No Bikinis?
Working with SQL's Model

Presented to:

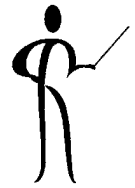
RMOUG Training Days

February 9, 2005

John Jay King
King Training Resources
john@kingtraining.com

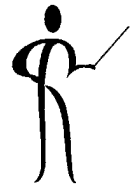
Download this paper and code examples from:
http://www.kingtraining.com

Session Objectives



- Learn how to use the Oracle 10g Model clause
- Be ready to use various options of Model to represent query data in a “spreadsheet”

10g Model Clause

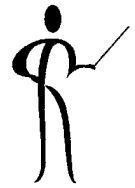


- The SQL MODEL clause is a powerful extension of the SELECT statement
- MODEL provides the ability to present the output of a SELECT in the form of multi-dimensional arrays (like a spreadsheet) and apply formulas to the array (cell) values
- The Model clause defines a multidimensional array by mapping the columns of a query into three groups: partitioning, dimension, and measure columns
  - Partitions
  - Dimensions
  - Measures

Partitions, Dimensions, Measures

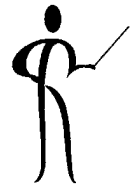
- Partitions define logical blocks of the result set in a way similar to the partitions of the analytical functions; each partition is viewed by the formulas as an independent array – model rules are applied the cells of each partition
- Dimensions identify each measure cell within a partition; each column identifies characteristics such as date, region and product name
- Measures are similar to the measures of a fact table in a star schema, they normally contain numeric values such as sales units or cost; each cell is accessed within its partition by specifying its full combination of dimensions

10g Model Syntax



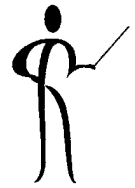
```
MODEL
[<global reference options>]
[<reference models>]
[MAIN <main-name>]
[PARTITION BY (<cols>)]
DIMENSION BY (<cols>)
MEASURES (<cols>)
[<reference options>]
[RULES] <rule options>
(<rule>, <rule>,.., <rule>)
<global reference options> ::= <reference options> <ret-opt>
<ret-opt> ::= RETURN {ALL|UPDATED} ROWS
<reference options> ::=
[IGNORE NAV | [KEEP NAV]
[UNIQUE DIMENSION | UNIQUE SINGLE REFERENCE]
<rule options> ::=
[UPSERT | UPDATE]
[AUTOMATIC ORDER | SEQUENTIAL ORDER]
[ITERATE (<number>) [UNTIL <condition>]]
<reference models> ::= REFERENCE ON <ref-name> ON (<query>)
DIMENSION BY (<cols>) MEASURES (<cols>) <reference options>
```

Oracle Goodies



- The Oracle Database Data Warehousing Guide provides many explanations and examples of the model clause and it use
- The examples in these notes were based upon the Oracle-supplied “SH” schema’s data
- The example on the next two pages uses a “sales_view” definition from the Oracle manual
- On a later page is a Materialized View definition used for the code examples in this paper

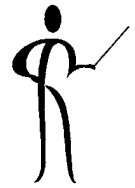
10g Model Example



```
SELECT SUBSTR(country,1,20) country, SUBSTR(prod,1,15) prod, year, sales
   FROM sales_view
   WHERE country IN ('Canada','Germany')
   MODEL RETURN UPDATED ROWS
         PARTITION BY (country)
         DIMENSION BY (prod, year)
         MEASURES (sale sales)
         RULES (
                  sales['Zoop Tube', 2002] = sales['Zoop Tube', 2001] +
                                             sales['Zoop Tube', 2000],
                  sales['Hula Twirl', 2002] = sales['Hula Twirl', 2001],
                  sales['HulaZoop Pkg', 2002] = sales['Zoop Tube', 2002] +
                                                sales['Hula Twirl', 2002])
   ORDER BY country, prod, year;

COUNTRY                PROD            YEAR        SALES
---------------------  --------------  ----------  ----------
Canada                 HulaZoop Pkg    2002         92613.16
Canada                 Zoop Tube       2002          9299.08
Canada                 Hula Twirl      2002         83314.08
Germany                HulaZoop Pkg    2002        103816.6
Germany                Zoop Tube       2002         11631.13
Germany                Hula Twirl      2002         92185.47
```

