Materialization Strategies

Definition

Materialization strategies determine **when and how tuples are reconstructed** in a **column-oriented database.** Since **column stores** store data **column-by-column,** materialization affects query performance and **memory usage**

Types of Materialization Strategies (PPD 6–61 to 6–63)

Strategy	Description	PPD Reference
Early Materialization (EM)	Tuples are **reconstructed early**, before any query operations are performed.	**PPD 6–61**
Late Materialization (LM)	Tuples are **reconstructed as late as possible**, allowing operations on compressed columns before reconstruction.	**PPD 6–61**
Hybrid Materialization	Combines **early and late materialization**, dynamically deciding which approach is optimal.	**PPD 6–63**

Feature	Early Materialization (EM)	Late Materialization (LM)
Tuple Reconstruction	Happens **before query execution**.	Happens **at the end of query execution**.
Performance	**Faster for small queries** since tuples are ready for use.	**Faster for large queries** since operations on compressed columns are more efficient.
Memory Usage	**Higher**—more memory is needed for reconstructed tuples.	**Lower**—operating on compressed columns reduces memory footprint.
Aggregate Queries (SUM, AVG, COUNT)	**Less efficient**—entire tuples are reconstructed, even if only one column is needed.	**More efficient**—aggregation can be performed on compressed columns before reconstruction.

Join Performance	Faster for row-wise operations, but joins become expensive due to row materialization.	Efficient for column-wise joins, since joins happen before materialization.
Use Case	Row-based processing (OLTP-like workloads).	Column-based processing (OLAP workloads).

Conclusion

- **Early materialization** is good for **small queries** but **wastes memory**.
- **Late materialization** is better for **OLAP queries**, as it **saves memory and improves performance**.
- **Hybrid approaches** dynamically decide when to materialize tuples.