# Analytic Functions : An Oracle Developer's Best Friend


**Tim Hall**

**Oracle ACE Director**

**Oracle ACE of the Year 2006**

**OakTable Network**

**OCP DBA (7, 8, 8i, 9i, 10g, 11g)**

**OCP Advanced PL/SQL Developer**

**Oracle Database: SQL Certified Expert**

**http://www.oracle-base.com**


**Books**

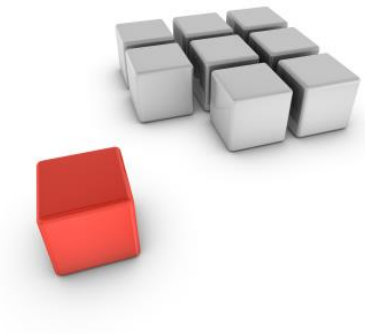
**Oracle PL/SQL Tuning**

**Oracle Job Scheduling**

# What are Analytic Functions?

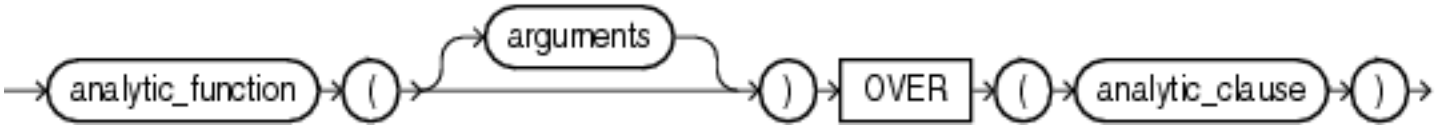
- Analytic Functions, or Windowing Functions, were introduced in Oracle8i.
- They compute aggregate values based on groups of data.
- Unlike aggregate functions, they don't reduce the number of rows returned.
- Analytic functions are processed after the result set is returned, but before the conventional ORDER BY operation.

(average.sql)

