



COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY



ORACLE ANALYTIC FUNCTIONS WINDOWING CLAUSE

Session 740
Dan Stober
Tuesday, April 24, 2012



COLLABORATE12
TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

Dan Stober



- Data Architect – Intermountain Healthcare
- Attended California State Univ., Fresno
- Working in Oracle databases since 2001
- Frequent presenter at local and national user group conferences
 - Oracle Open World - twice
- Private Instructor for Trutek
 - Teaching PLSQL
- Oracle Certified SQL Expert
- Board of Trustees – Utah Oracle Users Group (UTOUG)
 - Edit newsletter
 - Write SQL Tip column



COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY



Dan Stober – Personal Stuff

- Married for twenty years with two kids
- Run two marathons
 - Next one in four weeks
- Three Ragnars
- Enjoy
 - Photography
 - Travel





COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY



Intermountain Healthcare

- 23 hospitals in Utah and Idaho
- Non-profit integrated health care system
- 750 Employed physicians
- 32,000 employees
 - The largest non-government employer in Utah
- One of the largest and most complete clinical data warehouses in the world!





COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY



Session Norms

- Questions?
 - Interrupt Me!
- I learn something from every session I do!
 - Set the record straight!
- Cell phones?
 - OK!



COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY



Shameless Plug



- UTOUG Fall Symposium
- Thursday, September 6
 - Salt Lake City
- Call For Presentations is open!
 - Deadline: June 15
- All things Oracle: DBA, Development, APEX, EBS, Business Intelligence



COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY



Agenda

- Aggregate vs Analytic
 - PARTITION BY
 - ORDER BY
 - Window Clause
 - ROWS
 - RANGE
-

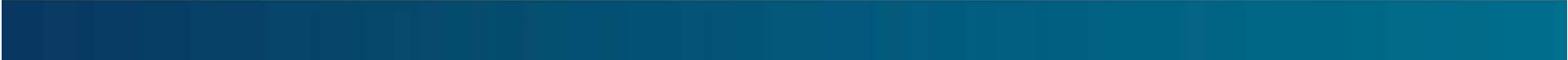


COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY



WHY USE ANALYTIC FUNCTIONS?

- Ability to see one row from another row in the results
 - Avoid self-join queries
 - Summary data in detail rows
 - Slice and dice within the results
- 



AGGREGATE OR ANALYTIC?

Which one are each of these?	Aggregate	Analytic
COUNT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
SUM	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
MAX	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
MIN	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

•What's the difference?

	SYNTAX	OUTPUT
Aggregate (traditional)	Query often includes the keywords GROUP BY	Output is a single row (or one row per group with GROUP BY)
Analytic	OVER (some other stuff)	Does not change number of rows



AGGREGATE EXAMPLES

```
SELECT COUNT ( * )
FROM scott.emp;
```

```
  COUNT(*)
-----
         14

1 row selected.
```

```
SELECT SUM ( sal )
FROM scott.emp;
```

```
  SUM(SAL)
-----
      29025

1 row selected.
```

```
SELECT COUNT ( * )
, SUM ( sal )
, MAX ( sal )
, MIN ( ename )
FROM scott.emp;
```

```
  COUNT(*)   SUM(SAL)   MAX(SAL)   MIN(ENAME)
-----
         14       29025       5000 ADAMS

1 row selected.
```

Deptno	Ename	Sal
10	Clark	2450
10	King	5000
10	Miller	1300
20	Adams	1100
20	Ford	3000
20	Jones	2975
20	Scott	3000
20	Smith	800
30	Allen	1600
30	Blake	2850
30	James	950
30	Martin	1250
30	Turner	1500
30	Ward	1250



COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

AGGREGATE EXAMPLES



```
SELECT COUNT ( * )
       , SUM ( sal )
FROM scott.emp
WHERE deptno = 30;
```

COUNT(*)	SUM(SAL)
6	9400

1 row selected.

```
SELECT deptno
       , COUNT ( * )
       , SUM ( sal )
FROM scott.emp
GROUP BY deptno;
```

DEPTNO	COUNT(*)	SUM(SAL)
10	3	8750
20	5	10875
30	6	9400

3 rows selected.

```
SELECT deptno
       , COUNT ( * )
       , SUM ( sal )
FROM scott.emp;
```

```
*
ERROR at line 1:
ORA-00937: not a single-group group function
```

One record for
each group

Deptno	Ename	Sal
10	Clark	2450
10	King	5000
10	Miller	1300
20	Adams	1100
20	Ford	3000
20	Jones	2975
20	Scott	3000
20	Smith	800
30	Allen	1600
30	Blake	2850
30	James	950
30	Martin	1250
30	Turner	1500
30	Ward	1250



ANALYTIC FUNCTIONS

What makes a function analytic?

- Keyword **OVER**
- Followed by set of parentheses

```
SELECT deptno, ename, sal  
  , COUNT ( * ) OVER ()  
  , SUM ( sal ) OVER ()  
FROM scott.emp;
```

DEPTNO	ENAME	SAL	COUNT(*) OVER()	SUM(SAL) OVER()
10	CLARK	2450	14	29025
10	KING	5000	14	29025
10	MILLER	1300	14	29025
20	ADAMS	1100	14	29025
20	FORD	3000	14	29025
20	JONES	2975	14	29025
20	SCOTT	3000	14	29025
20	SMITH	800	14	29025
30	ALLEN	1600	14	29025
30	BLAKE	2850	14	29025
30	JAMES	950	14	29025
30	MARTIN	1250	14	29025
30	TURNER	1500	14	29025
30	WARD	1250	14	29025

14 rows selected.

Returns **one result**
for each record in the dataset.
No grouping



ANALYTIC FUNCTIONS

With WHERE Clause...

