128 - Procedimientos almacenados (insertar)

Primer problema:

Un profesor guarda en una tabla llamada "alumnos" el nombre de los alumnos y su nota.

1- Elimine la tabla si existe y créela:

```
 if object_id('alumnos') is not null
  drop table alumnos;

 create table alumnos(
  documento char(8),
  nombre varchar(40),
  nota decimal(4,2),
  primary key(documento)
 );
```

2- Ingrese algunos registros:

```
 insert into alumnos values ('22222222','Pedro Lopez',5);
 insert into alumnos values ('23333333','Ana Lopez',4);
 insert into alumnos values ('24444444','Maria Juarez',8);
 insert into alumnos values ('25555555','Juan Garcia',5.6);
 insert into alumnos values ('26666666','Karina Torres',2);
 insert into alumnos values ('27777777','Nora Torres',7.5);
 insert into alumnos values ('28888888','Mariano Herrero',3.5);
```

3- Elimine la tabla "aprobados" si existe y créela con los mismos campos de la tabla "alumnos":

```
 if object_id('aprobados') is not null
  drop table aprobados;

 create table aprobados(
  documento char(8),
  nombre varchar(40),
  nota decimal(4,2)
 );
```

4- Elimine la tabla "desaprobados" si existe y créela con los siguientes campos:

```
 if object_id('desaprobados') is not null
  drop table desaprobados;

 create table desaprobados(
  documento char(8),
  nombre varchar(40)
 );
```

5- Elimine el procedimiento llamado "pa_aprobados", si existe:

```
 if object_id('pa_aprobados') is not null
  drop procedure pa_aprobados;
```

6- Cree el procedimiento para que seleccione todos los datos de los alumnos cuya nota es igual o
superior a 4.

7- Ingrese en la tabla "aprobados" el resultado devuelto por el procedimiento almacenado "pa_aprobados".

8- Vea el contenido de "aprobados":
```
 select *from aprobados;
```

9- Elimine el procedimiento llamado "pa_desaprobados", si existe:
```
 if object_id('pa_desaprobados') is not null
  drop procedure pa_desaprobados;
```

10- Cree el procedimiento para que seleccione el documento y nombre de los alumnos cuya nota es
menor a 4.

11- Ingrese en la tabla "desaprobados" el resultado devuelto por el procedimiento almacenado
"pa_desaprobados".

12- Vea el contenido de "desaprobados":
```
 select *from desaprobados;
```

131 - Procedimientos Almacenados (con join)

Primer problema:

Una librería almacena los datos de los libros en una tabla denominada "libros" y en una tabla
"ventas" las ventas de los mismos.

1- Elimine las tablas si existen y créelas:
```
 if (object_id('libros')) is not null
  drop table libros;
 if (object_id('ventas')) is not null
  drop table ventas;

 create table libros(
  codigo int identity,
  titulo varchar(40),
  autor varchar(30),
  editorial varchar(20),
  precio decimal(6,2),
  cantidad int,
  primary key (codigo)
 );

 create table ventas(
  numero int identity,
  codigo int not null,
  preciounitario decimal(6,2),
  cantidad int,
  constraint PK_ventas primary key (numero),
  constraint FK_ventas_codigolibro
```

```
  foreign key (codigo)
  references libros(codigo)
  on update cascade
);
```

2- Ingrese algunos registros para ambas:

```
insert into libros values('Uno','Richard Bach','Planeta',15,100);
insert into libros values('Ilusiones','Richard Bach','Planeta',18,150);
insert into libros values('El aleph','Borges','Emece',25,200);
insert into libros values('Matematica estas ahi','Paenza','Nuevo siglo',20,300);
insert into libros values('Aprenda PHP','Mario Molina','Nuevo siglo',45,200);

insert into ventas values(1,15,1);
insert into ventas values(2,18,1);
insert into ventas values(3,25,100);
insert into ventas values(1,15,50);
```

3- Elimine el procedimiento "pa_ventas", si existe:

```
if (object_id('pa_ventas')) is not null
 drop proc pa_ventas;
```

4- Cree un procedimiento que muestre los datos de las ventas (número, título, autor y editorial del
libro vendido, precio unitario, cantidad vendida, total por item)

5- Ejecute el procedimiento:

```
pa_ventas;
```

6- Elimine el procedimiento "pa_vender", si existe:

```
if (object_id('pa_vender')) is not null
 drop proc pa_vender;
```

