

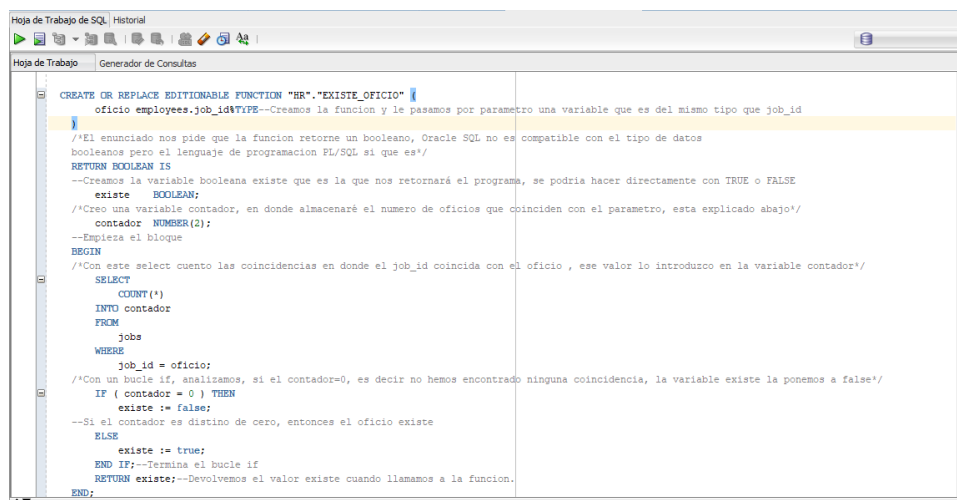
## Actividad 11. Tarea Individual. Trabajar con procedimientos

La actividad la comenzaríamos implementando la función para comprobar que al introducir un oficio nuevo, este oficio existe. Lo he realizado con un contador que cuenta las coincidencias al introducir el nuevo oficio con los oficios existentes, si el contador es cero, significa que no encuentra coincidencias. Quizás no es la manera mas correcta de hacerlo, pero el desconocimiento por ahora del código pl/Sql me hizo buscarme una solución.

```
-----
-- DDL for Function EXISTE_OFICIO
-----

CREATE OR REPLACE EDITIONABLE FUNCTION "HR"."EXISTE_OFICIO" (
    oficio employees.job_id%TYPE--Creamos la funcion y le pasamos por parametro una variable
    que es del mismo tipo que job_id
)
/*El enunciado nos pide que la funcion retorne un booleano, Oracle SQL no es compatible con
el tipo de datos
booleanos pero el lenguaje de programacion PL/SQL si que es*/
RETURN BOOLEAN IS
    --Creamos la variable booleana existe que es la que nos retornará el programa, se podria
    hacer directamente con TRUE o FALSE
    existe BOOLEAN;
    /*Creo una variable contador, en donde almacenaré el numero de oficios que coinciden con el
    parametro, esta explicado abajo*/
    contador NUMBER(2);
    --Empieza el bloque
BEGIN
    /*Con este select cuento las coincidencias en donde el job_id coincida con el oficio , ese
    valor lo introduzco en la variable contador*/
    SELECT
        COUNT(*)
    INTO contador
    FROM
        jobs
    WHERE
        job_id = oficio;
    /*Con una sentencia if, analizamos, si el contador=0, es decir no hemos encontrado ninguna
    coincidencia, la variable existe la ponemos a false*/
    IF ( contador = 0 ) THEN
        existe := false;
    --Si el contador es distinto de cero, entonces el oficio existe
    ELSE
        existe := true;
    END IF;--Termina la sentencia if
    RETURN existe;--Devolvemos el valor existe cuando llamamos a la funcion.
END;
```

