Universidad ORT Uruguay

Facultad de Ingeniería

Martin Alonso y Juan Carriquiry

Bases de datos 2

Segundo obligatorio doc parte 1

Grupo M5A
Docente: Cecilia Belleti
Docente: Tatiana Bellon

Formulario de Antecedentes
Curso Materia

Nro. Estudiante: **291799**
Nombres: **Martin**
Apellidos: **Alonso**
Grupo/Turno: **5A/Matutino**

Nro. Estudiante: **310190**
Nombres: **Juan**
Apellidos: **Carriquiry**
Grupo/Turno: **5A/Matutino**

1. Tabla de contenido

1. Tabla de contenido 3
1) Datos de prueba generales 4
a) Nota 1: 4
b) Nota 2: 4
c) Nota 3: 4
d) Nota 4: 4
e) Nota 5: 5
f) Uso: 5
2) Triggers a implementar: 5
3) Cambio necesario en el DDL: 7
a) Explicación primer cambio: 7
b) Implementación del primer cambio: 7
c) Explicación del segundo cambio: 8
d) Implementación del segundo cambio: 8
4) Implementación del primer trigger: 9
a) Código 9
b) Mensaje de ejecución: 10
5) Implementación del segundo trigger: 10
a) Código: 10
b) Mensaje de ejecución: 10
6) Pruebas: 11
a) Verificación de implementación: 11
b) Ubicación del script de pruebas: 11
c) Explicación breve del código: 11
d) Resultado esperado del trigger para validar cambio en característica: 11
e) Resultado esperado del trigger para dar inicio a las misiones: 12
f) Resultado real del trigger para validar cambio en característica: 12
g) Resultado real del trigger para validar cambio en característica: 13

1) Datos de prueba generales

a) Nota 1:

Estos son los datos de prueba más generales, cada parte puede contener algún dato de prueba adicional

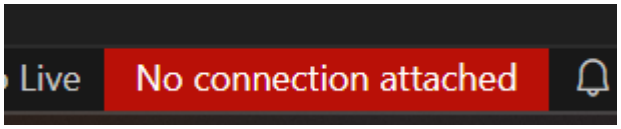
b) Nota 2:

Todo se mostrará usando la extensión de VSCode de SqlDeveloper.

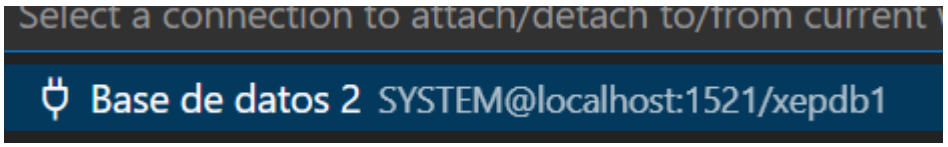
c) Nota 3:

Asumimos que antes de cada ejecución de código el usuario se conecta a la bd en cada archivo abierto:

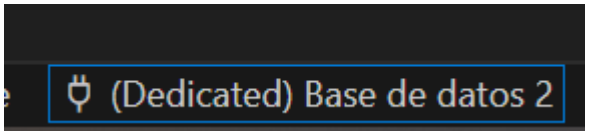
Click abajo a la derecha:



Seleccionamos la BD en el cuadro de interacción en el centro arriba:

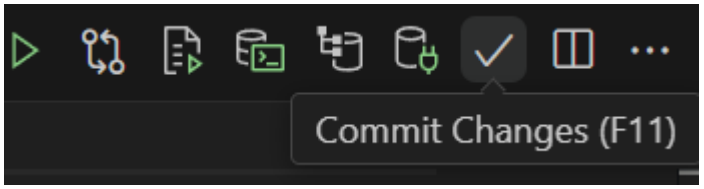


Resultado abajo a la derecha:



d) Nota 4:

Asumimos que luego de cualquier modificación a la base de datos el usuario commit los cambios arriba a la derecha:



e) Nota 5:

Es importante resaltar que hemos utilizado inteligencia artificial generativa como ChatGPT o Gemini para la generación de datos de prueba y darles un formato mas atractivo y entendible.

