Advanced PLSQL

Lesson 02: Design Considerations

Lesson Objectives

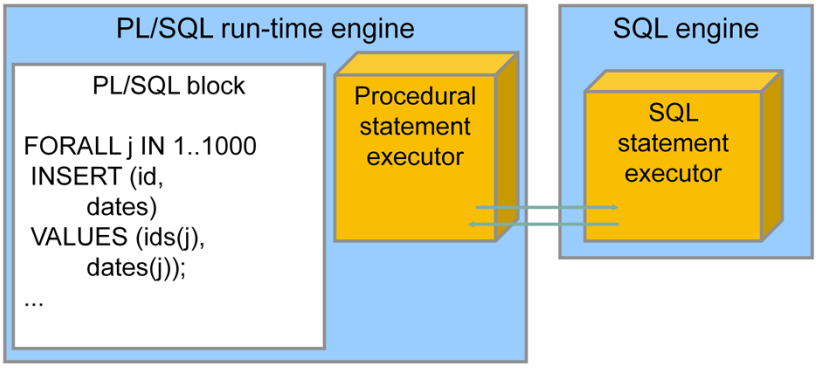
- To understand the following topics:
  - Design Considerations like
    - Bulk Binding concepts
    - Ref cursors
    - Using NOCOPY hint
    - Using PARALLEL_ENABLE hint
    - Using Cross-session PL/SQL function
    - Using DETERMINISTIC clause
    - Using returning clause

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

2.1: Design Considerations: bulk binding Bulk Binding

- Bulk binding binds whole arrays of values in a single operation, rather than using a loop to perform a FETCH, INSERT, UPDATE, and DELETE operation multiple times



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

2.1: Design Considerations: bulk binding

Bulk Binding: Syntax and Keywords

- The FORALL keyword instructs the PL/SQL engine to bulk bind input collections before sending them to the SQL engine.

```
FORALL index IN lower_bound .. upper_bound
  [SAVE EXCEPTIONS]
  sql_statement;
```

- The BULK COLLECT keyword instructs the SQL engine to bulk bind output collections before returning them to the PL/SQL engine.

```
... BULK COLLECT INTO
     collection_name[,collection_name] ...
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4

2.1: Design Considerations: bulk binding

Bulk Binding FORALL: Example

```
CREATE PROCEDURE raise_salary(p_percent NUMBER) IS
  TYPE numlist_type IS TABLE OF NUMBER
       INDEX BY BINARY_INTEGER;
  v_id  numlist_type; -- collection
BEGIN
 v_id(1):= 100; v_id(2):= 102; v_id(3):= 104; v_id(4) := 110;
 -- bulk-bind the PL/SQL table
  FORALL i IN v_id.FIRST .. v_id.LAST
    UPDATE employees
      SET salary = (1 + p_percent/100) * salary
      WHERE employee_id = v_id(i);
END;
/
```

```
EXECUTE raise_salary(10)
```

```
anonymous block completed
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

2.1: Design Considerations: bulk binding

Using BULK COLLECT INTO with Queries

- The SELECT statement supports the BULK COLLECT INTO syntax.

```
CREATE PROCEDURE get_departments(p_loc NUMBER) IS
  TYPE dept_tab_type IS
    TABLE OF departments%ROWTYPE;
  v_depts dept_tab_type;
BEGIN
  SELECT * BULK COLLECT INTO v_depts
  FROM departments
  WHERE location_id = p_loc;
  FOR i IN 1 .. v_depts.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(v_depts(i).department_id
     ||' '|| v_depts(i).department_name);
  END LOOP;
END;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

2.1: Design Considerations: bulk binding

Using BULK COLLECT INTO with Cursors

- The FETCH statement has been enhanced to support the BULK COLLECT INTO syntax.