/



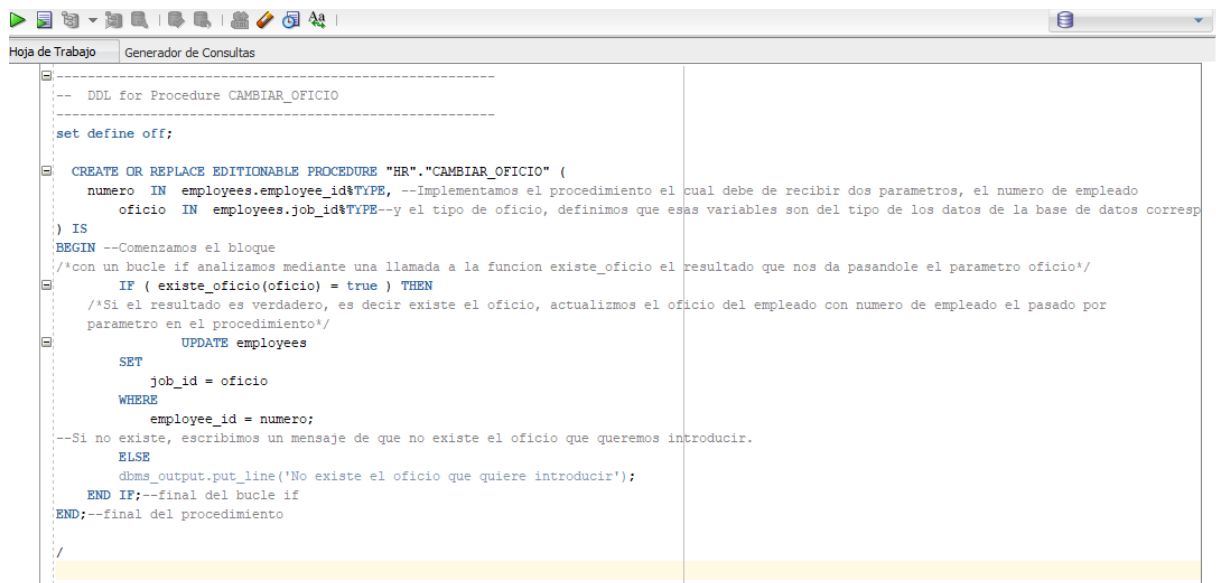
## Actividad 11. Tarea Individual. Trabajar con procedimientos

Cuando ya tenemos nuestra función implementada, podemos pasar a implementar nuestro procedimiento que recibirá dos parámetros de entrada.

```
-----
-- DDL for Procedure CAMBIAR_OFICIO
-----
set define off;

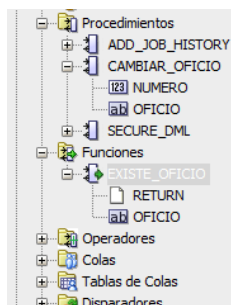
CREATE OR REPLACE EDITIONABLE PROCEDURE "HR"."CAMBIAR_OFICIO" (
    numero IN employees.employee_id%TYPE, --Implementamos el procedimiento el cual debe de
    recibir dos parametros, el numero de empleado
    oficio IN employees.job_id%TYPE--y el tipo de oficio, definimos que esas variables son
    del tipo de los datos de la base de datos correspondientes
) IS
BEGIN --Comenzamos el bloque
/*con una sentencia if analizamos mediante una llamada a la funcion existe_oficio el
resultado que
nos da pasandole el parametro oficio*/
    IF ( existe_oficio(oficio) = true ) THEN
        /*Si el resultado es verdadero, es decir existe el oficio, actualizmos el oficio del
empleado con numero de empleado el pasado por
parametro en el procedimiento*/
            UPDATE employees
            SET
                job_id = oficio
            WHERE
                employee_id = numero;
--Si no existe, escribimos un mensaje de que no existe el oficio que queremos introducir.
    ELSE
        dbms_output.put_line('No existe el oficio que quiere introducir');
    END IF;--final de la sentencia if
END;--final del procedimiento
```

/



## Actividad 11. Tarea Individual. Trabajar con procedimientos

En el Gestor de bases de datos ya podemos ver que se ha creado tanto la función como el procedimiento



