Practical - 02

Title: - Implementation of Analytical queries

Aim: - Implementation of Analytical queries like Roll_UP, CUBE, First, Last, Lead ,Lag,Rank AND Dense Rank

Lab Objectives: - Students will understand various analytical functions as follows:

I. Ranking Functions
II. Lead, Lag Functions
III. First, Last Functions
IV. Summary Functions
V. OLAP

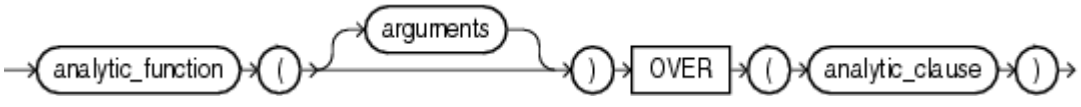
Description: -

Oracle has introduced SQL extensions called Analytic functions in Oracle8 I Release 2. Analytic calculations like moving averages, cumulative sums, ranking, percentile and lag/lead are critical for decision support applications. In order for ROLAP tools and applications to have access to such calculations in an efficient and scalable manner, it is important that relational databases provide a succinct representation in SQL which can be easily optimized.

Analytic functions have been introduced in Oracle 8*i*Release 2 to meet these requirements. These functions typically operate on a window defined on an ordered set of rows. That is, for each row in the input, a window is defined to encompass rows satisfying certain condition relative to the current row and the function is applied over the data values in the window.

Syntax

analytic_function([arguments]) OVER (analytic_clause)



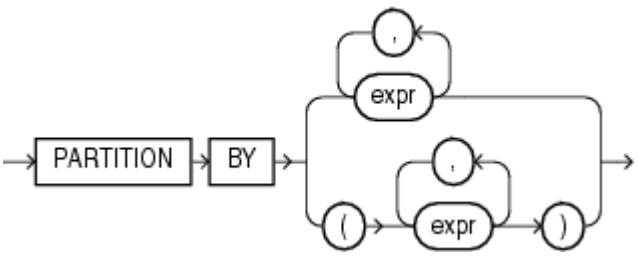
- **analytic_function**
    - Specify the name of an analytic function (see the listing of analytic functions following this discussion of semantics).
- **arguments**

- Analytic functions take 0 to 3 arguments. The arguments can be any numeric data type or any nonnumeric data type that can be implicitly converted to a numeric data type.

- **analytic_clause**

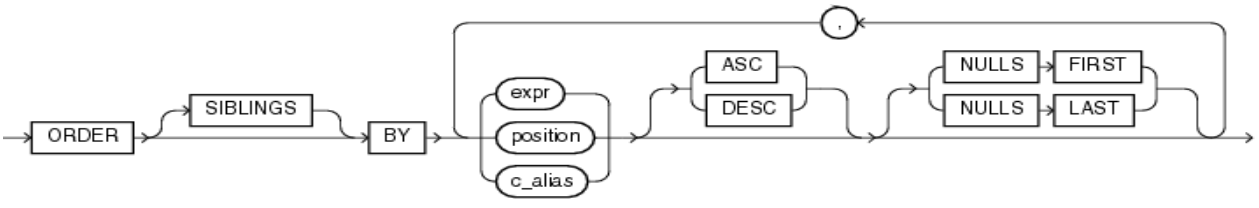
  - Use OVER analytic_clause to indicate that the function operates on a query result set.
  - This clause is computed after the FROM, WHERE, GROUP BY, and HAVING clauses.
  - You can specify analytic functions with this clause in the select list or ORDER BY clause.
  - To filter the results of a query based on an analytic function, nest these functions within the parent query, and then filter the results of the nested subquery.

- **query_partition_clause**



  - Use the PARTITION BY clause to partition the query result set into groups based on one or more value_expr.
  - If you omit this clause, then the function treats all rows of the query result set as a single group.

- **order_by_clause**



  - Use the order_by_clause to specify how data is ordered within a partition.

- For all analytic functions you can order the values in a partition on multiple keys, each defined by a value_expr and each qualified by an ordering sequence.

I. Ranking Functions

These functions assign ranks to rows based on some ordering criteria. Functions in this category are RANK, DENSE_RANK, ROW_NUMBER, NTILE, PERCENT_RANK and CUME_DIST.

