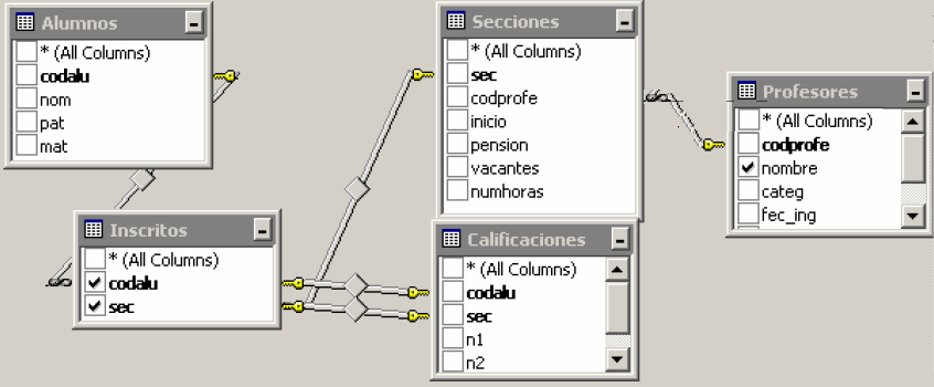
Actividad

Actividad 1 - Vistas

Generar la base de datos con las siguientes tablas:

1. Alumnos {# codalu, nom, pat, mat}
2. Inscritos {# codalu,FK sec}
3. Secciones {#sec, FK codprofe, inicio, pension, vacantes, numhoras}
4. Calificaciones {FK codalu, FK sec, n1, n2}
5. Profesores {#codprofe, nombre, apellidos, categ, fecha\_ing,pension}



1. Crear la vista relacion\_de\_promedios:

SELECT inscritos.coddalu as codigo, alumnos.nom + ' ' + alumnos.pat as nomape, inscritos.sec as seccion, profesores.nombre as profesor, (calificaciones.n1 + calificaciones.n2)/2 as promedio FROM alumnos, secciones,inscritos,calificaciones,profesores WHERE alumos.codalu=inscritos.codalu and inscritos.sec=secciones.sec and secciones.codprofe=profesores.codprofe and inscritos.sec=calificaciones.sec and inscritos.codalu=calificaciones.codalu and calificaciones.sec=secciones.sec

1. Para comprobar la creación de la vista en el Analizador de Consultas digite la siguiente instrucción:

Select \* From Relacion\_de\_Promedios

GO

1. Ver la sentencia que implementa la vista:

Sp\_HelpText Relacion\_de\_Promedios

GO

1. Ver las columnas que se presentan en la vista:

Sp\_Depends Relación de Promedios

GO

1. Desde el Query Anallyzer debe crear una vista que nos permita mostrar un informe resumen por sección que muestre la sección, el nombre del profesor, el total de alumnos, el mayor y menor promedio, el promedio de la sección y el monto acumulado de las pensiones.

CREATE VIEW Resumen

AS

Select Inscritos.Sec As Seccion, MAX(nombre) As Profesor, Count(Inscritos.Codalu) As Alumnado, Max((N1+N2)/2) As MayorPromedio, Min((N1+N2)/2) As MenorPromedio, Avg((N1+N2)/2) As PromedioSec, Sum(Pension) As Acumulado From Inscritos INNER JOIN Calificaciones On Inscritos.codAlu = Calificaciones.codalu AND Inscritos.sec = Calificaciones.Sec INNER JOIN Secciones On Inscritos.Sec = Secciones.Sec INNER JOIN Profesores On Secciones.CodProfe = Profesores.CodProfe Group by Inscritos.Sec

GO

1. Compruebe la información que devuelve la vista:

Select \* From Resumen

GO

1. Mostrar la implementación de la vista:

Sp\_HelpText Resumen

GO

Modificación de vista

ALTER VIEW RESUMEN

WITH ENCRYPTION

AS

Select Inscritos.Sec As Seccion, MAX(nombre) As Profesor, Count(Inscritos.Codalu) As Alumnado, Max((N1+N2)/2) As MayorPromedio, Min((N1+N2)/2) As MenorPromedio, Avg((N1+N2)/2) As PromedioSec, Sum(Pension) As Acumulado From Inscritos INNER JOIN Calificaciones On Inscritos.codAlu = Calificaciones.codalu AND Inscritos.sec = Calificaciones.Sec INNER JOIN Secciones On Inscritos.Sec = Secciones.Sec INNER JOIN Profesores On Secciones.CodProfe = Profesores.CodProfe Group by Inscritos.Sec

GO

1. Ahora muestre la implementación de la vista con el siguiente comando:

Sp\_HelpText Resumen

GO

1. Como se observa en el panel de resultados ahora ya no se muestra la implementación de la vista.
2. Lo mismo ocurrirá si consulta la información de la tabla del sistema SysComments.