7- Cree un procedimiento que permita ingresar una venta en "ventas" con los siguientes datos: código
del libro y cantidad que se vende. El procedimiento debe controlar que haya libros disponibles (es
decir, que la cantidad que se vende sea mayor o igual a la cantidad existente del libro) y luego
restar la cantidad vendida de la tabla "libros":

```
create procedure pa_vender
 @codigo int=null,
 @cantidad int=1
 as
  --verificamos que el código exista
  if not exists (select *from libros where codigo=@codigo) or (@codigo is null)
    select 'Ingrese un codigo de libro válido'
  else
  begin --verificamos que haya stock
    declare @disponibles int
    select @disponibles= cantidad from libros where codigo=@codigo
    if (@disponibles<@cantidad)
      select 'Solo hay '+cast(@disponibles as varchar(10))+' disponibles'
    else
```

```
    begin
      declare @precio decimal(6,2)
      select @precio= precio from libros where codigo=@codigo
      insert into ventas values(@codigo,@precio,@cantidad)
      update libros set cantidad=cantidad-@cantidad where @codigo=codigo
    end
  end;
```

Este procedimiento recibe parámetros, declara variables locales y modifica 2 tablas, en una de ellas
realiza una inserción y en la otra una actualización.

8- Ejecute el procedimiento "pa_vender".

9- Vea si las tablas se modificaron:

```
select *from ventas;
select *from libros;
```

10- Envíe al procedimiento "pa_vender" un código de libro inexistente.

11- Envíe una cantidad que supere el stock.

135 - Funciones escalares (crear y llamar)

Primer problema:

Una clínica almacena los turnos para los distintos médicos en una tabla llamada "consultas" y en
otra tabla "medicos" los datos de los médicos.

1- Elimine las tablas si existen:

```
if object_id('consultas') is not null
  drop table consultas;
if object_id('medicos') is not null
  drop table medicos;
```

2- Cree las tablas con las siguientes estructuras:

```
create table medicos (
  documento char(8) not null,
  nombre varchar(30),
  constraint PK_medicos
    primary key clustered (documento)
);

create table consultas(
  fecha datetime,
  medico char(8) not null,
  paciente varchar(30),
  constraint PK_consultas
    primary key (fecha,medico),
  constraint FK_consultas_medico
    foreign key (medico)
    references medicos(documento)
```

```
   on update cascade
   on delete cascade
 );
```

3- Ingrese algunos registros:

```
 insert into medicos values('22222222','Alfredo Acosta');
 insert into medicos values('23333333','Pedro Perez');
 insert into medicos values('24444444','Marcela Morales');

 insert into consultas values('2007/03/26 8:00','22222222','Juan Juarez');
 insert into consultas values('2007/03/26 8:00','23333333','Gaston Gomez');
 insert into consultas values('2007/03/26 8:30','22222222','Nora Norte');
 insert into consultas values('2007/03/28 9:00','22222222','Juan Juarez');
 insert into consultas values('2007/03/29 8:00','24444444','Nora Norte');
 insert into consultas values('2007/03/24 8:30','22222222','Hector Huerta');
 insert into consultas values('2007/03/24 9:30','23333333','Hector Huerta');
```

4- Elimine la función "f_nombreDia" si existe:

```
 if object_id('f_nombreDia') is not null
  drop function f_nombreDia;
```

5- Cree la función "f_nombreDia" que recibe una fecha (tipo string) y nos retorne el nombre del día
en español.

6- Elimine la función "f_horario" si existe:

```
 if object_id('f_horario') is not null
  drop function f_horario;
```

7- Cree la función "f_horario" que recibe una fecha (tipo string) y nos retorne la hora y minutos.

8- Elimine la función "f_fecha" si existe:

```
 if object_id('f_fecha') is not null
  drop function f_fecha;
```

9- Cree la función "f_fecha" que recibe una fecha (tipo string) y nos retorne la fecha (sin hora ni
minutos)

10- Muestre todas las consultas del médico llamado 'Alfredo Acosta', incluyendo el día (emplee la
función "f_nombreDia", la fecha (emplee la función "f_fecha"), el horario (emplee la función
"f_horario") y el nombre del paciente.

11- Muestre todos los turnos para el día sábado, junto con la fecha, de todos los médicos.

