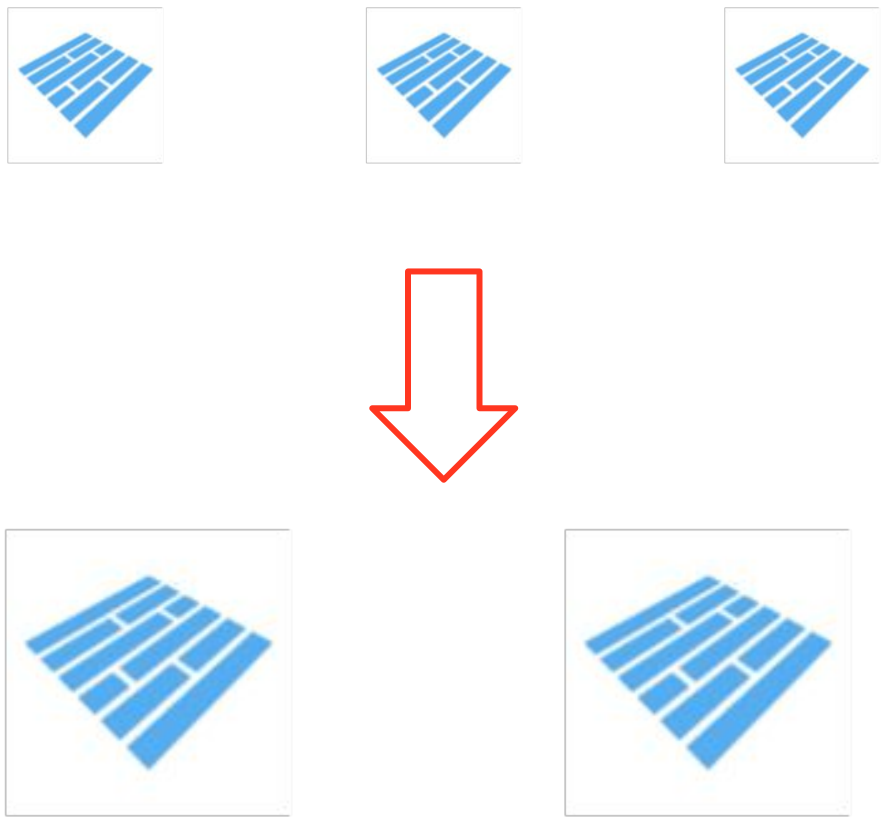
```python
path = "..."
numFiles = 16

(spark.read
 .format("delta")
 .load(path)
 .repartition(numFiles)
 .write
 .option("dataChange", "false")
 .format("delta")
 .mode("overwrite")
 .save(path))
```



Compaction

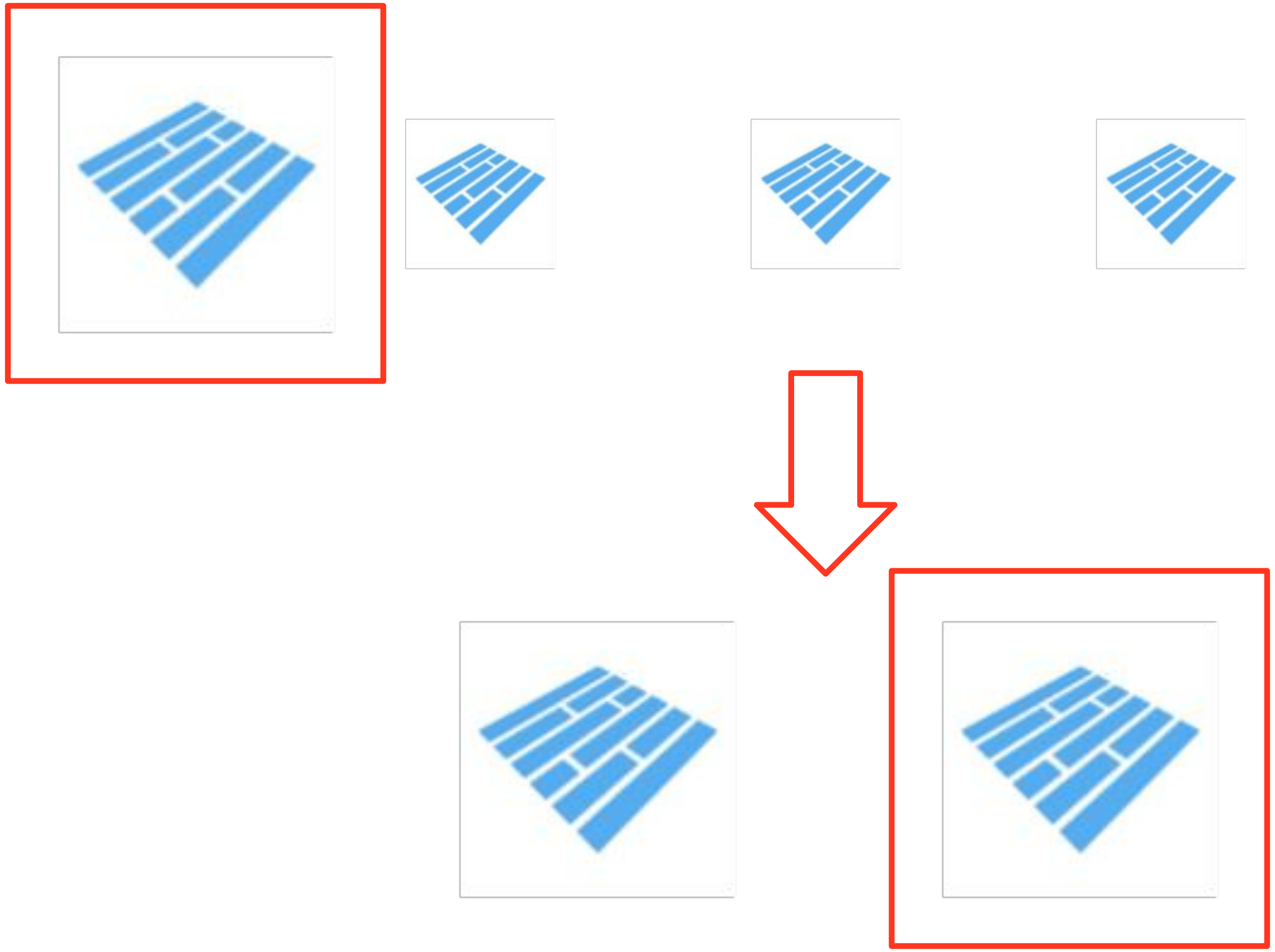
```python
path = "..."
numFiles = 16

(spark.read
 .format("delta")
 .load(path)
 .repartition(numFiles)
 .write
 .option("dataChange", "false")
 .format("delta")
 .mode("overwrite")
 .save(path))
```

