Tarea Unidad 6
Programación de bases de datos

Base de datos

Curso DAM 2024/25

Calificación obtenida: 10/10

ÍNDICE

Descripción de la tarea **3**
Actividad 1 ..... **3**
Actividad 2 ..... **10**
2.1. Crear un disparador **10**
Restricción 1: La longitud de la clave de un agente no puede ser inferior a 6 11
Restricción 2: La habilidad de un agente debe estar comprendida entre 0 y 9 (ambos inclusive) 12
Restricción 3: La categoría de un agente sólo puede ser 0, 1 o 2 13
Restricción 4: Si un agente tiene categoría 2 no puede pertenecer a ninguna familia y debe pertenecer a una oficina 14
Restricción 5: Si un agente tiene categoría 1 no puede pertenecer a ninguna oficina y debe pertenecer a una familia 17
Restricción 6: Todos los agentes deben pertenecer a una oficina o a una familia pero nunca a ambas a la vez 18
2.2. Restricciones de esquema utilizando DDL **20**
Restricción 1: La longitud de la clave de un agente no puede ser inferior a 6 21
Restricción 2: La habilidad de un agente debe estar comprendida entre 0 y 9 (ambos inclusive) 22
Restricción 3: La categoría de un agente sólo puede ser 0, 1 o 2 23
Restricción 4: Si un agente tiene categoría 2 no puede pertenecer a ninguna familia y debe pertenecer a una oficina 24
Restricción 5: Si un agente tiene categoría 1 no puede pertenecer a ninguna oficina y debe pertenecer a una familia 26
Restricción 6: Todos los agentes deben pertenecer a una oficina o a una familia pero nunca a ambas a la vez 27
Anexo **29**
Caso de estudio 29
Script 30
Disparador 33

Descripción de la tarea

Para la realización de la tarea 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:

Actividad 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.

Crear un procedimiento que permita cambiar a todos los agentes de una familia determinada (familia origen) a otra familia (familia destino).
Crear un sinónimo público
Crear sinónimos públicos para todas las tablas en el esquema C##AGENCIA
Lo primero que he hecho es crear un sinónimo público para todas las tablas del esquema C##AGENCIA para poder acceder a ellas sin tener que escribir el esquema completo cada vez.

```
BEGIN
    FOR t IN (SELECT table_name FROM all_tables WHERE owner = 'C##AGENCIA') LOOP
        EXECUTE IMMEDIATE 'CREATE PUBLIC SYNONYM ' || t.table_name || ' FOR C##AGENCIA.' || t.table_name;
    END LOOP;
END;
/
```

Verificar si los sinónimos públicos se han creado correctamente

```
SELECT * FROM all_synonyms WHERE owner = 'PUBLIC' AND table_owner = 'C##AGENCIA';
```

Salida de Script

Tarea terminada en 0,374 segundos

OWNER	SYNONYM_NAME
PUBLIC	AGENTES
PUBLIC	FAMILIAS
PUBLIC	OFICINAS

Comprender la estructura de datos

- La tabla AGENTES tiene una columna FAMILIA que indica a qué familia pertenece cada agente.
- La tabla FAMILIAS tiene la columna IDENTIFICADOR, que es la clave primaria.
- Queremos cambiar a todos los agentes de una familia origen a una familia destino, validando que ambas existen y son diferentes.

Escribir la estructura general del procedimiento

Crear la estructura del procedimiento

- Creamos un procedimiento llamado CambiarAgentesFamilia.
- Definimos dos parámetros de entrada (id_FamiliaOrigen y id_FamiliaDestino).
- NULL; indica que aún no hemos agregado la lógica.

```
CREATE OR REPLACE PROCEDURE CambiarAgentesFamilia (
    id_FamiliaOrigen IN NUMBER,
    id_FamiliaDestino IN NUMBER
)
IS
BEGIN
    NULL; -- Aquí agregaremos la lógica en los siguientes pasos
END;
/
```

Verificar que las familias existen

Antes de mover los agentes, necesitamos asegurarnos de que ambas familias existen en la tabla FAMILIAS.

- Declaramos dos variables de tipo NUMBER:
  - v_familia_origen_existente: Guardará el resultado de la verificación de existencia de la familia origen.
  - v_familia_destino_existente: Guardará el resultado de la verificación de existencia de la familia destino.
- Comprobación de la existencia de la familia origen:
  - SELECT COUNT(*): Cuenta cuántas filas existen en la tabla familias con un identificador igual a id_FamiliaOrigen (que es el parámetro de entrada).

- INTO v_familia_origen_existente: El resultado de COUNT(*) se guarda en la variable v_familia_origen_existente. Si la familia existe, el resultado será un número mayor que 0, y si no existe, el resultado será 0.
- FROM familias WHERE identificador = id_FamiliaOrigen;: Aquí estamos seleccionando la tabla familias y buscando las filas que tengan el identificador igual al valor de id_FamiliaOrigen.

- Comprobación de la existencia de la familia destino:
  - Similar a la verificación anterior, pero ahora verificamos si la familia destino (id_FamiliaDestino) existe en la tabla familias.
  - Guardamos el resultado en la variable v_familia_destino_existente.

```
CREATE OR REPLACE PROCEDURE CambiarAgentesFamilia (
    id_FamiliaOrigen IN NUMBER,
    id_FamiliaDestino IN NUMBER
) IS
    v_familia_origen_existente NUMBER;
    v_familia_destino_existente NUMBER;
BEGIN
    -- Verificamos que la familia origen exista
    SELECT COUNT(*) INTO v_familia_origen_existente
    FROM familias
    WHERE identificador = id_FamiliaOrigen;

    -- Verificamos que la familia destino exista
    SELECT COUNT(*) INTO v_familia_destino_existente
    FROM familias
    WHERE identificador = id_FamiliaDestino;

    -- Fin del procedimiento sin ninguna acción adicional
    NULL;
END
/
```

Comprobar si las familias son iguales

Necesitamos comprobar que las familias de origen y destino no sean la misma. Para ello, dentro del bloque BEGIN...END, agregamos una nueva verificación para asegurarnos de que las familias no sean iguales.

- IF id_FamiliaOrigen = id_FamiliaDestino: Esta línea comprueba si los identificadores de las dos familias son iguales.
- RAISE_APPLICATION_ERROR(-20003, *'Las familias origen y destino no pueden ser iguales.'*): Si las familias son iguales, lanzamos un error personalizado con el número de error -20003 y un mensaje que explica el problema.

```
-- Verificamos si las familias son iguales
IF id_FamiliaOrigen = id_FamiliaDestino THEN
RAISE_APPLICATION_ERROR(-20003, 'Las familias origen y destino no pueden ser iguales');
END IF;
```

Obtener los nombres de las familias

Como al final del proceso, queremos mostrar un mensaje al usuario informándole sobre cuántos agentes han sido trasladados de una familia a otra, si incluimos los nombres de las familias, en lugar de solo sus identificadores (ID) el mensaje será más claro. El mensaje que queremos generar será algo así como: "X agentes han sido trasladados de la familia 'Familia A' a la familia 'Familia B'".

- Agregamos las variables que guardarán los nombres de las familias, para luego usarlas en el mensaje que mostraremos cuando movamos los agentes.

```
    --Variables para obtener los nombres de las familias
    v_nombre_familia_origen VARCHAR2(40);
    v_nombre_familia_destino VARCHAR2(40);
BEGIN
```

- Si las familias son diferentes, ahora obtenemos los nombres de las familias usando dos consultas SELECT que se guardan en las variables v_nombre_familia_origen y v_nombre_familia_destino.

```
    -- Obtener los nombres de las familias
    SELECT nombre INTO v_nombre_familia_origen
    FROM familias
    WHERE identificador = id_FamiliaOrigen;

    SELECT nombre INTO v_nombre_familia_destino
    FROM familias
    WHERE identificador = id_FamiliaDestino;
END;
/
```

Contar los agentes que serán afectados

Contamos cuántos agentes pertenecen a la familia de origen para saber cuántos agentes serán movidos y para proporcionar un mensaje de confirmación más claro.

- la variable v_agentes_afectados almacenará el número total de agentes que están actualmente asignados a la familia de origen.

```
    -- Variable para contar los agentes afectados
    v_agentes_afectados NUMBER;

BEGIN
```

- Utilizamos un SELECT COUNT(*) para contar estos agentes.

```
    -- Contamos cuántos agentes pertenecen a la familia origen
    SELECT COUNT(*) INTO v_agentes_afectados
    FROM agentes
    WHERE familia = id_FamiliaOrigen;
END;
/
```

Verificar si hay agentes a mover

Si no hay agentes en la familia origen, lanzamos un error diciendo que no hay agentes para mover.

```
    -- Verificamos si hay agentes afectados
    IF v_agentes_afectados = 0 THEN
        RAISE_APPLICATION_ERROR(-20004, 'No hay agentes en la familia origen para mover');
    END IF;
END;
```

Actualizar los agentes que pertenecen a la familia de origen

Una vez que sabemos cuántos agentes están en la familia de origen, actualizamos su información para que ahora pertenezcan a la familia de destino. Esto lo hacemos mediante una instrucción UPDATE. Es decir, estamos actualizando los registros en la tabla AGENTES, cambiando el valor de la columna familia de los agentes que actualmente están en la familia de origen, para asignarlos a la familia de destino.

```
    -- Actualizamos los agentes que pertenecen a la familia origen y los movemos a la familia destino
    UPDATE agentes
    SET familia = id_FamiliaDestino
    WHERE familia = id_FamiliaOrigen;
END;
/
```

Mensaje de confirmación

Queremos mostrar un mensaje que indique cuántos agentes han sido trasladados. Para esto, usamos el paquete DBMS_OUTPUT de Oracle.