A continuación, solo nos quedaría implementar el bloque anónimo para ejecutar el procedimiento realizado. Aquí he tenido en consideración implementar un bucle if por si al usuario se le ocurre introducir un numero de empleado erróneo. Me gustaría haber hecho un bucle while que repitiera todo el rato la introducción por teclado del número de empleado si se introduce uno erróneo pero no he encontrado la forma.

```
SET SERVEROUTPUT ON;
/*Este es nuestro bloque anonimo para cambiar el oficio de un trabajador mediante un
procedimiento*/
DECLARE--Declaramos nuestras variables que tienen que tener el tipo de dato de los datos
correspondientes en la base de datos
    numero_trabajador employees.employee_id%TYPE;
    oficio              employees.job_id%TYPE;
    oficio_viejo        employees.job_id%TYPE;--Vamos a guardar en una variable su oficio antiguo
para visualizarlo al final
    /*Declaramos unas variables minimo y maximo para utilizarlas como limites en el bucle que
implementamos a continuacion*/
    minimo              NUMBER(4);
    maximo              NUMBER(4);
BEGIN
    /*Introducimos en las variables minimo y maximo los limites de los valores de numero de
empleado para el bucle if que implementamos a continuacion.*/
    SELECT
        MIN(employee_id),
        MAX(employee_id)
    INTO
        minimo,
        maximo
    FROM
        employees;

    /*Los valores de los argumentos tienen que ser recogidos en variables, pero en variables de
sustitucion*/
    numero_trabajador := &dame_numero_de_empleado;

    /*Hacemos esta sentencia if por si al usuario se le ocurre introducir un numero de empleado
que
no existe. Podriamos
haber hecho otra funcion para comprobarlo*/
    IF ( numero_trabajador < minimo OR numero_trabajador > maximo )
    THEN--Si el trabajador no existe sacamos un mensaje por pantalla.
        dbms_output.put_line('El numero de trabajador
