Unidad 4
SQL Procedural

4.1 Procedimientos y Funciones Almacenados.

4.2 Disparadores (*Triggers*).

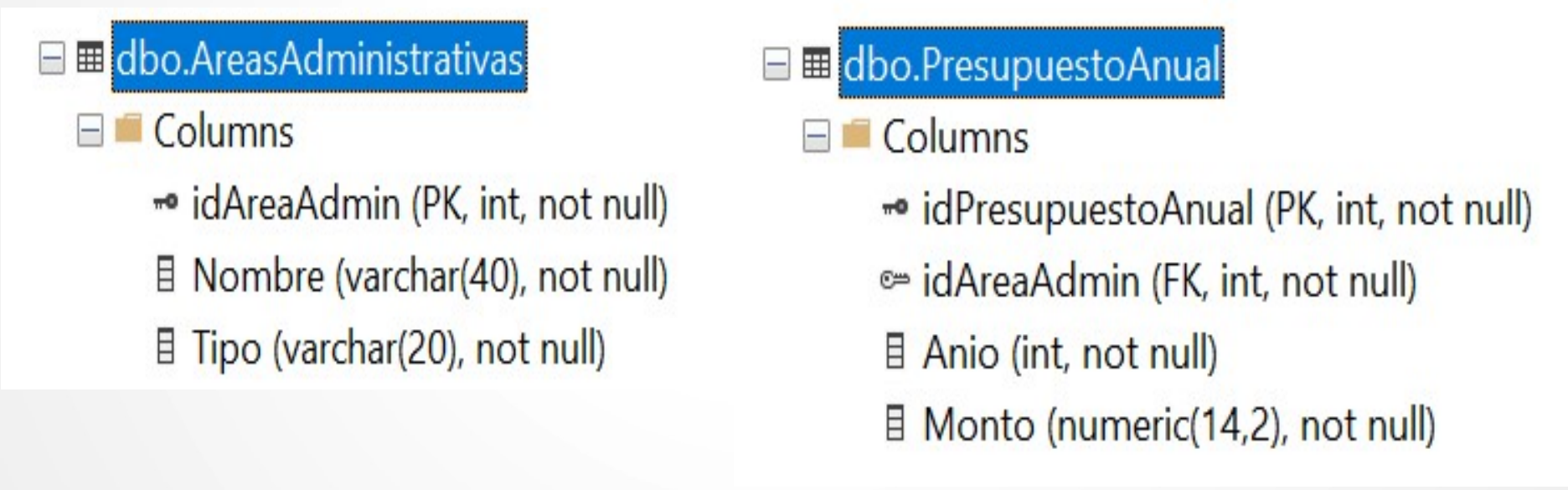
4.1 Procedimientos y Funciones Almacenados

En el ITD anualmente se asigna a las diferentes áreas administrativas una cantidad de recursos económicos para cubrir sus necesidades operativas.

Las diferentes áreas son la Dirección del Plantel, tres subdirecciones y más de veinte Departamentos Académicos, Administrativos y de Planeación.

4.1 Procedimientos y Funciones Almacenados

Hay que crear las tablas siguientes en la BD ITD para registrar los datos respecto a los presupuestos de todas las Áreas Administrativas.



4.1 Procedimientos y Funciones Almacenados

Añadir las tuplas que se indican a la tabla *AreasAdministrativas*.

```
insert into AreasAdministrativas (Nombre,Tipo)
Values  ('Dirección','Dirección'),
        ('Subdirección Académica','Subdirección'),
        ('Subdirección Administrativa','Subdirección'),
        ('Subdirección de Planeación','Subdirección'),
        ('Sistemas y Computación','Departamento'),
        ('Metal Mecánica','Departamento'),
        ('Química y Bioquímica','Departamento'),
        ('Eléctrica y Electrónica','Departamento'),
        ('Ciencias de la tierra','Departamento'),
        ('Recursos Financieros','Departamento'),
        ('Recursos Humanos','Departamento'),
        ('Servicios Escolares','Departamento'),
        ('Centro de Información','Departamento')
```

4.1 Procedimientos y Funciones Almacenados

Añadir las tuplas que se indican a la tabla *PresupuestoAnual*.

```
insert into PresupuestoAnual (idAreaAdmnistrativa, Anio, Monto)
values  (1,2024,100000),
        (2,2024,50000),
        (3,2024,50000),
        (4,2024,50000),
        (5,2024,500000),
        (6,2024,500000),
        (7,2024,500000),
        (8,2024,500000),
        (9,2024,500000),
        (10,2024,150000),
        (11,2024,150000),
        (12,2024,150000),
        (13,2024,150000)
```

4.1 Procedimientos y Funciones Almacenados

Si ejecutamos la siguiente consulta:

```
select sum(monto)
    from PresupuestoAnual
    where Anio=2024
```

Se obtiene 3,350,000.00 lo que significa que el presupuesto total anual para todas las áreas administrativas será esa cantidad.

