Advanced PLSQL

Lesson 05: Oracle 11g features

Lesson Objectives

In this lesson you will learn

- Oracle 11g Features
  - Use of sequence
  - Compound Triggers
  - Subprogram Inlining
  - PL/SQL Continue statement
  - PL/SQL Compiler Warnings

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

5.1 Use of Sequence

- A “Sequence” is an object, which can be used to generate sequential numbers.
- A Sequence is used to fill up columns, which are declared as UNIQUE or PRIMARY KEY.
- A Sequence uses “NEXTVAL” to retrieve the next value in the sequence order.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

Creating a Sequence

- Here is one more example of sequence:
  - s1 will generate numbers 1,2,3....,10000, and then stop.

```
CREATE SEQUENCE  s1
                INCREMENT BY 1
                START WITH 1
                MAXVALUE 10000
                NOCYCLE ;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4

NEXTVAL and CURRVAL pseudo columns

- NEXTVAL returns the next available sequence value.
  - It returns a unique value every time it is referenced, even for different users.
- CURRVAL obtains the current sequence value.
- NEXTVAL must be issued for the Sequence before CURRVAL can be referenced.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

Characteristics of Sequence

- Characteristics of a Sequence:
  - Caching the Sequence values in memory to give faster access to those Sequence values
- Gaps in Sequence values can occur when:
  - a rollback occurs
  - the system crashes
  - a Sequence is used in another table
- Viewing the next available value by querying the USER_SEQUENCES table, when the sequence is created with NOCACHE

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

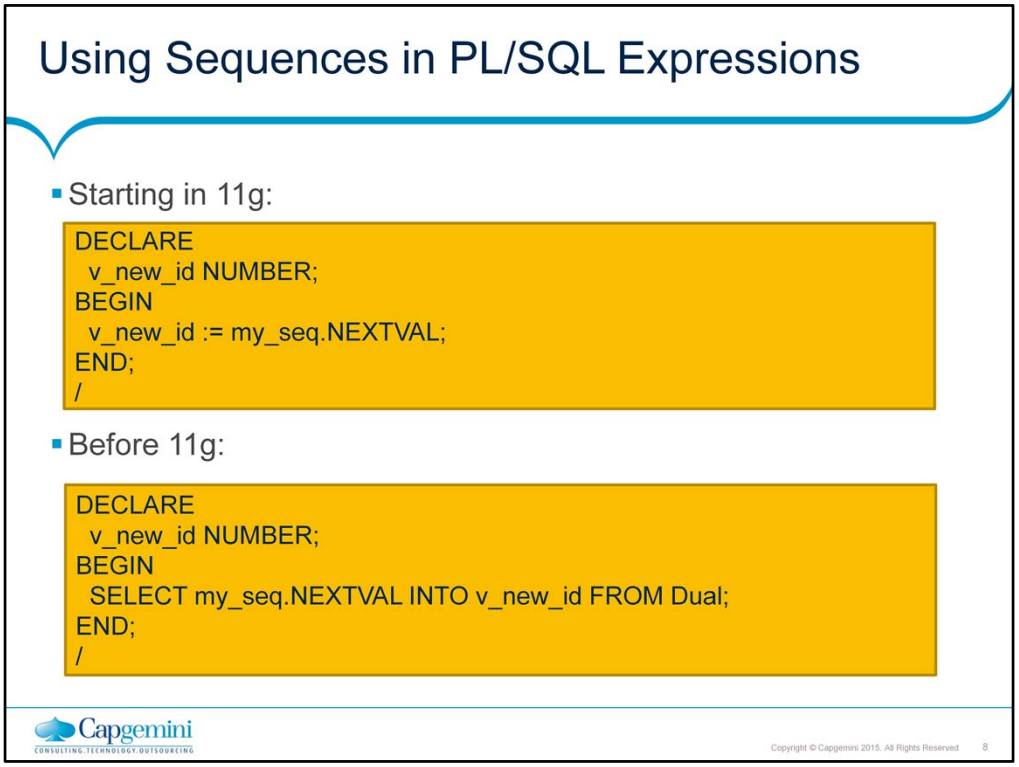
Drop a Sequence

- A Sequence can be removed from the data dictionary by using the DROP SEQUENCE statement.
- Once removed, the Sequence can no longer be referenced.