'||numero_trabajador||' no existe.FIN DE PROGRAMA');
    ELSE--si el numero del trabajador existe , cuando el compilador analiza los datos
introducidos ejecuta el procedimiento
        SELECT
            job_id
        INTO oficio_viejo
        FROM
            employees
        WHERE
            employee_id = numero_trabajador;--guardamos el oficio antiguo del empleado en la
variable
```

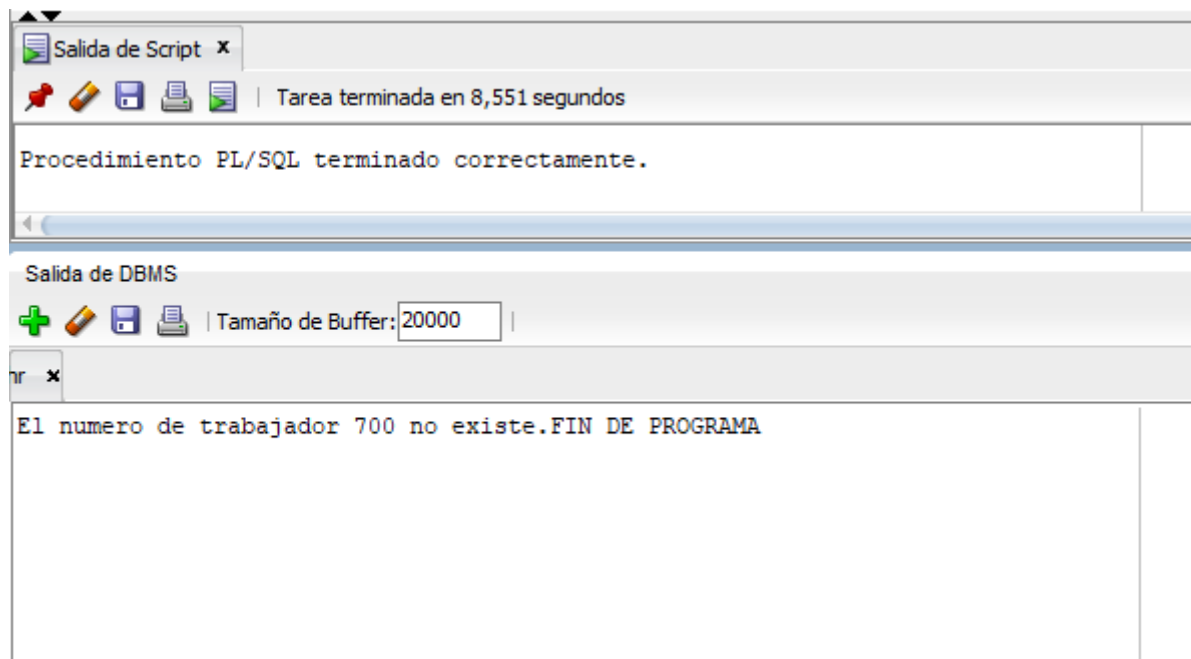
## Actividad 11. Tarea Individual. Trabajar con procedimientos

```
        oficio := '&Dame_oficio_nuevo_del_empleado';
--Se ejecuta el procedimiento con los valores pasados mediante las variables de
sustitucion.
        cambiar_oficio(numero_trabajador,
oficio);
        IF ( existe_oficio(oficio) = true ) THEN /*Para presentar los cambios en
consola, como quiero hacer una comparacion entre oficio nuevo y oficio viejo,
        utilizo la funcion existe_oficio para ver si existe el oficio que he introducido en
la variable de sustitucion. El procedimiento ya lo comprueba,
        pero para esta presentación me hace falta.
        /*Vemos por consola los cambios*/
                dbms_output.put_line('El trabajador numero '
                || numero_trabajador
                || ' ha cambiado su oficio de '
                || oficio_viejo
                || ' a '
                || oficio);

        dbms_output.put_line('FIN'); --Fin del programa
        ELSE --Si el oficio no existe, informamos de que el trabajador sigue con el
mismo oficio
                dbms_output.put_line('El trabajador '||numero_trabajador||' continua
con el mismo oficio');
        dbms_output.put_line('FIN');
        END IF;--Fin de la sentencia if para la presentacion de los datos
        END IF;--final de la sentencia if EXCEPTION
        WHEN OTHERS THEN --Introducimos el mensaje de error para cualquier excepcion que se produzca
        dbms_output.put_line('ERROR GENERAL');
END;
```