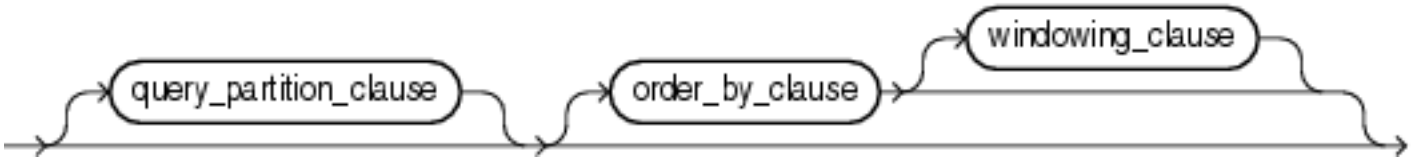


# Analytic Function Syntax

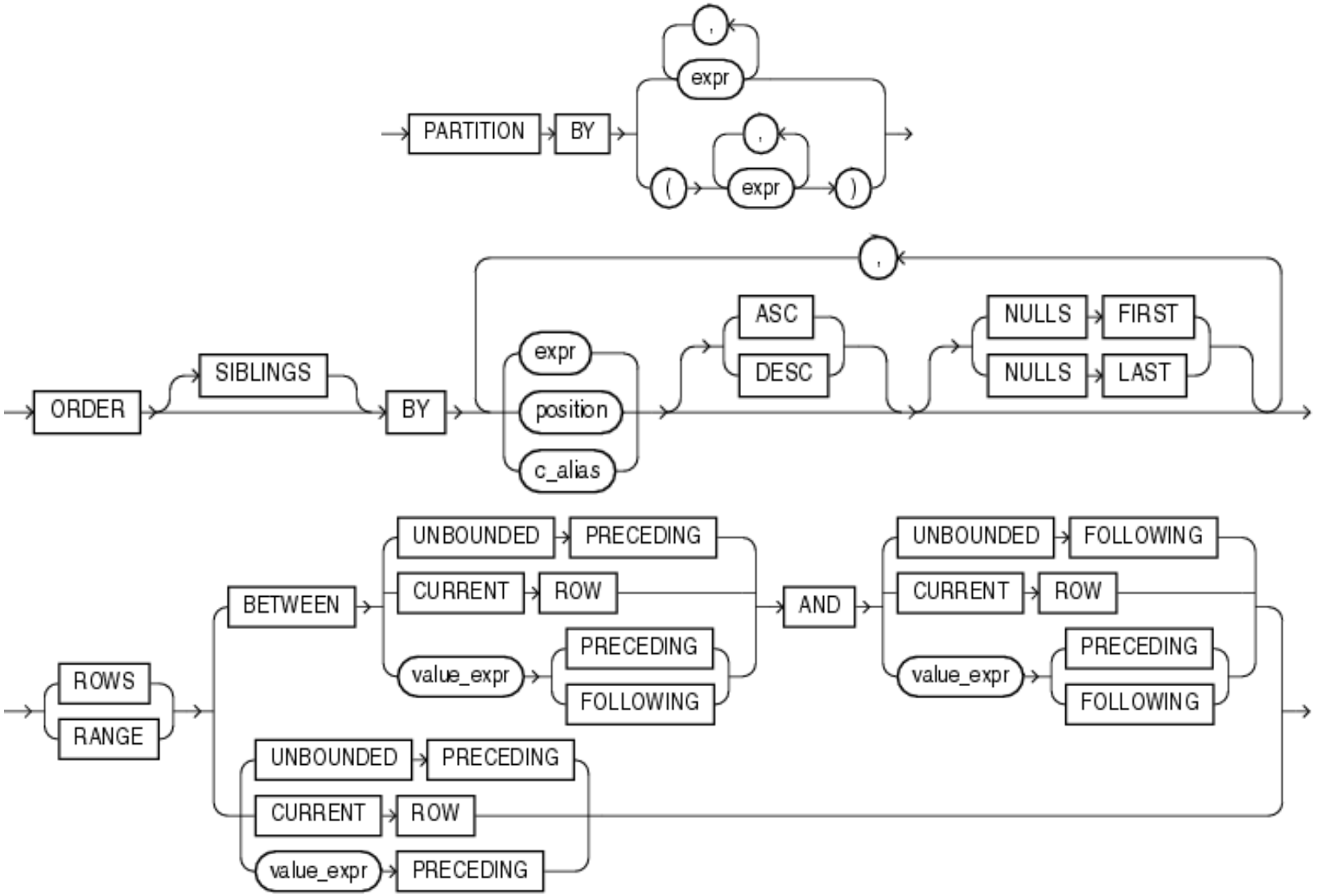
- **There is some variation between individual functions, but the basic syntax is similar.**




- **The analytic clause is broken down as follows.**

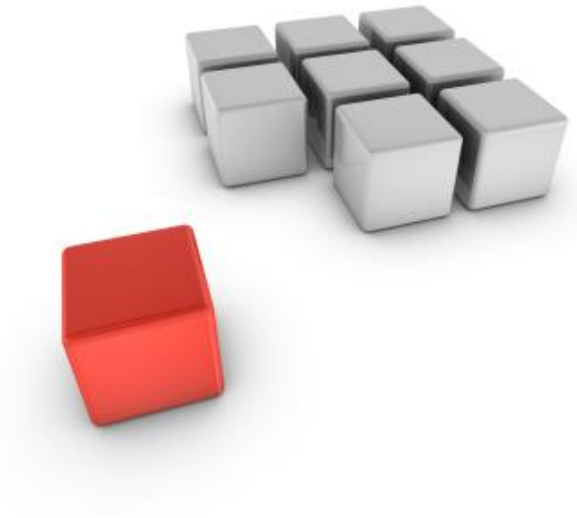


# Analytic Function Syntax



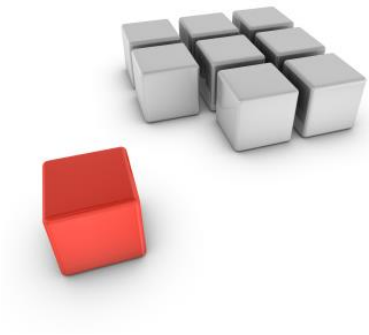
# query_partition_clause

- Divides data into partitions or groups, similar to GROUP BY.
- The operation of the analytic function is restricted to the boundary imposed by these partition.



