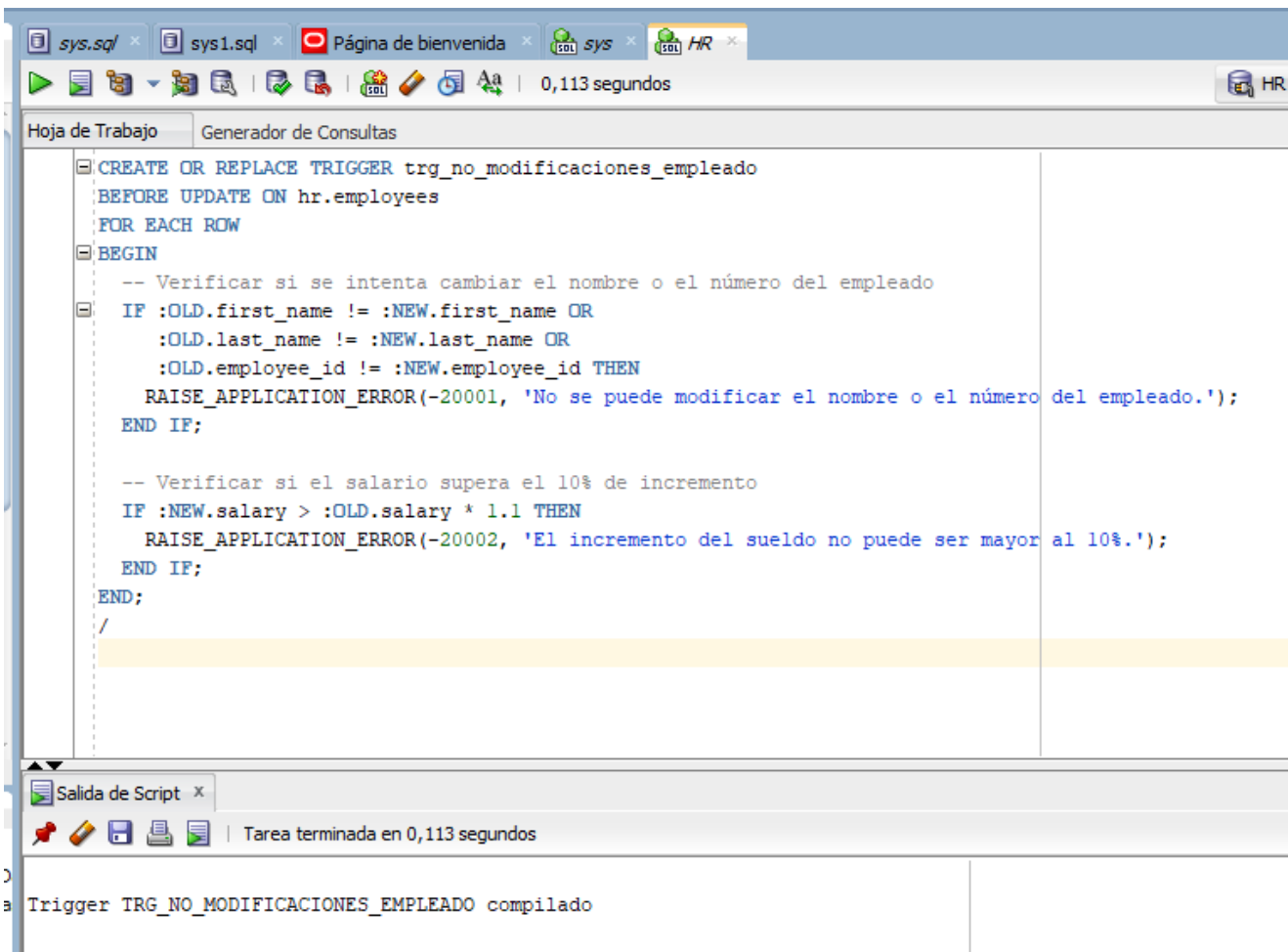


Sobre el esquema HR realizar los siguientes ejercicios:

1. Escribir un disparador que haga fallar cualquier operación de modificación del nombre o del número de un empleado, o que suponga una subida de sueldo superior al 10%.



