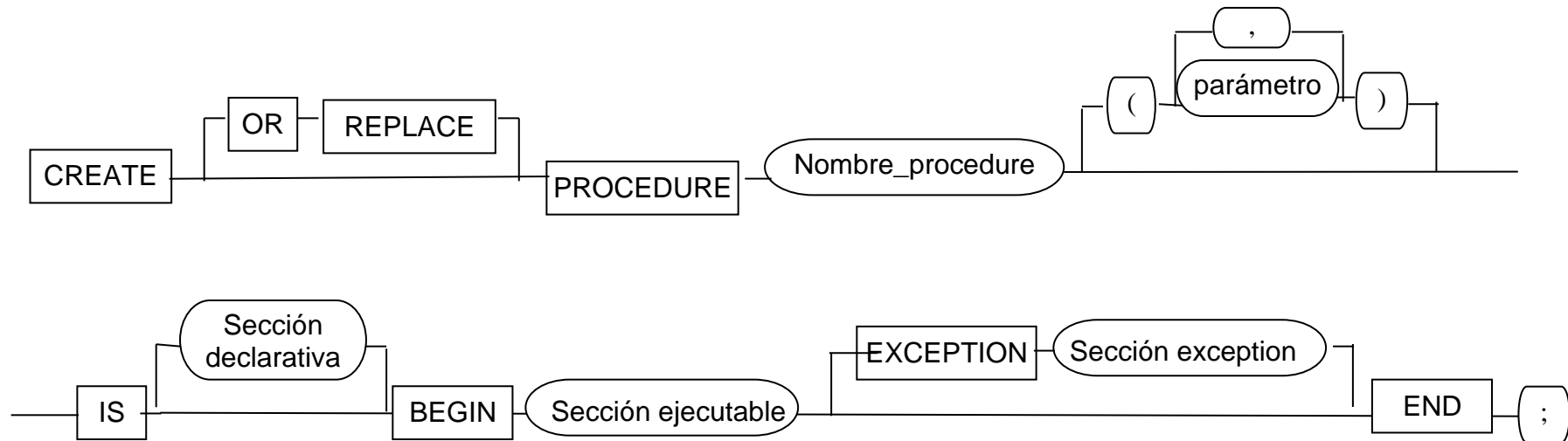


Sesión 9: Procedimientos y funciones

Procedimientos: CREATE PROCEDURE



Al definir los parámetros se debe poner: *nombre_parámetro tipo_de _parámetro tipo_de_datos (Sin longitud)*

Como tipo de parámetro puede ser:

- IN (por defecto) pudiendo pasarle valores en la llamada al procedimiento. Este valor no se puede modificar. Se le puede asignar un valor por defecto.
- OUT para devolver valores.
- IN OUT que permite pasarle valores en la llamada al procedimiento y luego devolver valores.

CREATE PROCEDURE EJEMPLO (entrada in VARCHAR2 default 'PEPE') IS ...

Para borrar un procedimiento **DROP PROCEDURE nombre_procedure**

Ejemplos:

```
create or replace procedure escribir (auxcad in varchar2) is  
begin  
    dbms_output.enable;  
    dbms_output.put_line(auxcad);  
end;
```

Se puede ejecutar directamente

```
BEGIN  
    ESCRIBIR('hola');  
END;
```

O bien desde dentro de otro procedimiento

```
create or replace procedure obtenerprecio(numrecurso in recurso_pago.codigo%TYPE) is  
    auxprecio recurso_pago.precio%TYPE;  
begin  
    select precio into auxprecio  
    from RECURSO_PAGO  
    where codigo = numrecurso;  
    escribir('El precio del recurso '||numrecurso||' es '||auxprecio);  
end;
```

