



Práctica Bases de Datos Avanzadas

Consultas Avanzadas

Felipe Ramírez | 2018-6104

Luis Franco | 2018-6342

Prof. Leandro Fondeur

25/7/2019

I. Realice los siguientes ejercicios luego de estudiar el documento D64254GC11_les07.ppt:

7.1 Construya un código PL/SQL que lea el ID del empleado, ID del supervisor y comisión. Si el empleado cobra comisión que guarde en una variable "Cobra comisión". Además, si el empleado se reporta a un supervisor que guarde en la misma variable "Tiene supervisor". Al final, mostrar en pantalla el siguiente mensaje:

"El empleado <<ID del empleado>> <<Mensaje>>".

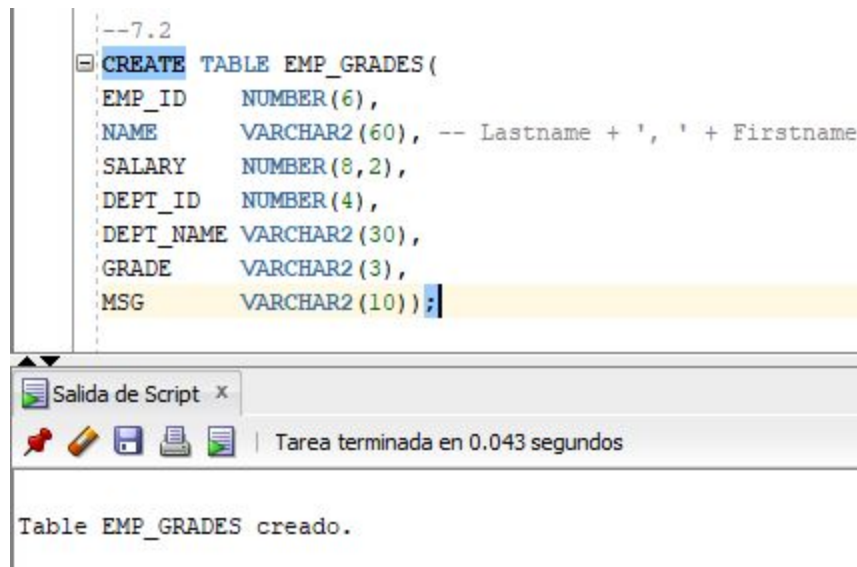
Guarde la información leída de la tabla de empleados en una variable tipo RECORD. Permitir que el código sea reusable.

Hoja de Trabajo	Generador de Consultas
	<pre>--7.1 SET SERVEROUTPUT ON; DECLARE TYPE t_rec_emp IS RECORD (v_emp_id employees.employee_id%TYPE, v_sup_id employees.manager_id%TYPE, v_comm employees.commission_pct%TYPE); v_rec_emp t_rec_emp; v_message VARCHAR2(60); v_min employees.employee_id%TYPE; v_max employees.employee_id%TYPE; BEGIN SELECT MIN(employee_id) INTO v_min FROM employees; SELECT MAX(employee_id) INTO v_max FROM employees; FOR i IN v_min..v_max LOOP BEGIN SELECT employee_id, manager_id, commission_pct INTO v_rec_emp.v_emp_id, v_rec_emp.v_sup_id, v_rec_emp.v_comm FROM employees WHERE employee_id = i; EXCEPTION WHEN NO_DATA_FOUND THEN null; END; v_message := ''; IF v_rec_emp.v_emp_id IS NOT NULL THEN IF v_rec_emp.v_comm IS NOT NULL THEN v_message := ' Cobra comisión'; END IF; IF v_rec_emp.v_sup_id IS NOT NULL THEN v_message := v_message ' Tiene supervisor'; END IF; DBMS_OUTPUT.PUT_LINE('El empleado ' TO_CHAR(v_rec_emp.v_emp_id) v_message); END IF; END LOOP; END; /</pre>

El empleado 166	Cobra comisión	Tiene supervisor
El empleado 167	Cobra comisión	Tiene supervisor
El empleado 168	Cobra comisión	Tiene supervisor
El empleado 169	Cobra comisión	Tiene supervisor
El empleado 170	Cobra comisión	Tiene supervisor
El empleado 171	Cobra comisión	Tiene supervisor
El empleado 172	Cobra comisión	Tiene supervisor
El empleado 173	Cobra comisión	Tiene supervisor
El empleado 174	Cobra comisión	Tiene supervisor
El empleado 175	Cobra comisión	Tiene supervisor
El empleado 176	Cobra comisión	Tiene supervisor
El empleado 177	Cobra comisión	Tiene supervisor
El empleado 178	Cobra comisión	Tiene supervisor
El empleado 179	Cobra comisión	Tiene supervisor
El empleado 180	Tiene supervisor	
El empleado 181	Tiene supervisor	
El empleado 182	Tiene supervisor	
El empleado 183	Tiene supervisor	
El empleado 184	Tiene supervisor	
El empleado 185	Tiene supervisor	
El empleado 186	Tiene supervisor	
El empleado 187	Tiene supervisor	
El empleado 188	Tiene supervisor	
El empleado 189	Tiene supervisor	
El empleado 190	Tiene supervisor	
El empleado 191	Tiene supervisor	

7.2 Primero deben crear una tabla [EMP_GRADES] que tenga la siguiente estructura:

```
EMP_ID  NUMBER(6)
NAME    VARCHAR2(60) -- Lastname + ', ' + Firstname
SALARY  NUMBER(8,2)
DEPT_ID NUMBER(4)
DEPT_NAME VARCHAR2(30)
GRADE   VARCHAR2(3)
MSG     VARCHAR2(10)
```



The screenshot shows a SQL script editor with the following code:

```
--7.2
CREATE TABLE EMP_GRADES (
  EMP_ID    NUMBER(6),
  NAME      VARCHAR2(60), -- Lastname + ', ' + Firstname
  SALARY    NUMBER(8,2),
  DEPT_ID   NUMBER(4),
  DEPT_NAME VARCHAR2(30),
  GRADE     VARCHAR2(3),
  MSG       VARCHAR2(10));
```

Below the script editor, a status bar indicates "Tarea terminada en 0.043 segundos". At the bottom, a message box displays "Table EMP_GRADES creado."

Cree una variable RECORD que contenga el registro completo de empleados, el nombre de departamento y el nivel salarial (para simplificar la construcción de la variable RECORD cree una vista). En el cuerpo del código, deberán leer la información correspondiente de las tablas de empleados, departamentos y niveles salariales y guardar los datos en la variable tipo RECORD. A continuación deberán construir una estructura CASE para evaluar el nivel salarial. Deberán almacenar en una variable un texto dependiendo del nivel salarial. Si es "A" almacenan "Low", si es "B" o "C" almacenan "Medium", si es "D" almacenan "High", de lo contrario almacenan "Elite". Finalmente deberán insertar en la tabla creada inicialmente los valores obtenidos. Fuera del código, revise el contenido de la tabla EMP_GRADES.