```
DROP SEQUENCE dept_deptid_seq;
Sequence dropped.
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7



Accessing Sequence ValuesIn Oracle Database 11*g*, you can use the NEXTVAL and CURRVAL pseudocolumns in any PL/SQL context, where an expression of the NUMBER data type may legally appear. Although the old style of using a SELECT statement to query a sequence is still valid, it is recommended that you do not use it.Before Oracle Database 11*g*, you were forced to write a SQL statement in order to use a sequence object value in a PL/SQL subroutine. Typically, you would write a SELECT statement to reference the pseudocolumns of NEXTVAL and CURRVAL to obtain a sequence number. This method created a usability problem. In Oracle Database 11*g*, the limitation of forcing you to write a SQL statement to retrieve a sequence value is eliminated. With the sequence enhancement feature:Sequence usability is improved.The developer has to type less.The resulting code is clearer

5.2 Compound Triggers

- A single trigger on a table that allows you to specify actions for each of the following four timing points:
  - Before the firing statement
  - Before each row that the firing statement affects
  - After each row that the firing statement affects
  - After the firing statement

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

Working with Compound Triggers

- The compound trigger body supports a common PL/SQL state that the code for each timing point can access.
- The compound trigger common state is:
  - Established when the triggering statement starts
  - Destroyed when the triggering statement completes
- A compound trigger has a declaration section and a section for each of its timing points.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

The Benefits of Using a Compound Trigger

- You can use compound triggers to:
  - Program an approach where you want the actions you implement for the various timing points to share common data.
  - Accumulate rows destined for a second table so that you can periodically bulk-insert them
  - Avoid the mutating-table error (ORA-04091)by allowing rows destined for a second table to accumulate and then bulk-inserting them

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

Timing-Point Sections of a Table Compound Trigger

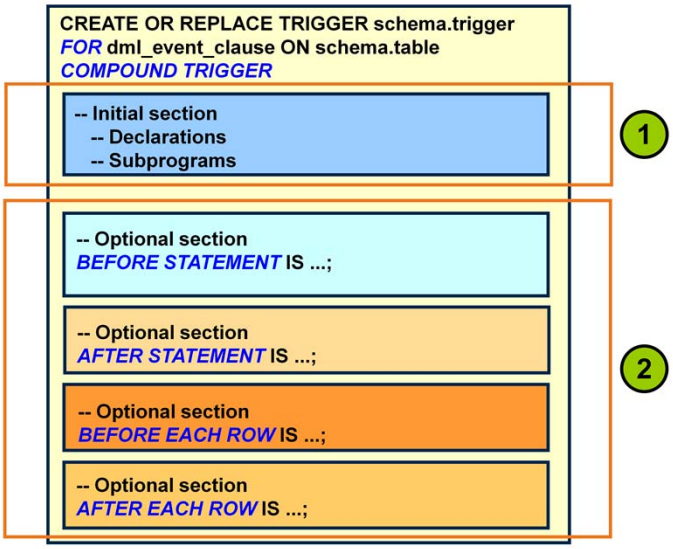
- A compound trigger defined on a table has one or more of the following timing-point sections. Timing-point sections must appear in the order shown in the table.

