MASTER CLASS ON ADVANCED SQL

For SunBeam PG-Diploma Mar-2022 batch

DBDA
DAC
DMC

SPEAKER: MR. NILESH GHULE

Contents

- GROUP BY + Aggregate fns
- ORDER BY
- JOINS
- Sub-Queries

Grouping with ROLLUP

RDBMS (DWH) → OLAP (Analytical)
RDBMS (DWH) → OLTP (Transactional)

- Improved OLAP (Online Analytical Processing) in SQL:1999.
- Added sub-aggregation & grand-aggregation features into GROUP BY.
- Getting totals along with group results is feasible using UNION operator.
- ROLLUP operation provide grand aggregation along with group results.
- For grand-aggregation grouped column values is NULL.
- Grouping on multiple columns with ROLLUP provide sub-aggregation as well as grand-aggregation.
- CUBE() helps getting all combinations of sub-aggregations. However this is not supported in MySQL.
- The GROUPING() function returns 1 indicating sub/grand aggregation on that column

Derived tables

- Derived tbl is a virtual tbl returned from a sub-query in FROM clause of outer query.
- This is also referred as "Inline view".
- SQL-92 allows stand-alone sub-query to be used in FROM clause.
- The derived table must have an alias.
- Syntax: SELECT columns FROM (SELECT … FROM … …) AS derived_alias …;
- SELECT columns FROM (SELECT … FROM … …) AS derived_alias (columns) …;
- Advantages/applications of Derived tables
  - More readable (than joins)
  - Not reusable (compared to views)
  - Overcome limitations of GROUP BY
- Limitations of derived tables
  - Cannot refer columns of preceding tables in same FROM clause
  - Cannot be recursive.

Common Table Expression (CTE)

- CTE is a virtual table returned from a SELECT query.
- It can be used for CRUD operations, creating table or view, …

- CTE is of two types
  - Non-recursive CTE
  - Recursive CTE

- Applications of CTE
  - Readable & Understandable
  - Better organization of large queries
  - Non-reusable view
  - Overcome limitations of GROUP BY
  - Recursion for hierarchical data

Common Table Expression (CTE)

- WITH cte_name (columns) AS (SELECT …) SELECT columns FROM cte_name;

- INSERT INTO tbl (columns) WITH cte_name (columns) AS (SELECT …);
- WITH cte_name (columns) AS (SELECT …) UPDATE …;
- WITH cte_name (columns) AS (SELECT …) DELETE ….;

- WITH cte_name1 (columns) AS (SELECT …), cte_name2 (columns) AS (SELECT …) SELECT …;

- WITH can be used in a sub-query, derived table or declaring a cursor.

- CREATE TABLE … WITH cte_name (columns) AS (SELECT …) SELECT columns FROM cte_name;

- CREATE VIEW… WITH cte_name (columns) AS (SELECT …) SELECT columns FROM cte_name;

Window Functions

- Window functions are added in SQL-2003. Supported in MySQL 8.0+.
- Most powerful tool to solve typical analytical problems.
- Aggregate functions operate on group of rows and generates summary (fewer rows).
- Window functions also operate on group of rows, but not reduce number of rows.
- Unlike grouping, windowing retain the other columns.
- Windowing enable dividing data into multiple partitions, sorting each partition and perform window operations on each row.

Window Functions

- Window functions are of two types
  - Aggregate functions
    - Can be used with or without windowing.
    - SUM(), AVG(), MAX(), MIN(), COUNT(), STDEV(), ...
  - Non-aggregate functions → HELP Window Functions; HELP RANK;
    - Can be used with windowing only. → OVER()
    - ROW_NUMBER(), RANK(), DENSE_RANK(), FIRST_VALUE(), LAST_VALUE(), LEAD(), LAG(), ...

- SELECT window_function(…) OVER (window_specification), col1, col2 FROM table;

- window_function(…) OVER(window_specification)
  - PARTITION BY columns
  - ORDER BY columns ASC | DESC
  - ROWS | RANGE BETWEEN frame_start AND frame_end

() → window including all rows

(PARTITION BY deptno) → window includes rows per dept (like group).