Hoja de Trabajo	Generador de Consultas
	<pre> CREATE VIEW vista_empleados AS SELECT e.*, d.department_name, g.grade_level FROM employees e JOIN job_grades g ON e.salary BETWEEN g.lowest_sal AND g.highest_sal JOIN departments d ON (e.department_id = d.department_id); DECLARE v_rec_emp vista_empleados%ROWTYPE; v_msg emp_grades.msg%TYPE; v_min employees.employee_id%TYPE; v_max employees.employee_id%TYPE; BEGIN SELECT MIN(employee_id) INTO v_min FROM employees; SELECT MAX(employee_id) INTO v_max FROM employees; FOR i IN v_min..v_max LOOP BEGIN SELECT e.*, d.department_name, g.grade_level INTO v_rec_emp FROM employees e JOIN job_grades g ON e.salary BETWEEN g.lowest_sal AND g.highest_sal JOIN departments d ON (e.department_id = d.department_id) WHERE employee_id = i; EXCEPTION WHEN NO_DATA_FOUND THEN null; END; CASE WHEN v_rec_emp.grade_level = 'A' THEN v_msg := 'Low'; WHEN v_rec_emp.grade_level = 'B' OR v_rec_emp.grade_level = 'C' THEN v_msg := 'Medium'; WHEN v_rec_emp.grade_level = 'D' THEN v_msg := 'High'; ELSE v_msg := 'Elite'; END CASE; INSERT INTO emp_grades VALUES(v_rec_emp.employee_id, v_rec_emp.last_name ', ' v_rec_emp.first_name, v_rec_emp.salary, v_rec_emp.department_id, v_rec_emp.department_name, v_rec_emp.grade_level, v_msg); END LOOP; END; / </pre>
<div> <div>Salida de Script x</div> <div>Resultado de la Consulta x</div> <div> </div> <div>Tarea terminada en 0.032 segundos</div> </div>	
<p>Procedimiento PL/SQL terminado correctamente.</p>	

Hoja de Trabajo

Generador de Consultas

SELECT * FROM emp_grades;

Salida de Script x

Resultado de la Consulta x

SQL

Se han recuperado 50 filas en 0.007 segundos

	EMP_ID	NAME	SALARY	DEPT_ID	DEPT_NAME	GRADE	MSG
1	100	King, Steven	24000	90	Executive	E	Elite
2	101	Kochhar, Neena	17000	90	Executive	E	Elite
3	102	De Haan, Lex	17000	90	Executive	E	Elite
4	103	Hunold, Alexander	9200	60	IT	C	Medium
5	104	Ernst, Bruce	6200	60	IT	C	Medium
6	105	Austin, David	5000	60	IT	B	Medium
7	106	Pataballa, Valli	5000	60	IT	B	Medium
8	107	Lorentz, Diana	4400	60	IT	B	Medium
9	108	Greenberg, Nancy	12208	100	Finance	D	High
10	109	Faviet, Daniel	9200	100	Finance	C	Medium
11	110	Chen, John	8400	100	Finance	C	Medium
12	111	Sciarra, Ismael	7900	100	Finance	C	Medium
13	112	Urman, Jose Manuel	8000	100	Finance	C	Medium
14	113	Popp, Luis	7100	100	Finance	C	Medium
15	114	Raphaely, Den	11000	30	Purchasing	D	High
16	115	Khoo, Alexander	3100	30	Purchasing	B	Medium

7.3 Construir un código PL/SQL usando una estructura de arreglo asociativa (INDEX-BY table). En la misma deberás guardar los registros de la tabla de empleados para los ID de empleados del 100 al 120. Las siguientes tareas deberán realizarse a partir del arreglo asociativo construido y usando los métodos de tablas INDEX BY:

Hoja de Trabajo	Generador de Consultas
	<pre>--7.3 DECLARE TYPE t_emp_arr IS TABLE OF employees%ROWTYPE INDEX BY PLS_INTEGER; v_emp_arr t_emp_arr; BEGIN FOR i IN 100..120 LOOP BEGIN SELECT * INTO v_emp_arr(i) FROM employees WHERE employee_id = i; EXCEPTION WHEN NO_DATA_FOUND THEN null; END; END LOOP; DBMS_OUTPUT.PUT_LINE('---1---'); DBMS_OUTPUT.PUT_LINE(v_emp_arr.COUNT); DBMS_OUTPUT.PUT_LINE('---2---'); DBMS_OUTPUT.PUT_LINE(v_emp_arr(v_emp_arr.FIRST).last_name); DBMS_OUTPUT.PUT_LINE('---3---'); DBMS_OUTPUT.PUT_LINE(v_emp_arr(v_emp_arr.LAST).last_name); DBMS_OUTPUT.PUT_LINE('---4---'); DBMS_OUTPUT.PUT_LINE(v_emp_arr(104).last_name); DBMS_OUTPUT.PUT_LINE('---5---'); DBMS_OUTPUT.PUT_LINE(v_emp_arr(v_emp_arr.PRIOR(104)).last_name); DBMS_OUTPUT.PUT_LINE('---6---'); DBMS_OUTPUT.PUT_LINE(v_emp_arr(v_emp_arr.NEXT(104)).last_name); DBMS_OUTPUT.PUT_LINE('---7---'); DBMS_OUTPUT.PUT_LINE(CASE WHEN v_emp_arr.EXISTS(8) = true THEN 'Existe' ELSE 'No existe' END); DBMS_OUTPUT.PUT_LINE('---8---'); DBMS_OUTPUT.PUT_LINE(CASE WHEN v_emp_arr.EXISTS(108) = true THEN 'Existe' ELSE 'No existe' END); DBMS_OUTPUT.PUT_LINE('---9---'); v_emp_arr.DELETE(104); DBMS_OUTPUT.PUT_LINE('Registro Borrado'); DBMS_OUTPUT.PUT_LINE('---10---'); v_emp_arr.DELETE(115,117); DBMS_OUTPUT.PUT_LINE('Registros Borrados'); DBMS_OUTPUT.PUT_LINE('---11---'); DBMS_OUTPUT.PUT_LINE(v_emp_arr.COUNT); DBMS_OUTPUT.PUT_LINE('---12---'); FOR i IN 100..120 LOOP BEGIN DBMS_OUTPUT.PUT_LINE(v_emp_arr(i).last_name); EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('No existe');</pre>

```

END;
END LOOP;
DBMS_OUTPUT.PUT_LINE('---13---');
v_emp_arr.DELETE;
DBMS_OUTPUT.PUT_LINE('Registros Borrados');
DBMS_OUTPUT.PUT_LINE('---14---');
DBMS_OUTPUT.PUT_LINE(v_emp_arr.COUNT);
END;
/