# query_partition_clause

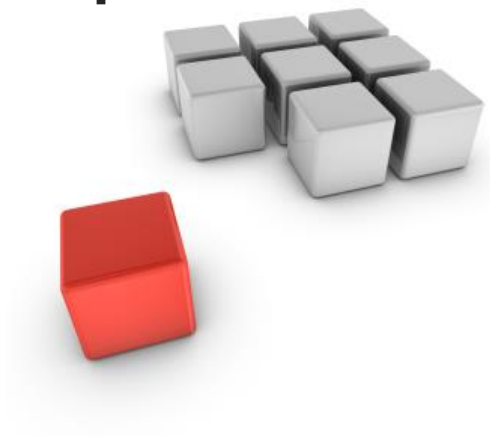
| EMPNO | DEPTNO | SAL |
|---|---|---|
| 7934 | 10 | 1300 |
| 7782 | 10 | 2450 |
| 7839 | 10 | 5000 |
| 7369 | 20 | 800 |
| 7876 | 20 | 1100 |
| 7566 | 20 | 2975 |
| 7788 | 20 | 3000 |
| 7902 | 20 | 3000 |
| 7900 | 30 | 950 |
| 7654 | 30 | 1250 |
| 7521 | 30 | 1250 |
| 7844 | 30 | 1500 |
| 7499 | 30 | 1600 |
| 7698 | 30 | 2850 |



# query_partition_clause

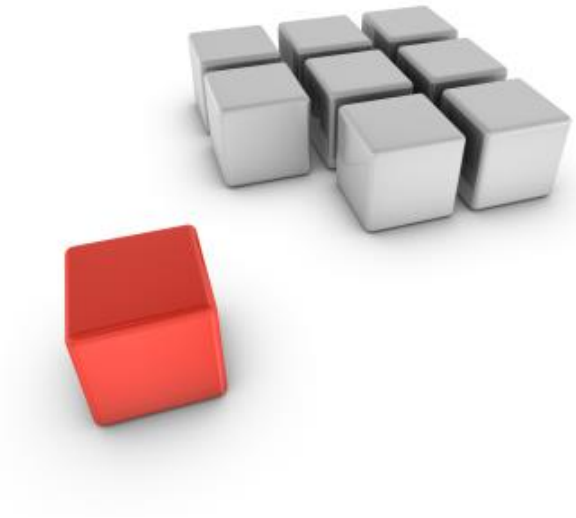
- Divides data into partitions or groups, similar to GROUP BY.
- The operation of the analytic function is restricted to the boundary imposed by these partition.
- If the query_partition_clause is empty, the partition is assumed to be the whole result set.

(query_partition_clause.sql)



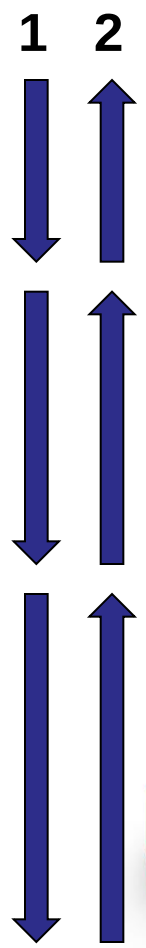
# order_by_clause

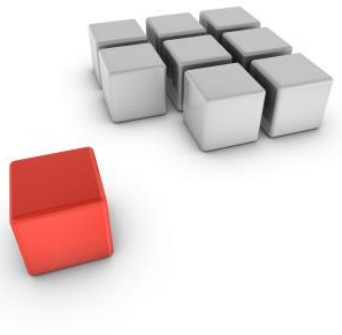
- **Orders rows, or siblings, within a partition.**



# order_by_clause

| EMPNO | DEPTNO | SAL |
|---|---|---|
| 7934 | 10 | 1300 |
| 7782 | 10 | 2450 |
| 7839 | 10 | 5000 |
| 7369 | 20 | 800 |
| 7876 | 20 | 1100 |
| 7566 | 20 | 2975 |
| 7788 | 20 | 3000 |
| 7902 | 20 | 3000 |
| 7900 | 30 | 950 |
| 7654 | 30 | 1250 |
| 7521 | 30 | 1250 |
| 7844 | 30 | 1500 |
| 7499 | 30 | 1600 |
| 7698 | 30 | 2850 |

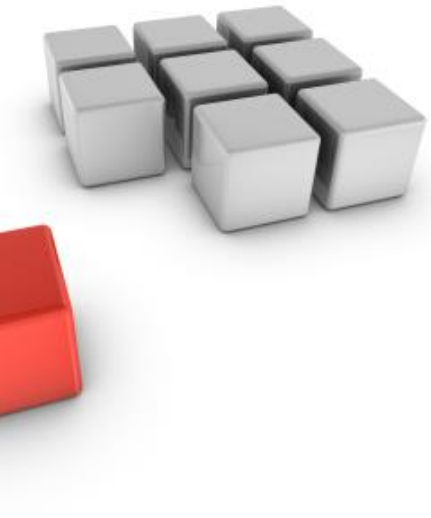


# order_by_clause

- Orders rows, or siblings, within a partition.
- Necessary with order-sensitive analytic functions.

