Top Ten List

Oracle 11g PL/SQL Features and Enhancements

Rick Michaud
Senior Sales Consultant
Oracle Corporation

Who Am I?

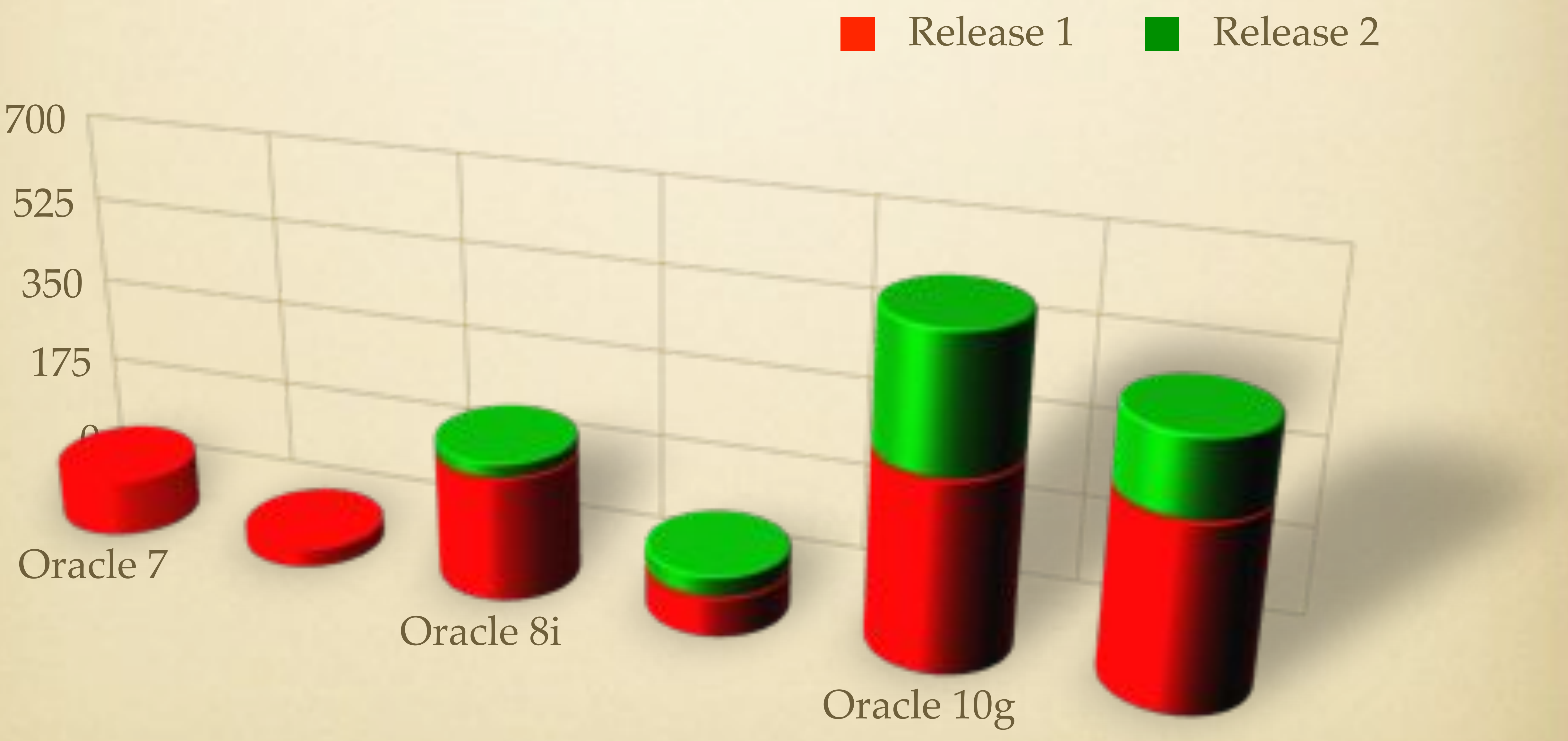
- Senior Sales Consultant for Oracle for the last 3 years.
- 20 years as an architect, developer and DBA.
- My last name is pronounced “Me Show”

Complex Agenda

- Introduction
- Top Ten List
- Q&A

Introduction

Oracle Features Added by Release



11g PL/SQL Feature Focus

1. Simplifying database PL/SQL development
2. Improving usability
3. Enhancing performance
4. Providing functionality and tools to do the above!

