

**Bases de Datos**

**Trabajo Práctico – 2da Parte**

**Asignatura:** Bases de Datos

**Profesores:** Seijas, Leticia

Genin, Fernando

**Integrantes:** Nucci, Manuel

Pico, Juan Fernando

Vilchez, Sol

**Fecha de entrega:** 29 de noviembre de 2018

**Trabajo Práctico – 2da Parte**

**Tablas**

**Acceso**

CREATE TABLE [dbo].[acceso](

[id\_empleado] [int] NOT NULL,

[id\_franja] [int] NOT NULL,

[num\_area] [int] NOT NULL,

CONSTRAINT [PK\_acceso] PRIMARY KEY CLUSTERED

(

[id\_empleado] ASC,

[id\_franja] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Area**

CREATE TABLE [dbo].[area](

[num\_area] [int] IDENTITY(1,1) NOT NULL,

[nombre] [varchar](50) NOT NULL,

[id\_nivel\_seg] [int] NOT NULL,

CONSTRAINT [PK\_area] PRIMARY KEY CLUSTERED

(

[num\_area] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Auditoria**

CREATE TABLE [dbo].[auditoria](

[id\_trabajo] [int] NOT NULL,

[num\_auditoria] [int] IDENTITY(1,1) NOT NULL,

[fecha\_hora] [smalldatetime] NOT NULL,

[resultado] [varchar](50) NOT NULL,

CONSTRAINT [PK\_auditoria] PRIMARY KEY CLUSTERED

(

[id\_trabajo] ASC,

[num\_auditoria] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Contratado en**

CREATE TABLE [dbo].[contratado\_en](

[id\_empleado] [int] NOT NULL,

[id\_trabajo] [int] NOT NULL,

[num\_area] [int] NOT NULL,

[inicio\_contrato] [date] NOT NULL,

[fin\_contrato] [date] NOT NULL,

CONSTRAINT [PK\_contratado\_en] PRIMARY KEY CLUSTERED

(

[id\_empleado] ASC,

[id\_trabajo] ASC,

[inicio\_contrato] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Datos confidenciales**

CREATE TABLE [dbo].[datos\_confidenciales](

[id\_datos\_confidenciales] [int] IDENTITY(1,1) NOT NULL,

[contrasena] [char](32) NOT NULL,

[huella\_dactilar] [char](32) NOT NULL,

CONSTRAINT [PK\_datos\_confidenciale] PRIMARY KEY CLUSTERED

(

[id\_datos\_confidenciales] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Empleado**

CREATE TABLE [dbo].[empleado](

[id\_empleado] [int] IDENTITY(1,1) NOT NULL,

[nombre] [varchar](50) NOT NULL,

[apellido] [varchar](50) NOT NULL,

[tipo\_doc] [char](3) NOT NULL,

[documento] [int] NOT NULL,

[e\_mail] [varchar](50) NOT NULL,

[telefono] [varchar](15) NOT NULL,

[tipo] [varchar](25) NOT NULL,

[id\_nivel\_seg] [int] NOT NULL,

[id\_datos\_confidenciales] [int] NOT NULL,

CONSTRAINT [PK\_empleado] PRIMARY KEY CLUSTERED

(

[id\_empleado] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Empleado jerárquico**

CREATE TABLE [dbo].[empleado\_jerarquico](

[id\_empleado] [int] NOT NULL,

[num\_area] [int] NOT NULL,

[fecha\_asignacion] [date] NOT NULL,

CONSTRAINT [PK\_empleado\_jerarquico] PRIMARY KEY CLUSTERED

(

[id\_empleado] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Empleado no profesional**

CREATE TABLE [dbo].[empleado\_no\_profesional](

[id\_empleado] [int] NOT NULL,

CONSTRAINT [PK\_empleado\_no\_profesional] PRIMARY KEY CLUSTERED

(

[id\_empleado] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Empleado profesional contratado**