4.1 Procedimientos y Funciones Almacenados

- El presupuesto asignado total a las áreas administrativas es una **cantidad constante**, no puede aumentar o disminuir.
- Aunque el importe total no se puede cambiar, los montos parciales de cada área si.
  - La condición para hacerlo es que se si se disminuye una cantidad presupuestada a una de las áreas, se aumente exactamente el mismo monto a otra.

4.1 Procedimientos y Funciones Almacenados

- Hay que crear un procedimiento almacenado para traspasar una cantidad del presupuesto de un área hacia otra.
- El procedimiento recibirá los siguientes parámetros:
  - idAreaAdmin origen.
  - idAreaAdmin destino.
  - Año.
  - Importe del traspaso.

4.1 Procedimientos y Funciones Almacenados

```
create procedure sp_TraspasoPresupuesto
    @vIdAreaOrigen int,
    @vIdAreaDestino int,
    @vAnio int,
    @vImporteMover numeric(12,2)
as
BEGIN
    update PresupuestoAnual
        set Monto=Monto-@vImporteMover
        where idAreaAdmin=@vIdAreaOrigen and Anio=@vAnio
    update PresupuestoAnual
        set Monto=Monto+@vImporteMover
        where idAreaAdmin=@vIdAreaDestino and Anio=@vAnio
END
```

Nombre del procedimiento almacenado

Lista de parámetros

Comandos que ejecutará el procedimiento

4.1 Procedimientos y Funciones Almacenados

```
select * from PresupuestoAnual
select sum(Monto)
   from PresupuestoAnual
   where Anio=2024
```

Se consulta el presupuesto de todas las áreas por separado y la suma total antes de realizar un traspaso.

Llamada a ejecución del procedimiento almacenado

```
execute dbo.sp_TraspasoPresupuesto 1,2,2024,10000
```

Argumentos

```
select * from PresupuestoAnual
select sum(Monto)
   from PresupuestoAnual
   where Anio=2024
```

Se consulta el presupuesto de todas las áreas por separado y la suma total después de realizado el traspaso.

4.1 Procedimientos y Funciones Almacenados

Ejercicio:

- Añadir a la tabla ***PresupuestoAnual*** una restricción *check* para que solo se permitan montos del presupuesto mayores a cero.
- En la diapositiva siguiente se modifica el procedimiento almacenado para atrapar el error por la restricción *check* recién añadida.
- Ejecute el procedimiento almacenado ya modificado para probar intentando retirar un monto mayor al disponible del área de origen.

4.1 Procedimientos y Funciones Almacenados

```
alter procedure sp_TraspasoPresupuesto
                @vIdAreaOrigen int,
                @vIdAreaDestino int,
                @vAnio int,
                @vImporteMover numeric(12,2) as
BEGIN
    BEGIN TRY
      update PresupuestoAnual
            set Monto=Monto-@vImporteMover
            where idAreaAdmin=@vIdAreaOrigen and Anio=@vAnio
      update PresupuestoAnual
            set Monto=Monto+@vImporteMover
            where idAreaAdmin=@vIdAreaDestino and Anio=@vAnio
    END TRY
    BEGIN CATCH
      if ERROR_MESSAGE() like '%CK_PresupuestoMonto%'
            RAISERROR('¡Monto insuficiente para el traspaso!',0,0)
    END CATCH
END
```

Modificar el procedimiento almacenado

Nombre del la restricción *check*

4.1 Procedimientos y Funciones Almacenados

Crear una función almacenada que calcule y regrese la suma total de los presupuestos de todas las áreas.

```
create function fn_PresupuestoTotal(@vAnio int)
        returns numeric(14,2)
as
BEGIN
    declare @vMontoTotal numeric(14,2)
    select @vMontoTotal=sum(Monto)
        from PresupuestoAnual
        where Anio=2024
return @vMontoTotal
END
```

