ODTUG KALEIDOSCOPE 2010

Washington, D.C. ■ June 27–July 1, 2010 ■ Washington Marriott Wardman Park Hotel

JOIN ODTUG FOR THE DEVELOPER EVENT OF THE YEAR

TOPICS

- Application Express
- Database Development
- Essbase
- Hyperion Application
- Hardcore Hyperion
- Middle Tier and Client-Side Development
- Oracle Business Intelligence, Data Warehousing, and Hyperion Reporting
- SOA and BPM
- Other

"I go to other conferences hoping to learn something, I go to Kaleidoscope knowing I will."

John Scott, Oracle ACE Director, Sumneva

HIGHLIGHTS

- 170+ Technical Sessions
- 4 Full Day Symposia
- Hands-on Training
- Leading technical experts from around the world
- Networking and Social Events
- Community Service Day

USE THE CODE WEB WHEN REGISTERING FOR A SPECIAL $300 DISCOUNT!!!

www.odtugkaleidoscope.com

New Features in PL/SQL for Oracle 11g

Presented by: John Jay King
King Training Resources - john@kingtraining.com

Download this paper from: http://www.kingtraining.com

Session Objectives

- Learn new Oracle 11g features that are geared to PL/SQL developers
- Use new PL/SQL features in Oracle 11g

Who Am I?

- John King – Partner, King Training Resources
- Providing training to Oracle and IT community for over 20 years – http://www.kingtraining.com
- "Techie" who knows Oracle, SQL, Java, and PL/SQL pretty well (along with many other topics)
- Leader in Service Oriented Architecture (SOA) design and implementation
- Home is Centennial, Colorado – I love it here!
- Member of ODTUG Board of Directors
- Active member of Rocky Mountain Oracle Users Group (RMOUG)

Oracle 11g R1

- Environment changes
- XML enhancements
- New/improved SQL statements
- New features in PL/SQL
- SQL & PL/SQL Results Caches
- Java, JDBC, and SQLJ improvements
- Pro* and OCI enhancements

Oracle 11g R2

- Results Cache Improvements
- New Analytic Functions
- XML Enhancements
- Java Enhancements
- Pro*C/Pro*COBOL Enhancements
- Edition-Based Redefinition (EBR)

Oracle 11g PL/SQL

- Oracle 11g has brought many interesting changes to PL/SQL including:
  - Use of new SQL functionality such as XMLType, BLOB, and Regular Expressions
  - PL/SQL compiler improvements
  - Trigger improvements (compound and follows)
  - PL/SQL result cache
  - continue statement
  - Sequence number ease of use
  - New data types
  - Improved CALL syntax

PL/SQL Result Cache

- PL/SQL allows specification of a result_cache for function/procedure calls
- Add the clause “result_cache” just before the “AS/IS” keyword in the Function and/or Procedure definition
  (Oracle 11g R1 also used now-obsolete “relies_on” clause)
- The results of a call to the Function or Procedure with a specific set of input parameters is stored for later re-use

PL/SQL Result Cache - Code

```
CREATE OR REPLACE FUNCTION RESULT_CACHE_ON
   (in_cust_id sh.customers.cust_id%type,  in_prod_id
   sh.sales.prod_id%type)
RETURN number
RESULT_CACHE -- RELIES_ON (SH.CUSTOMERS, SH.SALES)
authid definer
AS
 sales number(7,0);
BEGIN
select count(*) nbr_sales  into sales
 from sh.customers cust join sh.sales sales
      on cust.cust_id = sales.cust_id
 where cust.cust_id = in_cust_id
  and  prod_id = in_prod_id;
 return sales;
EXCEPTION
  when no_data_found then return 0;
END RESULT_CACHE_ON;
```

PL/SQL Result Cache - Timings

```
 1* select result_cache_on(4977,120) from dual
RESULT_CACHE_ON(4977,120)
-------------------------
                        14
Elapsed: 00:00:00.40

  1* select result_cache_on(4977,120) from dual
RESULT_CACHE_ON(4977,120)
-------------------------
                        14
Elapsed: 00:00:00.00

  1* select result_cache_on(4977,120) from dual
RESULT_CACHE_ON(4977,120)
-------------------------
                        14
Elapsed: 00:00:00.01
```

Compiler Enhancement

- In previous releases, the PL/SQL compiler required a standalone “C” compiler
- Oracle 11g now provides a native compiler for PL/SQL eliminating the need for a separate compiler

```
CREATE…
  COMPILE PLSQL_CODE_TYPE=NATIVE …
CREATE…
  COMPILE PLSQL_CODE_TYPE=INTERPRETED …
```

Compound Triggers

- Compound triggers allow the same code to be shared across timing points (previously accomplished using packages most of the time)
- Compound triggers have unique declaration and code sections for timing point
- All parts of a compound trigger share a common state that is initiated when the triggering statement starts and is destroyed when the triggering statement completes (even if an error occurs)