```

1. Mostrar en pantalla la cantidad de registros.

```

---1---
21

```

2. Mostrar el apellido del primer registro.

```

---2---
King

```

3. Mostrar el apellido del último registro.

```

---3---
Weiss

```

4. Mostrar el apellido del registro 104.

```

---4---
Ernst

```

5. Mostrar el apellido del registro anterior al 104.

```

---5---
Hunold

```

6. Mostrar el apellido del registro posterior al 104.

```

---6---
Austin

```

7. Mostrar si el registro 8 existe o no.

```

---7---
No existe

```

8. Mostrar si el registro 108 existen o no.

```

---8---
Existe

```

9. Eliminar el registro 104.

```

---9---
Registro Borrado

```

10. Eliminar el rango de registros del 115 al 117.

```

---10---
Registros Borrados

```


11. Mostrar la cantidad de registros.

```
---11---
```

```
17
```

12. Mostrar el apellido de todos los registros del 100 al 120, en caso de que un índice no exista en dicho rango, mostrar "No existe".

```
---12---
```

```
King
```

```
Kochhar
```

```
De Haan
```

```
Hunold
```

```
No existe
```

```
Austin
```

```
Pataballa
```

```
Lorentz
```

```
Greenberg
```

```
Faviet
```

```
Chen
```

```
Sciarra
```

```
Urman
```

```
Popp
```

```
Raphaely
```

```
No existe
```

```
No existe
```

```
No existe
```

```
Himuro
```

```
Colmenares
```

```
Weiss
```

13. Borrar todos los registros.

```
---13---
```

```
Registros Borrados
```

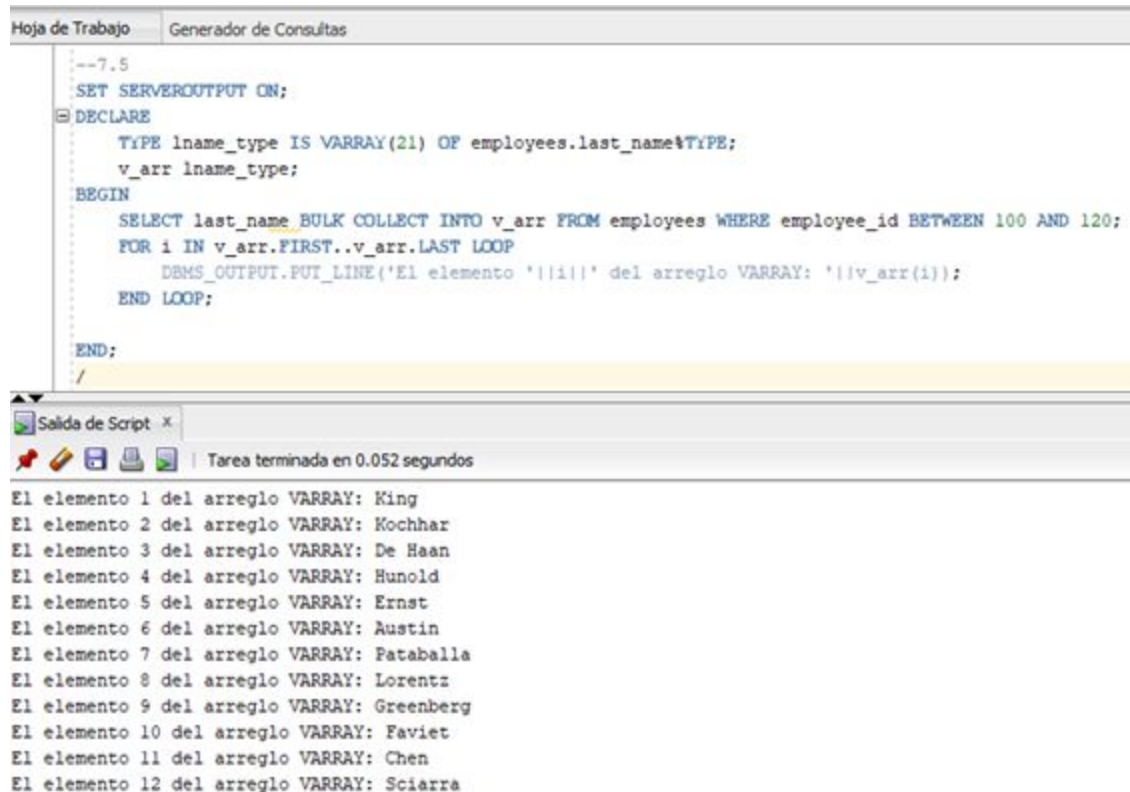
14. Mostrar la cantidad de registros.

```
---14---
```

```
0
```

7.5 Construir un código PL/SQL que cree una variable tipo VARRAY de tamaño 21 para almacenar los apellidos de los ID de empleados 100 al 120. Para llenar el arreglo, usen BULK COLLECT INTO. Mostrar por pantalla los datos obtenidos con el siguiente formato:

Elemento <<# de elemento>> del arreglo VARRAY: <<Apellido>>



```
--7.5
SET SERVEROUTPUT ON;
DECLARE
  TYPE lname_type IS VARRAY(21) OF employees.last_name%TYPE;
  v_arr lname_type;
BEGIN
  SELECT last_name BULK COLLECT INTO v_arr FROM employees WHERE employee_id BETWEEN 100 AND 120;
  FOR i IN v_arr.FIRST..v_arr.LAST LOOP
    DBMS_OUTPUT.PUT_LINE('El elemento '||i||' del arreglo VARRAY: '||v_arr(i));
  END LOOP;
END;
```