El valor calculado se asigna a una variable para regresarlo

Como es una función debe dar como resultado un valor

4.1 Procedimientos y Funciones Almacenados

Ejemplos de llamadas a ejecución:

```
select dbo.fn_PresupuestoTotal(2024)

select dbo.fn_PresupuestoTotal(2024)
   as PresupuestoTotal

begin
   declare @vPresupuesto numeric(14,2)
   select @vPresupuesto=dbo.fn_PresupuestoTotal(2024)
   select @vPresupuesto
end
```

La función siguiente regresa una tabla que contiene los valores válidos de una restricción *Check* (solo es aplicable a valores char/varchar).

```
CREATE FUNCTION fn_ExtraeDominioCheck(@vNombreCheck VARCHAR(MAX))
RETURNS @resultado TABLE (Dominio VARCHAR(MAX))
AS
BEGIN
  DECLARE @vExpresion varchar(MAX)
  SELECT @vExpresion=definition FROM sys.check_constraints WHERE name=@vNombreCheck
  DECLARE @Apostrofo VARCHAR(1) = CHAR(39)
  DECLARE @inicio INT, @fin INT, @vSubCadena VARCHAR(MAX)
  SET @inicio = CHARINDEX(@Apostrofo, @vExpresion)
  WHILE @inicio > 0
  BEGIN
    SET @fin = CHARINDEX(@Apostrofo, @vExpresion, @inicio + 1)
    IF @fin > 0
      BEGIN
      SET @vSubCadena = SUBSTRING(@vExpresion, @inicio + 1, @fin - @inicio - 1)
      INSERT INTO @resultado (Dominio) VALUES (@vSubCadena)
      SET @inicio = CHARINDEX(@Apostrofo, @vExpresion, @fin + 1)
      END
    ELSE
      BEGIN
      SET @inicio = 0
      END
    END
    RETURN
END
```

La sintaxis de funciones de tabla es diferente de las funciones escalares: se debe establecer el nombre de la tabla a regresar.

No se debe indicar la tabla que regresará, solo se escribe el RETURN que finaliza la función.

4.1 Procedimientos y Funciones Almacenados

Llamadas a ejecución de prueba:

```
select * from
   dbo.fn_ExtraeDominioCheck('CK_PersonasSexo')
select * from
   dbo.fn_ExtraeDominioCheck('CK_AlumInasistMotivo')
```

Uso practico de la función ***dbo.fn_ExtraeDominioCheck()***:

- Modificar el formulario que escribió hace unas sesiones (en la unidad anterior) para dar de alta los datos de un alumno.
- Deberá usar esta función para obtener los datos de la restricción Check y asignarlos a ***cbSexo*** en vez de usar una tabla de dominio.
- El de abajo es un ejemplo genérico en C#. Se deberán hacer los ajustes necesarios.

```
public Form1() {
   InitializeComponent();
   LlenarComboBox();
   }
private void LlenarComboBox() {
     string cadenaConexion = "Data Source=ING-ALANIS-LAP\\SQLEXPRESS;" +
            "Initial Catalog=ITD;Integrated Security=True";
     string consulta = "select * from dbo.fn_ExtraeDominioCheck('CK_AlumnosInasist')";
     using (SqlConnection conexion = new SqlConnection(cadenaConexion)) {
          SqlCommand comando = new SqlCommand(consulta, conexion);
          try {
             conexion.Open();
             SqlDataReader lector = comando.ExecuteReader();
             while (lector.Read()) {
                comboBox1.Items.Add(lector["Dominio"].ToString());
             }
             lector.Close();
          }
          catch (Exception ex) {
             MessageBox.Show("Error: " + ex.Message);
          }
     }
}
```

Ejercicio

- Añadir al menos las tuplas que se indican, a las tablas de las siguientes diapositivas.
- Luego resolver la consulta que se solicita.