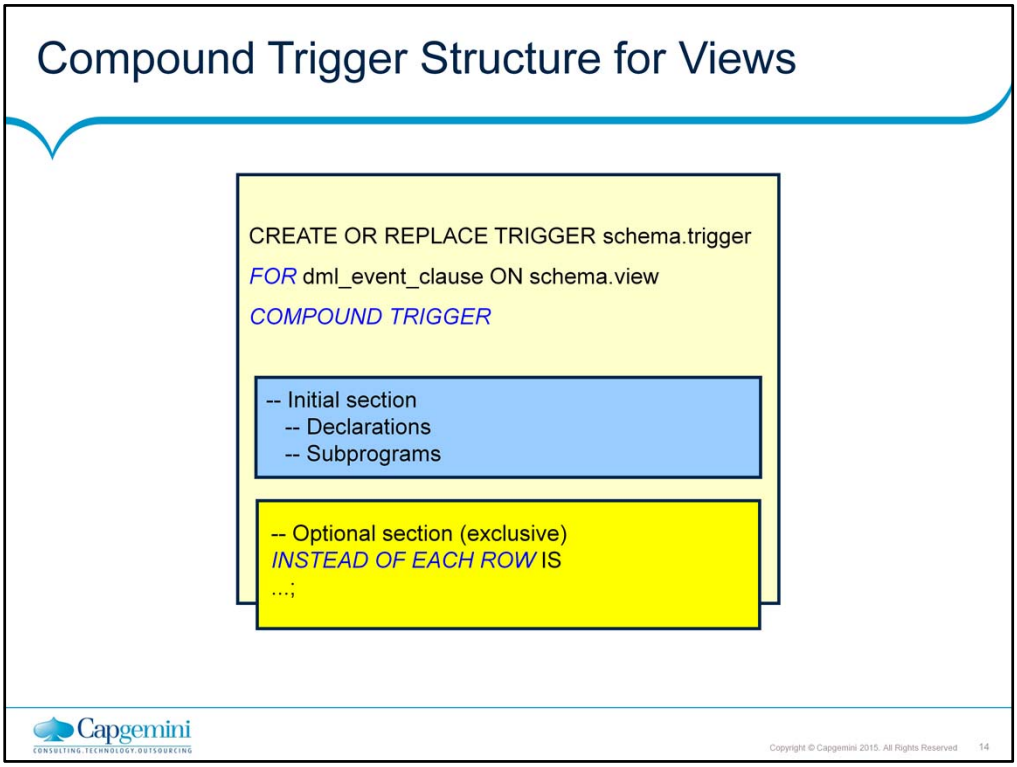
Timing Point	Compound Trigger Section
Before the triggering statement executes	`BEFORE` statement
After the triggering statement executes	`AFTER` statement
Before each row that the triggering statement affects	`BEFORE EACH ROW`
After each row that the triggering statement affects	`AFTER EACH ROW`

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

Compound Trigger Structure for Tables




Compound Trigger Structure for Views
CREATE OR REPLACE TRIGGER schema.trigger
FOR dml_event_clause ON schema.view
COMPOUND TRIGGER
-- Initial section
-- Declarations
-- Subprograms
-- Optional section (exclusive)
INSTEAD OF EACH ROW IS
...;
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
14

Compound Trigger Restrictions

- A compound trigger must be a DML trigger and defined on either a table or a view.
- The body of a compound trigger must be compound trigger block, written in PL/SQL.
- A compound trigger body cannot have an initialization block; therefore, it cannot have an exception section.
- An exception that occurs in one section must be handled in that section. It cannot transfer control to another section.
- `:OLD` and `:NEW` cannot appear in the declaration, `BEFORE STATEMENT`, or the `AFTER STATEMENT` sections.
- Only the `BEFORE EACH ROW` section can change the value of `:NEW`.
- The firing order of compound triggers is not guaranteed unless you use the `FOLLOWS` clause.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

Trigger Restrictions on Mutating Tables

- A mutating table is:
  - A table that is being modified by an UPDATE, DELETE, or INSERT statement, or
  - A table that might be updated by the effects of a DELETE CASCADE constraint
- The session that issued the triggering statement cannot query or modify a mutating table.
- This restriction prevents a trigger from seeing an inconsistent set of data.
- This restriction applies to all triggers that use the FOR EACH ROW clause.
- Views being modified in the INSTEAD OF triggers are not considered mutating.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

Mutating Table: Example

```
CREATE OR REPLACE TRIGGER check_salary
  BEFORE INSERT OR UPDATE OF salary, job_id
  ON employees
  FOR EACH ROW
  WHEN (NEW.job_id <> 'AD_PRES')
DECLARE
  v_minsalary employees.salary%TYPE;
  v_maxsalary employees.salary%TYPE;
BEGIN
  SELECT MIN(salary), MAX(salary)
   INTO v_minsalary, v_maxsalary
   FROM employees
   WHERE job_id = :NEW.job_id;
  IF :NEW.salary < v_minsalary OR :NEW.salary > v_maxsalary THEN
    RAISE_APPLICATION_ERROR(-20505,'Out of range');
  END IF;
END;
/
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Mutating Table: Example

```
UPDATE employees
SET salary = 3400
WHERE last_name = 'Stiles';
```

```
TRIGGER check_salary Compiled.