```
CREATE OR REPLACE PROCEDURE get_departments(p_loc NUMBER) IS
  CURSOR cur_dept IS
    SELECT * FROM departments
    WHERE location_id = p_loc;
  TYPE dept_tab_type IS TABLE OF cur_dept%ROWTYPE;
  v_depts dept_tab_type;
BEGIN
  OPEN cur_dept;
  FETCH cur_dept BULK COLLECT INTO v_depts;
  CLOSE cur_dept;
 FOR i IN 1 .. v_depts.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(v_depts(i).department_id
     ||' '|| v_depts(i).department_name);
  END LOOP;
END;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

2.1: Design Considerations: bulk binding

Using BULK COLLECT INTO with a RETURNING Clause

```
CREATE OR REPLACE PROCEDURE raise_salary(p_rate NUMBER) IS
  TYPE emplist_type IS TABLE OF NUMBER;
  TYPE numlist_type IS TABLE OF employees.salary%TYPE
   INDEX BY BINARY_INTEGER;
  v_emp_ids  emplist_type := emplist_type(100,101,102,104);
  v_new_sals numlist_type;
BEGIN
 FORALL i IN v_emp_ids.FIRST .. v_emp_ids.LAST
  UPDATE employees
   SET commission_pct = p_rate * salary
  WHERE employee_id = v_emp_ids(i)
  RETURNING salary BULK COLLECT INTO v_new_sals;
 FOR i IN 1 ..     v_new_sals.COUNT LOOP
  DBMS_OUTPUT.PUT_LINE(v_new_sals(i));
 END LOOP;
END;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

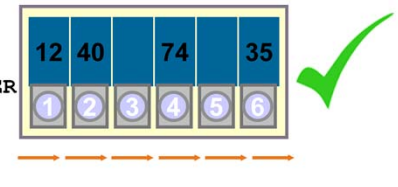
2.1: Design Considerations: bulk binding

Using Bulk Binds in Sparse Collections

- Current releases:

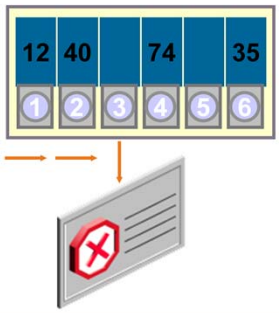
Index by
PLS_INTEGER

```
FOR ALL . . .
IN INDICES OF
```



- Prior releases:

Index by
BINARY_INTEGER



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

2.1: Design Considerations: bulk binding

Using Bulk Binds in Sparse Collections

```
-- The INDICES OF syntax allows the bound arrays
-- themselves to be sparse.

FORALL index_name IN INDICES OF sparse_array_name
       BETWEEN LOWER_BOUND AND UPPER_BOUND -- optional
      SAVE EXCEPTIONS -- optional, but recommended
       INSERT INTO table_name VALUES sparse_array(index_name);
. . .
```

```
-- The VALUES OF syntax lets you indicate a subset
-- of the binding arrays.

FORALL index_name IN VALUES OF index_array_name
       SAVE EXCEPTIONS -- optional,but recommended
       INSERT INTO table_name VALUES
     binding_array_name(index_name);
. . .
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

2.1: Design Considerations: bulk binding

Using Bulk Binds in Sparse Collections

- The typical application for this feature is an order entry and order processing system where:
  - Users enter orders through the Web
  - Orders are placed in a staging table before validation
  - Data is later validated based on complex business rules (usually implemented programmatically using PL/SQL)
  - Invalid orders are separated from valid ones
  - Valid orders are inserted into a permanent table for processing