Top Ten List

#10

Simplified Sequence Expressions

#10 Simplified Sequence Expressions

- Was there any real useful reason for the "Dual" table?
- Can now use sequence expression whenever there is a numeric expression.
- Before 11gR1:

```
DECLARE
     v BINARY_INTEGER;
     w BINARY_INTEGER;
BEGIN
     SELECT MY_SEQ.NEXTVAL INTO v FROM DUAL;
     SELECT MY_SEQ.CURRVAL INTO w FROM DUAL;
END;
```

- After 11gR2:

```
DECLARE
     v BINARY_INTEGER;
BEGIN
     v := MY_SEQ.NEXTVAL;
     w := MY_SEQ.CURRVAL;
END;
```

11gR1

#9

Fine-Grained Dependency Tracking

#9 Fine-Grained Dependency Tracking

- Prior to 11g:
    - "ORA-4068:Existing State of packages has been discarded or invalidated"
    - Invalid views until queried
    - Recompile Loop:
        - UTL_RECOMP...
        - SELECT COUNT(*) FROM DBA_OBJECTS WHERE STATUS='INVALID'
- With 11g:
    - Dependencies tracked at code level
    - 11gR2 - now adds support for triggers!
    - Can still get ORA-4068, but has been reduced.

11gR1

#8

Named and Mixed Notation

#8 Named and Mixed Notation User Functions in SQL

- Given:

```
FUNCTION f(
p1 IN INTEGER := 1,
p2 IN INTEGER := 2,
...
pn IN INTEGER := 99)
RETURN INTEGER
```

- Can now do:

```
SELECT f(1, pn=>3) FROM dual
```

- Instead of:

```
SELECT f(pn=>3, p2=>2, p1=>1) FROM dual
```

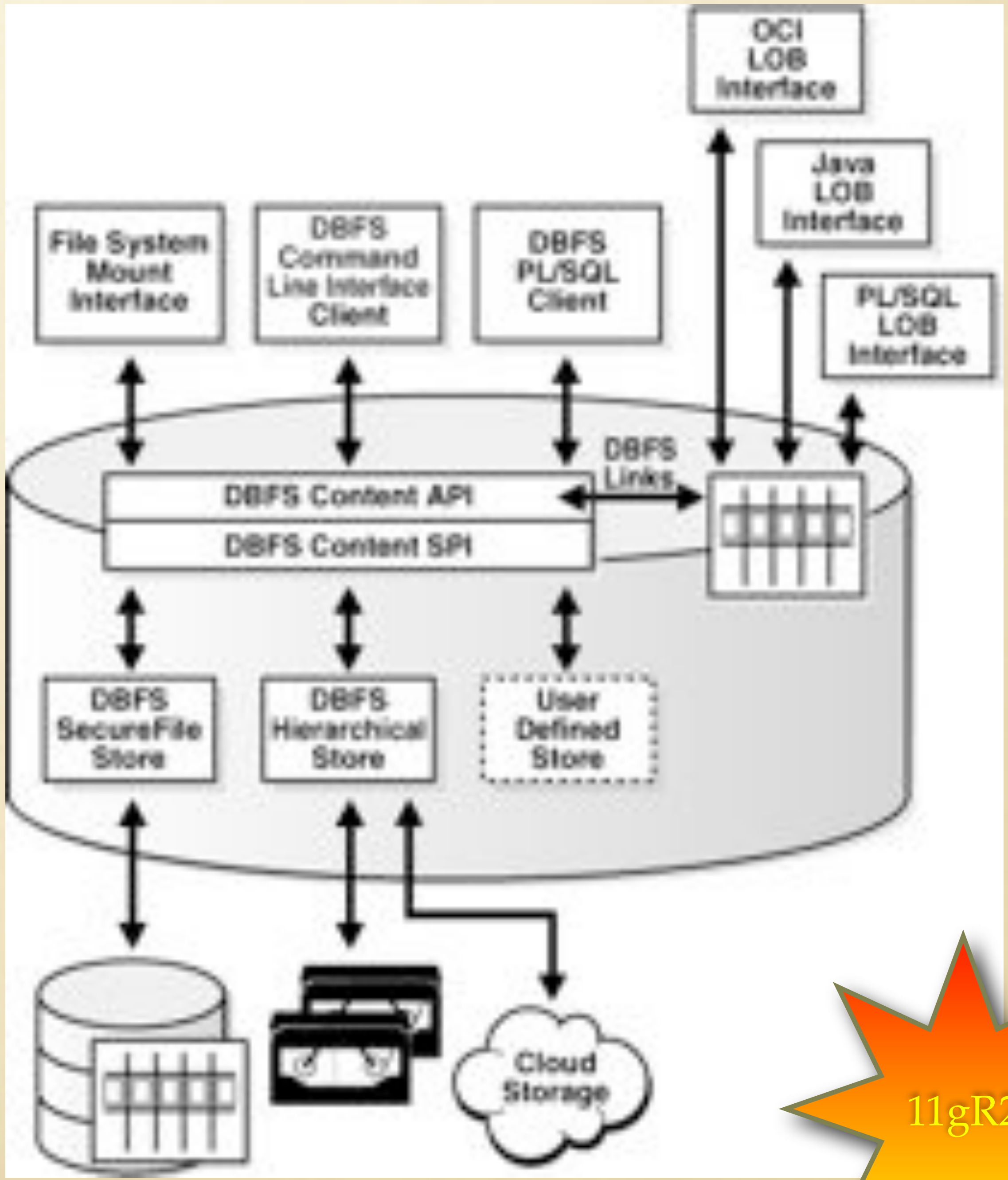
11gR1

#7

DBFS Content API

DBFS and the DBFS Content API

- DBFS is a posix compliant file system built on top of the database
- Why do this?
  - Allows unstructured content to be managed with relational content
  - High performance solution for parallel ETL (e.g. Staging Files)
  - Security and data
  - Data load for Database Machine (aka "Exadata")
- DBFS Content API allows interaction with filesystem at a PL/SQL programmatic level.
  - Great for data manufacturing/production.



11gR2

DBFS Content API Example

```
connect foo/******

declare
   ret integer;
   b   blob;
   str varchar2(1000)  := '' || chr(10) ||
'#include <stdio.h>' || chr(10) || '' || chr(10) || 'int main(int argc, char** argv)' || chr(10) ||
'{' || chr(10) || '    (void) printf("hello world\n");' || chr(10) || '    return 0;' || chr(10) ||