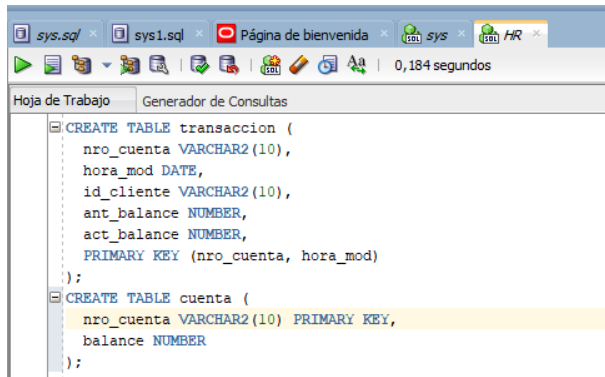
The screenshot shows the Oracle SQL Developer interface. The main window is titled 'Hoja de Trabajo' and 'Generador de Consultas'. It contains a PL/SQL script for creating a trigger. The script is as follows:

```
CREATE OR REPLACE TRIGGER trg_no_modificaciones_empleado
BEFORE UPDATE ON hr.employees
FOR EACH ROW
BEGIN
    -- Verificar si se intenta cambiar el nombre o el número del empleado
    IF :OLD.first_name != :NEW.first_name OR
       :OLD.last_name != :NEW.last_name OR
       :OLD.employee_id != :NEW.employee_id THEN
        RAISE_APPLICATION_ERROR(-20001, 'No se puede modificar el nombre o el número del empleado.');
```

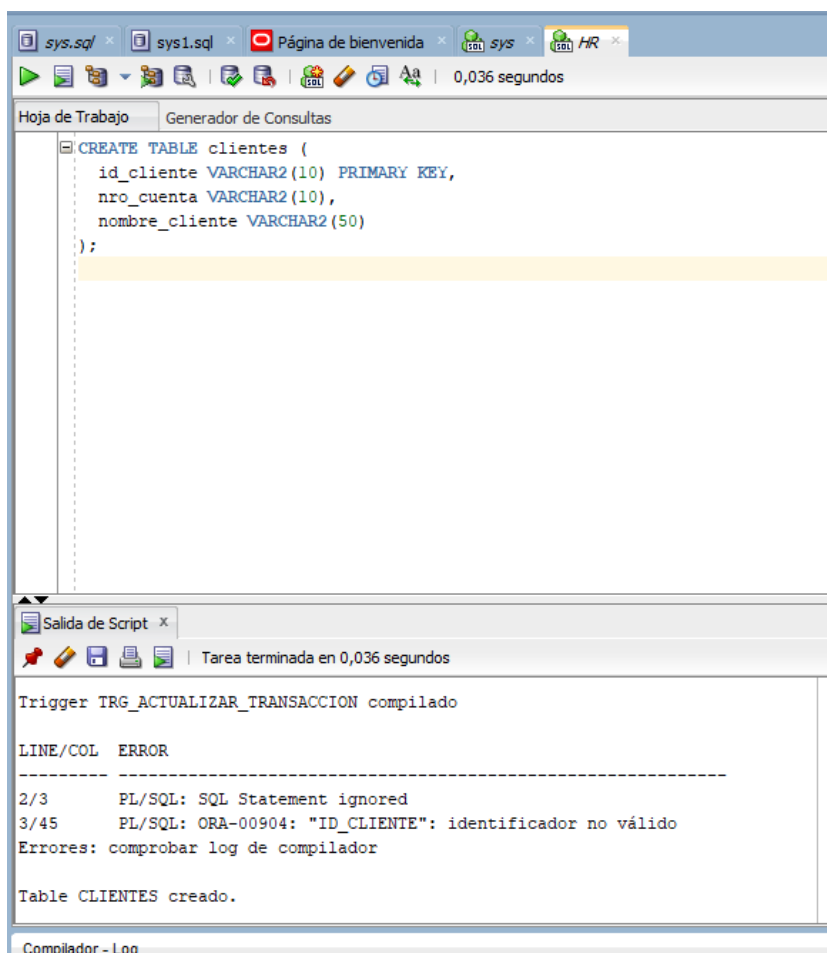
The script is partially visible, with the rest of the code (the second IF statement and the END) obscured by a yellow highlight. Below the script editor, there is a 'Salida de Script' window showing the message: 'Tarea terminada en 0,113 segundos'.