12- Envíe a la función "f_nombreDia" una fecha y muestre el valor retornado:

```
 declare @valor char(30)
 set @valor='2007/04/09'
 select dbo.f_nombreDia(@valor);
```

Segundo problema:

Una empresa almacena datos de sus empleados en una tabla denominada "empleados".

1- Elimine la tabla si existe y créela con la siguiente estructura:

```
 if object_id('empleados') is not null
  drop table empleados;
 create table empleados(
  documento char(8) not null,
  nombre varchar(30),
  fechanacimiento datetime,
  fechaingreso datetime,
  telefono char(12),
  mail varchar(50)
 );
```

2- Ingrese algunos registros:

```
  insert into empleados values('22222222', 'Ana Acosta', '1970/10/02', '1995/10/10', '4556677', 'anitaacosta@hotmail.com');
  insert into empleados values('25555555', 'Bernardo Bustos', '1973/01/15', '1999/02/15', '4789012', null);
  insert into empleados values('30000000', 'Carlos Caseros', '1980/5/25', '2001/05/05', null, null);
  insert into empleados values('32222222', 'Estela Esper', '1985/02/20', '2006/12/12', null, 'estelaesper@gmail.com');
```

3- Elimine la función "f_edad" si existe:

```
 if object_id('f_edad') is not null
  drop function f_edad;
```

4- Cree la función "f_edad" que reciba 2 fechas (de tipo datetime) y nos retorne un valor positivo
correspondiente a la diferencia entre ambas.
Recuerde que en las funciones definidas por el usuario no pueden incluir funciones no determinísticas (como getdate), por ello, debemos enviar la fecha actual.
Note que la función retorna un valor positivo (tinyint), en ella se valida que la primera fecha a la
cual se le resta la segunda fecha sea mayor. Si quisiéramos calcular la cantidad de años entre dos
fechas podríamos emplear la función del sistema "datediff" que retorna un int, esta función (ya
vista) retorna un valor negativo si la primera fecha es menor a la segunda fecha enviada. Pero
nosotros queremos la edad de una persona, así que siempre enviaremos como primera fecha una
posterior a la segunda.

5- Muestre los nombres de los empleados y la edad (calculada con la función anteriormente creada)

6- Muestre el nombre de los empleados y la edad (calculada con la función "f_edad") que tenían al
ingresar a la empresa y los años de servicio.

7- Llame a la función "f_edad" enviándole la fecha actual y su fecha de nacimiento y muestre el
valor retornado.

8- Intente invocar la función sin enviarle valores.
Mensaje de error.

9- Llame a la función para que tome el valor por defecto del segundo argumento.

10- Elimine la función "f_valorNulo" si existe:
```
  if object_id('f_valorNulo') is not null
    drop function f_valorNulo;
```

11- Cree una función para reemplazar un valor "null" por el texto "No tiene".

12- Muestre todos los empleados, empleando la función "f_valorNulo" enviándole como argumento los
campos "mail" y "telefono".

136 - Funciones de tabla de varias instrucciones

Primer problema:
Una empresa almacena los datos de sus empleados en una tabla denominada "empleados".
1- Elimine la tabvla si existe:
```
 if object_id('empleados') is not null
  drop table empleados;
```

2- Cree la tabla con la siguiente estructura:
```
 create table empleados(
  documento char(8) not null,
  apellido varchar(30) not null,
  nombre varchar(30) not null,
  domicilio varchar(30),
  ciudad varchar(30),
  fechanacimiento datetime,
  constraint PK_empleados
   primary key(documento)
 );
```

3- Ingrese algunos registros:
```
 insert into empleados values('22222222','Acosta','Ana','Avellaneda
123','Cordoba','1970/10/10');
 insert into empleados values('23333333','Bustos','Bernardo','Bulnes
234','Cordoba','1972/05/15');
 insert into empleados values('24444444','Caseros','Carlos','Colon 356','Carlos
Paz','1980/02/25');
 insert into empleados values('25555555','Fuentes','Fabiola','Fragueiro 987','Jesus
Maria','1984/06/12');
```

4- Elimine la función "f_empleados" si existe:

```
 if object_id('f_empleados') is not null
  drop function f_empleados;
```

5- Cree una función que reciba como parámetro el texto "total" o "parcial" y muestre, en el primer
caso, todos los datos de los empleados y en el segundo caso (si recibe el valor "parcial"): el
documento, apellido, ciudad y año de nacimiento.

