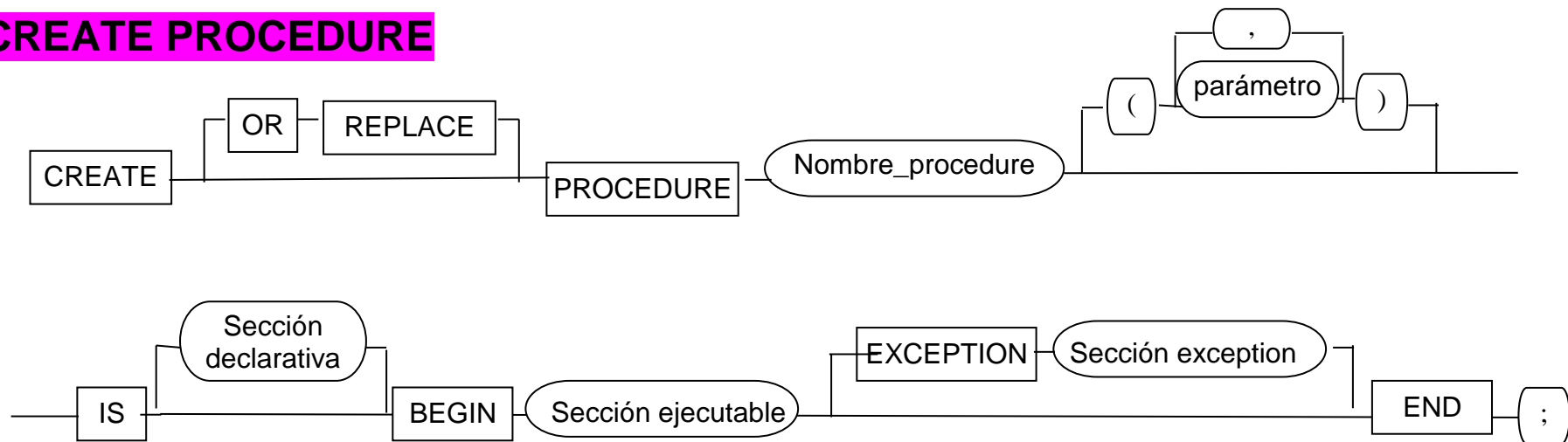


Sesión 9: CREATE PROCEDURE

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 obtenerpreciobase(numhab in number, posiblereserva date) is  
    auxprecio number(3);  
begin  
    select pSA into auxprecio  
    from habitación h , pvptemporada p , calendario c  
    where numero=numhab and fecha=posiblereserva and  
          h.categoria=p.categoria and p.temporada =c.temporada  
    escribir('El precio base para la habitación '||numhab||' en esa fecha es '||auxprecio);  
end;
```

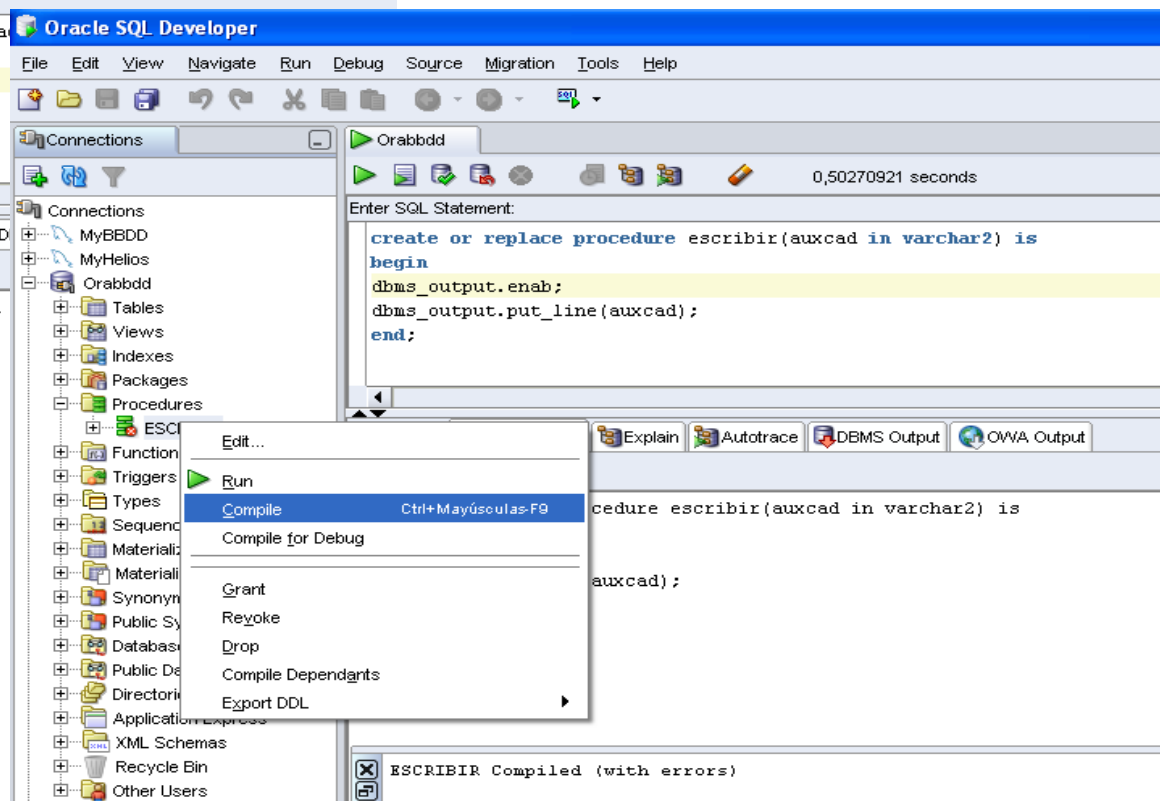
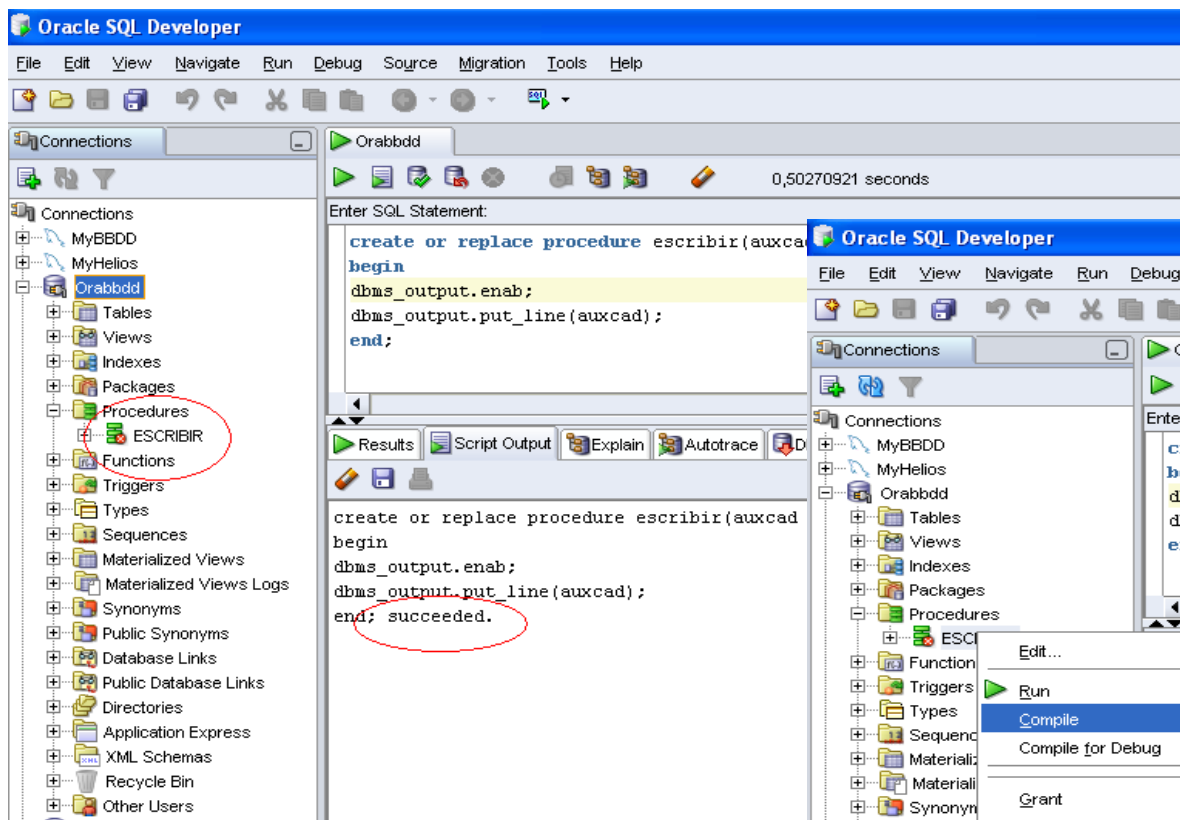
Ejemplo completo:

