


## TP 1 SEMINARIO DE ACTUALIZACIÓN

1)

Para crear las tablas en SQL Server las separe por lotes y modifique la columna genero agregando una restricción check con dos valores posibles ya que enum no es compatible con SQL Server. 

```
USE Empleados;
```

```
go
```

```
CREATE TABLE empleados (  
id_emp INT NOT NULL,  
fecha_nacimiento DATE NOT NULL,  
nombre VARCHAR(14) NOT NULL,  
apellido VARCHAR(16) NOT NULL,  
genero char(1) check(genero in('M','F')) NOT NULL,  
fecha_alta DATE NOT NULL,  
CONSTRAINT PK_empleados PRIMARY KEY (id_emp)  
);
```

```
go
```

```
CREATE TABLE departamentos (  
id_dept CHAR(4) NOT NULL,  
nombre_dept VARCHAR(40) NOT NULL UNIQUE,  
CONSTRAINT PK_departamentos PRIMARY KEY (id_dept)  
);
```

```
go
```

```
CREATE TABLE dept_respo (  
id_emp INT NOT NULL,  
id_dept CHAR(4) NOT NULL,  
fecha_desde DATE NOT NULL,  
fecha_hasta DATE NOT NULL,  
CONSTRAINT FK_dept_respo_empleados FOREIGN KEY (id_emp) REFERENCES empleados (id_emp),  
CONSTRAINT FK_dept_respo_departamentos FOREIGN KEY (id_dept) REFERENCES departamentos  
(id_dept),  
CONSTRAINT PK_dept_respo PRIMARY KEY (id_emp,id_dept)  
);
```

```
go
```

```
CREATE TABLE dept_emp (  
id_emp INT NOT NULL,  
id_dept CHAR(4) NOT NULL,  
fecha_desde DATE NOT NULL,  
fecha_hasta DATE NOT NULL,  
CONSTRAINT FK_dept_emp_empleados FOREIGN KEY (id_emp) REFERENCES empleados (id_emp),  
CONSTRAINT FK_dept_emp_departamentos FOREIGN KEY (id_dept) REFERENCES departamentos  
(id_dept),  
CONSTRAINT PK_dept_emp PRIMARY KEY (id_emp,id_dept)  
);
```

```
go
```

```
CREATE TABLE puestos (
id_emp INT NOT NULL,
puesto VARCHAR(50) NOT NULL,
fecha_desde DATE NOT NULL,
fecha_hasta DATE,
CONSTRAINT FK_puestos_empleados FOREIGN KEY (id_emp) REFERENCES empleados (id_emp),
CONSTRAINT PK_puestos PRIMARY KEY (id_emp,puesto, fecha_desde)
);
```

go

```
CREATE TABLE sueldos (
id_emp INT NOT NULL,
sueldo INT NOT NULL,
fecha_desde DATE NOT NULL,
fecha_hasta DATE NOT NULL,
CONSTRAINT FK_sueldos_empleados FOREIGN KEY (id_emp) REFERENCES empleados (id_emp),
CONSTRAINT PK_sueldos PRIMARY KEY (id_emp, fecha_desde)
);
```

2)

Como los archivos dados para realizar la carga de datos tienen ciento de miles de registros, tuve que modificar los archivos 02.empleados.sql, 03.dep\_emp.sql, 05.puestos.sql, 06.sueldos.sql, 07.sueldos.sql, 08.sueldos.sql, convirtiéndolos en formato .csv. Luego modifiqué el contenido de estos archivos quitándole las comas finales, paréntesis, comillas, los punto y coma, todas las sentencias INSERT que aparecían, y los espacios en blanco al final del documento, quedando el contenido en un formato de texto plano, por ejemplo:

```
XXXX,XXXX-XX-XX,XXXX,XXX
XXXX,XXXX-XX-XX,XXXX,XXX
XXXX,XXXX-XX-XX,XXXX,XXX
```