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

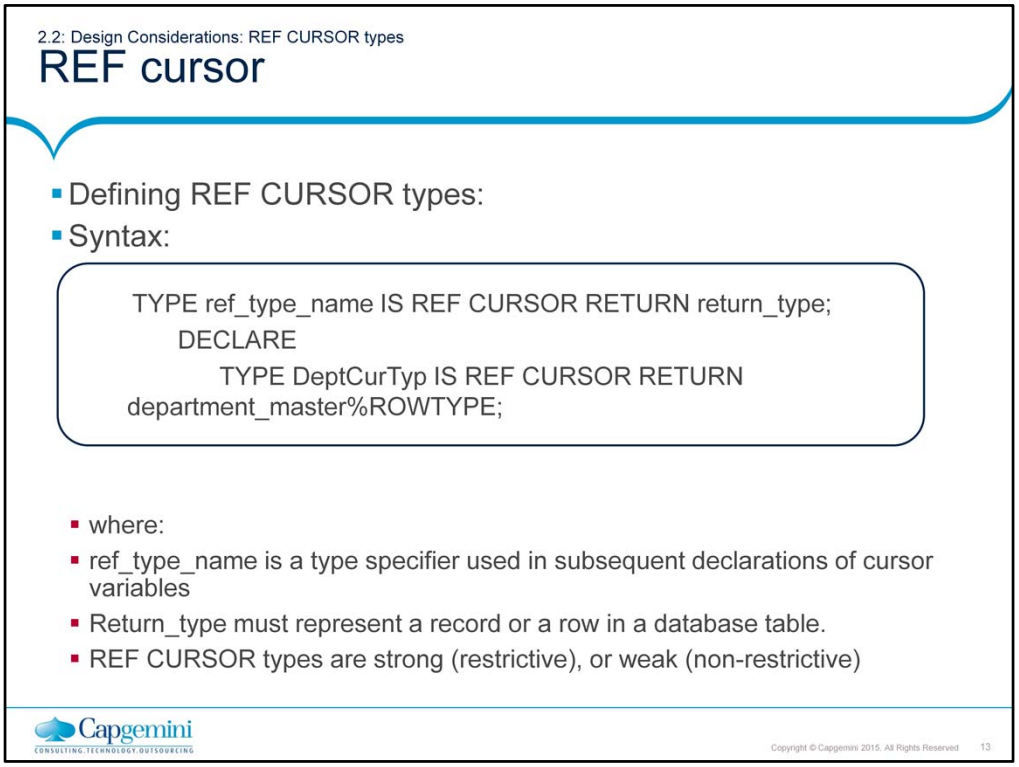
2.1: Design Considerations: bulk binding