•Which happens first?

```
SELECT deptno, ename, sal
, COUNT ( * ) OVER ( )
, SUM ( sal ) OVER ( )
FROM scott.emp
WHERE deptno = 30;
```

Even with OVER() and empty parens, the function operates only on the records which meet the conditions of the WHERE clause

DEPTNO	ENAME	SAL	COUNT(*)OVER()	SUM(SAL)OVER()
30	ALLEN	1600	6	9450
30	BLAKE	2850	6	9450
30	JAMES	950	6	9450
30	MARTIN	1250	6	9450
30	TURNER	1500	6	9450
30	WARD	1250	6	9450

6 rows selected.

Deptno	Ename	Sal
10	Clark	2450
10	King	5000
10	Miller	1300
20	Adams	1100
20	Ford	3000
20	Jones	2975
20	Scott	3000
20	Smith	800
30	Allen	1600
30	Blake	2850
30	James	950
30	Martin	1250
30	Turner	1500
30	Ward	1250



DISTINCT vs GROUP BY

When analytic functions are involved:

YES

Is there a difference?

```
SELECT deptno
, COUNT(*) OVER ( ) AS empcnt
FROM scott.emp
GROUP BY deptno;
```

DEPTNO	EMPCNT
10	3
20	3
30	3

3 rows selected.

```
SELECT DISTINCT deptno
, COUNT(*) OVER ( ) AS empcnt
FROM scott.emp;
```

DEPTNO	EMPCNT
10	14
20	14
30	14

3 rows selected.

1. Table Joins
2. WHERE clause filters
3. GROUP BY
4. Analytic Functions
5. DISTINCT
6. Ordering



COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY



The Analytic Clause

The stuff inside the parentheses





COLLABORATE12
TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

THE ANALYTIC CLAUSE



- Within the set of parentheses
- Expressions telling the function to calculate differently
- Three possible components
 - Partition
 - Order
 - Windowing
- Some or all are optional, depending upon the function
- Components *must* be in this order



PARTITION BY



Analytic function
calculated on a subset of
the records

Can differ for each one

```
SELECT deptno, ename, sal, job
, COUNT ( * ) OVER ( PARTITION BY job ) jobcount
, SUM ( sal ) OVER ( PARTITION BY deptno ) deptsum
FROM scott.emp;
```

DEPTNO	ENAME	SAL	JOB	JOBCOUNT	DEPTSUM
10	CLARK	2450	MANAGER	3	8750
10	KING	5000	PRESIDENT	1	8750
10	MILLER	1300	CLERK	4	8750
20	ADAMS	1100	CLERK	4	10875
20	FORD	3000	ANALYST	2	10875
20	JONES	2975	MANAGER	3	10875
20	SCOTT	3000	ANALYST	2	10875
20	SMITH	800	CLERK	4	10875
30	ALLEN	1600	SALESMAN	4	9400
30	BLAKE	2850	MANAGER	3	9400
30	JAMES	950	CLERK	4	9400
30	MARTIN	1250	SALESMAN	4	9400
30	TURNER	1500	SALESMAN	4	9400
30	WARD	1250	SALESMAN	4	9400

14 rows selected.



HERE'S THE SAME QUERY

Using aggregate
functions in the
traditional manner

Correlated
scalar
subqueries

Same Results as
prior slide

```
SELECT deptno, ename, sal, job
, ( SELECT COUNT ( * ) FROM scott.emp WHERE job = e.job ) jobcount
, ( SELECT SUM ( sal ) FROM scott.emp WHERE deptno = e.deptno ) deptsum
FROM scott.emp e;
```

DEPTNO	ENAME	SAL	JOB	JOBCOUNT	DEPTSUM
10	CLARK	2450	MANAGER	3	8750
10	KING	5000	PRESIDENT	1	8750
10	MILLER	1300	CLERK	4	8750
20	ADAMS	1100	CLERK	4	10875
20	FORD	3000	ANALYST	2	10875
20	JONES	2975	MANAGER	3	10875
20	SCOTT	3000	ANALYST	2	10875
20	SMITH	800	CLERK	4	10875
30	ALLEN	1600	SALESMAN	4	9400
30	BLAKE	2850	MANAGER	3	9400
30	JAMES	950	CLERK	4	9400
30	MARTIN	1250	SALESMAN	4	9400
30	TURNER	1500	SALESMAN	4	9400
30	WARD	1250	SALESMAN	4	9400

14 rows selected.



COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

EXPLAIN PLAN



PLAN_TABLE_OUTPUT

Plan hash value: 1174980467

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	294	3 (0)	00:00:01
1	SORT AGGREGATE		1	8		
* 2	TABLE ACCESS FULL	EMP	14	294	3 (0)	00:00:01
3	SORT AGGREGATE		1	8		
* 4	TABLE ACCESS FULL	EMP	14	294	3 (0)	00:00:01
5	TABLE ACCESS FULL	EMP	14	294	3 (0)	00:00:01

Traditional
aggregate
syntax.
Three passes
over the table

PLAN_TABLE_OUTPUT

Plan hash value: 4086863039

Predicate Information (identified by name)

- 2 - filter("JOB"=: B1)
- 4 - filter("DEPTNO"=: B1)

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	294	5 (40)	00:00:01
1	WINDOW SORT		14	294	5 (40)	00:00:01
2	WINDOW SORT		14	294	5 (40)	00:00:01
3	TABLE ACCESS FULL	EMP	14	294	3 (0)	00:00:01

Analytic SQL.
ONE PASS!



COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

TWO NEW FUNCTIONS



- LAG
- LEAD
- Usage:
 - `LAG (field_name, num_recs) OVER ()`
 - Return the value from a field when looking one record (or more) behind/ahead
 - Using the order specified
 - ORDER BY is required
 - Does not have to be order used in the query
 - Optional second param to look more than one record
 - These functions are analytic only



LAG DEMONSTRATION



```
SELECT deptno, ename, hiredate
, LAG ( ename ) OVER ( ORDER BY hiredate ) prior_hire
FROM scott.emp
ORDER BY deptno, ename;
```

DEPTNO	ENAME	HI REDATE	PRI OR_HI RE
10	CLARK	09-JUN-81	BLAKE
10	KING	17-NOV-81	MARTIN
10	MILLER	23-JAN-82	FORD
20	ADAMS	23-MAY-87	SCOTT
20	FORD	03-DEC-81	JAMES
20	JONES	02-APR-81	WARD
20	SCOTT	09-DEC-82	MILLER
20	SMITH	17-DEC-80	
30	ALLEN	20-FEB-81	SMITH
30	BLAKE	01-MAY-81	JONES
30	JAMES	03-DEC-81	KING
30	MARTIN	28-SEP-81	TURNER
30	TURNER	08-SEP-81	CLARK
30	WARD	22-FEB-81	ALLEN

14 rows selected.

ENAME	HIREDATE
ADAMS	1/12/1983
ALLEN	2/20/1981
BLAKE	5/1/1981
CLARK	6/9/1981
FORD	12/3/1981
JAMES	12/3/1981
JONES	4/02/1981
KING	11/17/1981
MARTIN	9/28/1981
MILLER	1/23/1982
SCOTT	12/9/1982
SMITH	12/17/1980
TURNER	9/08/1981
WARD	2/22/1981





COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

ORDER BY VARIATION



```

SELECT deptno, ename, sal
, LAG ( ename ) OVER ( ORDER BY ename ) f1
, LAG ( ename , 2 ) OVER ( ORDER BY ename ) f2
, LEAD ( ename ) OVER ( ORDER BY ename DESC ) f3
, LAG ( sal ) OVER ( ORDER BY ename ) f4
FROM scott.emp
ORDER BY deptno, ename;

```

DEPTNO	ENAME	SAL	F1	F2	F3	F4
10	CLARK	2450	BLAKE	ALLEN	BLAKE	2850
10	KING	5000	JONES	JAMES	JONES	2975
10	MILLER	1300	MARTIN	KING	MARTIN	1250
20	ADAMS	1100				
20	FORD	3000	CLARK	BLAKE	CLARK	2450
20	JONES	2975	JAMES	FORD	JAMES	950
20	SCOTT	3000	MILLER	MARTIN	MILLER	1300
20	SMITH	800	SCOTT	MILLER	SCOTT	3000
30	ALLEN	1600	ADAMS		ADAMS	1100
30	BLAKE	2850	ALLEN	ADAMS	ALLEN	1600
30	JAMES	950	FORD	CLARK	FORD	3000
30	MARTIN	1250	KING	JONES	KING	5000
30	TURNER	1500	SMITH	SCOTT	SMITH	800
30	WARD	1250	TURNER	SMITH	TURNER	1500

14 rows selected.

ENAME
ADAMS
ALLEN
BLAKE
CLARK
FORD
JAMES
JONES
KING
MARTIN
MILLER
SCOTT
SMITH
TURNER
WARD



COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

ORDER BY WITH PARTITION BY



```

SELECT deptno, ename, sal
, LAG ( ename ) OVER ( ORDER BY ename ) f1
, LAG ( ename ) OVER ( PARTITION BY deptno ORDER BY ename ) f2
, LAG ( ename ) OVER ( PARTITION BY deptno ORDER BY sal DESC) f3
FROM scott.emp
ORDER BY deptno, ename;

```

Deptno	Ename	Sal
10	Clark	2450
10	King	5000
10	Miller	1300
20	Adams	1100
20	Ford	3000
20	Jones	2975
20	Scott	3000
20	Smith	800
30	Allen	1600
30	Blake	2850
30	James	950
30	Martin	1250
30	Turner	1500
30	Ward	1250

DEPTNO	ENAME	SAL	F1	F2	F3
10	CLARK	2450	BLAKE		KING
10	KING	5000	JONES	CLARK	
10	MILLER	1300	MARTIN	KING	CLARK
20	ADAMS	1100			JONES
20	FORD	3000	CLARK	ADAMS	SCOTT
20	JONES	2975	JAMES	FORD	FORD
20	SCOTT	3000	MILLER	JONES	
20	SMITH	800	SCOTT	SCOTT	ADAMS
30	ALLEN	1600	ADAMS		BLAKE
30	BLAKE	2850	ALLEN	ALLEN	
30	JAMES	950	FORD	BLAKE	WARD
30	MARTIN	1250	KING	JAMES	TURNER
30	TURNER	1500	SMITH	MARTIN	ALLEN
30	WARD	1250	TURNER	TURNER	MARTIN

14 rows selected.



ORDER OF ITEMS IN ANALYTIC CLAUSE

```
SELECT deptno, empno, ename, sal  
  , MIN ( sal ) OVER ( ORDER BY ename PARTITION BY deptno ) mi nsal  
FROM scott.emp;
```

```
  , MIN ( sal ) OVER ( ORDER BY ename PARTITION BY deptno ) mi nsal  
                        *
```

ERROR at line 2:
ORA-00907: missing right parenthesis

Components
must be in
correct order

Web Images Videos Maps News Shopping Gmail more ▾



ORA-00907 missing right parenthesis

About 26,700 results (0.05 seconds)



Advanced search

ORA-00907 missing right parenthesis

Cause: A left parenthesis has been entered without a closing right parenthesis, or extra information was contained in the parentheses. All parentheses must be entered in pairs.

Action: Correct the syntax and retry the statement.