  (order_by_clause.sql)
- Ordering NULLs:
  - ASC is default.
  - When ASC is used, NULLS LAST is default.
  - When DESC is used, NULLS FIRST is default.
- order_by_clause affects processing order but may not affect display order consistently. MUST use an ORDER BY!



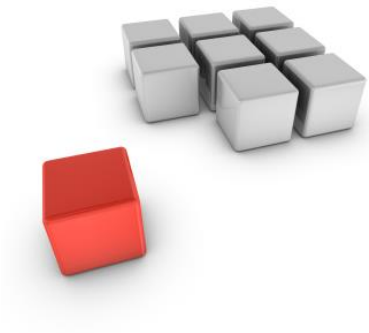
# windowing_clause

- **Extension of the order_by_clause, giving finer control of the window of operation for some functions.**

```
RANGE BETWEEN start_point AND end_point
ROWS BETWEEN start_point AND end_point
```

# windowing_clause

| EMPNO | DEPTNO | SAL |
|---|---|---|
| 7934 | 10 | 1300 |
| 7782 | 10 | 2450 |
| 7839 | 10 | 5000 |
| 7369 | 20 | 800 |
| 7876 | 20 | 1100 |
| 7566 | 20 | 2975 |
| 7788 | 20 | 3000 |
| 7902 | 20 | 3000 |
| 7900 | 30 | 950 |
| 7654 | 30 | 1250 |
| 7521 | 30 | 1250 |
| 7844 | 30 | 1500 |
| 7499 | 30 | 1600 |
| 7698 | 30 | 2850 |



# windowing_clause

- Extension of the order_by_clause, giving finer control of the window of operation for some functions.

```
RANGE BETWEEN start_point AND end_point
ROWS BETWEEN start_point AND end_point
```

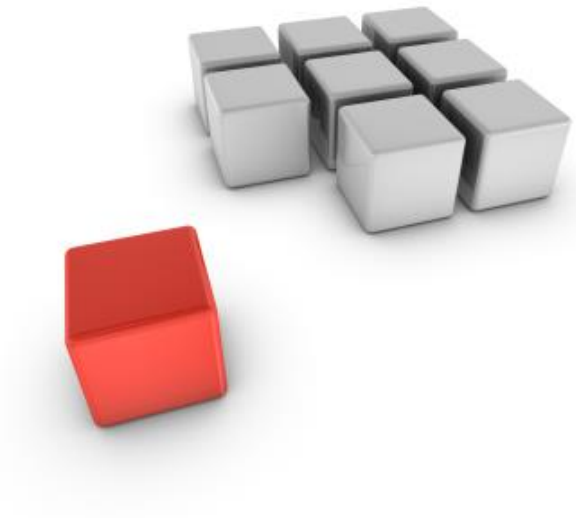
- Start and end points.
  - UNBOUNDED PRECEDING : Window starts at first row of partition.
  - UNBOUNDED FOLLOWING : Window ends at last row of partition
  - CURRENT ROW : Window starts or ends at the current row
  - value_expr PRECEDING : Physical/logical offset before current row.
  - value_expr FOLLOWING : Physical/logical offset after current row.

- Default is “RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW”
(windowing_clause.sql)

# Analytic Functions

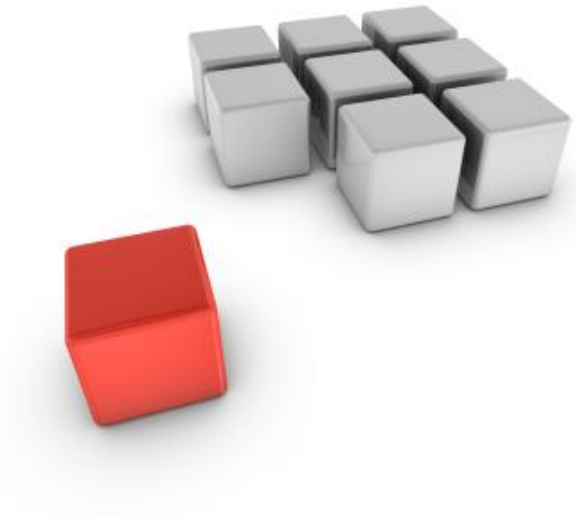
- AVG *
- CORR *
- COUNT *
- COVAR_POP *
- COVAR_SAMP *
- CUME_DIST
- DENSE_RANK
- FIRST
- FIRST_VALUE *
- LAG
- LAST
- LAST_VALUE *
- LEAD
- LISTAGG
- MAX *
- MIN *
- NTH_VALUE *
- NTILE
- PERCENT_RANK
- PERCENTILE_CONT
- PERCENTILE_DISC
- RANK
- RATIO_TO_REPORT
- REGR_ (Linear Regression) Functions *
- ROW_NUMBER
- STDDEV *
- STDDEV_POP *
- STDDEV_SAMP *
- SUM *
- VAR_POP *
- VAR_SAMP *
- VARIANCE *