A) RANK()

- RANK is a simple analytical function that does not take any column list arguments. It returns a number or a rank based on the ordering of the rows; the ordering is based on a defined condition.
- Similar values are ranked the same
- Syntax

  RANK( ) OVER ([query_partition_clause] order_by_clause)

B) DENSE_RANK()

- DENSE RANK works like RANK in that it needs no additional arguments and it ranks items in descending or ascending order.
- The only difference is that DENSE RANK does not allow “gaps” between groups.
- Synatx

  DENSE_RANK( ) OVER([query_partition_clause] order_by_clause)

C) NTILE()

- NTILE divides rows into equal groups and returns the number of the group that the row belongs to.
- This function is not as widely used as either RANK or DENSE RANK.
- Syntax

  NTILE(expr) OVER ([query_partition_clause] order_by_clause)

D) ROW NUMBER()

- ROW NUMBER is different than DENSE RANK and RANK in that it does not treat identical values in any special way.
- It simply lists them as they occur in some order
- Syntax

  ROW_NUMBER( ) OVER ([query_partition_clause] order_by_clause)

E) PERCENT_RANK()

- Calculate the percent rank of a value relative to a group of values

- The range of values returned by PERCENT_RANK is 0 to 1, inclusive. The first row in any set has a PERCENT_RANK of 0. The return value is NUMBER.
- Syntax

  PERCENT_RANK( ) OVER ([query_partition_clause] order_by_clause)

F) PERCENTILE_CONT()

- Calculates a percentile based on a continuous distribution of the column value
- The result is interpolated and might not be equal to any of the specific values in the column.
- Syntax

  PERCENTILE_CONT(expr) WITHIN GROUP (ORDER BY expr [DESC | ASC])
  [OVER (query_partition_clause)]

G) CUME_DIST()

- CUME_DIST calculates the cumulative distribution of a value in a group of values.
- The range of values returned by CUME_DIST is >0 to <=1.
- Tie values always evaluate to the same cumulative distribution value.
- Syntax

  CUME_DIST() OVER ([query_partition_clause] order_by_clause)

II. LEAD/LAG Functions

The LAG and LEAD analytic functions were introduced in 8.1.6 to give access to multiple rows within a table, without the need for a self-join.

LAG is an analytical function that can be used to get the value of a column in a previous row.

If you want to retrieve the value of the next row, use LEAD instead of lag.

Because the functions provide access to more than one row of a table at the same time without a self-join, they can enhance processing speed.

A) LEAD()

- LEAD returns an offset (incrementally increased) value of an argument column.
- The offset amount can be defined in the code; its default amount is “1”.
- The new value is returned in the same row.
- Syntax

  LEAD (value_expression [,offset] [,default]) OVER ([query_partition_clause] order_by_clause)

B) LAG()

- LAG is the opposite of LEAD. We can even implement LAG using LEAD and vice versa.
- The difference is in the direction we look for the offset value.
- Syntax

LAG (value_expression [,offset] [,default]) OVER ([query_partition_clause] order_by_clause)

III. FIRST/LAST Functions

FIRST and LAST are very similar functions. Both are aggregate and analytic functions that operate on a set of values from a set of rows that rank as the FIRST or LAST with respect to a given sorting specification. If only one row ranks as FIRST or LAST, then the aggregate operates on the set with only one element.

If you omit the OVER clause, then the FIRST and LAST functions are treated as aggregate functions. You can use these functions as analytic functions by specifying the OVER clause.

A) FISRT()

- Return the first value from an ordered sequence.
- Syntax

aggregate_function **KEEP** (**DENSE_RANK FIRST ORDER BY** expr [**DESC** | **ASC**] [**NULLS** { **FIRST** | **LAST** }] [, expr [**DESC** | **ASC**] [**NULLS** { **FIRST** | **LAST** }]]...)
[**OVER** ([query_partition_clause])]

B) LAST()

- Return the last value from an ordered sequence.
- Syntax

aggregate_function **KEEP** (**DENSE_RANK LAST ORDER BY** expr [**DESC** | **ASC**] [**NULLS** { **FIRST** | **LAST** }] [, expr [**DESC** | **ASC**] [**NULLS** { **FIRST** | **LAST** }]]...)
[**OVER** ([query_partition_clause])]

IV. Grouping/Aggregate Functions

- An aggregate function performs a calculation on a set of values and returns a single value.
- Except for COUNT(*), aggregate functions ignore null values.
- Aggregate functions are often used with the GROUP BY clause of the SELECT statement.