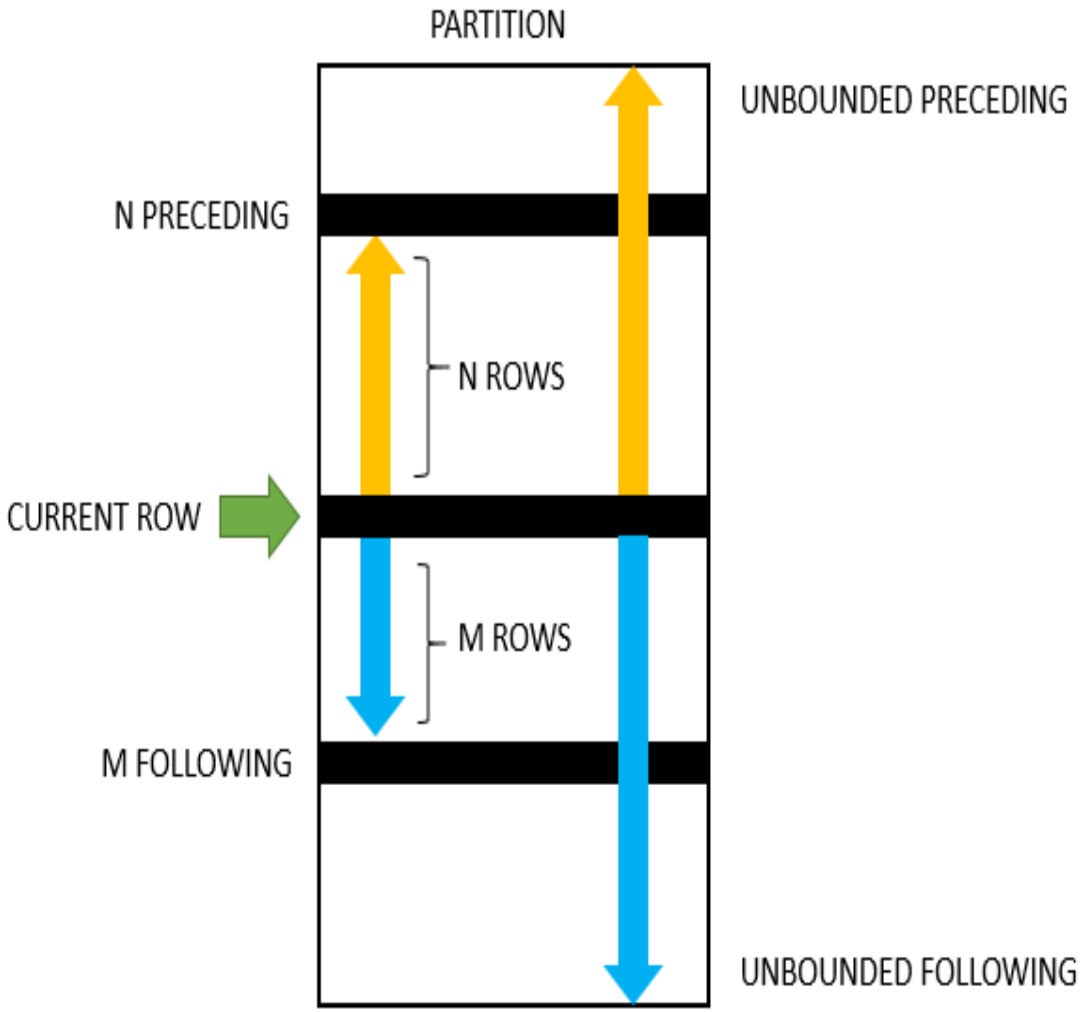
Window Functions

- Partition
  - Breaks up rows into partitions/groups.
  - Rows can be partitioned by one or more columns/expressions.
  - Window function is performed within partitions and re-initialized when crossing partition boundary.
- Order
  - Sorts rows within a partition in ASC or DESC order.
  - Rows can be sorted on one or more columns/expressions.
  - Partition and Order is supported by all window functions.
  - Order by is useful only for order-sensitive functions e.g. ROW_NUMBER(), RANK(), etc.
- Frame
  - Frame is a subset of current partition.
  - Frame is defined w.r.t. current row and is moving within a partition depending on position of the current row.
  - The default frame is RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW.

Window Functions (Non-aggregate)

Name	Description
ROW_NUMBER	Assigns a sequential integer to every row within its partition
RANK	Assigns a rank to every row within its partition based on the ORDER BY clause. It assigns the same rank to the rows with equal values.
DENSE_RANK	Same as RANK(), but if two or more rows have the same rank, then there will be no gaps in the sequence of ranked values.
PERCENT_RANK	Calculates the percentile rank of a row in a partition. (RANK() – 1) / (NumberOfRows – 1)
FIRST_VALUE	Returns the value of the specified expression with respect to the first row in the window frame.
LAST_VALUE	Returns the value of the specified expression with respect to the last row in the window frame.
LAG	Returns the value of the Nth row before the current row in a partition. It returns NULL if no preceding row exists.
LEAD	Returns the value of the Nth row after the current row in a partition. It returns NULL if no subsequent row exists.
NTH_VALUE	Returns value of argument from Nth row of the window frame
NTILE	Distributes the rows for each window partition into a specified number of ranked groups.
CUME_DIST	Calculates the cumulative distribution of a value in a set of values. Same as COUNT(*) OVER (ORDER BY Col1) / COUNT(*) OVER ().

Window Frame



- ROWS/RANGE BETWEEN *start* AND *end*.
- *start* is one of:
  - UNBOUNDED PRECEDING: The window starts in the first row of the partition
  - CURRENT ROW: The window starts in the current row
  - N PRECEDING or M FOLLOWING
- *end* is one of:
  - UNBOUNDED FOLLOWING: The window ends in the last row of the partition
  - CURRENT ROW: The window ends in the current row
  - N PRECEDING or M FOLLOWING
- ROWS count number of rows, while RANGE follows number of unique values in given column.

THANK YOU!

NILESH GHULE <nilesh@sunbeaminfo.com>