Oracle 10g analytical SQL for Business Intelligence Reporting

Simay Alpoge
Next Information Systems, Inc.

Oracle 10g analytical SQL for Business Intelligence Reporting

AGENDA

- Windowing aggregate functions.
- Reporting aggregate functions.
- LAG/LEAD functions.
- FIRST/LAST functions.
- Hypothetical Rank and Distribution Functions.
- Defining histograms with CASE statement.
- Data densification for business intelligence reporting.
- Analytical functions vs conventional techniques.

Oracle 10g analytical SQL for Business Intelligence Reporting

```
analytic_function([ arguments ])
  OVER (analytic_clause)

where analytic_clause =
 [ query_partition_clause ]
 [ order_by_clause [ windowing_clause ] ]

and query_partition_clause =
  PARTITION BY { value_expr[, value_expr ]... | (
    value_expr[, value_expr ]... ) }
```

Oracle 10g analytical SQL for Business Intelligence Reporting

windowing_clause = { ROWS | RANGE }
{ BETWEEN { UNBOUNDED PRECEDING |
CURRENT ROW |
value_expr { PRECEDING | FOLLOWING } }
AND { UNBOUNDED FOLLOWING |
CURRENT ROW | value_expr { PRECEDING |
FOLLOWING } }
OR
{ UNBOUNDED PRECEDING | CURRENT ROW |
value_expr PRECEDING } }

Oracle 10g analytical SQL for Business Intelligence Reporting

Processing Order of analytical functions in queries:

1. Joins, WHERE, GROUP BY, HAVING clauses performed.

2. Partitions are created with GROUP BY. Analytical functions are applied to each row in each partition.

3. ORDER BY is processed.

Oracle 10g analytical SQL for Business Intelligence Reporting

- Analytical functions divide a query result sets into partitions.
- Current row is the reference point to determine starting and ending point of the window in a partition.
- Window size can be based on physical number of rows or logical interval.
- Window size of each row can also vary based on specific condition.

Oracle 10g analytical SQL for Business Intelligence Reporting

- Windowing aggregate functions:

  Used to compute cumulative, moving, centered aggregates.

  Access to more than one row of a table without self-join.

  Can ONLY be used in the SELECT and ORDER BY clause of a query.

Oracle 10g analytical SQL for Business Intelligence Reporting

Windowing function with LOGICAL offset

- Constant - RANGE 5
- Interval -RANGE INTERVAL N DAY/MONTH/YEAR ... expression

Multiple sort keys with analytical ORDER BY

- RANGE BETWEEN UNBOUNDED PRECEDING/FOLLOWING

Windowing function with PHYSICAL offset

- Ordering expression have to be unique.

Oracle 10g analytical SQL for Business Intelligence Reporting

Cumulative Aggregate

```
SELECT REGION, QUARTER ,
       SUM(SALES) Q_SALES,
       SUM(SUM(SALES)) OVER (PARTITION BY REGION
   ORDER BY REGION,  QUARTER
       ROWS UNBOUNDED PRECEDING) CUMULATIVE_SALES
    FROM SALES S, TIMES T, LOCATION L
     WHERE S.TIME_ID = T.TIME_ID
       AND S.LOCATION_ID = L.LOCATION_ID
      AND T.CALENDAR_YEAR = '2006'
      AND L.LOCATION_ID IN (234, 356,780)
    GROUP BY REGION, QUARTER
    ORDER BY  REGION, QUARTER ;
```

Oracle 10g analytical SQL for Business Intelligence Reporting

Region	Quarter	Q_Sales	Cumulative_Sales
East	01-2006	100.90	100.90
East	02-2006	150.75	251.65
East	03-2006	200.00	451.65
East	04-2006	500.00	951.65
West	01-2006	1100.00	1100.00
West	02-2006	875.00	1975.00
West	03-2006	950.78	2925.78
West	04-2006	1200.00	4125.78

Oracle 10g analytical SQL for Business Intelligence Reporting

Centered Aggregate

```
SELECT  C_MONTH_ID,
        SUM(SALES) M_SALES,
        AVG(SUM(SALES))  OVER
    (ORDER BY C_MONTH_ID RANGE BETWEEN  INTERVAL '1'
    MONTH PRECEEDING AND '1' MONTH FOLLOWING)
    3_MONTH_SALES
     FROM SALES S, TIMES T
       WHERE S.TIME_ID = T.TIME_ID
          AND T.CALENDAR_YEAR = '2006'
          AND C_MONTH_ID BETWEEN  1 AND 6
      GROUP BY  C_MONTH_ID
      ORDER BY  C_MONTH_ID ;
```

Oracle 10g analytical SQL for Business Intelligence Reporting

C_Month_Id	M_Sales	3_Month_Sales
1	500.00	350.00
2	200.00	300.00
3	200.00	300.00
4	500.00	500.00
5	800.00	633.33
6	600.00	700.00