10g Model Example Explained



- The statement on the preceding page calculates sales values for two products and defines sales for a new product based upon the other two products
  - Statement partitions data by country, so formulas are applied to one country at a time, sales fact data ends with 2001, any rules defining values for 2002 or later will insert new cells
  - First rule defines sales of “Zoop Tube” game in 2002 as the sum of its sales in 2000 and 2001
  - The second rule defines sales for “Hula Twirl” in 2002 to be the same value they were for 2001
  - Third rule defines "HulaZoop Pkg" that is the sum of the Zoop Tube and Hula Twirl values for 2002 -- the rules for Zoop Tube and Hula Twirl must be executed before the HulaZoop Pkg rule

Creating the MView

```
CREATE materialized VIEW sales_mview AS
   SELECT  substr(country_name,1,20) country
          ,substr(prod_name,1,15)    product
          ,calendar_year             year
          ,SUM(amount_sold)          tot_amt
          ,SUM(quantity_sold)        tot_qty
          ,COUNT(amount_sold)        tot_sales
        FROM sh.sales join sh.times
                        on sales.time_id = times.time_id
                      join sh.products
                        on sales.prod_id = products.prod_id
                      join sh.customers
                        on sales.cust_id = customers.cust_id
                      join sh.countries
                        on customers.country_id = countries.country_id
   GROUP BY country_name
                ,prod_name
                ,calendar_year
         ORDER BY country
                ,product
                ,year
```

Query Existing Data

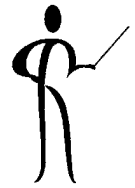
- The query below retrieves data from two products and two countries for all years in the existing data

```
select country
        ,product
        ,year
        ,round(tot_sales,0) sales
  from sales_mview
  where country in ('Australia','Canada')
    and product in ('Mouse Pad','Deluxe Mouse')
  order by country
          ,product
          ,year
```

Existing Data

COUNTRY	PRODUCT	YEAR	SALES
Australia	Deluxe Mouse	1998	86
Australia	Deluxe Mouse	1999	140
Australia	Deluxe Mouse	2000	78
Australia	Deluxe Mouse	2001	211
Australia	Mouse Pad	1998	195
Australia	Mouse Pad	1999	311
Australia	Mouse Pad	2000	264
Australia	Mouse Pad	2001	332
Canada	Deluxe Mouse	1998	65
Canada	Deluxe Mouse	1999	109
Canada	Deluxe Mouse	2000	38
Canada	Deluxe Mouse	2001	61
Canada	Mouse Pad	1998	93
Canada	Mouse Pad	1999	174
Canada	Mouse Pad	2000	144
Canada	Mouse Pad	2001	186

Predicting the Future



- The existing data goes through 2001, what if we want to project future sales?
- The model clause allows the creation of a “spreadsheet” layout where each result represents a “cell”
- The rules sub-clause allows us to establish a set of rules for treating cell-values and even predict future values

First Model

```
model return all rows
   partition by (country)
   dimension by (product,year)
   measures (tot_sales sales)
   rules (
    sales['Mouse Pad',2002] =
                sales['Mouse Pad',2001] * 1.1
   ,sales['Deluxe Mouse',2002] =
                sales['Deluxe Mouse',2001] * 1.3
   )
```

- Return all rows (both existing and new)
- Group (partition) rows by country
- Define product and year as dimension values
- Define tot_sales as the value of each cell, rename “sales”
- Use rules to set sales for 2002 (a new year) to 2001’s value times some multiplier (I made them up…)

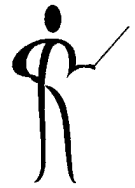
RETURN ALL ROWS

```
select country
      ,product
      ,year
      ,round(sales,0) sales
from sales_mview
where country in ('Australia','Canada')
  and product in ('Mouse Pad','Deluxe Mouse')
model return all rows
   partition by (country)
   dimension by (product,year)
   measures (tot_sales sales)
   rules (
    sales['Mouse Pad',2002] =
         sales['Mouse Pad',2001] * 1.1
   ,sales['Deluxe Mouse',2002] =
         sales['Deluxe Mouse',2001] * 1.3
   )
order by country
        ,product
        ,year
```

