

Para la realización de los ejercicios de esta unidad nos basaremos en el caso de estudio expuesto en los contenidos de la misma en el Anexo I. La tarea que te pedimos que realices consta de 2 actividades:

Ejercicio 1

Crear un procedimiento que permita cambiar a todos los agentes de una familia determinada (familia origen) a otra familia (familia destino).

El procedimiento tendrá la siguiente cabecera `CambiarAgentesFamilia (id_FamiliaOrigen, id_FamiliaDestino)`, donde cada uno de los argumentos corresponde a un identificador de Familia. Cambiará la columna `Identificador de Familia` de todos los agentes, de la tabla `AGENTES`, que pertenecen a la Familia con código `id_FamiliaOrigen` por el código `id_FamiliaDestino`

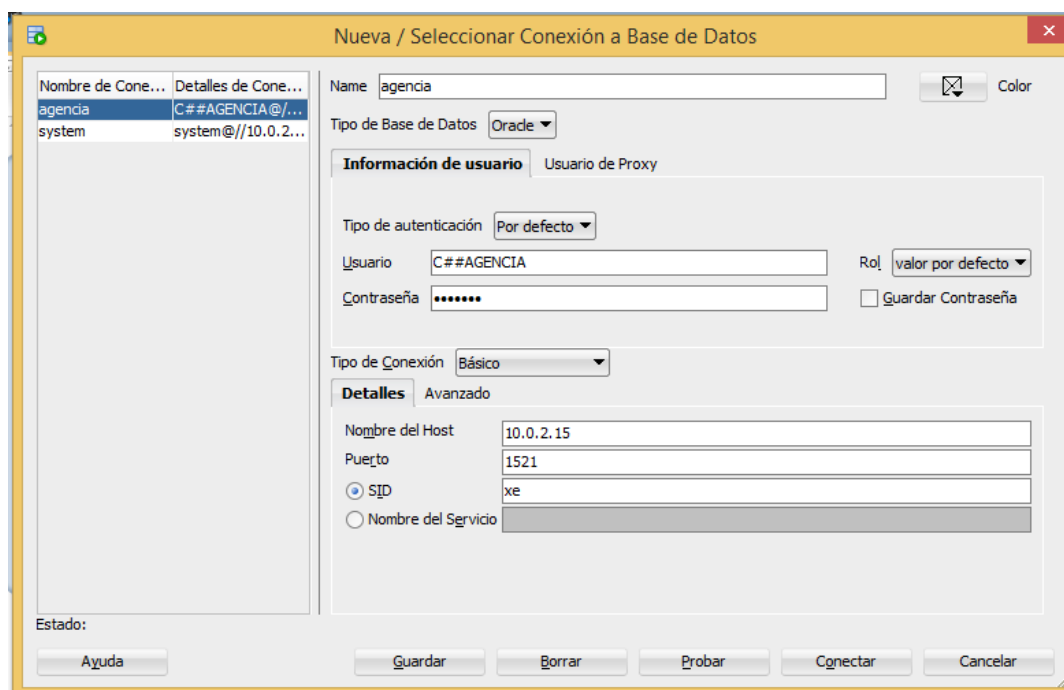
Previamente comprobará que ambas familias existen y que no son iguales.

Para la comprobación de la existencia de las familias se puede utilizar un cursor variable, o contar el número de filas y en caso de que no exista, se visualizará el mensaje correspondiente mediante una excepción del tipo `RAISE_APPLICATION_ERROR`. También se mostrará un mensaje en caso de que ambos argumentos tengan el mismo valor.

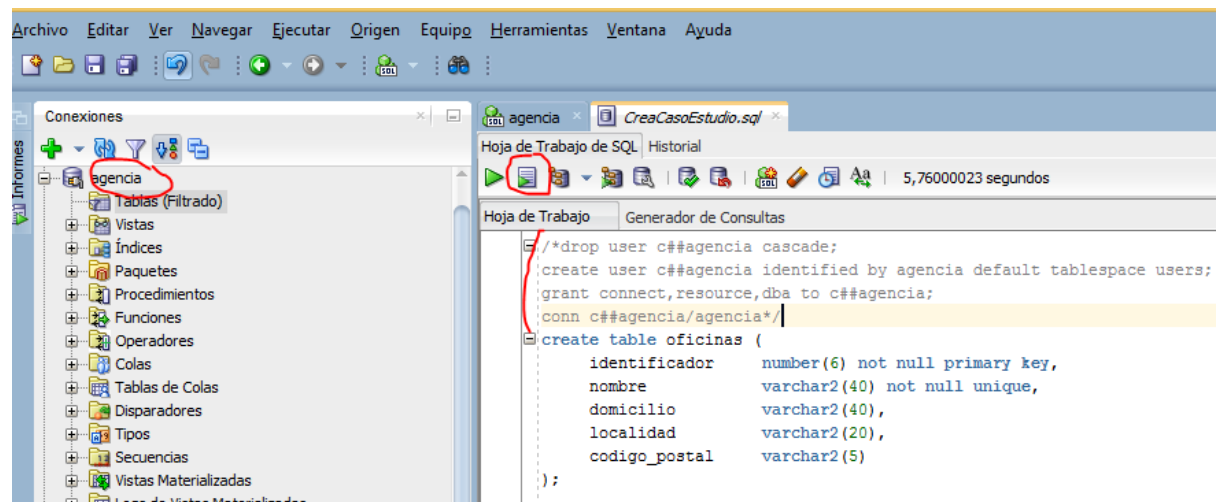
El procedimiento visualizará el mensaje "Se han trasladado XXX agentes de la familia XXXXXX a la familia ZZZZZZ" donde XXX es el número de agentes que se han cambiado de familia, XXXXXX es el nombre de la familia origen y ZZZZZZ es el nombre de la familia destino.

SOLUCION

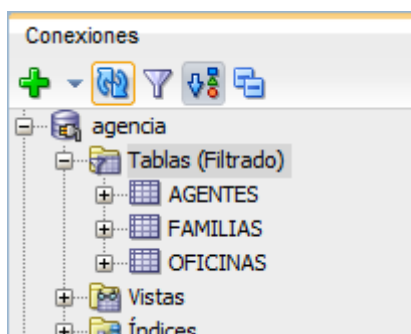
1. Puesto que se ha creado el usuario `C##AGENTE`, yo me crearía una conexión para ese usuario y conectaría con ella.



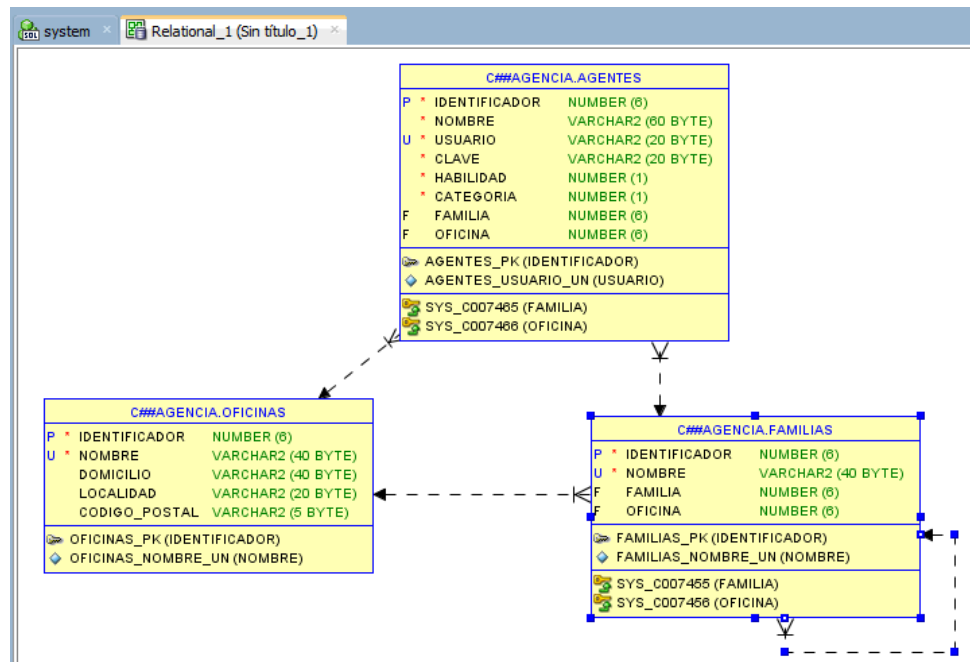
2. Ejecutar el script del caso de estudio, comentando las primeras líneas para que no dé el error de intentar eliminar un usuario que está conectado.



