

Optimizing Data File Layout

Learning Objectives

- ▶ The concept of data file layout
- ▶ Exploring optimization techniques.

Data File Layout

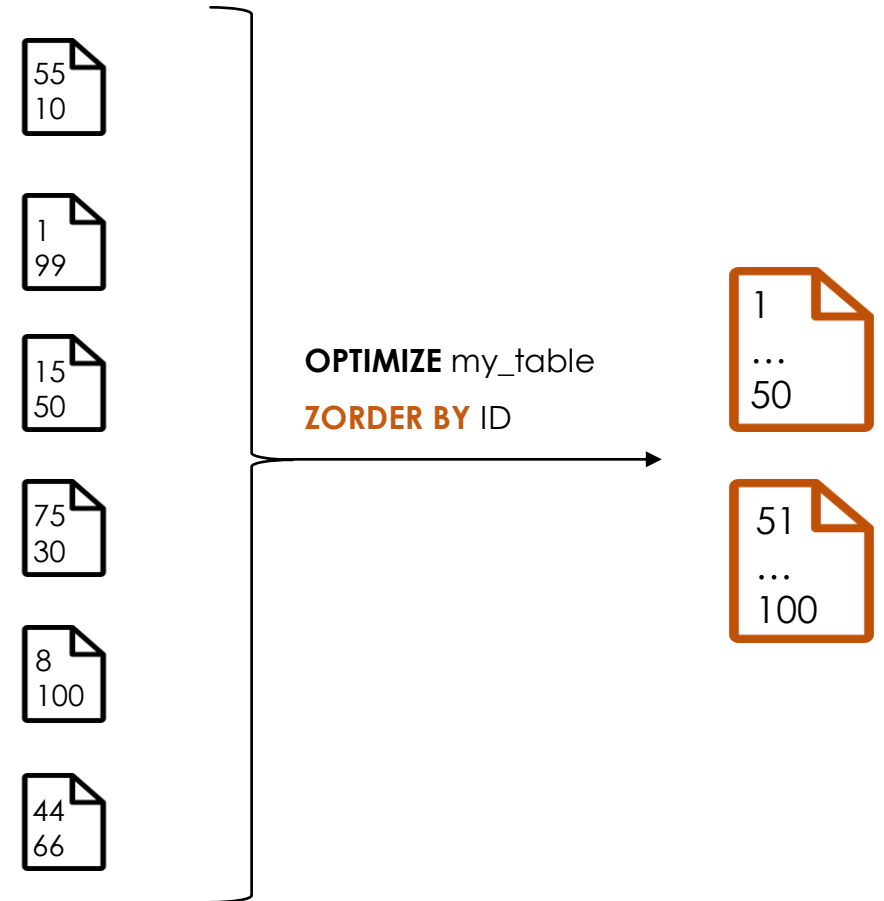
- ▶ The organization and storage structure of the underlying data files that make up a Delta table.
- ▶ Optimizing layout helps leveraging data-skipping algorithms
- ▶ Optimization techniques:
 - ▶ Partitioning
 - ▶ Z-Order Indexing
 - ▶ Liquid Clustering