CREATE TABLE [dbo].[empleado\_prof\_contratado](

[id\_empleado] [int] NOT NULL,

CONSTRAINT [PK\_empleado\_prof\_contratado] PRIMARY KEY CLUSTERED

(

[id\_empleado] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Empleado profesional permanente**

CREATE TABLE [dbo].[empleado\_prof\_permanente](

[id\_empleado] [int] NOT NULL,

[num\_area] [int] NOT NULL,

CONSTRAINT [PK\_empleado\_planta\_permanente] PRIMARY KEY CLUSTERED

(

[id\_empleado] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Empleado profesional**

CREATE TABLE [dbo].[empleado\_profesional](

[id\_empleado] [int] NOT NULL,

[tipo] [varchar](25) NOT NULL,

CONSTRAINT [PK\_empleado\_profesional] PRIMARY KEY CLUSTERED

(

[id\_empleado] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Evento**

CREATE TABLE [dbo].[evento](

[num\_area] [int] NOT NULL,

[num\_evento] [int] IDENTITY(1,1) NOT NULL,

[fecha\_hora] [smalldatetime] NOT NULL,

[descripcion] [varchar](150) NOT NULL,

CONSTRAINT [PK\_evento\_1] PRIMARY KEY CLUSTERED

(

[num\_area] ASC,

[num\_evento] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Franja horaria**

CREATE TABLE [dbo].[franja\_horaria](

[id\_franja] [int] IDENTITY(1,1) NOT NULL,

[horario\_inicio] [time](0) NOT NULL,

[horario\_fin] [time](0) NOT NULL,

CONSTRAINT [PK\_franja\_horaria] PRIMARY KEY CLUSTERED

(

[id\_franja] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Nivel seguridad**

CREATE TABLE [dbo].[nivel\_seguridad](

[id\_nivel\_seg] [int] IDENTITY(1,1) NOT NULL,

[nombre] [varchar](25) NOT NULL,

[categoria] [varchar](20) NOT NULL,

[descripcion] [varchar](100) NOT NULL,

CONSTRAINT [PK\_nivel\_seguridad] PRIMARY KEY CLUSTERED

(

[id\_nivel\_seg] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Registro**

CREATE TABLE [dbo].[registro](

[id\_empleado] [int] NOT NULL,

[num\_area] [int] NOT NULL,

[num\_registro] [int] IDENTITY(1,1) NOT NULL,

[accion] [varchar](15) NOT NULL,

[fecha\_hora] [smalldatetime] NOT NULL,

[autorizado] [char](2) NOT NULL,

CONSTRAINT [PK\_registro] PRIMARY KEY CLUSTERED

(

[id\_empleado] ASC,

[num\_area] ASC,

[num\_registro] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Trabajo**

CREATE TABLE [dbo].[trabajo](

[id\_trabajo] [int] IDENTITY(1,1) NOT NULL,

[descripcion] [varchar](100) NOT NULL,

CONSTRAINT [PK\_trabajo] PRIMARY KEY CLUSTERED

(

[id\_trabajo] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**Funciones**

CREATE FUNCTION validador

(@id\_empleado INT,

@num\_area INT,

@tipo\_empleado INT) -- 1 Emp Jerarq, 2 EPPermanente, 3 EPContratado, 4 EmpNoProf

RETURNS INT

AS

BEGIN

DECLARE

@id\_nivel\_seg\_emp INT,

@nombre\_id\_nivel\_seg\_emp VARCHAR(15),

@id\_nivel\_seg\_area INT,

@nombre\_id\_nivel\_seg\_area VARCHAR(15),

@nReturn INT;

SELECT @id\_nivel\_seg\_emp = E.id\_nivel\_seg, @nombre\_id\_nivel\_seg\_emp = NS.nombre

FROM empleado E

INNER JOIN nivel\_seguridad NS ON E.id\_nivel\_seg = NS.id\_nivel\_seg

WHERE E.id\_empleado = @id\_empleado;

SELECT @id\_nivel\_seg\_area = A.id\_nivel\_seg, @nombre\_id\_nivel\_seg\_area = NS.nombre