- Commit: COMMIT guarda los cambios que hemos realizado (en este caso, la actualización de los agentes).
- Usamos DBMS_OUTPUT.PUT_LINE para imprimir el mensaje en la consola con el número de agentes trasladados y los nombres de las familias.

```
    -- Mostrar el mensaje de los agentes trasladados
    COMMIT; -- Guardamos los cambios

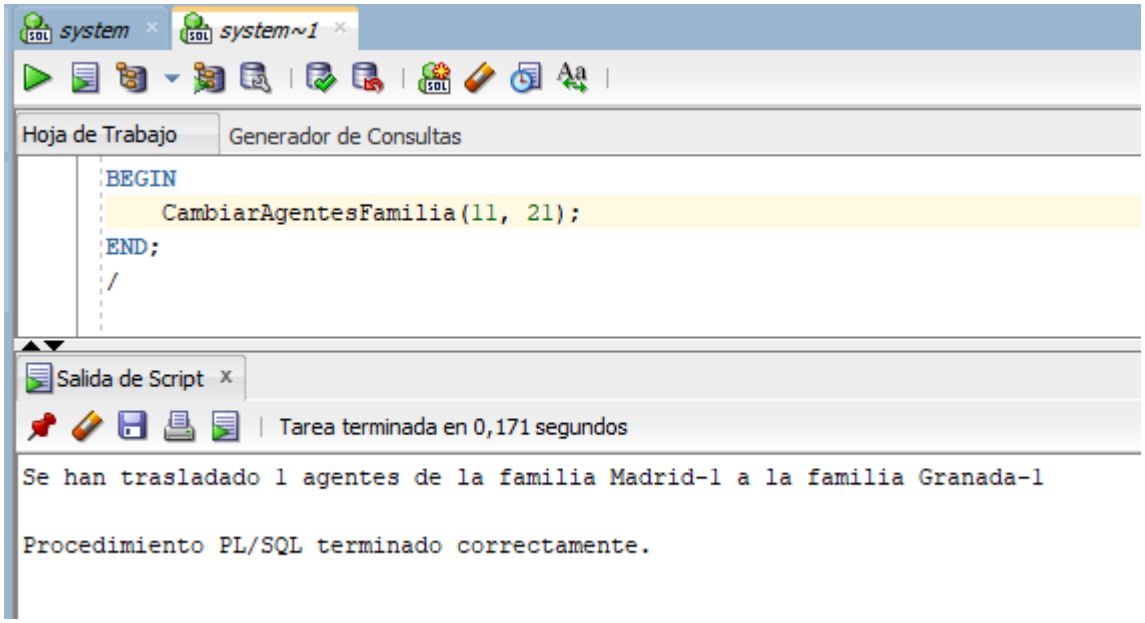
    DBMS_OUTPUT.PUT_LINE('Se han trasladado ' || v_agentes_afectados || ' agentes de la familia ' || v_nombre_familia_origen || ' a la familia ' || v_nombre_familia_destino);
END;
```

Probar el procedimiento

Verificar la salida del procedimiento

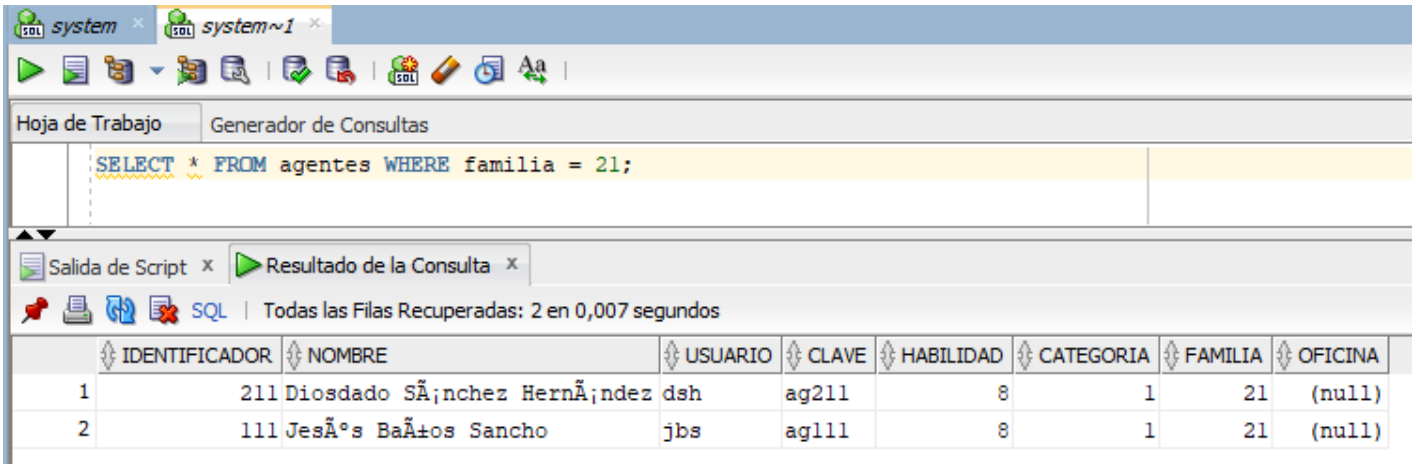
Ejecutamos una llamada al procedimiento:

- 11: Familia de origen (agentes que se van a mover).
- 21: Familia de destino (a dónde se van a mover los agentes).



Verificar los cambios en la base de datos

Ejecutamos la siguiente consulta y muestra los agentes que ahora pertenecen a la familia 21



	IDENTIFICADOR	NOMBRE	USUARIO	CLAVE	HABILIDAD	CATEGORIA	FAMILIA	OFICINA
1	211	Diosdado SÃ¡nchez HernÃ¡ndez	dsh	ag211	8	1	21	(null)
2	111	JesÃºs BaÃ±os Sancho	jbs	ag111	8	1	21	(null)

CÓDIGO RESULTANTE

```
CREATE OR REPLACE PROCEDURE CambiarAgentesFamilia (
    id_FamiliaOrigen IN NUMBER,
    id_FamiliaDestino IN NUMBER
) IS
    v_familia_origen_existente NUMBER;
    v_familia_destino_existente NUMBER;
```

```
    --Variables para obtener los nombres de las familias
    v_nombre_familia_origen VARCHAR2(40);
    v_nombre_familia_destino VARCHAR2(40);

    -- Variable para contar los agentes afectados
    v_agentes_afectados NUMBER;

BEGIN
    -- Verificamos que la familia origen exista
    SELECT COUNT(*) INTO v_familia_origen_existente
    FROM familias
    WHERE identificador = id_FamiliaOrigen;

    -- Verificamos que la familia destino exista
    SELECT COUNT(*) INTO v_familia_destino_existente
    FROM familias
    WHERE identificador = id_FamiliaDestino;

    -- Verificamos si las familias son iguales
    IF id_FamiliaOrigen = id_FamiliaDestino THEN
        RAISE_APPLICATION_ERROR(-20003, 'Las familias origen y destino no pueden ser
iguales');
    END IF;

    -- Obtener los nombres de las familias
    SELECT nombre INTO v_nombre_familia_origen
    FROM familias
    WHERE identificador = id_FamiliaOrigen;

    SELECT nombre INTO v_nombre_familia_destino
    FROM familias
    WHERE identificador = id_FamiliaDestino;

    -- Contamos cuántos agentes pertenecen a la familia origen
    SELECT COUNT(*) INTO v_agentes_afectados
    FROM agentes
    WHERE familia = id_FamiliaOrigen;

    -- Verificamos si hay agentes afectados
    IF v_agentes_afectados = 0 THEN
        RAISE_APPLICATION_ERROR(-20004, 'No hay agentes en la familia origen para mover');
```

```
    END IF;

    -- Actualizamos los agentes que pertenecen a la familia origen y los movemos a la
familia destino
    UPDATE agentes
    SET familia = id_FamiliaDestino
    WHERE familia = id_FamiliaOrigen;

    -- Mostrar el mensaje de los agentes trasladados
    COMMIT; -- Guardamos los cambios

    DBMS_OUTPUT.PUT_LINE('Se han trasladado ' || v_agentes_afectados || ' agentes de la
familia ' || v_nombre_familia_origen || ' a la familia ' || v_nombre_familia_destino);
END;
/
```

Actividad 2.

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

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

2.1. Crear un disparador

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.

Restricción 1: La longitud de la clave de un agente no puede ser inferior a 6.

Código en PL/SQL para la Restricción 1

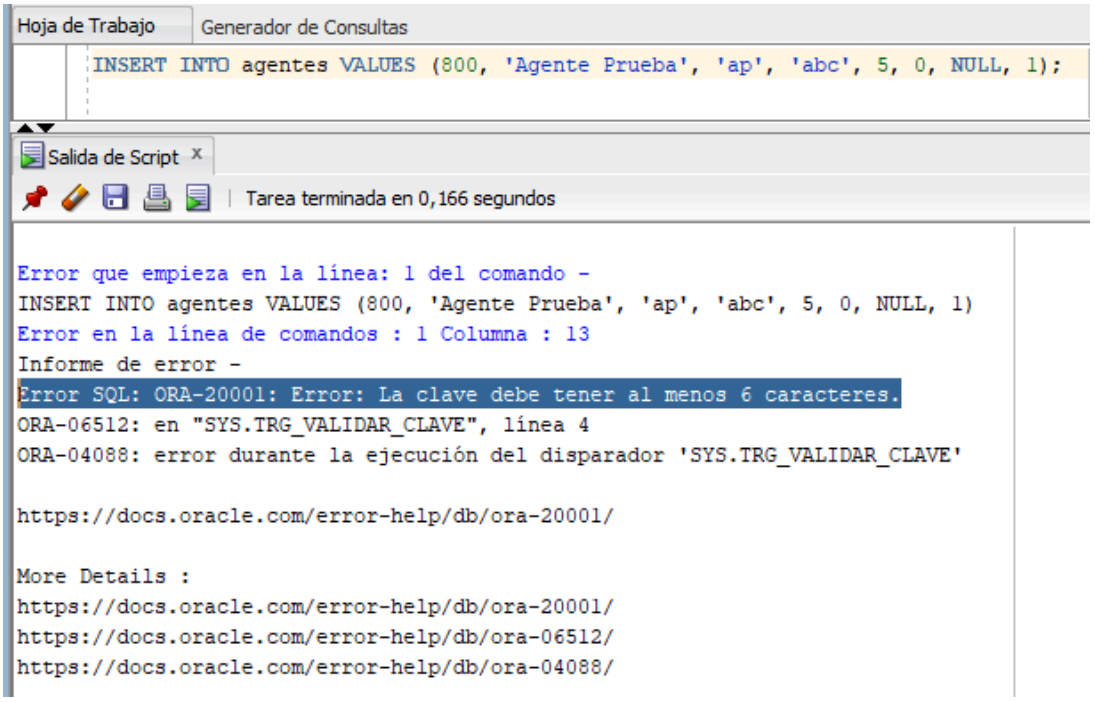
- Cuando un agente se inserte o se actualice, su clave (clave) debe tener al menos 6 caracteres.
- Si la clave tiene menos de 6 caracteres, lanzaremos un error con RAISE_APPLICATION_ERROR.