Luego procedí a cargar los datos en el orden dado. Los archivos 01.departamentos.sql y 04.depr\_respo.sql al tener pocos datos no tuve que modificarlos, solo copie y pegue su contenido y los ejecute sin problemas. Para cargar los demás archivos, ahora en formato .csv, tuve que usar la sentencia BULK INSERT que me permite cargar datos de un archivo en formato .csv, .txt, etc, desde un directorio externo.

```
use Empleados;
```

go

```
BULK INSERT dbo.empleados
FROM 'C:\Users\FACU\Downloads\02.empleados.csv'
WITH (
    FIELDTERMINATOR = ',', -- Delimitador de campo en el archivo CSV (puede ser diferente si
el archivo utiliza un delimitador diferente)
    ROWTERMINATOR = '\n', -- Delimitador de fila en el archivo CSV (generalmente una nueva
línea)
    FIRSTROW = 2, -- Número de fila desde la cual iniciar la inserción (si el archivo CSV
tiene una fila de encabezado)
    BATCHSIZE = 1000000000000000000 -- Tamaño del lote de inserción (puedes ajustarlo según
tus necesidades)
);
```

go

```

BULK INSERT dbo.dept_emp
FROM 'C:\Users\FACU\Downloads\03.dept_emp.csv'
WITH (
    FIELDTERMINATOR = ',', -- Delimitador de campo en el archivo CSV (puede ser diferente si
    el archivo utiliza un delimitador diferente)
    ROWTERMINATOR = '\n', -- Delimitador de fila en el archivo CSV (generalmente una nueva
    línea)
    FIRSTROW = 2, -- Número de fila desde la cual iniciar la inserción (si el archivo CSV
    tiene una fila de encabezado)
    BATCHSIZE = 1000000000000000000 -- Tamaño del lote de inserción (puedes ajustarlo según
    tus necesidades)
);

go

```

```

BULK INSERT dbo.puestos
FROM 'C:\Users\FACU\Desktop\facu\ analisis de sistemas\3 año\bd\tp1\consultas que use\
05.puestos-libre.csv'
WITH (
    FIELDTERMINATOR = ',', -- Delimitador de campo en el archivo CSV (puede ser diferente si
    el archivo utiliza un delimitador diferente)
    ROWTERMINATOR = '\n', -- Delimitador de fila en el archivo CSV (generalmente una nueva
    línea)
    FIRSTROW = 2, -- Número de fila desde la cual iniciar la inserción (si el archivo CSV
    tiene una fila de encabezado)
    BATCHSIZE = 1000000000000000000 -- Tamaño del lote de inserción (puedes ajustarlo según
    tus necesidades)
);

go

```

```

BULK INSERT dbo.sueldos
FROM 'C:\Users\FACU\Desktop\facu\ analisis de sistemas\3 año\bd\tp1\consultas que use\
06.sueldos.csv'
WITH (
    FIELDTERMINATOR = ',', -- Delimitador de campo en el archivo CSV (puede ser diferente si
    el archivo utiliza un delimitador diferente)
    ROWTERMINATOR = '0x0a', -- Delimitador de fila en el archivo CSV (generalmente una nueva
    línea)
    FIRSTROW = 2, -- Número de fila desde la cual iniciar la inserción (si el archivo CSV
    tiene una fila de encabezado)
    BATCHSIZE = 1000000000000000000 -- Tamaño del lote de inserción (puedes ajustarlo según
    tus necesidades)
);

go

```

```

BULK INSERT dbo.sueldos
FROM 'C:\Users\FACU\Desktop\facu\ analisis de sistemas\3 año\bd\tp1\consultas que use\
07.sueldos.csv'
WITH (
    FIELDTERMINATOR = ',', -- Delimitador de campo en el archivo CSV (puede ser diferente si
    el archivo utiliza un delimitador diferente)
    ROWTERMINATOR = '0x0a', -- Delimitador de fila en el archivo CSV (generalmente una nueva
    línea)
    FIRSTROW = 2, -- Número de fila desde la cual iniciar la inserción (si el archivo CSV
    tiene una fila de encabezado)
    BATCHSIZE = 1000000000000000000 -- Tamaño del lote de inserción (puedes ajustarlo según
    tus necesidades)
);

go