f) Uso:

Primero ejecutamos el script presente en SCRIPTS_NUEVOS/Parte1/DDL.sql para crear las tablas base y todo lo necesario:

Ya que el mensaje de confirmación es muy largo subimos la última parte (puede ser que los drops al inicio de errores si las tablas no estaban previamente creadas, simplemente ignorarlos)

```
Table MISION_ES_PREVIA_DE_HABILIDAD created.

Table PERSONAJE_MISION created.

Table LOG_AUMENTO_NIVEL created.

Table LOG_PREMIO_DIARIO created.
```

2) Triggers a implementar:

Para esta explicación extraemos los triggers de la Documentación de la Parte 1 del primer obligatorio, extrayendo los que tienen de tipo de restricción como “GLOBAL” y de tipo de implementación “NO ESTRUCTURAL”

Restricción	Relación	tipo de restricción	tipo de implementación	comentario
Equipado	Posee	Global	No estructural	Si alguna de las características afectadas por el item deja al usuario con una característica con un valor mayor a 100 o menor a 0 deja el atributo en el valor correcto

				más cercano.
Estado	Misión	Global	No estructural	Para iniciar la misión primero se tiene que chequear que todas sus misiones previas estén en el estado de “Completada”

3) Cambio necesario en el DDL:

a) Explicación primer cambio:

Para la implementación del segundo trigger necesitamos llevar un registro del progreso de cada personaje (misiones completadas) la cual usaremos para ver si un personaje puede cambiar el estado de una misión (completarla), basandonos que el personaje ya debería de tener registros en esa tabla con todas las previas de la misión que quiere completar.

Importante, los registros de las misiones previas tienen que tener estado “Completada”

b) Implementación del primer cambio:

Agregamos el drop:
DROP TABLE Personaje_Mision CASCADE CONSTRAINTS;

Agregamos la tabla:

```
CREATE TABLE Personaje_Mision ( -- se genera esta tabla
    emailJugador        VARCHAR2(40)    NOT NULL,
    idMision            NUMBER(5)       NOT NULL,
    estado_mision_pers  VARCHAR2(15)    NOT NULL
        CHECK (estado_mision_pers IN ('En progreso', 'Completada')),
    recompensas_recibidas CHAR(1)       DEFAULT 'N' NOT NULL
        CHECK (recompensas_recibidas IN ('S', 'N')),
    fecha_estado DATE DEFAULT SYSDATE NOT NULL,
    CONSTRAINT pk_Personaje_Mision
        PRIMARY KEY (emailJugador, idMision),
    CONSTRAINT fk_Personaje_Mision_Pers
        FOREIGN KEY (emailJugador) REFERENCES Personaje(email_Jugador),
    CONSTRAINT fk_Personaje_Mision_Mision
        FOREIGN KEY (idMision) REFERENCES Misiones(id)
);
```

c) Explicación del segundo cambio:

Para evitar redundancias y para el correcto uso de los atributos y gestión general de las misiones se elimina de la tabla “Misiones” el atributo “Estado” ya que esto hacía que la misión tuviera un estado general y no para cada personaje.

También se puede ver que se agrega otro atributo pero eso es de otra parte del obligatorio.

d) Implementación del segundo cambio:

```
CREATE TABLE Misiones (
    id          NUMBER(5)  NOT NULL,
    nombre      VARCHAR2(20) NOT NULL,
    descripcion VARCHAR2(50) NOT NULL,
    nivelMin    NUMBER(3)    NOT NULL CHECK (nivelMin BETWEEN 0 AND 342),
    estado      VARCHAR2(10) NOT NULL CHECK (estado IN ('Principal','Secundaria','Especial')),
    CONSTRAINT pk_Misiones PRIMARY KEY (id)
);
```