IF LENGTH(:NEW.clave) < 6 THEN
RAISE_APPLICATION_ERROR(-20001, 'Error: La clave debe tener al menos 6 caracteres.');
END IF;

```
-- Restricción 1: La clave debe tener al menos 6 caracteres
IF LENGTH(:NEW.clave) < 6 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Error: La clave debe tener al menos 6 caracteres.');
END IF;
```

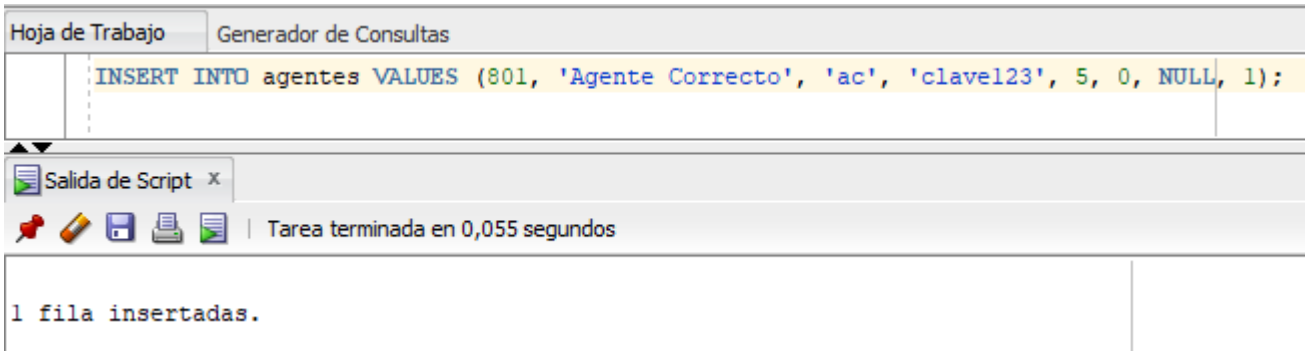
Prueba de la restricción 1 con un caso incorrecto: Un agente con una clave menor a 6 caracteres

INSERT INTO agentes
VALUES (800, ‘Agente prueba’, ‘ap’, ‘abc’, 5, 0, NULL, 1);



Prueba de la restricción 1 con un caso correcto: Agente con una clave de 7 caracteres

INSERT INTO agentes
VALUES (801, ‘Agente correcto’, ‘ac’, ‘clave123’, 5, 0, NULL, 1);



Restricción 2: La habilidad de un agente debe estar comprendida entre 0 y 9 (ambos inclusive)

Código en PL/SQL para la Restricción 2

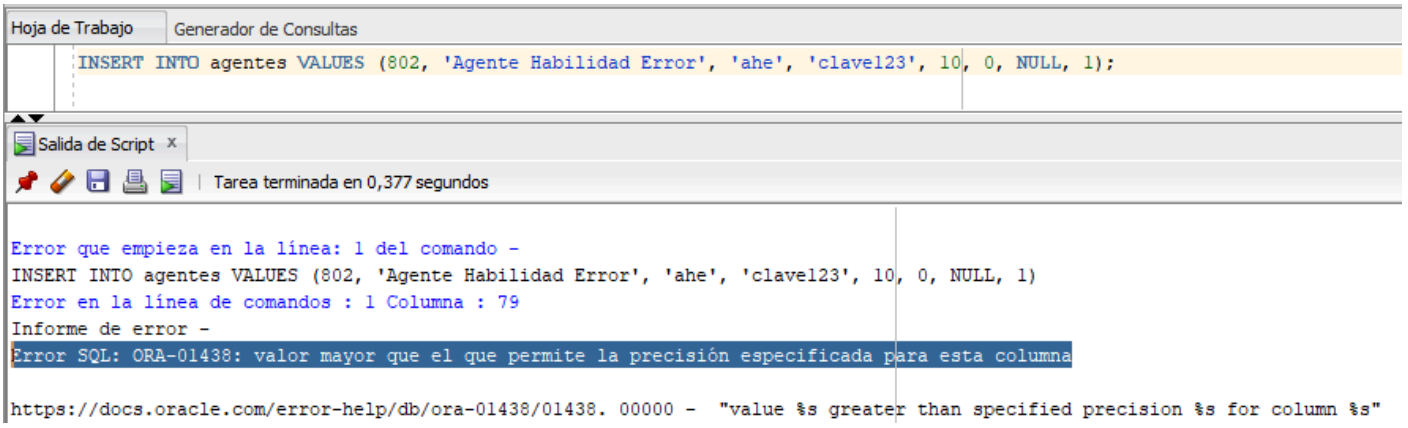
- La habilidad de un agente (habilidad) debe estar en el rango de 0 a 9, es decir, incluso el 0 y el 9 son válidos.
- Si la habilidad está fuera de este rango, se debe generar un error.

IF :NEW.habilidad NOT BETWEEN 0 AND 9 THEN
RAISE_APPLICATION_ERROR(-20002, ‘Error: La habilidad debe estar enter 0 y 9.’);
END IF;

```
-- Validar que la habilidad esté entre 0 y 9
IF :NEW.habilidad NOT BETWEEN 0 AND 9 THEN
    RAISE_APPLICATION_ERROR(-20002, 'Error: La habilidad debe estar entre 0 y 9.');
END IF;
```

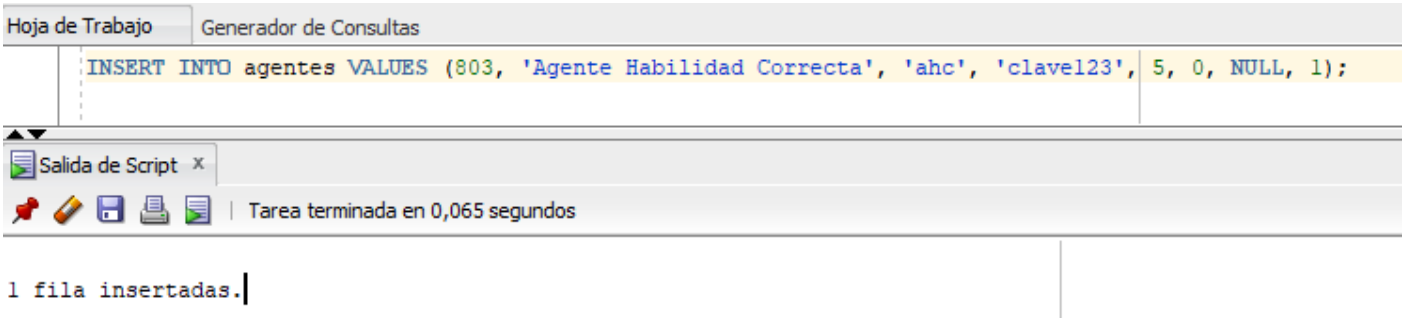
Prueba de la restricción 2 con un caso incorrecto: Un agente con habilidad 10

INSERT INTO agentes
VALUES (802, ‘Agente habilidad error’, ‘ahe’, ‘clave123’, 10, 0, NULL, 1);



Prueba de la restricción 2 con un caso correcto: Un agente con habilidad 5

INSERT INTO agentes
VALUES (803, 'Agente habilidad correcta', 'ahc' 'clave123', 5, 0, NULL, 1);



Restricción 3: La categoría de un agente sólo puede ser 0, 1 o 2

Código en PL/SQL para la Restricción 3

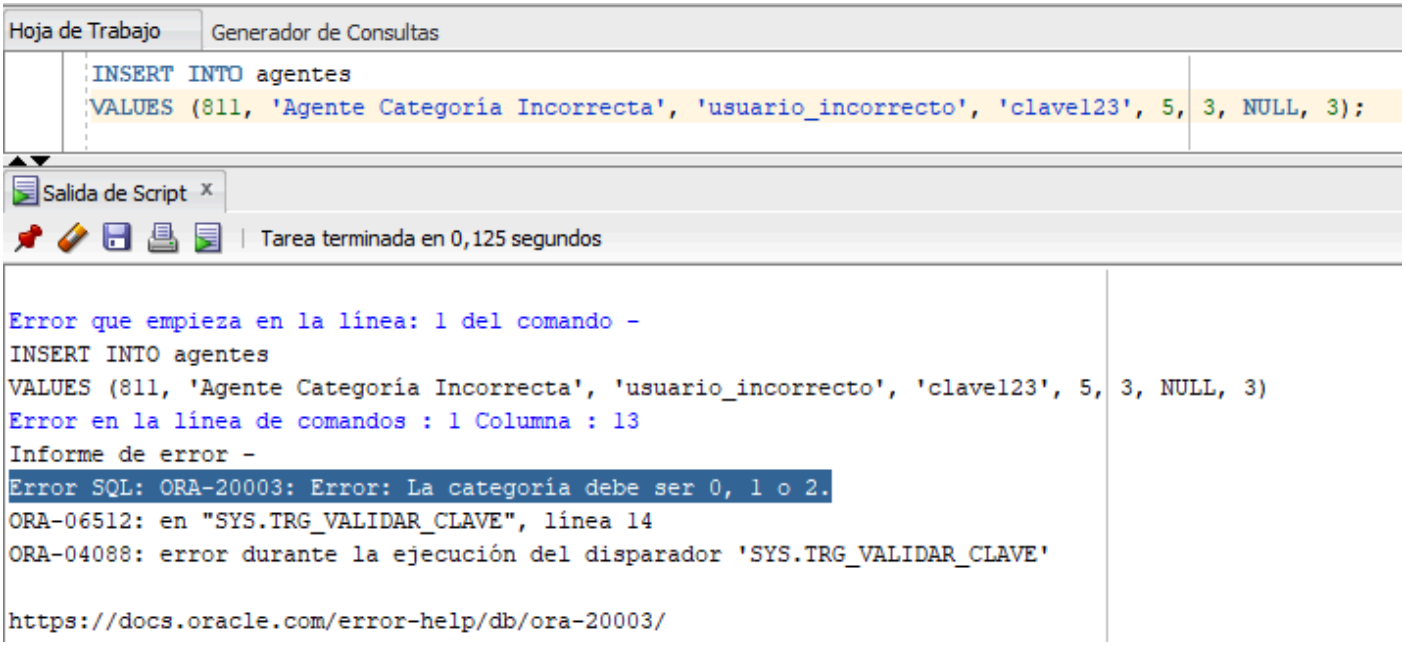
- La categoría de un agente debe ser uno de los siguientes valores: 0, 1 o 2.
- Si se intenta insertar o actualizar con un valor distinto, el disparador debe lanzar un error.

```
IF :NEW.categoria NOT IN (0, 1, 2) THEN
    RAISE_APPLICATION_ERROR(-20003, 'Error: La categoría solo puede ser 0, 1 o 2.');
END IF;
```

```
    -- Restricción 3: La categoría debe ser 0, 1 o 2
    IF :NEW.categoria NOT IN (0, 1, 2) THEN
        RAISE_APPLICATION_ERROR(-20003, 'Error: La categoría debe ser 0, 1 o 2.');
    END IF;
END;
/
```

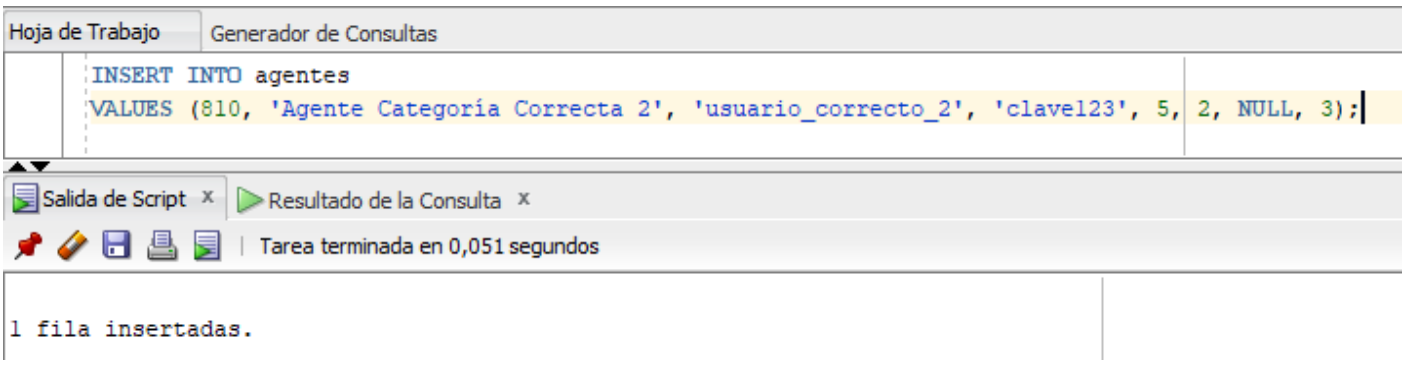
Prueba de la restricción 3 con un caso incorrecto: Un agente con categoría 3.