3. El resultado, las tablas creadas.



4. Aunque no es parte de la solución yo empezaría por generar el diagrama entidad-relacional para comprobar que relaciones existen entre las tablas, que ya debéis saber cómo se genera, el resultado es:



Y efectivamente las relaciones están creadas correctamente. Como respuesta a una duda que plantea un compañero sobre cómo y cuando se crean las claves foráneas (Ver UD3. [Restricciones](#)).

- Sobre cuando se crean, pues depende, si se conocen en el momento de crear las tablas, se pueden crear en ese momento y si las tablas ya están creadas se pueden crear a posteriori utilizando la sentencia ALTER TABLE.
- Sobre cómo se crean, hay varias formas:
 - Haciendo referencia a la tabla y los campos de donde procede.

```

CREATE TABLE USUARIOS (
  Cod_Partida NUMBER(8)
  CONSTRAINT Cod_Part_FK
  REFERENCES PARTIDAS(Cod_Partida));
  
```

- O si el campo al que hace referencia es clave principal en su tabla no es necesario indicar el nombre del campo:

```

CREATE TABLE USUARIOS (
  Cod_Partida NUMBER(8)
  CONSTRAINT Cod_Part_FK
  REFERENCES PARTIDAS);
  
```

Esta forma es la que se utiliza en el ejercicio que nos ocupa:

```

create table oficinas (
  identificador number(6) not null primary key,
  nombre        varchar2(40) not null unique,
  domicilio     varchar2(40),
  localidad     varchar2(20),
  codigo_postal varchar2(5)
);

create table familias (
  identificador number(6) not null primary key,
  nombre        varchar2(40) not null unique,
  familia       number(6) references familias,
  oficina       number(6) references oficinas
);

create table agentes (
  identificador number(6) not null primary key,
  nombre        varchar2(60) not null,
  usuario       varchar2(20) not null unique,
  clave        varchar2(20) not null,
  habilidad     number(1) not null,
  categoria     number(1) not null,
  familia       number(6) references familias,
  oficina       number(6) references oficinas
);

```

5. Comprobar la Integridad Referencial. Observando la definición de las claves foráneas en las tablas y en ausencia de otros valores, el valor por defecto es NO ACTION. También se puede comprobar gráficamente como se muestra en la imagen.

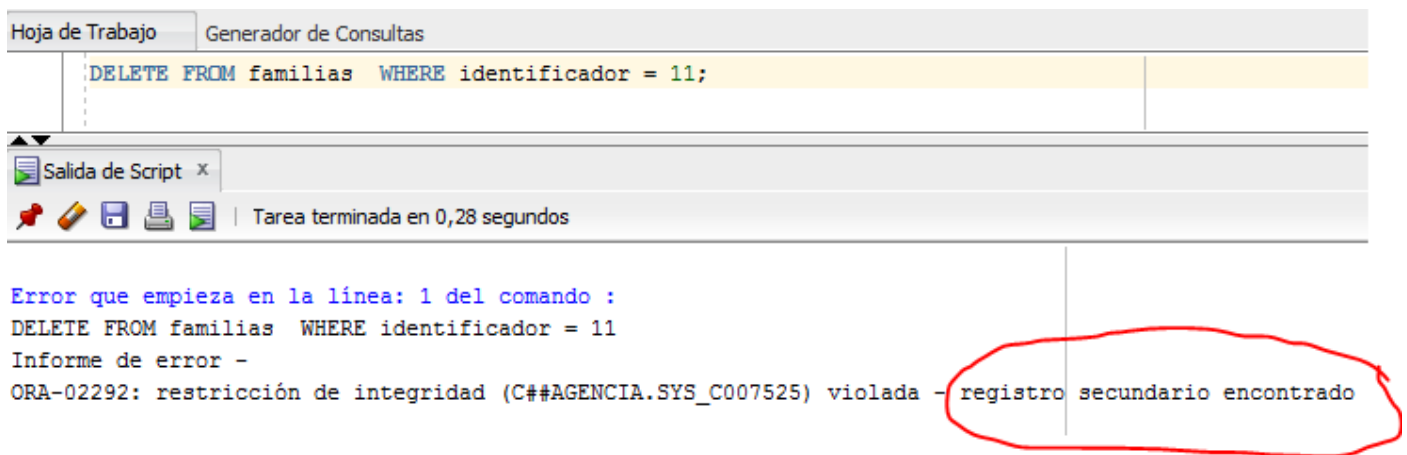
CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STA
1 SYS_C007511	Check	"IDENTIFICADOR" IS NOT NULL	(null)	(null)	(null)	(null)	ENABI
2 SYS_C007512	Check	"NOMBRE" IS NOT NULL	(null)	(null)	(null)	(null)	ENABI
3 SYS_C007513	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABI
4 SYS_C007514	Unique	(null)	(null)	(null)	(null)	(null)	ENABI
5 SYS_C007515	Foreign_Key	(null)	C##AGENCIA	FAMILIAS	SYS_C007513	NO ACTION	ENABI
6 SYS_C007516	Foreign_Key	(null)	C##AGENCIA	OFICINAS	SYS_C007509	NO ACTION	ENABI

Probemos cómo funciona la Integridad Referencial:

Veamos los datos de la tabla 'FAMILIA' y no fijamos en el registro con identificador 11, dicho registro está relacionado con los registros 2 y 3 en la tabla FAMILIAS y no está relacionado con ningún registro de la tabla AGENTES.

IDENTIFICADOR	NOMBRE	FAMILIA	OFICINA
1	11 Madrid-1	(null)	1
2	111 Madrid-1.1	11	(null)
3	112 Madrid-1.2	11	(null)
4	1121 Madrid-1.2.1	112	(null)
5	1122 Madrid-1.2.2	112	(null)
6	1123 Madrid-1.2.3	112	(null)
7	21 Granada-1	(null)	2
8	211 Granada-1.1	21	(null)
9	212 Granada-1.2	21	(null)
10	213 Granada-1.3	21	(null)
11	31 Jaén-1	(null)	3