THREE MORE NEW FUNCTIONS

- Ordering (Ranking) functions:
 - RANK
 - DENSE_RANK
 - ROW_NUMBER
- Usage:
 - RANK () OVER (ORDER BY *field_name*)
- Where does this record fall, when the records are placed in a certain order?
 - Does not have to be order used in the query
- All three functions return a number
- Difference between functions is how they handle ties
- These functions are analytic only



COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

RANKING FUNCTIONS



```
SELECT deptno, ename, sal
, RANK ( ) OVER ( ORDER BY ename ) f1
, DENSE_RANK ( ) OVER ( ORDER BY ename ) f2
, ROW_NUMBER ( ) OVER ( ORDER BY ename ) f3
FROM scott.emp
ORDER BY deptno, sal;
```

DEPTNO	ENAME	SAL	F1	F2	F3
10	MILLER	1300	10	10	10
10	CLARK	2450	4	4	4
10	KING	5000	8	8	8
20	SMITH	800	12	12	12
20	ADAMS	1100	1	1	1
20	JONES	2975	7	7	7
20	FORD	3000	5	5	5
20	SCOTT	3000	11	11	11
30	JAMES	950	6	6	6
30	WARD	1250	14	14	14
30	MARTIN	1250	9	9	9
30	TURNER	1500	13	13	13
30	ALLEN	1600	2	2	2
30	BLAKE	2850	3	3	3

14 rows selected.

When there are no ties, all three of these functions return the same values.



RANKING FUNCTIONS WITH TIES

```
SELECT deptno, ename, sal
, RANK ( ) OVER ( ORDER BY sal ) f1
, DENSE_RANK ( ) OVER ( ORDER BY sal ) f2
, ROW_NUMBER ( ) OVER ( ORDER BY sal ) f3
FROM scott.emp
ORDER BY deptno, sal ;
```

DEPTNO	ENAME	SAL	F1	F2	F3
10	MILLER	1300	6	5	6
10	CLARK	2450	9	8	9
10	KING	5000	14	12	14
20	SMITH	800	1	1	1
20	ADAMS	1100	3	3	3
20	JONES	2975	11	10	11
20	FORD	3000	12	11	13
20	SCOTT	3000	12	11	12
30	JAMES	950	2	2	2
30	WARD	1250	4	4	4
30	MARTIN	1250	4	4	5
30	TURNER	1500	7	6	7
30	ALLEN	1600	8	7	8
30	BLAKE	2850	10	9	10

14 rows selected.

RANK and DENSE_RANK will assign the same number to multiple records with the same sort value

The difference is in how each one handles the record which follows

ROW_NUMBER assigns a unique number to each record. The highest value assigned by ROW_NUMBER will be equal to COUNT(*)



COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

ORDER BY CAVEAT #1

There is no assurance the row_number() assignments would not be different for the \$3000 sal on the next time the query is executed

```
SELECT deptno, ename, job, sal, hiredate  
  , ROW_NUMBER ( ) OVER ( ORDER BY sal DESC) r1  
  , ROW_NUMBER ( ) OVER ( PARTITION BY job ORDER BY sal ) r2  
FROM scott.emp;
```

DEPTNO	ENAME	JOB	SAL	HI	REDATE	R1	R2
10	CLARK	MANAGER	2450	09-JUN-81		6	1
10	KING	PRESIDENT	5000	17-NOV-81		1	1
10	MILLER	CLERK	1300	23-JAN-82		9	4
20	ADAMS	CLERK	1100	23-MAY-87		12	3
20	FORD	ANALYST	3000	03-DEC-81		2	1
20	JONES	MANAGER	2975	02-APR-81		4	3
20	SCOTT	ANALYST	3000	19-APR-87		3	2
20	SMITH	CLERK	800	17-DEC-80		14	1
30	ALLEN	SALESMAN	1600	20-FEB-81		7	4
30	BLAKE	MANAGER	2850	01-MAY-81		5	2
30	JAMES	CLERK	950	03-DEC-81		13	2
30	MARTIN	SALESMAN	1250	28-SEP-81		10	1
30	TURNER	SALESMAN	1500	08-SEP-81		8	3
30	WARD	SALESMAN	1250	22-FEB-81		11	2

14 rows selected.



ORDER BY CAVEAT #2

- On many functions, using ORDER BY changes window
 - SUM, COUNT, MAX, MIN, LAST_VALUE

```
SELECT deptno, ename, sal
, SUM ( sal ) OVER ( ORDER BY ename ) s
, COUNT ( * ) OVER ( ORDER BY ename ) c
, MIN ( sal ) OVER ( ORDER BY ename ) mn
, MAX ( sal ) OVER ( ORDER BY ename ) mx
FROM scott.emp
WHERE deptno = 10;
```

On each record, results are from the beginning of the partition to the current record, as defined by the ORDER BY

DEPTNO	ENAME	SAL	S	C	MN	MX
10	CLARK	2450	2450	1	2450	2450
10	KING	5000	7450	2	2450	5000
10	MILLER	1300	8750	3	1300	5000

3 rows selected.



WHY?

This is the default behavior.

If you include an ORDER BY where one would not be necessary, Oracle assumes it is there for a reason.

$$1 + 3 + 5 = 9 \text{ and } 5 + 1 + 3 = 9$$

Very powerful for running calculations, such as MTD:

Week Number	Sales	Month To Date
1	11,000	11,000
2	15,000	26,000
3	12,000	38,000
4	16,000	54,000

```
SUM ( sales ) OVER  
(ORDER BY week_number)
```



DEFAULT WINDOWING

... OVER (PARTITION BY cust)

Cust	Order_Date
A	12/25/2010
A	1/15/2011
A	2/28/2011
B	6/16/2010
B	9/15/2010
B	1/1/2011
B	2/12/2011

Calculation on each of these records includes all three of these records

Calculation on each of these records includes all four of these records

COUNT (*)
OVER ...

COUNT
3
3
3
4
4
4
4



DEFAULT WINDOWING

... OVER (PARTITION BY cust
ORDER BY order_date)

COUNT (*)
OVER ...