* = full syntax



# Analytic Functions : Examples

- ranking.sql
- first_last.sql
- listagg.sql
- lag_lead.sql
- first_value.sql
- last_value.sql
- row_number.sql

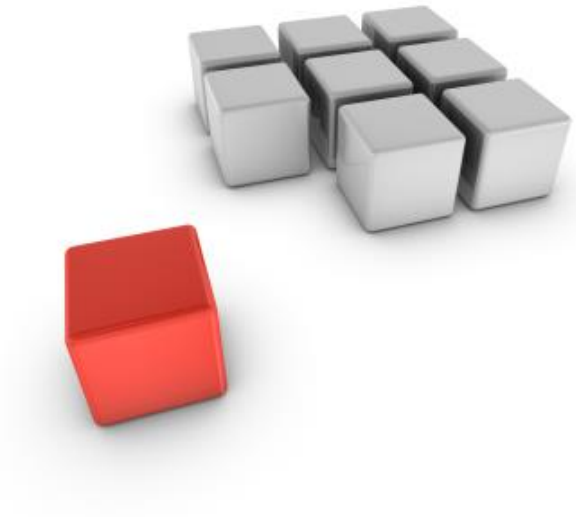


# Top-N Queries (12c)

- Oracle Database 12c finally has a row limiting clause.
- Makes Top-N Queries and resultset paging easy.

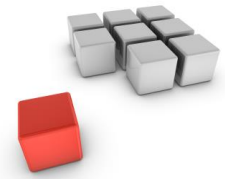
```
SELECT empno, sal
FROM   scott.emp
ORDER BY sal DESC
FETCH FIRST 5 ROWS ONLY;
```

- How is this implemented?



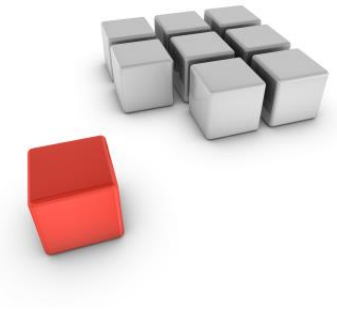
# Top-N Queries (12c)

```
CONN / AS SYSDBA

ALTER SESSION SET EVENTS '10053 trace name context forever, level 1';

SELECT empno, sal
FROM   scott.emp
ORDER BY sal DESC
FETCH FIRST 5 ROWS ONLY;

ALTER SESSION SET EVENTS '10053 trace name context off';


Final query after transformations:******* UNPARSED QUERY IS *******

SELECT "from$_subquery$_002"."EMPNO" "EMPNO",
       "from$_subquery$_002"."SAL" "SAL"
FROM  (SELECT
         "EMP"."EMPNO" "EMPNO",
         "EMP"."SAL" "SAL",
         "EMP"."SAL" "rowlimit_$_  0",
         ROW_NUMBER() OVER ( ORDER BY "EMP"."SAL" DESC ) "rowlimit_$$_rownumber"
       FROM "SCOTT"."EMP" "EMP") "from$_subquery$_002"
WHERE "from$_subquery$_002"."rowlimit_$$_rownumber"<=5
ORDER BY "from$_subquery$_002"."rowlimit_$_  0" DESC
```



# Summary

- **Analytic Functions make complex post-query processing simple.**
- **Typically much faster than performing similar operations using procedural languages.**
- **Very flexible.**
- **Takes time to get used to the analytic clause, so keep playing!**



# The End…

- **Slides and Demos:**

  **http://www.oracle-base.com/workshops**

- **Questions?**