Salida de Script x

Tarea terminada en 0.052 segundos

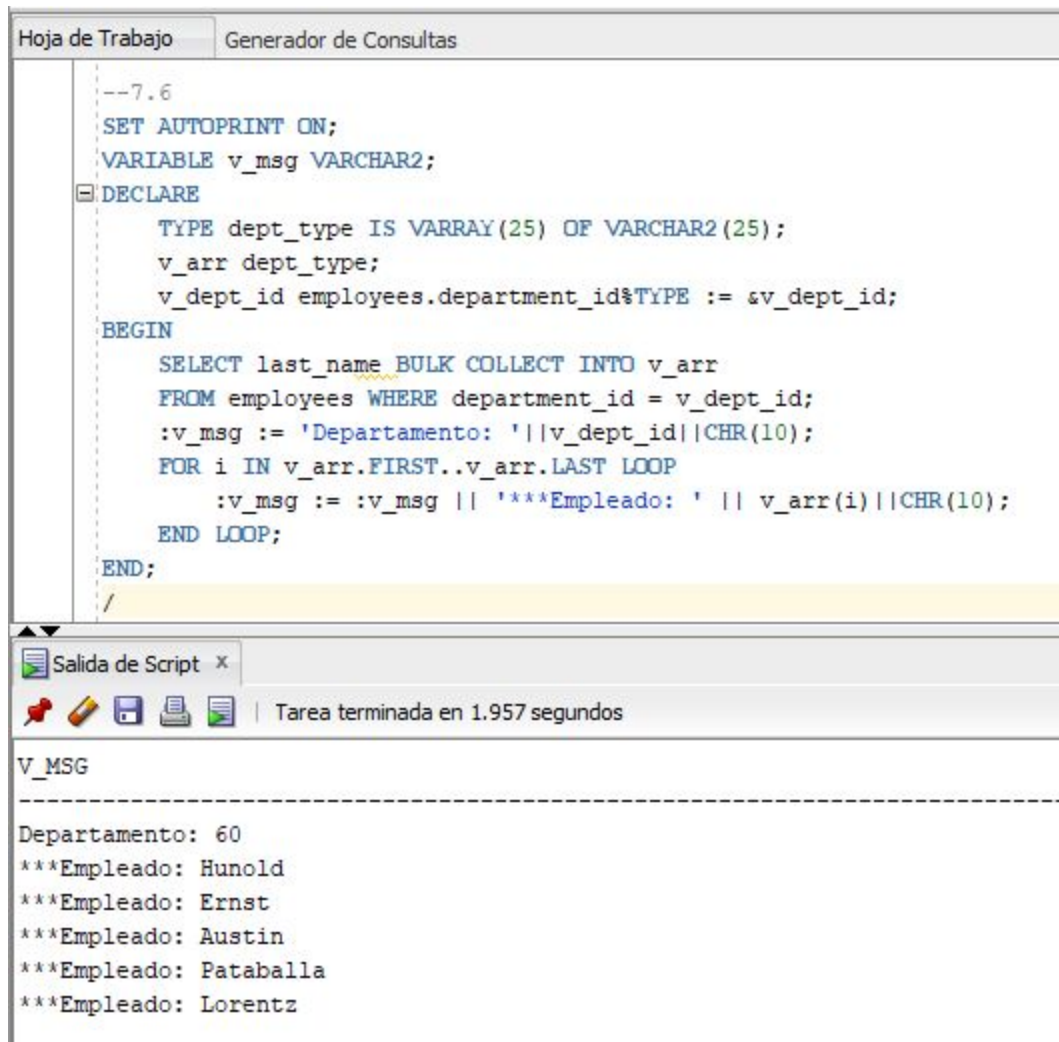
El elemento 1 del arreglo VARRAY: King
El elemento 2 del arreglo VARRAY: Kochhar
El elemento 3 del arreglo VARRAY: De Haan
El elemento 4 del arreglo VARRAY: Hunold
El elemento 5 del arreglo VARRAY: Ernst
El elemento 6 del arreglo VARRAY: Austin
El elemento 7 del arreglo VARRAY: Pataballa
El elemento 8 del arreglo VARRAY: Lorentz
El elemento 9 del arreglo VARRAY: Greenberg
El elemento 10 del arreglo VARRAY: Faviet
El elemento 11 del arreglo VARRAY: Chen
El elemento 12 del arreglo VARRAY: Sciarra

7.6 Crear un tipo de dato VARRAY (dept_type) que sea VARCHAR2(25). El tamaño del arreglo debe ser de 25. Luego deberán crear una variable que use el tipo VARRAY creado anteriormente. En dicha variable almacenen todos los apellidos de los empleados que trabajan en un departamento dado por pantalla. Mostrar la información de la siguiente forma:

Departamento: <<ID del departamento>>

*** Empleado: <<Apellido>>

Usen los comandos para definir una variable de ambiente y para no visualizar el texto de un comando.



The screenshot shows the Oracle SQL Developer interface. The top pane, titled 'Generador de Consultas', contains a PL/SQL script. The script defines a VARRAY type, declares a variable, and uses a loop to collect employee last names for a specific department. The bottom pane, titled 'Salida de Script', shows the output of the script, which displays the department ID and a list of employee last names.

```
--7.6
SET AUTOPRINT ON;
VARIABLE v_msg VARCHAR2;
DECLARE
    TYPE dept_type IS VARRAY(25) OF VARCHAR2(25);
    v_arr dept_type;
    v_dept_id employees.department_id%TYPE := &v_dept_id;
BEGIN
    SELECT last_name BULK COLLECT INTO v_arr
    FROM employees WHERE department_id = v_dept_id;
    :v_msg := 'Departamento: ' || v_dept_id || CHR(10);
    FOR i IN v_arr.FIRST..v_arr.LAST LOOP
        :v_msg := :v_msg || '***Empleado: ' || v_arr(i) || CHR(10);
    END LOOP;
END;
/
```