- Number of target files?



Compaction

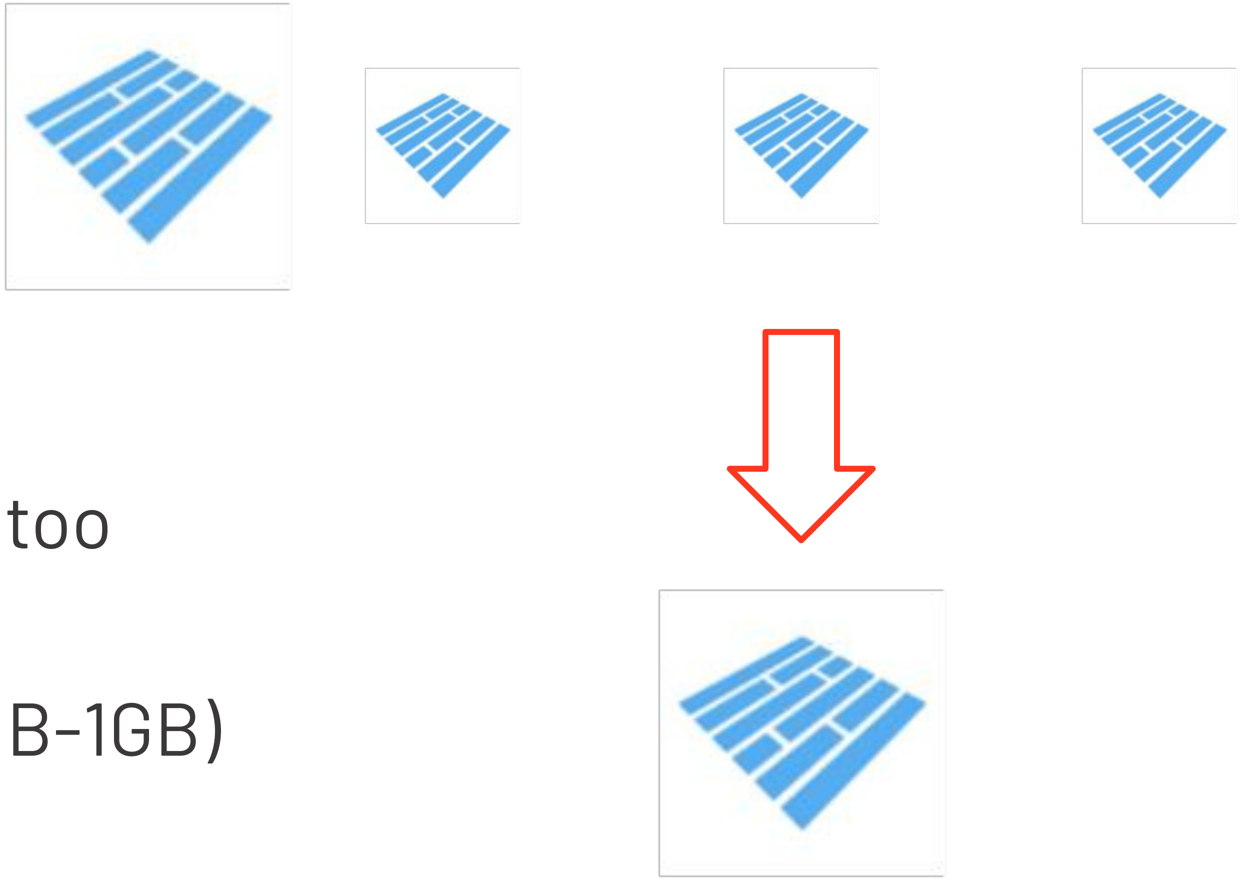
```
path = "..."
numFiles = 16

(spark.read
 .format("delta")
 .load(path)
 .repartition(numFiles)
 .write
 .option("dataChange", "false")
 .format("delta")
 .mode("overwrite")
 .save(path))
```



- Number of files?
- Overwrites the whole table(partition)

