UNIVERSIDAD NACIONAL
de MAR DEL PLATA

FACULTAD DE
INGENIERIA

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'));
```