```

```

BULK INSERT dbo.sueldos
FROM 'C:\Users\FACU\Desktop\facu\ analisis de sistemas\3 año\bd\tp1\consultas que use\
08.sueldos.csv'
WITH (
    FIELDTERMINATOR = ',', -- Delimitador de campo en el archivo CSV (puede ser diferente si
el archivo utiliza un delimitador diferente)
    ROWTERMINATOR = '0x0a', -- Delimitador de fila en el archivo CSV (generalmente una nueva
línea)
    FIRSTROW = 2, -- Número de fila desde la cual iniciar la inserción (si el archivo CSV
tiene una fila de encabezado)
    BATCHSIZE = 1000000000000000000 -- Tamaño del lote de inserción (puedes ajustarlo según
tus necesidades)
);

```

3)

El archivo 09.test.sql tuvo que ser adaptado a SQL Server quedando:

```
USE Empleados;
```

```
go
/*
```

```
SET @tiempoini=NOW(6);
```

Esto genero un error ya que primero se debe declarar la variable @tiempoini.  
La función NOW() en MYSQL nos devuelve la fecha y hora actual, admite un parámetro para indicarle la precision de segundos que necesitamos, en este caso 6. Si bien sta función no es compatible en SQL Server se puede obtener el mismo resultado usando la función SYSDATETIME() para tener más precipicio en segundos y convertirla en el tipo DATETIME2(6), indicándole con el 6 la precision en segundos que necesito.



```

*/
DECLARE @tiempoini DATETIME2(6) = SYSDATETIME();

select CAST(SYSDATETIME() AS datetime2(6))

```

Results		Messages
(No column name)		
1	2023-06-29 18:17:04.834324	

```

/*
DROP TABLE IF EXISTS valores_esperados, valores_encontrados;
Esta forma de eliminar una tabla solo es compatible a partir de SQL Server 2016 en adelante.
Como estoy trabajando en SQL Server 2014 utilice la función OBJECT_ID(), que devuelve el
identificador del objeto si existe, o NULL si no existe.
El parámetro 'U' se utiliza para verificar si el objeto es una tabla. Si la tabla existe, se
ejecuta la sentencia DROP TABLE para eliminarla.

```

```

*/
IF OBJECT_ID('valores_esperados', 'U') IS NOT NULL
    DROP TABLE valores_esperados;

IF OBJECT_ID('valores_encontrados', 'U') IS NOT NULL
    DROP TABLE valores_encontrados;

```



```

CREATE TABLE valores_esperados (
    tabla VARCHAR(30) NOT NULL PRIMARY KEY,
    regs INT NOT NULL,
    crc_md5 VARCHAR(100) NOT NULL
);

```

```
/*
CREATE TABLE valores_encontrados LIKE valores_esperados; Esta forma de copiar una tabla no
es compatible en SQL Server. Para lograrlo utilice las siguientes sentencias primero
copiando la tabla valores_esperados en valores_encontrados, y como no se copia con las
restricciones, luego agregue la restricción de clave primaria al campo tabla de
valores_encontrados.
*/
```

```
select * into valores_encontrados from valores_esperados;
alter table valores_encontrados add constraint PK PRIMARY KEY (tabla);
```



```
INSERT INTO valores_esperados VALUES
('empleados', 300024, '4ec56ab5ba37218d187cf6ab09ce1aa1'),
('departamentos', 9, '26eb605e3ec58718f8d588f005b3d2aa'),
('dept_respo', 24, '8720e2f0853ac9096b689c14664f847e'),
('dept_emp', 331603, 'ccf6fe516f990bdaa49713fc478701b7'),
('puestos', 443308, 'bfa016c472df68e70a03facafa1bc0a8'),
('sueldos', 2844047, 'fd220654e95aea1b169624ffe3fca934');
SELECT tabla, regs AS registros_esperados, crc_md5 AS crc_esperado FROM valores_esperados;
```

```
/*
SET @crc= ''; Todavía no esta declarada la variable así que no se puede asignarle un valor.
Para resolverlo declaro la
variable @crc y le asigno el valor ''.
*/
```

```
declare @crc varchar(33)='';
```

```
/*
DROP TABLE IF EXISTS tchecksum;
CREATE TABLE tchecksum (chk char(100));
```