Oracle 10g analytical SQL for Business Intelligence Reporting

Moving Aggregate

```
SELECT REGION, QUARTER ,
       SUM(SALES) Q_SALES,
       AVG(SUM(SALES)) OVER (PARTITION BY REGION
   ORDER BY REGION,  QUARTER
       ROWS 2 PRECEDING) MOVING_AVG_SALES
    FROM SALES S, TIMES T, LOCATION L
     WHERE S.TIME_ID = T.TIME_ID
        AND S.LOCATION_ID = L.LOCATION_ID
       AND T.CALENDAR_YEAR = '2006'
       AND L.LOCATION_ID IN (234, 356,780)
    GROUP BY REGION, QUARTER
    ORDER BY  REGION, QUARTER;
```

Oracle 10g analytical SQL for Business Intelligence Reporting

Region	Quarter	Q_Sales	Moving_Avg_Sales
East	01-2006	100.90	100.90
East	02-2006	150.75	125.83
East	03-2006	200.00	150.55
East	04-2006	500.00	283.58
West	01-2006	1100.00	1100.00
West	02-2006	875.00	987.50
West	03-2006	950.78	975.26
West	04-2006	1200.00	1008.59

Oracle 10g analytical SQL for Business Intelligence Reporting

- Reporting aggregate functions

  Returns same aggregate value for every row in a partition.

  It does multiple passes of data in a single query block. Excellent query performance result.

Oracle 10g analytical SQL for Business Intelligence Reporting

```
SELECT store_name, prod_grp_desc, tot_sales, tot_costs
FROM (SELECT prod_grp_name,
                  store_name,
                   SUM(sales) tot_sales,
                   SUM(costs) tot_costs,
   MAX(SUM(sales)) OVER (PARTITION BY prod_grp_cd) top_sales,
   MAX(SUM(costs)) OVER (PARTITION BY prod_grp_cd) top_costs
   FROM sales_hist sh, store s, product_grp p inv_major m
    WHERE sh.store_cd = s.store_cd
      AND sh.mjr_cd  = m.inv_mjr_cd
      AND m.prod_grp_cd = p.product_grp_cd
      AND sh.s_date = TO_DATE('01-FEB-2007')
   GROUP BY  prod_grp_name, store_name)
WHERE tot_costs <= top_costs
  AND   tot_sales = top_sales
```

Oracle 10g analytical SQL for Business Intelligence Reporting

Inner query results

PROD_GRP_NAME	STORE_NAME	TOT_SALES	TOT_COSTS	TOP_SALES	TOP_COSTS
ACCESSORIES	5th Ave	1000	200	1000	200
ACCESSORIES	Brooklyn	500	150	1000	200
LADIES SHOES	5th Ave	3000	750	3000	1000
LADIES SHOES	Brooklyn	2000	500	3000	1000
LADIES SHOES	San Francisco	2500	1000	3000	1000
CHILDREN	Houston	3000	650	4000	1000
CHILDREN	5th Ave	4000	900	4000	1000
CHILDREN	Brooklyn	3000	1000	4000	1000

Oracle 10g analytical SQL for Business Intelligence Reporting

Final result

PROD_GRP_ NAME	STORE_ NAME	TOT_ SALES	TOT_ COSTS	TOP_ SALES	TOP_ COSTS
ACCESSORIES	5th Ave	1000	200	1000	200
LADIES SHOES	5th Ave	3000	750	3000	1000
CHILDREN	5th Ave	4000	900	4000	1000

Oracle 10g analytical SQL for Business Intelligence Reporting

LAG/LEAD function

Access of a row at a given offset prior to / after current position.

Access to more than one row of a table at the same time without self-join.

Oracle 10g analytical SQL for Business Intelligence Reporting

```
SELECT SALES_TY, LAG_SALES, LEAD_SALES, SALES_MONTH, SALES_YEAR
FROM
(SELECT SUM(SALES) SALES_TY,
        TO_CHAR(SALES_DT, 'DD-MON') SALES_MONTH,
        TO_CHAR(SALES_DT,'RRRR')    SALES_YEAR,
LAG(SUM(SALES),1) OVER (ORDER BY TO_CHAR(SALES_DT, 'DD-MON'))
  AS LAG_SALES,
LEAD(SUM(SALES),1) OVER (ORDER BY TO_CHAR(SALES_DT,'DD-MON'))
 AS LEAD_SALES
FROM SALES
WHERE TO_CHAR(SALES_DT,'RRRR') IN ('2005','2006')
AND SALES_DT BETWEEN '20-AUG-2006' AND '22-AUG-2006'
OR SALES_DT BETWEEN TO_DATE('20-AUG-2006','DD-MON-RRRR')- 364 AND
    TO_DATE('22-AUG-2006','DD-MON-RRRR') - 364
GROUP BY TO_CHAR(SALES_DT, 'DD-MON'), TO_CHAR(SALES_DT,'RRRR')
ORDER BY SALES_MONTH DESC, SALES_YEAR DESC)
WHERE SALES_YEAR = '2006'
ORDER BY SALES_MONTH
```

