Trabajo Práctico N°2 de Base de Datos I - TUIA

Integrantes:

Donnarumma, Cesar Julián

Demarré, Lucas Federico

Previgliano, Marco

Comisión: 2

Año: 2022

Considere la situación planteada en el Trabajo Práctico 1, en donde se está desarrollando la base de datos para el sistema de venta electrónica de entradas del Cine Paraíso.

1) Determine si las relaciones presentadas en la resolución del Trabajo Práctico 1 están en 3FN. En caso afirmativo, justificar por qué. En caso negativo, explicar por qué no lo están, y realizar las modificaciones necesarias para que si están en 3FN. Incluya el diagrama entidad-relación original, y el modificado (en caso de que haya hecho alguna modificación).

1FN) En primer caso deberemos determinar si las relaciones están en 1FN:

"Una relación está en 1FN si no posee atributos compuestos ni multivaluados".

Teniendo en cuenta nuestras relaciones:

- Ninguna posee atributos multivaluados
- Vamos a evaluar si alguna relación tiene atributos compuestos:

  En Sucursal: "Id" es la *PK* (un número incremental), y "ciudad" es el nombre de la ciudad, ambos son atómicos.

  En Sala: "Id" es parte de la *PK* (un número incremental), "cantidad_butacas" es un número, e "id_sucursal" también es un número, todos atómicos.

  En Película: "Id" es parte de la *PK* (un número incremental), "nombre" es el nombre de la película, "atp" y "subtítulos" son ambos *bit*, y "género" es el género al que pertenece la película, todos son indivisibles.

  En función: "Id" es un número incremental (parte de la *PK*), "id_pelicula" es un número, "id_sala" y "cantidad_butacas" ambos números (*FK*), "día" es una fecha y "horario" la hora a la que empieza la función, todos son atómicos.

  En Entrada: "id_funcion" (*FK*) y "nro_butaca" (juntos *PK*) son ambos números, "vendida" es un bit, "id_sala" y "máximo_butacas" (juntos *FK*) son ambos números, todos atómicos también.

  Por lo tanto, nuestras relaciones están en 1FN.

2FN) Ahora es hora de determinar si están en **2FN**:

"Una relación está en 2FN si y sólo si está en 1FN y todos los atributos no clave son totalmente dependientes de la clave".

- Ya dijimos que está en **1FN**.
- Nos toca ver si los atributos que no son clave primaria son completamente dependientes de la clave:

En sucursal:

Atributos clave: id
Atributos no clave: ciudad

$\boldsymbol{Id \rightarrow Ciudad}$ es df total
Por lo tanto, sucursal **está** en **2FN**

En sala:

Atributos clave: id, cantidad_butacas
Atributos no clave: id_sucursal

$\{\boldsymbol{id, cantidad\_butacas}\} \rightarrow \boldsymbol{id\_sucursal}$ es df, pero no es total, podríamos eliminar el atributo cantidad_butacas de X y la df sigue siendo válida.
Por lo tanto, sala **no está** en **2FN**

Solución: *Quitar como clave primaria a cantidad_butacas. Quedaría así:*

Atributos: id
Atributos no clave: cantidad_butacas, id_sucursal

$\boldsymbol{Id \rightarrow cantidad\_butacas}$ es df total
$\boldsymbol{Id \rightarrow id\_sucursal}$ es df total
Luego de corregirlo **si está** en **2FN**

En película:

Atributos clave: id
Atributos no clave: nombre, atp, subtítulos, genero

$\boldsymbol{Id \rightarrow nombre}$ es df funcional total
$\boldsymbol{Id \rightarrow atp}$ es df funcional total
$\boldsymbol{Id \rightarrow subtítulos}$ es df funcional total
$\boldsymbol{Id \rightarrow género}$ es df funcional total
Por lo tanto, película **está** en **2FN**

En función:

Atributos clave: id
Atributos no clave: id_pelicula, id_sala, cantidad_butacas, día, horario