4.1 Procedimientos y Funciones Almacenados

```
insert into Maestros
       (Rfc, MaxGradoEstudios, Sueldo, idPersona)
values ('PP11','Doctorado',15000,4),
       ('AA00','Licenciatura',12000,5),
       ('AA11','Licenciatura',12500,8),
       ('GG22','Maestría',13000,2),
       ('TT44','Maestría',25000,9),
       ('BB99','Maestría',20000,10)
```

IdMaestro	RFC	MaxGradoEstudios	Sueldo	IdPersona
1	PP11	Doctorado	15,000	4
2	AA00	Licenciatura	12,000	5
3	AA11	Licenciatura	12,500	8
4	GG22	Maestría	13,000	2
5	TT44	Maestría	25,000	9
6	BB99	Maestría	20,000	10

4.1 Procedimientos y Funciones Almacenados

```
insert into Materias
    (ClaveInterna, ClaveOficial, Nombre, Horasteoria, HorasPractica)
    values ('1000','ISC-2345','Algebra Lineal',5,0),
           ('1100','ISC-2360','Fundamentos de Bases de Datos',3,2),
           ('1500','ISC-4300','Taller de Bases de Datos',0,4)
```

IdMateria	ClaveInterna	ClaveOficial	Nombre	HorasTeoria	HorasPractica	Creditos
1	1000	ISC-2345	Algebra Lineal	5	0	10
2	1100	ISC-2360	Fundamentos de Bases de Datos	3	2	8
3	1500	ISC-4300	Taller de Bases de Datos	0	4	4

4.1 Procedimientos y Funciones Almacenados

```
insert into Grupos (idMateria,Periodo,Paquete)
    values    (2,'2016B','4Z’),
              (3,'2016B','5W’),
              (3,'2016A','5W’),
              (1,'2016V','1A’),
              (1,'2016V','1B')
```

idGrupo	idMateria	Periodo	Paquete
1	2	2016B	4Z
2	3	2016B	5W
3	3	2016A	5W
4	1	2016V	1A
5	1	2016V	1B

4.1 Procedimientos y Funciones Almacenados

```
insert into
   GruposMaestro (idGrupo, idMaestro)
values
   (1,3), (3,2), (4,1), (5,1)
```

idGposMtro	idGrupo	idMaestro
1	1	3
2	3	2
3	4	1
4	5	1

```
insert into GruposHorarios
    (idGrupo, Dia, HoraInicial,
    HoraFinal, Salon)
values
    (1,'L',  '19:00', '20:00', 'T4'),
    (1,'Ma', '19:00', '20:00', 'T4'),
    (1,'Mi', '19:00', '20:00', 'T4'),
    (1,'J',  '19:00', '20:00', 'T4'),
    (1,'V',  '19:00', '20:00', 'T4'),
    (2,'L',  '13:00', '14:00', 'LC3'),
    (2,'Ma', '13:00', '14:00', 'LC3’),
    (2,'Mi', '13:00', '14:00', 'LC3'),
    (2,'J',  '13:00', '14:00', 'LC3'),
    (3,'L',  '13:00', '14:00', 'LC4'),
    (3,'Ma', '13:00', '14:00', 'LC4'),
    (3,'Mi', '13:00', '14:00', 'LC4'),
    (3,'J’,  '13:00', '14:00', 'LC4'),
    (4,'L’,  '08:00', '09:00', 'SC3'),
    (4,'Ma', '08:00', '09:00', 'SC3’),
    (4,'Mi', '08:00', '09:00', 'SC3'),
    (4,'J',  '08:00', '09:00', 'SC3’),
    (5,'L',  '14:00', '15:00', 'SC12'),
    (5,'Ma', '14:00', '15:00', 'SC12'),
    (5,'Mi', '14:00', '15:00', 'SC12'),
    (5,'J',  '14:00', '15:00', 'SC12')
```

idGrupoHorario	idGrupo	Dia	HoraInicial	HoraFinal	Salon
1	1	L	19:00	20:00	T4
2	1	Ma	19:00	20:00	T4
3	1	Mi	19:00	20:00	T4
4	1	J	19:00	20:00	T4
5	1	V	19:00	20:00	T4
6	2	L	13:00	14:00	LC3
7	2	Ma	13:00	14:00	LC3
8	2	Mi	13:00	14:00	LC3
9	2	J	13:00	14:00	LC3
10	3	L	13:00	14:00	LC4
11	3	Ma	13:00	14:00	LC4
12	3	Mi	13:00	14:00	LC4
13	3	J	13:00	14:00	LC4
14	4	L	08:00	09:00	SC3
15	4	Ma	08:00	09:00	SC3
16	4	Mi	08:00	09:00	SC3
17	4	J	08:00	09:00	SC3
18	5	L	14:00	15:00	SC12
19	5	Ma	14:00	15:00	SC12
20	5	Mi	14:00	15:00	SC12
21	5	J	14:00	15:00	SC12

4.1 Procedimientos y Funciones Almacenados

A partir de las tablas anteriores escriba una consulta para mostrar los grupos ofrecidos indicando, además de la información de cada grupo, la clave interna de la materia, nombre de la materia, horario asignado y nombre completo del profesor. Se sugiere usar vistas anidadas para simplificar la obtención de la consulta.

