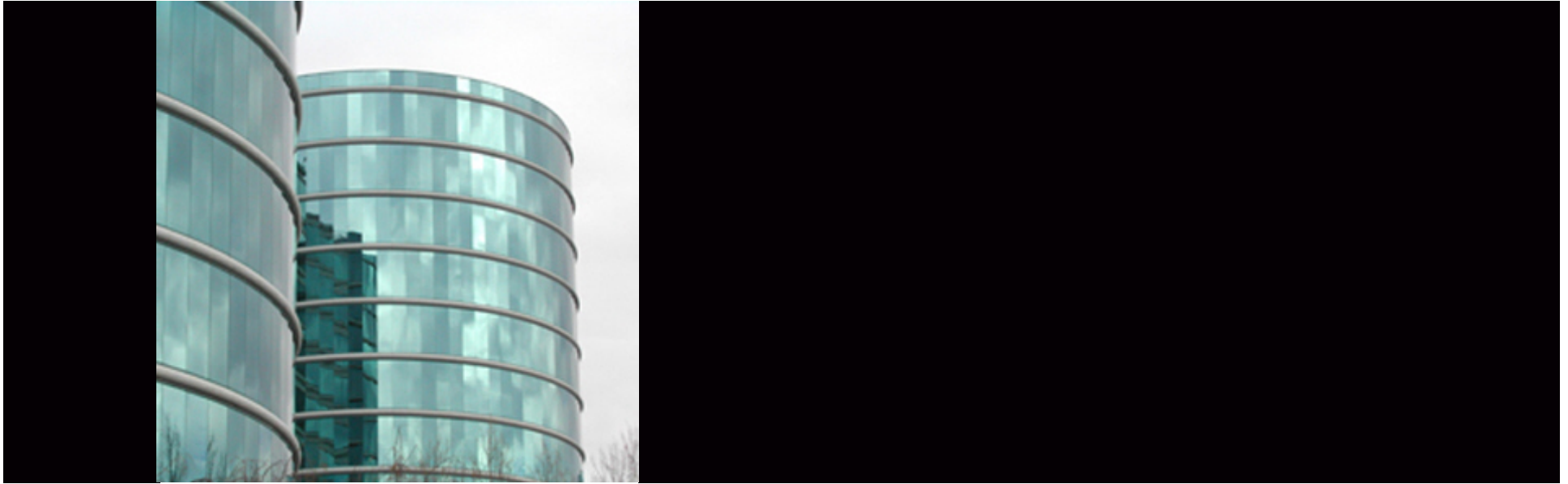

ORACLE®



ORACLE®

PL/SQL 11g New Features

Nick Donatone
Principal Grid Sales Consultant

Agenda

- Discuss
 - PL/SQL tips and techniques
 - New 11g features
 - Efficient PL/SQL Coding,
 - Disabled Triggers,
 - SIMPLE_INTEGER Data Type,
 - Compound Triggers,
 - Trigger Ordered Executions,
 - CONTINUE command,
 - Sequences,
 - Compile-time warning



What is PL/SQL?

- **PL/SQL (Procedural Language/Structured Query Language)** is
 - Oracle Corporation's proprietary procedural extension to the SQL database language.
 - PL/SQL's syntax strongly resembles that of Ada.
 - The key strength of PL/SQL is its tight integration with the Oracle database.
 - PL/SQL is one of three languages embedded in the Oracle Database, the other two being SQL and Java.





Tips and Techniques

- **Oracle SQL Developer: SQL Worksheet**
- **Oracle SQL Developer: PLSQL Editing and Debugging**
- **PLSQL_CODE_TYPE=Native**



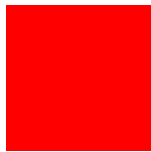
Oracle SQL Developer: SQL Worksheet

- The SQL Worksheet supports the creation of SQL, PL/SQL and SQL *Plus commands.
- These can be run individually or consecutively.
- A SQL History option makes recalling previous commands easy, while the Explain Plan option allows users to see the execution plan for selected statements.