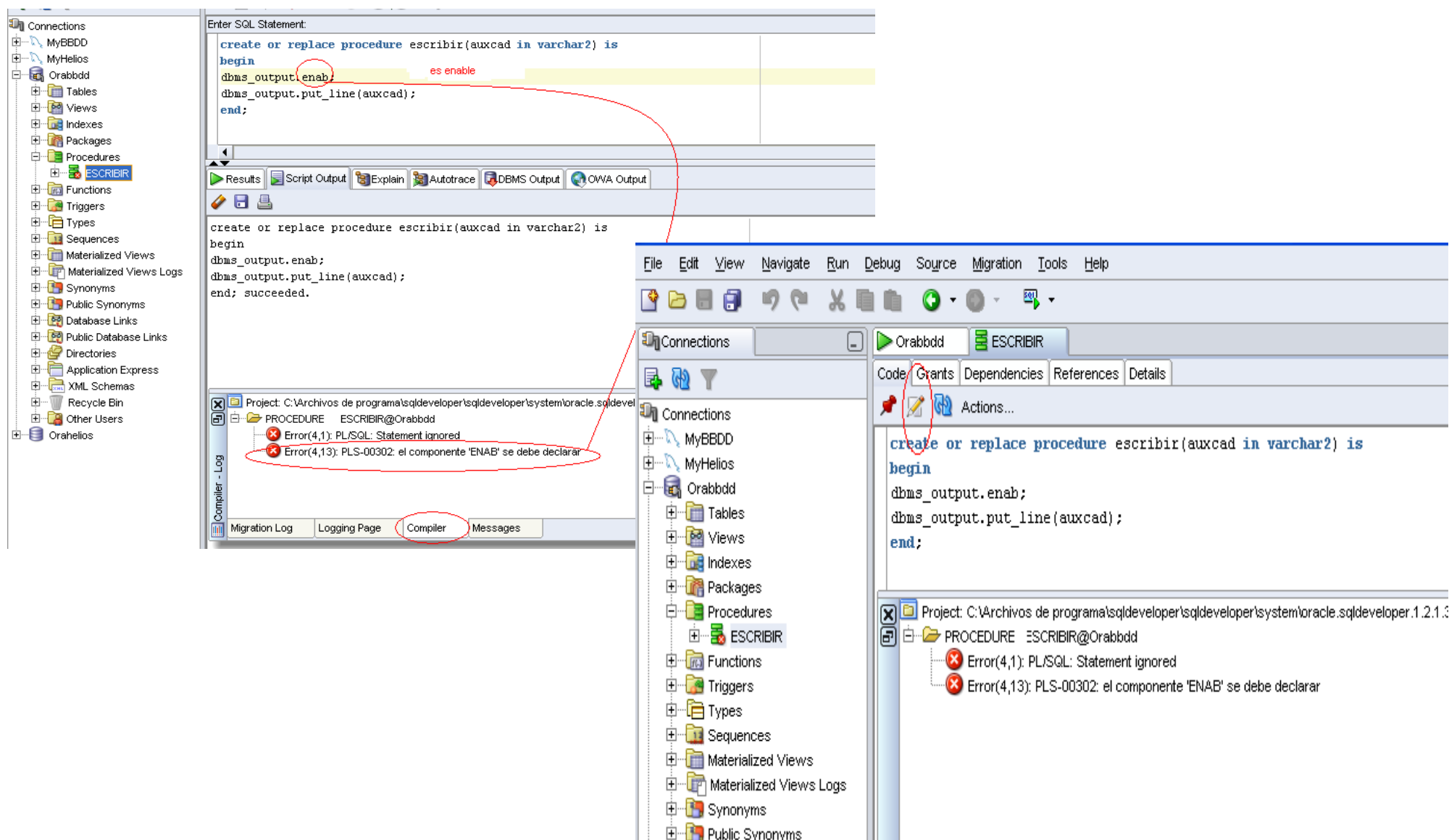
```
create or replace procedure incrementaPrecio (pRecurso in number, pPorcen in number,  
pPrecioFinal OUT number ) is  
  
xprecio number(9,2);  
BEGIN  
    BEGIN  
        SELECT PRECIO INTO XPRECIO FROM RECURSO_PAGO  
        WHERE CODIGO = pRECURSO;  
    EXCEPTION  
        WHEN NO_DATA_FOUND 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 NUMBER;  
BEGIN  
    INCREMENTAPRECIO(9,15,PPRECIOFINAL);  
    DBMS_OUTPUT.PUT_LINE('PPRECIOFINAL = ' || PPRECIOFINAL);  
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.**





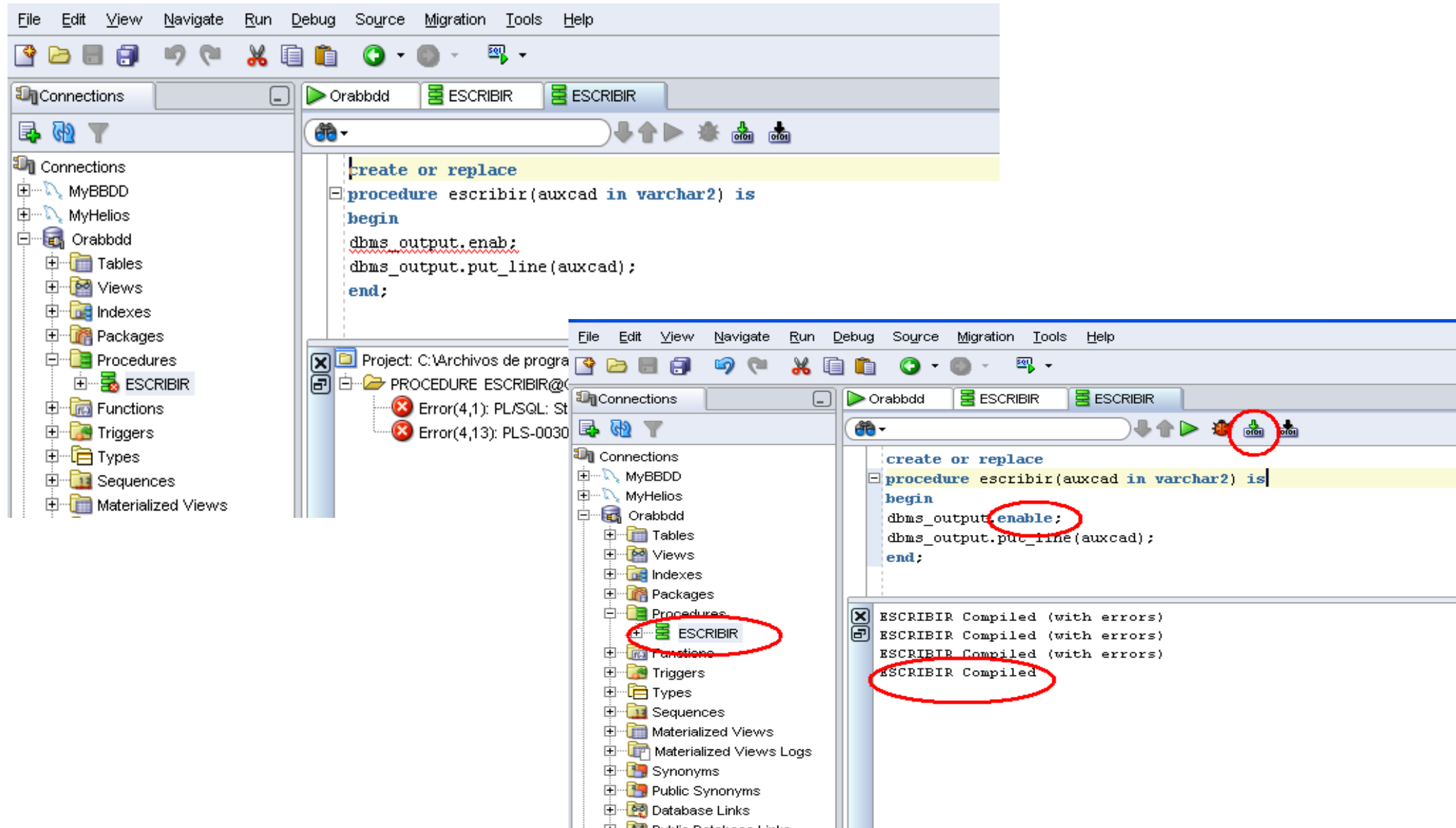
The screenshot displays the Oracle SQL Developer interface with two windows open. The left window shows the 'Enter SQL Statement' editor with the following code:

```
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, with a red arrow pointing to the error message in the right window. The right window shows the 'Code' tab for the procedure 'ESCRIBIR' with the same code. Below the code, the 'Compiler' tab shows two error messages:

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

The 'Compiler' tab is also circled in red. The 'Connections' pane on the left shows the 'Orabdd' connection selected.



The screenshot displays the Oracle SQL Developer interface with two windows open. The main window shows a PL/SQL procedure named 'escribir' with the following code:

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

The code contains two errors: 'dbms_output.enab;' (Error(4,1): PL/SQL: ST) and 'dbms_output.put_line(auxcad);' (Error(4,13): PLS-0030). The 'Procedures' folder in the left pane is circled in red, and the 'ESCRIBIR' procedure is highlighted. The bottom window shows the compilation results:

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

The 'ESCRIBIR Compiled' line is circled in red. The 'dbms_output.enable;' line in the code is also circled in red.