INSERT INTO agentes
VALUES (811, ‘Agente Categoría Incorrecta’, ‘usuario_incorrecto’, ‘clave123’, 5, 3, NULL, 3);



Prueba de la restricción 3 con un caso correcto: Un agente con categoría 2.

INSERT INTO agentes
VALUES (810, ‘Agente Categoría Correcta 2’. ‘usuario_correcto_2’, ‘clave123’, 5, 2, NULL, 3);



Restricción 4: Si un agente tiene categoría 2 no puede pertenecer a ninguna familia y debe pertenecer a una oficina.

Código en PL/SQL para la Restricción 4

- Si un agente tiene categoría 2, no puede pertenecer a ninguna familia y debe pertenecer a

una oficina.

IF :NEW.categoria = 2 AND (:NEW.familia IS NOT NULL OR :NEW.oficina IS NULL) THEN
RAISE_APPLICATION_ERROR(-20004, 'Error: Los agentes con categoría 2 no pueden pertenecer a una familia y deben pertenecer a una oficina.');
END IF;

```
-- Restricción 4: Validar que si la categoría es 2, no pertenezca a ninguna familia y sí a una oficina
IF :NEW.categoria = 2 AND (:NEW.familia IS NOT NULL OR :NEW.oficina IS NULL) THEN
    RAISE_APPLICATION_ERROR(-20004, 'Error: Los agentes con categoría 2 no pueden pertenecer a una familia y deben pertenecer a una oficina.');
END IF;
```

Prueba de la restricción 4 con un caso incorrecto: Un agente con categoría 2 pertenece a una familia y pertenece a una oficina

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10001, 'Laura Sánchez', 'lsanchez', 'clave9876', 6, 2, 31, 2);

```
-- Intentamos insertar un agente con categoría 2, que pertenece tanto a una oficina como a una familia
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10001, 'Laura Sánchez', 'lsanchez', 'clave9876', 6, 2, 31, 2);
```

Salida de Script

Tarea terminada en 0,118 segundos

```
Error que empieza en la línea: 2 del comando -
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10001, 'Laura Sánchez', 'lsanchez', 'clave9876', 6, 2, 31, 2)
Error en la línea de comandos : 2 Columna : 13
Informe de error -
Error SQL: ORA-20004: Error: Los agentes con categoría 2 no pueden pertenecer a una familia y deben pertenecer a una oficina.
ORA-06512: en "SYS.TRG_VALIDAR_AGENTES", línea 19
ORA-04088: error durante la ejecución del disparador 'SYS.TRG_VALIDAR_AGENTES'
```

Prueba de la restricción 4 con un caso incorrecto:
Un agente con categoría 2 no pertenece a una familia y no pertenece a una oficina

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10002, 'Carlos Martínez', 'cmartinez', 'clave12345', 6, 2, NULL, NULL);

```
-- Intentamos insertar un agente con categoría 2, que no pertenece a ninguna familia ni a ninguna oficina
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10002, 'Carlos Martínez', 'cmartinez', 'clave12345', 6, 2, NULL, NULL);
```

```
Salida de Script
Tarea terminada en 0,056 segundos

Error que empieza en la línea: 2 del comando -
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10002, 'Carlos Martínez', 'cmartinez', 'clave12345', 6, 2, NULL, NULL)
Error en la línea de comandos : 2 Columna : 13
Informe de error -
Error SQL: ORA-20004: Error: Los agentes con categoría 2 no pueden pertenecer a una familia y deben pertenecer a una oficina.
ORA-06512: en "SYS.TRG_VALIDAR_AGENTES", línea 19
ORA-04088: error durante la ejecución del disparador 'SYS.TRG_VALIDAR_AGENTES'
```

Prueba de la restricción 4 con un caso incorrecto:
Un agente con categoría 2 pertenece a una familia y no pertenece a una oficina

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10003, ‘Ana Gómez’, ‘agomez’, ‘clave54321’, 8, 2, 31, NULL);

```
-- Intentamos insertar un agente con categoría 2, que pertenece a una familia pero no a una oficina
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10003, 'Ana Gómez', 'agomez', 'clave54321', 8, 2, 31, NULL);
```

```
Salida de Script
Tarea terminada en 0,047 segundos

Error que empieza en la línea: 2 del comando -
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10003, 'Ana Gómez', 'agomez', 'clave54321', 8, 2, 31, NULL)
Error en la línea de comandos : 2 Columna : 13
Informe de error -
Error SQL: ORA-20004: Error: Los agentes con categoría 2 no pueden pertenecer a una familia y deben pertenecer a una oficina.
ORA-06512: en "SYS.TRG_VALIDAR_AGENTES", línea 19
ORA-04088: error durante la ejecución del disparador 'SYS.TRG_VALIDAR_AGENTES'
```

Prueba de la restricción 4 con un caso correcto:
Agente con Categoría 2 no pertenece a una familia y pertenece a una oficina

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10004, ‘Laura Martínez’, ‘lmartinez’, ‘clave67890’, 7, 2, NULL, 3);

```
-- Intentamos insertar un agente con categoría 2, que no pertenece a ninguna familia y pertenece a una oficina
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10004, 'Laura Martínez', 'lmartinez', 'clave67890', 7, 2, NULL, 3);
```

Salida de Script

Tarea terminada en 0,048 segundos

```
1 fila insertadas.
```

Restricción 5: Si un agente tiene categoría 1 no puede pertenecer a ninguna oficina y debe pertenecer a una familia

Código en PL/SQL para la Restricción 5

- Si un agente tiene categoría 1:
  - Pertenece a una familia
  - No pertenece a una oficina

```
IF :NEW.categoria = 1 THEN
  -- Validar que no pertenezca a una oficina y sí pertenezca a una familia
    IF :NEW.oficina IS NOT NULL OR :NEW.familia IS NULL THEN
    RAISE_APPLICATION_ERROR(-20005, 'Error: Los agentes con categoría 1 no pueden pertenecer a una oficina y deben pertenecer a una familia.');
    END IF;
  END IF;
```

```
-- Restricción 5: Validar que si la categoría es 1, no pertenezca a ninguna oficina y sí a una familia
IF :NEW.categoria = 1 THEN
-- Validar que no pertenezca a una oficina y sí pertenezca a una familia
    IF :NEW.oficina IS NOT NULL OR :NEW.familia IS NULL THEN
    RAISE_APPLICATION_ERROR(-20005, 'Error: Los agentes con categoría 1 no pueden pertenecer a una oficina y deben pertenecer a una familia.');
    END IF;
END IF;
```

Prueba de la restricción 5 con un caso incorrecto
Agente con categoría 1 que pertenece a una oficina y no pertenece a una familia

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (100011, ‘Laura Gómez’, ‘lgomez’, ‘clave67890’, 5, 1, NULL, 102);

```
-- Intentar insertar un agente con categoría 1 que pertenece a una oficina y a ninguna familia
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10011, 'Laura Gómez', 'lgomez', 'clave67890', 5, 1, NULL, 102);
```

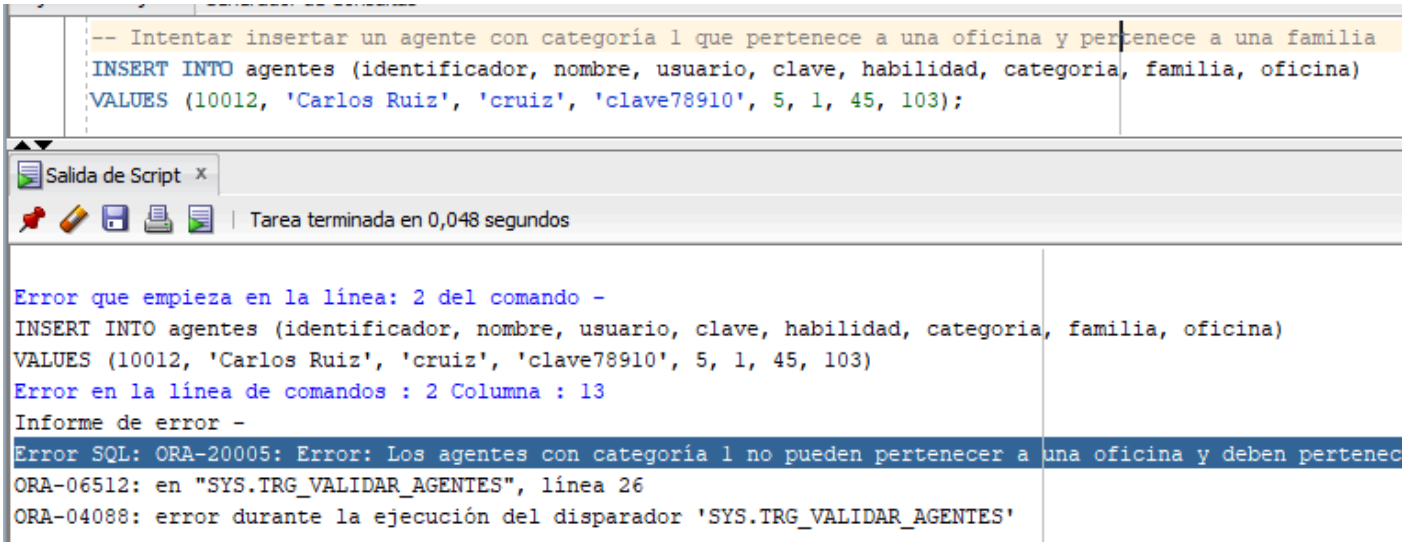
Salida de Script

Tarea terminada en 0,138 segundos

```
Error que empieza en la línea: 2 del comando -
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10011, 'Laura Gómez', 'lgomez', 'clave67890', 5, 1, NULL, 102)
Error en la línea de comandos : 2 Columna : 13
Informe de error -
Error SQL: ORA-20005: Error: Los agentes con categoría 1 no pueden pertenecer a una oficina y deben pertenecer a una familia.
ORA-06512: en "SYS.TRG_VALIDAR_AGENTES", línea 26
ORA-04088: error durante la ejecución del disparador 'SYS.TRG_VALIDAR_AGENTES'
```