Error starting at line 1 in command:
UPDATE employees
 SET salary = 3400
 WHERE last_name = 'Stiles'
Error report:
SQL Error: ORA-04091: table ORA42.EMPLOYEES is mutating, trigger/function may not see it
ORA-06512: at "ORA42.CHECK_SALARY", line 5
ORA-04088: error during execution of trigger 'ORA42.CHECK_SALARY'
04091. 00000 -  "table %s.%s is mutating, trigger/function may not see it"
*Cause:    A trigger (or a user defined plsql function that is referenced in
           this statement) attempted to look at (or modify) a table that was
           in the middle of being modified by the statement which fired it.
*Action:   Rewrite the trigger (or function) so it does not read that table.
```


Using a Compound Trigger to Resolve the Mutating Table Error

```
CREATE OR REPLACE TRIGGER check_salary
  FOR INSERT OR UPDATE OF salary, job_id
  ON employees
  WHEN (NEW.job_id <> 'AD_PRES')
COMPOUND TRIGGER
  TYPE salaries_t          IS TABLE OF employees.salary%TYPE;
  min_salaries             salaries_t;
  max_salaries             salaries_t;

  TYPE department_ids_t    IS TABLE OF employees.department_id%TYPE;
  department_ids           department_ids_t;

  TYPE department_salaries_t  IS TABLE OF employees.salary%TYPE
                       INDEX BY VARCHAR2(80);
  department_min_salaries     department_salaries_t;
  department_max_salaries     department_salaries_t;

  -- example continues on next slide
```

Using a Compound Trigger to Resolve the Mutating Table Error

```
. . .
BEFORE STATEMENT IS
  BEGIN
    SELECT MIN(salary), MAX(salary), NVL(department_id, -1)
     BULK COLLECT INTO  min_Salaries, max_salaries, department_ids
    FROM    employees
    GROUP BY department_id;
    FOR j IN 1..department_ids.COUNT() LOOP
      department_min_salaries(department_ids(j)) := min_salaries(j);
      department_max_salaries(department_ids(j)) := max_salaries(j);
    END LOOP;
END BEFORE STATEMENT;

AFTER EACH ROW IS
  BEGIN
    IF :NEW.salary < department_min_salaries(:NEW.department_id)
      OR :NEW.salary > department_max_salaries(:NEW.department_id) THEN
      RAISE_APPLICATION_ERROR(-20505,'New Salary is out of acceptable
                              range');
    END IF;
  END AFTER EACH ROW;
END check_salary;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

5.3 Subprogram Inlining

- Inlining in PL/SQL is an optimization where the PL/SQL compiler replaces calls to subprograms (functions and procedures) with the code of the subprograms.
- A performance gain is almost always achieved because calling a subprogram requires the creation of a callstack entry, possible creation of copies of variables, and the handling of return values.
- With inlining, those steps are avoided.
- Automatic subprogram inlining can reduce the overheads associated with calling subprograms, whilst leaving your original source code in its normal modular state.
  - The process of subprogram inlining is controlled by the PLSQL_OPTIMIZE_LEVEL parameter and the INLINE pragma