Cust	Order_Date
A	12/25/2010
A	1/15/2011
A	2/28/2011
B	6/16/2010
B	9/15/2010
B	1/1/2011
B	2/12/2011

Calculation on each of these records includes only the records which preceded it in the partition

Calculation on each of these records includes only the records which preceded it in the partition

COUNT
3
3
3
4
4
4
4
4

COUNT
1
2
3
1
2
3
4



- Demonstration of default windowing
- With and without ORDER BY

```
SELECT deptno, ename, sal
, SUM ( sal ) OVER ( ) sum1
, SUM ( sal ) OVER ( ORDER BY ename ) sum2
, SUM ( sal ) OVER ( ORDER BY ename
    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING ) sum3
, SUM ( sal ) OVER ( ORDER BY ename
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW ) sum4
FROM scott.emp
WHERE deptno = 10;
```

SUM 1 is the same as SUM3
SUM 2 is the same as SUM4

Default windowing saves a lot of typing and eliminates clutter

DEPTNO	ENAME	SAL	SUM1	SUM2	SUM3	SUM4
10	CLARK	2450	8750	2450	8750	2450
10	KING	5000	8750	7450	8750	7450
10	MILLER	1300	8750	8750	8750	8750

3 rows selected.



WINDOWING

```
SELECT deptno, ename, sal
, SUM ( sal ) OVER ( ORDER BY ename
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING ) sum1
, SUM ( sal ) OVER ( PARTITION BY deptno ORDER BY ename
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING ) sum2
FROM scott.emp;
```

DEPTNO	ENAME	SAL	SUM1	SUM2
10	CLARK	2450	8300	7450
10	KING	5000	9225	8750
10	MILLER	1300	5550	6300
20	ADAMS	1100	2700	4100
20	FORD	3000	6400	7075
20	JONES	2975	8925	8975
20	SCOTT	3000	5100	6775
20	SMITH	800	5300	3800
30	ALLEN	1600	5550	4450
30	BLAKE	2850	6900	5400
30	JAMES	950	6925	5050
30	MARTIN	1250	7550	3700
30	TURNER	1500	3550	4000
30	WARD	1250	2750	2750

14 rows selected.

- Selects a smaller subset than the partition
- Based on a number of records before/after
 - Or a time period before/after

2450+5000

2450+5000+1300

5000+1300

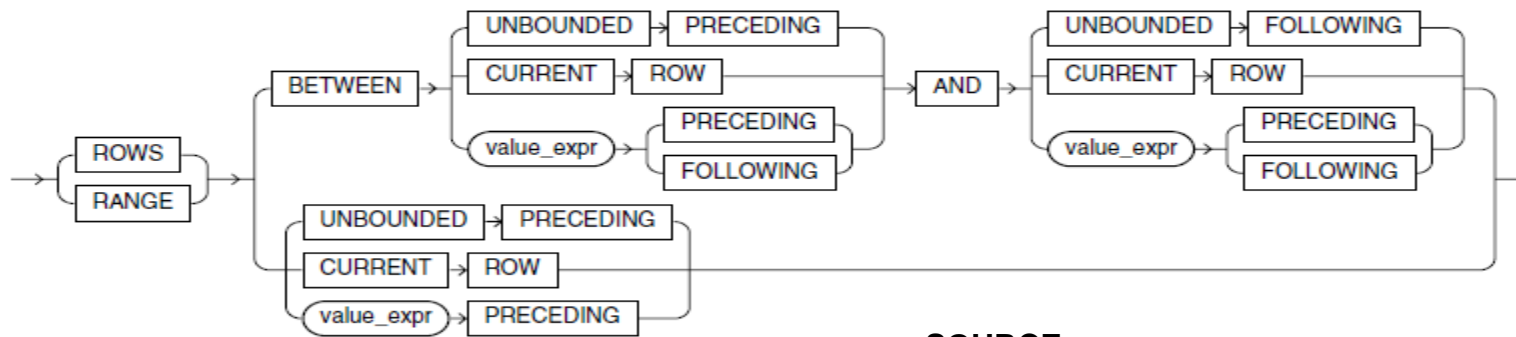
1100+3000

1100+3000+2975



SYNTAX DIAGRAM

windowing_clause::=



SOURCE:

Oracle Database SQL Language Reference
11g Release 2 (E10592-04)
Page 5-12



WINDOWING CLAUSE COMPARISON

Rows

Row windowing:
Restricts window by records
Based on ORDER BY

```
ROWS BETWEEN 10 PRECEDING  
AND 10 FOLLOWING
```

Analytic function will include the 10 records just before this record and the 10 records after

Range

Range windowing:
Restricts window by a period of time or a value
References field used in ORDER BY

```
RANGE BETWEEN INTERVAL '10' DAY PRECEDING  
AND INTERVAL '10' DAY FOLLOWING
```

Analytic function will include all records within 10 days of the record in question



RANGE WINDOWING

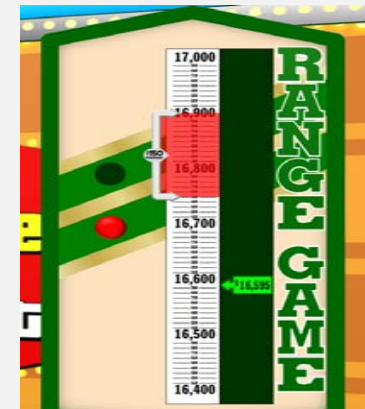
RANGE
not ROWS

```
SELECT ename, sal
, COUNT(*) OVER ( ORDER BY sal RANGE BETWEEN 200 PRECEDING
AND 200 FOLLOWING ) emps_200_sal
FROM scott.emp;
```

ENAME	SAL	EMPS_200_SAL
SMITH	800	2
JAMES	950	3
ADAMS	1100	5
WARD	1250	4
MARTIN	1250	4
MILLER	1300	5
TURNER	1500	3
ALLEN	1600	2
CLARK	2450	1
BLAKE	2850	4
JONES	2975	4
SCOTT	3000	4
FORD	3000	4
KING	5000	1

Consider only those records within \$200 of the value from the current record

Which field?
SAL: The field that is used in the ORDER BY



14 rows selected.