La tabla tchecksum se usa para almacenar el hash MD5 acumulado en la variable @crc de cada uno de los registros de las tablas empleados, puestos, sueldos, departamentos, dept\_emp, dept\_respo, y luego se elimina. Teniendo en cuenta que solamente guarda registros temporalmente, que no son utilizados para realizar ningún otro tipo de consultas, y que con su ausencia adaptando el código se puede lograr el mismo resultado, decidí no usarla.

```
*/
```

```
/*INSERT INTO tchecksum
SELECT @crc :=
MD5(CONCAT_WS('#',@crc,id_emp,fecha_nacimiento,nombre,apellido,genero,fecha_alta))
FROM empleados ORDER BY id_emp;
INSERT INTO valores_encontrados VALUES ('empleados', (SELECT COUNT(*) FROM empleados),@crc);
```

Como en SQL Server no existe una función MD5() de encriptado use la función HASHBYTES(), que recibe dos parámetros que son el algoritmo de encriptado a usar y lo que se desea encriptar, y devuelve un valor de tipo VARBINARY que convertiremos al tipo VARCHAR con la función CONVERT(). Como al resultado lo necesitamos en minúscula usamos la función LOWER(). Este resultado se almacena en la variable @crc que ingresaremos en la tabla valores\_encontrados. La función CONCAT\_WS() no esta disponible en SQL Server 2014 así que la reemplace por la función CONCAT().



```
*/
```

```
SELECT @crc = LOWER(CONVERT(VARCHAR(32)
,HASHBYTES('MD5',CONCAT(@crc,'#',id_emp,'#',fecha_nacimiento,'#',nombre,'#',apellido,'#',gen
ero,'#',fecha_alta) ),2) )
FROM empleados ORDER BY id_emp;
```

```
insert into valores_encontrados values('empleados',(select count(*) from empleados),@crc);
```

```

SET @crc = '';

/*
INSERT INTO tchecksum
    SELECT @crc := MD5(CONCAT_WS('#',@crc, id_dept,nombre_dept))
    FROM departamentos ORDER BY id_dept;
INSERT INTO valores_encontrados VALUES ('departamentos', (SELECT COUNT(*) FROM
departamentos), @crc);
*/

SELECT @crc = LOWER(CONVERT(VARCHAR(32), HASHBYTES('MD5', CONCAT(@crc,'#', id_dept,'#',
nombre_dept)), 2))
FROM departamentos ORDER BY id_dept;

insert into valores_encontrados values('departamentos',(select count(*) from
departamentos),@crc);

SET @crc = '';

/*
INSERT INTO tchecksum
    SELECT @crc := MD5(CONCAT_WS('#',@crc, id_dept,id_emp, fecha_desde,fecha_hasta))
    FROM dept_respo ORDER BY id_dept,id_emp;
INSERT INTO valores_encontrados VALUES ('dept_respo', (SELECT COUNT(*) FROM dept_respo),
@crc);
*/

SELECT @crc = LOWER(CONVERT(VARCHAR(32), HASHBYTES('MD5', CONCAT(@crc,'#', id_dept,'#',
id_emp,'#', fecha_desde,'#', fecha_hasta)) , 2))
FROM dept_respo ORDER BY id_dept, id_emp;

insert into valores_encontrados values('dept_respo',(select count(*) from dept_respo),@crc);

SET @crc = '';

/*
INSERT INTO tchecksum
    SELECT @crc := MD5(CONCAT_WS('#',@crc, id_dept,id_emp, fecha_desde,fecha_hasta))
    FROM dept_emp ORDER BY id_dept,id_emp;
INSERT INTO valores_encontrados VALUES ('dept_emp', (SELECT COUNT(*) FROM dept_emp), @crc);
*/

SELECT @crc = LOWER(CONVERT(VARCHAR(32), HASHBYTES('MD5', CONCAT( @crc,'#', id_dept,'#',
id_emp,'#', fecha_desde,'#', fecha_hasta)), 2))
FROM dept_emp ORDER BY id_dept, id_emp;

insert into valores_encontrados values('dept_emp',(select count(*) from dept_emp),@crc);

SET @crc = '';

/*
INSERT INTO tchecksum
    SELECT @crc := MD5(CONCAT_WS('#',@crc, id_emp, puesto, fecha_desde,fecha_hasta))
    FROM puestos ORDER BY id_emp,puesto,fecha_desde;
INSERT INTO valores_encontrados VALUES ('puestos', (SELECT COUNT(*) FROM puestos), @crc);
*/

SELECT @crc = LOWER(CONVERT(VARCHAR(32), HASHBYTES('MD5', CONCAT(@crc,'#', id_emp,'#',
puesto,'#', fecha_desde,'#', fecha_hasta)), 2))
FROM puestos ORDER BY id_emp, puesto, fecha_desde;

insert into valores_encontrados values('puestos',(select count(*) from puestos),@crc);

SET @crc = '';

/*