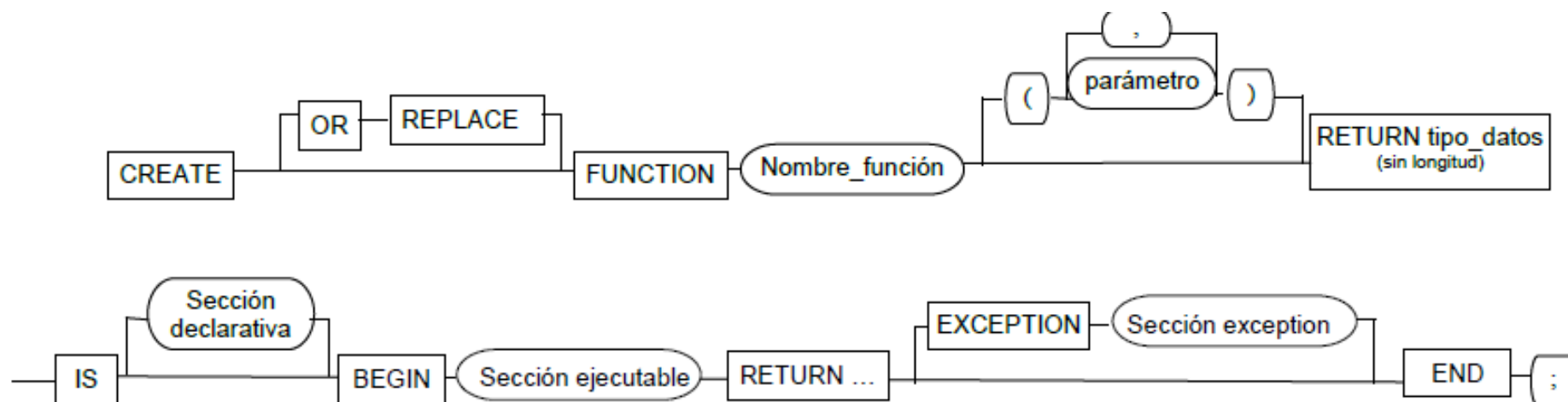
Ejemplo completo: Dado un número de recurso de pago y un porcentaje de incremento, obtener el precio incrementado o un error si el número de recurso de pago no existe en la tabla.

```
create or replace procedure incrementaPrecio (pRecurso in number, pPorcen in number,  
pPrecioFinal OUT recurso_pago.precio%TYPE) is  
  
xprecio recurso_pago.precio%TYPE;  
BEGIN  
    BEGIN  
        SELECT PRECIO INTO XPRECIO FROM RECURSO_PAGO  
        WHERE CODIGO = pRECURSO;  
    EXCEPTION  
        WHEN NO_DATA_FOUND THEN  
            RAISE_APPLICATION_ERROR (-20001,'No existe el recurso');  
    END;  
    pprecioFinal := xprecio + ( xprecio * pPorcen/100);  
    UPDATE RECURSO_PAGO set precio = pprecioFinal WHERE CODIGO = pRECURSO;  
END;
```

Para probar la ejecución de este procedimiento de ejemplo podremos usar este bloque PL/SQL:

```
DECLARE  
    PPRECIOFINAL Recurso_pago.precio%TYPE;  
BEGIN  
    INCREMENTAPRECIO(9,15,PPRECIOFINAL);  
    DBMS_OUTPUT.PUT_LINE('PPRECIOFINAL = ' || PPRECIOFINAL);  
END;
```

Funciones: CREATE FUNCTION



Al definir los parámetros se debe poner: nombre_parámetro tipo_de _parámetro tipo_de_datos (Sin longitud)

Como tipo de parámetro puede ser:

- IN (por defecto) pudiendo pasarle valores en la llamada a la función. Este valor no se puede modificar. Se le puede asignar un valor por defecto.
- Los tipos OUT e IN OUT se suelen utilizar en las funciones de igual forma que en los procedimientos.

CREATE FUNCTION EJEMPLO (entrada in number default 10) RETURN varchar IS ...

.

Para borrar una función **DROP FUNCTION nombre_función**

Ejemplo: Una función que calcula la edad en días de un usuario, del que tenemos su DNI y su fecha de nacimiento en sendas columnas de la tabla USUARIOS:

create or replace function edad (pDni in varchar2) return number is

```
xfechaNac date;  
xedad_endias number;
```

```
BEGIN
```

```
begin
```

```
  Select fnacimiento
```

```
  Into xFechaNac
```

```
  From usuarios where dni = pDni;
```

```
  Xedad_endias := sysdate - xFechaNac;
```

```
Exception
```

```
When others then
```

```
  Xedad_endias:=0;
```

```
End;
```

```
Return Xedad_endias;
```

```
END;
```