FROM area A

INNER JOIN nivel\_seguridad NS ON A.id\_nivel\_seg = NS.id\_nivel\_seg

WHERE A.num\_area = @num\_area;

IF @id\_nivel\_seg\_emp = @id\_nivel\_seg\_area OR

(@tipo\_empleado <> 4 AND

(@nombre\_id\_nivel\_seg\_emp = 'Alto' OR

(@nombre\_id\_nivel\_seg\_emp = 'Medio' AND @nombre\_id\_nivel\_seg\_area = 'Bajo')))

SET @nReturn = 1;

ELSE

SET @nReturn = 0;

RETURN @nReturn;

END;

GO

**Triggers**

**Acceso – Empleado no profesional – Insert**

CREATE OR ALTER TRIGGER validar\_area\_emp\_no\_prof\_insert

ON dbo.acceso

INSTEAD OF INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE

@id\_empleado INT,

@id\_franja INT,

@num\_area INT;

DECLARE cur CURSOR FOR

SELECT id\_empleado, id\_franja, num\_area

FROM inserted

OPEN cur;

FETCH NEXT FROM cur INTO @id\_empleado, @id\_franja, @num\_area;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF dbo.validador(@id\_empleado, @num\_area, 4) = 1

BEGIN

INSERT INTO [dbo].[acceso]

([id\_empleado]

,[id\_franja]

,[num\_area])

VALUES

(@id\_empleado

,@id\_franja

,@num\_area);

END;

ELSE

BEGIN

PRINT 'El registro del empleado con id = ' + CAST(@id\_empleado AS VARCHAR) + ' que lo

vincula con una franja horaria y un área no pudo ser insertado por ser inválida

el área.';

END;

FETCH NEXT FROM cur INTO @id\_empleado, @id\_franja, @num\_area;

END;

CLOSE cur;

DEALLOCATE cur;

END;

GO

**Acceso – Empleado no profesional – Update**

CREATE OR ALTER TRIGGER validar\_area\_emp\_no\_prof\_update

ON dbo.acceso

INSTEAD OF UPDATE

AS

BEGIN

SET NOCOUNT ON;

DECLARE

@id\_empleado INT,

@id\_franja INT,

@num\_area INT;

DECLARE cur CURSOR FOR

SELECT id\_empleado, id\_franja, num\_area

FROM inserted

OPEN cur;

FETCH NEXT FROM cur INTO @id\_empleado, @id\_franja, @num\_area;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF dbo.validador(@id\_empleado, @num\_area, 4) = 1

BEGIN

UPDATE [dbo].[acceso]

SET [id\_empleado] = @id\_empleado

,[id\_franja] = @id\_franja

,[num\_area] = @num\_area

WHERE id\_empleado = @id\_empleado AND id\_franja = @id\_franja

END;

ELSE

BEGIN

PRINT 'El registro del empleado con id = ' + CAST(@id\_empleado AS VARCHAR) + ' que lo

vincula con una franja horaria y un área no pudo ser modificado por ser inválida

el área.';

END;

FETCH NEXT FROM cur INTO @id\_empleado, @id\_franja, @num\_area;

END;

CLOSE cur;

DEALLOCATE cur;

END;

GO

**Contratado en – Empleado profesional contratado – Insert**

CREATE OR ALTER TRIGGER validar\_area\_emp\_prof\_contr\_insert

ON dbo.contratado\_en

INSTEAD OF INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE

@id\_empleado INT,

@id\_trabajo INT,

@num\_area INT,

@inicio\_contrato DATE,

@fin\_contrato DATE,

@cond\_trabajo INT;

DECLARE cur CURSOR FOR

SELECT id\_empleado, id\_trabajo, num\_area, inicio\_contrato, fin\_contrato

FROM inserted

OPEN cur;

