PRACTICA N°2

1. Agregar a la tabla **tDeudas** las siguientes columnas: Id del tipo int, Identity(1,1), pTributo, pReajuste, pinteres, pGasto, estas columnas se usarán para el registro de los pagos que se hagan por cada uno de esos conceptos y por defecto tienen valor cero, pagado de tipo bit para indicar si una fila de deuda ya está cancelada por defecto tiene valor cero. Crear una tabla **tPago** en la cual se registrarán los pagos de las deudas, debe contener las siguientes columnas: Id del tipo identidad, código del contribuyente, año, codTrib, idDeuda (foránea de id de tDeudas), nTributo, nReajuste, ninteres, nGasto, numrecibo, fechapago, anulado del tipo bit (1: Anulado, 0: No anulado), por defecto cero. Crear una un disparador trPagos en la tabla tPago, el mismo que se ejecuta cuando se hace un insert o update a esa tabla. Funcionará de la siguiente manera.

```
ALTER TABLE tDeudas ADD id int identity(1,1), pTributo numeric(11,2) DEFAULT 0,
pReajuste numeric(11,2) DEFAULT 0, pInteres numeric(11,2) DEFAULT 0,
pGasto numeric(11, 2) DEFAULT 0, pagado bit DEFAULT 0

ALTER TABLE tDeudas ADD CONSTRAINT PK_tDeudas PRIMARY KEY (id)
DROP TABLE tPago
CREATE TABLE tPago (
id int identity(1, 1) PRIMARY KEY,
cCod_cont char(11) NOT NULL,
cAño char(4) NOT NULL,
cCod_Trib char(7) NOT NULL,
idDeuda int NOT NULL,
nTributo numeric(11, 2) DEFAULT 0,
nReajuste numeric(11, 2) DEFAULT 0,
nInteres numeric(11, 2) DEFAULT 0,
nGasto numeric(11, 2) DEFAULT 0,
numRecibo int, fechaPago datetime,
anulado bit DEFAULT 0,
CONSTRAINT FK_tPago_tDeudas
FOREIGN KEY (idDeuda) REFERENCES tDeudas (id)
)
```

a) Cuando se hace **Insert** deberá actualizar en la tabla tDeudas los campos pTributo, pReajuste, plnteres, pGasto, sumando los respectivos valores de la tabla tPago. Además se debe verificar si en la tabla tDeudas las suma de nTributo+ninteres+nReajste+nGasto es menor o igual a pTributo+pinteres+pReajste+pGasto, deberá cambiar el campo pagado a 1, para indicar que esa deuda ya está pagada.

b) Cuando se haga Update: Si el campo anulado cambia a 1 (se anula el recibo), entonces en tDeudas deberá restar a las columnas pTributo, plnteres, pReajste, pGasto los valores que hay en ese registro del recibo anulado. También debe verificar en la tabla tDeudas que si el campo pagado está en 1, volverlo a cero si no se cumple la condición por la cual se cambió a 1 en el momento que se hizo la inserción al pago.

```
CREATE TRIGGER tr_Pagos
ON tPago
AFTER INSERT,UPDATE
AS
BEGIN
	DECLARE @pTributo numeric(11, 2),
	@pReajuste numeric(11, 2),
	@pInteres numeric(11, 2),
	@pGasto numeric(11, 2),
	@idDeuda int,
	@anulado bit

	SELECT @idDeuda = idDeuda,
		@pTributo = nTributo,
		@pReajuste = nReajuste,
		@pInteres = nInteres,
		@pGasto = nGasto
	FROM inserted

	IF NOT EXISTS (SELECT * FROM deleted) --Inserción
	BEGIN
		UPDATE tDeudas
		SET pTributo = pTributo+ @pTributo,
		    pReajuste = pReajuste+ @pReajuste,
		    pInteres = pInteres+ @pInteres,
		    pGasto = pGasto+ @pGasto
		WHERE id = @idDeuda

		  UPDATE tDeudas
		  SET pagado = 1
		  WHERE id = @idDeuda
		  AND (nTributo+nReajuste+nInteres+nGasto) <=
		      (pTributo+pReajuste+pInteres+pGasto)
	END
	ELSE
	BEGIN
		IF UPDATE(anulado)
		BEGIN
		    SELECT @anulado = anulado
		    FROM inserted

		    IF @anulado = 1
		    BEGIN
			UPDATE tDeudas
			SET pTributo = pTributo- @pTributo,
			    pReajuste = pReajuste- @pReajuste,
			    pInteres = pInteres- @pInteres,
			    pGasto = pGasto- @pGasto
			WHERE id = @idDeuda

			  UPDATE tDeudas
			  SET pagado = 0
			  WHERE id = @idDeuda
			  AND (nTributo+nReajuste+nInteres+nGasto) >=
			      (pTributo+pReajuste+pInteres+pGasto)
		    END
		END
	END
END
```

PRACTICA N°2

2. Elaborar un procedimiento almacenado **paActualiza** Pago que permita hacer la actualización de la tabla tPago. Se pasan como parámetros el id del pago, iddeuda, numrecibo, fechapago, pTributo, pinteres, pReajuste, pGasto, anulado. Cuando el id del pago es cero entonces quiere decir que se va a hacer una inserción, con el iddeuda se obtienen las otras columnas que se requieren desde la tabla tDeudas. Cuando el id del pago es diferente de cero, quiere decir que se va a hacer una actualización, por un tema de seguridad las actualizaciones sólo se pueden hacer a la columna anulado, entonces se pasará en el parámetro anulado el valor de 1, entonces los únicos parámetros que se requieren son el id de pago y anulado. Todos los parámetros pueden tener valores por defecto, 0 para los números, espacio vacío (") para los char y 01/01/1900 para la fecha.

```
SELECT top 100 *
FROM tDeudas
CREATE PROCEDURE paActualizaPago (@idPago int=0, @idDeuda int=0, @numrecibo int=0,
                @fechaPago datetime='1900-01-01', @pTributo numeric(11, 2)=0,
                @pReajuste numeric(11, 2)=0, @pInteres numeric(11, 2)=0,
                @pGasto numeric(11, 2)=0, @anulado bit=0)
AS
BEGIN
      IF (@idPago = 0) -Inserción
        BEGIN
            INSERT INTO tPago (cCod_cont, cAño, cCod_Trib, idDeuda,nTributo,
                               nReajuste,nInteres,nGasto,numRecibo,fechaPago)
            SELECT cCod_cont,
                   cAño,
                   cCod_Trib,
                   @idDeuda,
                   @pTributo,
                   @pReajuste,
                   @pInteres,
                   @pGasto,
                   @numrecibo,
                   @fechaPago
            FROM tDeudas
            WHERE id = @idDeuda
        END
      ELSE -Actualizacion
      BEGIN
            UPDATE tPago
            SET anulado = @anulado
            WHERE id = @idPago
        END
END

--PRUEBAS

SELECT getdate() 2024-02-19 18:11:18.543
SELECT cast('19-02-2024' AS datetime)
SELECT top 100 *
FROM tDeudas EXEC paActualizaPago 0, 5, 1, '19-02-2024 18:11:18.543',30,0,5,7
SELECT *
FROM tPago EXEC paActualizaPago 1, 0,0 ,'01-01-1900',0,0,0,0,1
```