Oracle SQL Developer: PL/SQL Editing and Debugging

- Robust editing environment
 - users can create and edit PL/SQL
 - take advantage of the code formatting
 - add bookmarks and use code insight
- When it comes to debugging PL/SQL, breakpoints, smart data, a debugger stack and watches are all available.
 - These features allow the user to set a break point and run and test the code, supplying alternate data at runtime while debugging.
- Creating PL/SQL in the editor or using the SQL Worksheet is made easier by the availability of snippets
 - Snippets are code fragments, such as SQL functions, Optimizer hints or miscellaneous PL/SQL programming techniques, which users can drag onto the PL/SQL Editor or the SQL Worksheet



Oracle SQL Developer

File Edit View Navigate Run Debug Source Tools Help

oe INSERT_ORD_LINE

Code Dependencies Details

Actions...

```
create or replace TRIGGER "OE".insert_ord_line
BEFORE INSERT ON order_items
FOR EACH ROW
DECLARE
    new_line number;
BEGIN
    SELECT (NVL(MAX(line_item_id),0)+1) INTO new_line
    FROM order_items
    WHERE order_id = :new.order_id;
    :new.line_item_id := new_line;
END;
```

Logging Page - Log

L...	Sequence	Elapsed	Source	Message
------	----------	---------	--------	---------

Messages Logging Page

TRIGGER OE.INSERT_ORD_LINE@oe | Editing



PLSQL_CODE_TYPE=Native

- Syntax `PLSQL_CODE_TYPE = { INTERPRETED | NATIVE }`
- Default value `INTERPRETED`
- Modifiable `ALTER SESSION, ALTER SYSTEM`
- C compiler no longer required
 - Compiles to native DLL
 - Windows `.dll` (dynamically linkable library)
 - Unix `.so`



Efficient PL/SQL Coding

- New features of 11g make PL/SQL programming easier:
 - Ability to force triggers of the same type to follow a sequence
 - New CONTINUE statement
 - Compound triggers
- PL/SQL is a comprehensive development platform
- It has more and more functionality and requires less coding
- Oracle Database 11g new functionality in PL/SQL can help programmers write code more efficiently



Disabled Triggers

- You can now create a Trigger as Disabled

```
CREATE OR REPLACE  
TRIGGER MY_DISABLED_TRIGGER  
BEFORE INSERT ON CUSTOMERS  
DISABLE  
BEGIN  
  NULL;  
END;
```

- The default is enabled



Oracle SQL Developer : TABLE OE.CUSTOMERS@oe

File Edit View Navigate Run Debug Source Tools Help

Connections Reports

oe CUSTOMERS MY_DISABLED_TRIGGER

Columns Data Constraints Grants Statistics Column Statistics Triggers Dependencies Detail

Actions...

Trigger Name	Trigger Type	Triggering Event	Status	Object ID
MY_DISABLED_TRIGGER	BEFORE STATEMENT	INSERT	DISABLED	71058

Refresh: 0

Logging Page - Log

L...	Sequence	Elapsed	Source	Message
------	----------	---------	--------	---------

Messages Logging Page

All Rows Fetched: 1

oe | OE | CUSTOMERS | Editing



SIMPLE_INTEGER Data Type

- The SIMPLE_INTEGER Data Type has semantics that exactly match those of the hardware's integer operations
 - It can not be NULL (has a not null constraint)
 - Rather than overflowing it wraps
 - Range is (-2147483648 to +2147483648)
 - It's faster than *pls_integer* when using native PL/SQL compilation



Compound Triggers

- The Compound Trigger is a single database object that “feels” rather like a package body, allows you to create a “pseudo-procedure” for each of the four timing points below:
 - Before the firing statement
 - Before each row that the firing statement impacts
 - After each row that the firing statement impacts
 - After the firing statement
 - after each row is
 - begin
 -
 - end after each row
- Use it to eliminate the “mutating table” error; audit table



