Improving File Skipping for TPC-H Q10 on Delta Lake

Vegim Bytyqi

November 1, 2025

Motivation

Why File Skipping?

- **File Skipping** allows query engines to avoid reading irrelevant files via metadata (min/max statistics, partitions).
- Without partitioning, Delta Lake scans all files even for narrow date filters.
- **Goal:** Optimize TPC-H Query 10 performance by improving Delta table layout and metadata utilization.

TPC-H Query 10

Query Definition

```
SELECT
  c.c_custkey, c.c_name,
  SUM(l.l_extendedprice * (1 - l.l_discount)) AS revenue,
  n.n_name
FROM
  customer c, orders o, lineitem l, nation n
WHERE
  c.c_custkey = o.o_custkey
  AND l.l_orderkey = o.o_orderkey
  AND c.c_nationkey = n.n_nationkey
  AND o.o_orderdate >= DATE '1993-10-01'
  AND o.o_orderdate < DATE '1994-01-01'
  AND l.l_returnflag = 'R'
GROUP BY c.c_custkey, c.c_name, n.n_name
ORDER BY revenue DESC;
```

Delta Writer Script

Partitioning for Efficient Skipping

```
lineitem['order_year']  = lineitem['o_orderdate'].dt.year.astype('int32')
lineitem['order_month'] = lineitem['o_orderdate'].dt.month.astype('int32')

lineitem = lineitem.sort_values(
  ['l_returnflag', 'order_year', 'order_month', 'l_orderkey']
).reset_index(drop=True)

write_deltalake(
  'lineitem-delta-part',
  lineitem,
  mode='overwrite',
  partition_by=['l_returnflag', 'order_year', 'order_month']
)
```

Effect: Locality of data by `l_returnflag` and month enables pruning of irrelevant partitions.

Reader Script & Profiling

Measuring File Skipping

```
utils.measure_skipping(f"""
  SELECT count(*)
  FROM delta_scan('{args.delta_path}')
  WHERE l_shipdate BETWEEN
        DATE '1993-01-01' AND DATE '1993-02-01'
""")
```

Enhancements:

- Joins across `customer`, `orders`, `lineitem`, and `nation`.
- Filters on `o_orderdate` and `l_returnflag`.
- Uses `delta_scan()` to leverage partition metadata for file skipping.

Optimization Techniques

Partitioning

`partition_by=['l_returnflag', 'order_year', 'order_month']` enables partition pruning based on query predicates.

Sorting

Sorting by flag and date enhances min/max statistics for metadata pruning.

Profiling

Used `PRAGMA enable_profiling='json'` in DuckDB to extract file I/O metrics and latency.

Experimental Results

Observed Metrics

- Files read: **42**
- Total files: **128**
- Files skipped: **86**
- Query latency: **0.1037 s**

Interpretation: 67% of files were skipped → major I/O reduction and faster query time.

Conclusion

Key Takeaways

- File skipping performance depends on **table layout** and **metadata granularity**.
- Delta partitioning and sorting yield substantial scan-time savings.
- DuckDB profiling provides lightweight and transparent insights.

Questions?