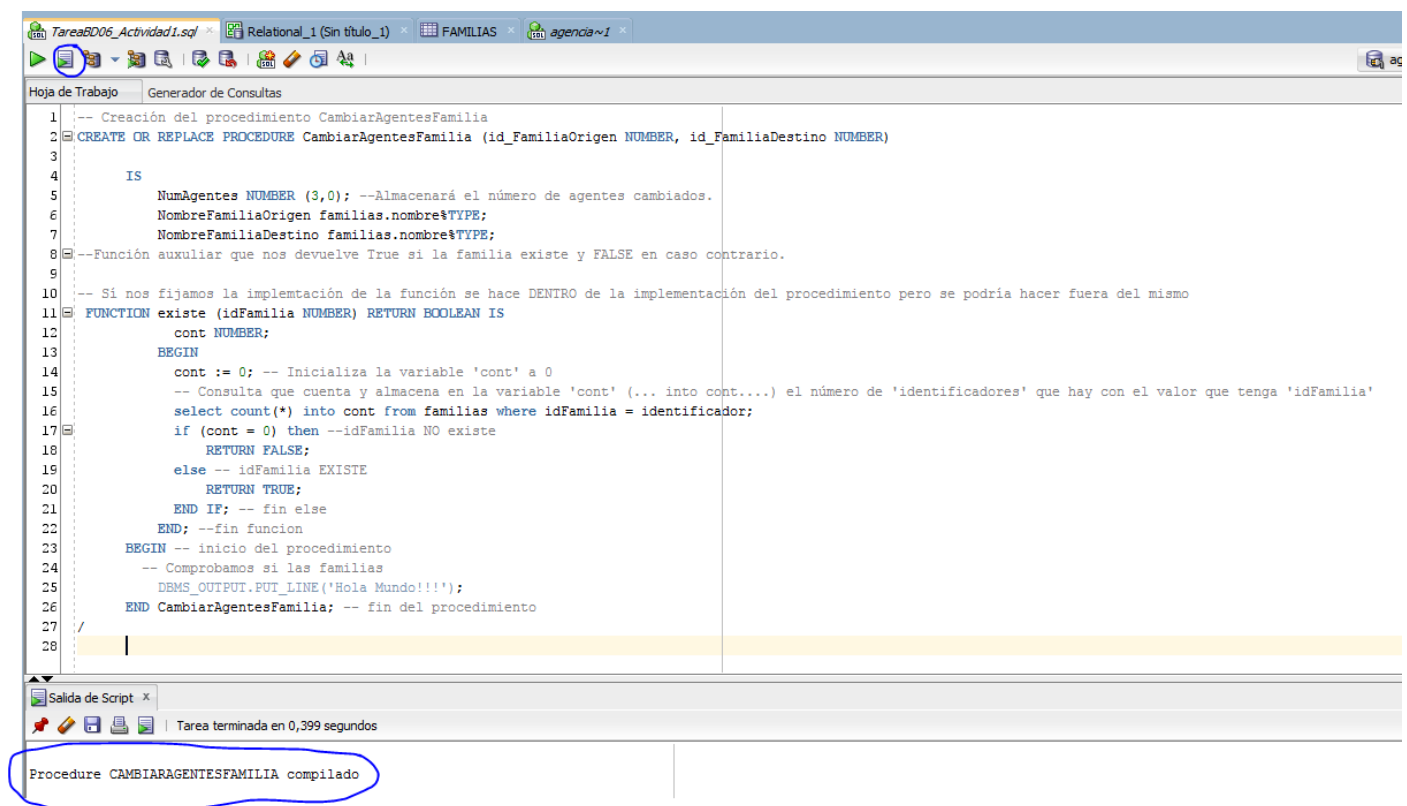
Si lo intentamos eliminar, este es el resultado



Y creo que no me equivoco si dijo que todos los registros de la tabla FAMILIAS están relacionados con alguno o varios registros de la tabla FAMILIAS y/o AGENTES.

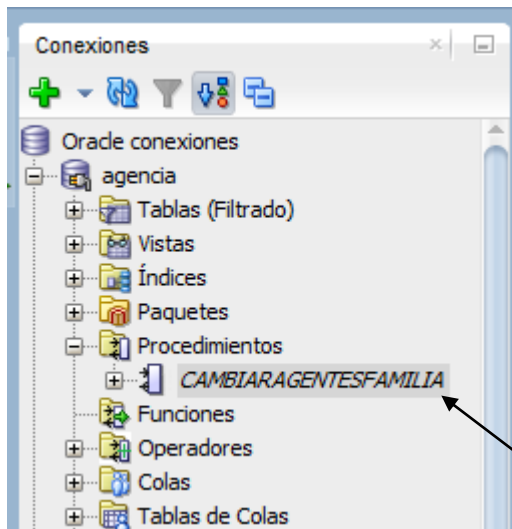
Hasta aquí el punto de partido, seguimos con la **Actividad 1**. Escribiré y compilaré el código por partes para gestionar mejor los posibles errores.

✓ Escribimos la primera parte código:

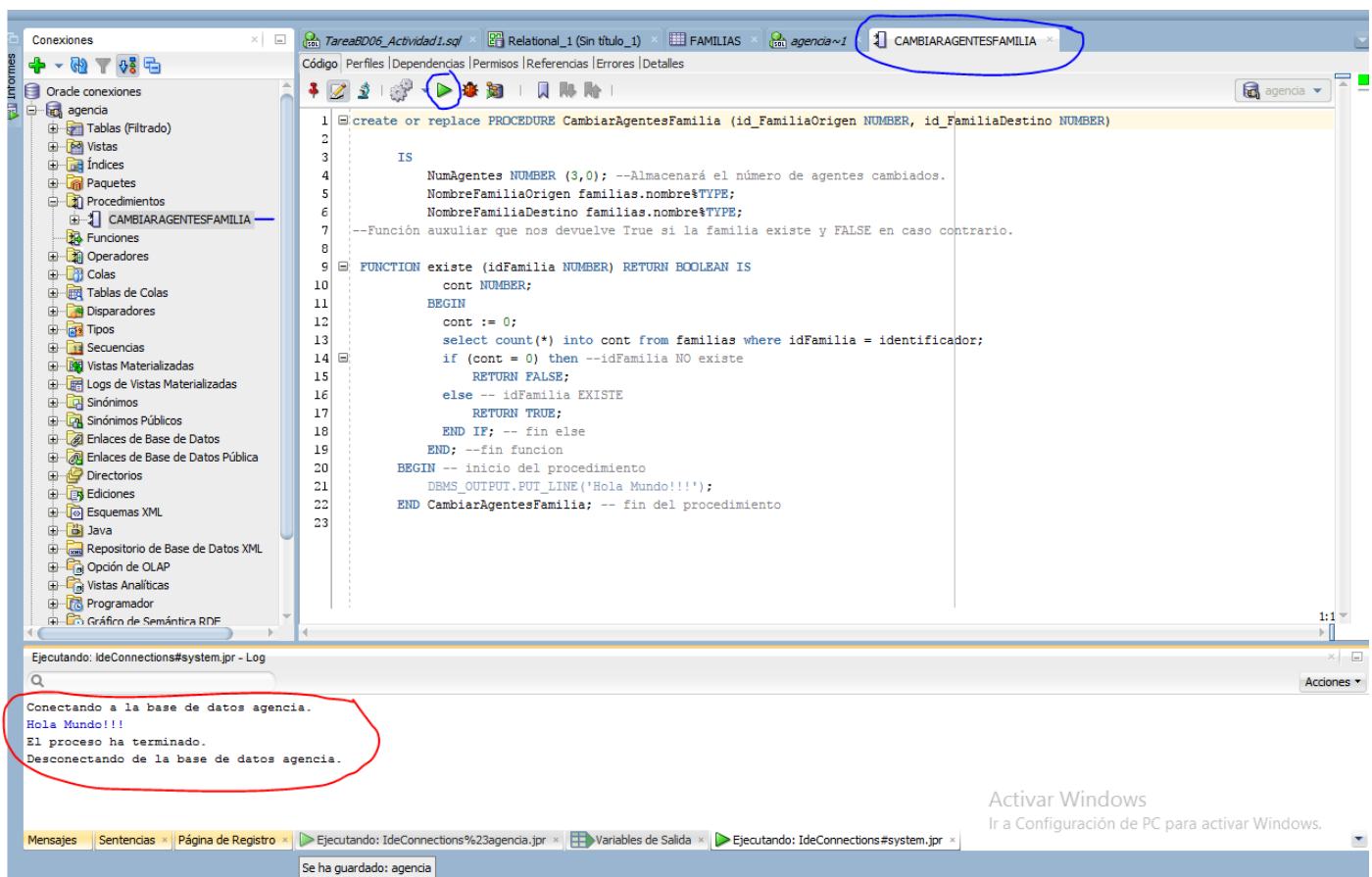


Temporalmente en el cuerpo de procedimiento solo tengo una instrucción que muestra por pantalla del mensaje 'Hola Mundo!!!' (línea 25) después meteré el resto de código.