$Id \rightarrow id\_pelicula$ es df funcional total
$Id \rightarrow id\_sala$ es df funcional total
$Id \rightarrow cantidad\_butacas$ es df funcional total
$Id \rightarrow día$ es df funcional total
$Id \rightarrow horario$ es df funcional total
Por lo tanto, función **está** en **2FN**

En entrada:

Atributos clave: id_funcion, nro_butaca
Atributos no clave: vendida, id_sala, máximo_butacas

$\{Id\_funcion, nro\_butaca\} \rightarrow vendida$ es df funcional total
$\{Id\_funcion, nro\_butaca\} \rightarrow id\_sala$ no es df funcional total, es parcial. Se puede eliminar "nro_butaca" y la df sigue siendo válida, y en ese caso si es total.

$\{Id\_funcion, nro\_butaca\} \rightarrow máximo\_butacas$ no es df funcional total, es parcial. Se puede eliminar "nro_butaca" y la df sigue siendo válida, y en ese caso si es total.
Por lo tanto, entrada **no está** en **2FN**.

Solución: Quitar como atributos (y por consecuencia como *FK*) a "id_sala" y "máximo_butacas" (para las consultas a ambos datos podemos acceder a través de "id_funcion" y la relación función). Quedaría así:

Atributos*:* id_funcion, nro_butaca
Atributos no clave*:* vendida

$\{Id\_funcion, nro\_butaca\} \rightarrow vendida$ *es df funcional total*
Luego de corregirlo **si está** en **2FN**.

Luego del anterior análisis de dependencias funcionales y las modificaciones podemos afirmar que el modelo ahora si está en 2FN.

3FN) Ahora es hora de determinar si están en **3FN:**

"Una relación está en 3FN si y solo si está en 2FN y todos los atributos no clave dependen de manera no transitiva de la clave primaria".

- Ya dijimos que está en 2FN.
- Nos toca ver si los atributos que no son clave primaria no dependen de manera no transitiva de la clave primaria:

En sucursal:

Atributos clave: id
Atributos no clave: ciudad

$Id \rightarrow ciudad$ es df no transitiva
Por lo tanto, sucursal **está** en **3FN.**

En sala:

Atributos clave: id
Atributos no clave: cantidad_butacas, id_sucursal

$Id \rightarrow cantidad\_butacas$ es df no transitiva
$Id \rightarrow id\_sucursal$ es df no transitiva
Por lo tanto, sala **está** en **3FN.**

En película:

Atributos clave: id
Atributos no clave: nombre, atp, subtítulos, género

$Id \rightarrow nombre$ es df no transitiva
$Id \rightarrow atp$ es df funcional no transitiva
$Id \rightarrow subtítulos$ es df funcional no transitiva
$Id \rightarrow género$ es df funcional no transitiva
Por lo tanto, película **está** en **3FN.**

En función:

Atributos clave: id
Atributos no clave: id_pelicula, id_sala, día, horario

$Id \rightarrow id\_pelicula$ es df no transitiva
$Id \rightarrow id\_sala$ es df no transitiva
$Id \rightarrow día$ es df funcional no transitiva
$Id \rightarrow horario$ es df funcional no transitiva
Por lo tanto, función **está** en **3FN.**