6- Llame a la función creada anteriormente enviándole "total".

7- Llame a la función anteriormente creada sin enviar argumento.
Mensaje de error.

8- Llame a la función enviándole una cadena vacía.

9- Ejecute la función "f_empleados" enviando "parcial" como argumento y recupere solamente los
registros cuyo domicilio es "Cordoba".

142 - Disparador de inserción (insert trigger)

Primer problema:
Una empresa almacena los datos de sus empleados en una tabla denominada "empleados" y en otra tabla
llamada "secciones", el código de la sección y el sueldo máximo de cada una de ellas.
1- Elimine las tablas si existen:

```
 if object_id('empleados') is not null
  drop table empleados;
 if object_id('secciones') is not null
  drop table secciones;
```

2- Cree las tablas, con las siguientes estructuras:

```
 create table secciones(
  codigo int identity,
  nombre varchar(30),
  sueldomaximo decimal(8,2),
  constraint PK_secciones primary key(codigo)
 );

 create table empleados(
  documento char(8) not null,
  nombre varchar(30) not null,
  domicilio varchar(30),
  codigoseccion int not null,
  sueldo decimal(8,2),
  constraint PK_empleados primary key(documento),
  constraint FK_empelados_seccion
   foreign key (codigoseccion) references secciones(codigo)
 );
```

3- Ingrese algunos registros en ambas tablas:

```
 insert into secciones values('Administracion',1500);
 insert into secciones values('Sistemas',2000);
 insert into secciones values('Secretaria',1000);

 insert into empleados values('22222222','Ana Acosta','Avellaneda 88',1,1100);
 insert into empleados values('23333333','Bernardo Bustos','Bulnes 345',1,1200);
 insert into empleados values('24444444','Carlos Caseres','Colon 674',2,1800);
 insert into empleados values('25555555','Diana Duarte','Colon 873',3,1000);
```

4- Cree un disparador para que se ejecute cada vez que una instrucción "insert" ingrese datos en "empleados"; el mismo debe verificar que el sueldo del empleado no sea mayor al sueldo máximo establecido para la sección, si lo es, debe mostrar un mensaje indicando tal situación y deshacer la transacción.

5- Ingrese un nuevo registro en "empleados" cuyo sueldo sea menor o igual al establecido para la sección.

6- Verifique que el disparador se ejecutó consultando la tabla "empleados":

```
 select *from empleados;
```

7- Intente ingresar un nuevo registro en "empleados" cuyo sueldo sea mayor al establecido para la sección.
El disparador se ejecutó mostrando un mensaje y la transacción se deshizo.

8- Verifique que el registro no se agregó en "empleados":

```
 select *from empleados;
```

9- Intente ingresar un empleado con código de sección inexistente.
Aparece un mensaje de error porque se viola la restricción "foreign key"; el trigger no llegó a ejecutarse.

143 - Disparador de borrado (delete trigger)

Primer problema:

Un comercio que vende artículos de informática almacena los datos de sus artículos en una tabla denominada "articulos".

1- Elimine la tabla si existe:

```
 if object_id('articulos') is not null
  drop table articulos;
```

2- Cree la tabla, con la siguiente estructura:

```
 create table articulos(
  codigo int identity,
```

```
  tipo varchar(30),
  descripcion varchar(40),
  precio decimal(8,2),
  stock int,
  constraint PK_articulos primary key (codigo)
 );
```

3- Ingrese algunos registros:

```
 insert into articulos values ('impresora','Epson Stylus C45',400,100);
 insert into articulos values ('impresora','Epson Stylus C85',500,200);
 insert into articulos values ('impresora','Epson Stylus Color 600',400,0);
 insert into articulos values ('monitor','Samsung 14',900,0);
 insert into articulos values ('monitor','Samsung 17',1200,0);
 insert into articulos values ('monitor','xxx 15',1500,0);
 insert into articulos values ('monitor','xxx 17',1600,0);
 insert into articulos values ('monitor','zzz 15',1300,0);
```

4- Cree un disparador para controlar que no se elimine un artículo si hay stock. El disparador se
activará cada vez que se ejecuta un "delete" sobre "articulos", controlando el stock, si se está
eliminando un artículo cuyo stock sea mayor a 0, el disparador debe retornar un mensaje de error y
deshacer la transacción.