- ✓ De momento todo va bien, al no tener fallos sintácticos y por tanto compilar correctamente, el resultado es que se ha creado el procedimiento, lo podemos ver en el árbol de la izquierda:



- ✓ También podemos ejecutarlo, solo hacer click sobre el procedimiento se abrirá otra pestaña y pulsar en Ejecutar. Veamos el resultado:

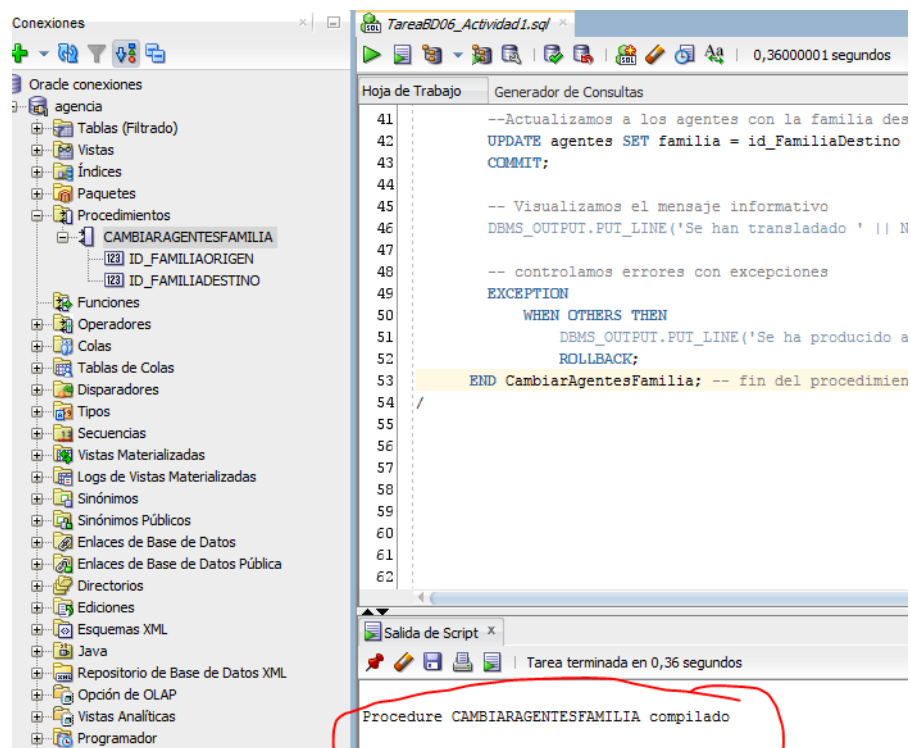


- ✓ Seguimos escribiendo el código:

```

23 BEGIN -- inicio del procedimiento
24 IF (id_FamiliaOrigen = id_FamiliaDestino) THEN
25     RAISE_APPLICATION_ERROR(-20011, 'Las familias origen y destino no pueden ser iguales');
26 END IF;
27 -- Comprobamos si existe la familia origen LLAMANDO a la funcion EXISTE
28 IF NOT existe(id_FamiliaOrigen) THEN
29     RAISE_APPLICATION_ERROR(-20012, 'La familia ' || id_FamiliaOrigen || 'no existe' );
30 END IF;
31 -- Comprobamos si existe la familia destino LLAMANDO a la funcion EXISTE
32 IF NOT existe(id_FamiliaDestino) THEN
33     RAISE_APPLICATION_ERROR(-20013, 'La familia ' || id_FamiliaDestino || 'no existe' );
34 END IF;
35
36 -- Obtenemos los datos para la visualización del mensaje por pantalla
37 SELECT COUNT(*) INTO NumAgentes FROM agentes WHERE familia=id_FamiliaOrigen;
38 SELECT nombre INTO NombreFamiliaOrigen FROM familias WHERE familia=id_FamiliaOrigen;
39 SELECT nombre INTO NombreFamiliaDestino FROM familias WHERE familia=id_FamiliaDestino;
40
41 --Actualizamos a los agentes con la familia destino
42 UPDATE agentes SET familia = id_FamiliaDestino WHERE familia = id_FamiliaDestino;
43 COMMIT;
44
45 -- Visualizamos el mensaje informativo
46 DBMS_OUTPUT.PUT_LINE('Se han trasladado ' || NumAgentes || ' agentes de familia ' || NombreFamiliaOrigen || ' a la familia ' || NombreFamiliaDestino );
47
48 -- controlamos errores con excepciones
49 EXCEPTION
50     WHEN OTHERS THEN
51         DBMS_OUTPUT.PUT_LINE('Se ha producido al erro' || SQLCODE || ' correspondiente a ' || SUBSTR(SQLERRM,1,200));
52         ROLLBACK;
53 END CambiarAgentesFamilia; -- fin del procedimiento
54
55 /
  
```

- ✓ Compilamos y Ejecutamos como antes. Todo correcto.

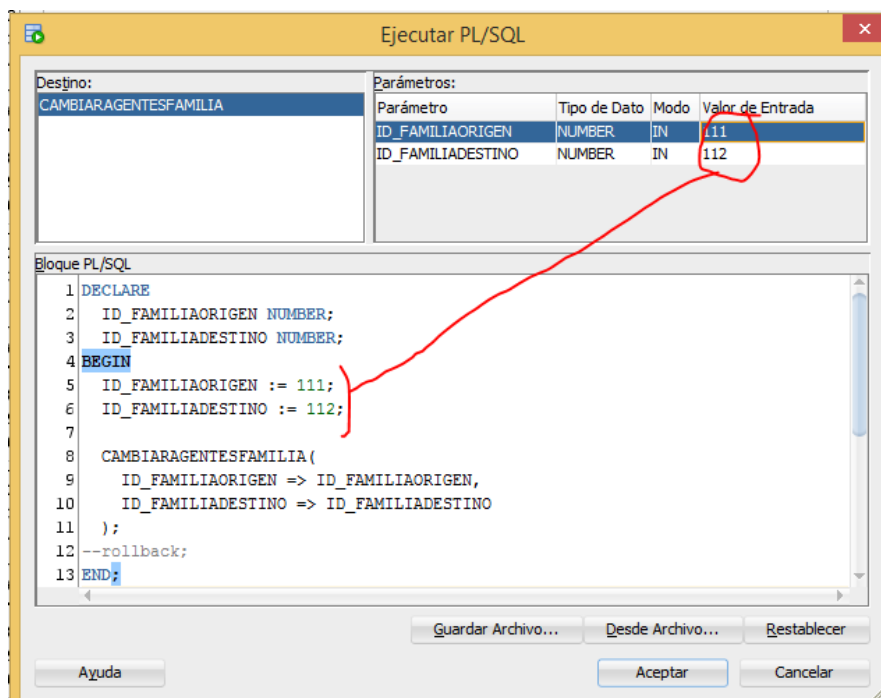


✓ Para terminar solo queda probarlo. Para probarlo tenemos dos opciones:

- Con SQL:

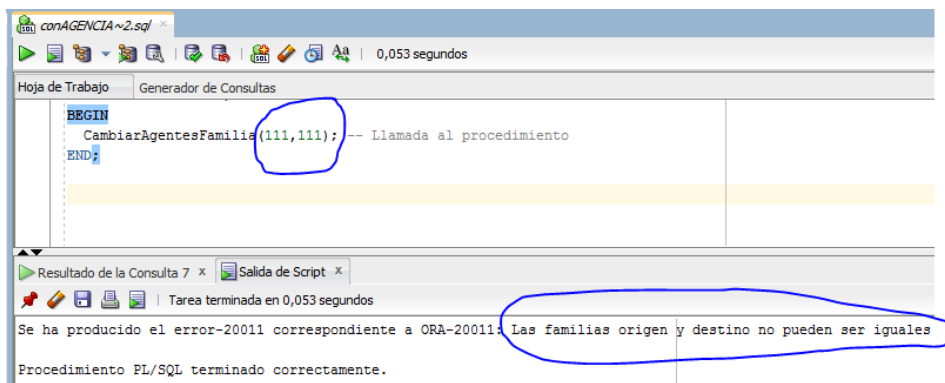
```
SET SERVEROUTPUT ON;
BEGIN
  CambiarAgentesFamilia(111,112); -- Llamada al procedimiento
END;
```

- Gráficamente:

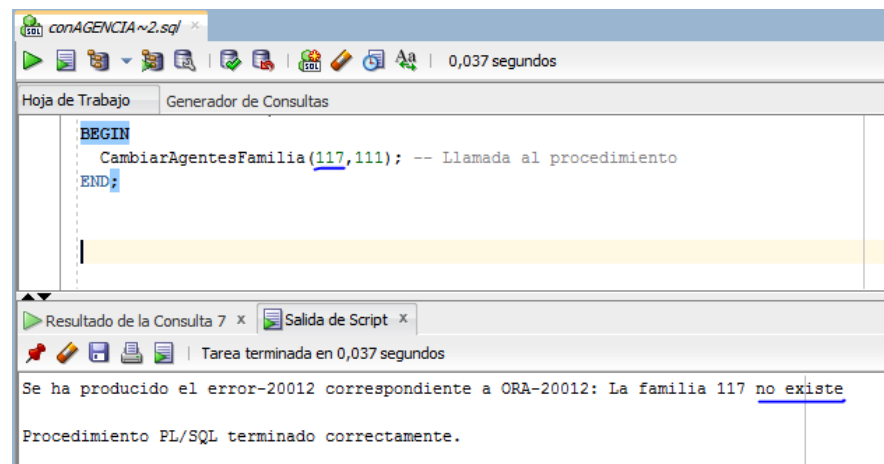


Usar la que más os guste.....yo lo haré con SQL

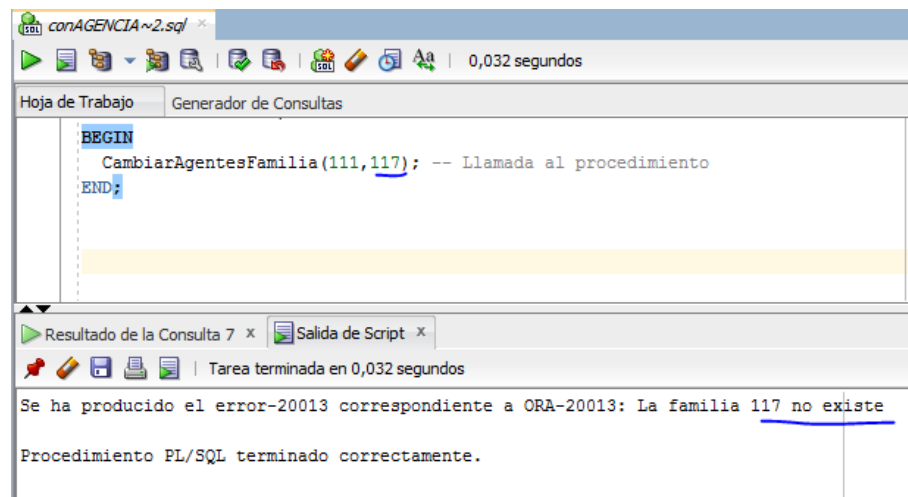
- Comenzaremos probando si tenemos bien controlados los errores:
 - Si las familias origen y destino SON IGUALES.



- Si la familia origen NO EXISTE.



- Si la familia destino NO EXISTE.



✓ Comprobado que los errores están controlados, pasaremos a probar con datos correctos.

- Id_FamiliaOrigen = 111
- Id_FamiliaDestino = 112

Ambos identificadores existen y no son iguales, luego el programa se ejecutaría de forma correcta, es decir, realizará la modificación y mostrará el mensaje correspondiente.

Antes de ejecutarlo vamos a ver que registros se verán afectados y como quedarán los datos una vez realizada la modificación. Los agentes de la familia 111 pasarán a pertenecer a la familia 112. Para conocer cuántas y qué filas se verán afectadas en la actualización haremos la siguiente consulta:

Hoja de Trabajo Generador de Consultas	
<pre>SELECT nombre, familia FROM agentes WHERE familia = 111;</pre>	
Salida de Script x	
Tarea terminada en 0,046 segundos	
NOMBRE	FAMILIA
Salvador Romero Villegas	111
José Javier Bermúdez Hernández	111
Alfonso Bonillo Sierra	111

- ✓ Ejecutamos el programa escribiendo el valor de los parámetros de entrada como se indica en la imagen:

Ejecutar PL/SQL

Destino: CAMBIARAGENTESFAMILIA

Parámetro	Tipo de Dato	Modo	Valor de Entrada
ID_FAMILIAORIGEN	NUMBER	IN	111
ID_FAMILIADESTINO	NUMBER	IN	112

Bloque PL/SQL

```

DECLARE
  ID_FAMILIAORIGEN NUMBER;
  ID_FAMILIADESTINO NUMBER;
BEGIN
  ID_FAMILIAORIGEN := 111;
  ID_FAMILIADESTINO := 112;

  CAMBIARAGENTESFAMILIA (
    ID_FAMILIAORIGEN => ID_FAMILIAORIGEN,
    ID_FAMILIADESTINO => ID_FAMILIADESTINO
  );

  --rollback;
END;

```

Ejecutando: IdeConnections%23conAGENCIA-jpr - Log

Conectando a la base de datos conAGENCIA.

Se han trasladado 3 agentes de familia Madrid-1.1 a la familia Madrid-1.2

El proceso ha terminado.

Desconectando de la base de datos conAGENCIA.

Ejercicio 2.

Queremos controlar algunas restricciones a la hora de trabajar con agentes:

- ✓ La longitud de la clave de un agente no puede ser inferior a 6.
- ✓ La habilidad de un agente debe estar comprendida entre 0 y 9 (ambos inclusive).
- ✓ La categoría de un agente sólo puede ser igual a 0, 1 o 2.
- ✓ Si un agente tiene categoría 2 no puede pertenecer a ninguna familia y debe pertenecer a una oficina.
- ✓ Si un agente tiene categoría 1 no puede pertenecer a ninguna oficina y debe pertenecer a una familia.
- ✓ Todos los agentes deben pertenecer a una oficina o a una familia pero nunca a ambas a la vez.

Se pide crear un disparador para asegurar estas restricciones. El disparador deberá lanzar todos los errores que se puedan producir en su ejecución mediante errores que identifiquen con un mensaje adecuado por qué se ha producido dicho error.

Algunas de las restricciones implementadas con el disparador se pueden incorporar a la definición del esquema de la tabla utilizando el Lenguaje de Definición de Datos (Check, Unique,...). Identifica cuáles son y con qué tipo de restricciones las implementarías.