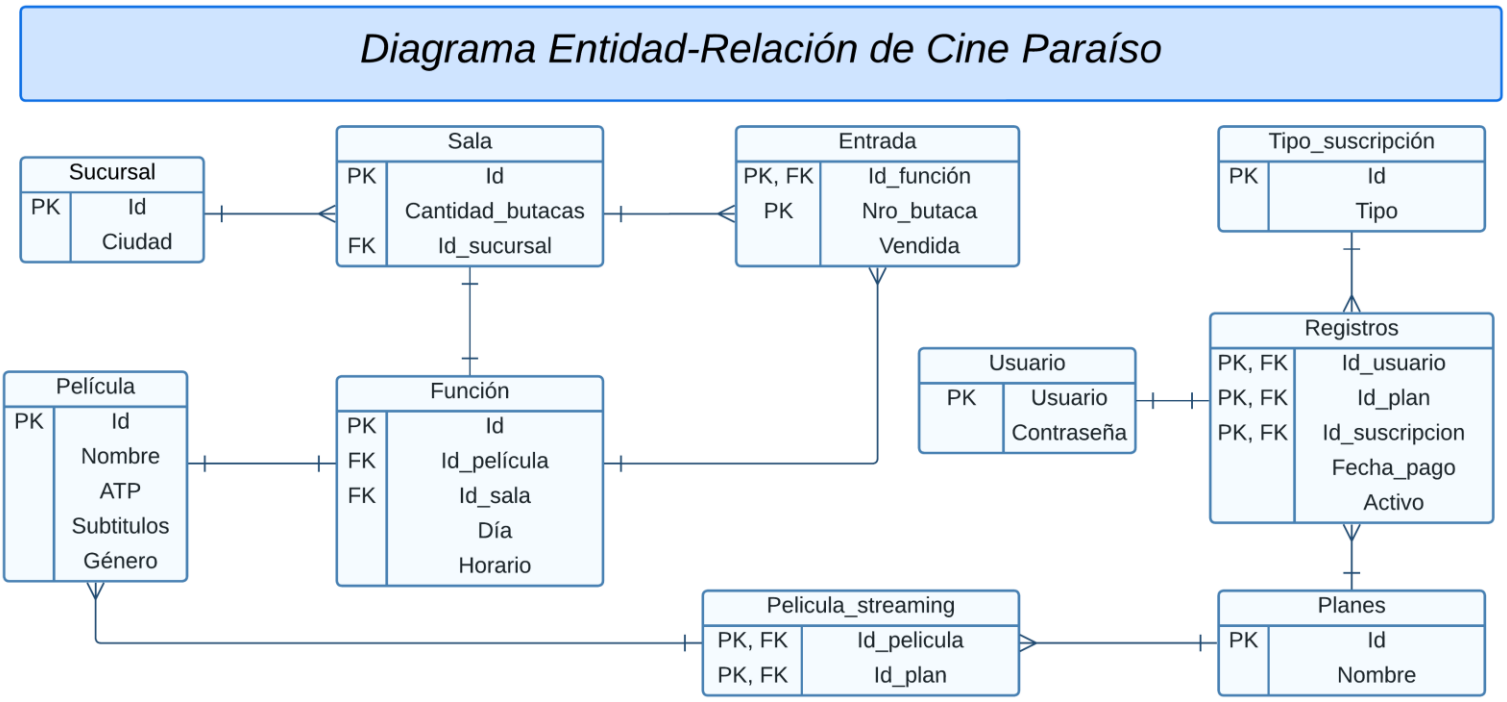
En entrada:

Atributos clave: id_funcion, nro_butaca
Atributos no clave: vendida

$\{Id\_funcion, nro\_butaca\} \rightarrow vendida$ es df funcional no transitiva
Por lo tanto, entrada **está** en **3FN.**

Solo resta hacerle la última consulta en rojo al profesor acerca de para determinar si está o no en 3FN.

2) Escriba el diagrama entidad-relación para dichas relaciones, explicando brevemente qué representa cada atributo, y justifique por qué las relaciones se encuentran normalizadas.



Tipo Suscripción: en esta entidad se instancian los “Tipos de Subscripción”, es decir, si las subscripciones son mensuales o anuales. “Id” es la *PK*, es un número incremental e identifica cada tipo, mientras que “Tipo” es un varchar (palabra) que es donde se guarda si el tipo es mensual o anual

Planes: en esta entidad se instancian los planes que existen en el servicio de streaming y el usuario puede elegir. “Id” es la *PK,* es un número incremental al igual que en **Tipo Suscripción**, este identifica cada plan, mientras que en “Nombre” se guarda como varchar el nombre justamente del plan.