2. Supongamos que tenemos las siguientes tablas con los siguientes atributos:

- **cuenta:** nro_cuenta (10 caracteres) y balance (numérico). El campo nro_cuenta es la clave primaria.
- **transaccion:** nro_cuenta (10 caracteres), hora_mod (tipo fecha), id_cliente (10 caracteres), ant_balance (numérico), act_balance (numérico). Los campos nro_cuenta y hora_mod forman la clave primaria.



```
CREATE TABLE transaccion (  
  nro_cuenta VARCHAR2(10),  
  hora_mod DATE,  
  id_cliente VARCHAR2(10),  
  ant_balance NUMBER,  
  act_balance NUMBER,  
  PRIMARY KEY (nro_cuenta, hora_mod)  
);  
  
CREATE TABLE cuenta (  
  nro_cuenta VARCHAR2(10) PRIMARY KEY,  
  balance NUMBER  
);
```



```
CREATE TABLE clientes (  
  id_cliente VARCHAR2(10) PRIMARY KEY,  
  nro_cuenta VARCHAR2(10),  
  nombre_cliente VARCHAR2(50)  
);
```

Salida de Script x

Tarea terminada en 0,036 segundos

Trigger TRG_ACTUALIZAR_TRANSACCION compilado

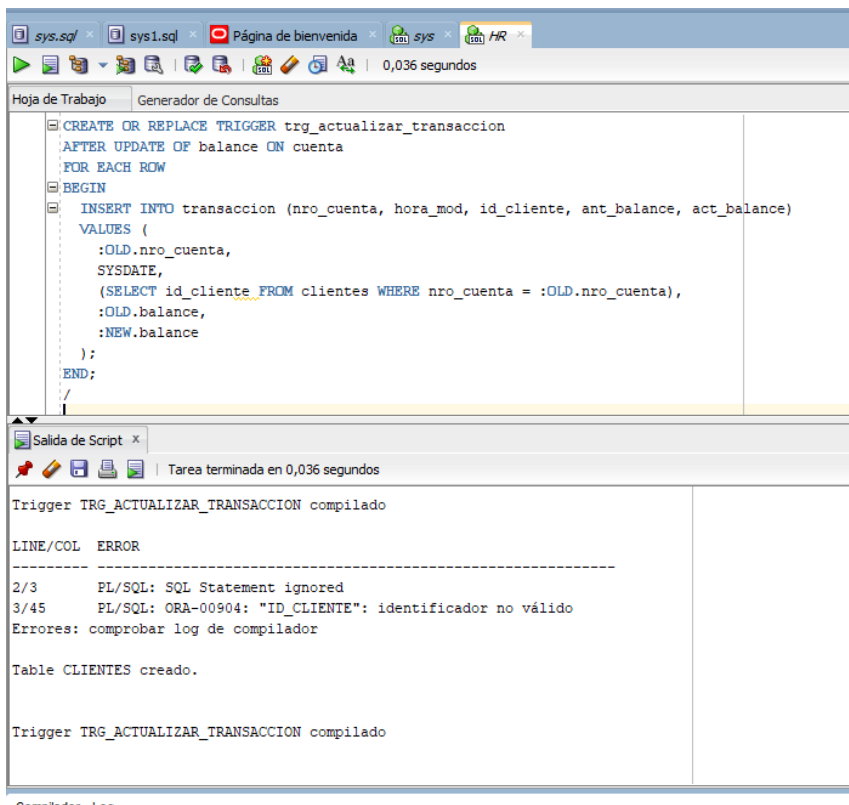
LINE/COL	ERROR
2/3	PL/SQL: SQL Statement ignored
3/45	PL/SQL: ORA-00904: "ID_CLIENTE": identificador no válido

Errores: comprobar log de compilador

Table CLIENTES creado.