Oracle 10g analytical SQL for Business Intelligence Reporting

Inner query results :

SALES_TY	SALES_MONTH	SALES_YEAR	LAG_SALES	LEAD_SALES
5000	20-AUG	2006		3500
3500	20-AUG	2005	5000	
4500	21-AUG	2006		6700
6700	21-AUG	2005	4500	
8300	22-AUG	2006		9500
9500	22-AUG	2005	8300	

Oracle 10g analytical SQL for Business Intelligence Reporting

Final query results :

SALES_TY	SALES_MONTH	SALES_YEAR	LAG_SALES	LEAD_SALES
5000	20-AUG	2006		3500
4500	21-AUG	2006		6700
8300	22-AUG	2006		9500

Oracle 10g analytical SQL for Business Intelligence Reporting

FIRST/LAST function

```
SELECT  prod_category, prod_name, sales,
    MIN(sales) KEEP (DENSE_RANK FIRST ORDER BY cost)
    OVER (PARTITION BY  prod_category) low_sales,
    MAX(sales) KEEP (DENSE_RANK LAST ORDER BY cost) OVER
    (PARTITION BY  prod_category) high_sales
FROM sales_hist
GROUP BY prod_category, prod_name, sales
ORDER BY  prod_category, sales
```

Oracle 10g analytical SQL for Business Intelligence Reporting

Prod Category	Prod Name	Sales	Cost	Low Sales	High Sales
Accessories	Leader Belt	200	50	100	500
...					
Accessories	Leader Belt	100	50	100	300
Accessories	Silk Scarf	600	100	200	600
...					
Accessories	Silk Scarf	800	150	300	800
Handbag	LV	5000	500	3000	35000
Handbag	Coach	8000	400	8000	12000
	RL Shirt	600	45	600	600

Oracle 10g analytical SQL for Business Intelligence Reporting

Prod Category	Prod Name	Sales	Low Sales	High Sales
Accessories	Leader Belt	100	100	300
Accessories	Silk Scarf	600	200	600
Handbag	LV	5000	3000	35000
Handbag	Coach	8000	8000	12000
	RL Shirt	600	600	600

Oracle 10g analytical SQL for Business Intelligence Reporting

Hypothetical rank and distribution functions:
Primarly used for “What if analysis”

RANK
DENSE_RANK
PERCENT_RANK
CUM_DIST

They can not be used as reporting or windowing aggregate functions.

Oracle 10g analytical SQL for Business Intelligence Reporting

```
SELECT REGION, MAX(SCORE) MAX_SCORE,
       MIN(SCORE) MIN_SCORE,
       COUNT(SCORE) SCORE_COUNT,
       RANK (120) WITHIN GROUP (ORDER BY SCORE
   DESC NULLS FIRST) H_RANK
FROM LEAGUE_SCORES
WHERE  T_LEVEL = 3
   AND   G_TYPE = 'BG14'
GROUP BY  REGION
```

Oracle 10g analytical SQL for Business Intelligence Reporting

REGION	MAX_SCORE	MIN_SCORE	SCORE_COUNT	H_RANK
Long Island	100	2	80	1
Metro	200	5	150	12
Northern	180	3	100	7
Southern	110	5	95	1
Western	300	10	165	27

Oracle 10g analytical SQL for Business Intelligence Reporting

Histograms with CASE

```
SELECT SUM (CASE WHEN SALES BETWEEN 100 AND 5000
        THEN 1 ELSE 0 END) AS “100 – 5000”,
SUM(CASE WHEN SALES BETWEEN 5001 AND 15000
        THEN 1 ELSE 0 END) AS “5001 – 15000”,
SUM (CASE WHEN SALES BETWEEN 15001 AND 25000
        THEN 1 ELSE 0 END ) AS “15001 – 25000”
FROM SALES
WHERE REGION = ‘WEST’
```

Oracle 10g analytical SQL for Business Intelligence Reporting

100 – 5000	5001 – 15000	15001 – 25000
10	5	18

Oracle 10g analytical SQL for Business Intelligence Reporting

```
SELECT  (CASE WHEN SALES BETWEEN 100 AND 5000
                THEN '100 – 5000'
                WHEN SALES BETWEEN 5001 AND 15000
                THEN '5001 – 15000'
                WHEN SALES BETWEEN  15001 AND 25000
                THEN '15001 – 25000' END ) AS SALES_BUCKET,
        COUNT(*) AS SALES_CNT
FROM  SALES
  WHERE REGION = 'WEST'
 GROUP BY (CASE WHEN SALES BETWEEN 100 AND 5000
                  THEN '100 – 5000'
                 WHEN SALES BETWEEN 5001 AND 15000
                  THEN '5001 – 15000'
                WHEN SALES BETWEEN  15001 AND 25000
                 THEN '15001 –  25000' END )
```