5- Solicite la eliminación de un articulo que no tenga stock.
Se activa el disparador y permite la transacción.

6- Intente eliminar un artículo para el cual haya stock.
El trigger se dispara y deshace la transacción. Puede verificar que el artículo no fue eliminado
consultando la tabla "articulos".

7- Solicite la eliminación de varios artículos que no tengan stock.
Se activa el disparador y permite la transacción. Puede verificar que se borraron 2 artículos
consultando la tabla "articulos".

8- Intente eliminar varios artículos, algunos con stock y otros sin stock.
El trigger se dispara y deshace la transacción, es decir, ningún artículo fue eliminado, tampoco los
que tienen stock igual a 0.

9- Cree un trigger para evitar que se elimine más de 1 artículo.
Note que hay 2 disparadores para el mismo suceso (delete) sobre la misma tabla.

10- Solicite la eliminación de 1 artículo para el cual no haya stock.
Ambos disparadores "DIS_articulos_borrar" y "DIS_articulos_borrar2" se activan y permiten la
transacción.

11- Solicite la eliminación de 1 artículo que tenga stock.

El disparadores "DIS_articulos_borrar" se activa y no permite la transacción. El disparador
"DIS_articulos_borrar2" no llega a activarse.

12- Solicite la eliminación de 2 artículos para los cuales no haya stock.
El disparador "DIS_articulos_borrar" se activa y permite la transacción pero el disparador
"DIS_articulos_borrar2" no permite la transacción.

13- Solicite la eliminación de 2 artículos para los que haya stock.
El disparador "DIS_articulos_borrar" se activa y no permite la transacción. El disparador
"DIS_articulos_borrar2" no llega a activarse.

144 - Disparador de actualización (update trigger)

Primer problema:
Un club almacena los datos de sus socios en una tabla denominada "socios", las inscripciones en
"inscriptos" y en otra tabla "morosos" guarda los documentos de los socios que deben matrículas.
1- Elimine las tablas si existen:

```
 if object_id('inscriptos') is not null
  drop table inscriptos;
 if object_id('socios') is not null
  drop table socios;
 if object_id('morosos') is not null
  drop table morosos;
```

2- Cree las tablas, con las siguientes estructuras:

```
 create table socios(
  documento char(8) not null,
  nombre varchar(30),
  domicilio varchar(30),
  constraint PK_socios primary key(documento)
 );

 create table inscriptos(
  numero int identity,
  documento char(8) not null,
  deporte varchar(20),
  matricula char(1),
  constraint FK_inscriptos_documento
   foreign key (documento)
   references socios(documento),
  constraint CK_inscriptos_matricula check (matricula in ('s','n')),
  constraint PK_inscriptos primary key(documento,deporte)
 );

 create table morosos(
  documento char(8) not null
```

```
 );
```

3- Ingrese algunos registros en las 3 tablas:

```
 insert into socios values('22222222','Ana Acosta','Avellaneda 800');
 insert into socios values('23333333','Bernardo Bustos','Bulnes 345');
 insert into socios values('24444444','Carlos Caseros','Colon 382');
 insert into socios values('25555555','Mariana Morales','Maipu 234');

 insert into inscriptos values('22222222','tenis','s');
 insert into inscriptos values('22222222','natacion','n');
 insert into inscriptos values('23333333','tenis','n');
 insert into inscriptos values('24444444','futbol','s');
 insert into inscriptos values('24444444','natacion','s');

 insert into morosos values('22222222');
 insert into morosos values('23333333');
```

4- Cree un disparador para la tabla "inscriptos" que se active ante una sentencia "update" y no
permita actualizar más de un registro.

5- Cree otro disparador para la tabla "inscriptos" que se active ante una sentencia "update". Si se
actualiza el pago de la matrícula a 's', el socio debe eliminarse de la tabla "morosos"; no debe
permitir modificar a 'n' una matrícula paga.

6- Actualice cualquier campo (diferente de "matricula") de un registro de la tabla "inscriptos".
Ambos disparadores se activaron permitiendo la transacción.

7- Actualice cualquier campo (diferente de "matricula") de varios registros de la tabla "inscriptos".
El disparador "dis_inscriptos_actualizar1" se activa y no permite la transacción. El disparador
"dis_inscriptos_actualizar_matricula" no llega a activarse.