WITH AN INTERVAL

```
SELECT empno, ename, hiredate  
, COUNT(*) OVER ( ORDER BY hiredate  
                    RANGE BETWEEN INTERVAL '1 3' DAY TO HOUR FOLLOWING  
                    AND INTERVAL '1-6' YEAR TO MONTH FOLLOWING ) AS
```

example

FROM scott.emp;

- This is just an extreme example:
 - Window includes people hired from
 - One day and three hours after the current record, to
 - One year and six months after the current record
- The real point on display here ...
 - How do you use intervals?



ASIDE

- Designating an Interval
- An Interval is a period of time
 - Between two dates or two timestamps

```
INTERVAL '10' DAY
```

1. Why is the number enclosed in single quotes?
2. Why is the unit singular?
 - “DAY” instead of “DAYS”?



DESIGNATING AN INTERVAL

I N T E R V A L ' 3 ' D A Y

Keyword

Number of Units

Unit Type(s)

I N T E R V A L ' 7 ' H O U R

I N T E R V A L ' 7 : 4 5 ' H O U R T O M I N U T E

I N T E R V A L ' 7 : 4 5 ' M I N U T E T O S E C O N D

I N T E R V A L ' 7 : 4 5 : 0 0 ' H O U R T O S E C O N D

I N T E R V A L ' 3 7 : 4 5 : 0 0 ' D A Y T O S E C O N D

I N T E R V A L ' 3 7 : 4 5 ' D A Y T O M I N U T E

Varchar

- ▶ Number of Units is a varchar string
 - ▶ (enclosed in single quotes)
- ▶ Number of units can include values for more than one unit type
- ▶ Multiple units: specify first and last, separated by keyword "TO"

Think of these units designations as akin to a format mask used with TO_DATE. You are specifying the significance of the numbers. Note that you include only the first and last units.



DESIGNATING AN INTERVAL

```
SELECT INTERVAL '3' DAY AS interv_1
, INTERVAL '3 00:00:00' DAY TO SECOND AS interv_2
, INTERVAL '72' HOUR AS interv_3
, INTERVAL '4320' MINUTE AS interv_4
FROM dual ;
```

INTERV_1	INTERV_2	INTERV_3	INTERV_4
+03 00:00:00	+03 00:00:00.000000	+03 00:00:00	+03 00:00:00

1 row selected.

All of these express
the interval three days



ORA-00923: FROM keyword not found where expected

- This is a generic error, raised in many situations
- But, one possibility with Intervals is...

INTERVAL 3 DAY



Results in ORA-00923

Solution

INTERVAL '3' DAY



INTERVAL ERROR

ORA-30089: missing or invalid <datetime field>

INTERVAL '03-04-05' YEAR TO DAY

Results in ORA-30089

Solution

You cannot specify an interval that spans between months and days.

The two valid ranges for interval units are:

YEAR >> MONTH

DAYS >> SECOND



ORA-01867: the interval is invalid

- Don't you love unhelpful error messages?

INTERVAL '03:04:05' HOUR TO MINUTE

Results in ORA-01867

Solution

The unit specification
does not match the
literal

INTERVAL '03:04:05' HOUR TO SECOND



INTERVAL ERROR

ORA-01873: the leading precision of the interval is too small

- Meaning: value specified exceeds the default precision specification for the interval component
- Solution, specify a higher precision

I N T E R V A L ' 300 ' D A Y

Results in ORA-01873

Solution

I N T E R V A L ' 300 ' D A Y (3)

Unit Component	Default Precision
DAY	2
HOUR	3
MINUTE	5
SECOND	7



RANGE EXAMPLE



```

SELECT empno, ename, hiredate
, COUNT(*) OVER ( ORDER BY hiredate
                  RANGE BETWEEN INTERVAL '6' MONTH PRECEDING
                  AND INTERVAL '6' MONTH FOLLOWING ) AS six_mo
, COUNT(*) OVER ( ORDER BY hiredate
                  RANGE BETWEEN CURRENT ROW
                  AND INTERVAL '6' MONTH FOLLOWING ) AS six_mo_after
FROM scott.emp;

```

How many people were hired within six months of this person?

How many people were hired six months after this person?

EMPNO	ENAME	HI REDATE	SIX_MO	SIX_MO_AFTER
7369	SMI TH	17-DEC-80	6	6
7499	ALLEN	20-FEB-81	6	5
7521	WARD	22-FEB-81	6	4
7566	JONES	02-APR-81	8	5
7698	BLAKE	01-MAY-81	8	4
7782	CLARK	09-JUN-81	11	6
7844	TURNER	08-SEP-81	9	6
7654	MARTIN	28-SEP-81	9	5
7839	KING	17-NOV-81	7	4
7900	JAMES	03-DEC-81	7	3
7902	FORD	03-DEC-81	7	3
7934	MI LLER	23-JAN-82	6	1
7788	SCOTT	09-DEC-82	2	2
7876	ADAMS	12-JAN-83	2	1

14 rows selected.



THREE LEVELS OF CONDITIONS

```
SELECT ename, job, sal
, COUNT(*) OVER ( PARTITION BY job
                  ORDER BY sal
                  RANGE BETWEEN 200 PRECEDING
                  AND 200 FOLLOWING ) emps_200_sal
FROM scott.emp
WHERE ename < 'M'
ORDER BY deptno, empno;
```

ENAME	JOB	SAL	EMPS_200_SAL
CLARK	MANAGER	2450	1
KING	PRESIDENT	5000	1
JONES	MANAGER	2975	2
ADAMS	CLERK	1100	2
FORD	ANALYST	3000	1
ALLEN	SALESMAN	1600	1
BLAKE	MANAGER	2850	2
JAMES	CLERK	950	2

8 rows selected.