Usuario: en esta entidad se instancian cada cuenta de cada usuario que se registre en el servicio. Sus PK son sus dos únicos atributos, “usuario” en donde se guarda como varchar la palabra que eligió el usuario como su nombre de usuario y “contraseña” que se guarda, valga la redundancia, su contraseña.

Registros: en esta entidad se instancian todos los registros que cada uno de los usuarios, es decir, se guarda la situación en la que la cuenta del usuario se encuentra, el plan elegido y, dependiendo si es de pago o no, su tipo y última fecha de pago. “Id_usuario” es *FK* y *PK*, es una clave foránea que viene desde la entidad **Usuarios**, en esta se identifica dentro del registro a quien pertenece el mismo. “Id_plan”, otra *FK* y *PK*, esta viene de **Planes** e identifica cual plan eligió el usuario. “Id_subcripción”, este atributo también es *FK* y *PK*, viene **Tipo Suscripción** e identifica si el usuario eligió mensual o anual y, por último, “Fecha_pago”, se guarda la fecha de la última vez que el usuario pago su plan en caso de ser necesario. “Activo”, representa si la suscripción del usuario esta activa, es decir, está al día con los pagos en caso de que su plan así lo requiera.

Pelicula_streaming: esta entidad tiene como PK sus dos únicos atributos “Id_plan” (*FK* de planes), para identificar a que plan pertenece cada entrada e “Id_pelicula” (*FK* de película) para identificar la película de cada entrada de la entidad. En esta entidad se guardan las películas que pertenecen a cada plan que existe actualmente en el servicio de streaming, la idea es permitir que se pueda saber a qué películas tienen acceso cada plan, para así mostrarle al usuario sus opciones dependiendo del plan elegido

Normalización y justificación:

En Tipo_suscripción:
Atributos clave: id
Atributos no clave: tipo
Todos los atributos son atómicos y mono evaluados, **está** en **1FN.**

$\boldsymbol{Id \rightarrow tipo}$ es df total
La relación **está** en **2FN.**

$\boldsymbol{Id \rightarrow tipo}$ es df no transitiva
Por lo tanto, tipo_suscripcion **está** en **3FN.**

En planes:
Atributos clave: id
Atributos no clave: nombre
Todos los atributos son atómicos y mono valuados, **está** en **1FN.**

$\boldsymbol{Id \rightarrow nombre}$ es df total
La relación **está** en **2FN.**

$Id \rightarrow tipo$ es df no transitiva
Por lo tanto, planes **está** en **3FN.**

En usuario:
Atributos clave: usuario
Atributos no clave: contraseña
Todos los atributos son atómicos y mono valuados, **está** en **1FN.**

$usuario \rightarrow contraseña$ es df total
La relación **está** en **2FN.**

$usuario \rightarrow contraseña$ es df no transitiva
Por lo tanto, usuario **está** en **3FN.**

En registro:
Atributos clave: id_usuario, id_plan, id_suscripcion
Atributos no clave fecha_pago, activo
Todos los atributos son atómicos y mono valuados, **está** en **1FN.**

$\{id\_usuario,\ id\_plan,\ id\_suscripcion\} \rightarrow fecha\_pago$ es df total
$\{id\_usuario,\ id\_plan,\ id\_suscripcion\} \rightarrow activo$ es df total
La relación **está** en **2FN.**

$\{id\_usuario,\ id\_plan,\ id\_suscripcion\} \rightarrow fecha\_pago$ es df no transitiva
$\{id\_usuario,\ id\_plan,\ id\_suscripcion\} \rightarrow activo$ es df no transitiva
Por lo tanto, registro **está** en **3FN.**

En pelicula_streaming:
Atributos clave: id_pelicula, id_plan
Todos los atributos son atómicos y mono valuados, **está** en **1FN.**

Al no existir otros atributos no claves, la tabla **está** en **2FN.**
Por lo tanto, película familiar **está** en **3FN.**

Luego del anterior análisis, podemos afirmar que las relaciones están en 3FN.