Salida de Script x

Tarea terminada en 1.957 segundos

V_MSG

Departamento: 60

***Empleado: Hunold

***Empleado: Ernst

***Empleado: Austin

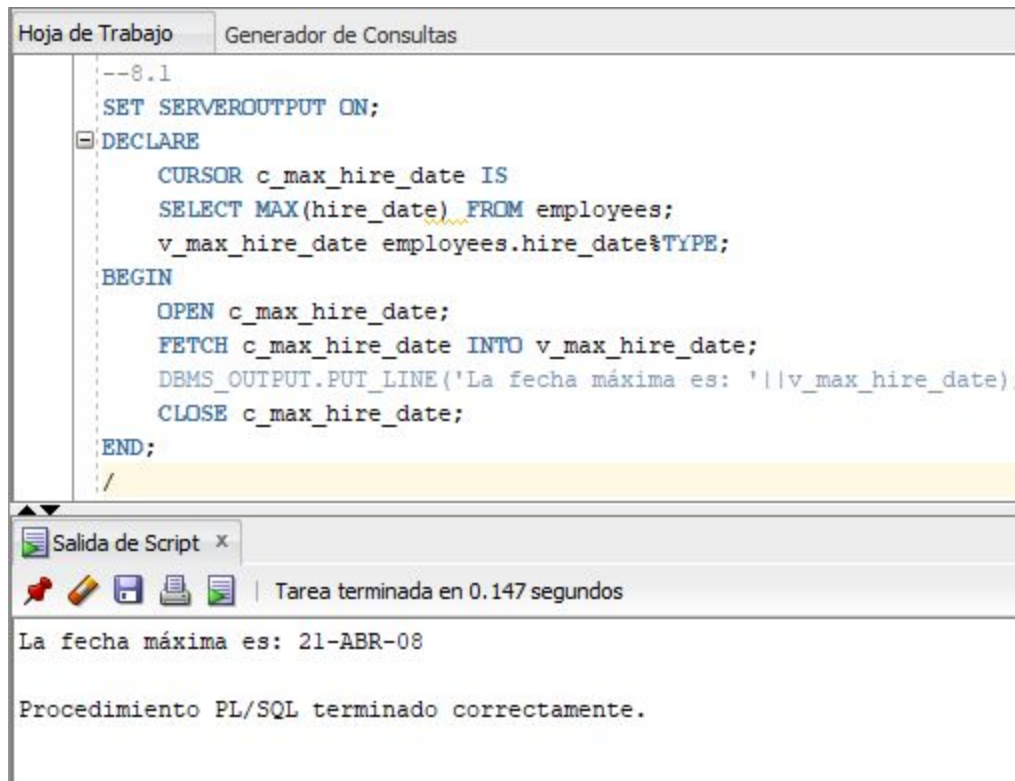
***Empleado: Pataballa

***Empleado: Lorentz

II. Realice los siguientes ejercicios luego de estudiar el documento D64254GC11_les08.ppt:

8.1 Construya un código PL/SQL que usando un CURSOR explícito busque la fecha máxima de contratación en la tabla de empleados. Use FETCH para leer cada fila del cursor. Analice cuántos registros retornará el cursor. Basado en su análisis elija la mejor forma de construir el código. Muestre el siguiente mensaje:

La máxima fecha es <<Fecha>>



The screenshot shows a PL/SQL development environment with two main windows. The top window, titled 'Hoja de Trabajo' and 'Generador de Consultas', contains a PL/SQL script. The script starts with a comment '--8.1', followed by 'SET SERVEROUTPUT ON;'. It then declares a cursor 'c_max_hire_date' and a variable 'v_max_hire_date' of type 'employees.hire_date%TYPE'. The script opens the cursor, fetches the maximum hire date into the variable, outputs the result using 'DBMS_OUTPUT.PUT_LINE', closes the cursor, and ends with 'END;'. The bottom window, titled 'Salida de Script', shows the execution results. It indicates that the task was completed in 0.147 seconds and displays the output: 'La fecha máxima es: 21-ABR-08' and 'Procedimiento PL/SQL terminado correctamente.'

```
--8.1
SET SERVEROUTPUT ON;
DECLARE
  CURSOR c_max_hire_date IS
    SELECT MAX(hire_date) FROM employees;
  v_max_hire_date employees.hire_date%TYPE;
BEGIN
  OPEN c_max_hire_date;
  FETCH c_max_hire_date INTO v_max_hire_date;
  DBMS_OUTPUT.PUT_LINE('La fecha máxima es: '||v_max_hire_date);
  CLOSE c_max_hire_date;
END;
```

Salida de Script x

Tarea terminada en 0.147 segundos

La fecha máxima es: 21-ABR-08

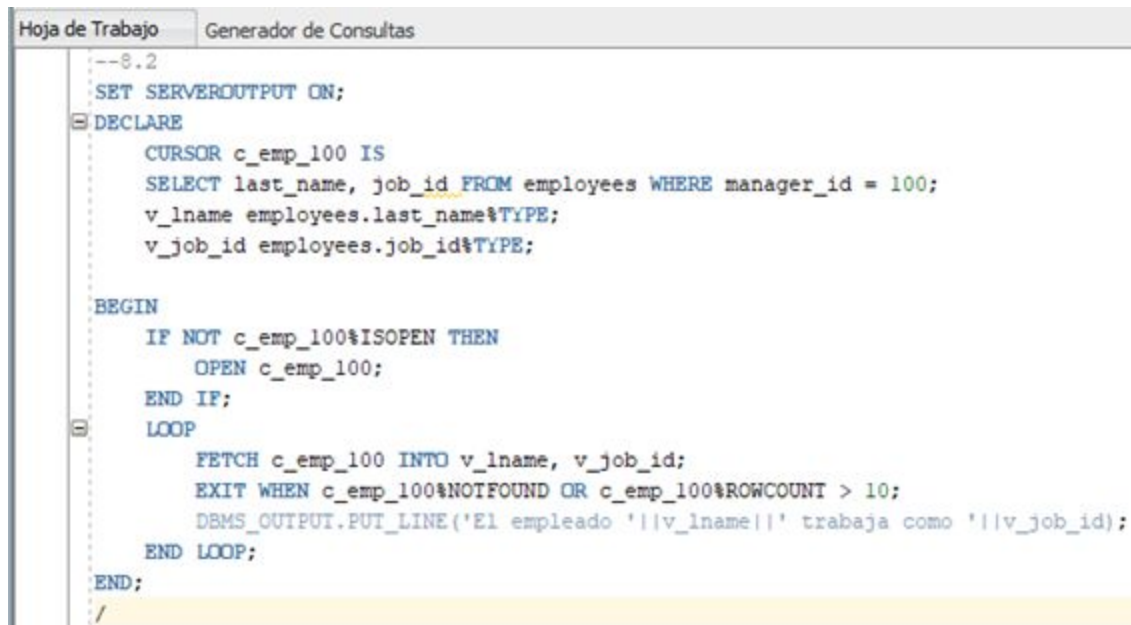
Procedimiento PL/SQL terminado correctamente.

Como las funciones de múltiples filas devuelven un solo valor por sí mismas, el Cursor sólo tendrá un valor, por tanto, con un solo FETCH sin iterar es suficiente para mostrar el resultado.

8.2 Construya un código PL/SQL usando un CURSOR explícito que busque el apellido y la posición de todos los empleados que se reportan al supervisor 100. Mostrar el siguiente mensaje sólo para los primeros 10 empleados:

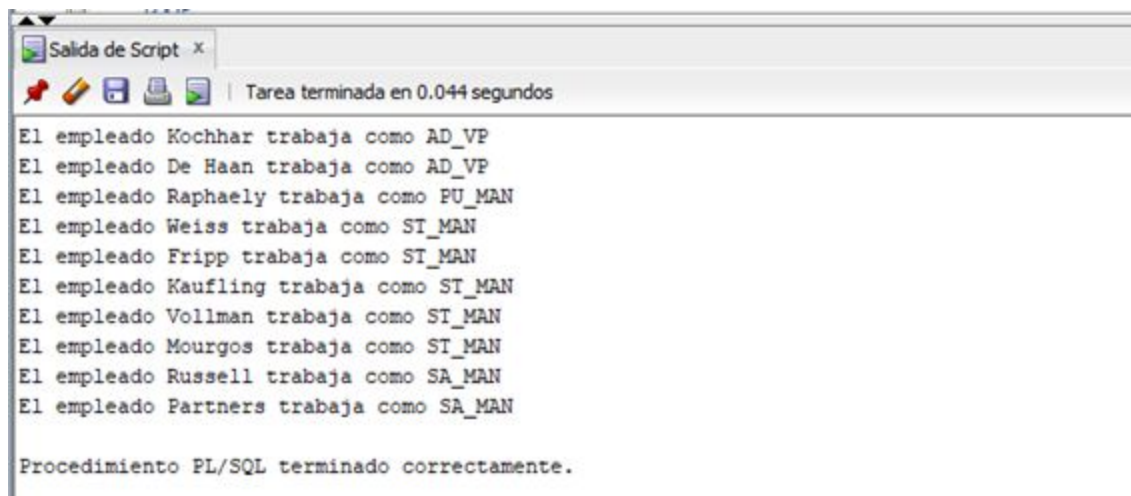
El empleado <<Apellido>> trabaja como <<Posición>>

Utilice FETCH, %NOTFOUND y %ROWCOUNT. Ante de abrir el CURSOR valide que el mismo no se encuentre abierto usando %ISOPEN.



```
--8.2
SET SERVEROUTPUT ON;
DECLARE
    CURSOR c_emp_100 IS
        SELECT last_name, job_id FROM employees WHERE manager_id = 100;
    v_lname employees.last_name%TYPE;
    v_job_id employees.job_id%TYPE;

BEGIN
    IF NOT c_emp_100%ISOPEN THEN
        OPEN c_emp_100;
    END IF;
    LOOP
        FETCH c_emp_100 INTO v_lname, v_job_id;
        EXIT WHEN c_emp_100%NOTFOUND OR c_emp_100%ROWCOUNT > 10;
        DBMS_OUTPUT.PUT_LINE('El empleado '||v_lname||' trabaja como '||v_job_id);
    END LOOP;
END;
```



```
Salida de Script x
Tarea terminada en 0.044 segundos

El empleado Kochhar trabaja como AD_VP
El empleado De Haan trabaja como AD_VP
El empleado Raphaely trabaja como PU_MAN
El empleado Weiss trabaja como ST_MAN
El empleado Fripp trabaja como ST_MAN
El empleado Kaufling trabaja como ST_MAN
El empleado Vollman trabaja como ST_MAN
El empleado Mourgos trabaja como ST_MAN
El empleado Russell trabaja como SA_MAN
El empleado Partners trabaja como SA_MAN

Procedimiento PL/SQL terminado correctamente.
```

8.3 Construya un código PL/SQL usando cursor FOR LOOP que lea los ID del empleado, apellido, salario, nombre de departamento y nivel salarial. Relacione sus datos con la tabla JOB_GRADES para conocer su nivel salarial. Si el nivel del empleado es 'A' se desea desplegar 'Low', si es 'B' o 'C' que despliegue 'Medium', si es 'D' que despliegue 'High' y si no es ninguno de estos niveles, que despliegue 'Elite'. Ordene los datos por el nivel salarial. Muestre la información por pantalla como sigue:

Id: <<Id del empleado>> * Apellido: <<Apellido>> *

Departamento: <<Nombre del Departamento>> * Salary: <<Salario>> * Nivel: <<Nivel>>

El salario debe salir con el siguiente formato: 'fm\$99,999'

```

Hoja de Trabajo  Generador de Consultas
--8.3
DECLARE
    CURSOR c_emp IS
        SELECT e.employee_id,e.last_name, e.salary, d.department_name, g.grade_level
        FROM employees e JOIN job_grades g ON e.salary BETWEEN g.lowest_sal AND g.highest_sal
        JOIN departments d ON (e.department_id = d.department_id) ORDER BY e.employee_id;
    v_level VARCHAR2(7);
BEGIN
    FOR v_emp IN c_emp LOOP
        CASE
            WHEN v_emp.grade_level = 'A' THEN
                v_level := 'Low';
            WHEN v_emp.grade_level = 'B' OR v_emp.grade_level = 'C' THEN
                v_level := 'Medium';
            WHEN v_emp.grade_level = 'D' THEN
                v_level := 'High';
            ELSE
                v_level := 'Elite';
            END CASE;
        DBMS_OUTPUT.PUT_LINE('ID: '||v_emp.employee_id||' * Apellido: '||v_emp.last_name||' * Departamento: '
        ||v_emp.department_name||' * Salario: '||
        TO_CHAR(v_emp.salary, 'fm$99,999')||' * Nivel: '||v_level);
    END LOOP;
END;
/

```

```

Salida de Script x
Tarea terminada en 0.033 segundos

ID: 100 * Apellido: King * Departamento: Executive * Salario: $24,000 * Nivel: Elite
ID: 101 * Apellido: Kochhar * Departamento: Executive * Salario: $17,000 * Nivel: Elite
ID: 102 * Apellido: De Haan * Departamento: Executive * Salario: $17,000 * Nivel: Elite
ID: 103 * Apellido: Munold * Departamento: IT * Salario: $9,200 * Nivel: Medium
ID: 104 * Apellido: Ernst * Departamento: IT * Salario: $6,200 * Nivel: Medium
ID: 105 * Apellido: Austin * Departamento: IT * Salario: $5,000 * Nivel: Medium
ID: 106 * Apellido: Pataballa * Departamento: IT * Salario: $5,000 * Nivel: Medium
ID: 107 * Apellido: Lorentz * Departamento: IT * Salario: $4,400 * Nivel: Medium
ID: 108 * Apellido: Greenberg * Departamento: Finance * Salario: $12,208 * Nivel: High
ID: 109 * Apellido: Faviet * Departamento: Finance * Salario: $9,200 * Nivel: Medium
ID: 110 * Apellido: Chen * Departamento: Finance * Salario: $8,400 * Nivel: Medium
ID: 111 * Apellido: Sciarra * Departamento: Finance * Salario: $7,900 * Nivel: Medium
ID: 112 * Apellido: Urman * Departamento: Finance * Salario: $8,000 * Nivel: Medium
ID: 113 * Apellido: Popp * Departamento: Finance * Salario: $7,100 * Nivel: Medium
ID: 114 * Apellido: Raphaely * Departamento: Purchasing * Salario: $11,000 * Nivel: High
ID: 115 * Apellido: Khoo * Departamento: Purchasing * Salario: $3,100 * Nivel: Medium
ID: 116 * Apellido: Baida * Departamento: Purchasing * Salario: $2,900 * Nivel: Low
ID: 117 * Apellido: Tobias * Departamento: Purchasing * Salario: $2,800 * Nivel: Low