- 1) WHERE ename < 'M'
- 2) PARTITION BY job
- 3) RANGE BETWEEN ...



ANOTHER RANGE LIMITATION

- Only one sort key allowed when windowing with RANGE
 - Because range depends on the ORDER BY to derive the field

```
SELECT ename, sal  
, COUNT(*) OVER ( ORDER BY sal, comm  
                    RANGE BETWEEN 200 PRECEDING  
                    AND 200 FOLLOWING ) emps_200_sal  
FROM scott.emp;
```

```
, COUNT(*) OVER ( ORDER BY sal, comm  
                  *
```

ERROR at line 2:

ORA-30486: invalid window aggregation group in the window specification

What the error message means: You cannot specify two sort fields with a RANGE window

EXCEPT...
in two (common)
clauses:

RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING



RANGE EXAMPLE

```
SELECT empno, ename, hiredate
, LAG (ename) OVER ( ORDER BY hiredate
                     RANGE BETWEEN CURRENT ROW
                     AND INTERVAL '6' MONTH FOLLOWING ) AS nexthire
FROM scott.emp;
```

RANGE BETWEEN CURRENT ROW

*

ERROR at line 3:

ORA-00907: missing right parenthesis

What happened?

You cannot use LAG or LEAD with a RANGE window

WHY?

A range sort does not specify individual records.
Consider the two records with the same hiredate

EMPNO	ENAME	HI REDATE
7369	SMITH	17-DEC-80
7499	ALLEN	20-FEB-81
7521	WARD	22-FEB-81
7566	JONES	02-APR-81
7698	BLAKE	01-MAY-81
7782	CLARK	09-JUN-81
7844	TURNER	08-SEP-81
7654	MARTIN	28-SEP-81
7839	KING	17-NOV-81
7900	JAMES	03-DEC-81
7902	FORD	03-DEC-81
7934	MILLER	23-JAN-82
7788	SCOTT	09-DEC-82
7876	ADAMS	12-JAN-83

14 rows selected.



EXAMPLE



Show date of next order for customer

customer_id	order_date	Order_total	Next Order_Date	Next order_amt
103	3/29/1997	310.00	9/01/1998	13550.00
103	9/01/1998	13550.00	9/13/1999	78.00
103	9/13/1999	78.00	10/02/1999	6653.40
103	10/02/1999	6653.40		
105	3/20/1999	1926.60	8/31/1999	22150.00
105	8/31/1999	22150.10	1/08/2000	7826.00
105	1/08/2000	7826.00	1/26/2000	29473.80
105	1/26/2000	29473.80		

- Do we need a PARTITION BY?
 - If so, which field(s)?
- Do we need an ORDER BY?
 - If yes, which field(s)?
- How will this be windowed?
 - RANGE or ROWS?



NEXT ORDER

customer_id	order_date	Order_total	Next Order_Date	Next order_amt
103	3/29/1997	310.00	9/01/1998	13550.00
103	9/01/1998	13550.00	9/13/1999	78.00
103	9/13/1999	78.00	10/02/1999	6653.40
103	10/02/1999	6653.40		
105	3/20/1999	1926.60	8/31/1999	22150.00
105	8/31/1999	22150.10	1/08/2000	7826.00
105	1/08/2000	7826.00	1/26/2000	29473.80
105	1/26/2000	29473.80		

Show date of next order

- Here are five ways:
 - LEAD
 - LAG with reverse order
 - MAX with ROWS current to 1 following
 - MIN with ROWS 1 to unbounded following
 - MIN or MAX with window only on 1 row following



DATE OF NEXT ORDER

```
SELECT customer_id
       , TRUNC ( order_date ) AS order_date
       , order_total
       , LEAD ( TRUNC ( order_date ) ) OVER
         ( PARTITION BY customer_id ORDER BY order_date ) AS next_order_date
FROM oe.orders
WHERE customer_id IN (103, 105)
ORDER BY 1, 2;
```

CUSTOMER_ID	ORDER_DAT	ORDER_TOTAL	NEXT_ORDE
103	29-MAR-97	310	01-SEP-98
103	01-SEP-98	13550	13-SEP-99
103	13-SEP-99	78	02-OCT-99
103	02-OCT-99	6653.4	
105	20-MAR-99	1926.6	31-AUG-99
105	31-AUG-99	22150.1	08-JAN-00
105	08-JAN-00	7826	26-JAN-00
105	26-JAN-00	29473.8	

8 rows selected.

```
LAG ( TRUNC ( order_date ) ) OVER ( PARTITION BY customer_id
                                   ORDER BY order_date DESC )
```

```
MAX ( TRUNC ( order_date ) ) OVER ( PARTITION BY customer_id
                                   ORDER BY order_date
                                   ROWS BETWEEN CURRENT ROW AND 1 FOLLOWING )
```

```
MIN ( TRUNC ( order_date ) ) OVER ( PARTITION BY customer_id
                                   ORDER BY order_date
                                   ROWS BETWEEN 1 FOLLOWING AND UNBOUNDED FOLLOWING )
```

```
MIN ( TRUNC ( order_date ) ) OVER ( PARTITION BY customer_id
                                   ORDER BY order_date
                                   ROWS BETWEEN 1 FOLLOWING AND 1 FOLLOWING )
```