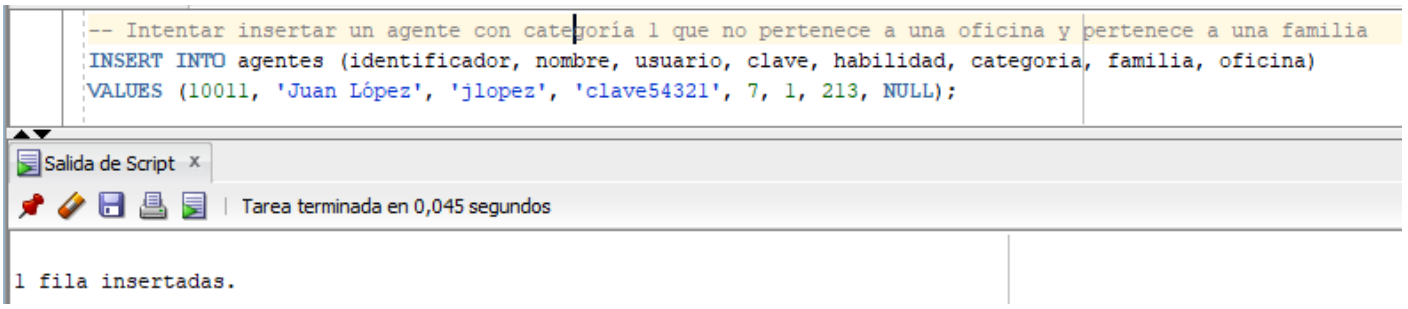
Prueba de la restricción 5 con un caso incorrecto:
Agente de categoría 1 que pertenece a una oficina y pertenece a una familia

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10012, 'Carlos Ruiz', 'cruiz', 'clave78910', 5, 1, 45, 103);



Prueba de la restricción 5 con un caso correcto:
Agente de categoría 1 no pertenece a una oficina y pertenece a una familia

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10011, 'Juan López', 'jlopez', 'clave54321', 7, 1, 213, NULL);



Restricción 6: Todos los agentes deben pertenecer a una oficina o a una familia pero nunca a ambas a la vez.

Código en PL/SQL para la Restricción 6

- Todos los agentes:
  - Pertenecen a una familia y
  - No pertenece a una oficina
- O todos los agentes:

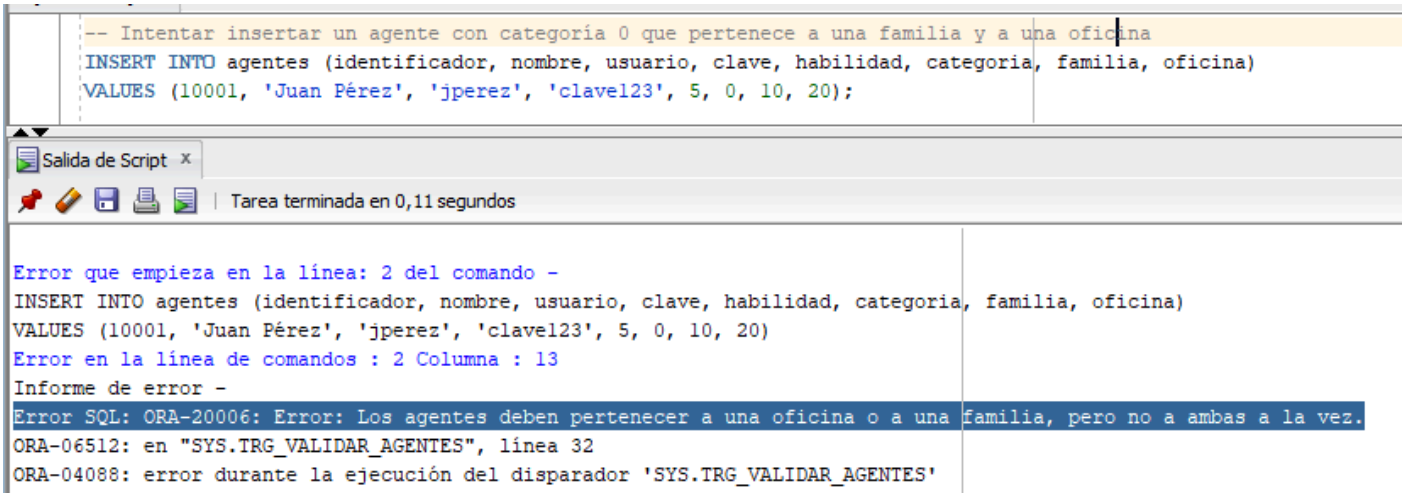
- No pertenecen a una familia y
- Sí pertenecen a una oficina

IF (:NEW.familia IS NULL AND :NEW.oficina IS NULL) OR (:NEW.familia IS NOT NULL AND :NEW.oficina IS NOT NULL) THEN
RAISE_APPLICATION_ERROR(-20006, 'Error: Los agentes deben pertenecer a una oficina o a una familia, pero no a ambas a la vez.');
END IF;

```
-- Restricción 6: Validar que todos los agentes pertenezcan a una oficina o una familia, pero no a ambas
IF (:NEW.familia IS NULL AND :NEW.oficina IS NULL) OR (:NEW.familia IS NOT NULL AND :NEW.oficina IS NOT NULL) THEN
    RAISE_APPLICATION_ERROR(-20006, 'Error: Los agentes deben pertenecer a una oficina o a una familia, pero no a ambas a la vez.');
END IF;
```

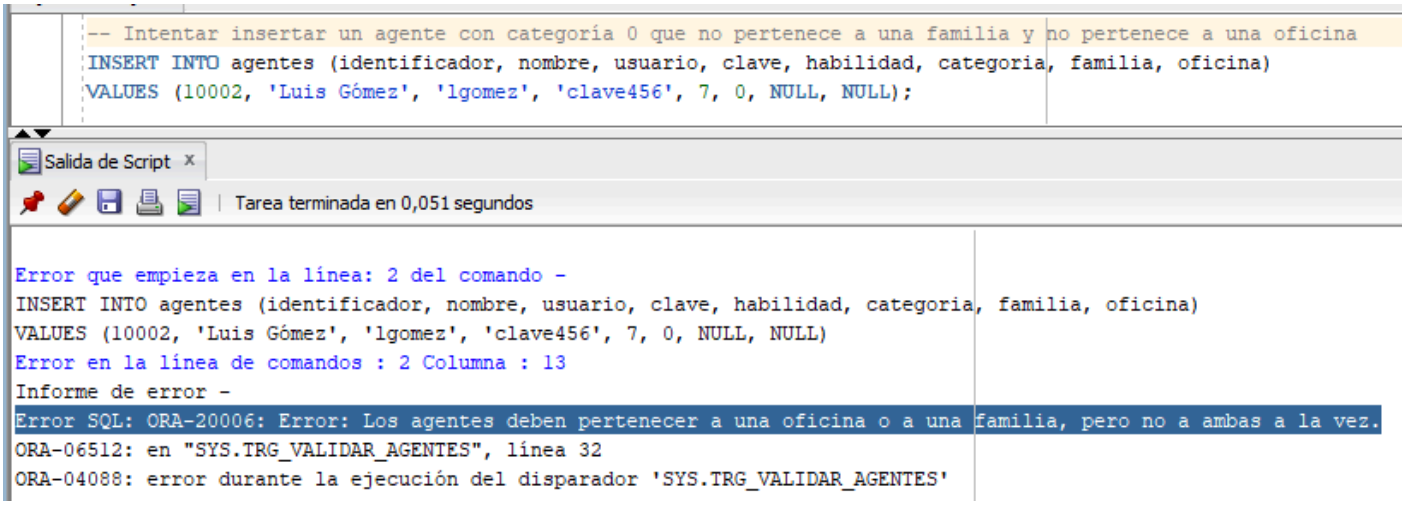
Prueba de la restricción 6 con un caso incorrecto: Agente con categoría 0 pertenece a una familia y pertenece a una oficina

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10001, 'Juan Pérez', 'jperez', 'clave123', 5, 0, 10, 20);



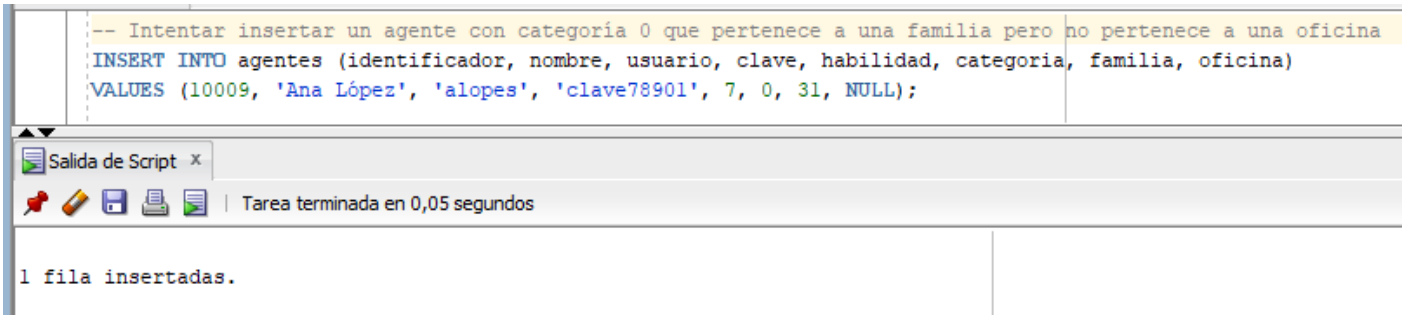
Prueba de la restricción 6 con un caso incorrecto: Agente de categoría 0 no pertenece a una familia y no pertenece a una oficina

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10002, 'Luis Gómez', 'lgomez', clave456', 7, 0, NULL, NULL);



Prueba de la restricción 6 con un caso correcto:
Agente de categoría 0 pertenece a una familia y no pertenece a una oficina

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10009, 'Ana López', 'alops', 'clave78901', 7, 0, 31, NULL);

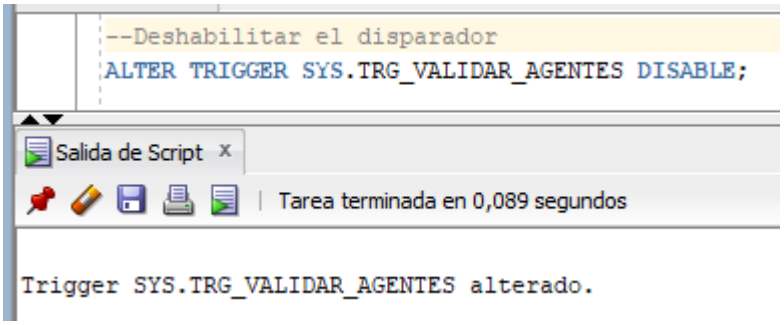


2.2. Restricciones de esquema utilizando DDL

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.