Using Bulk Bind with Index Array

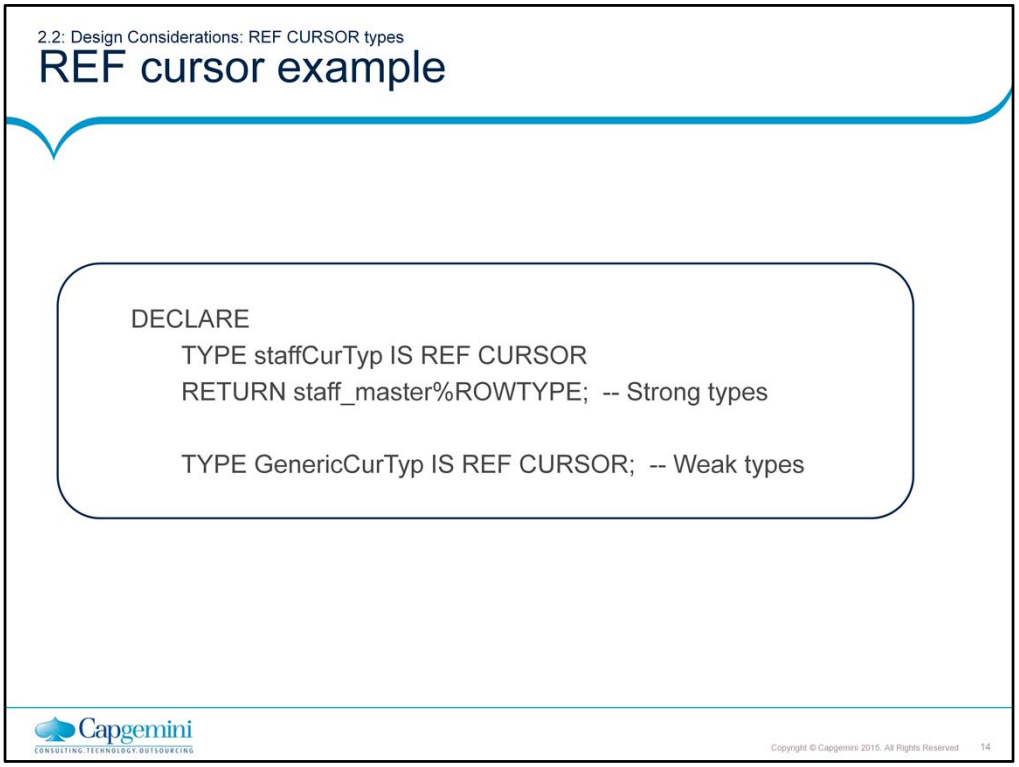
```
CREATE OR REPLACE PROCEDURE ins_emp2 AS
  TYPE emptab_type IS TABLE OF employees%ROWTYPE;
  v_emp emptab_type;
  TYPE values_of_tab_type IS TABLE OF PLS_INTEGER
        INDEX BY PLS_INTEGER;
  v_num   values_of_tab_type;
    . . .
BEGIN
    . . .
    FORALL k IN VALUES OF v_num
    INSERT INTO new_employees VALUES v_emp(k);
END;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

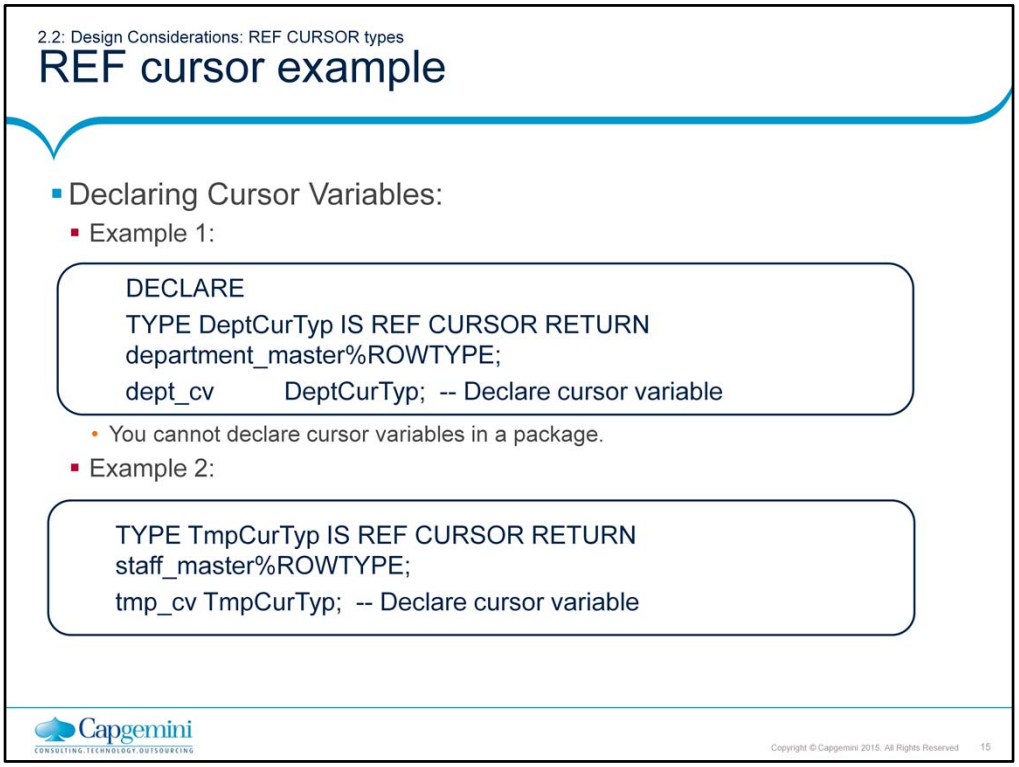
 12



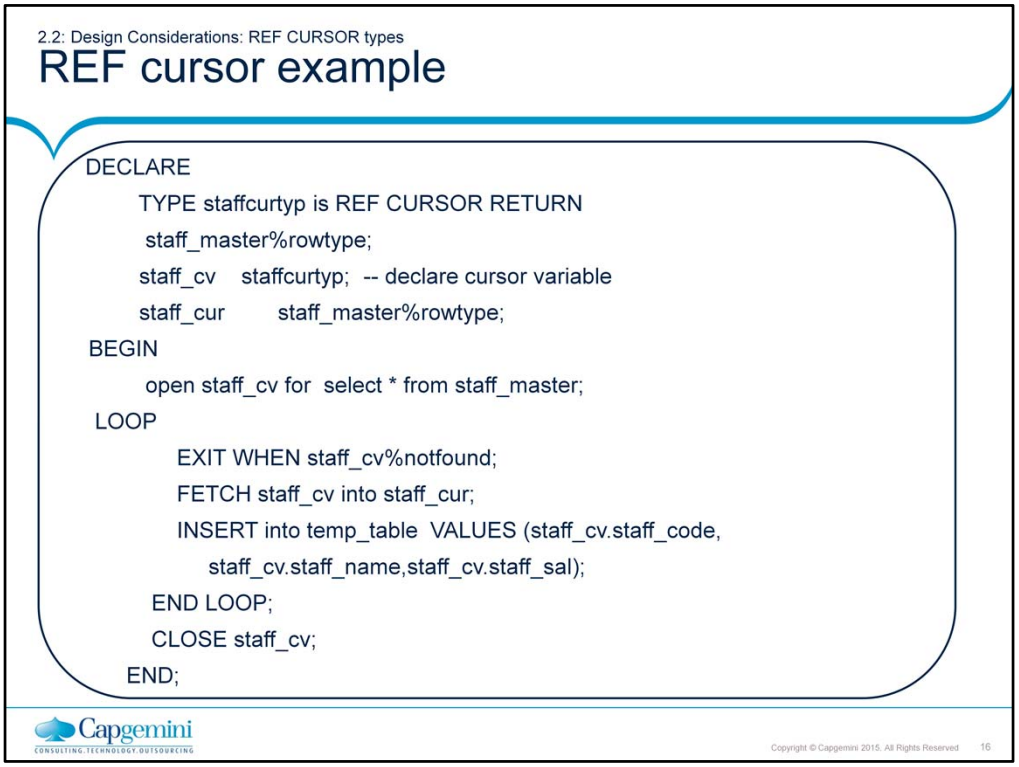
Add the notes here.



Add the notes here.



Add the notes here.



Add the notes here.

2.3: Design Considerations: using NOCOPY

Using the NOCOPY Hint

- The typical application for this feature is an order entry and order processing system where:
  - Users enter orders through the Web
  - Orders are placed in a staging table before validation
  - Data is later validated based on complex business rules (usually implemented programmatically using PL/SQL)
  - Invalid orders are separated from valid ones
  - Valid orders are inserted into a permanent table for processing

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

2.3: Design Considerations: using NOCOPY

Effects of the NOCOPY Hint

- If the subprogram exits with an exception that is not handled:
  - You cannot rely on the values of the actual parameters passed to a `NOCOPY` parameter
  - Any incomplete modifications are not “rolled back”
- The remote procedure call (RPC) protocol enables you to pass parameters only by value.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

When Does the PL/SQL Compiler Ignore the NOCOPY Hint?

- The NOCOPY hint has no effect if:
  - The actual parameter:
    - Is an element of associative arrays (index-by tables)
    - Is constrained (for example, by scale or NOT NULL)
    - And formal parameter are records, where one or both records were declared by using %ROWTYPE or %TYPE, and constraints on corresponding fields in the records differ
    - Requires an implicit data type conversion
  - The subprogram is involved in an external or remote procedure call

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

2.4: Design Considerations: using PARALLEL_ENABLE

Using the PARALLEL_ENABLE Hint

- Can be used in functions as an optimization hint
- Indicates that a function can be used in a parallelized query or parallelized DML statement