```


8.4 Construya un código PL/SQL usando un cursor FOR LOOP con subquery. Muestre el ID, apellido y país de todos los empleados que trabajen en Canadá y United Kingdom. Ordene la data por país y apellido de manera ascendente. El formato de la salida debe ser como sigue:

ID *** Apellido *** País

```
set serveroutput on
begin
for emp_inf in (
select employee_id , last_name, country_name from employees
  join departments using (department_id)
  join locations using (location_id)
  join countries using (country_id)
 where country_id = ('CA')OR country_id = ('UK')
 order by country_name ASC, last_name ASC)
loop
DBMS_OUTPUT.PUT_LINE(emp_inf.employee_id||'***'||emp_inf.last_name||'***'||emp_inf.country_name);
end loop;
end;
/
```

Salida de Script x

Tarea terminada en 0.063 segundos

```
202***Fay***Canada
201***Hartstein***Canada
174***Abel***United Kingdom
166***Ande***United Kingdom
167***Banda***United Kingdom
172***Bates***United Kingdom
151***Bernstein***United Kingdom
169***Bloom***United Kingdom
154***Cambrault***United Kingdom
148***Cambrault***United Kingdom
160***Doran***United Kingdom
147***Errazuriz***United Kingdom
170***Fox***United Kingdom
```

8.5 Construya un código PL/SQL con un cursor que busque el salario total y el salario promedio que trabajan en un departamento dado. Al momento de abrir el cursor pase los valores 20 y 30 como parámetros. Muestre la información como sigue:

Departamento(<<ID>>): Salario total <<total>> *** Salario promedio <<promedio>>
Los valores del salario deben salir con el siguiente formato: 'fm\$99,999'

```
deptno number (15);
CURSOR c_emp (deptno NUMBER) is
select department_id, sum(salary), avg(salary) from employees
where department_id = deptno
GROUP BY department_id
;
var_Sum EMPLOYEES.SALARY%TYPE;
var_avg EMPLOYEES.SALARY%TYPE;
begin
    open c_emp (20);

    FETCH c_emp INTO deptno, var_Sum, var_avg;
    DBMS_OUTPUT.PUT_LINE('Departamento '||deptno||': '||'Salario total '||to_char( var_Sum, 'fm$99,999' )
    ||' *** '||'Salario promedio '||to_char( var_avg, 'fm$99,999' ));

    close c_emp;

    open c_emp (30);

    FETCH c_emp INTO deptno, var_Sum, var_avg;
    DBMS_OUTPUT.PUT_LINE('Departamento '||deptno||': '||'Salario total '||to_char( var_Sum, 'fm$99,999' )
    ||' *** '||'Salario promedio '||to_char( var_avg, 'fm$99,999' ));
```

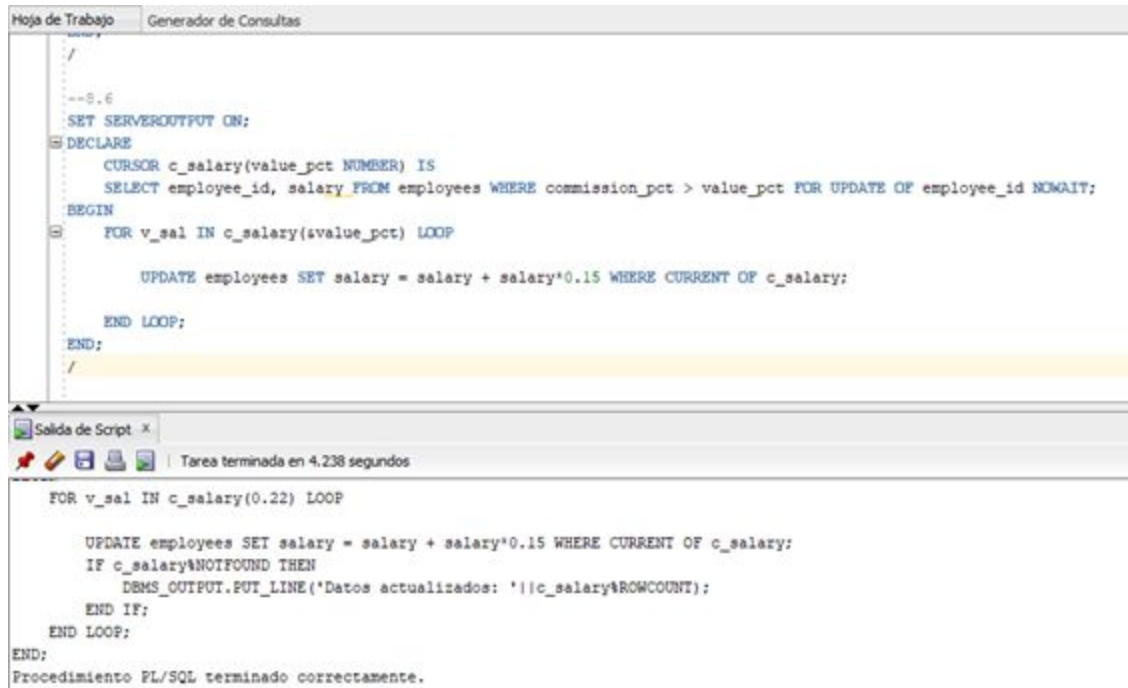
Salida de Script x

Tarea terminada en 0.057 segundos

Departamento 20: Salario total \$19,000 *** Salario promedio \$9,500
Departamento 30: Salario total \$29,383 *** Salario promedio \$4,897

Procedimiento PL/SQL terminado correctamente.

8.6 Construya un código PL/SQL que busque el ID y salario de los empleados que ganan una comisión mayor a un valor dado y bloquee el salario para que no sea actualizado por otra sesión. En la sección de ejecución actualice el salario de los empleados que vengan en el cursor en un 15%. Para realizar esta práctica use un cursor FOR LOOP con parámetros, FOR UPDATE OF, NOWAIT y CURRENT OF.



The screenshot shows the Oracle SQL Developer interface. The top pane, titled 'Hoja de Trabajo' and 'Generador de Consultas', contains a PL/SQL script. The script declares a cursor 'c_salary' that selects employee IDs and salaries where the commission percentage is greater than a specified value. It then uses a 'FOR UPDATE OF' clause to lock the salary column for the selected rows. A 'FOR LOOP' iterates over the cursor, updating each employee's salary by 15% using the 'CURRENT OF' clause. The bottom pane, titled 'Salida de Script', shows the execution output, indicating that the task was completed in 4.238 seconds and that the salary was updated for 22 employees.