Partitioning Limitations

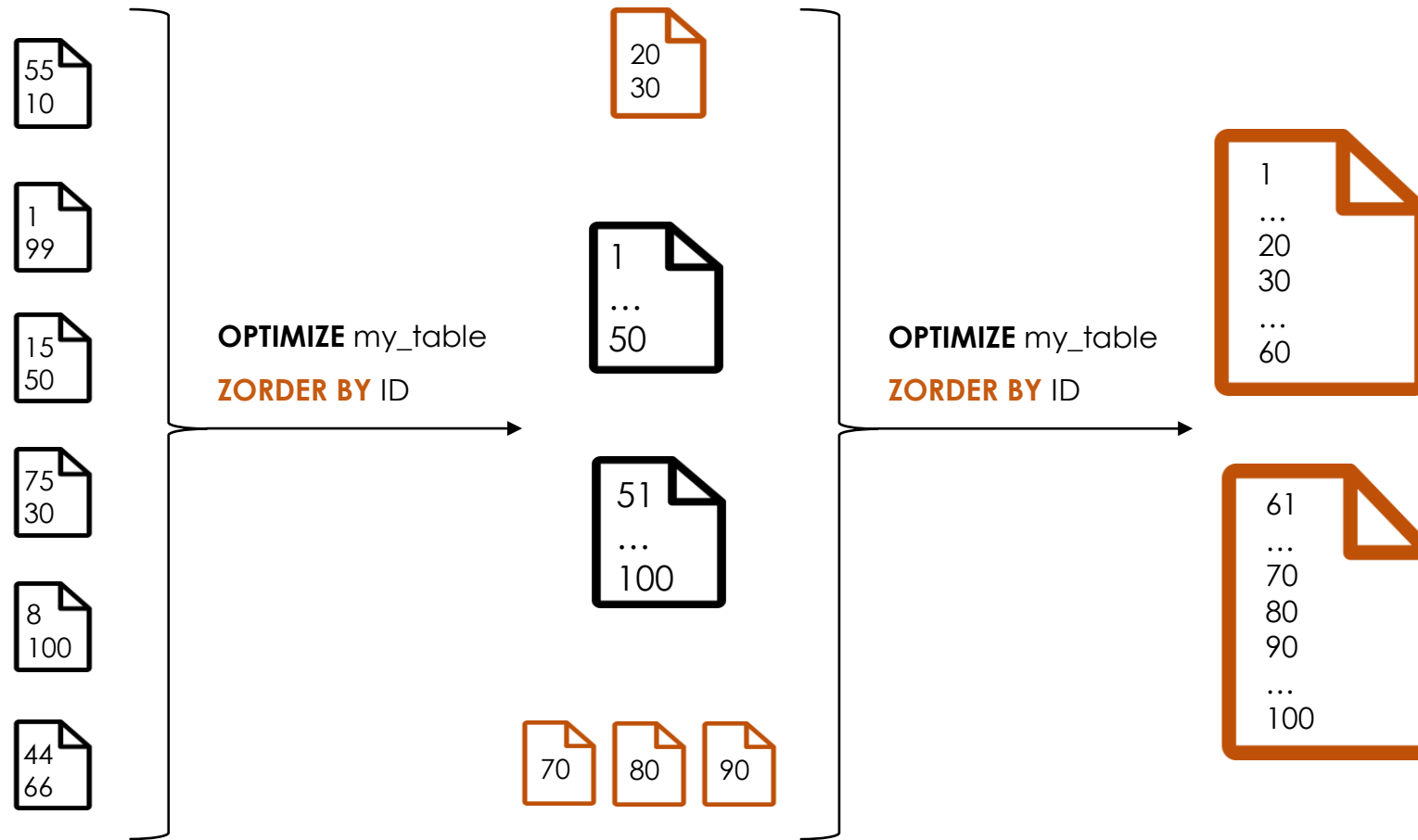
- ▶ Prevents file compaction across partition boundaries
 - ▶ Results in a small files problem
- ▶ Inefficient for high-cardinality columns
 - ▶ Results in a small files problem
- ▶ Static: Re-partitioning requires a full table rewrite

Z-Order Indexing

- ▶ Group similar data into optimized files without creating directories
 - ▶ `OPTIMIZE my_table`
ZORDER BY column_name
- ▶ Leverage data-skipping algorithms
- ▶ Effective for High-cardinality columns



Z-Ordering: Not Incremental



Liquid Clustering

- ▶ Improved version of Z-order indexing with more flexibility and better performance.

- ▶ Table-level definition

- ▶ New tables:

```
CREATE TABLE table1(col1, INT, col2 STRING, col3 DATE)  
CLUSTER BY (col1, col3)
```

