Analytical Functions

Agenda

- Analytical Functions
  - Rank()
  - Dense_Rank()
  - Row_number()
  - LAG() & LEAD() Function
  - First_Value()
  - Last_Value()
  - NTILE()
  - CUME_DIST()
- Recursive Query Expression

Analytical Functions

Analytical Functions

- Many RDBMS provides Analytical functions to perform complex operations and evaluate the results efficiently.
- Analytical functions reduces the use of JOINS as these perform many self join operations on same database table.
- Analytical functions are otherwise called as *windowing* or Online Analytical Processing (OLAP) functions.

Advanced Aggregate Functions

- The Window/ Analytical function uses OVER() clause to calculate aggregate results on group of rows based on candidate key.
- The aggregate result thus produced from group of rows is again shared for each row in the group.
- This is an advanced feature of GROUP BY clause by sharing aggregate result at row level.

Advanced Aggregate Functions - Benefits

- Reporting shows the comparison between current record entry with aggregate result.
- Build statistics on the cumulative results rather than aggregate results.
- More granular level of cost controlling in Financial organization whereas strategic reports are usually generated by GROUP by clauses as they show overall financial performance.

Window Functions - Ranking

- The ranking functions assign a rank for each row in an ordered group of rows.
- Rows are ranked sequentially.
- For each partition, the rows are ranked starting with 1.
- There are 3 types of ranking functions supported in MySQL-
  - rank()
  - dense_rank()
  - percent_rank()

Rank() Function

Displays the rank for each record based on highest value of a desired column by calculating on group of rows divided by corresponding candidate key.

CRITICAL NOTE

Rank() function will keep skipping the subsequent ranks based on the count of similar column values.

No. of Ranks skipped = No. of gaps between similar column values.

Dense_Rank() Function

- Dense_rank displays the rank based on highest value of a desired column, but it preserves the rank for next following record without skipping.

Row_number() Function

Row_number displays the unique ID to identify the rows. The values in each row is not considered.

LAG() and LEAD() Function

In a normal select query , all records are interpreted serially; Sometimes there is a need to look back and forth while you are retrieving the current record in SELECT query.

FIRST_VALUE() using order by

FIRST_VALUE () function analyses the results of analytical expression which is defined as OVER(),

and then returns the first value from the ordered set of rows.

LAST_VALUE() using Range of values in a row order

LAST_VALUE () function analyzes the results of analytical expression which is defined as OVER(),

and then returns the last value from an ordered set of rows.

LAST_VALUE() using Range of values in a row order

In this example,

- Initially, all of the table rows are ordered by using Balance column values.
- Secondly, RANGE between unbounded PRECEDING and FOLLOWING is used to define the range of values that are returned in an ordered set of rows.
- Finally, the last_Value () choses the last value from the range in an order set of rows and then assigns the last value across each record in the total output.

NTILE() categorize the records into buckets

NTILE () function analyses the results of analytical expression which is defined as OVER().

NTILE () splits the total records into predefined number of buckets.

CUME_DIST() - Cumulative distribution

Distribution of records means - the percentage of a record occupied in the total record set.
Cumulative distribution means , the cumulative percentage of records from first to current row is calculated

out of total result.

Aggregate Functions with Window Functions

- Similarly other grouping functions like below can be used along with analytical functions:
  - AVG()
  - MIN()
  - MAX()

NOTE:

Analytical functions are widely used in organizations ,because it reduces the number of calls to the same table. Especially when there is a need for SELF Join.

The performance of the Query is high because it consumes less CPU utilization for mapping of rows between tables. However, it depends on the business logic.

Cumulative calculations are also performed in GROUP by clauses, but it is difficult move across the rows among the group.