FETCH NEXT FROM cur INTO @id\_empleado, @id\_trabajo, @num\_area, @inicio\_contrato, @fin\_contrato;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF dbo.validador(@id\_empleado, @num\_area, 3) = 1 AND DATEDIFF(day, @inicio\_contrato, @fin\_contrato) > 0

BEGIN

SELECT cond\_trabajo = COUNT(\*)

FROM contratado\_en

WHERE id\_trabajo = @id\_trabajo

GROUP BY id\_trabajo, num\_area, inicio\_contrato

IF @cond\_trabajo = 1 -- Los trabajos para un grupo de empleados son exclusivos por área y contrato

BEGIN

INSERT INTO [dbo].[contratado\_en]

([id\_empleado]

,[id\_trabajo]

,[num\_area]

,[inicio\_contrato]

,[fin\_contrato])

VALUES

(@id\_empleado

,@id\_trabajo

,@num\_area

,@inicio\_contrato

,@fin\_contrato);

END;

ELSE

BEGIN

PRINT 'Se quiso insertar más de un trabajo para un mismo empleado, área y contrato.'

END;

END;

ELSE

BEGIN

PRINT 'El registro del empleado con id = ' + CAST(@id\_empleado AS VARCHAR) +

' que lo vincula con un trabajo y un área no pudo ser insertado por ser inválida

el área.';

END;

FETCH NEXT FROM cur INTO @id\_empleado, @id\_trabajo, @num\_area, @inicio\_contrato, @fin\_contrato;

END;

CLOSE cur;

DEALLOCATE cur;

END;

GO

**Contratado en – Empleado profesional contratado – Update**

CREATE OR ALTER TRIGGER validar\_area\_emp\_prof\_contr\_update

ON dbo.contratado\_en

INSTEAD OF UPDATE

AS

BEGIN

SET NOCOUNT ON;

DECLARE

@id\_empleado INT,

@id\_trabajo INT,

@num\_area INT,

@inicio\_contrato DATE,

@fin\_contrato DATE;

DECLARE cur CURSOR FOR

SELECT id\_empleado, id\_trabajo, num\_area, inicio\_contrato, fin\_contrato

FROM inserted

OPEN cur;

FETCH NEXT FROM cur INTO @id\_empleado, @id\_trabajo, @num\_area, @inicio\_contrato, @fin\_contrato;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF dbo.validador(@id\_empleado, @num\_area, 3) = 1 AND DATEDIFF(day, @inicio\_contrato, @fin\_contrato) > 0

BEGIN

UPDATE [dbo].[contratado\_en]

SET [id\_empleado] = @id\_empleado

,[id\_trabajo] = @id\_trabajo

,[num\_area] = @num\_area

,[inicio\_contrato] = @inicio\_contrato

,[fin\_contrato] = @fin\_contrato

WHERE id\_empleado = @id\_empleado AND id\_trabajo = @id\_trabajo AND inicio\_contrato = @inicio\_contrato

END;

ELSE

BEGIN

PRINT 'El registro del empleado con id = ' + CAST(@id\_empleado AS VARCHAR) +

' que lo vincula con un trabajo y un área no pudo ser modificado por ser inválida

el área.';

END;

FETCH NEXT FROM cur INTO @id\_empleado, @id\_trabajo, @num\_area, @inicio\_contrato, @fin\_contrato;

END;

CLOSE cur;

DEALLOCATE cur;

END;

GO

**Empleado jerárquico – Insert**

CREATE OR ALTER TRIGGER validar\_area\_emp\_jerarq\_insert

ON dbo.empleado\_jerarquico

INSTEAD OF INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE

@id\_empleado INT,

@num\_area INT,

@fecha\_asignacion DATE;

DECLARE cur CURSOR FOR

SELECT id\_empleado, num\_area, fecha\_asignacion

FROM inserted

OPEN cur;

FETCH NEXT FROM cur INTO @id\_empleado, @num\_area, @fecha\_asignacion;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF dbo.validador(@id\_empleado, @num\_area, 1) = 1

BEGIN

INSERT INTO [dbo].[empleado\_jerarquico]