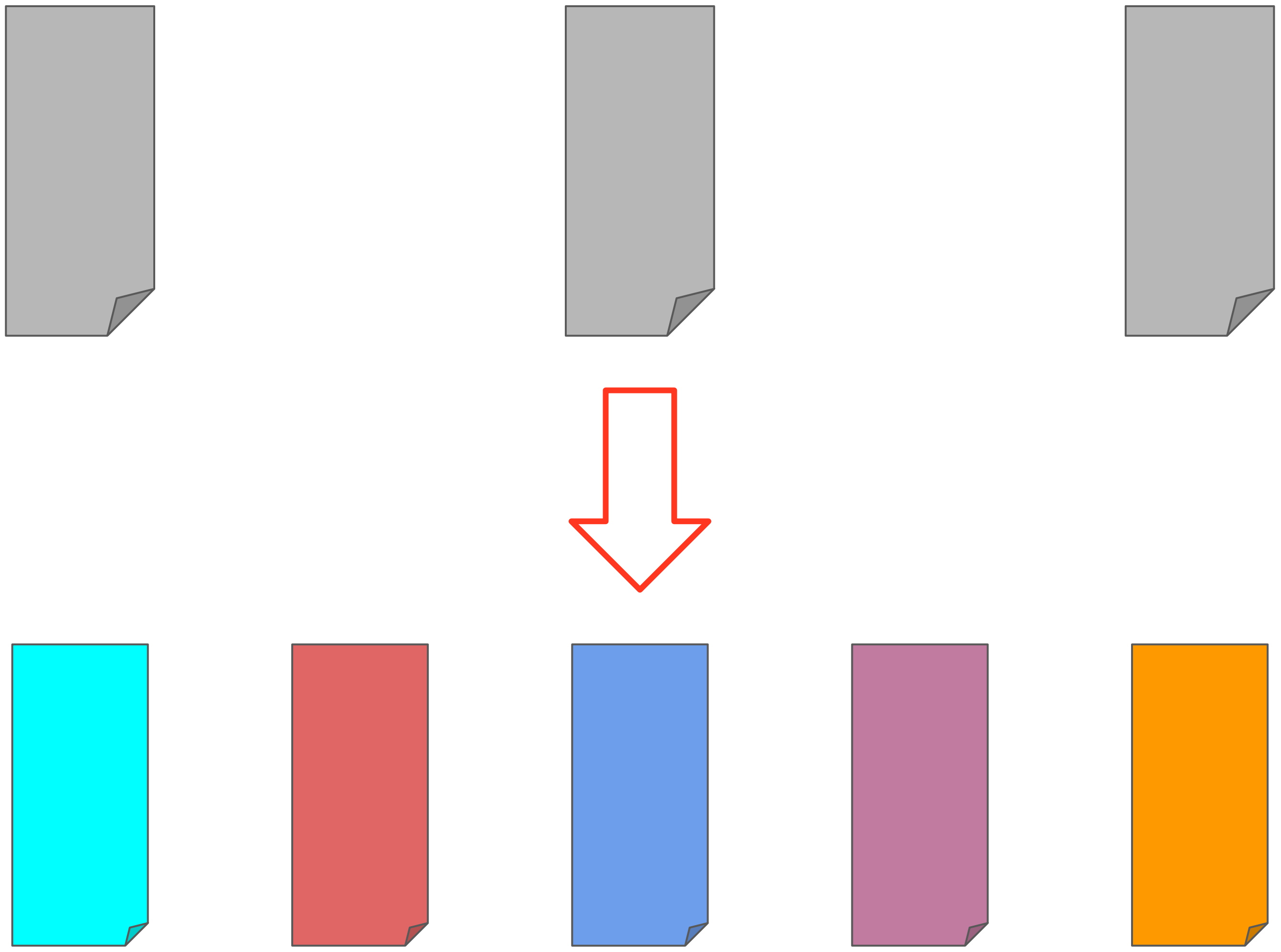
Optimize command

```
OPTIMIZE my_table
```



- Compacts only files that are too small
- Creates “good” file sizes(64MB-1GB)

Data clustering



databricks

Sorting - 1 dimension

X	Y
1	7
1	3
3	4
4	5

000.parquet

x>=1 && x<=4

X	Y
5	1
6	2
6	3
7	5

001.parquet

x>=5 && x<=7

```
SELECT * FROM table WHERE x = 5
```

Sorting - 2 dimension

X	Y
1	7
1	3
1	4
1	5

000.parquet

X	Y
1	1
1	2
6	3
7	5

001.parquet

```
SELECT * FROM table WHERE x = 1 AND y = 3
```

Sorting - 2 dimension

X	Y
1	1
1	2
1	3
1	3

000.parquet

X	Y
1	4
1	5
6	5
7	7

001.parquet

```
SELECT * FROM table WHERE x = 1 AND y = 3
```

databricks

Sorting - 2 dimension

X	Y
1	1
2	2
2	3
3	5

000.parquet

X	Y
4	4
5	3
6	5
7	7

001.parquet

```
SELECT * FROM table WHERE y = 3
SELECT * FROM table WHERE x = 1 OR y = 3
```

databricks

Z-order - Multi dimensional clustering

```
OPTIMIZE store_sales ZORDER BY x, y
```

X	Y
1	1
2	2
3	3
5	3

000.parquet

X	Y
3	4
5	4
6	5
7	7

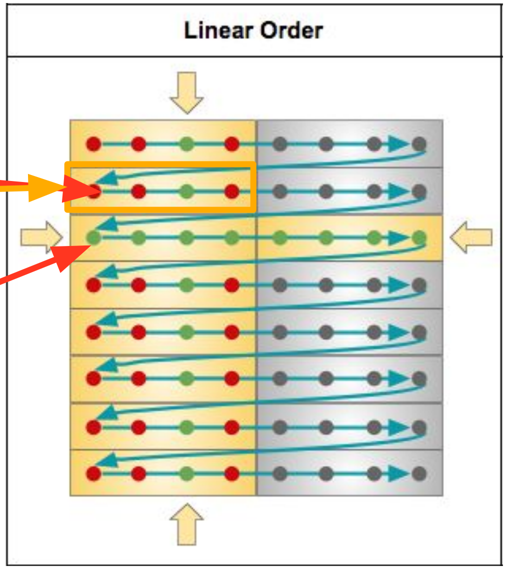
001.parquet

```
SELECT * FROM table WHERE y = 3
SELECT * FROM table WHERE x = 1 OR y = 3
```

