Usar ado.net o ado en delphi para la parte del cliente

SQL server tiene que estar running. Está en modo manual

Carpeta SQLServer para dividir los datos en diferentes carpetas.. En C no poner nada!

Filegroup: posibilidad de crear varios archivos y apuntarlos a un solo lugar.

En el Mapping interno - conceptual se guardan los filegroup

Initial Size es el espacio que necesita

Autogrowth/maxsize es lo que va a crecer. Puede ser en MG o en porcentaje (no se usa). Hay que poner una gran cantidad de MG para que no pida a cada rato espacio.

Parte interna

New database

Poner nombre y asignar las direcciones correspondientes.

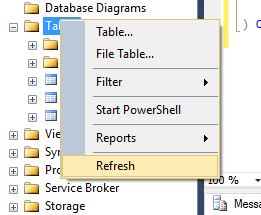
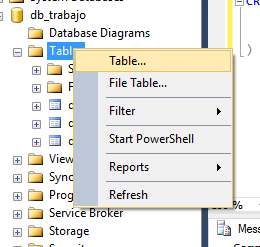
Hacer filegroup y ponerlo como default

Definir tamaños

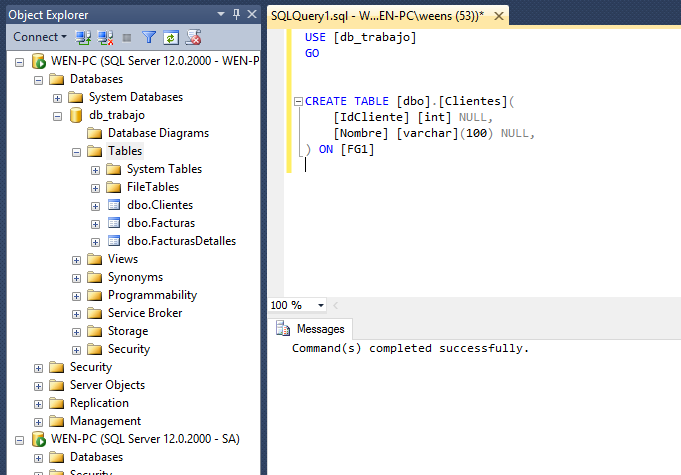
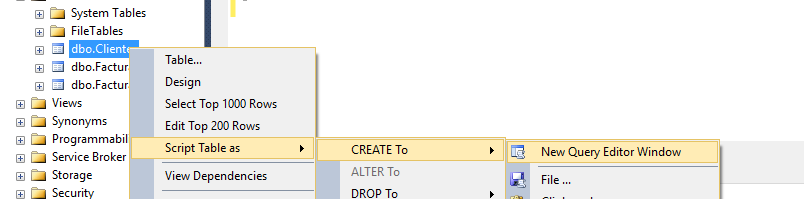
Crear

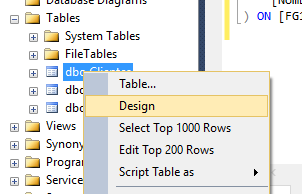
Parte conceptual

Crear tablas!

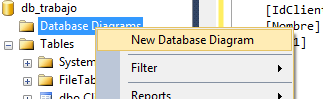


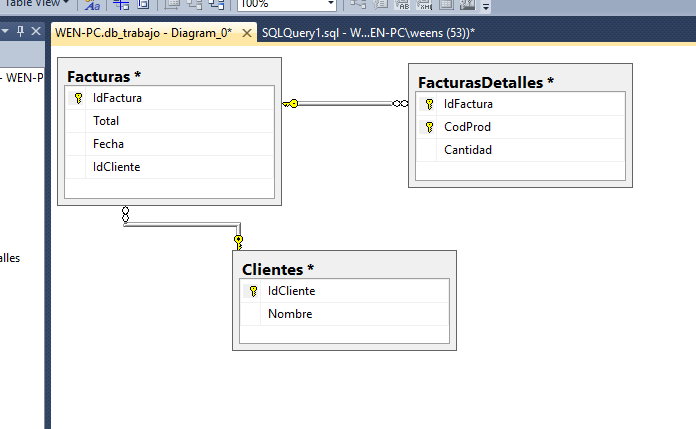
Crear tabla mediante script:



Para la clave primaria: 

Relaciones entre tablas:





Empezar siempre de uno a varios

El script no se guarda

crear base de datos

modelo de datos . crear tablas

relacionar en el diagrama

claves primarias

CHECK: para condiciones (ej: que un saldo no pueda ser negativo)

tables-bd-constraints-check constraints-expression (id>0)

//con poner en una tabla un check alcanza

VISTAS: para elegir que info mostrar. (ej: mostrar solo dos columnas)

databases-bdejemplo-views

CREATE VIEW vw\_clientes

AS

SELECT

FROM

GO

modificacion de la vista

ALTER VIEW....