AMOUNT OF NEXT ORDER

```
SELECT customer_id
       , TRUNC ( order_date ) AS order_date
       , order_total
       , LEAD ( order_total ) OVER ( PARTITION BY customer_id
                                     ORDER BY order_date ) AS next_order_total
FROM oe.orders
WHERE customer_id IN (103, 105)
ORDER BY 1, 2;
```

```
LAG ( order_total ) OVER ( PARTITION BY customer_id
                           ORDER BY order_date DESC)
```

CUSTOMER_ID	ORDER_DAT	ORDER_TOTAL	NEXT_ORDER_TOTAL
103	29-MAR-97	310	13550
103	01-SEP-98	13550	78
103	13-SEP-99	78	6653.4
103	02-OCT-99	6653.4	
105	20-MAR-99	1926.6	22150.1
105	31-AUG-99	22150.1	7826
105	08-JAN-00	7826	29473.8
105	26-JAN-00	29473.8	

8 rows selected.

```
MAX ( order_total ) OVER ( PARTITION BY customer_id
                           ORDER BY order_date
                           ROWS BETWEEN 1 FOLLOWING AND 1 FOLLOWING)
```



WHY SO MANY?

```
SELECT customer_id
, TRUNC ( order_date ) AS order_date
, order_total
, LEAD ( order_total ) OVER
  ( PARTITION BY customer_id ORDER BY order_date ) AS next_order
, order_total + LEAD ( order_total ) OVER
  ( PARTITION BY customer_id ORDER BY order_date ) AS this_plus_next
, SUM ( order_total ) OVER
  ( PARTITION BY customer_id ORDER BY order_date
    ROWS BETWEEN CURRENT ROW AND 1 FOLLOWING ) AS sum_this_next
FROM oe.orders
WHERE customer_id IN (103, 105)
ORDER BY 1, 2;
```

CUSTOMER_ID	ORDER_DATE	ORDER_TOTAL	NEXT_ORDER	THIS_PLUS_NEXT	SUM_THIS_NEXT
103	29-MAR-97	310	13550	13860	13860
103	01-SEP-98	13550	78	13628	13628
103	13-SEP-99	78	6653.4	6731.4	6731.4
103	02-OCT-99	6653.4		6653.4	6653.4
105	20-MAR-99	1926.6	22150.1	24076.7	24076.7
105	31-AUG-99	22150.1	7826	29976.1	29976.1
105	08-JAN-00	7826	29473.8	37299.8	37299.8
105	26-JAN-00	29473.8		29473.8	29473.8

8 rows selected.

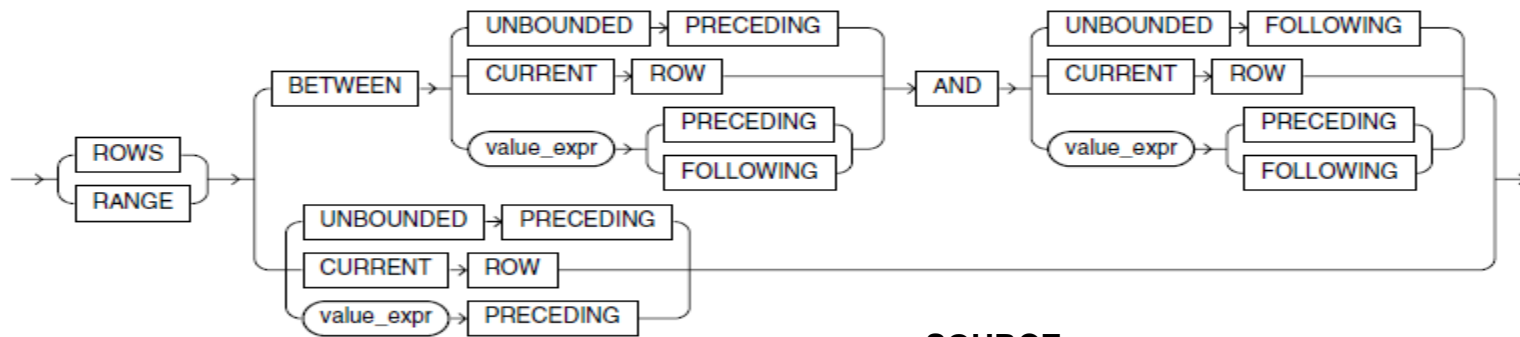
Using LEAD results in NULL at edge of partition.

Using SUM and windowing, avoids nulls.



SYNTAX DIAGRAM

windowing_clause::=



SOURCE:
Oracle Database SQL Language Reference
11g Release 2 (E10592-04)
Page 5-12



Shortcut

- If...
 - You omit BETWEEN and AND
 - And the windowing value is \leq CURRENT ROW
- Then...
 - The second argument is assumed to be CURRENT ROW

ROWS UNBOUNDED PRECEDING



ROWS BETWEEN UNBOUNDED PRECEDING
AND CURRENT ROW

ROWS 10 PRECEDING



ROWS BETWEEN 10 PRECEDING
AND CURRENT ROW

ROWS CURRENT ROW



ROWS BETWEEN CURRENT ROW
AND CURRENT ROW



```
SELECT ename, sal  
       , COUNT(*) OVER ( ORDER BY sal ROWS UNBOUNDED FOLLOWING)  
FROM scott.emp;
```

```
       , COUNT(*) OVER ( ORDER BY sal ROWS UNBOUNDED FOLLOWING)  
                               *
```

```
ERROR at line 2:  
ORA-00905: missing keyword
```



Search

About 15,600 results (0.15 seconds)

Upon encountering ORA-00905, you must correct syntax because there is a missing keyword.



COLLABORATE12
TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

Recap



- Aggregate vs Analytic
- PARTITION BY
- ORDER BY
- Window Clause
 - ROWS
 - RANGE



COLLABORATE12

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY



THANK-YOU!

This is session #740

DAN STOBER

Questions? Comments?
dan.stober@utoug.org

Twitter: [@dstober](https://twitter.com/dstober)
Slides available on [slideshare.net](https://www.slideshare.net)