8- Actualice el campo "matricula" a 's' de un inscripto que deba la matrícula.
Ambos disparadores se activaron y permitieron la actualización.

9- Verifique que el campo se actualizó y que el socio ya no está en "morosos":

```
 select *from inscriptos;
 select *from morosos;
```

10-Actualice el campo "matricula" a 'n' de un inscripto que tenga la matrícula paga.
Ambos disparadores se activaron; "dis_inscriptos_actualizar_matricula" deshace la transacción.

145 - Disparadores (varios eventos)

Primer problema:

Una empresa almacena los datos de sus empleados en una tabla denominada "empleados" y los datos de
las distintas sucursales en una tabla "sucursales".
1- Elimine las tablas si existen:

```
 if object_id('empleados') is not null
  drop table empleados;
 if object_id('sucursales') is not null
  drop table sucursales;
```

2- Cree las tablas, con las siguientes estructuras:

```
 create table sucursales(
  codigo int identity,
  domicilio varchar(30),
  constraint PK_sucursales primary key (codigo)
 );

 create table empleados(
  documento char(8) not null,
  nombre varchar(30),
  domicilio varchar(30),
  sucursal int not null,
  constraint PK_empleados primary key (documento),
  constraint FK_empleados_sucursal foreign key(sucursal)
   references sucursales(codigo)
 );
```

3- Ingrese algunos registros en las dos tablas:

```
 insert into sucursales values ('Colon 123');
 insert into sucursales values ('Sucre 234');
 insert into sucursales values ('Rivadavia 345');

 insert into empleados values ('22222222','Ana Acosta','Avellaneda 1258',1);
 insert into empleados values ('23333333','Betina Bustos','Bulnes 345',2);
 insert into empleados values ('24444444','Carlos Caseres','Caseros 948',3);
 insert into empleados values ('25555555','Fabian Fuentes','Francia 845',1);
 insert into empleados values ('26666666','Gustavo Garcia','Guemes 587',2);
 insert into empleados values ('27777777','Maria Morales','Maipu 643',3);
```

4- Cree un disparador de inserción, eliminación y actualización que no permita modificaciones en la
tabla "empleados" si tales modificaciones afectan a empleados de la sucursal de 1.

5- Ingrese un empleado en la sucursal 3.
El trigger se dispara permitiendo la transacción;

6- Intente ingresar un empleado en la sucursal 1.
El trigger se dispara y deshace la transacción.

7- Ejecute un "update" sobre "empleados" que permita la transacción.

8- Ejecute un "update" sobre "empleados" que el trigger deshaga.

9- Elimine un empleado (o varios) que no sean de la sucursal 1.
El trigger se ejecuta y la transacción se realiza.

10- Intente eliminar un empleado (o varios) de la sucursal 1.
El trigger deshace la transacción.

146 - Disparador (Instead Off y after)

Primer problema:
Un club almacena los datos de sus socios en una tabla denominada "socios", los distintos cursos que
dictan en "cursos" y las inscripciones de los distintos socios en los distintos cursos en "inscriptos".
1- Elimine las tablas si existen:

```
 if object_id('inscriptos') is not null
  drop table inscriptos;
 if object_id('socios') is not null
  drop table socios;
 if object_id('cursos') is not null
  drop table cursos;
```

2- Cree las tablas, con las siguientes estructuras:

```
 create table socios(
  documento char(8) not null,
  nombre varchar(30),
  domicilio varchar(30),
  constraint PK_socios primary key(documento)
 );
 create table cursos(
  numero tinyint identity,
  deporte char(20),
  cantidadmaxima tinyint,
  constraint PK_cursos primary key(numero)
 );

 create table inscriptos(
  documento char(8) not null,
  numerocurso tinyint,
  fecha datetime,
  constraint PK_inscriptos primary key(documento,numerocurso),
  constraint FK_inscriptos_documento
   foreign key (documento)
   references socios(documento),
  constraint FK_inscriptos_curso
   foreign key (numerocurso)
   references cursos(numero)
 );
```

Los cursos tiene un número que los identifica, y según el deporte, hay un límite de inscriptos (por
ejemplo, en tenis, no pueden inscribirse más de 4 socios; en natación, solamente se aceptan 6

alumnos como máximo). Cuando un curso está completo, es decir, hay tantos inscriptos como valor
tiene el campo "cantidadmaxima"), el socio queda inscripto en forma condicional. El club guarda esa
información en una tabla denominada "condicionales" que luego analiza, porque si se inscriben muchos
para un deporte determinado, se abrirá otro curso.