Antes de pasar a los ejercicios **3)** y **4),** vamos a crear la base de datos de las tablas únicamente de este trabajo práctico siguiendo su **MER**.

```
CREATE DATABASE cinema_paraiso_streaming;
USE cinema_paraiso_streaming;

CREATE TABLE tipo_suscripcion (
        id int IDENTITY(1,1) PRIMARY KEY,
        tipo varchar(20) UNIQUE NOT NULL
);

CREATE TABLE planes (
        id int IDENTITY(1,1) PRIMARY KEY,
        nombre varchar(20) UNIQUE NOT NULL
);

CREATE TABLE usuario (
        usuario varchar(40) PRIMARY KEY,
        contraseña varchar(40) NOT NULL
);

CREATE TABLE registros (
        id_usuario varchar(40),
        id_plan int,
        id_suscripcion int,
        fecha_pago date NULL,
        activo bit NOT NULL,
        PRIMARY KEY (id_usuario, id_plan, id_suscripcion),
        CONSTRAINT FK_registros_usuario FOREIGN KEY (id_usuario)
        REFERENCES usuario(usuario),
        CONSTRAINT FK_registros_planes FOREIGN KEY (id_plan)
        REFERENCES planes(id),
        CONSTRAINT FK_registros_suscripcion FOREIGN KEY (id_suscripcion)
        REFERENCES tipo_suscripcion(id)
);

CREATE TABLE pelicula(
  id int IDENTITY(1,1) PRIMARY KEY,
  nombre varchar(80) NOT NULL,
  atp bit NOT NULL,
  subtitulos bit NOT NULL,
  genero varchar(20) NOT NULL,
  );

CREATE TABLE pelicula_streaming (
  id_pelicula int FOREIGN KEY REFERENCES pelicula(id),
  id_plan int FOREIGN KEY REFERENCES planes(id)
  PRIMARY KEY(id_pelicula, id_plan),
  );
```

```
INSERT INTO planes VALUES ('Gratuito');
INSERT INTO planes VALUES ('Premium');
INSERT INTO planes VALUES ('Familiar');

INSERT INTO tipo_suscripcion VALUES ('Mensual');
INSERT INTO tipo_suscripcion VALUES ('Anual');

INSERT INTO usuario VALUES ('usuario1', '1234');
INSERT INTO usuario VALUES ('usuario2', '5678');
INSERT INTO usuario VALUES ('usuario3', '9123');
INSERT INTO usuario VALUES ('usuario4', '9123');
INSERT INTO usuario VALUES ('usuario5', '4253');
INSERT INTO usuario VALUES ('usuario6', '3235');
INSERT INTO usuario VALUES ('usuario7', '6523');
INSERT INTO usuario VALUES ('usuario8', '5423');
INSERT INTO usuario VALUES ('usuario9', '8790');

INSERT INTO registros VALUES ('usuario1', 1, 1, NULL, 1);
INSERT INTO registros VALUES ('usuario2', 2, 1, '2022-08-29', 1);
INSERT INTO registros VALUES ('usuario3', 2, 1, '2022-11-14', 1);
INSERT INTO registros VALUES ('usuario4', 3, 1, '2022-10-30', 1);
INSERT INTO registros VALUES ('usuario5', 3, 2, '2022-02-01', 1);
INSERT INTO registros VALUES ('usuario6', 1, 1, NULL, 1);
INSERT INTO registros VALUES ('usuario7', 1, 1, NULL, 1);
INSERT INTO registros VALUES ('usuario8', 3, 2, '2021-11-01', 1);
INSERT INTO registros VALUES ('usuario9', 3, 2, '2020-03-04', 1);
```

3) Suponga que se desea verificar mensualmente si los planes de cada uno de los usuarios están al día con los pagos y, en función de eso, actualizar el plan como activo o inactivo. Cree el procedimiento almacenado correspondiente, y proponga los criterios a tener en cuenta para pasar un plan de activo a inactivo.