//poner 1 o 2 vistas

PROCEDIMIENTOS ALMACENADOS: abstraer la carga. No cambiar en todo las modificaciones, solo el procedimiento

ventajas de los prodecimientos: seguridad, desempeño y mantenibilidad

exec sp\_CargaCliente id=10, nombre="dgjj"....

go

CREAT PROCEDURE sp\_CargaClientes

//campos

AS

INSERT INTO Producto (campos)

VALUES ()

GO

**TP**

Nivel conceptual:

Hacer si o si un check

Si hay tiempo agregar:

Clave de unicidad: UNIQUE

TRIGGER

Nivel externo:

Alta cliente en procedimiento para llamar solo eso desde c#

Procedimiento para cambiar estado, crear factura, etc.. Esto si hay tiempo.

CLIENTE

Solo tiene permiso para ejecutar el procedimiento almacenado de alta cliente (externo). No puede acceder al modelo de datos (conceptual).

Restringir seguridad. Crear un usuario en BD diferente a la del administrador para conectarse desde el programa con eso y poner los permisos.

Necesitamos en C# librerías de acceso a datos de BD.

ADO.NET - seguridad integrada que se conecte a un usuario diferente al de la bd. (algo de sentencia GRANT procedimiento TO usuario)

SELECT PP.[Cod-Producto], SUM(Cantidad) AS 'Cantidad de productos'

INTO TMP

FROM Pedidos P, [Pedidos-Productos] PP

WHERE P.[Cod-Pedido] = PP.[Cod-Pedido] AND [Fecha-Ingreso] > (GETDATE() - 30)

GROUP BY PP.[Cod-Producto]

SELECT P.[Cod-Producto], [Tipo-Producto]

FROM TMP, Productos P

WHERE P.[Cod-Producto] = TMP.[Cod-Producto] AND [Cantidad de productos] IN (SELECT MAX([Cantidad de productos]) FROM TMP)

SELECT P.[Cod-Producto], [Tipo-Producto]

FROM TMP, Productos P

WHERE P.[Cod-Producto] = TMP.[Cod-Producto] AND [Cantidad de productos] IN (SELECT MIN ([Cantidad de productos]) FROM TMP)

ALTER PROCEDURE [dbo].[sp\_Estado]

@CodPedido [nchar](10),

@CodEstado [nchar](10),

@NroCamion [int],

@CodCamionero [int],

@NroCentroDist [int],

@NroRemito [int],

@NroFactura [nchar](10),

@NroRecibo [nchar](10)

AS

DECLARE

@Existe [int] = 0,

@Pago [nchar](2) = 0

IF @CodEstado = 1

INSERT [Pedidos-Estados] ([Cod-Pedido],[Cod-Estado])

VALUES (@CodPedido,@CodEstado)

RETURN;

IF @CodEstado = 2

SELECT @Existe = COUNT([Cod-Estado])

FROM [Pedidos-Estados]

WHERE [Cod-Estado] = 1

IF @Existe = 1

BEGIN

INSERT [Pedidos-Estados] ([Cod-Pedido],[Cod-Estado],[Nro-Camion],[Cod-Camionero],[Nro-Centro-Dist],[Nro-Remito])

VALUES (@CodPedido,@CodEstado,@NroCamion,@CodCamionero,@NroCentroDist,@NroRemito)

--Baja de stock en almacen

SELECT PP.[Cod-Producto], Cantidad INTO TMP2

FROM [Pedidos-Productos] PP, [Pedidos-Estados] PE

WHERE PP.[Cod-Pedido] = PE.[Cod-Pedido]

UPDATE Productos

SET [Cant-Existente] = [Cant-Existente] - Cantidad

FROM TMP2 T, Productos P

WHERE T.[Cod-Producto] = P.[Cod-Producto]

DROP TABLE TMP2

--RETURN;

END

ELSE

RETURN 'Estado no valido';

IF @CodEstado = 3

SELECT @Existe = COUNT([Cod-Estado])

FROM [Pedidos-Estados]

WHERE [Cod-Estado] = 2

IF @Existe = 1

BEGIN

INSERT [Pedidos-Estados] ([Cod-Pedido],[Cod-Estado])

VALUES (@CodPedido,@CodEstado)

--RETURN;

END

ELSE

RETURN 'Estado no valido';

IF @CodEstado = 4

SELECT @Existe = COUNT([Cod-Estado])

FROM [Pedidos-Estados]

WHERE [Cod-Estado] = 3

IF @Existe = 1

BEGIN

INSERT [Pedidos-Estados] ([Cod-Pedido],[Cod-Estado],[Nro-Factura])

VALUES (@CodPedido,@CodEstado,@NroFactura)

--RETURN;

END

ELSE

RETURN 'Estado no valido';

IF @CodEstado = 5

SELECT @Existe = COUNT([Cod-Estado])

FROM [Pedidos-Estados]

WHERE [Cod-Estado] = 4

IF @Existe = 1

BEGIN

SELECT @Pago = [Pago-Contado]

FROM Pedidos

WHERE [Cod-Pedido] = @CodPedido AND [Pago-Contado] = 'Si'

IF @Pago = 'Si'

INSERT [Pedidos-Estados] ([Cod-Pedido],[Cod-Estado],[Nro-Recibo])

VALUES (@CodPedido,@CodEstado,@NroRecibo)

ELSE

INSERT [Pedidos-Estados] ([Cod-Pedido],[Cod-Estado])

VALUES (@CodPedido,@CodEstado)

--RETURN;

END

ELSE

RETURN 'Estado no valido';