4) Implementación del primer trigger:

a) Código

```
CREATE OR REPLACE TRIGGER TRG_AJUSTAR_CARACTERISTICAS
BEFORE UPDATE ON Personaje
FOR EACH ROW
BEGIN
    -- Validamos y ajustamos la FUERZA
    IF :new.fuerza > 100 THEN
        :new.fuerza := 100;
    ELSIF :new.fuerza < 0 THEN
        :new.fuerza := 0;
    END IF;

    -- Validamos y ajustamos la AGILIDAD
    IF :new.agilidad > 100 THEN
        :new.agilidad := 100;
    ELSIF :new.agilidad < 0 THEN
        :new.agilidad := 0;
    END IF;

    -- Validamos y ajustamos la INTELIGENCIA
    IF :new.inteligencia > 100 THEN
        :new.inteligencia := 100;
    ELSIF :new.inteligencia < 0 THEN
        :new.inteligencia := 0;
    END IF;

    -- Validamos y ajustamos la VITALIDAD
    IF :new.vitalidad > 100 THEN
        :new.vitalidad := 100;
    ELSIF :new.vitalidad < 0 THEN
        :new.vitalidad := 0;
    END IF;

    -- Validamos y ajustamos la RESISTENCIA
    IF :new.resistencia > 100 THEN
        :new.resistencia := 100;
    ELSIF :new.resistencia < 0 THEN
        :new.resistencia := 0;
    END IF;
END;
```

b) Mensaje de ejecución:

El mensaje esperado por consola al implementar este trigger es el siguiente:

```
Trigger TRG_AJUSTAR_CARACTERISTICAS compiled
```

5) Implementación del segundo trigger:

a) Código:

```
CREATE OR REPLACE TRIGGER TRG_VALIDAR_INICIO_MISION
BEFORE INSERT OR UPDATE ON Personaje_Mision
FOR EACH ROW
DECLARE
    v_conteo_previas_no_completadas NUMBER;
BEGIN
    -- solo si inserto o actualizo algo y el nuevo estado es En progreso
    IF (INSERTING OR UPDATING) AND :new.estado_mision_pers = 'En progreso' THEN

       --cuento las previas que no cumplen
        SELECT COUNT(*)
        INTO v_conteo_previas_no_completadas
        FROM Mision_Es_Previa_De_Mision pre
        WHERE
            pre.codMision2 = :new.idMision
            AND NOT EXISTS (
                SELECT 1
                FROM Personaje_Mision pm
                WHERE pm.idMision = pre.codMision1
                  AND pm.emailJugador = :new.emailJugador
                  AND pm.estado_mision_pers = 'Completada'
            );

        IF v_conteo_previas_no_completadas > 0 THEN
            RAISE_APPLICATION_ERROR(-20001, 'No se puede iniciar la misión. Existen misiones previas no completadas.');
        END IF;
    END IF;
END;
```

b) Mensaje de ejecución:

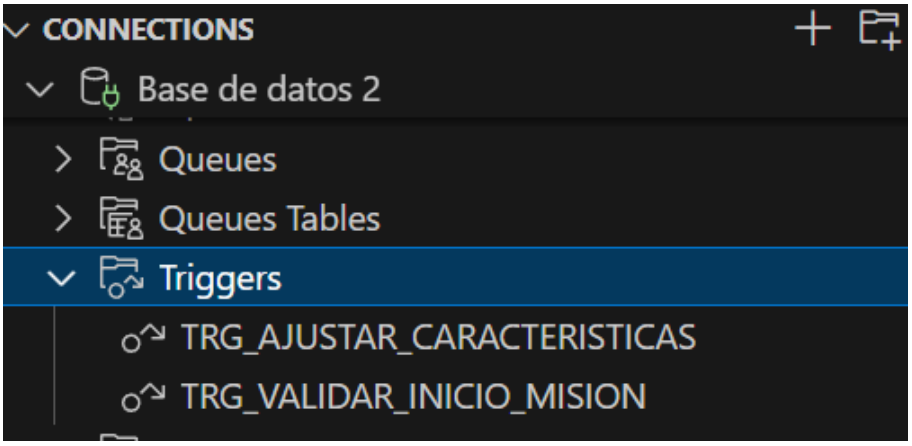
El mensaje esperado por consola al implementar este trigger es el siguiente:

```
Trigger TRG_VALIDAR_INICIO_MISION compiled
```