Z-order - Multi dimensional clustering

X	Y
1	1
1	2
1	3
1	4

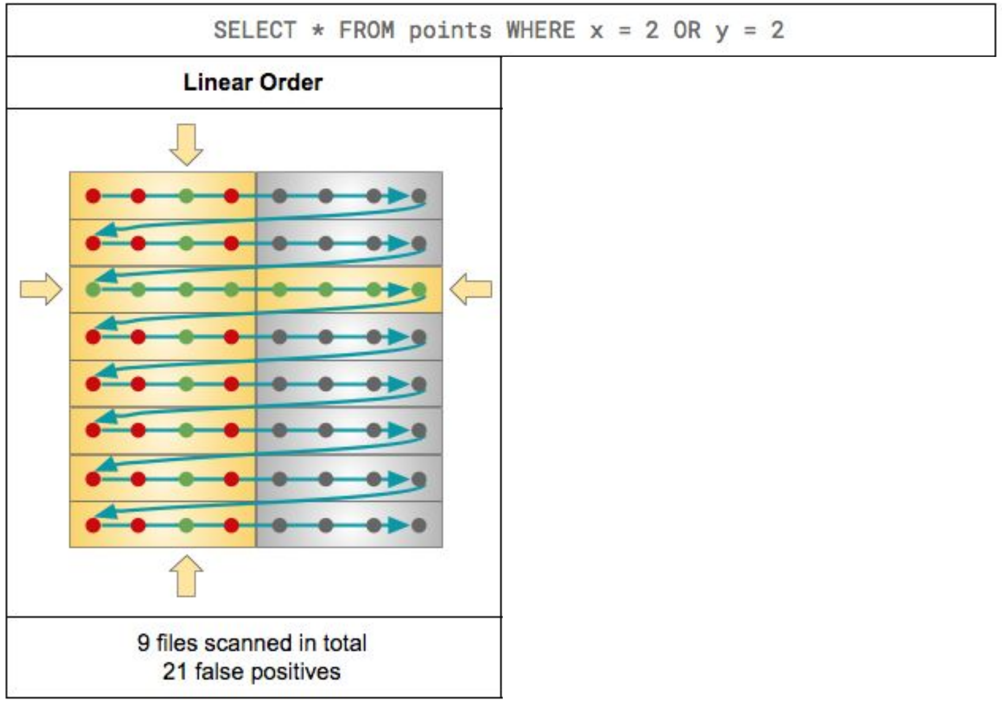
X	Y
2	1
2	2
2	3
2	4



Z-order - Multi dimensional clustering

X	Y
1	1
1	2
1	3
1	4

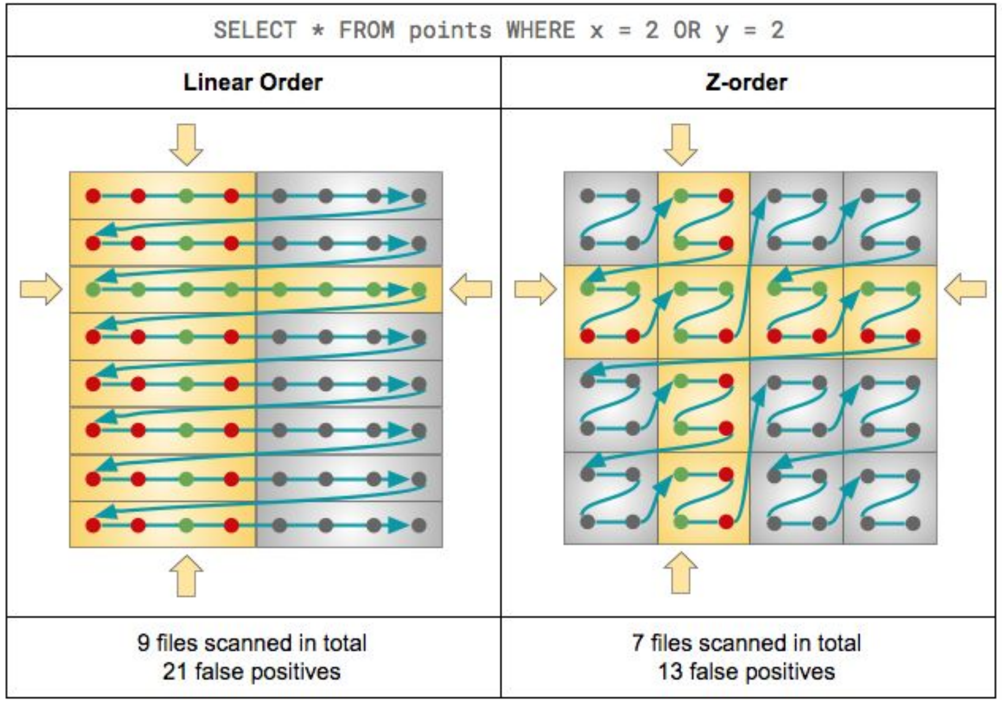
X	Y
2	1
2	2
2	3
2	4



Z-order - Multi dimensional clustering

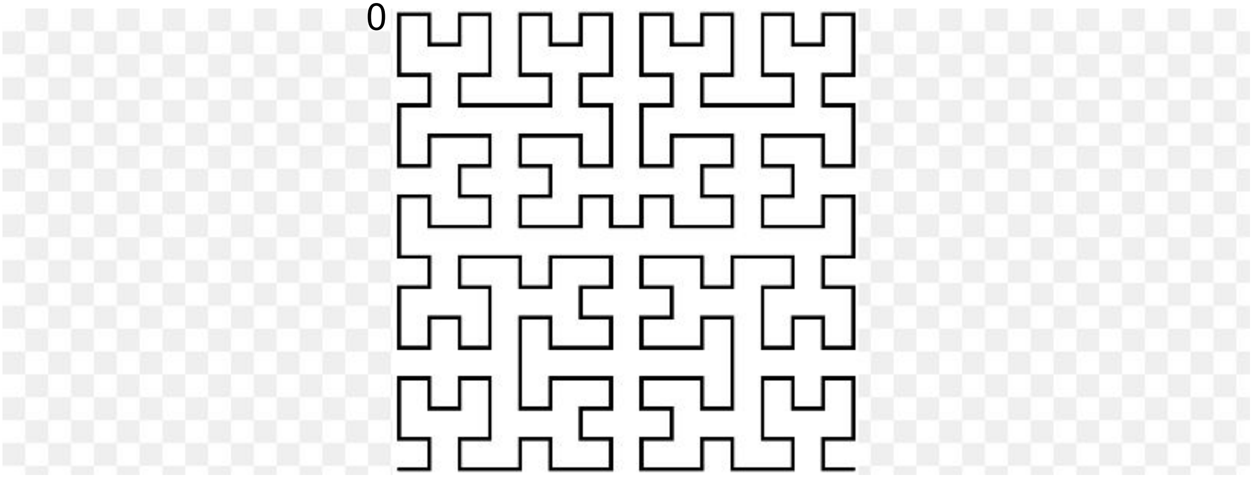
X	Y
1	1
1	2
2	1
2	2

X	Y
1	1
1	2
2	3
2	4



databricks

Hilbert curve



Auxiliary index - Bloom Filter

```
CREATE BLOOMFILTER INDEX
ON TABLE my_table
FOR COLUMNS(cust_id OPTIONS (fpp=0.1, numItems=50000000))
```

Is this value in this file?

- ❑ Maybe
- ❑ No

Bloom Filter

0 0 0 0 0

databricks

Bloom Filter

1

0	0	0	0	0
0	0	0	1	0

Bloom Filter

	0	0	0	0	0
1	0	0	0	1	0
3	0	1	0	1	0

Bloom Filter

1
3
7

0	0	0	0	0
0	0	0	1	0
0	1	0	1	0
0	1	1	1	0

Bloom Filter

	0	0	0	0	0
1	0	0	0	1	0
3	0	1	0	1	0
7	0	1	1	1	0

- Is 1 in the bitmap -> MAYBE
- Is 0 in the bitmap -> NO
- Is 2 in the bitmap -> MAYBE - False positive

Data layouts are hard

- Compaction jobs need to be scheduled
- Many knobs to turn
- Workloads change
- Data changes - skew