Compilador - Log

Crear un disparador que consiga mantener actualizada la tabla *transacción* cada vez que se modifique la tabla *cuenta* (se cambie el atributo *balance*).



The screenshot shows the SQL Developer interface. The top pane displays the SQL script for creating a trigger. The bottom pane shows the output of the script execution, indicating that the trigger was compiled successfully.

```
CREATE OR REPLACE TRIGGER trg_actualizar_transaccion
AFTER UPDATE OF balance ON cuenta
FOR EACH ROW
BEGIN
  INSERT INTO transaccion (nro_cuenta, hora_mod, id_cliente, ant_balance, act_balance)
  VALUES (
    :OLD.nro_cuenta,
    SYSDATE,
    (SELECT id_cliente FROM clientes WHERE nro_cuenta = :OLD.nro_cuenta),
    :OLD.balance,
    :NEW.balance
  );
END;
```

Trigger TRG_ACTUALIZAR_TRANSACCION compilado

LINE/COL ERROR

2/3 PL/SQL: SQL Statement ignored

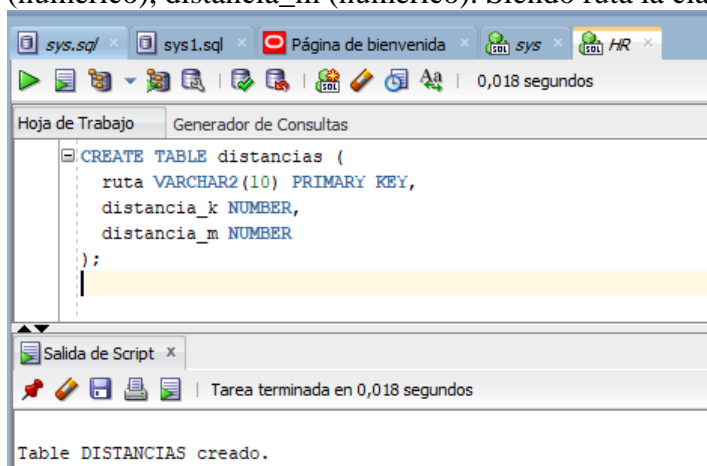
3/45 PL/SQL: ORA-00904: "ID_CLIENTE": identificador no válido

Errores: comprobar log de compilador

Table CLIENTES creado.

Trigger TRG_ACTUALIZAR_TRANSACCION compilado

3. Supongamos que tenemos una tabla de distancias de diferentes rutas y que queremos guardar estas distancias en kilómetros y en millas. *Distancias*: ruta (10 caracteres), distancia_k (numérico), distancia_m (numérico). Siendo ruta la clave principal.