([id\_empleado]

,[num\_area]

,[fecha\_asignacion])

VALUES

(@id\_empleado

,@num\_area

,@fecha\_asignacion);

END;

ELSE

BEGIN

PRINT 'El registro del empleado con id = ' + CAST(@id\_empleado AS VARCHAR) +

' no ha podido ser insertado.';

END;

FETCH NEXT FROM cur INTO @id\_empleado, @num\_area, @fecha\_asignacion;

END;

CLOSE cur;

DEALLOCATE cur;

END;

GO

**Empleado jerárquico – Update**

CREATE OR ALTER TRIGGER validar\_area\_emp\_jerarq\_update

ON dbo.empleado\_jerarquico

INSTEAD OF UPDATE

AS

BEGIN

SET NOCOUNT ON;

DECLARE

@id\_empleado INT,

@num\_area INT,

@fecha\_asignacion DATE;

DECLARE cur CURSOR FOR

SELECT id\_empleado, num\_area, fecha\_asignacion

FROM inserted

OPEN cur;

FETCH NEXT FROM cur INTO @id\_empleado, @num\_area, @fecha\_asignacion;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF dbo.validador(@id\_empleado, @num\_area, 1) = 1

BEGIN

UPDATE [dbo].[empleado\_jerarquico]

SET [id\_empleado] = @id\_empleado

,[num\_area] = @num\_area

,[fecha\_asignacion] = @fecha\_asignacion

WHERE id\_empleado = @id\_empleado

END;

ELSE

BEGIN

PRINT 'El registro del empleado con id = ' + CAST(@id\_empleado AS VARCHAR) +

' no ha podido ser modificado.';

END;

FETCH NEXT FROM cur INTO @id\_empleado, @num\_area, @fecha\_asignacion;

END;

CLOSE cur;

DEALLOCATE cur;

END;

GO

**Empleado profesional permanente – Insert**

CREATE OR ALTER TRIGGER validar\_area\_emp\_prof\_perm\_insert

ON dbo.empleado\_prof\_permanente

INSTEAD OF INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE

@id\_empleado INT,

@num\_area INT;

DECLARE cur CURSOR FOR

SELECT id\_empleado, num\_area

FROM inserted

OPEN cur;

FETCH NEXT FROM cur INTO @id\_empleado, @num\_area;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF dbo.validador(@id\_empleado, @num\_area, 2) = 1

BEGIN

INSERT INTO [dbo].[empleado\_prof\_permanente]

([id\_empleado]

,[num\_area])

VALUES

(@id\_empleado

,@num\_area);

END;

ELSE

BEGIN

PRINT 'El registro del empleado con id = ' + CAST(@id\_empleado AS VARCHAR) +

' no ha podido ser insertado.';

END;

FETCH NEXT FROM cur INTO @id\_empleado, @num\_area;

END;

CLOSE cur;

DEALLOCATE cur;

END;

GO

**Empleado profesional permanente – Update**

CREATE OR ALTER TRIGGER validar\_area\_emp\_prof\_perm\_update

ON dbo.empleado\_prof\_permanente

INSTEAD OF UPDATE

AS

BEGIN

SET NOCOUNT ON;

DECLARE

@id\_empleado INT,

@num\_area INT;

DECLARE cur CURSOR FOR

SELECT id\_empleado, num\_area

FROM inserted

OPEN cur;

FETCH NEXT FROM cur INTO @id\_empleado, @num\_area;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF dbo.validador(@id\_empleado, @num\_area, 2) = 1

BEGIN

UPDATE [dbo].[empleado\_prof\_permanente]

SET [id\_empleado] = @id\_empleado

,[num\_area] = @num\_area

WHERE id\_empleado = @id\_empleado

END;

ELSE

BEGIN

PRINT 'El registro del empleado con id = ' + CAST(@id\_empleado AS VARCHAR) +

' no ha podido ser modificado.';

END;

FETCH NEXT FROM cur INTO @id\_empleado, @num\_area;

END;