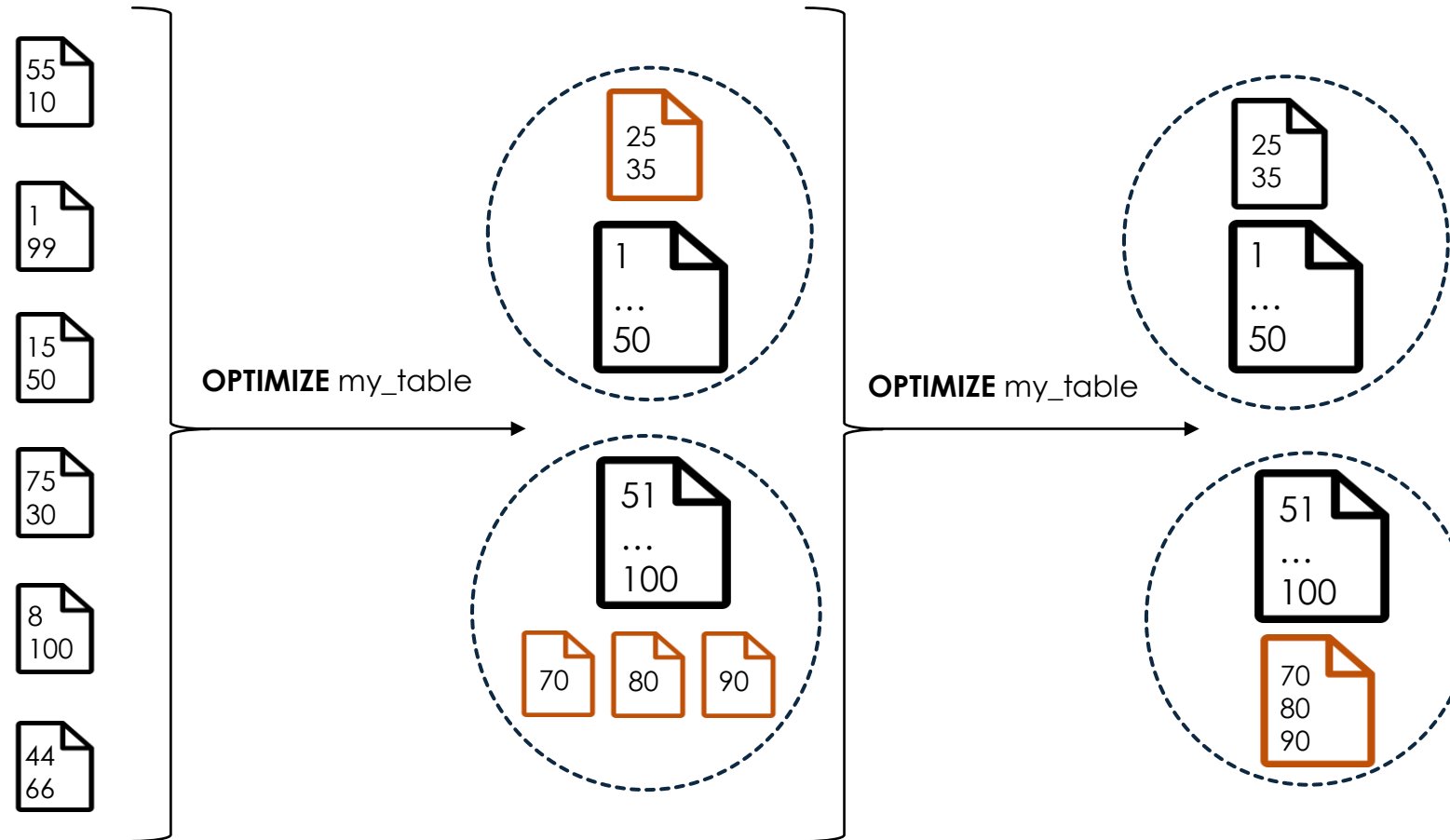
- ▶ Existing tables:

```
ALTER TABLE table2  
CLUSTER BY (<clustering_columns>)
```

- ▶ Clustering is not compatible with partitioning or ZORDER

Incremental Clustering

```
ALTER TABLE my_table  
CLUSTER BY ID
```



Choosing Clustering Keys

- ▶ Flexible to redefine clustering keys without rewriting existing data
- ▶ Choose clustering keys based on your query pattern

Automatic Liquid Clustering

- ▶ Databricks automatically chooses clustering keys by analyzing the table historical query workload.
- ▶ Requires Predictive Optimization on Unity Catalog managed tables

- ▶ Syntax

- ▶ New tables:

```
CREATE TABLE table1(col1, INT, col2 STRING, col3 DATE)  
CLUSTER BY AUTO
```

- ▶ Existing tables:

```
ALTER TABLE table2  
CLUSTER BY AUTO
```