

TÍTULO

DESARROLLO DE APLICACIONES I

Sesión Nro. 14





Objetivos de la sesión:

Al culminar la presente podrás:

- Implementar el registro de información en tablas cabecera detalle .
- Resaltar la importancia de conocer criterios de implementación en SQL Server por parte del desarrollador.
- Manejar conceptos transaccionales dentro de un sitio WEB.



Visual
Studio



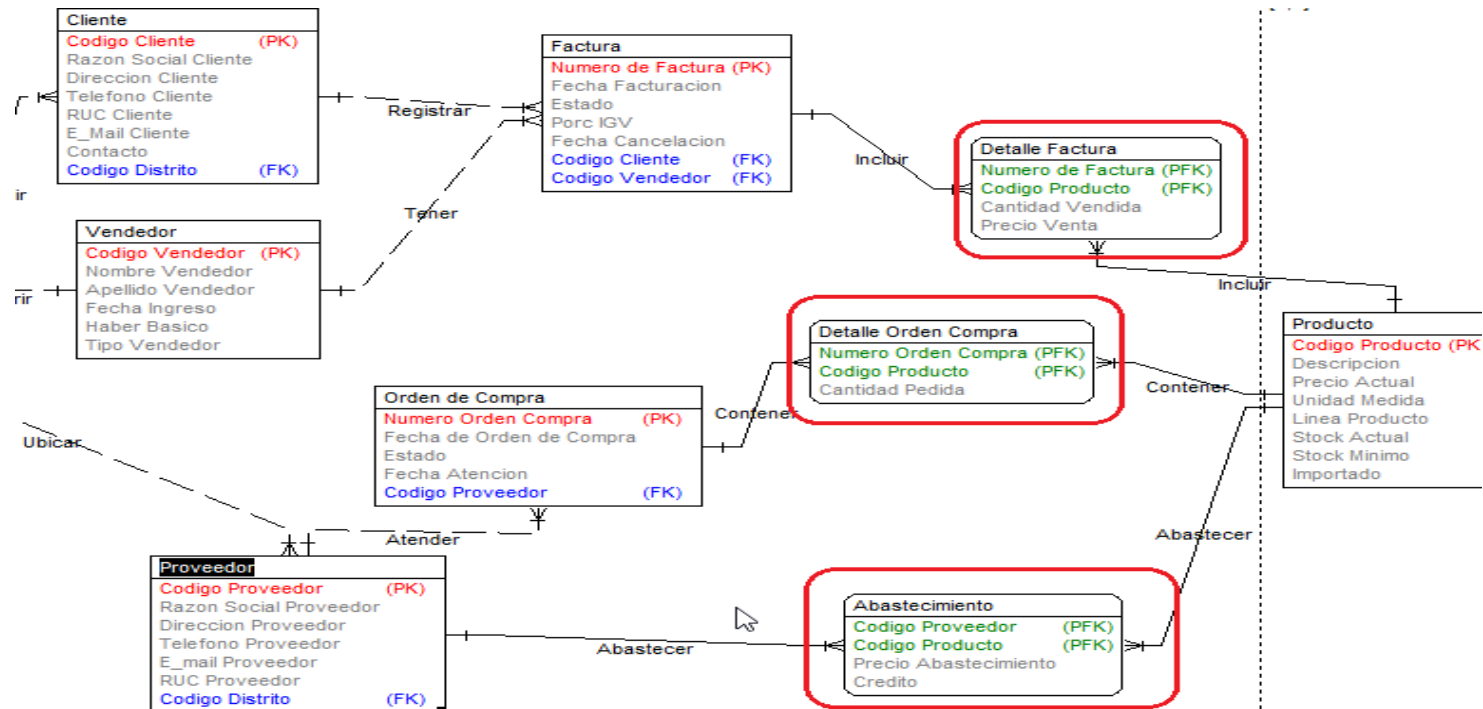
Temario

1. Implementar consultas paginadas con Store Procedure
2. Registrar datos en tablas maestro - detalle
3. Los parámetros tipo tabla en SQL Server
4. El escenario para registrar una orden de compra en la BD VentasLeon



1.1 ¿Qué es una tabla Maestro Detalle?

Es una tabla que se implementa desde una relación de muchos a muchos dentro del Modelo Lógico de BD.



Por consecuencia del DER anterior podemos implementar tablas como Orden_Compra (maestro) y Detalle Orden Compra (detalle) como también Factura y Detalle_Factura

1.2 ¿Qué es una transacción?