'}' || chr(10) || '';

    begin
        ret := dbms_fuse.fs_mkdir('/mnt1/src');
        ret := dbms_fuse.fs_creat('/mnt1/src/hello.c', content => b);
        dbms_lob.writeappend(b, length(str), utl_raw.cast_to_raw(str));
        commit;
    end;
    /
    show errors;

    -- verify newly created directory and file
    select pathname, pathtype, length(filedata),
        utl_raw.cast_to_varchar2(filedata)
        from dbfs_content
            where pathname like '/mnt1/src%'
            order by pathname;
```

#6

PL/SQL Subprogram Inlining

#6: PL/SQL Subprogram Inlining

- Performance enhancement feature to alleviate overhead of subprogram calls.
- Replaces a subprogram invocation with a copy of the invoked subprogram.
- Used for “small” frequently-used subprograms.
- Similar to inlining in C
- Can be enabled with the following:
  - Automatic: Init.ora parameter: PLSQL_OPTIMIZE_LEVEL = 3)
  - Manual: PRAGMA INLINE (Must have PLSQL_OPTIMIZE_LEVEL = 2)
- Will improve performance in the majority of cases, however:
  - Remember to test performance to measure impacts
  - Make sure you use only on small, frequently used subprograms.
  - Disable when debugging code.

11gR1

#5

PL/Scope

#5 PL/Scope

- Developer productivity tool that enables navigation of code by named tags
- Useful for analyzing:
    - Impact of code changes beforehand
    - “Inherited” code from a developer who didn’t believe in documentation.
- Similar to c-scope for C Developers
- Data For PL/SQL program units generated at compile time and stored in data dictionary
- Tooling available in SQL Developer

11gR1

#4

PL/SQL Hierarchical Profiler

#4 PL/SQL Hierarchical Profiler

- Identifies bottlenecks and performance issues in PL/SQL applications
- Reports the dynamic execution profile of the PL/SQL program organized by subprogram calls.
- Requires no special source or compile time preparation.
- Stores results in hierarchical profiler tables.
- Provides information such as:
  - Number of calls to subprogram
  - Time spent in subprogram
  - Time spent in descendant subprograms
  - Callers of a given subprogram
  - All called subprograms of a particular subprogram