Steps to automation at Databricks

databricks

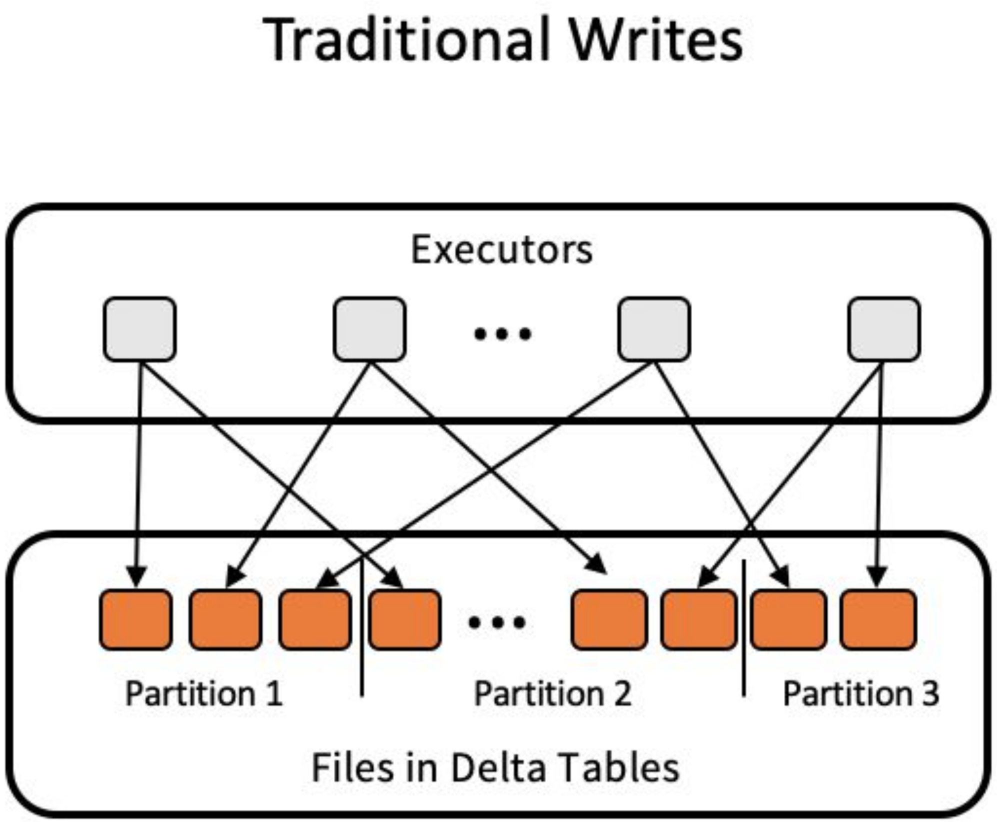
Auto-compaction

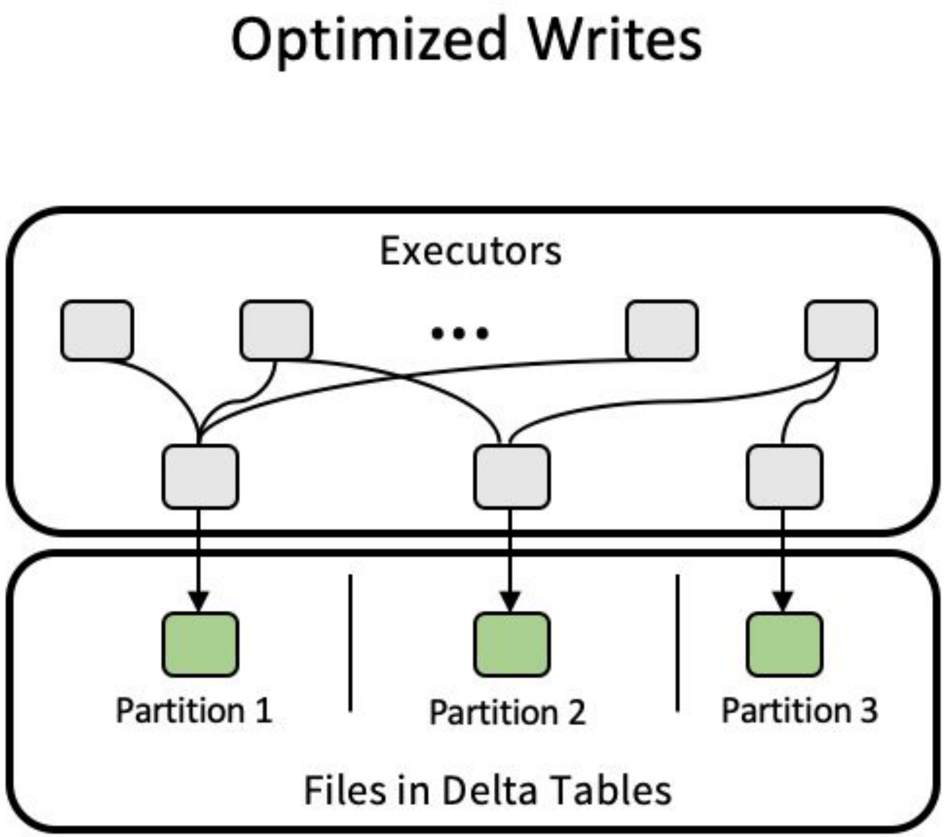
- Compact files after every commit automatically
- Should not run for too long
- Could conflict with other transactions