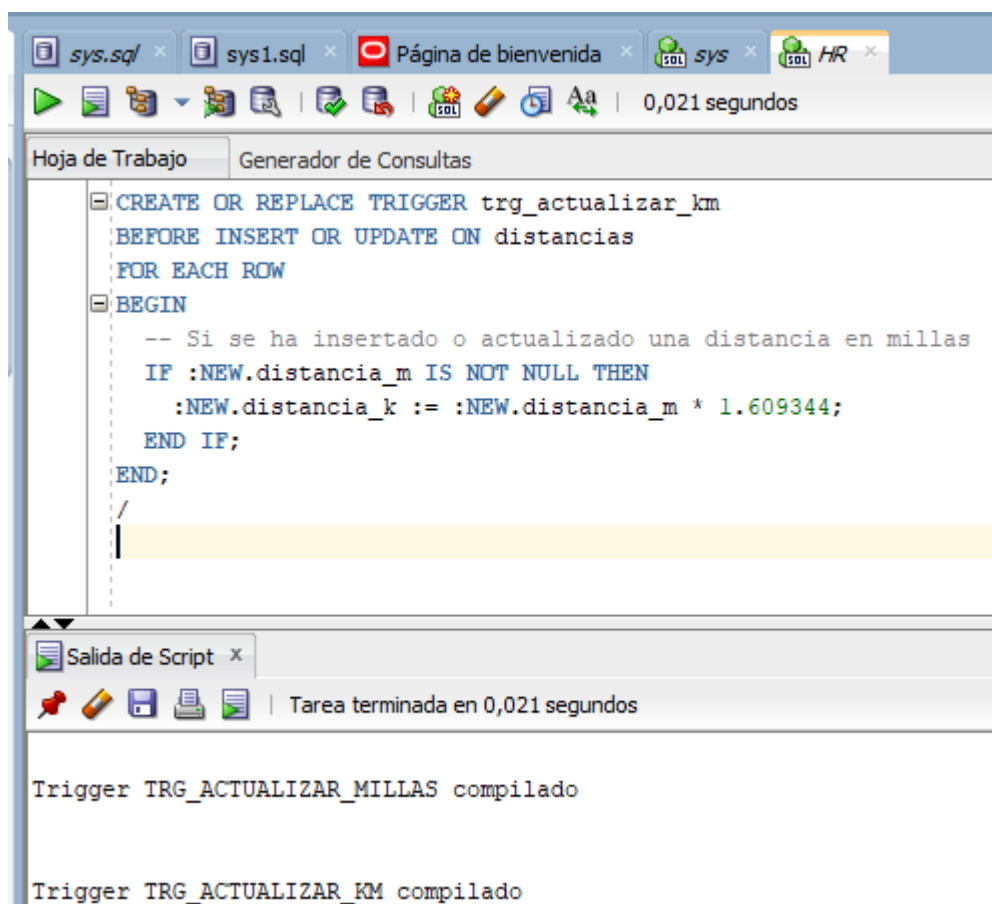
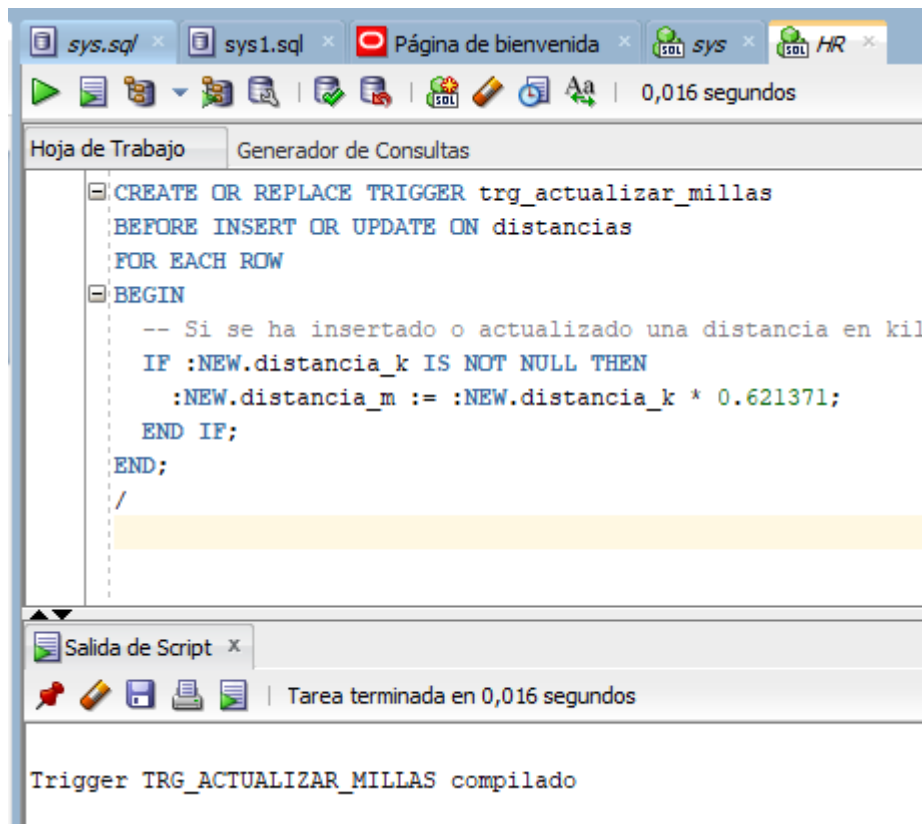


The screenshot shows the SQL Developer interface. The top pane displays the SQL script for creating a table. The bottom pane shows the output of the script execution, indicating that the table was created successfully.