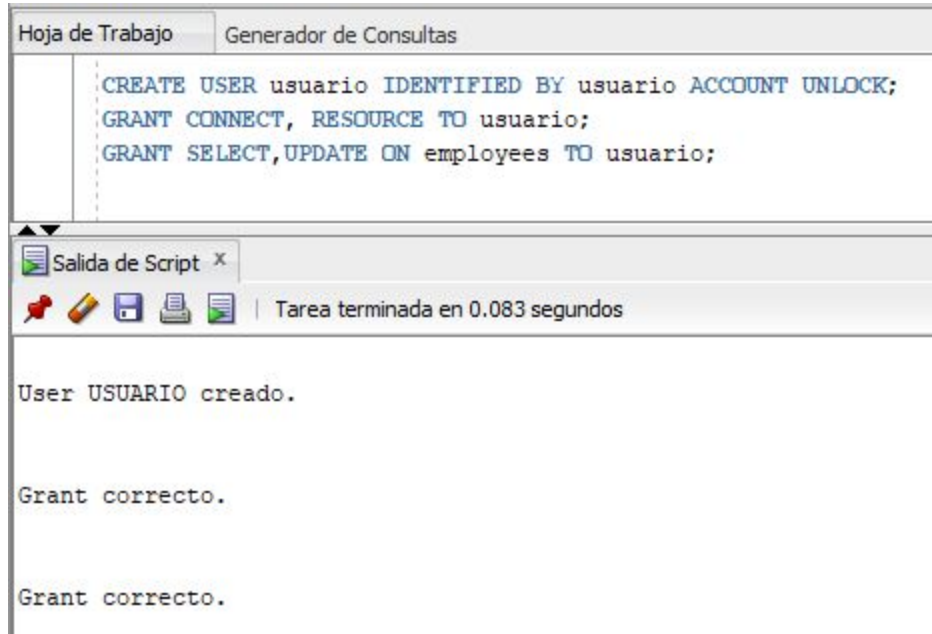
```
--8.6
SET SERVEROUTPUT ON;
DECLARE
    CURSOR c_salary(value_pct NUMBER) IS
        SELECT employee_id, salary FROM employees WHERE commission_pct > value_pct FOR UPDATE OF employee_id NOWAIT;
BEGIN
    FOR v_sal IN c_salary(:value_pct) LOOP
        UPDATE employees SET salary = salary + salary*0.15 WHERE CURRENT OF c_salary;
    END LOOP;
END;
```

Tarea terminada en 4.238 segundos

```
FOR v_sal IN c_salary(0.22) LOOP

    UPDATE employees SET salary = salary + salary*0.15 WHERE CURRENT OF c_salary;
    IF c_salary%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('Datos actualizados: '||c_salary%ROWCOUNT);
    END IF;
END LOOP;
END;
Procedimiento PL/SQL terminado correctamente.
```

8.7 Realice la práctica 8.6 usando WAIT por 60 segundos. Cree un usuario al cual se le dé privilegios de actualización sobre la tabla EMPLOYEES de HR. Actualice el salario en un 15% al empleado 146 (no asiente los cambios). Corra el programa y comente qué pasa. Luego en la otra sesión, deshaga el cambio y comente el comportamiento del programa.



The screenshot shows the Oracle SQL Developer interface. At the top, there are two tabs: 'Hoja de Trabajo' (Worksheet) and 'Generador de Consultas' (Query Generator). The 'Hoja de Trabajo' tab is active, displaying a SQL script with three lines: `CREATE USER usuario IDENTIFIED BY usuario ACCOUNT UNLOCK;`, `GRANT CONNECT, RESOURCE TO usuario;`, and `GRANT SELECT, UPDATE ON employees TO usuario;`. Below the script editor, there is a 'Salida de Script' (Script Output) window. This window has a title bar with a close button and a status bar that reads 'Tarea terminada en 0.083 segundos' (Task completed in 0.083 seconds). The output area contains three lines of text: 'User USUARIO creado.' (User USUARIO created.), 'Grant correcto.' (Grant correct.), and 'Grant correcto.' (Grant correct.).

```
CREATE USER usuario IDENTIFIED BY usuario ACCOUNT UNLOCK;
GRANT CONNECT, RESOURCE TO usuario;
GRANT SELECT, UPDATE ON employees TO usuario;
```

Salida de Script x

Tarea terminada en 0.083 segundos

User USUARIO creado.

Grant correcto.

Grant correcto.

```
Hoja de Trabajo  Generador de Consultas

--8.7
SET SERVEROUTPUT ON;
DECLARE
  CURSOR c_salary(value_pct NUMBER) IS
    SELECT employee_id, salary FROM employees WHERE commission_pct > value_pct
    FOR UPDATE OF employee_id WAIT 60;
BEGIN
  FOR v_sal IN c_salary(:value_pct) LOOP
    UPDATE employees SET salary = salary + salary*0.15 WHERE CURRENT OF c_salary;

  END LOOP;
END;
/

Salida de Script x
Tarea terminada en 63.358 segundos
FOR v_sal IN c_salary(:value_pct) LOOP

    UPDATE employees SET salary = salary + salary*0.15 WHERE CURRENT OF c_salary;

END LOOP;
END;
Informe de error -
ORA-30006: resource busy; acquire with WAIT timeout expired
ORA-06512: at line 3
ORA-06512: at line 5
30006. 00000 - "resource busy; acquire with WAIT timeout expired"
*Cause:      The requested resource is busy.
*Action:     Retry the operation later.
```

El programa estuvo esperando por 60 segundos en la espera de que el recurso se liberara para efectuar el UPDATE. No se pudo efectuar porque no se habían asentado los cambios realizados por usuario.

Tras hacer el rollback desde usuario, los recursos fueron liberados y el programa pudo ejecutarse correctamente.

Hoja de Trabajo	Generador de Consultas
<pre>--8.7 SET SERVEROUTPUT ON; DECLARE CURSOR c_salary(value_pct NUMBER) IS SELECT employee_id, salary FROM employees WHERE commission_pct > value_pct FOR UPDATE OF employee_id WAIT 60; BEGIN FOR v_sal IN c_salary(0.22) LOOP UPDATE employees SET salary = salary + salary*0.15 WHERE CURRENT OF c_salary; END LOOP; END; /</pre>	
<div>Salida de Script x</div> <div>Tarea terminada en 3.01 segundos</div> <pre>DECLARE CURSOR c_salary(value_pct NUMBER) IS SELECT employee_id, salary FROM employees WHERE commission_pct > value_pct FOR UPDATE OF employee_id WAIT 60; BEGIN FOR v_sal IN c_salary(0.22) LOOP UPDATE employees SET salary = salary + salary*0.15 WHERE CURRENT OF c_salary; END LOOP; END; Procedimiento PL/SQL terminado correctamente.</pre>	