Criterios para determinar si un plan está activo o inactivo

- *Solo vamos a trabajar con los registros cuyo plan no sea el gratuito. Consideramos que si el plan es el gratuito la fecha de pago (que es la fecha del último pago) se pone en **NULL** y que no hay pago que verificar en ese caso.*

- *Del resto de los planes: en caso de ser mensual se evaluará si pasaron 30 o más del día que pago y en caso de ser anual se evaluará si pasaron 12 meses o más para desactivar la cuenta.*

El ejercicio fue dividido en 2 procedimientos almacenados para mejorar la lectura del mismo. El primero desactiva registros individuales en caso de que se cumplan las condiciones necesarias. Y el segundo recorre la totalidad de los registros que no sean cuentas gratuitas (ya que estas no habría que verificarlas) y tomando una por una (a través de un bucle while) va llamando al anterior procedimiento almacenado para cumplir con la consigna

CREACION DEL PRIMER SP

El siguiente procedimiento almacenado cambia el valor de activo a inactivo de una sola cuenta. Recibe como parámetros de entrada la PK del registro que queremos comprobar, con esos datos busca la fecha de pago para luego (dependiendo de si su tipo de plan es mensual o anual) evaluar si paso el tiempo correspondiente y pasar de activo a inactivo a un registro.

```
GO
CREATE PROCEDURE usp_desactivacion_registro -- Nombre del procedimiento almacenado
        @id_usuario varchar(40), -- Parámetros de entrada: clave primaria de la tupla para identificarla correctamente
        @id_plan int,
        @id_suscripcion int
AS

DECLARE @fecha_pago date -- En esta variable guardaremos la fecha de pago

SELECT @fecha_pago = fecha_pago
FROM registros
WHERE id_usuario = @id_usuario AND id_plan = @id_plan AND id_suscripcion = @id_suscripcion

IF (@id_suscripcion = 1) -- Si la suscripción es mensual
        IF (SELECT DATEDIFF (DAY, @fecha_pago, GETDATE ())) >= 30 -- Si la diferencia de días entre el ultimo día que pago y el momento actual es mayor de 30

                UPDATE registros -- Seteamos en la tupla correspondiente activo = 0
                SET activo = 0
                WHERE id_usuario = @id_usuario AND id_plan = @id_plan AND id_suscripcion = @id_suscripcion

IF (@id_suscripcion = 2) -- Si la suscripción es anual
        IF (SELECT DATEDIFF (MONTH, @fecha_pago, GETDATE ())) >= 12 -- Si la diferencia de meses entre el último mes que pago y el mes actual es mayor de 12
                UPDATE registros -- Seteamos en la tupla correspondiente activo = 0
                SET activo = 0
                WHERE id_usuario = @id_usuario AND id_plan = @id_plan AND id_suscripcion = @id_suscripcion
GO
```