- The GROUP BY clause specifies how to group rows from a data table when aggregating information, while the HAVING clause filters out rows that do not belong in specified groups.
- The OVER clause may follow all aggregate functions, except the STRING_AGG, GROUPING or GROUPING_ID functions.
- The GROUP BY clause enables you to use aggregate functions to answer more complex managerial questions such as:
    - What is the average salary of employees in each department?
    - How many employees work in each department?
    - How many employees are working on a particular project?
- Group by function establishes data groups based on columns and aggregates the information within a group only.
- The grouping criterion is defined by the columns specified in GROUP BY clause.
- Following this hierarchy, data is first organized in the groups and then WHERE clause restricts the rows in each group.

A) AVG()

- Returns a running average
- Syntax

  AVG([**DISTINCT** | **ALL**] expr) [**OVER**(analytic_clause)]

B) SUM()

- Computes the cumulative sample running sum
- Syntax

  SUM([**DISTINCT** | **ALL**] expr)
  [**OVER** (analytic_clause)]

C) COUNT()

- Returns a running count of all records or by partition
- Syntax

  COUNT({ * | [**DISTINCT** | **ALL**] expr }) [**OVER** (analytic_clause)]

D) MIN() & MAX()

- The MIN and MAX aggregate functions are used to calculate the minimum and maximum values of a set of data respectively.
- The syntax for MIN and MAX aggregate functions are

SELECT MIN(column_name) FROM table_name

SELECT MAX(column_name) FROM table_name Analytical Functions using query_partion_clause

V. OLAP Operations

OLAP provides a user-friendly environment for interactive data analysis. A number of OLAP data cube operations exist to materialize different views of data, allowing interactive querying and analysis of the data.

The most popular end user operations on dimensional data are:

A) ROLL UP / DRIL UP

The roll-up operation (also called drill-up or aggregation operation) performs aggregation on a data cube, either by climbing up a concept hierarchy for a dimension or by climbing down a concept hierarchy, i.e. dimension reduction.

B) ROLL DOWN

The roll down operation (also called drill down) is the reverse of roll up. It navigates from less detailed data to more detailed data. It can be realized by either stepping down a concept hierarchy for a dimension or introducing additional dimensions.

C) SLICING

Slice performs a selection on one dimension of the given cube, thus resulting in a subcube i.e. focus on a particular slice of the cube

WHERE clause in SQL

D) DICING

The dice operation defines a subcube by performing a selection on two or more dimensions.

GROUP BY clause in SQL

Exercise

1. Create tables Employee and Department as per the given schema and insert data into them.

dept(deptno number(2,0),dname varchar2(14),loc varchar2(13),constraint pk_dept primary key (deptno));

emp(empno number(4,0),ename varchar2(10), job varchar2(9), mgr number(4,0),hiredate date,sal number(7,2),comm number(7,2),deptno number(2,0), constraint pk_emp primary key (empno),constraint fk_deptno foreign key (deptno) references dept (deptno));

Execute the following queries.

Create a table employee with attribute empid, name, deptid,deptname, salary and joining date.
Write the queries -
1. To return the first salary reported in each department.
2. To show us how the average salary has changed over the years
3.To display the salary of each employee, along with the lowest and highest within their department
4. To divide the whole result set into five buckets based on salary
5. To display for each employee in Department 30 in the employees table, the hire date of the employee hired just after

2. Create the table Sales and insert records as given. Write analytical queries -
1. To find total profit for each country.
2. Display Country with maximum profit.
3. Display products with maximum and minimum profit in each country.
4. Display average sale of each product in each country.
5. Find total profit product wise.

year	country	product	profit
2000	Finland	Computer	1500
2000	Finland	Phone	100
2001	Finland	Phone	10
2000	India	Calculator	75
2000	India	Calculator	75
2000	India	Computer	1200
2000	USA	Calculator	75
2000	USA	Computer	1500
2001	USA	Calculator	50
2001	USA	Computer	1500
2001	USA	Computer	1200
2001	USA	TV	150
2001	USA	TV	100

3. The research is about to create the star schema for the sales system. The research consists of all the information related to the sale's record like items, location and the time etc. Create a schema (database) with fact and dimension tables. Perform the OLAP operations on your schema.

- Suppose we want to record in a warehouse information about every Item sale, e.g.:
  - Product number,
  - location from where the product was sold,
  - date of the sale, and
  - Units sold.
- The fact table is thus:

  Sales(item_key, loc_key, time_key, units)

- The dimension tables include information about the Items, Location, and time "dimensions":

  Loc(loc_key, city,state,country)
  items(item_key,item_name, item_category, color,price)
  Time(time_key,sdate,week,month,quarter,syear),

1. Display data for quarter 1.

2. Display total sales of pen or jeans from "mumbai" or "chennai" for quarter 1 or 2.

3. Find the total units sales in each state.

4. Find the total units sales in each city