11gR1

#4 PL/SQL Hierarchical Profiler

- How to run manually:
  1. Grant execute on DBMS_HPROF.
  2. Setup directory for tracefiles via CREATE DIRECTORY
  3. Start profiling.
  4. Call your subprogram.
  5. Stop profiling.
  6. Analyze results:
     1. Review raw trace file.
     2. Using DBMS_HPROF.ANALYZE*
     3. Using plshprof utility to generate HTML Report
     4. * Requires Data Dictionary tables to be created with dbmshptab.sql
- Tooling available through SQLDeveloper 2.0!

```
CREATE DIRECTORY PLSHPROF_DIR as '/private/plshprof/
results';
GRANT READ, WRITE ON DIRECTORY PLSHPROF_DIR TO HR;
GRANT EXCUTE on DBMS_HPROF to HR;
```

```
BEGIN
  /* Start profiling.
     Write raw profiler output to file test.trc in a
directory
     that is mapped to directory object PLSHPROF_DIR
     (see note following example). */

  DBMS_HPROF.START_PROFILING('PLSHPROF_DIR',
'test.trc');
END;
/
-- Execute procedure to be profiled
BEGIN
  test;
END;
/
BEGIN
  -- Stop profiling
  DBMS_HPROF.STOP_PROFILING;
END;
/
```

#3

Edition-based Redefinition

#3 Edition-Based Redefinition

- Enables application developers to upgrade objects while in use
- Reduces system downtime
- Enables seamless application upgrades
- Data changes are possible through cross-edition triggers.
- Supported objects:
  - Synonym
  - Function
  - Package Specification
  - Type Specification
  - Library
  - View
  - Procedure
  - Package Body
  - Type Body
  - Trigger

11gR2

Simple Example

```
CREATE OR REPLACE FUNCTION NORMALIZED_NAME (FIRST_NAME IN VARCHAR2, LAST_NAME IN VARCHAR2) RETURN VARCHAR2 AS
BEGIN
  RETURN FIRST_NAME || ' ' || LAST_NAME;
END NORMALIZED_NAME;
/
DECLARE
  v_Return VARCHAR2(200);
BEGIN
  v_Return := NORMALIZED_NAME('john','doe');
  DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
END;
/
>>v_Return = john doe

CREATE EDITION patch_normalized_name_1;
ALTER SESSION SET EDITION = patch_normalized_name_1;
CREATE OR REPLACE FUNCTION NORMALIZED_NAME (FIRST_NAME IN VARCHAR2, LAST_NAME IN VARCHAR2) RETURN VARCHAR2 AS
BEGIN
  RETURN INITCAP(LAST_NAME) || ', ' || INITCAP(FIRST_NAME);
END NORMALIZED_NAME;
/
DECLARE
  v_Return VARCHAR2(200);
BEGIN
  v_Return := NORMALIZED_NAME('john','doe');
  DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
END;
/
>>v_Return = Doe, John

ALTER SESSION SET EDITION = ora$base;;
DECLARE
  v_Return VARCHAR2(200);
BEGIN
  v_Return := NORMALIZED_NAME('john','doe');
  DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
END;
/
>>v_Return = john doe
```

2

Native Database Web Services

#2 Native Database Web Services

- Easily expose functionality in Database as a web service
  - PL/SQL packages, procedures, functions
  - SQL Queries
  - XQuery
- No need for app server
  - Uses a servlet architecture on XDB HTTP Server
  - Can be secured by integrated with a WS-Security provider

11gR1

1

PL/SQL Native Compiler

PL/SQL Native Compilation

- Useful feature for PL/SQL performance improvement
  - Especially computational code
  - Great for DW server-side transformations
- Released in 9i
  - Required platform native compiler to be installed.
  - Was cumbersome to configure and setup
  - Sometimes this wasn't allowed -- e.g. PRODUCTION!!
- 11g compiler is built-in
  - Doesn't require a 3rd party compiler
  - No DLLs generated, compiled code stored in database catalog
- Gotchas:
  - Not available on all platforms
  - For those 10g Native Compilation option is still available
  - For debugging processes, stick with interpreted code as compiling takes time

11gR1

And a few more...

Sorry I couldn’t help myself!

And a few more...

- Pipelined Table Functions
- Loop CONTINUE Statement
- "Simple" Types:
  - SIMPLE_FLOAT
  - SIMPLE_INTEGER
  - SIMPLE_DOUBLE
  - PL/SQL Function Result Cache
- More control over Triggers:
  - ENABLE
  - DISABLE
  - FOLLOWS
- IGNORE_ROW_ON_DUPKEY_INDEX hint
- Analytic Functions 2.0
- DBMS_PARALLEL_EXECUTE package
- Scheduler Improvements:
  - File Watcher
  - Email Notification
  - Remote Database Jobs
- XStream API
- XML DB Improvements
  - Binary XML
  - XDB Repository Improvements
  - XMLIndex Improvements
  - XMLType Partitioning

Burning Questions?

For more information

- Oracle Home Page: http://www.oracle.com
- Oracle Technology Network: http://otn.oracle.com
- Oracle Search: http://search.oracle.com
- Oracle Iron Man 2 Information: http://www.oracle.com/us/ironman2/index.html