Debe haber un RETURN con el valor

Se puede ejecutar

1.- directamente

```
Select edad('11111111A') from dual;
```

2.- En una select (siempre que no se modifiquen datos en la base de datos)

```
select nif, edad(nif) from empleados where edad(dni) > 18;
```

3.- Creación de una vista (siempre que no modifiquen datos en la base de datos). Ésta es una POTENTE herramienta para tener visiones del modelo de datos con columnas calculadas:

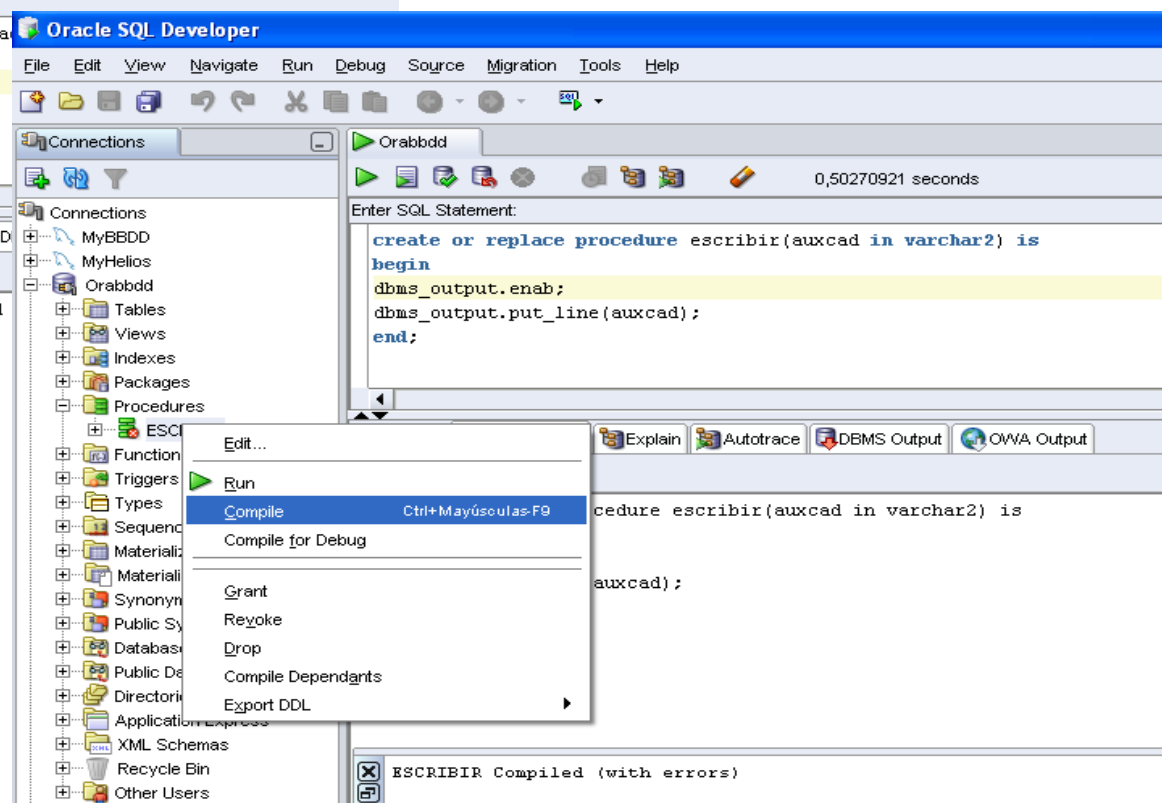
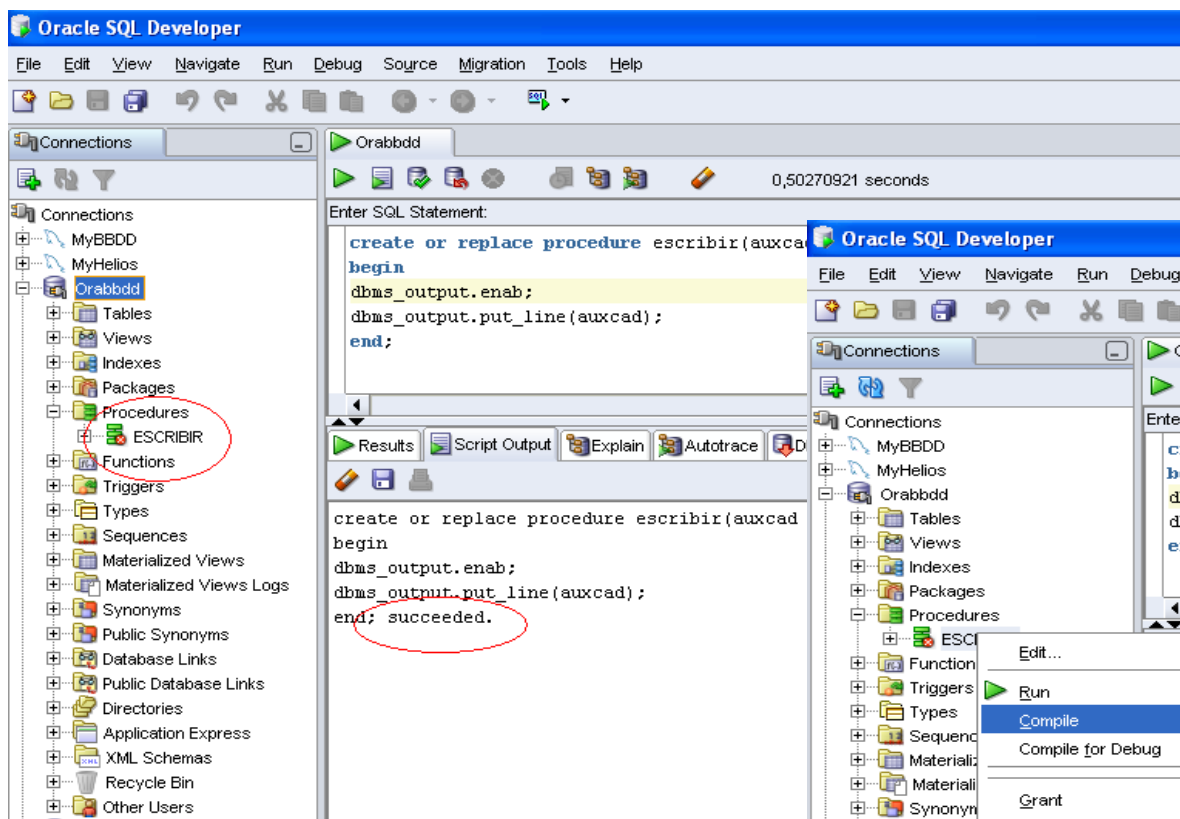
```
create or replace view SaldosClientes  
as select dni, nombre, SaldoContable(dni) as saldo from clientes;
```