Compound Trigger Timing

- If multiple compound triggers exist for the same table; they fire together:
  - All before statement code fires first
  - All before row code fires next
  - All after row code fires next
  - All after statement code finishes
- The sequence of trigger execution can be controlled only using the FOLLOWS clause

Compound Trigger Syntax

```
CREATE TRIGGER compound_trigger
  FOR UPDATE OF sal ON emp
    COMPOUND TRIGGER
  -- Global Declaration Section
  BEFORE STATEMENT IS
  BEGIN …
  BEFORE EACH ROW IS
  BEGIN …
  AFTER EACH ROW IS
  BEGIN …
END compound_trigger;
/
```

TRIGGER … FOLLOWS

- Oracle 11g adds the “FOLLOWS” clause to trigger creation allowing control over the sequence of execution when multiple triggers share a timing point
- FOLLOWS indicates the including trigger should happen after the named trigger(s); the named trigger(s) must already exist
- If some triggers use “FOLLOWS” and others do not; only the triggers using “FOLLOWS” are guaranteed to execute in a particular sequence

How FOLLOWS Works

- FOLLOWs only distinguishes between triggers at the same timing point:
  - BEFORE statement
  - BEFORE row
  - AFTER row
  - AFTER statement
  - INSTEAD OF
- In the case of a compound trigger, FOLLOWS applies only to portions of triggers at the same timing point (e.g. if a BEFORE ROW simple trigger names a compound trigger with FOLLOWS the compound trigger must have a BEFORE ROW section and vice versa)

FOLLOWS Syntax

```
CREATE OR REPLACE TRIGGER myTrigger
    BEFORE/AFTER/INSTEAD OF  someEvent
    FOR EACH ROW
    FOLLOWS someschema.otherTrigger
    WHEN (condition=true)
    /* trigger body */
```

- FOLLOWS may specify a list (and designate sequence)

```
FOLLOWS otherTrigger1, otherTrigger2, etc
```

New PL/SQL Datatypes

- Oracle 11g adds three new PL/SQL datatypes: Simple_integer, Simple_float, Simple_double
  - Types use native compilation producing faster arithmetic via direct hardware implementation
  - SIMPLE_INTEGER provides a binary integer that is neither checked for nulls nor overflows
  - SIMPLE_INTEGER values may range from -2147483648 to 2147483647 and is always NOT NULL
  - Likewise, SIMPLE_FLOAT & SIMPLE_DOUBLE do not use null or overflow checks

Example SIMPLE_INTEGER

```
declare
--  mytestvar pls_integer := 2147483645;
  mytestvar simple_integer := 2147483645;
begin
   loop
      mytestvar := mytestvar + 1;
      dbms_output.put_line('Value of mytestvar is now '
                           || mytestvar);
      exit when mytestvar < 10;
  end loop;
end;
Results in:
Value of mytestvar is now 2147483646
Value of mytestvar is now 2147483647
Value of mytestvar is now -2147483648
```

Without SIMPLE_INTEGER

- If the “mytestvar” variable is switched to PLS_INTEGER, an ORA-1426 NUMERIC OVERFLOW exception occurs

```
Error report:
   ORA-01426: numeric overflow
   ORA-06512: at line 7
   01426. 00000 -  "numeric overflow"
   *Cause:    Evaluation of an value expression causes
   an overflow/underflow.
   *Action:   Reduce the operands.
   Value of mytestvar is now 2147483646
   Value of mytestvar is now 2147483647
```

Sequences in PL/SQL

- Sequence values NEXTVAL and CURRVAL may be use in PL/SQL assignment statement

```
myvar := myseq.nextval;
```

CONTINUE

- CONTINUE “iterates” a loop; branching over the rest of the code in the loop and returning to the loop control statement

```
begin
    dbms_output.put_line('Counting down to blastoff!');
    for loopctr in reverse 1 .. ctr loop
      if loopctr in (4,2) then
          continue;
      end if;
      dbms_output.put_line(to_char(loopctr));
    end loop;
    dbms_output.put_line('Blast Off!');
end;
Counting down to blastoff!
6
5
3
1
Blast Off!
```

<-Values “4” and “2” do not appear in the output

REGEXP_COUNT

- REGEXP_COUNT counts the number of times a pattern occurs in a source string

```
select ename,regexp_count(ename,'l',1,'i') from emp;
SMITH    0
ALLEN    2
WARD     0
JONES    0
MARTIN   0
BLAKE    1
/** more rows ***/
MILLER   2
```

  - String expression and/or column to match pattern
  - Regular Expression pattern
  - Beginning position in the source string (default=1)
  - Match parameters (i = case insensitive, c = case sensitive, m = multiple line source delimited by '^' or '$', n = matches '.' newline characters (default no), and x = ignore whitespace characters (default is to match)