6) Pruebas:

a) Verificación de implementación:

Primero que nada podemos verificar que la BD reconoció a los triggers si vamos a la extensión en vs Code y de ahí abrimos la coneccion y bajamos hasta la pestaña de triggers donde deberíamos ver algo similar a esto:



b) Ubicación del script de pruebas:

El script de pruebas se encuentra disponible en SCRIPTS_NUEVOS\Parte1\DatosDePrueba.sql

c) Explicación breve del código:

En este archivo podemos encontrar que primero creamos un spool para poder ver todos los mensajes de los triggers ya sì validar sus salidas, luego encontraremos que se borran todos los datos de la BD en las tablas que vamos a probar y se hace el commit

Por último se insertan todos los datos necesarios para las pruebas

d) Resultado esperado del trigger para validar cambio en característica:

1) Actualización normal (exito)
2) Intento de superar el límite superior de una característica (la corrige)
3) Intento de descender del límite inferior en una característica (la corrige)
4) Intento de llegar al límite sin pasarle (éxito)

e) Resultado esperado del trigger para dar inicio a las misiones:

1) Intento de dar inicio a misión sin previas (exito)
2) Intento de dar inicio con prerrequisito no completado (error)
3) Intento completar un prerrequisito y luego completar la mision(exito)

f) Resultado real del trigger para validar cambio en característica:

1) Resultado 1:

```
-- Caso 1 (Feliz): Actualización normal de 50 a 75.


PL/SQL procedure successfully completed.


1 row updated.
```

2) Resultado 2:

```
-- Caso 2 (Triste): Intento de poner agilidad a 150. Debería quedar en 100.


PL/SQL procedure successfully completed.


1 row updated.
```

3) Resultado 3:

```
-- Caso 3 (Triste): Intento de poner inteligencia a -50. Debería quedar en 0.


PL/SQL procedure successfully completed.


1 row updated.
```

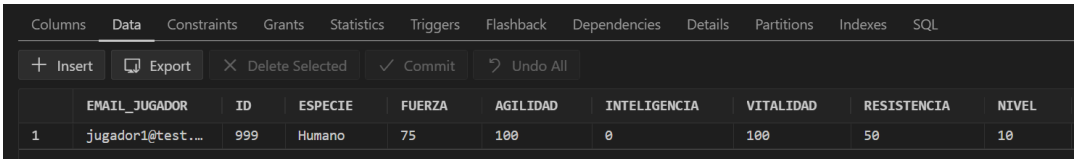
4) Resultado 4:

```
-- Caso 4 (Feliz): Poner vitalidad en 100. Debería aceptarlo.


PL/SQL procedure successfully completed.


1 row updated.
```

5) Visualizando resultados:
Yendo a los datos de la tabla en la extensión se puede ver como quedan bien los datos:



	EMAIL_JUGADOR	ID	ESPECIE	FUERZA	AGILIDAD	INTELIGENCIA	VITALIDAD	RESISTENCIA	NIVEL
1	jugador1@test.…	999	Humano	75	100	0	100	50	10

g) Resultado real del trigger para validar cambio en característica:

1) Resultado 1:

```
-- Caso 1 (Feliz): Iniciar Mision A (ID 1001), que no tiene previas.


PL/SQL procedure successfully completed.


1 row inserted.
```

2) Resultado 2:

```
-- Caso 2 (Triste): Intentar iniciar Mision C (ID 1003). Fallará porque Mision B no está completa.


PL/SQL procedure successfully completed.

>> ÉXITO: El trigger bloqueó la operación con el error esperado.


PL/SQL procedure successfully completed.


Rollback complete.
```

3) Resultado 3:

```
-- Caso 3 (Feliz): Completar Mision A e intentar iniciar Mision B (ID 1002).


PL/SQL procedure successfully completed.


1 row updated.


1 row inserted.


Commit complete.
```