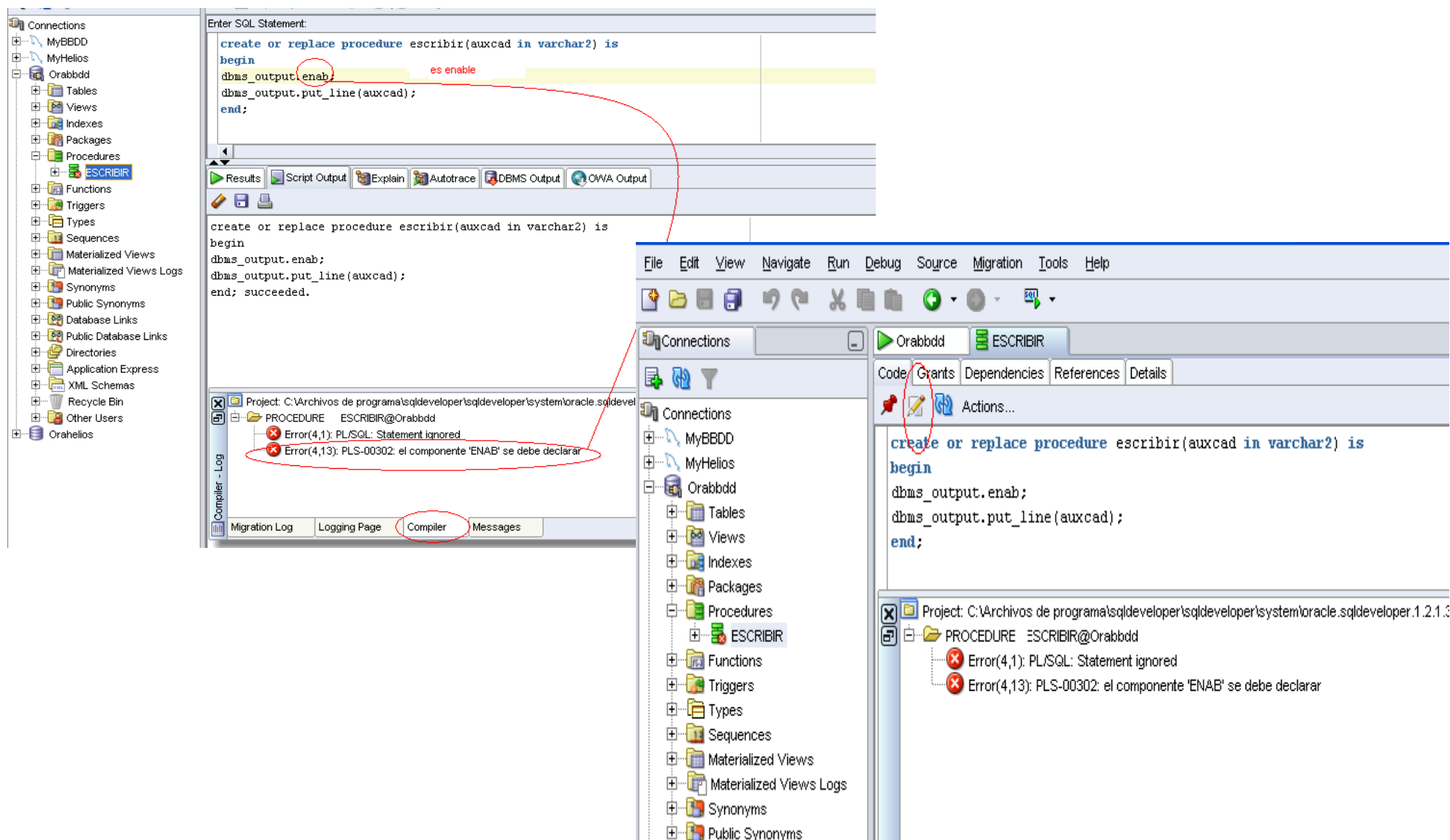
```
select saldo from SaldosClientes where dni='762651872';
```