SOLUCION

1. Implementar el TRIGGER y compilar.

The screenshot shows the SQL Developer interface. On the left, the 'Conexiones' pane shows the 'INTEGRIDAD_AGENTES' schema selected. The main editor window displays the following SQL code for creating a trigger:

```

CREATE OR REPLACE TRIGGER Integridad_Agentes -- Crear un disparador
BEFORE INSERT OR UPDATE ON Agentes -- que se ejecute antes de hacer una 'inserción' o 'modificación' en la tabla 'agentes'
FOR EACH ROW -- para cada fila a insertar o actualizar
BEGIN
    --Comprobamos que la longitud de la clave no es inferior a 6
    IF ( LENGTH(:new.clave) < 6) THEN
        RAISE_APPLICATION_ERROR(-20021, 'La longitud de la clave no puede ser inferior a 6 caracteres');
    END IF;

    --Comprobamos que la habilidad del agente está comprendida entre 0 y 9
    IF (:new.habilidad < 0 OR :new.habilidad > 9) THEN
        RAISE_APPLICATION_ERROR(-20022, 'La habilidad del agente es errónea');
    END IF;

    --Comprobamos que la categoría del agente está comprendida entre 0 y 2
    IF (:new.categoria < 0 OR :new.categoria > 2) THEN
        RAISE_APPLICATION_ERROR(-20023, 'La categoría del agente es errónea');
    END IF;

    -- Todos los agentes deben pertenecer a una oficina o a una familia pero nunca a ambas a la vez.
    IF (:new.oficina IS NOT NULL and :new.familia IS NOT NULL) or (:new.oficina IS NULL and :new.familia IS NULL) THEN
        RAISE_APPLICATION_ERROR(-20024, 'Los agentes deben pertenecer a una oficina o a una familia según su categoría');
    END IF;

    IF (:new.familia IS NULL and :new.categoria <> 2) THEN
        RAISE_APPLICATION_ERROR(-20025, 'Si el agente no es supervisor(categoría 2) debe pertenecer a una familia');
    END IF;

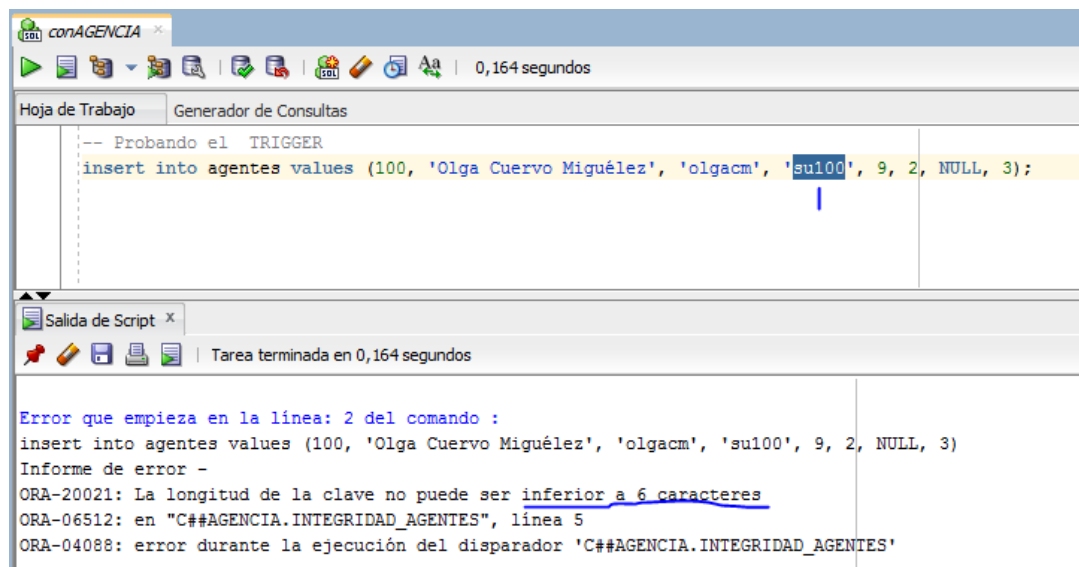
    IF (:new.oficina IS NULL and :new.categoria = 2) THEN
        RAISE_APPLICATION_ERROR(-20026, 'La categoría de un agente que no pertenece a una oficina directamente debe ser distinta de 2');
    END IF;
END;

```

At the bottom, the 'Salida de Script' pane shows the message: 'Tarea terminada en 0,406 segundos' and 'Trigger INTEGRIDAD_AGENTES compilado'.

2. Iniciamos las pruebas en el orden en el que aparecen creadas.

a. Longitud de la CLAVE



```
-- Probando el TRIGGER
insert into agentes values (100, 'Olga Cuervo Miguélez', 'olgacm', 'su100', 9, 2, NULL, 3);
```

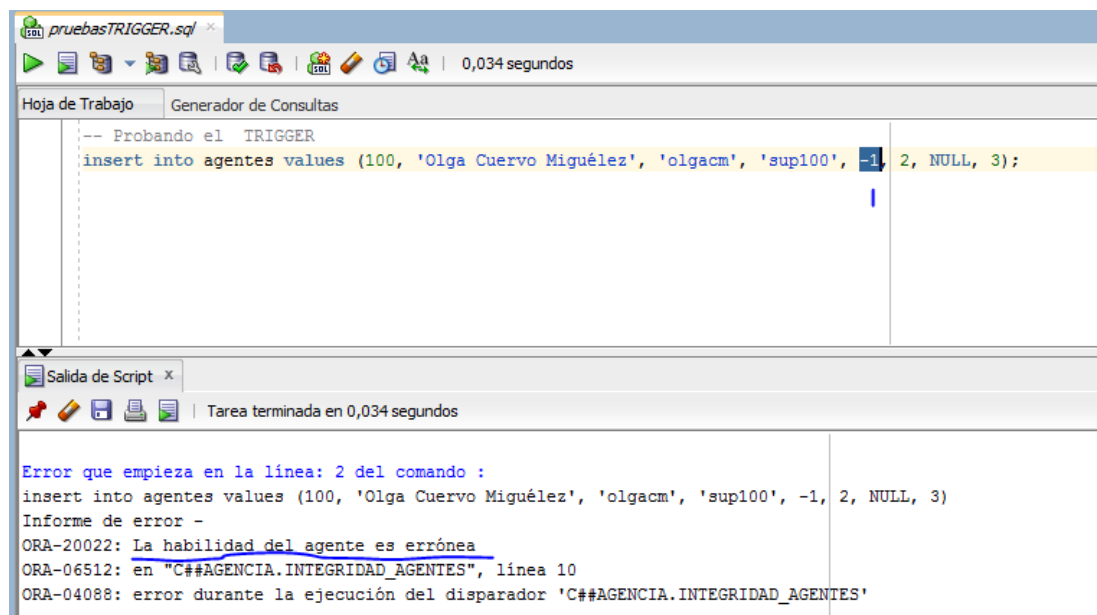
Salida de Script x

Tarea terminada en 0,164 segundos

Error que empieza en la línea: 2 del comando :
insert into agentes values (100, 'Olga Cuervo Miguélez', 'olgacm', 'su100', 9, 2, NULL, 3)
Informe de error -
ORA-20021: La longitud de la clave no puede ser inferior a 6 caracteres
ORA-06512: en "C##AGENCIA.INTEGRIDAD_AGENTES", línea 5
ORA-04088: error durante la ejecución del disparador 'C##AGENCIA.INTEGRIDAD_AGENTES'

b. Habilidad entre 0 y 9.