CLOSE cur;

DEALLOCATE cur;

END;

GO

**Franja horaria – Insert**

CREATE OR ALTER TRIGGER validar\_franja\_horaria\_insert

ON dbo.franja\_horaria

INSTEAD OF INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE

@id\_franja INT,

@horario\_inicio TIME,

@horario\_fin TIME;

DECLARE cur CURSOR FOR

SELECT id\_franja, horario\_inicio, horario\_fin

FROM inserted

OPEN cur;

FETCH NEXT FROM cur INTO @id\_franja, @horario\_inicio, @horario\_fin;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF @horario\_inicio < @horario\_fin

BEGIN

INSERT INTO [dbo].[franja\_horaria]

([horario\_inicio]

,[horario\_fin])

VALUES

(@horario\_inicio

,@horario\_fin);

END;

ELSE

BEGIN

PRINT 'No se ha podido insertar el registro de la franja horaria.';

END;

FETCH NEXT FROM cur INTO @id\_franja, @horario\_inicio, @horario\_fin;

END;

CLOSE cur;

DEALLOCATE cur;

END;

GO

**Franja horaria – Update**

CREATE OR ALTER TRIGGER validar\_franja\_horaria\_update

ON dbo.franja\_horaria

INSTEAD OF UPDATE

AS

BEGIN

SET NOCOUNT ON;

DECLARE

@id\_franja INT,

@horario\_inicio TIME,

@horario\_fin TIME;

DECLARE cur CURSOR FOR

SELECT id\_franja, horario\_inicio, horario\_fin

FROM inserted

OPEN cur;

FETCH NEXT FROM cur INTO @id\_franja, @horario\_inicio, @horario\_fin;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF @horario\_inicio < @horario\_fin

BEGIN

UPDATE [dbo].[franja\_horaria]

SET [horario\_inicio] = @horario\_inicio

,[horario\_fin] = @horario\_fin

WHERE id\_franja = @id\_franja

END;

ELSE

BEGIN

PRINT 'No se ha podido insertar el registro de la franja horaria.';

END;

FETCH NEXT FROM cur INTO @id\_franja, @horario\_inicio, @horario\_fin;

END;

CLOSE cur;

DEALLOCATE cur;

END;

GO

**Registro – Insert**

CREATE OR ALTER TRIGGER validar\_ingreso\_egreso\_area\_insert

ON dbo.registro

INSTEAD OF INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE

@id\_empleado INT,

@num\_area INT,

@num\_registro INT,

@accion VARCHAR(15),

@fecha\_hora SMALLDATETIME,

@autorizado CHAR(2),

@ultima\_accion VARCHAR(15),

@condicion CHAR(2),

@categoria VARCHAR(20);

SET @condicion = 'No';

DECLARE cur CURSOR FOR

SELECT id\_empleado, num\_area, num\_registro, accion, fecha\_hora, autorizado

FROM inserted

OPEN cur;

FETCH NEXT FROM cur INTO @id\_empleado, @num\_area, @num\_registro, @accion, @fecha\_hora, @autorizado;

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT @categoria = categoria

FROM area

INNER JOIN nivel\_seguridad NS ON area.id\_nivel\_seg = NS.id\_nivel\_seg

WHERE area.num\_area = @num\_area;

IF @categoria = 'Restringido'

BEGIN

-- Buscar la última acción del empleado en esa área, sea del día actual o un día anterior

SELECT @ultima\_accion = accion, @condicion = autorizado

FROM registro R1

WHERE id\_empleado = @id\_empleado AND

num\_area = @num\_area AND

R1.fecha\_hora = (SELECT MAX(R2.fecha\_hora)

FROM registro R2

WHERE R2.id\_empleado = @id\_empleado AND

R2.num\_area = @num\_area);

IF (@accion = @ultima\_accion AND @condicion = 'No') -- El empleado quiere volver a realizar la misma acción luego de un intento fallido