```

```

INSERT INTO tchecksum
    SELECT @crc := MD5(CONCAT_WS('#',@crc, id_emp, sueldo, fecha_desde,fecha_hasta))
    FROM sueldos ORDER BY id_emp,fecha_desde,fecha_hasta;
INSERT INTO valores_encontrados VALUES ('sueldos', (SELECT COUNT(*) FROM sueldos), @crc);
*/

```

```

SELECT @crc = LOWER(CONVERT(VARCHAR(32), HASHBYTES('MD5', CONCAT( @crc,'#', id_emp,'#',
sueldo,'#', fecha_desde,'#', fecha_hasta)), 2))
FROM sueldos ORDER BY id_emp, fecha_desde, fecha_hasta;

```

```

insert into valores_encontrados values('sueldos',(select count(*) from sueldos),@crc);

```

```

SET @crc = '';

```

```

--DROP TABLE IF EXISTS tchecksum; Esto no se debe ejecutar ya que no hago uso de la tabla
tchecksum.

```

```

SELECT tabla, regs AS 'registros_encontrados', crc_md5 AS crc_encontrado FROM
valores_encontrados;

```

	tabla	registros_esperados	crc_esperado
1	departamentos	9	26eb605e3ec58718f8d588f005b3d2aa
2	dept_emp	331603	ccf6fe516f990bdaa49713fc478701b7
3	dept_respo	24	8720e2f0853ac9096b689c14664f847e
4	empleados	300024	4ec56ab5ba37218d187cf6ab09ce1aa1
5	puestos	443308	bfa016c472df68e70a03facafa1bc0a8
6	sueldos	2844047	fd220654e95aea1b169624ffe3fca934

	tabla	registros_encontrados	crc_encontrado
1	departamentos	9	26eb605e3ec58718f8d588f005b3d2aa
2	dept_emp	331603	ccf6fe516f990bdaa49713fc478701b7
3	empleados	300024	4ec56ab5ba37218d187cf6ab09ce1aa1
4	dept_respo	24	8720e2f0853ac9096b689c14664f847e
5	puestos	443308	bfa016c472df68e70a03facafa1bc0a8
6	sueldos	2844047	fd220654e95aea1b169624ffe3fca934

```

/*
SELECT
    e.tabla,
    IF(e.regs=f.regs,'OK', 'No OK') AS coinciden_registros,
    IF(e.crc_md5=f.crc_md5,'OK','No OK') AS coindicen_crc
FROM
    valores_esperados e INNER JOIN valores_encontrados f ON e.tabla=f.tabla;

```

En SQL Server 2014 no esta disponible la función IF, por lo que tuve que reemplazarlo por la estructura condicional case.

```

*/

```

```

SELECT
    e.tabla,
    case
        when e.regs=f.regs then 'OK'
        else 'No OK'
    end AS coinciden_registros,

    case
        when e.crc_md5=f.crc_md5 then 'OK'
        else 'No OK'
    end AS coindicen_crc
FROM
    valores_esperados e INNER JOIN valores_encontrados f ON e.tabla=f.tabla;

```

	tabla	coinciden_registros	coinciden_crc
1	departamentos	OK	OK
2	dept_emp	OK	OK
3	empleados	OK	OK
4	dept_respo	OK	OK
5	puestos	OK	OK
6	sueldos	OK	OK

```

declare @crc_fail int, @count_fail int;

```

```

/*
Las variables @crc_fail y @count_fail no se encuentran declaradas por lo tanto no se puede
asignarles un valor. Primero debemos declarar ambas variables*/

```

```

SET @crc_fail=(SELECT COUNT(*) FROM valores_esperados e INNER JOIN valores_encontrados f ON
(e.tabla=f.tabla) WHERE f.crc_md5 != e.crc_md5);