i. Probando con -1



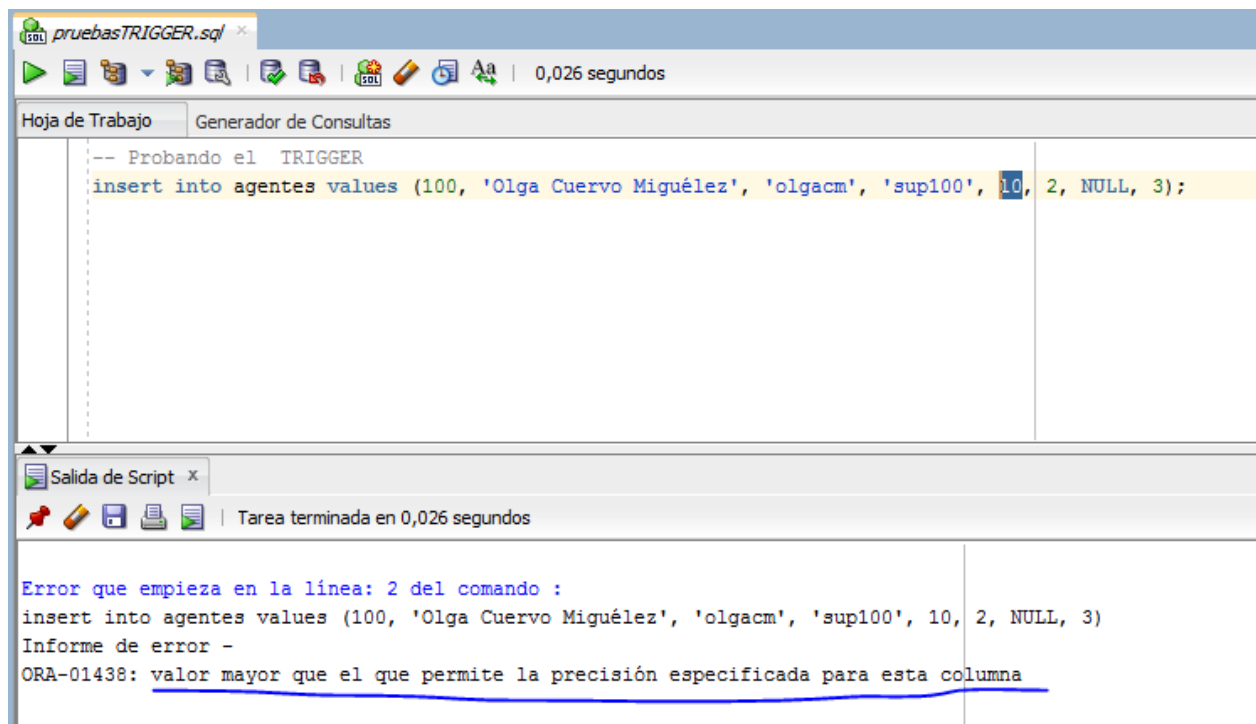
```
-- Probando el TRIGGER
insert into agentes values (100, 'Olga Cuervo Miguélez', 'olgacm', 'sup100', -1, 2, NULL, 3);
```

Salida de Script x

Tarea terminada en 0,034 segundos

Error que empieza en la línea: 2 del comando :
insert into agentes values (100, 'Olga Cuervo Miguélez', 'olgacm', 'sup100', -1, 2, NULL, 3)
Informe de error -
ORA-20022: La habilidad del agente es errónea
ORA-06512: en "C##AGENCIA.INTEGRIDAD_AGENTES", línea 10
ORA-04088: error durante la ejecución del disparador 'C##AGENCIA.INTEGRIDAD_AGENTES'

ii. Probando con 10.

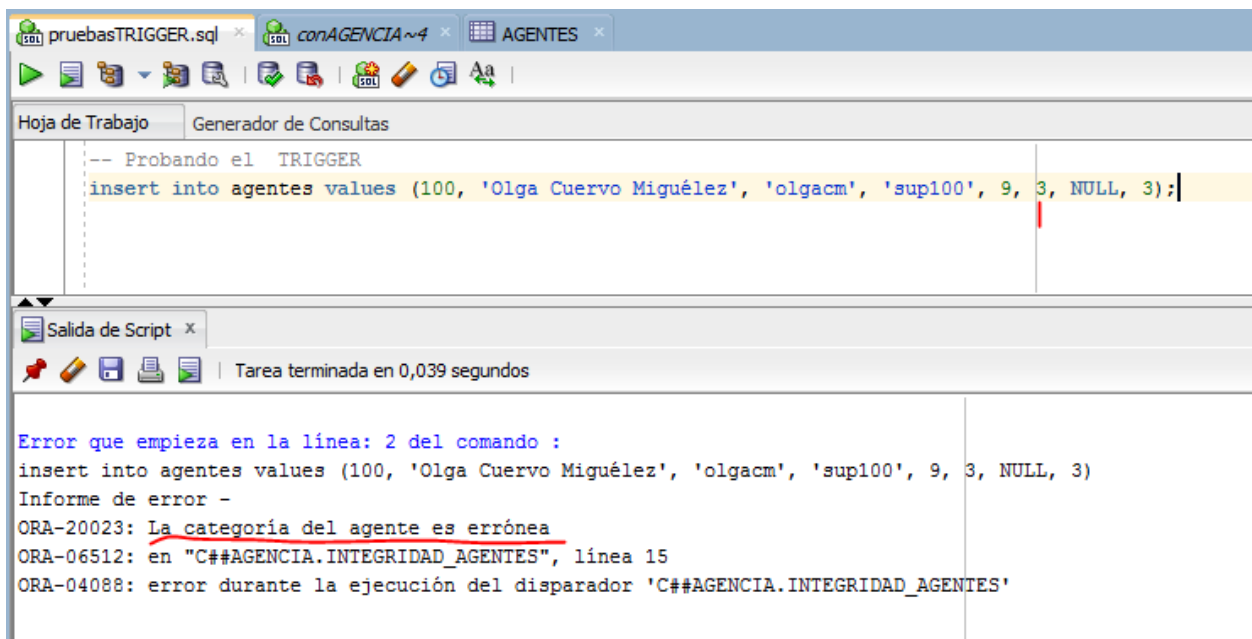


CUIDADO!!!! El error que muestra NO es el del TRIGGER sino el que corresponde por estar intentando insertar un número de dos dígitos en un campo que solo admite 1 entonces 'salta' antes que el del trigger . Ver definición de la tabla. Como opción podéis modificar el límite superior en el trigger y probar con el valor 9, por ejemplo.

The screenshot shows the 'AGENTES' table structure in SQL Developer. The table has 8 columns. The 'HABILIDAD' column is highlighted in blue, and its data type 'NUMBER(1,0)' is underlined in red.

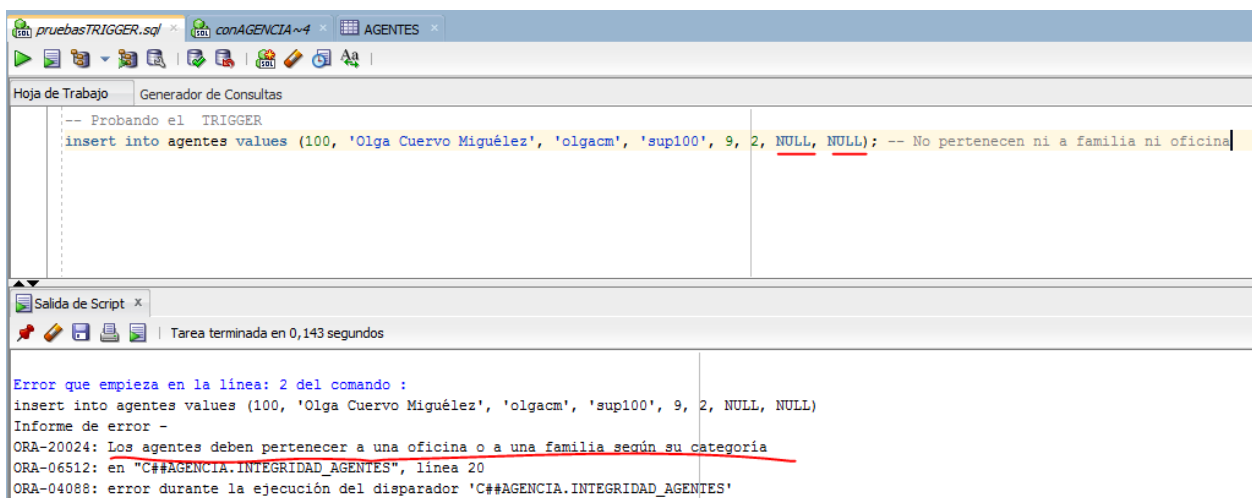
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	IDENTIFICADOR	NUMBER(6,0)	No	(null)	1	(null)
2	NOMBRE	VARCHAR2(60 BYTE)	No	(null)	2	(null)
3	USUARIO	VARCHAR2(20 BYTE)	No	(null)	3	(null)
4	CLAVE	VARCHAR2(20 BYTE)	No	(null)	4	(null)
5	HABILIDAD	NUMBER(1,0)	No	(null)	5	(null)
6	CATEGORIA	NUMBER(1,0)	No	(null)	6	(null)
7	FAMILIA	NUMBER(6,0)	Yes	(null)	7	(null)
8	OFICINA	NUMBER(6,0)	Yes	(null)	8	(null)

- c. Categoría entre 0 y 2.