Compound Triggers

- Mutating table
- An example is creating an entry in an audit table
 - Create an audit record each time an employees salary changes
 - Could have many employee salary changes (bulk update)
 - 10.2 you would have to use the “ancillary package paradigm”
 - Initialize globals in a before statement
 - Batch and flush rows in before each row
 - Final flush in an after statement

Audit Table Example

```
create or replace
trigger audit_customer_changes_tgr
for insert or update on customers
compound trigger
threshold constant simple_integer := 5;
type audit_info_t is table of audit_customer_changes%rowtype
index by pls_integer;
v_audit_info audit_info_t;
idx simple_integer:=0;
-- subprog
PROCEDURE flush_audit_array is
n constant simple_integer := v_audit_info.count();
BEGIN
FORALL j in 1..n
INSERT INTO audit_customer_changes VALUES v_audit_info (j);
v_audit_info.delete();
idx := 0;
END flush_audit_array;
-- Now the 11g code
BEFORE STATEMENT IS
BEGIN
v_audit_info.delete();
idx := 0;
END BEFORE STATEMENT;

AFTER EACH ROW IS
BEGIN
idx := idx + 1;
v_audit_info(idx).id := :new.id;
v_audit_info(idx).modified_date := SYSDATE();
v_audit_info(idx).user_name := USER();
IF idx >= threshold THEN
flush_audit_array();
END IF;
END AFTER EACH ROW;
-----
AFTER STATEMENT IS
BEGIN
flush_audit_array();
END AFTER STATEMENT;
END audit_customer_changes_tgr
ALTER TRIGGER "OE"."AUDIT_CUSTOMER_CHANGES_TGR" ENABLE
```





Let's Update the Customer Table

```
update customers
  set cust_first_name = 'Harry Dean'
 where customer_id=193;
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

results:

	CUSTOMER_ID	CUST_FIRST_NAME	CUST_LAST_NAME	CUST_ADDRESS	PHI
1	193	Harry dean	Kinski	OE.CUST_ADDRESS_TYP	VAI

Audit_Customer_Changes table

	ID	MODIFIED_DATE	USER_NAME
1	1	28-JAN-08	OE



Ordered Execution in Triggers

- Follows clause
- Use it to define the execution order of a like set of triggers
 - Before insert
 - After insert
 - Before update
 - After update

```
create or replace trigger my_second_before_insert_trigger
before insert on emp
follows my_first_before_insert_trigger
begin
    null;
end;
/
```



CONTINUE

- Continue can be used inside a looping structure to jump out of the loop before executing all the steps inside the loop