ALL ROWS Model Results

COUNTRY	PRODUCT	YEAR	SALES
Australia	Deluxe Mouse	1998	86
Australia	Deluxe Mouse	1999	140
Australia	Deluxe Mouse	2000	78
Australia	Deluxe Mouse	2001	211
Australia	Deluxe Mouse	2002	274
Australia	Mouse Pad	1998	195
Australia	Mouse Pad	1999	311
Australia	Mouse Pad	2000	264
Australia	Mouse Pad	2001	332
Australia	Mouse Pad	2002	365
Canada	Deluxe Mouse	1998	65
Canada	Deluxe Mouse	1999	109
Canada	Deluxe Mouse	2000	38
Canada	Deluxe Mouse	2001	61
Canada	Deluxe Mouse	2002	79
Canada	Mouse Pad	1998	93
Canada	Mouse Pad	1999	174
Canada	Mouse Pad	2000	144
Canada	Mouse Pad	2001	186
Canada	Mouse Pad	2002	205

UPDATED ROWS only



```
select country
      ,product
      ,year
      ,round(units_sold,0) units_sold
from sales_mview
where country in ('Australia','Canada')
  and product in ('Mouse Pad','Deluxe Mouse')
model return updated rows
  partition by (country)
  dimension by (product,year)
  measures (tot_sales units_sold)
  rules (
    units_sold['Mouse Pad',2002] =
                units_sold['Mouse Pad',2001] * 1.1
   ,units_sold['Deluxe Mouse',2002] =
                units_sold['Deluxe Mouse',2001] * 1.3
   )
order by country
        ,product
        ,year
```

Updated Rows

COUNTRY	PRODUCT	YEAR	UNITS_SOLD
Australia	Deluxe Mouse	2002	274
Australia	Mouse Pad	2002	365
Canada	Deluxe Mouse	2002	79
Canada	Mouse Pad	2002	205

Model-Related SQL Functions

- CV — Current value of dimension in model clause
- ITERATION_NUMBER — Returns iteration number in model clause rules
- PRESENTNNV — Present Value of cell in model clause (nulls converted)
- PRESENTV — Present Value of cell in model clause
- PREVIOUS — Returns cell value at beginning of model clause iteration

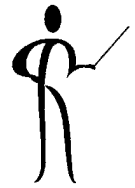
Projecting into the Future

- The model clause includes a special FOR construct allowing the modification and/or creation of many new rows (called UPSERT)
  - Values may range as desired
  - Increment and/or decrement value
  - Use cv() function to use a cell's current value
  - UPSERT is limited to 10,000 rows

```
FOR  dimension_value
          FROM lowval TO hival

          INCREMENT | DECREMENT incrval
```

Projection SQL



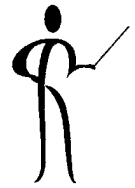
```
select country,product,year,round(units_sold,0) units_sold
from sales_mview
where country in ('Australia','Canada')
  and product in ('Mouse Pad','Deluxe Mouse')
model return updated rows
  partition by (country)
  dimension by (product,year)
  measures (tot_sales units_sold)
  rules (
    units_sold['Mouse Pad',
        for year from 2001 to 2005 increment 1]
         = units_sold['Mouse Pad',cv(year)-1] * 1.1
   ,units_sold['Deluxe Mouse',
        for year from 2001 to 2005 increment 1]
         = units_sold['Deluxe Mouse',cv(year)-1] * 1.3
   )
order by country
        ,product
        ,year
```