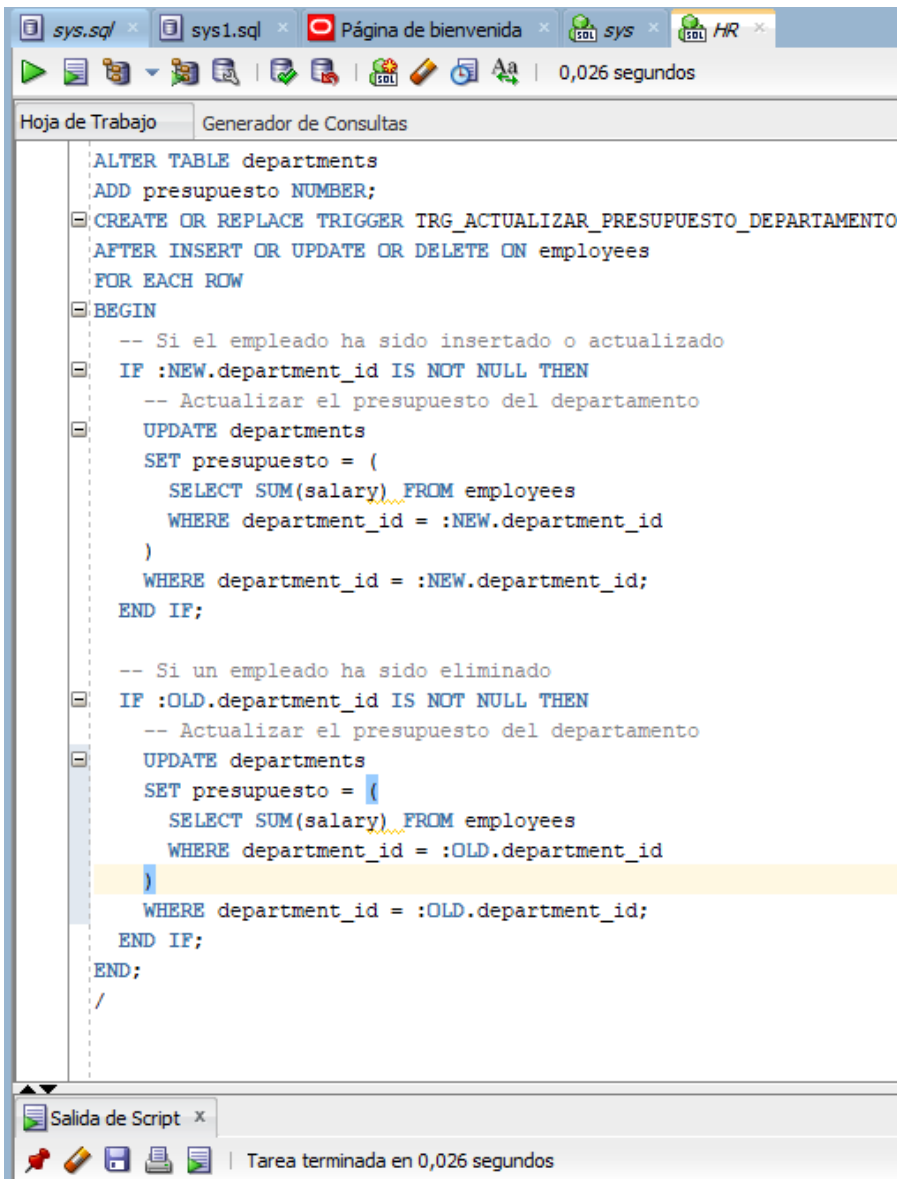
```
CREATE TABLE distancias (
  ruta VARCHAR2(10) PRIMARY KEY,
  distancia_k NUMBER,
  distancia_m NUMBER
);
```

Table DISTANCIAS creado.