Deshabilitar el disparador para poder hacer las comprobaciones

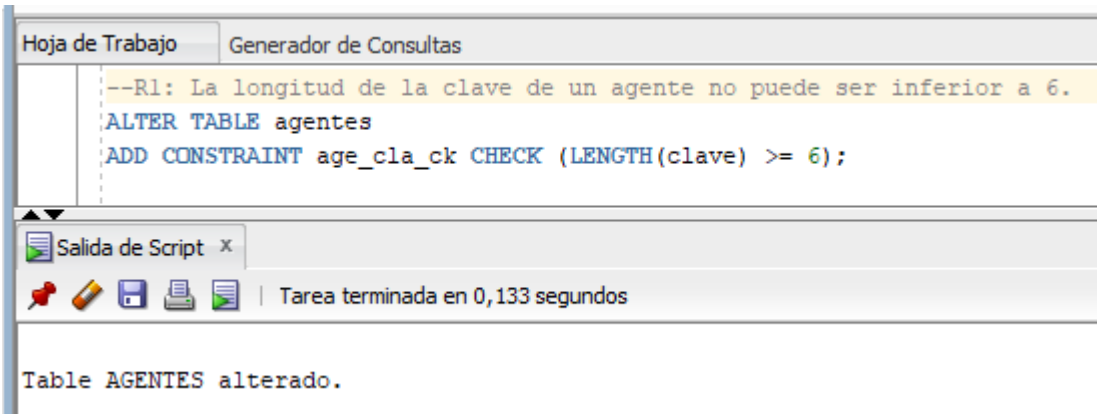
ALTER TRIGGER SYS.TRG_VALIDAR_AGENTES DISABLE;



Restricción 1: La longitud de la clave de un agente no puede ser inferior a 6.

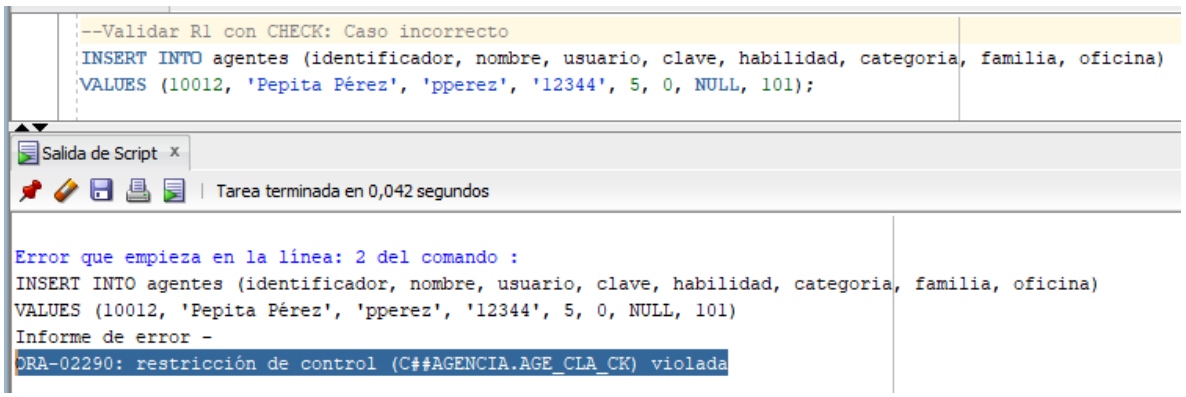
- La podemos implementar con un CHECK

ALTER TABLE agentes
ADD CONSTRAINT age_cla_ck CHECK (LENGTH(clave) >=6);



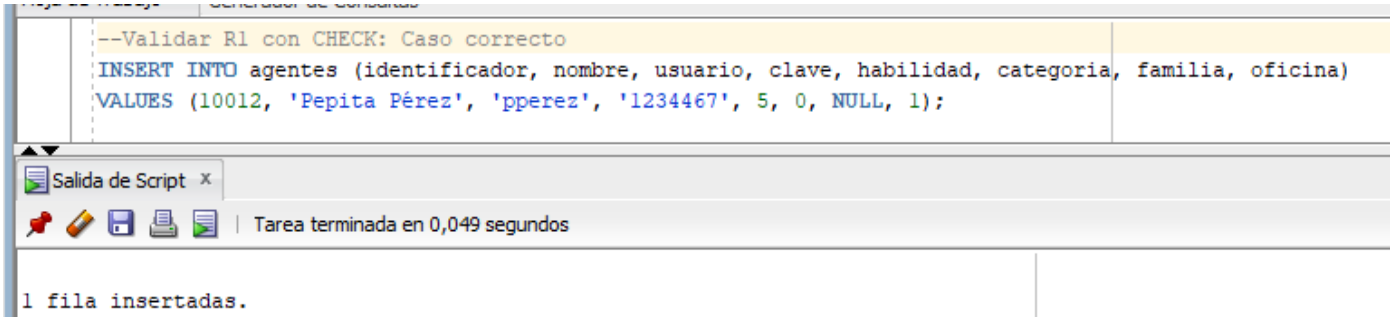
Prueba de la restricción 1 con un caso incorrecto: Una clave de longitud 5

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10012, 'Pepita Pérez', 'pperez', '12344', 5, 0, NULL, 101);



Prueba de la restricción 1 con un caso correcto: Una clave de longitud 7

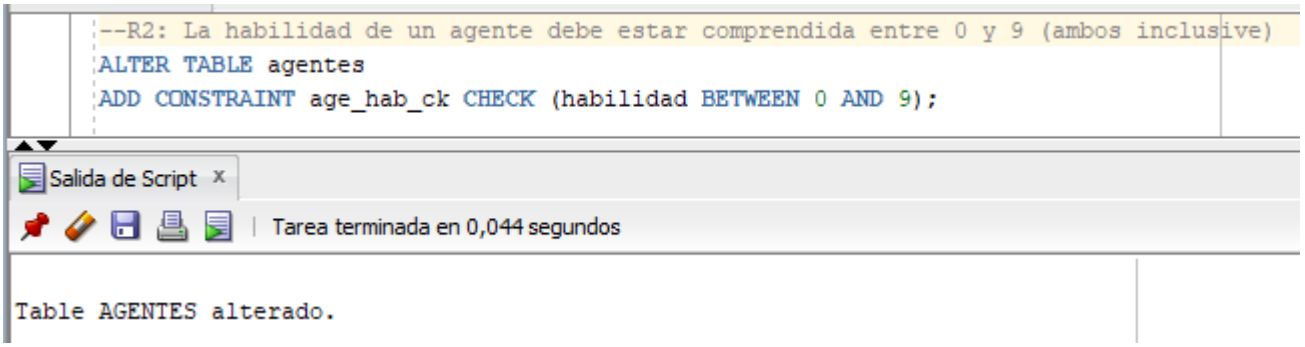
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10012, 'Pepita Pérez', 'pperez', '1234467', 5, 0, NULL, 1);



Restricción 2: La habilidad de un agente debe estar comprendida entre 0 y 9 (ambos inclusive)

- La podemos implementar con un CHECK

ALTER TABLE agentes
ADD CONSTRAINT age_hab_ck CHECK (habilidad BETWEEN 0 AND 9);



Prueba de la restricción 2 con un caso incorrecto: Un agente con habilidad 11

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10013, 'Emilia Pérez', 'eperez', '123456', 11, 0, NULL, NULL);

```
--Validar R2 con CHECK: Caso incorrecto: Agente con habilidad 11
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10013, 'Emilia Pérez', 'eperez', '123456', 11, 0, NULL, NULL);
```

```
Salida de Script
Tarea terminada en 0,123 segundos

Error que empieza en la línea: 2 del comando -
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10013, 'Emilia Pérez', 'eperez', '123456', 11, 0, NULL, NULL)
Error en la línea de comandos : 3 Columna : 53
Informe de error -
Error SQL: ORA-01438: valor mayor que el que permite la precisión especificada para esta columna
```

Prueba de la restricción 2 con un caso correcto: Un agente con habilidad 5

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10013, 'Emilia Pérez', 'eperez', '123456', 5, 0, 112, NULL);

```
--Validar R2 con CHECK: Caso correcto: Agente con habilidad 5
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10013, 'Emilia Pérez', 'eperez', '123456', 5, 0, 112, NULL);
```

```
Salida de Script
Tarea terminada en 0,049 segundos

1 fila insertadas.
```

Restricción 3: La categoría de un agente sólo puede ser 0, 1 o 2

- La podemos implementar con un CHECK

ALTER TABLE agentes
ADD CONSTRAINT age_cat_ck CHECK (categoria IN (0, 1, 2));

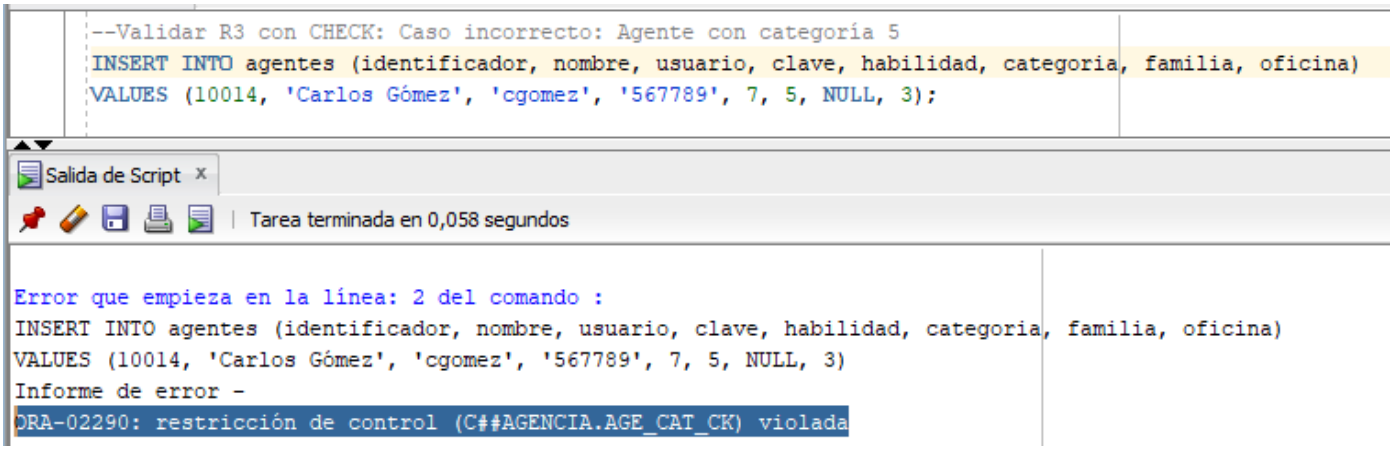
```
--R3: La categoría de un agente sólo puede ser 0, 1 o 2
ALTER TABLE agentes
ADD CONSTRAINT age_cat_ck CHECK (categoria IN (0, 1, 2));
```

```
Salida de Script
Tarea terminada en 0,066 segundos

Table AGENTES alterado.
```

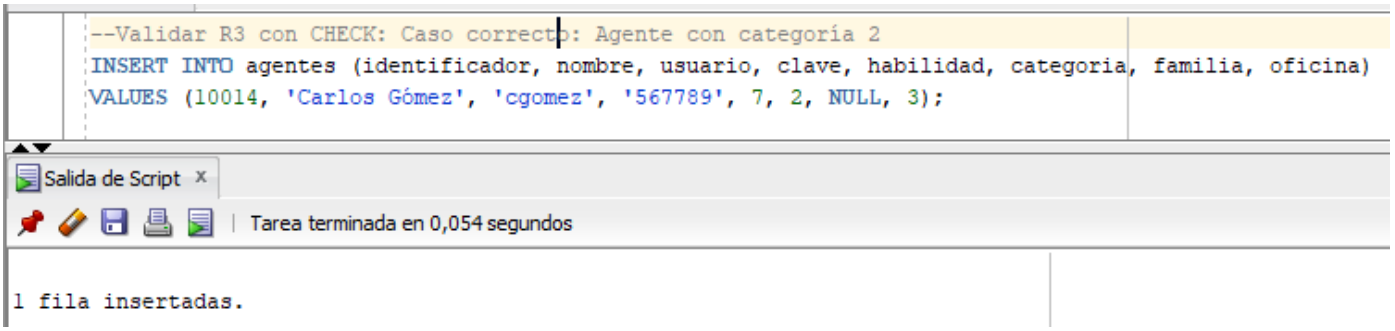
Prueba de la restricción 3 con un caso incorrecto: Un agente con categoría 5

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10014, 'Carlos Gómez', 'cgomez', '567789', 7, 5, NULL, 3);



Prueba de la restricción 3 con un caso correcto: Un agente con categoría 2

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10014, 'Carlos Gómez', 'cgomez', '567789', 7, 2, NULL, 3);



Restricción 4: Si un agente tiene categoría 2 no puede pertenecer a ninguna familia y debe pertenecer a una oficina.