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

Subprogram Inlining

- As an example, consider the following, where the PL/SQL optimization level is explicitly set to 2, and then an anonymous PL/SQL block is executed:

```
ALTER SESSION SET PLSQL_OPTIMIZE_LEVEL=2;
DECLARE
PROCEDURE wr(pStr IN VARCHAR2) IS
BEGIN
dbms_output.put_line(rpad(lpad(pStr,15,'='),30,'='));
dbms_output.put_line(dbms_utility.format_call_stack());
END;
BEGIN
wr('At Start');
wr('Done');
END;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

5.4 PL/SQL Continue statement

- Many programming languages have a continue statement that can be used inside a looping structure to cause the next iteration of the loop to occur, rather than process the remainder of the current loop iteration
- In complex looping structures this can save some processing time
- Oracle Database 11g now provides the continue statement for this very purpose
- Syntactically it is the same as the EXIT statement, and it allows for an optional WHEN clause and label

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

Continue Example

```
Declare
v_counter number:=0;
begin
for count in 1..10
LOOP
v_counter:=v_counter + 1;
dbms_output.put_line('v_Counter = '||v_counter);
continue when v_counter > 5;
v_counter:=v_counter+1;
END LOOP;
end;
/
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

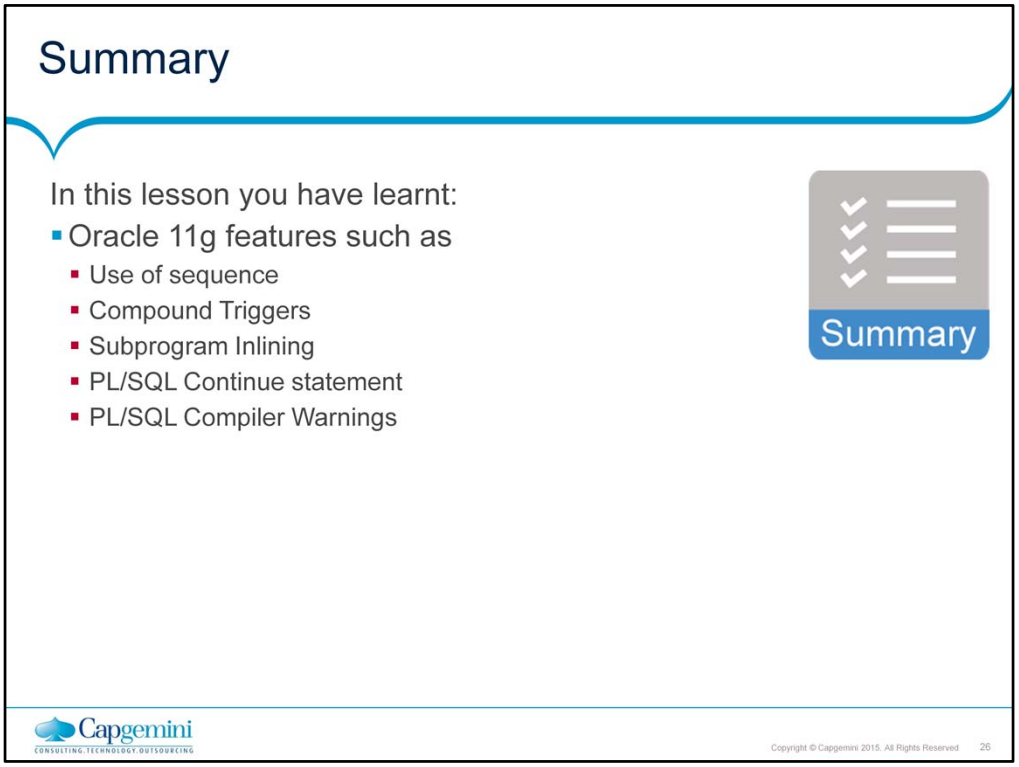
 24

5.5 PL/SQL Compiler Warnings

- The PL/SQL compiler now warns the user if the WHEN OTHERS exception handler does not raise an error.
- The user should include RAISE or RAISE_APPLICATION_ERROR to indicate the exact exception.
- Set the parameter
  - PLSQL_WARNINGS='enable:all';

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25



Add the notes here.

Review Question

- Question 1: The compound trigger makes it easier to program an approach where you want the actions you implement for the various timing points to share common data.
  - True/False
- Question 2:We can now directly assign the next and current value of a sequence as an assignment as in PL/SQL code using _____ and _____.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 27

Add the notes here.