El horario debe mostrarse como una cadena que debe obtenerse a partir de las tuplas de la tabla ***GruposHorarios*** del primer esquema de la BD diseñado en la Unidad 1.
Debe escribir una función almacenada para convertir la información de las tuplas en una cadena que sea similar a la del ejemplo.

Periodo	ClaveInterna	NombreMateria	Paquete	Horario	NombreMaestro
2016B	1100	Fundamentos de Bases de Datos	4Z	L Ma Mi J V 19-20 T4	Aristóteles
2016B	1500	Taller de Bases de Datos	5W	L Ma Mi J 13-14 LC3	NULL
2016A	1500	Taller de Bases de Datos	5W	L Ma Mi J 13-14 LC4	Abraham
2016V	1000	Algebra Lineal	1A	L Ma Mi J 8-9 SC3	Pasteur
2016V	1000	Algebra Lineal	1B	L Ma Mi J 14-15 SC12	Pasteur

4.2 Disparadores (Triggers)

Un ***trigger*** es un objeto de la B.D., que está asociado con una tabla y que se activa cuando cierto evento ocurre.

Los eventos que el DBMS monitorea para efectos de los *triggers* son: **insert, update, delete**.

4.2 Disparadores (Triggers)

Los *triggers* pueden configurarse para que se activen **antes** o **después** del evento deseado, por ejemplo:

Un *trigger* que se active

- **antes** de que se borre una tupla.
- **después** de que se modifique una tupla.
- Por mencionar algunas de las posibilidades.