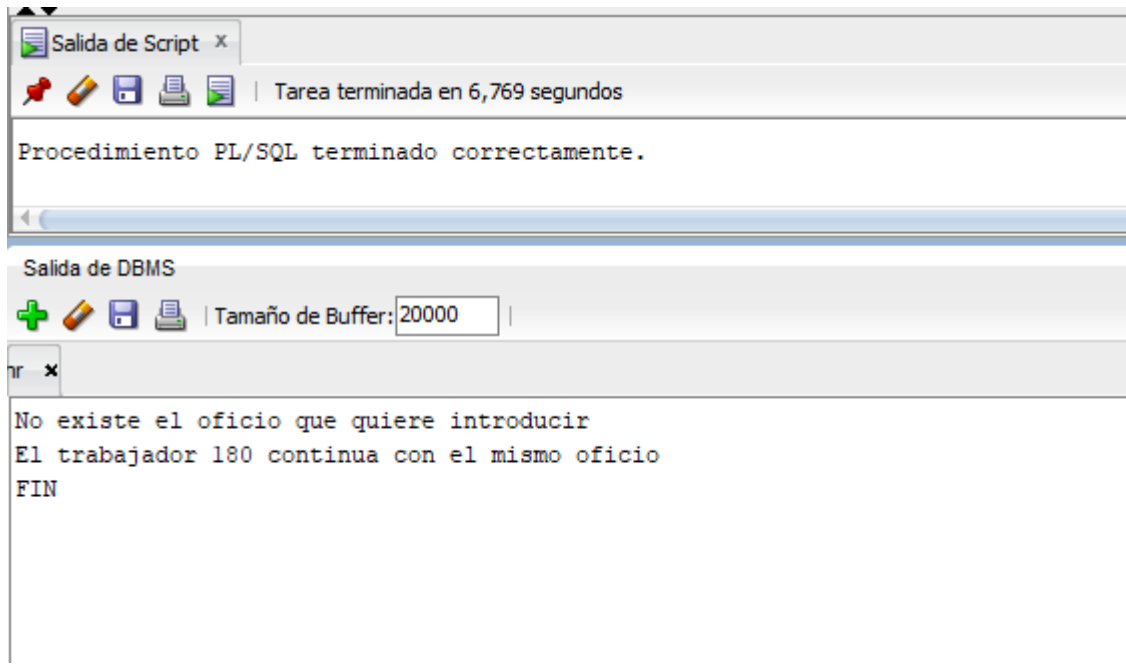
### EJECUCIÓN CASOS PROBABLES

#### 1-Introducción de un numero de empleado fuera de rango:



## Actividad 11. Tarea Individual. Trabajar con procedimientos

### 2-Introducción de un numero de trabajador correcto pero un oficio que no existe



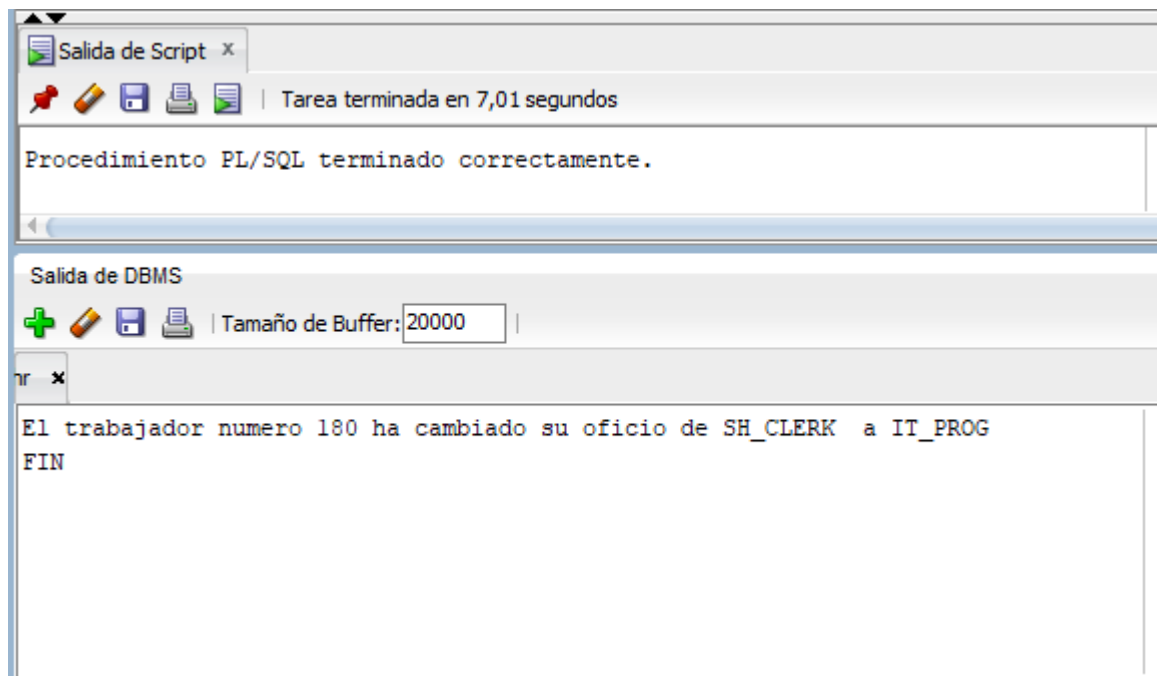
```
Salida de Script x
Tarea terminada en 6,769 segundos

Procedimiento PL/SQL terminado correctamente.

Salida de DBMS
Tamaño de Buffer: 20000

No existe el oficio que quiere introducir
El trabajador 180 continua con el mismo oficio
FIN
```

### 3-Introducción de un numero de trabajador correcto y un oficio correcto



```
Salida de Script x
Tarea terminada en 7,01 segundos

Procedimiento PL/SQL terminado correctamente.

Salida de DBMS
Tamaño de Buffer: 20000

El trabajador numero 180 ha cambiado su oficio de SH_CLERK a IT_PROG
FIN
```

En el .zip del proyecto se adjunta los archivos .sql de función, procedimiento y bloque anónimo para la comprobación en caso de ser necesario.