Crear disparadores para conseguir que cuando se introduzca (o se modifique) una distancia en kilómetros, automáticamente se introduzca también en millas y viceversa. (1 Km=0.621371 millas y 1 Milla=1.609344 Km)



4. Añadir a la tabla *departments* una nueva columna que almacene el presupuesto total asignado al departamento. Mantener actualizado este nuevo campo mediante disparadores, de tal forma que siempre que se asigne o se quite un empleado del departamento o se modifique su salario quede reflejado en el campo.



The screenshot shows the Oracle SQL Developer interface. The main window displays a PL/SQL script with the following code:

```
ALTER TABLE departments
ADD presupuesto NUMBER;
CREATE OR REPLACE TRIGGER TRG_ACTUALIZAR_PRESUPUESTO_DEPARTAMENTO
AFTER INSERT OR UPDATE OR DELETE ON employees
FOR EACH ROW
BEGIN
    -- Si el empleado ha sido insertado o actualizado
    IF :NEW.department_id IS NOT NULL THEN
        -- Actualizar el presupuesto del departamento
        UPDATE departments
        SET presupuesto = (
            SELECT SUM(salary) FROM employees
            WHERE department_id = :NEW.department_id
        )
        WHERE department_id = :NEW.department_id;
    END IF;

    -- Si un empleado ha sido eliminado
    IF :OLD.department_id IS NOT NULL THEN
        -- Actualizar el presupuesto del departamento
        UPDATE departments
        SET presupuesto = (
            SELECT SUM(salary) FROM employees
            WHERE department_id = :OLD.department_id
        )
        WHERE department_id = :OLD.department_id;
    END IF;
END;
/
```

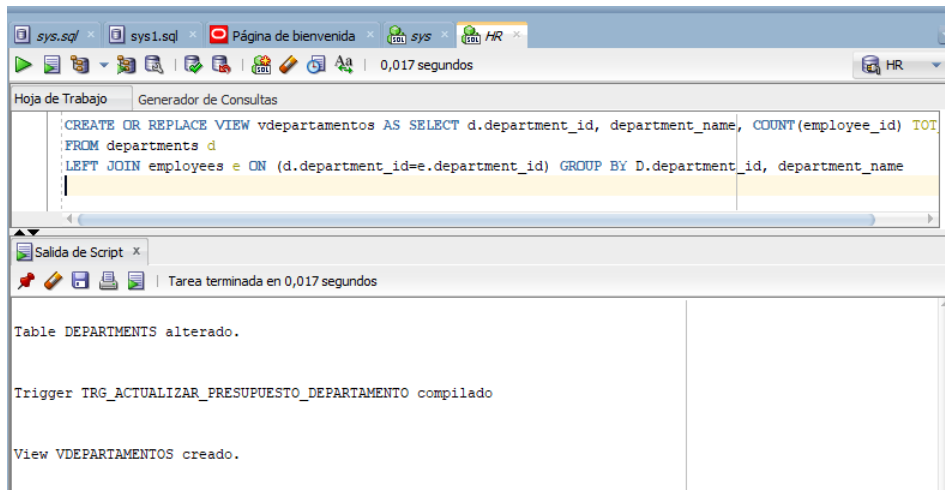
Below the script editor, the 'Salida de Script' (Script Output) window shows the execution results:

```
Table DEPARTMENTS alterado.

Trigger TRG_ACTUALIZAR_PRESUPUESTO_DEPARTAMENTO compilado
```

5. Suponiendo que disponemos de la vista

```
CREATE OR REPLACE VIEW vdepartamentos AS
SELECT d.department_id, department_name,
       COUNT(employee_id) TOT_EMPLE
FROM departments d
LEFT JOIN employees e ON (d.department_id=e.department_id)
GROUP BY D.department_id, department_name
```



Construir un disparador que permita realizar operaciones de actualización en la tabla *departments* a partir de la vista *vdepartamentos*. Se contemplarán las siguientes operaciones:

- Insertar departamento.
- Borrar departamento.