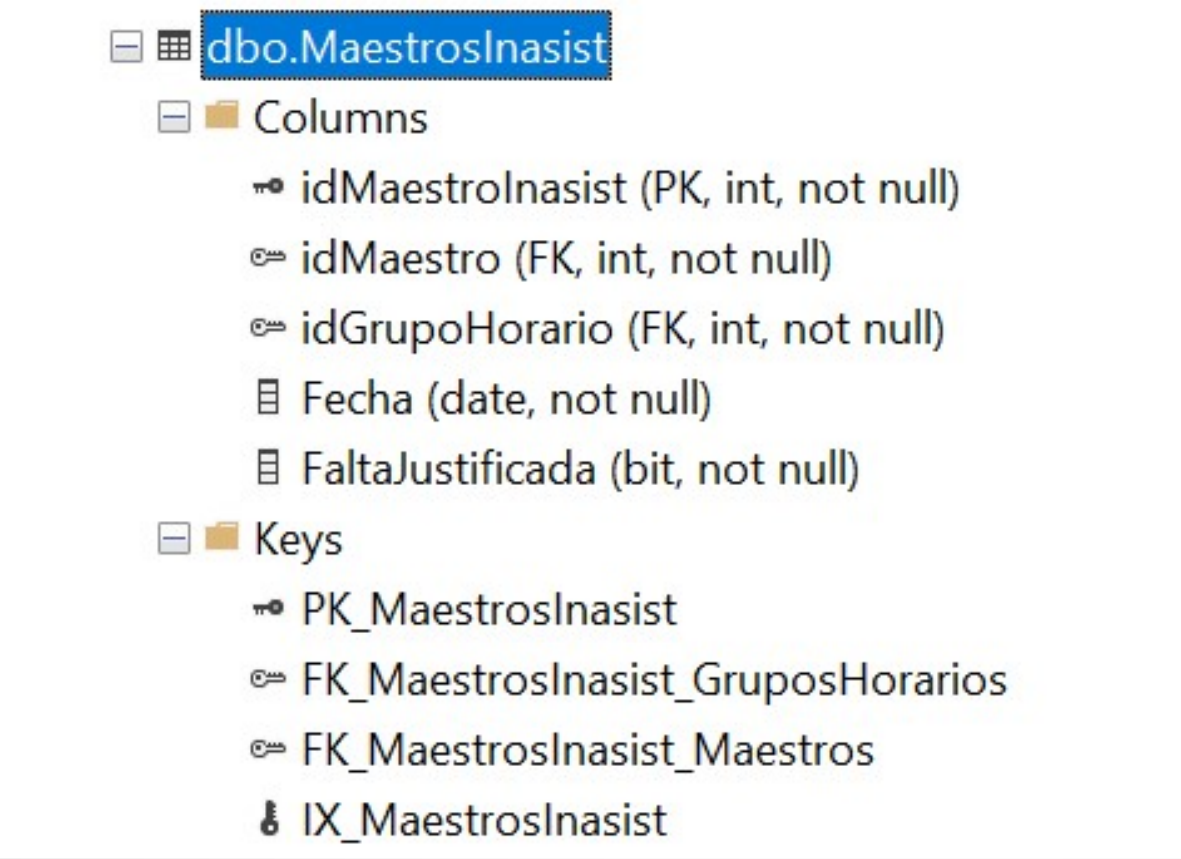
Ejemplo

Usando la base de datos ITD, crear una vista llamada ***vMaestrosCorreo*** que contenga los siguientes datos de los profesores:

idMaestro	NombreCompleto	Email
1	Luis Pasteur	pasteur@lala.mx
2	Abraham Zabludovsky	abraham@televisa.com
6	Sandro Botticelli	sandro.boticelli@florencia.it

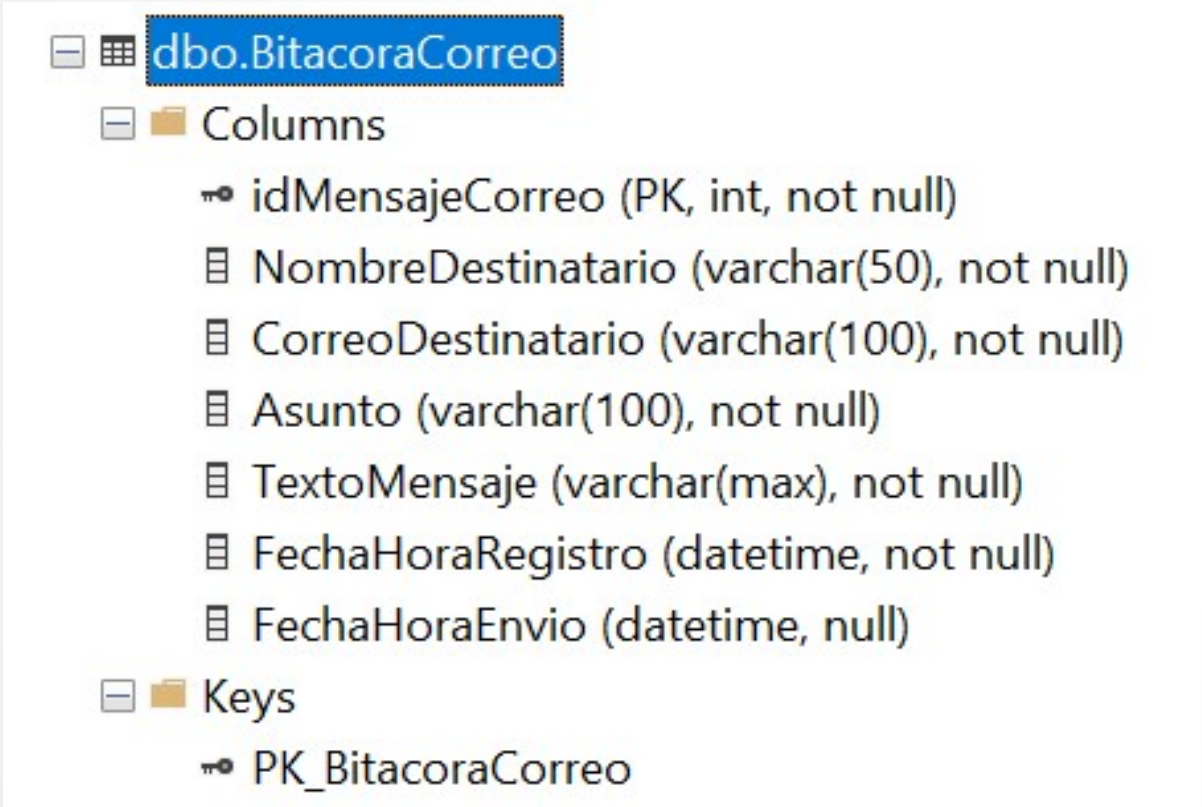
4.2 Disparadores (Triggers)

Crear la tabla MaestrosInasist con el siguiente esquema:



4.2 Disparadores (Triggers)

Crear la tabla BitacoraCorreo. Las bitácoras son recipientes de eventos.