```

```

SET @count_fail=(SELECT COUNT(*) FROM valores_esperados e INNER JOIN valores_encontrados f
ON (e.tabla=f.tabla) WHERE f.regs != e.regs);

```

```

DROP TABLE valores_esperados,valores_encontrados;

```

```

/*
SELECT 'UUID' AS Resumen, @@server_uuid AS Resultado
UNION ALL
SELECT 'CRC', IF(@crc_fail = 0, 'OK', 'Error' )
UNION ALL
SELECT 'Cantidad', IF(@count_fail = 0, 'OK', 'Error' )
UNION ALL
SELECT 'Tiempo', TIMESTAMPDIFF(MICROSECOND,@tiempoini,NOW(6))/1000;

```

Se debe reemplazar la función IF no disponible en SQL Server 2014, por la estructura condicional case.

En SQL Server no existe la variable de sistema @@server\_uuid, por lo tanto como alternativa se puede utilizar la función NEWID() que genera un identificador único similar a un UUID. A esta debemos convertirla a VARCHAR para poder realizar las uniones.

La función TIMESTAMPDIFF() en MYSQL se utiliza para calcular la diferencia de tiempo entre dos fechas. En SQL Server no es admitida, para lograr lo mismo se utiliza DATEDIFF().

Como la función NOW() no es compatible en SQL Server, como al principio usaremos la función SYSDATETIME() y la convertiremos al tipo DATETIME2(6). A esta debemos convertirla a VARCHAR para poder realizar las uniones.

```

*/

```



```

SELECT 'UUID' AS Resumen, CAST(NEWID() AS VARCHAR(36)) AS Resultado
UNION ALL
SELECT 'CRC',
    case
        when @crc_fail = 0 then 'OK'
        else 'Error'
    end
UNION ALL
SELECT 'Cantidad',
    case
        when @count_fail = 0 then 'OK'
        else 'Error'
    end
UNION ALL
SELECT 'Tiempo', CAST(DATEDIFF(MICROSECOND,@tiempoini,CAST(SYSDATETIME() AS DATETIME2(6)))/1000 AS VARCHAR);

```

	Resumen	Resultado
1	UUID	2B46C2A9-B835-4298-B9E5-D09D960AA873
2	CRC	OK
3	Cantidad	OK
4	Tiempo	27628

4)

A continuación se detalla lo hecho en este punto:

/\*

Primero creo la tabla puesto\_descripción que tiene como clave primaria a la columna 'id' de tipo SMALLINT y que se incrementa automáticamente de 1 en 1. La columna 'puesto\_descr\_en' se utilizara para almacenar la descripción de los puestos en ingles, mientras que en la columna puesto\_descr\_es se se almacenan las descripciones de los puestos en español.

\*/

```

CREATE TABLE puesto_descripcion(
    id SMALLINT PRIMARY KEY IDENTITY(1,1),
    puesto_descr_en VARCHAR(100) NOT NULL,
    puesto_descr_es VARCHAR(100) NOT NULL
)

```

/\*

Una vez creada la tabla procedemos a ingresar los datos. Para ello selecciono de la tabla puestos el campo puesto, que tiene las descripciones en ingles, y aplico una estructura case sobre el campo puesto para devuelva y seleccione la traducción correspondiente a cada descripción. Al final debo agrupar por el campo puesto para eliminar los datos repetidos quedando solo 7 descripciones, si no lo hago se agregaran todas las descripciones de la tabla puesto.

\*/

```

INSERT INTO puesto_descripcion
SELECT puesto, case puesto

                WHEN 'Senior Engineer' THEN 'Ingeniero superior'
                WHEN 'Staff' THEN 'Personal'
                WHEN 'Engineer' THEN 'Ingeniero'
                WHEN 'Senior Staff' THEN 'Personal superior'
                WHEN 'Assistant Engineer' THEN 'Ingeniero Asistente'
                WHEN 'Technique Leader' THEN 'Líder de Técnica'
                WHEN 'Manager' THEN 'Gerente'
END
FROM puestos
GROUP BY puesto

```

/\*

Para relacionar la tabla puestos con puesto\_descripcion primero debo modificar los valores de la columna puesto de la tabla puestos por los id auto generados correspondientes a la descripcion en ingles de la tabla puesto\_descripcion, es decir, tengo que comparar las descripcion del campo puesto de la tabla puestos con el campo puesto\_descr\_en de la tabla puestos\_descripcion. De esta forma me quedaran números de tipo VARCHAR en la columna puesto de la tabla puestos.