Optimized Writes

- Shuffled writes for partitioned tables
- Keeps files well sized

Optimized Writes

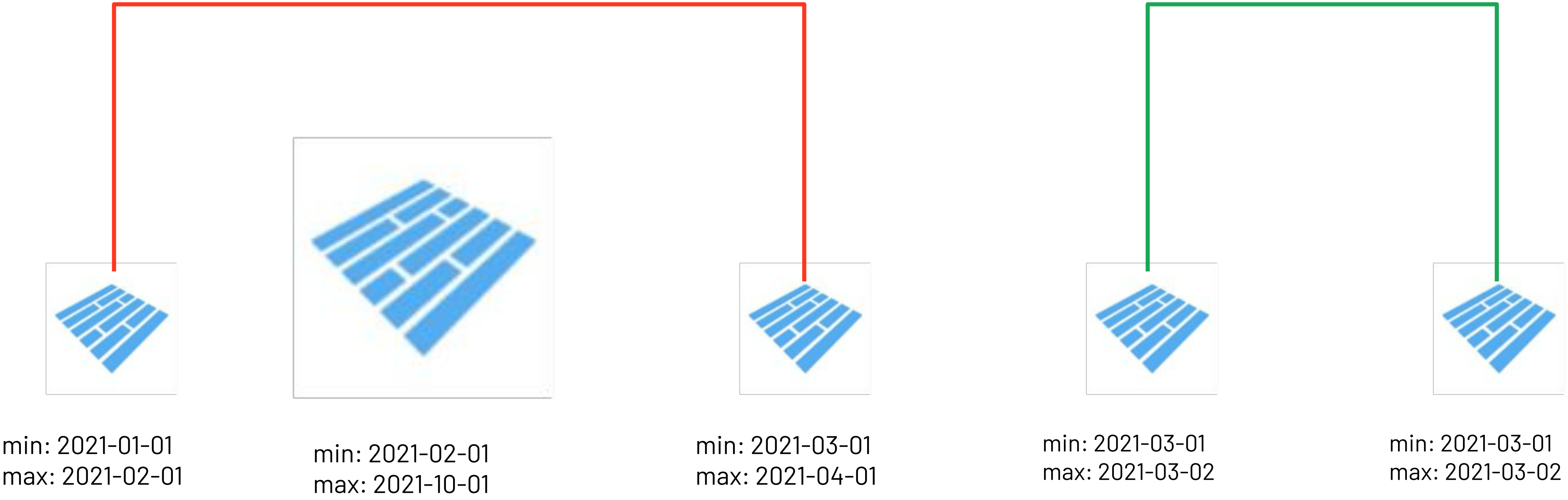




Insertion order

- Preserve the original data clustering
- Good for time series data, event logs

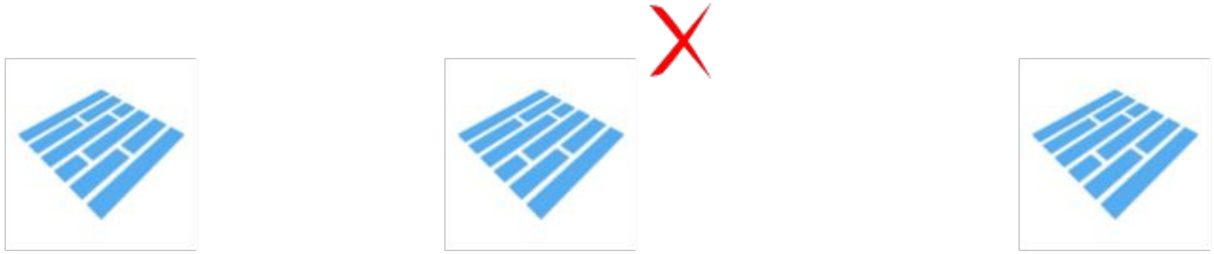
Compaction challenge



- Compact only neighboring files in timeline

Update/Delete challenge

```
DELETE FROM table WHERE date = '2021-02-01'
```

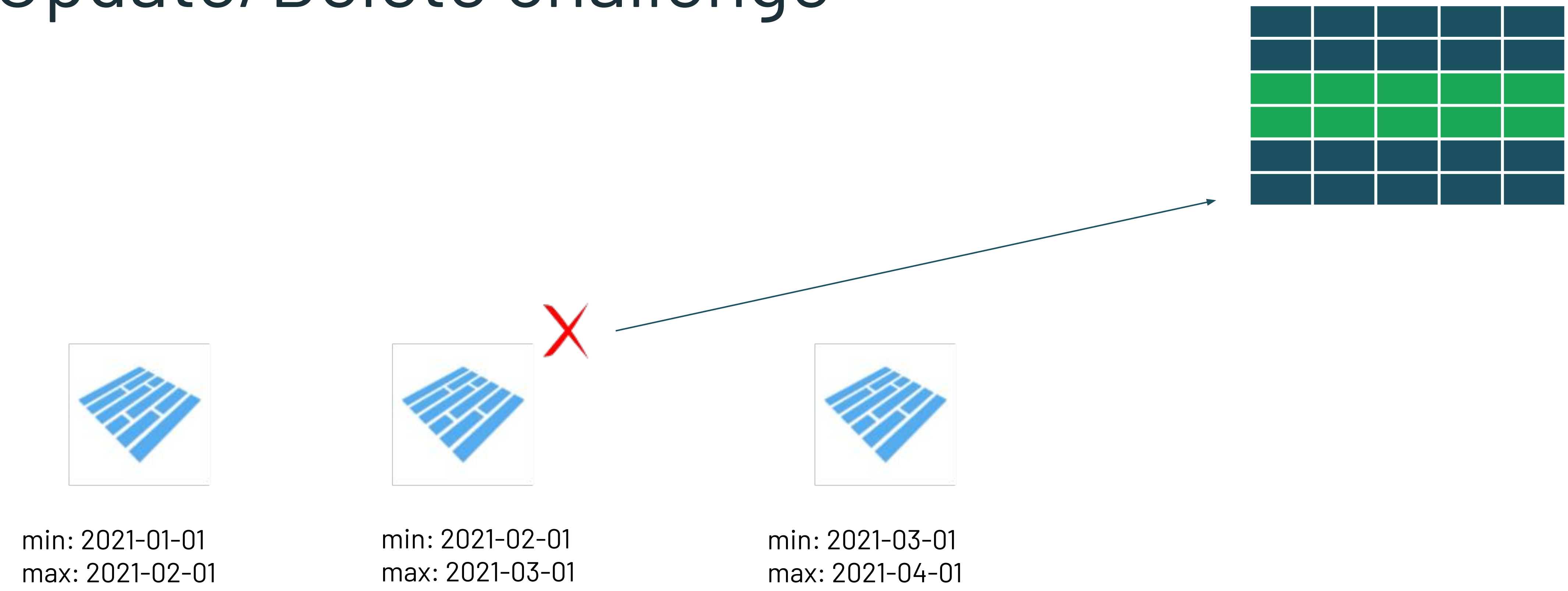


min: 2021-01-01
max: 2021-02-01

min: 2021-02-01