La podemos implementar con CHECK:

- **Primera restricción (age_cat2_fam_ck):**
  - Si la categoría del agente no es 2, entonces no se aplica ninguna restricción sobre la familia.
  - Si la categoría es 2, entonces la familia debe ser NULL, es decir, el agente no puede pertenecer a una familia.
- **Segunda restricción (age_cat2_ofi_ck):**
  - Si la categoría del agente es 2, entonces la oficina debe ser no nula (el agente debe

pertenecer a una oficina).

- No es necesario añadir condiciones adicionales para otras categorías, ya que la restricción solo se aplica a los agentes con categoría 2.

```
--R4: Si un agente tiene categoría 2 no puede pertenecer a ninguna familia y debe pertenecer a una oficina.
ALTER TABLE agentes
ADD CONSTRAINT age_cat2Fam_ck CHECK (categoria <> 2 OR familia IS NULL);

ALTER TABLE agentes
ADD CONSTRAINT age_cat2Fam_ck CHECK (categoria = 2 AND oficina IS NOT NULL);
```

```
Salida de Script
Tarea terminada en 0,17 segundos

Table AGENTES alterado.
```

Prueba de la restricción 4 con un caso incorrecto: Un agente con categoría 2 que pertenece a una familia y no pertenece a ninguna oficina.

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10015, 'Laura Rodríguez', 'lrodriguez', '12345678', 6, 2, 21, NULL);

```
--Validar R4 con CHECK: Caso correcto: Agente con categoría 2 pertenece a familia y no pertenece a oficina
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10015, 'Laura Rodríguez', 'lrodriguez', '12345678', 6, 2, 21, NULL);
```

```
Salida de Script
Tarea terminada en 0,076 segundos

Error que empieza en la línea: 2 del comando :
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10015, 'Laura Rodríguez', 'lrodriguez', '12345678', 6, 2, 21, NULL)
Informe de error -
ORA-02290: restricción de control (C##AGENCIA.AGE_CAT2FAM_CK) violada
```

Prueba de la restricción 4 con un caso correcto: Un agente con categoría 2 que no pertenece a ninguna familia y pertenece a una oficina.

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10015, 'Laura Rodríguez', 'lrodriguez', '12345678', 6, 2, NULL, 3);

```
--Validar R4 con CHECK: Caso correcto: Agente con categoría 2 NO pertenece a familia y SÍ pertenece a oficina
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10015, 'Laura Rodríguez', 'lrodriguez', '12345678', 6, 2, NULL, 3);
```

Salida de Script

Tarea terminada en 0,052 segundos

```
1 fila insertadas.
```

Restricción 5: Si un agente tiene categoría 1 no puede pertenecer a ninguna oficina y debe pertenecer a una familia

La implementamos con CHECK:

- **Primera restricción (age_catOfi_ck):** Asegura que los agentes con categoría 1 no estén asignados a ninguna oficina.
  - Si la categoría no es 1 (categoria <> 1), la restricción no se aplica (porque no hay ninguna restricción para los agentes con otras categorías).
  - Si la categoría es 1 (categoria = 1), entonces la oficina debe ser NULL, es decir, el agente no debe estar asignado a ninguna oficina.
- **Segunda restricción (age_catFam_ck):** Asegura que los agentes con categoría 1 estén asignados a una familia.
  - Si la categoría es 1 (categoria = 1), entonces la familia debe ser no nula (familia IS NOT NULL), es decir, el agente debe pertenecer a una familia.
  - No hay necesidad de comprobar los agentes con categorías diferentes, porque el comportamiento solo se aplica cuando la categoría es 1.

```
--R5: Si un agente tiene categoría 1 no puede pertenecer a ninguna oficina y debe pertenecer a una familia
ALTER TABLE agentes
ADD CONSTRAINT age_catOfi_ck CHECK (categoria <> 1 OR oficina IS NULL);

ALTER TABLE agentes
ADD CONSTRAINT age_catFam_ck CHECK (categoria = 1 AND familia IS NOT NULL);
```

Salida de Script

Tarea terminada en 0,05 segundos

```
Table AGENTES alterado.
```

Prueba de la restricción 5 con un caso incorrecto: Un agente con categoría 1 que pertenece a una oficina y no pertenece a una familia

```
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10017, 'Marta Gómez', 'mgomez', '123345', 8, 1, NULL, 3);
```

```
--Validar R5 con CHECK: Caso incorrecto: Un agente con categoría 1 SÍ oficina y NO familia
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10017, 'Marta Gómez', 'mgomez', '123345', 8, 1, NULL, 3);
```

Salida de Script

Tarea terminada en 0,069 segundos

```
Error que empieza en la línea: 2 del comando :
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10017, 'Marta Gómez', 'mgomez', '123345', 8, 1, NULL, 3)
Informe de error -
ORA-02290: restricción de control (C##AGENCIA.AGE_CATOFI_CK) violada
```

Prueba de la restricción 5 con un caso correcto: Un agente con categoría 1 que no pertenece a ninguna oficina y pertenece a una familia

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10017, 'Marta Gómez', 'mgomez', '123345', 8, 1, 11, NULL);

```
--Validar R5 con CHECK: Caso correcto: Un agente con categoría 1 NO oficina y SÍ familia
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10017, 'Marta Gómez', 'mgomez', '123345', 8, 1, 11, NULL);
```

Salida de Script

Tarea terminada en 0,053 segundos

```
1 fila insertadas.
```

Restricción 6: Todos los agentes deben pertenecer a una oficina o a una familia pero nunca a ambas a la vez.

- La podemos implementar con un CHECK

```
ALTER TABLE agentes
ADD CONSTRAINT age_ofi_o_fam_ck CHECK (
    (familia IS NOT NULL AND oficina IS NULL) OR (familia IS NULL AND oficina IS NOT NULL));
```

```
--R6: Todos los agentes deben pertenecer  a una oficina o a una familia pero nunca a ambas a la vez.
ALTER TABLE agentes
ADD CONSTRAINT age_ofi_o_fam_ck CHECK (
    (familia IS NOT NULL AND oficina IS NULL) OR (familia IS NULL AND oficina IS NOT NULL)
);
```

Salida de Script

Tarea terminada en 0,048 segundos

```
Table AGENTES alterado.
```

Prueba de la restricción 6 con un caso incorrecto: Un agente con categoría 0 que pertenece a una familia y a una oficina

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10019, 'Ana Gómez', 'agomez', '103456', 6, 0, 21, 2);

```
--Validar R6 con CHECK: Caso incorrecto: Un agente con categoría 0 SÍ oficina y SÍ familia
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10019, 'Ana Gómez', 'agomez', '103456', 6, 0, 21, 2);
```

Salida de Script

Tarea terminada en 0,077 segundos

```
Error que empieza en la línea: 2 del comando :
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10019, 'Ana Gómez', 'agomez', '103456', 6, 0, 21, 2)
Informe de error -
ORA-02290: restricción de control (C##AGENCIA.AGE_OFI_O_FAM_CK) violada
```

Prueba de la restricción 6 con un caso correcto: Un agente con categoría 0 que pertenece a una oficina y no pertenece a una família

INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10019, 'Ana Gómez', 'agomez', '103456', 6, 0, NULL, 2);

```
--Validar R6 con CHECK: Caso correcto: Un agente con categoría 0 SÍ oficina y NO familia
INSERT INTO agentes (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina)
VALUES (10019, 'Ana Gómez', 'agomez', '103456', 6, 0, NULL, 2);
```

Salida de Script

Tarea terminada en 0,072 segundos

```
1 fila insertadas.
```

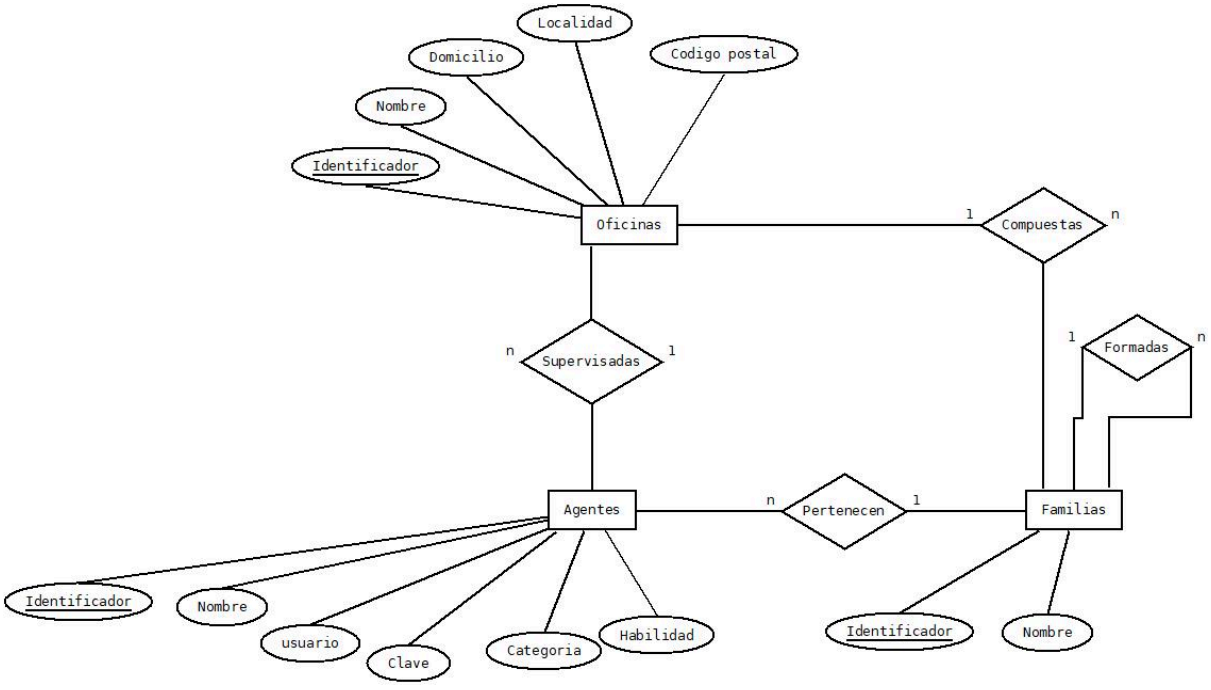
Anexo

Caso de estudio

Una empresa de telefonía tiene sus centros de llamadas distribuidos por la geografía española en diferentes oficinas. Estas oficinas están jerarquizadas en familias de agentes telefónicos. Cada familia, por tanto, podrá contener agentes u otras familias. Los agentes telefónicos, según su categoría, además se encargarán de supervisar el trabajo de todos los agentes de una oficina o de coordinar el trabajo de los agentes de una familia dada. El único agente que pertenecerá directamente a una oficina y que no formará parte de ninguna familia será el supervisor de dicha oficina, cuya categoría es la 2. Los coordinadores de las familias deben pertenecer a dicha familia y su categoría será 1 (no todas las familias tienen por qué tener un coordinador y dependerá del tamaño de la oficina, ya que de ese trabajo también se puede encargar el supervisor de la oficina). Los demás agentes deberán pertenecer a una familia, su categoría será 0 y serán los que principalmente se ocupen de atender las llamadas.