4.- Dentro de otro procedimiento o función:

```
Procedure Prueba (pDni in Number) is  
xedad number;  
BEGIN  
Select edad(pDni) into xedad from dual;  
If xedad > 18 then  
...  
END;
```

Al crear un procedure, una función, o un trigger (en próximas sesiones) con SQL-Developer, os situáis sobre él y seleccionáis la opción COMPILE. **Si se os indica que se ha creado con errores de compilación debéis corregirlos.**





The screenshot displays the Oracle SQL Developer interface. The main window shows the 'Enter SQL Statement' editor with the following PL/SQL code:

```
create or replace procedure escribir(auxcad in varchar2) is
begin
  dbms_output.enable;
  dbms_output.put_line(auxcad);
end;
```

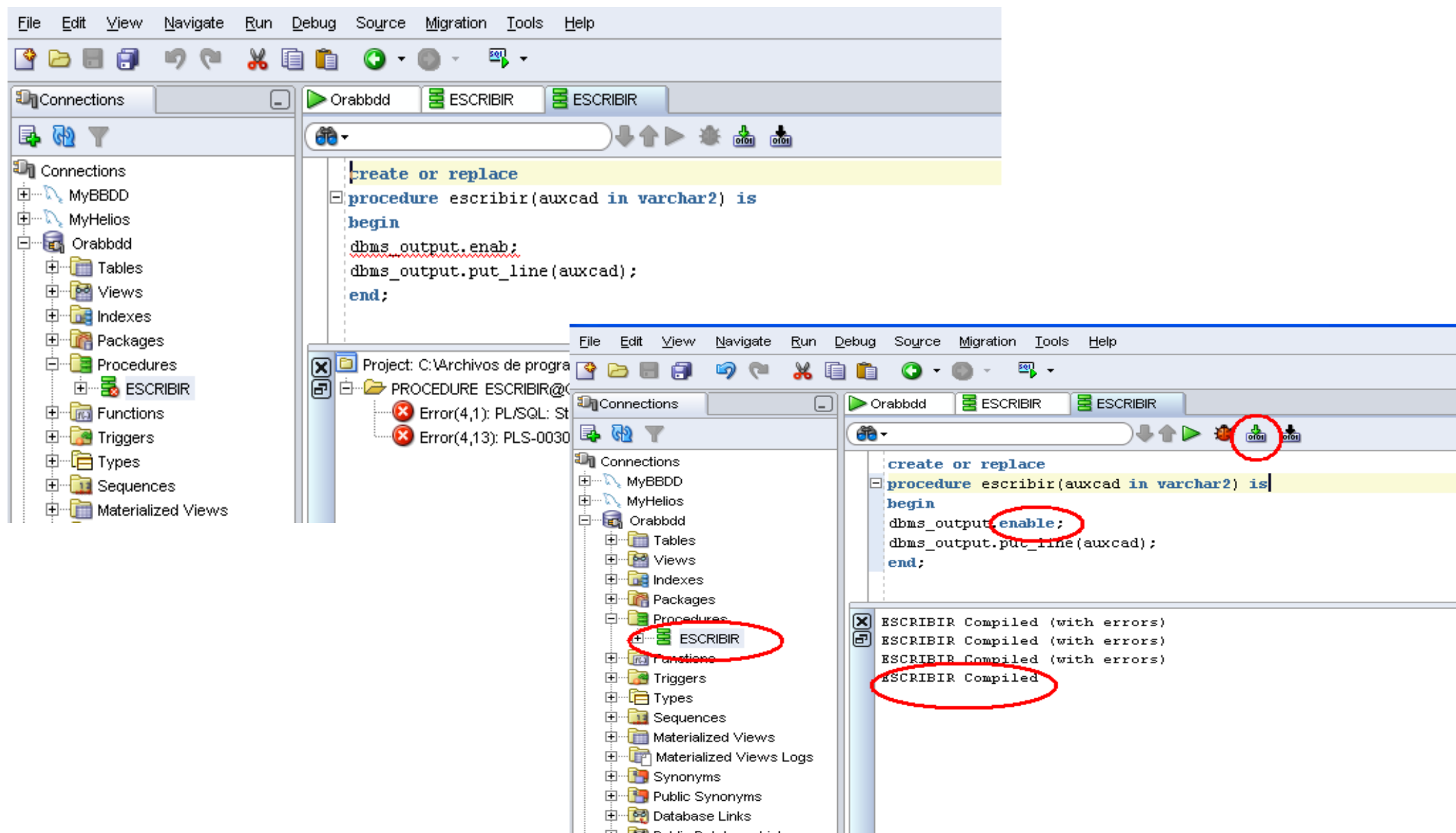
The word 'enable' in the second line is circled in red. Below the editor, the 'Results' tab is selected, showing the execution output: 'end; succeeded.'.

The 'Compiler - Log' window at the bottom left shows the following error messages:

- Error(4,1): PL/SQL: Statement ignored
- Error(4,13): PLS-00302: el componente 'ENAB' se debe declarar

The 'Compiler' tab is circled in red. The 'Connections' window on the right shows the 'Orabdd' connection selected, and the 'ESCRIBIR' procedure is highlighted in the 'Procedures' tree.

The 'Code' tab in the 'ESCRIBIR' window shows the same PL/SQL code as the main editor. The 'Grants' tab is also visible.



The image displays two overlapping screenshots of the Oracle SQL Developer interface. The top screenshot shows a PL/SQL procedure named 'escribir' being edited. The code is as follows:

```
create or replace
procedure escribir(auxcad in varchar2) is
begin
  dbms_output.enab;
  dbms_output.put_line(auxcad);
end;
```

The bottom screenshot shows the same procedure with errors highlighted. The errors are:

- Error(4,1): PL/SQL: St
- Error(4,13): PLS-0030

The procedure is highlighted in the 'Procedures' folder of the 'Orabdd' connection. The compilation status is shown in the bottom right, indicating that the procedure was compiled with errors.

ESCRIBIR Compiled (with errors)
ESCRIBIR Compiled (with errors)
ESCRIBIR Compiled (with errors)
ESCRIBIR Compiled