max: 2021-03-01

min: 2021-03-01
max: 2021-04-01

Update/Delete challenge



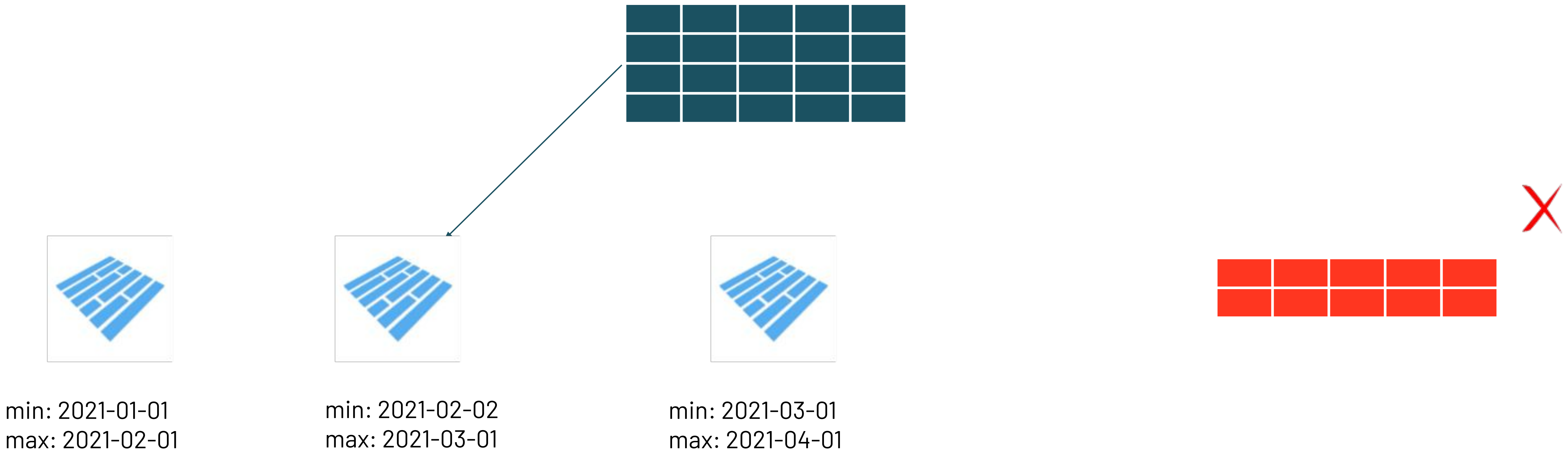
databricks

Update/Delete challenge

min: 2021-01-01
max: 2021-02-01

min: 2021-03-01
max: 2021-04-01

Update/Delete challenge

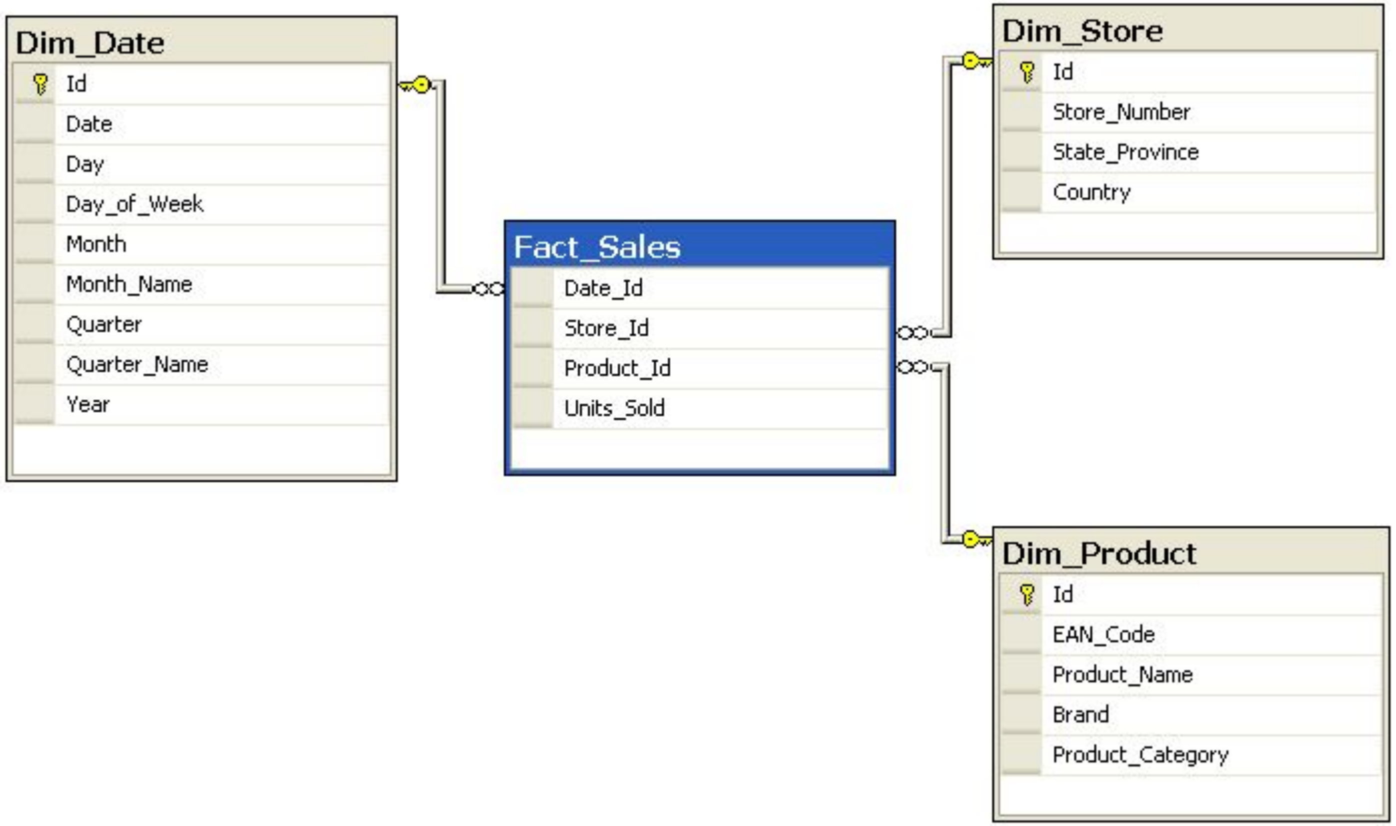


Future: Advisor/background optimizations

- Recognize common query patterns
- Propose optimal data layout
- Run in the background on idle machines

Star schema pattern

Star schema



https://en.wikipedia.org/wiki/Star_schema

Star schema

- Consider denormalizing, precomputed joins, storage is cheap
- Use DELTA for your fact and dimension tables
- OS Delta: partition the fact by date and compact files from time to time
- Z-ORDER fact table by "foreign" keys columns
- Z-ORDER dimension tables by "primary" key columns
- Consider creating bloom filter index for pinpoint predicates

Thanks!

We're hiring for internships and full time engineers in Amsterdam, Berlin, San Francisco, Seattle and Toronto!

databricks.com/company/careers

Sabir Akhadov
linkedin.com/in/akhadov