Oracle 10g analytical SQL for Business Intelligence Reporting

SALES_BUCKET	SALES_CNT
100 – 5000	10
5001 – 15000	5
15001 – 25000	18

Oracle 10g analytical SQL for Business Intelligence Reporting

Data densification

Process of converting sparse data into dense form.

SELECT ..... FROM table_reference
PARTITION BY (expr [, expr]...)
RIGHT OUTER JOIN table_reference

Partition outer join fills the gaps in time series or any other dimensions.

Oracle 10g analytical SQL for Business Intelligence Reporting

Product	Year	Month	Day	Sales
Oracle Fusion	2007	01	01	100
Oracle Fusion	2007	01	08	370
Global Economy	2007	01	04	300
Global Economy	2007	01	07	500

Oracle 10g analytical SQL for Business Intelligence Reporting

```
Select product, day, NVL(sales,0) SALES
FROM
(Select Day, Product, SUM(Sales) Sales
FROM sales s, f_calendar f, products p
WHERE s.sale_date = f.cal_date
AND s.product_id = p.product_id
AND f.cal_year = '2007' AND f.cal_mnth = '01'
AND f.day between '01' and '08'
GROUP BY Product, Day) x
PARTITION BY (product)
RIGHT OUTER JOIN
(SELECT day FROM f_calendar
 WHERE cal_year = '2007' AND cal_mnth = '01'
  AND day between '01' and '08') ff
ON (ff.day = x.day))
ORDER BY product, day
```

Oracle 10g analytical SQL for Business Intelligence Reporting

Product	Day	Sales
Oracle Fusion	01	100
Oracle Fusion	02	0
Oracle Fusion	03	0
Oracle Fusion	04	0
Oracle Fusion	05	0
Oracle Fusion	06	0
Oracle Fusion	07	0
Oracle Fusion	08	370
Global Economy	01	0
Global Economy	02	0
Global Economy	03	0
Global Economy	04	300
Global Economy	05	0
Global Economy	06	0
Global Economy	07	500
Global Economy	08	0

Oracle 10g analytical SQL for Business Intelligence Reporting

Partition outer join repeating value

Inventory table :

Product	Time_id	Quantity
Oracle Fusion	03-Feb-07	10
Oracle Fusion	05-Feb-07	30
Oracle Fusion	10-Feb-07	35
Global Economy	03-Feb-07	45
Global Economy	05-Feb-07	15
Global Economy	10-Feb-07	25

Oracle 10g analytical SQL for Business Intelligence Reporting

```
SELECT PRODUCT,TIME_ID, QUANTITY,
LAST_VALUE (QUANTITY, IGNORE NULLS)
OVER
  (PARTITION BY product
       ORDER BY time_id ) R_QUANTITY
FROM (
  SELECT times.time_id, product, quantity
  FROM inventory
    PARTITION BY (product)
    RIGHT OUTER JOIN times
    ON (times.time_id=inventory.time_id) );
```

Oracle 10g analytical SQL for Business Intelligence Reporting

Product	Time_id	Quantity	R_Quantity
Oracle Fusion	03-Feb-07	10	10
Oracle Fusion	04-Feb-07		10
Oracle Fusion	05-Feb-07	30	30
Oracle Fusion	06-Feb-07		30
Oracle Fusion	07-Feb-07		30
Oracle Fusion	08-Feb-07		30
Oracle Fusion	09-Feb-07		30
Oracle Fusion	10-Feb-07	35	35
Global Economy	03-Feb-07	45	45
Global Economy	04-Feb-07		45
Global Economy	05-Feb-07	15	15
Global Economy	06-Feb-07		15
Global Economy	07-Feb-07		15
Global Economy	08-Feb-07		15
Global Economy	09-Feb-07		15
Global Economy	10-Feb-07	25	25

Oracle 10g analytical SQL for Business Intelligence Reporting

Analytical functions vs conventional techniques

Cumulative/Moving aggregation – PL/SQL tables, permenant/temporary tables.

Windowing – Multiple sub-queries/views.

Densification – Temporary/Permenant tables.

UNION/UNION ALL

Views

Oracle 10g analytical SQL for Business Intelligence Reporting

References:

http://www.oracle.com/technology/documentation/index.html

http://www.asktom.oracle.com

THANK YOU.

alpoges@aol.com