OR

(@accion <> @ultima\_accion AND @condicion = 'Si') -- El empleado quiere realizar la acción opuesta a lo último registrado luego de un éxito previo

BEGIN

SET @autorizado = 'Si';

END;

ELSE

BEGIN

SET @autorizado = 'No';

PRINT CAST(@accion AS VARCHAR) + ' no autorizado.';

END;

INSERT INTO [dbo].[registro]

([id\_empleado]

,[num\_area]

,[accion]

,[fecha\_hora]

,[autorizado])

VALUES

(@id\_empleado

,@num\_area

,@accion

,@fecha\_hora

,@autorizado);

END;

ELSE

BEGIN

PRINT 'El área que se intentó insertar no es de acceso restringido.'

END;

FETCH NEXT FROM cur INTO @id\_empleado, @num\_area, @num\_registro, @accion, @fecha\_hora, @autorizado;

END;

CLOSE cur;

DEALLOCATE cur;

END;

GO

**Stored Procedures y Views**

**Funcionalidad 1**

CREATE OR ALTER PROCEDURE consulta\_1

AS

SELECT empleado.nombre, empleado.apellido, empleado.id\_empleado

FROM empleado

WHERE NOT EXISTS (SELECT \*

FROM area

WHERE NOT EXISTS (SELECT \*

FROM acceso

WHERE acceso.id\_empleado = empleado.id\_empleado AND

acceso.num\_area = area.num\_area));

**Funcionalidad 2**

CREATE OR ALTER VIEW intentos\_fallidos

AS

SELECT empleado.id\_empleado,

empleado.nombre,

empleado.apellido,

area.num\_area,

area.nombre AS 'nombre\_area'

FROM empleado INNER JOIN registro R1 ON empleado.id\_empleado = R1.id\_empleado

INNER JOIN area ON R1.num\_area = area.num\_area

WHERE CONVERT(date, R1.fecha\_hora, 101) = CONVERT(date, GETDATE(), 101)

AND R1.autorizado = 'No'

AND R1.accion = 'Ingreso'

AND CONVERT(time(0), R1.fecha\_hora) = (SELECT MAX(CONVERT(time(0), R2.fecha\_hora))

FROM registro R2

WHERE R2.id\_empleado = empleado.id\_empleado

AND R2.accion = 'Ingreso'

AND CONVERT(date, R2.fecha\_hora, 101) = CONVERT(date, GETDATE(), 101));

**Funcionalidad 3**

CREATE OR ALTER PROCEDURE consulta\_3

AS

SELECT empleado.id\_empleado, empleado.nombre, empleado.apellido, empleado.documento

FROM empleado

INNER JOIN registro ON empleado.id\_empleado = registro.id\_empleado

INNER JOIN area ON registro.num\_area = area.num\_area

WHERE DATEDIFF(DAY, CONVERT(date, registro.fecha\_hora, 101), CONVERT(date, GETDATE(), 101)) <= 30

AND registro.autorizado = 'No'

GROUP BY empleado.id\_empleado, empleado.nombre, empleado.apellido, empleado.documento

HAVING COUNT(\*) > 5

UNION

SELECT empleado.id\_empleado, empleado.nombre, empleado.apellido, empleado.documento

FROM empleado

INNER JOIN registro ON empleado.id\_empleado = registro.id\_empleado

INNER JOIN area ON registro.num\_area = area.num\_area

INNER JOIN nivel\_seguridad NSE ON empleado.id\_nivel\_seg = NSE.id\_nivel\_seg

INNER JOIN nivel\_seguridad NSA ON area.id\_nivel\_seg = NSA.id\_nivel\_seg

WHERE DATEDIFF(DAY, CONVERT(date, registro.fecha\_hora, 101), CONVERT(date, GETDATE(), 101)) <= 30

AND registro.autorizado = 'No'

AND ((NSE.nombre = 'Bajo' AND (NSA.nombre = 'Medio' OR NSA.nombre = 'Alto'))

OR (NSE.nombre = 'Medio' AND NSA.nombre = 'Alto'));