\*/

```

UPDATE puestos set puestos.puesto= pd.id FROM puestos p JOIN puesto_descripcion pd
ON p.puesto=pd.puesto_descr_en;

```

/\*

Como los id de la tabla puesto\_descripcion son del tipo SMALLINT, debo modificar el tipo de dato de la columna puesto de la tabla puestos, que esta en VARCHAR, a SMALLINT para poder relacionarlo. La columna puesto de la tabla puestos tiene una restricción de clave primaria, así que para modificarla primero debemos eliminar esta restricción y luego modificar la columna.

\*/

```

alter table puestos drop constraint PK_puestos;

alter table puestos alter column puesto SMALLINT not null ;

```

/\*

Una vez modificada la estructura de la columna puesto podemos continuar con la relación entre las tablas. Para lograr esto se debe agregar una restricción de clave foránea sobre la columna puesto que haga referencia a la columna id de la tabla puesto\_descripcion, quedando ambas tablas relacionadas por el campo puesto de la tabla puestos y el campo id de la tabla puesto\_descripcion.

\*/

```

alter table puestos add constraint fk_puesto_descripcion foreign key( puesto ) references
puesto_descripcion(id);

```

/\*

Por ultimo debo agregar la restricción de clave primaria de la tabla puestos, eliminada anteriormente, para mantener una estructura similar a la original.

\*/

```
alter table puestos add constraint PK_puestos primary key (id_emp,puesto,fecha_desde);
```

```
select * from puesto_descripcion;
```

Results		Messages	
	id	puesto_descr_en	puesto_descr_es
1	1	Manager	Gerente
2	2	Engineer	Ingeniero
3	3	Staff	Personal
4	4	Technique Leader	Líder de Técnica
5	5	Senior Engineer	Ingeniero superior
6	6	Senior Staff	Personal superior
7	7	Assistant Engineer	Ingeniero Asistente

```
select * from puestos;
```

Results		Messages		
	id_emp	puesto	fecha_desde	fecha_hasta
1	10001	5	1986-06-26	9999-01-01
2	10002	3	1996-08-03	9999-01-01
3	10003	5	1995-12-03	9999-01-01
4	10004	2	1986-12-01	1995-12-01
5	10004	5	1995-12-01	9999-01-01
6	10005	3	1989-09-12	1996-09-12
7	10005	6	1996-09-12	9999-01-01
8	10006	5	1990-08-05	9999-01-01
9	10007	3	1989-02-10	1996-02-11
10	10007	6	1996-02-11	9999-01-01
11	10008	7	1998-03-11	2000-07-31
12	10009	2	1990-02-18	1995-02-18
13	10009	5	1995-02-18	9999-01-01
14	10009	7	1985-02-18	1990-02-18
15	10010	2	1996-11-24	9999-01-01
16	10011	3	1990-01-22	1996-11-09
17	10012	2	1992-12-18	2000-12-18
18	10012	5	2000-12-18	9999-01-01



```
CREATE OR ALTER FUNCTION [dbo].[MD5]
(
    -- Add the parameters for the function here
    @pcadena varchar(max)
)
RETURNS varchar(32)
AS
BEGIN
    -- Declare the return variable here
    DECLARE @Result varchar(32)

    -- Add the T-SQL statements to compute the return value here
    SELECT @Result = LOWER(CONVERT(VARCHAR(32), HASHBYTES('MD5', @pcadena), 2))

    -- Return the result of the function
    RETURN @Result
END
```

```
SET @crc= '';
| SELECT @crc = dbo.MD5(CONCAT_WS('#',@crc,id_emp,fecha_nacimiento,nombre,apellido,genero,fecha_alta))
  FROM empleados ORDER BY id_emp;
INSERT INTO valores_encontrados VALUES ('empleados', (SELECT COUNT(*) FROM empleados),@crc);
```