Dentro del ámbito del desarrollo de software se define como transacción a una operación que consta de varios pasos, de tal manera que el éxito de la misma depende del éxito de cada uno de los pasos que la constituyen.

Para que la operación transaccional se realice satisfactoriamente, todos los pasos que la componen deben darse de manera correcta.

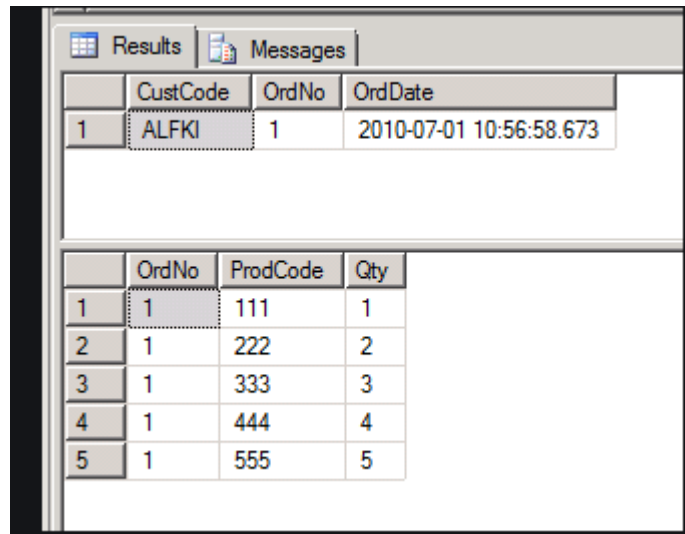
Las transacciones se pueden dar desde el ámbito de la base de datos (recomendable en Store Procedure) o en código de la capa de datos.

1.2 ¿Qué es una transacción?

En el siguiente ejemplo Sql Server, la transacción contiene 2 Update de la tabla Cuentas. El Begin Transaction inicia la operación, la cual se confirma (si todo esta OK) con Commit Transaction. Como todo esta dentro de un bloque Begin Try, si algunos de los Update provoca una excepción, en el bloque Begin Catch se ejecuta un RollBack Transaction, cancelando toda la operación.

```
BEGIN TRY
    BEGIN TRANSACTION
        UPDATE cuentas SET balance = balance - 1250 WHERE nombreCliente = 'Antonio';
        UPDATE cuentas SET balance = balance + 1250 WHERE nombreCliente = 'Claudio';
    COMMIT TRANSACTION
    PRINT 'Transacción completada'
END TRY
BEGIN CATCH
    ROLLBACK TRANSACTION
    PRINT 'Transacción cancelada'
END CATCH
```

Tema 2: Los parámetros tipo tabla en SQL Server



The screenshot displays two result sets from a SQL query. The top result set contains one row with the following data:

	CustCode	OrdNo	OrdDate
1	ALFKI	1	2010-07-01 10:56:58.673

The bottom result set contains five rows of data:

	OrdNo	ProdCode	Qty
1	1	111	1
2	1	222	2
3	1	333	3
4	1	444	4
5	1	555	5



2.1 ¿Qué es un parámetro tipo tabla?

Normalmente cuando usamos Store Procedure en Sql Server solemos pasarle parámetros. Estos pueden ser de un tipo soportado por Sql como varchar , smalldatetime, int , etc. En estos parámetros por cada uno de ellos solo podemos pasar un dato, por lo que son parámetros escalares.

Si nuestra necesidad es pasarle a un Store Procedure no solo un dato sino un conjunto de registros en un solo parámetro, estamos enfocándonos a los llamados parámetros tipo tabla.



2.1 ¿Qué es un parámetro tipo tabla?

En un parámetro tipo tabla podemos pasarle a un Store Procedure un conjunto de registros para que el los pueda operar de manera transaccional dentro del ámbito del propio Store.

Para el caso de los registros de tablas maestro detalle, se hara imprescindible este tipo de parámetros, dado que a través de ellos pasaremos todo el detalle que se desea registrar, en un solo parámetro.



2.2 ¿Cómo se crea un parámetro tipo tabla?

La creación de un parámetro tipo tabla es como se muestra en la siguiente imagen:

```
if(Type_Id('dbo.MisDetallesOC') is not null)
    drop type dbo.MisDetallesOC
go

CREATE TYPE [dbo].[MisDetallesOC] AS TABLE
(
    [Cod] [char](4) NOT NULL,
    [Descrip] [varchar] (50) NULL,
    [Can] [int] NULL
)
go
```

Aquí se crea un parámetro tipo tabla llamado “MisDetallesOC” con 3 campos: Cod (como char(4)), Descrip (como varchar(50)) y Can (como int)

Tema 3: Preparando el escenario para registrar una OC en la base de datos VentasLeon



3.1 ¿Qué tablas están asociadas para registrar una orden de compra?