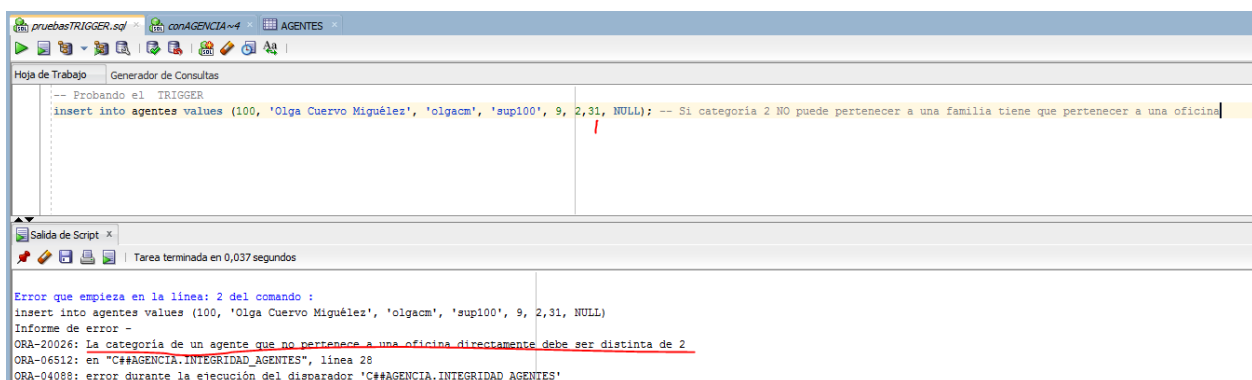


The screenshot shows the SQL Developer interface with a script named 'pruebasTRIGGER.sql'. The script contains a comment '-- Probando el TRIGGER' followed by an SQL insert statement: `insert into agentes values (100, 'Olga Cuervo Miguélez', 'olgacm', 'sup100', 9, 3, NULL, 3);`. The output pane shows the execution result: 'Tarea terminada en 0,039 segundos'. Below this, an error message is displayed: 'Error que empieza en la línea: 2 del comando : insert into agentes values (100, 'Olga Cuervo Miguélez', 'olgacm', 'sup100', 9, 3, NULL, 3) Informe de error - ORA-20023: La categoría del agente es errónea ORA-06512: en "C##AGENCIA.INTEGRIDAD_AGENTES", línea 15 ORA-04088: error durante la ejecución del disparador 'C##AGENCIA.INTEGRIDAD_AGENTES''.

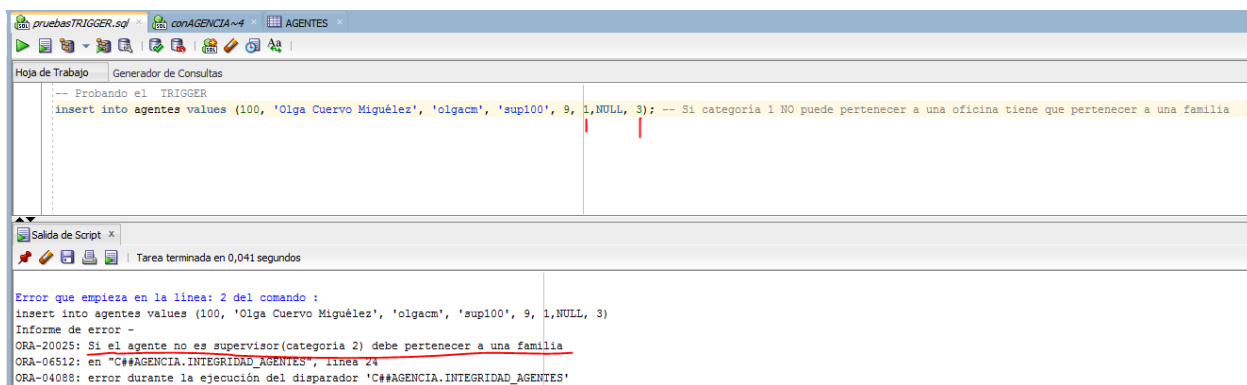
- d. Pertener a oficina o familia.



The screenshot shows the SQL Developer interface with a script named 'pruebasTRIGGER.sql'. The script contains a comment '-- Probando el TRIGGER' followed by an SQL insert statement: `insert into agentes values (100, 'Olga Cuervo Miguélez', 'olgacm', 'sup100', 9, 2, NULL, NULL); -- No pertenecen ni a familia ni oficina`. The output pane shows the execution result: 'Tarea terminada en 0,143 segundos'. Below this, an error message is displayed: 'Error que empieza en la línea: 2 del comando : insert into agentes values (100, 'Olga Cuervo Miguélez', 'olgacm', 'sup100', 9, 2, NULL, NULL) Informe de error - ORA-20024: Los agentes deben pertenecer a una oficina o a una familia según su categoría ORA-06512: en "C##AGENCIA.INTEGRIDAD_AGENTES", línea 20 ORA-04088: error durante la ejecución del disparador 'C##AGENCIA.INTEGRIDAD_AGENTES''.



The screenshot shows the SQL Developer interface with a script named 'pruebasTRIGGER.sql'. The script contains a comment '-- Probando el TRIGGER' followed by an SQL insert statement: `insert into agentes values (100, 'Olga Cuervo Miguélez', 'olgacm', 'sup100', 9, 2,31, NULL); -- Si categoria 2 NO puede pertenecer a una familia tiene que pertenecer a una oficina`. The output pane shows the execution result: 'Tarea terminada en 0,037 segundos'. Below this, an error message is displayed: 'Error que empieza en la línea: 2 del comando : insert into agentes values (100, 'Olga Cuervo Miguélez', 'olgacm', 'sup100', 9, 2,31, NULL) Informe de error - ORA-20026: La categoría de un agente que no pertenece a una oficina directamente debe ser distinta de 2 ORA-06512: en "C##AGENCIA.INTEGRIDAD_AGENTES", línea 28 ORA-04088: error durante la ejecución del disparador 'C##AGENCIA.INTEGRIDAD_AGENTES''.



The screenshot shows the SQL Developer interface. The top toolbar includes icons for file operations, execution, and debugging. The main window is titled 'pruebasTRIGGER.sql' and contains the following SQL script:

```
-- Probando el TRIGGER
insert into agentes values (100, 'Olga Cuervo Miguélez', 'olgacm', 'sup100', 9, 1, NULL, 3); -- Si categoría 1 NO puede pertenecer a una oficina tiene que pertenecer a una familia
```

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

```
Tarea terminada en 0,041 segundos

Error que empieza en la línea: 2 del comando :
insert into agentes values (100, 'Olga Cuervo Miguélez', 'olgacm', 'sup100', 9, 1, NULL, 3)
Informe de error -
ORA-20025: Si el agente no es supervisor(categoría 2) debe pertenecer a una familia
ORA-06512: en "C##AGENCIA.INTEGRIDAD_AGENTES", línea 24
ORA-04088: error durante la ejecución del disparador 'C##AGENCIA.INTEGRIDAD_AGENTES'
```