```
declare v_my_counter number:=0;
begin
    for count l in 1...20
    LOOP
        v_my_counter := v_my_counter+1
        dbms_output.put_line('Counter='||v_my_counter);
        continue when v_my_counter>5;
        v_my_counter := v_counter+1
    END LOOP;
end;
/
```



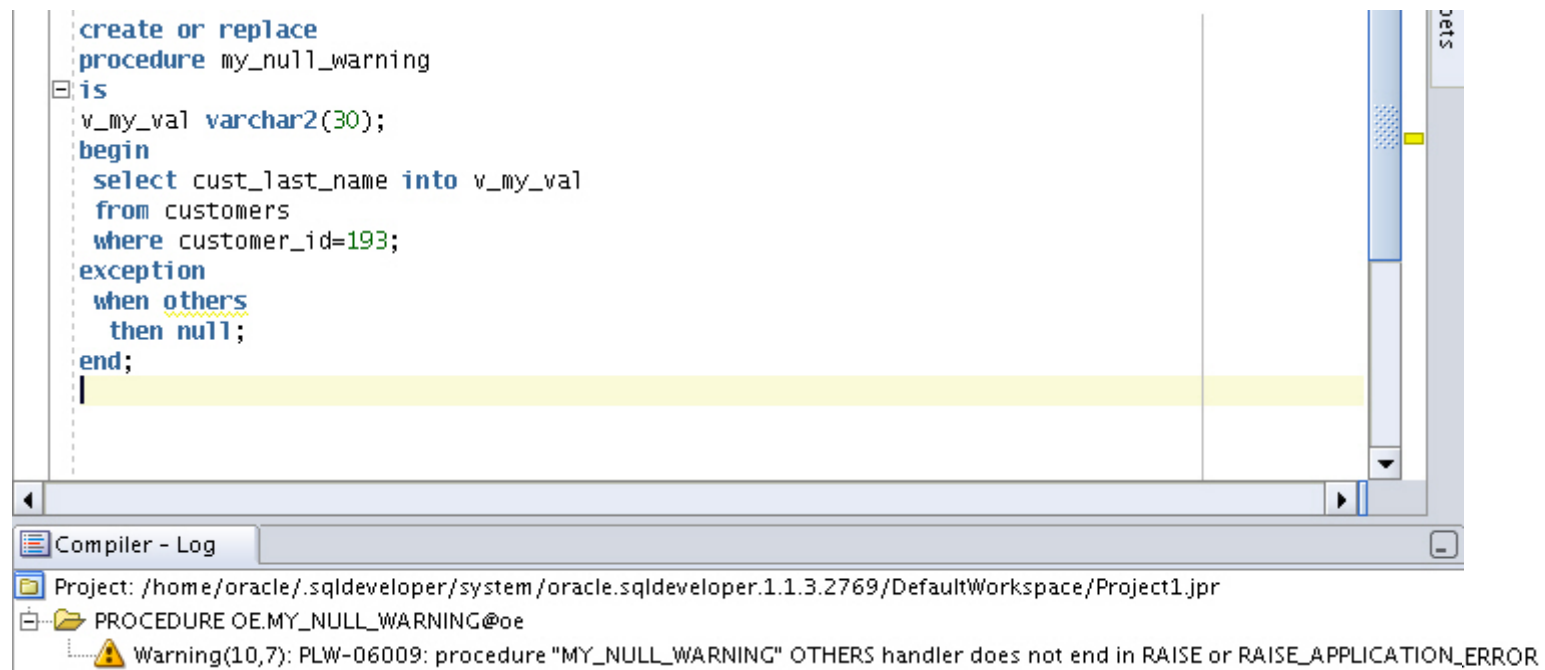
Straight Sequences

- You no longer have to select `my_sequence.nextval` into `v_seq_no` from `dual`
- Now just us use
 - `seq1:=my_sequence.nextval`
 - or `seq2:=my_sequence.currval`

Compile-time Warning

- “when others then null” will now generate a warning
 - PLW-06009

Alter session set plsql_warnings = 'enable:all';



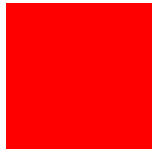
```
create or replace
procedure my_null_warning
is
v_my_val varchar2(30);
begin
select cust_last_name into v_my_val
from customers
where customer_id=193;
exception
when others
then null;
end;
```

Compiler - Log

Project: /home/oracle/.sqldeveloper/system/oracle.sqldeveloper.1.1.3.2769/DefaultWorkspace/Project1.jpr

PROCEDURE OE.MY_NULL_WARNING@oe

Warning(10,7): PLW-06009: procedure "MY_NULL_WARNING" OTHERS handler does not end in RAISE or RAISE_APPLICATION_ERROR



Others

- Inlining
- PL/SQL Function Result Cache
- Dynamic SQL
 - SQL statements can now exceed 32kb
 - dbms.parse is overloaded for CLOBS
 - dbms_sql now supports abstract data types
 - dbms_sql allows bulk binds
- Dynamic SQL and REF Cursors
 - A REF Cursor can now be converted into a dbms_sql cursor
 - And a dbms_sql cursor can be converted into a REF Cursor



Q U E S T I O N S
A N S W E R S

<http://otn.oracle.com/rac>

ORACLE



For More Information

<http://search.oracle.com>

or

http://www.oracle.com/technology/tech/pl_sql/pdf/plsql_new_in_11gr1.pdf



ORACLE IS THE INFORMATION COMPANY