- De los agentes queremos conocer su nombre, su clave y contraseña para entrar al sistema, su categoría y su habilidad que será un número entre 0 y 9 indicando su habilidad para atender llamadas.
- Para las familias sólo nos interesa conocer su nombre.
- Finalmente, para las oficinas queremos saber su nombre, domicilio, localidad y código postal de la misma.

Un posible modelo entidad-relación para el problema expuesto podría ser el siguiente:



El Modelo Relacional resultante sería:

- OFICINAS (identificador, nombre, domicilio, localidad, codigo_postal)
- FAMILIAS (identificador, nombre, familia (fk), oficina (fk))
- AGENTES (identificador, nombre, usuario, clave, habilidad, categoría, familia (fk), oficina (fk))

De este modelo de datos surgen tres tablas.

Script

```
drop user c##agencia cascade;
create user c##agencia identified by agencia default tablespace users;
grant connect,resource,dba to c##agencia;
conn c##agencia/agencia
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
);
```

```
insert into oficinas values (1, 'Madrid', 'Gran vía, 37', 'Madrid', '28000');
insert into oficinas values (2, 'Granada', 'Camino Ronda, 50', 'Granada',
'36000');
insert into oficinas values (3, 'Jaén', 'Gran Eje, 80', 'Jaén', '27000');

insert into familias values (11, 'Madrid-1', NULL, 1);
insert into familias values (111, 'Madrid-1.1', 11, NULL);
insert into familias values (112, 'Madrid-1.2', 11, NULL);
insert into familias values (1121, 'Madrid-1.2.1', 112, NULL);
insert into familias values (1122, 'Madrid-1.2.2', 112, NULL);
insert into familias values (1123, 'Madrid-1.2.3', 112, NULL);
insert into familias values (21, 'Granada-1', NULL, 2);
insert into familias values (211, 'Granada-1.1', 21, NULL);
insert into familias values (212, 'Granada-1.2', 21, NULL);
insert into familias values (213, 'Granada-1.3', 21, NULL);
insert into familias values (31, 'Jaén-1', NULL, 3);

insert into agentes values (31, 'José Ramón Jiménez Reyes', 'jrjr', 'sup31',
9, 2, NULL, 3);
insert into agentes values (311, 'Pedro Fernández Arias', 'pfa', 'ag311', 5,
0, 31, NULL);
insert into agentes values (312, 'Vanesa Sánchez Rojo', 'vsr', 'ag312', 5, 0,
31, NULL);
insert into agentes values (313, 'Francisco Javier García Escobedo', 'fjge',
'ag313', 5, 0, 31, NULL);
insert into agentes values (314, 'Pilar Ramirez Pérez', 'prp', 'ag314', 5, 0,
31, NULL);
insert into agentes values (315, 'José Luis García Martínez', 'jlgm', 'ag315',
5, 0, 31, NULL);
insert into agentes values (21, 'Sebastián López Ojeda', 'slo', 'sup21', 9, 2,
NULL, 2);
insert into agentes values (211, 'Diosdado Sánchez Hernández', 'dsh', 'ag211',
8, 1, 21, NULL);
insert into agentes values (2111, 'José Juan Cano Pardo', 'jjcp', 'ag2111', 5,
0, 211, NULL);
insert into agentes values (2112, 'Flor Moncada Añón', 'ag2112', 'fma', 5, 0,
211, NULL);
insert into agentes values (2113, 'Juan Manuel Alcazar Donaire', 'jmad',
'ag2113', 5, 0, 211, NULL);
insert into agentes values (2121, 'Manuel Jesús Rubia Mateos', 'mjrm',
'ag2121', 5, 0, 212, NULL);
```

```
insert into agentes values (2122, 'Esther López Delgado', 'eld', 'ag2122', 5,
0, 212, NULL);
insert into agentes values (2123, 'Francisco Javier Cabrerizo Membrilla',
'fjcm', 'ag2123', 5, 0, 212, NULL);
insert into agentes values (2124, 'Verónica Cabrerizo Menbrilla', 'vcm',
'ag2124', 5, 0, 212, NULL);
insert into agentes values (2125, 'María José Navascués Morales', 'mjnm',
'ag2125', 5, 0, 212, NULL);
insert into agentes values (2131, 'Isabel Cruz Granados', 'icg', 'ag2131', 5,
0, 213, NULL);
insert into agentes values (2132, 'Antonio Casado Fernández', 'acf', 'ag2132',
5, 0, 213, NULL);
insert into agentes values (2133, 'Gabriel Callejón García', 'gcg', 'ag2133',
5, 0, 213, NULL);
insert into agentes values (2134, 'Enrique Cano Balsera', 'ecb', 'ag2134', 5,
0, 213, NULL);
insert into agentes values (11, 'Narciso Jáimez Toro', 'njt', 'sup11', 9, 2,
NULL, 1);
insert into agentes values (111, 'Jesús Baños Sancho', 'jbs', 'ag111', 8, 1,
11, NULL);
insert into agentes values (1111, 'Salvador Romero Villegas', 'srv', 'ag1111',
7, 1, 111, NULL);
insert into agentes values (1112, 'José Javier Bermúdez Hernández', 'jjbh',
'ag1112', 7, 1, 111, NULL);
insert into agentes values (1113, 'Alfonso Bonillo Sierra', 'abs', 'ag1113',
7, 1, 111, NULL);
insert into agentes values (1121, 'Silvia Thomas Barrós', 'stb', 'ag1121', 7,
1, 112, NULL);
insert into agentes values (11211, 'Ernesto Osoro Gorrotxategi', 'eog',
'ag11211', 5, 0, 1121, NULL);
insert into agentes values (11212, 'Guillermo Campos Guillén', 'gcag',
'ag11212', 5, 0, 1121, NULL);
insert into agentes values (11213, 'Antonio Fernández Ruíz', 'afr', 'ag11213',
5, 0, 1121, NULL);
insert into agentes values (11214, 'María Luisa López Caballero', 'mllc',
'ag11214', 5, 0, 1121, NULL);
insert into agentes values (11221, 'Virginia Morenas Rubio', 'vmr', 'ag11221',
5, 0, 1121, NULL);
insert into agentes values (11222, 'Rosario Castro García', 'rcg', 'ag11222',
5, 0, 1122, NULL);
insert into agentes values (11223, 'Antonio Álvarez Palomeque', 'aap',
'ag11223', 5, 0, 1122, NULL);
```

```
insert into agentes values (11224, 'David Martínez Martínez', 'dmm',
'ag11224', 5, 0, 1122, NULL);
insert into agentes values (11225, 'Juan Corral González', 'jcg', 'ag11225',
5, 0, 1122, NULL);
insert into agentes values (11226, 'Eduardo Alfada Pereira', 'eap', 'ag11226',
5, 0, 1122, NULL);
insert into agentes values (11231, 'Cayetano García Herrera', 'cgh',
'ag11231', 5, 0, 1123, NULL);
insert into agentes values (11232, 'José Antonio Sieres Vega', 'jasv',
'ag11232', 5, 0, 1123, NULL);
insert into agentes values (11233, 'Juan Manuel Guzmán García', 'jmgg',
'ag11233', 5, 0, 1123, NULL);

commit;
```

Disparador

```
CREATE OR REPLACE TRIGGER trg_validar_agentes
BEFORE INSERT OR UPDATE ON agentes
FOR EACH ROW
BEGIN
    -- Restricción 1: Validar longitud mínima de la clave
    IF LENGTH(:NEW.clave) < 6 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Error: La clave debe tener al menos 6
caracteres.');
    END IF;

    -- Restricción 2: Validar que la habilidad esté entre 0 y 9
    IF :NEW.habilidad NOT BETWEEN 0 AND 9 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Error: La habilidad debe estar entre 0 y 9.');
    END IF;

    -- Restricción 3: Validar que la categoría solo pueda ser 0, 1 o 2
    IF :NEW.categoria NOT IN (0, 1, 2) THEN
        RAISE_APPLICATION_ERROR(-20003, 'Error: La categoría solo puede ser 0, 1 o 2.');
    END IF;

    -- Restricción 4: Validar que si la categoría es 2, no pertenezca a ninguna familia
y sí a una oficina
    IF :NEW.categoria = 2 AND (:NEW.familia IS NOT NULL OR :NEW.oficina IS NULL) THEN
    RAISE_APPLICATION_ERROR(-20004, 'Error: Los agentes con categoría 2 no pueden
pertenecer a una familia y deben pertenecer a una oficina.');
    END IF;
```

```
    -- Restricción 5: Validar que si la categoría es 1, no pertenezca a ninguna oficina
y sí a una familia
    IF :NEW.categoria = 1 THEN
    -- Validar que no pertenezca a una oficina y sí pertenezca a una familia
        IF :NEW.oficina IS NOT NULL OR :NEW.familia IS NULL THEN
        RAISE_APPLICATION_ERROR(-20005, 'Error: Los agentes con categoría 1 no pueden
pertenecer a una oficina y deben pertenecer a una familia.');
        END IF;
    END IF;

    -- Restricción 6: Validar que todos los agentes pertenezcan a una oficina o una
familia, pero no a ambas
    IF (:NEW.familia IS NULL AND :NEW.oficina IS NULL) OR (:NEW.familia IS NOT NULL AND
:NEW.oficina IS NOT NULL) THEN
        RAISE_APPLICATION_ERROR(-20006, 'Error: Los agentes deben pertenecer a una
oficina o a una familia, pero no a ambas a la vez.');
    END IF;
END;
```