CALL with Mixed Parameters

- PL/SQL allows function and procedure parameters to be specified in two ways; by position and by name
- With Oracle 11g SQL, parameter types may now be mixed
- Given the following function:

```
CREATE OR REPLACE
FUNCTION TEST_CALL (inval1 IN NUMBER, inval2 IN
  NUMBER,
    inval3 IN NUMBER) RETURN NUMBER AS
BEGIN
  RETURN inval1 + inval2 + inval3;
END TEST_CALL;
```

- The following calls all now work:

```
test_call(vara,varb,varc)
test_call(inval3=>varc,inval1=>vara,inval2=>varb)
test_call(vara,inval3=>varc,inval2=>varb)
```

Non-Oracle PL/SQL

- Microsoft .NET and Visual Studio .NET (Visual Studio .NET 2008, 2005, & 2003)
  - PL/SQL Debugging in Visual Studio .NET
  - Designer and integration using Data Windows via Visual Studio .NET DDEX
  - Oracle Data Provider for .NET (ODP.NET)
  - .NET stored procedures (Oracle 11g on Windows)

Wrapping it all Up

- Oracle 11g adds significant new functionality to the already robust PL/SQL
- Developers will obtain better performance through the incorporation of improved PL/SQL compiler and result caching
- Developers will find development easier due to the improvements in syntax, datatypes, and triggers provided by Oracle 11g

ODTUG KALEIDOSCOPE 2010

Washington, D.C. ■ June 27–July 1, 2010 ■ Washington Marriott Wardman Park Hotel

ANNOUNCING ODTUG KALEIDOSCOPE 2010 IN WASHINGTON, D.C.

TOPICS

- Application Express
- Database Development
- Essbase
- Hyperion Application
- Hardcore Hyperion
- Middle Tier and Client-Side Development
- Oracle Business Intelligence, Data Warehousing, and Hyperion Reporting
- SOA and BPM
- Other

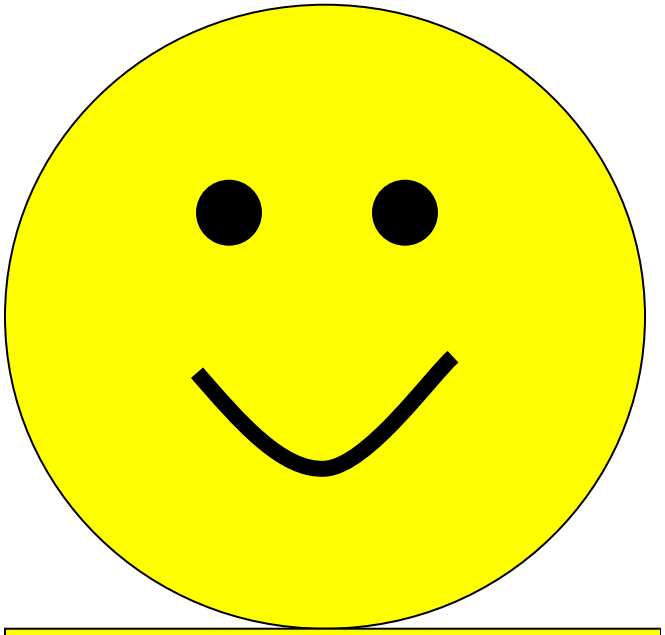
Kaleidoscope has it all - more than 170 technical sessions, day-long symposia, hands-on training, chats with participants and speakers, and even a community service project.

Don't take our word for it, here's what one participant said about Kaleidoscope 2009:

"Before the conference, I try to come up with a list of questions that have been bugging me all year, and as usual, I leave with my questions answered. The conference is a great opportunity to ask some tough questions to some great experts."

www.odtugkaleidoscope.com

New Features in PL/SQL for Oracle 11g



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

Today's slides and examples are on the web:

http://www.kingtraining.com

http://www.odtug.com

ODTUG KALEIDOSCOPE 2010

Washington, D.C. ■ June 27–July 1, 2010 ■ Washington Marriott Wardman Park Hotel

JOIN ODTUG FOR THE DEVELOPER EVENT OF THE YEAR

TOPICS

- Application Express
- Database Development
- Essbase
- Hyperion Application
- Hardcore Hyperion
- Middle Tier and Client-Side Development
- Oracle Business Intelligence, Data Warehousing, and Hyperion Reporting
- SOA and BPM
- Other

"I go to other conferences hoping to learn something, I go to Kaleidoscope knowing I will."

John Scott, Oracle ACE Director, Sumneva

HIGHLIGHTS

- 170+ Technical Sessions
- 4 Full Day Symposia
- Hands-on Training
- Leading technical experts from around the world
- Networking and Social Events
- Community Service Day

USE THE CODE WEB WHEN REGISTERING FOR A SPECIAL $300 DISCOUNT!!!

www.odtugkaleidoscope.com