CREACION DEL SEGUNDO SP

Este procedimiento almacenado lo que hace es ir recorriendo cada una de las filas que no sean registros de plan gratuito y va tomando los valores de las PK una por una en un bucle WHILE y pasándoselas al anterior procedimiento para que el mismo cambie los valores en caso de ser necesario.

```
GO
CREATE PROCEDURE usp_verificacion_mensual -- Creación de procedimiento almacenado
AS

DECLARE @registros_provisorios table (id_usuario varchar (40), id_plan int,
id_suscripcion int)
-- La variable de arriba es de tipo tabla, son lo mismo que las tablas que creamos para los registros con la diferencia que se crea al ejecutar la Query y elimina al terminar la ejecución

INSERT INTO @registros_provisorios SELECT id_usuario, id_plan, id_suscripcion
FROM registros WHERE id_plan != 1
-- En la variable provisoria tipo tabla ingresamos los atributos id_usuario, id_plan, id_suscripcion de las filas de la tabla registro donde el plan no sea gratuito. esta variable va a contener los registros que tenemos que pasarle al procedimiento almacenado que desactiva

DECLARE @contador int = (SELECT COUNT(*) FROM @registros_provisorios)
-- Esta variable es un contador con el total de filas que tenemos que recorrer, nos sirve como condición del while, para que el mismo termine

WHILE @contador > 0 -- Mientras haya registros aun en la variable tipo tabla creada se ejecuta
BEGIN
        DECLARE @id_usuario varchar (40)
        DECLARE @id_plan int
        DECLARE @id_suscripcion int
        -- En estas 3 variables vamos a guardar los datos de la PK de cada fila para pasársela como parámetro de entrada, al procedimiento que nos desactiva cuentas

        SELECT TOP 1 @id_usuario = id_usuario, @id_plan = id_plan,
@id_suscripcion = id_suscripcion FROM @registros_provisorios
        -- Guardamos en nuestras variables los datos de la PK del primer registro. siempre de los registros que tenemos que comprobar vamos agarrando el de arriba.

        EXECUTE usp_desactivacion_registro @id_usuario, @id_plan,
@id_suscripcion
        -- Ejecutamos nuestro proceso de almacenamiento que desactiva

        DELETE FROM @registros_provisorios WHERE id_usuario = @id_usuario AND
id_plan = @id_plan AND id_suscripcion = @id_suscripcion
        -- Eliminamos el registro que recién acabamos de comprobar en el renglón de arriba para continuar con el de abajo en la próxima iteración (si es que aún quedan registros)

        SET @contador = (SELECT COUNT(*) FROM @registros_provisorios)
        -- Le seteamos a nuestro contador el valor correspondiente a la cantidad de registros que quedan en la variable tipo tabla. sí es != 0 seguiremos iterando, sino terminamos
END
GO
```

4) Cree un procedimiento almacenado que reciba como parámetros un usuario y una contraseña, y devuelva 1 si el login es correcto (es decir, coincide usuario, contraseña, y el plan está activo) y 0 en cualquier otro caso.

```
-- Recibe como entrada un usuario, una contraseña y devuelve un entero como salida.
GO
CREATE PROCEDURE usp_login
        @usuario varchar (40), -- Parámetros de entrada
        @contraseña varchar (40),
        @salida int OUTPUT -- Parámetro de salida
AS

        DECLARE @correcto int -- Variable que guardara el resultado de si hay coincidencia

        -- Esta consulta nos devuelve un conteo de la cantidad de registros donde un usuario y contraseña de la base de datos (dados de alta) coincidan con los usuarios y contraseña que pasamos. este resultado se guarda en la variable que declaramos arriba
        SELECT @correcto = COUNT (*)
        FROM registros AS reg, usuario AS usu,
        (SELECT @usuario AS 'usuario', @contraseña AS 'contraseña', 1 AS 'activo') as entrada
        WHERE reg.id_usuario = usu.usuario AND entrada.usuario = usu.usuario
        AND entrada.contraseña = usu.contraseña AND entrada.activo = reg.activo

        IF (@correcto = 1) -- Si hay una coincidencia
                SET @salida = 1 -- Seteamos 1 a la variable de salida
        ELSE -- si no la hay
                SET @salida = 0 -- Seteamos 0 a la variable de salida

        RETURN @salida -- Devolvemos la variable
GO

-- Bloque de código de login:
GO
DECLARE @salida int
EXECUTE usp_login 'usuario2', '5678', @salida OUTPUT -- El primer parámetro es el usuario, el segundo la contraseña
SELECT @salida AS 'Resultado login (1-Correcto, 0-Incorrecto)'
GO
```