2- Elimine la tabla "condicionales" si existe:

```
if object_id('condicionales') is not null
  drop table condicionales;
```

3- Cree la tabla, con la siguiente estructura:

```
create table condicionales(
  documento char(8) not null,
  codigocurso tinyint not null,
  fecha datetime
);
```

4- Ingrese algunos registros en las tablas "socios", "cursos" e "inscriptos":

```
insert into socios values('22222222','Ana Acosta','Avellaneda 800');
insert into socios values('23333333','Bernardo Bustos','Bulnes 345');
insert into socios values('24444444','Carlos Caseros','Colon 382');
insert into socios values('25555555','Mariana Morales','Maipu 234');
insert into socios values('26666666','Patricia Palacios','Paru 587');

insert into cursos values('tenis',4);
insert into cursos values('natacion',6);
insert into cursos values('basquet',20);
insert into cursos values('futbol',20);

insert into inscriptos values('22222222',1,getdate());
insert into inscriptos values('22222222',2,getdate());
insert into inscriptos values('23333333',1,getdate());
insert into inscriptos values('23333333',3,getdate());
insert into inscriptos values('24444444',1,getdate());
insert into inscriptos values('24444444',4,getdate());
insert into inscriptos values('25555555',1,getdate());
```

5- Cree un trigger "instead of" para el evento de inserción para que, al intentar ingresar un
registro en "inscriptos" controle que el curso no esté completo (tantos inscriptos a tal curso como
su "cantidadmaxima"); si lo estuviese, debe ingresarse la inscripción en la tabla "condicionales" y
mostrar un mensaje indicando tal situación. Si la "cantidadmaxima" no se alcanzó, se ingresa la
inscripción en "inscriptos".

6- Inscriba un socio en un curso que no esté completo.
Verifique que el trigger realizó la acción esperada consultando las tablas:

```
select *from inscriptos;
select *from condicionales;
```

7- Inscriba un socio en un curso que esté completo.
Verifique que el trigger realizó la acción esperada consultando las tablas:
```
 select *from inscriptos;
 select *from condicionales;
```

152 - Disparador (condicionales)

Primer problema:

Una empresa almacena los datos de sus empleados en una tabla denominada "empleados" y en otra tabla
denominada "secciones", información sobre las distintas secciones de la empresa.
1- Elimine las tablas si existen y créelas con los siguientes campos:

```
 if object_id('empleados') is not null
  drop table empleados;
 if object_id('secciones') is not null
  drop table secciones;

 create table secciones(
  codigo int identity,
  nombre varchar(30),
  constraint pk_secciones primary key (codigo)
 );

 create table empleados(
  documento char(8) not null,
  nombre varchar(30),
  domicilio varchar(30),
  seccion int not null,
  constraint pk_empleados primary key (documento),
  constraint fk_empleados_seccion foreign key(seccion)
   references secciones (codigo)
 );
```

2- Ingrese algunos registros:

```
 insert into secciones values('Secretaria');
 insert into secciones values('Sistemas');
 insert into secciones values('Contaduría');
 insert into secciones values('Gerencia');

 insert into empleados values('22222222','Alejandro Acosta','Avellaneda 90',1);
 insert into empleados values('22333333','Betina Bustos','Bulnes 345',2);
 insert into empleados values('23444444','Camila Costa','Colon 234',1);
 insert into empleados values('23555555','Daniel Duarte','Duarte Quiros 345',3);
 insert into empleados values('23666666','Estela Esperanza','España 211',4);
```

3- Cree un disparador de eliminación sobre la tabla "empleados" que permita borrar varios empleados
a la vez, pero ningún empleado de la sección "Gerencia".
Se eliminan todos los empleados solicitados en cualquier sentencia "delete", y luego se vuelven a

insertar aquellos de la sección "Gerencia".

4- Elimine varios registros entre los cuales haya un empleado de "Gerencia".

5- Vea el resultado:
 select *from empleados;
Solamente se eliminaron aquellos que no pertenecen a la sección "Gerencia".