Como ya se menciono las tabla Tb_Orden_Compra (maestro) y Tb_Detalle_Compra (detalle) son los principales protagonistas del proceso, teniendo a la tabla Tb_Producto como “artista invitada”.

```

dbo.Tb_Orden_Compra
├── Columns
│   ├── Num_oco (PK, nvarchar(6), not null)
│   ├── Fec_oco (datetime, not null)
│   ├── Cod_prv (FK, nchar(4), not null)
│   ├── Fec_ate (datetime, null)
│   ├── Est_oco (int, not null)
│   ├── Fec_Registro (datetime, null)
│   ├── Usu_Registro (nvarchar(20), null)
│   ├── Fec_Ult_Mod (datetime, null)
│   └── Usu_Ult_Mod (nvarchar(20), null)

```

```

dbo.Tb_Detalle_Compra
├── Columns
│   ├── Num_oco (PK, FK, nvarchar(6), not null)
│   ├── Cod_pro (PK, FK, nchar(4), not null)
│   └── Can_ped (int, null)
└── Keys

```

```

dbo.Tb_Producto
├── Columns
│   ├── Cod_pro (PK, nchar(4), not null)
│   ├── Des_pro (nvarchar(50), null)
│   ├── Pre_pro (money, null)
│   ├── Stk_act (int, null)
│   ├── Stk_min (int, null)
│   ├── Id_UM (FK, int, null)
│   ├── Id_Cat (FK, int, null)
│   ├── Importado (int, null)
│   ├── Fec_Registro (datetime, null)
│   ├── Usu_Registro (nvarchar(20), null)
│   ├── Fec_Ult_Mod (datetime, null)
│   ├── Usu_Ult_Mod (nvarchar(20), null)
│   └── Est_pro (int, null)

```

3.2 ¿Qué objetos necesitamos tener en la BD VentasLeon para registrar una orden de compra?

Debemos entonces tener un tipo de dato tabla para que a través de el podamos pasarle al Store Procedure encargado del registro de la OC, un parámetro con todos los detalles de la orden:

```
CREATE TYPE [dbo].[MisDetallesOC] AS TABLE
(
    [Cod] [char](4) NOT NULL,
    [Descrip] [varchar] (50) NULL,
    [Can] [int] NULL
)
GO
```

3.2 ¿Qué objetos necesitamos tener en la BD VentasLeon para registrar una orden de compra?

Creado el tipo de dato tabla se requiere el Store Procedure que se haga cargo del registro de la OC, que reciba los datos de cabecera en parámetros escalares y el detalle de la orden en un parámetro de tipo tabla. Sera el propio Store quien genere el siguiente numero de OC, siendo precisamente este el valor que retornara el Store. Veamos:




3.2 ¿Qué objetos necesitamos tener en la BD VentasLeon para registrar una orden de compra?

```
]CREATE PROCEDURE [dbo].[usp_RegistrarOcompra]
    @vfecoco smalldatetime,
    @vcodprv char(4),
    @vfecate smalldatetime,
    @vestoco int,
    @vUsu_registro varchar(20),
    @vnumoco char(5) output,
    @detalles MisdetallesOC readonly -- parametro tipo tabla
AS
-- INICIAMOS TRANSACCION
Begin Transaction
]    Begin Try -- Iniciamos bloque Try en SQL
        declare @vcont int
        set @vcont=(Select count(*) from tb_Orden_Compra)
]    if @vcont=0
        set @vnumoco = 'OC0001'
    else
        --Se genera el numero de Orden de Compra
]    Set @vnumoco=(select 'OC' + right(max(right(num_oco,4)+10001),4)
                    from Tb_Orden_Compra)
-
```


3.3 Que necesitamos en nuestra aplicación para hacer efectivo el Registro de una Orden de Compra

Una vez que tengamos creados los objetos en la BD, corresponde acceder a ellos creando una entidad de negocio que maneje correctamente los datos de cabecera y los detalles de la OC ,crear una clase en la capa de datos (ADO) con un método que invoque a los objetos creados en la BD para luego definir la clase y métodos en la capa de negocios y finalmente el diseño del formulario en la capa de presentación que consuma todo lo Implementado.

A decorative graphic in the bottom right corner consisting of overlapping blue and yellow diagonal lines.

