-: Predicate Pushdown :-

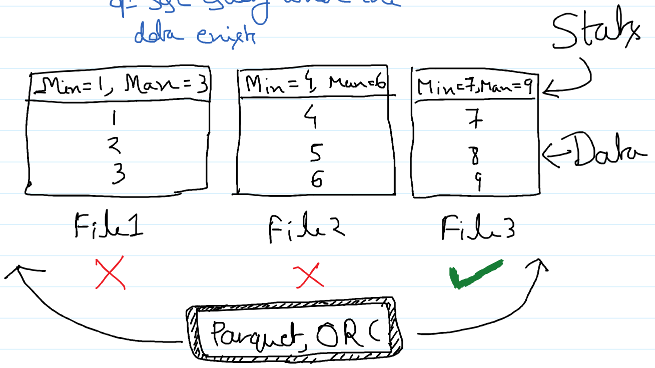
SQL Query

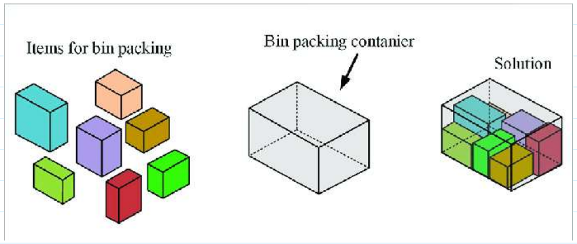
Push down specific section of SQL query where the data exists

select * from table1 where col1 = 9

① Huge Data Transfer via network

② Long Loading Time in memory





-: Z-Order Optimization :-

File1 → id= 5, 8, 9, 20 (min=5, max=20)

File2 → id = 10, 15, 30, 70 (min=10, max=70)

File3 → id= 2, 6, 30 45 (min=2 max=45)

File4 → id =10, 12, 18, 19 (min=10 max=19)

—optimize→
(Bin-packing)

File1+File2

id = 5, 8, 9, 20
10, 15, 30, 70

(min = 5, max = 70)

File3 + File4

id = 2, 6, 30, 45
10, 12, 18, 19

★ (min= 2 max=45)

select * from table where id = 12

File1 → id = 5, 8, 9, 20 (min = 5, max = 20)

File2 → id = 10, 15, 30, 70 (min = 10, max = 70)

File3 → id = 2, 6, 30, 45 (min = 2, max = 45))

File4 → id = 10, 12, 18, 19 (min = 10, max = 19))

—— optimize (z-ordering) →

id = 2, 5, 6, 8, 9, 10, 10, 12 (min = 2, max = 12)

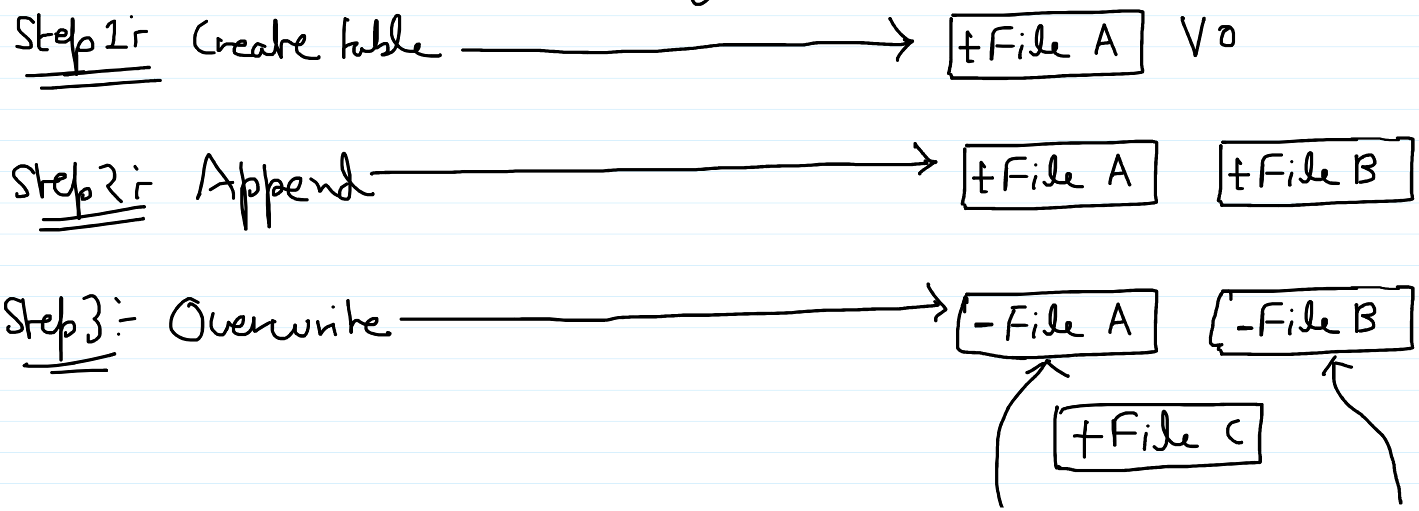
id = 15, 18, 19, 20, 30, 30, 45, 7

(min = 15, max = 70)

select * from table where id = 12

So overall, z-ordering helps your queries run faster because it makes it more likely that data can be skipped.

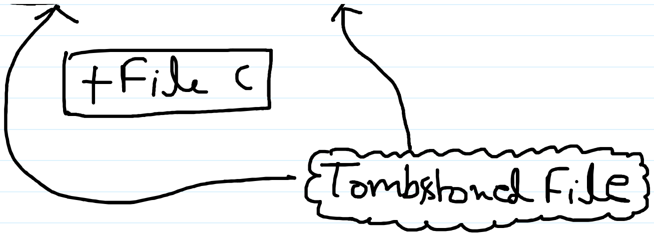
:- Vacuuming :-



Step1:- Create table

Step2:- Append

Step3:- Overwrite



Deleting log files

```
my_table/
├── _delta_log
│   ├── 0.json
│   ├── 1.json
│   │   ⋮
│   ├── 10.json
│   └── 10.checkpoint.parquet
├── file1.parquet ✗
├── file2.parquet ✓
└── file3.parquet ✓
```

delta.logRetentionDuration (interval 30 days)

↳ Each time a checkpoint is written Databricks automatically cleans up log entries older than logRetentionDuration.

(automatic)

vaccum - [delta.deletedFileRetentionDuration]

(default 1 week)

(not automatic)

{delta.logRetentionDuration
delta.deletedFileRetentionDuration} => Both impact time-travel

Q→ If I need to perform time-travel upto 200 days ?