ANSI SQL:2016 EXECUTION ORDER & OPTIMIZATION MATRIX

Quick-Reference Desk Card · Owen Thackeray · Precision Technical Press

1. The 6-Stage Physical & Logical Query Execution Pipeline

Stage	SQL Clause	Engine Evaluation & Architectural Invariant
Stage 1	`FROM / JOIN`	Resolves base relations, evaluates Cartesian products, and filters join predicates (`ON`). Outer join row preservation happens here.
Stage 2	`WHERE`	Filters surviving tuples using three-valued predicate logic (`TRUE`, `FALSE`, `UNKNOWN`). `SELECT` aliases are not yet visible.
Stage 3	`GROUP BY`	Collapses filtered rows into distinct partition buckets. Every non-aggregated column in `SELECT` must appear here.
Stage 4	`HAVING`	Filters aggregated group partitions (`COUNT(*) > 5`, `SUM(net) >= 1000`). Never use `HAVING` for unaggregated row filters.
Stage 5	`SELECT / OVER()`	Projects output expressions, assigns column aliases, and evaluates analytical window functions (`ROW_NUMBER`, `SUM() OVER`).
Stage 6	`ORDER BY / LIMIT`	Sorts projected output tuples (`ASC/DESC`, `NULLS LAST`) and applies `LIMIT` / `FETCH FIRST` or keyset pagination.

2. SARGable Predicate Refactoring Patterns (B-Tree Index Seek vs Table Scan)

Anti-Pattern (Full Table Scan)	SARGable Refactor (B-Tree Seek)	Why It Matters
`WHERE YEAR(order_date) = 2026`	`WHERE order_date >= '2026-01-01' AND order_date < '2027-01-01'`	Wrapping an indexed column in a function disables B-Tree range traversal.
`WHERE amount * 1.08 > 500`	`WHERE amount > 500 / 1.08`	Isolate the indexed column on one side of the comparison operator.
`WHERE LOWER(email) = 'a@b.com'`	Create functional index `ON customers(LOWER(email))`	Matches expression signature directly against index leaf nodes.
`WHERE sku LIKE '%MECH%'`	`WHERE sku LIKE 'SKU-MECH%'`	Leading wildcards force full leaf scan; prefix searches seek directly.

3. Window Framing Mechanics (`ROWS` vs `RANGE`) & Isolation Levels

Construct	Deterministic Behavior
`ROWS BETWEEN ...`	Counts physical row offsets before/after current row. Deterministic even when `ORDER BY` contains duplicate ties.
`RANGE BETWEEN ...`	Groups peer rows sharing the same `ORDER BY` value into a single logical step. Default when `ORDER BY` is present without explicit frame.
`ROW_NUMBER()`	Assigns strictly sequential integers (`1, 2, 3, 4`) within partition; ties broken arbitrarily unless tie-breaker column added.
`RANK()` vs `DENSE_RANK()`	`RANK()` leaves gaps after ties (`1, 2, 2, 4`); `DENSE_RANK()` preserves contiguous rank numbers (`1, 2, 2, 3`).
`READ COMMITTED`	Prevents Dirty Reads (`G1`). Permits Non-Repeatable Reads (`P2`) and Phantom Reads (`A3`). Default in PostgreSQL.

`REPEATABLE READ` / `SERIALIZABLE`	`REPEATABLE READ` prevents `G1` & `P2`; `SERIALIZABLE` prevents `G1`, `P2`, Phantom Reads, and write-skew anomalies.