King Training Resources

Projection Results

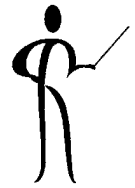
COUNTRY	PRODUCT	YEAR	UNITS_SOLD
Australia	Deluxe Mouse	2001	101
Australia	Deluxe Mouse	2002	132
Australia	Deluxe Mouse	2003	171
Australia	Deluxe Mouse	2004	223
Australia	Deluxe Mouse	2005	290
Australia	Mouse Pad	2001	290
Australia	Mouse Pad	2002	319
Australia	Mouse Pad	2003	351
Australia	Mouse Pad	2004	387
Australia	Mouse Pad	2005	425
Canada	Deluxe Mouse	2001	49
Canada	Deluxe Mouse	2002	64
Canada	Deluxe Mouse	2003	83
Canada	Deluxe Mouse	2004	109
Canada	Deluxe Mouse	2005	141
Canada	Mouse Pad	2001	158
Canada	Mouse Pad	2002	174
Canada	Mouse Pad	2003	192
Canada	Mouse Pad	2004	211
Canada	Mouse Pad	2005	232

Additional Rules



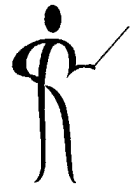
- SEQUENTIAL ORDER rules are defined in the order they appear in the rules sub-clause
- AUTOMATIC ORDER rules are considered according to dependencies
- IGNORE NAV treats missing values as:
  - 0 for numeric data
  - Empty string for character data
  - '01-JAN-2001' for date data
  - NULL for other data types
- KEEP NAV treats nulls normally
- UNIQUE DIMENSION (default), PARTITION BY and DIMENSION BY columns must uniquely identify each and every cell
- UNIQUE SINGLE REFERENCE, PARTITION BY and DIMENSION BY uniquely identify single point references on the right-hand side of the rules
- ITERATE (n) iterates rules specified number of times, ITERATION_NUMBER returns current value (starts with 0)

Oracle Documentation



- Oracle9i and Oracle10g
  - Oracle9i SQL Reference
  - Oracle9i PL/SQL User's Guide and Reference
  - Oracle9i Application Developer's Guide - Object-Relational Features
  - Oracle9i Concepts
  - Oracle Database Data Warehousing Guide
- Oracle10g
  - Oracle10g SQL Reference
  - Oracle10g PL/SQL User's Guide and Reference
  - Oracle10g Application Developer's Guide - Object-Relational Features
  - Oracle10g Concepts
  - Oracle Database Data Warehousing Guide
- Lots of papers and examples:
  http://technet.oracle.com
  http://tahiti.oracle.com

Wrapping it all Up



- The Oracle 10g Model clause adds new capabilities to better server the users of our data
- The Model clause's ability to provide the data in a cell-by-cell "spreadsheet" makes output more readable than ever before
- Rules provide the ability to add new cells to the model based upon calculations performed on existing data cells

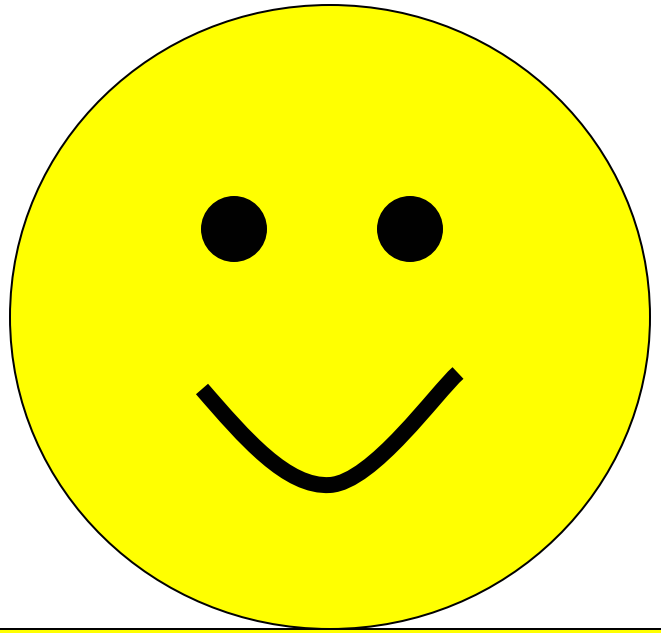
Training Days 2006

Mark your calendar for:

February 15-16, 2005!

King Training Resources

King Training Resources



Thanks for your attention!

To contact the author:

John King

King Training Resources

6341 South Williams Street

Littleton, CO 80121-2627 USA

1.800.252.0652 - 1.303.798.5727

Email: john@kingtraining.com

Paper & Sample Code: www.kingtraining.com