```
CREATE OR REPLACE FUNCTION f2 (p_p1 NUMBER)
  RETURN NUMBER PARALLEL_ENABLE IS
BEGIN
  RETURN p_p1 * 2;
END f2;
```

```
FUNCTION f2 Compiled.
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

2.5: Design Considerations: using Cross-session PL/SQL function

Using the Cross-Session PL/SQL Function Result Cache

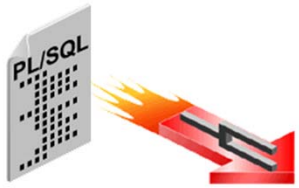
- Each time a result-cached PL/SQL function is called with different parameter values, those parameters and their results are stored in cache.
- The function result cache is stored in a shared global area (SGA), making it available to any session that runs your application.
- Subsequent calls to the same function with the same parameters uses the result from cache.
- Performance and scalability are improved.
- This feature is used with functions that are called frequently and dependent on information that changes infrequently.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

Enabling Result-Caching for a Function

- You can make a function result-cached as follows:
  - Include the RESULT_CACHE clause in the following:
    - The function declaration
    - The function definition
  - Include an optional RELIES_ON clause to specify any tables or views on which the function results depend.



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Declaring and Defining a Result-Cached Function: Example

```
CREATE OR REPLACE FUNCTION emp_hire_date (p_emp_id NUMBER)
      RETURN VARCHAR
 RESULT_CACHE RELIES_ON (employees) IS
 v_date_hired DATE;
BEGIN
 SELECT hire_date INTO v_date_hired
 FROM HR.Employees
 WHERE Employee_ID = p_emp_ID;
 RETURN to_char(v_date_hired);
END;
```

```
FUNCTION emp_hire_date Compiled.
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

2.6: Design Considerations: using DETERMINISTIC clause

Using the DETERMINISTIC Clause with Functions

- Specify DETERMINISTIC to indicate that the function returns the same result value whenever it is called with the same values for its arguments.
- This helps the optimizer avoid redundant function calls.
- If a function was called previously with the same arguments, the optimizer can elect to use the previous result.
- Do not specify DETERMINISTIC for a function whose result depends on the state of session variables or schema objects.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24

2.7: Design Considerations: using returning clause

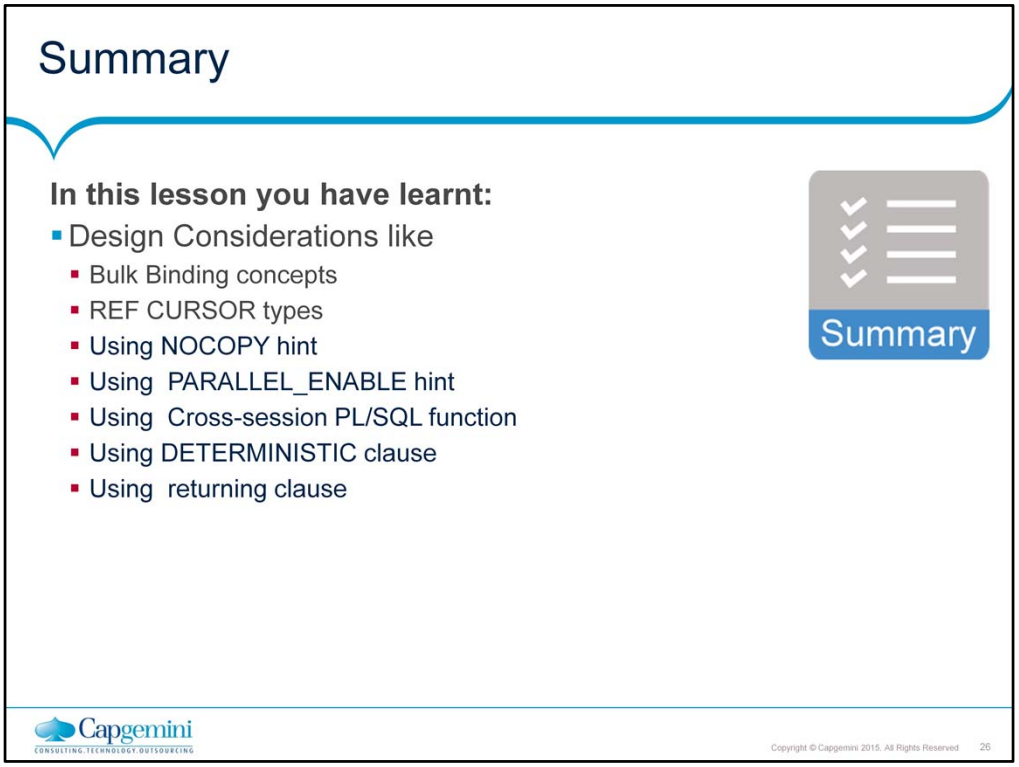
Using the RETURNING Clause

- Improves performance by returning column values with INSERT, UPDATE, and DELETE statements
- Eliminates the need for a SELECT statement

```
CREATE OR REPLACE PROCEDURE update_salary(p_emp_id NUMBER) IS
  v_name    employees.last_name%TYPE;
  v_new_sal employees.salary%TYPE;
BEGIN
  UPDATE employees
    SET salary = salary * 1.1
  WHERE employee_id = p_emp_id
  RETURNING last_name, salary INTO v_name, v_new_sal;
  DBMS_OUTPUT.PUT_LINE(v_name || ' new salary is ' ||
      v_new_sal);
END update_salary;
/
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25



Add the notes here.

Review Question

- To binding entire columns of Oracle data ____ clause is used by users
  - BIND_COLLECT
  - BULK_COLLECT
  - GROUP_COLUMN
  - GROUP_COLUMNS
- REF CURSOR types are _____ or _____ type.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 27

Add the notes here.