3.4 Registro de una Orden de Compra. Entidad de Negocios

Entidad de Negocio OrdenBE

```
public class OrdenBE
{
    public String NumOco
    {
        get;
        set;
    }
    public System.DateTime FecOco
    {
        get;
        set;
    }
    public String CodPrv
    {
        get;
        set;
    }
    public System.DateTime FecAte
    {
        get;
        set;
    }
    public String EstOco
    {
        get;
        set;
    }

    // Para el detalle
    private DataTable Detalles;
    public DataTable DetallesOC
    {
        get { return Detalles; }
        set { Detalles =value; }
    }
}
```

```
// Propiedades de Auditoria

public DateTime Fec_Registro
{
    get;
    set;
}

public String Usu_Registro
{
    get;
    set;
}

public DateTime Fec_Ult_Mod
{
    get;
    set;
}

public String Usu_Ult_Mod
{
    get;
    set;
}
}
```

3.5 Registro de una Orden de Compra. Capa de datos

Clase de Acceso a Datos OrdenADO

```
public String RegistrarOrden(OrdenBE objOrdenBE)
{
    try
    {
        cnx.ConnectionString = MiConexion.GetCnx();
        cmd.Connection = cnx;
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "usp_RegistrarOCompra";
        cmd.Parameters.Clear();
        cmd.Parameters.AddWithValue("@vfecoco", objOrdenBE.FecOco);
        cmd.Parameters.AddWithValue("@vcodprv", objOrdenBE.CodPrv);
        cmd.Parameters.AddWithValue("@vfecate", objOrdenBE.FecAte);
        cmd.Parameters.AddWithValue("@vestoco", objOrdenBE.EstOco);
        cmd.Parameters.AddWithValue("@vUsu_registro", objOrdenBE.Usu_Registro);
        // Parametro de salida
        cmd.Parameters.Add(new SqlParameter("@vnumoco", SqlDbType.Char, 5));
        cmd.Parameters["@vnumoco"].Direction = ParameterDirection.Output;
        // Parametro tipo tabla con los detalles
        cmd.Parameters.Add(new SqlParameter("@detalles", SqlDbType.Structured));
        cmd.Parameters["@detalles"].Value = objOrdenBE.DetallesOC;
        cnx.Open();
        cmd.ExecuteNonQuery();
        // Retorna el NUMOCO generado.
        return cmd.Parameters["@vnumoco"].Value.ToString ();
    }
    catch (Exception ex)
    {
        return String.Empty;
    }
    finally
    {
        if (cnx.State == ConnectionState.Open)
        {
            cnx.Close();
        }
    }
}
```

3.6 Registro de una Orden de Compra. Capa de negocios

Clase de Lógica de Negocios OrdenBL

```
using System.Linq;  
using System.Data;  
using ProyVentas_ADO;  
using ProyVentas_BE;  
  
namespace ProyVentas_BL  
{  
    public class OrdenBL  
    {  
        OrdenADO objOrdenADO = new OrdenADO();  
  
        public string RegistrarOrden(OrdenBE objOrdenBE)  
        {  
            return objOrdenADO.RegistrarOrden(objOrdenBE);  
        }  
    }  
}
```

3.7 Diseño del formulario de Registro de una Orden de Compra

Capa de Presentación: Formulario de Registro de Orden de Compra

Registrar Orden de Compra

Ingrese fecha OC:

Seleccione Proveedor:

Ingrese fecha atencion:

Agregar Detalle
Grabar
Cancelar

	Codigo Producto	Descripción	Cantidad Pedida
✖	DataBound	DataBound	DataBound
✖	DataBound	DataBound	DataBound
✖	DataBound	DataBound	DataBound
✖	DataBound	DataBound	DataBound
✖	DataBound	DataBound	DataBound

[Retornar Menu](#)

asp:Panel#PanelDetalle

Registrar Detalle

Seleccione Producto:

Cantidad Pedida:

Laboratorio: Implemente la especificación funcional de registro de Orden de Compra



Conclusiones de la sesión:

- Es muy eficiente manejar la paginación por Store Procedure por encima de la paginación del control GridView. El rendimiento del aplicativo es mucho mas eficiente.
- Podemos establecer procesos transaccionales dentro de sitios WEB de la mano de los Store Procedure y buena metodología en el uso de herramientas de programación.
- Esto implica que ya estamos en condiciones de implementar escenarios que emulen a los matriculas, reserva de habitaciones, pedidos entre otros , emulando a los conocidos “Carritos de Compra” que encontramos en muchos sitios WEB de prestigio.