4.2 Disparadores (Triggers)

En seguida se crea un *trigger* que se activará cada vez que se añada una tupla correspondiente a la Inasistencia de un profesor al aula.

El *trigger* causará el *insert* de una tupla en la tabla ***BitacoraCorreo***. Esta tabla registra los eventos de envío de mensajes de correo, en este caso a los profesores comunicándoles que deberán justificar la inasistencia o procederá un descuento en su salario.

```
CREATE TRIGGER tMaestrosInasist
ON MaestrosInasist AFTER INSERT
AS
BEGIN
     DECLARE @vIdMaestro INT
     DECLARE @vIdGrupoHorario INT
     DECLARE @vFechaFalta DATE
     DECLARE @vHoraFalta TIME
     DECLARE @vNombreCompleto VARCHAR(MAX)
     DECLARE @vEmail VARCHAR(MAX)
     SELECT
          @vIdMaestro=idMaestro,
          @vIdGrupoHorario=idGrupoHorario,
          @vFechaFalta=Fecha FROM INSERTED
     SELECT
          @vNombreCompleto=NombreCompleto,
          @vEmail=Email FROM vMaestrosCorreo WHERE idMaestro=@vIdMaestro
     SELECT
        @vHoraFalta=HoraInicial FROM GruposHorarios WHERE idGrupoHorario=@vIdGrupoHorario
     INSERT INTO
          BitacoraCorreo (
          NombreDestinatario,
          CorreoDestinatario,
          Asunto,
          TextoMensaje,
          FechaHoraRegistro)
     VALUES
          (@vNombreCompleto,
           @vEmail,
          'Informe de inasistencia al aula', CONCAT_WS(' ',
          'Favor de justificar inasistencia el', CONVERT(VARCHAR,@vFechaFalta,106),
          'a las',CONVERT(VARCHAR(15),@vHoraFalta,100),
          'o procederá descuento'),GETDATE())
END
```

Después de cada *insert* en la tabla ***MaestrosInasist*** se ejecutará el *script* completo agrupado en el bloque BEGIN-END.

Para acceder a la tupla recién insertada se debe hacer referencia, a una tabla temporal llamada INSERTED. Cuando el trigger se activa a consecuencia de un update, las tablas temporales INSERTED y DELETED contienen la tupla nueva y original respectivamente.

Este código indica que deseamos que el mes de la fecha se muestre con letra (May, Sep, etc).

Este código hace que se muestre la hora en formato de 12 horas (AM/PM)

4.2 Disparadores (Triggers)

Añada al menos la siguiente tupla:

```
INSERT INTO
   MaestrosInasist
   (idMaestro, idGrupoHorario, Fecha, FaltaJustificada)
VALUES
   (1,76,GETDATE(),0)
```

Haga las siguientes consultas:

```
SELECT * FROM MaestrosInasist

SELECT * FROM BitacoraCorreo
```

4.2 Disparadores (Triggers)

Se asume que algún otro proceso seleccionara los mensajes de correo de la tabla **BitacoraCorreo** que aun no han sido enviados para hacerlo.

4.2 Disparadores (Triggers)

Ejercicio

Crear otro *trigger* para que en caso de que la inasistencia de un profesor cambie a **justificada**, se registre un mensaje en la bitácora, en el que se le comunica al profesor que su inasistencia se ha justificado y no se le realizará descuento alguno.

4.2 Disparadores (Triggers)

Para asegurarse de que el nuevo *trigger* realice su función correctamente, haga algún cambio a otro de los atributos (por ejemplo, la fecha).

Si es necesario, haga las correcciones correspondientes para que se registre en la bitácora el mensaje relativo a la justificación o a que se realizó un cambio a la fecha, según el caso.