Select \* From SysComments

GO

Eliminación de vistas

DROP VIEW [Relacion\_de\_Promedios]

GO

Select \* From [Relacion de Promedios]

GO

Actividad 2 – Funciones

Ejemplo 1:

create function OrdersMonth(@MyMonth as int)   
returns table   
as   
return (select \* from orders where month(OrderDate)=@MyMonth)

Ejemplo 2:

Create FUNCTION Suma (@a int, @b int)   
RETURNS int   
AS   
BEGIN   
DECLARE @c int   
SET @c = @a + @b   
RETURN (select @c)   
END

Ejemplo 3:

CREATE FUNCTION [dbo].[funcDatePart]   
( variable de entrada y tipo @DateType varchar(10), @Date Datetime )

RETURNS int   
AS   
BEGIN   
   
DECLARE   
\*agregamos las variables que necesitemos para realizar la funcion   
@Period int   
   
SELECT @Period = (   
CASE   
WHEN @DateType = 'year' THEN datepart(year,@Date)   
WHEN @DateType = 'month' THEN datepart(month,@Date)   
WHEN @DateType = 'quarter' THEN datepart(quarter,@Date)   
WHEN @DateType = 'semester' THEN   
(CASE   
WHEN datepart(month,@Date) between 1 and 6   
THEN '1'   
WHEN datepart(month,@Date) between 7 and 12   
THEN '2'   
END   
)   
ELSE   
'0'   
END)

\*Retorno el valor   
RETURN @Period   
END

Ejemplo 4:

Create function NumeroALetra(@numero int )   
returns varchar(200)   
as   
begin   
Declare @rango int   
Declare @Resultado varchar(200)   
Declare @ResultAux varchar(200)   
Declare @aux varchar(20)   
set @resultado=''   
set @resultaux=''   
set @aux =convert(varchar,@numero)   
Set @rango= len(@aux)

Si estamos por encima de 1000000 todo vuelve a ser igual que por debajo.   
Trescientos cincuenta y un mil cuatrocientos treinta millones doscientas cincuenta mil tres   
Si estuviese solo la primera parte sería igual, pero sin millones.

Ejemplo 5:

if @rango>6   
begin   
set @Resultado = dbo.NumeroALetra(convert(int,left(@aux,len(@aux)-6))) + ' millones '   
set @aux=right(@aux,6)   
end   
-- por aquí el rango siempre va a ser <=6 o si es mayor no me importa   
 set @aux = convert(varchar,convert(int,@aux))   
-- si es 000345 solo interesa la última parte...   
set @rango=len(@aux)   
if @rango>3   
begin   
set @Resultado =isnull(@resultado,'')+   
dbo.NumeroALetra(convert(int,left(@aux,len(@aux)-3))) + ' mil '   
set @aux=right(@aux,3)   
end   
else   
 set @aux = right('000'+@aux,3) -- necesito que tenga tres dígitos para que   
-- el algoritmo funcione bien....   
 select @resultAux =   
case left(@aux,1)   
when '1' then 'Ciento '   
when '2' then 'Doscientos '   
when '3' then 'Trescientos '   
when '4' then 'Cuatrocientos '   
when '5' then 'Quinientos '   
when '6' then 'Seiscientos '   
when '7' then 'Setecientos '   
when '8' then 'Ochocientos '   
when '9' then 'Novecientos '   
else ''   
end   
set @aux=right(@aux,2)

Ejemplo 6:

if convert(int,@aux)<16   
begin   
select @resultaux = isnull(@resultaux,'') +   
case convert(int,@aux)   
when 1 then 'Uno'   
when 2 then 'Dos'   
when 3 then 'Tres'   
when 4 then 'Cuatro'   
when 5 then 'Cinco'   
when 6 then 'Seis'   
when 7 then 'Siete'   
when 8 then 'Ocho '   
when 9 then 'Nueve '   
when 10 then 'Diez '   
when 11 then 'Once '   
when 12 then 'Doce '   
when 13 then 'Trece '   
when 14 then 'Catorce '   
when 15 then 'Quince '   
else ''   
end   
end   
else -- es un número de dos cifras >15   
begin   
select @resultaux=isnull(@resultaux,'')+   
case left(@aux,1)   
when '1' then 'Diez '   
when '2' then 'Veinte '   
when '3' then 'Treinta '   
when '4' then 'Cuarenta '   
when '5' then 'Cincuenta '   
when '6' then 'Sesenta '   
when '7' then 'Setenta '   
when '8' then 'Ochenta '   
when '9' then 'Noventa '   
else ''   
end

Ejemplo 7:

if right(@aux,1)<>'0'   
begin   
if left(@aux,1)='1'   
set @resultaux=left(@resultaux,len(@resultaux)-1)+'ci'   
else   
-- Esto de arriba es el arreglo para poner diez o dieciseis.....   
set @resultaux = @resultaux + 'y '   
end   
select @resultaux=isnull(@resultaux,'')+   
case right(@aux,1)   
when '1' then 'Uno'   
when '2' then 'Dos'   
when '3' then 'Tres'   
when '4' then 'Cuatro'   
when '5' then 'Cinco'   
when '6' then 'Seis'   
when '7' then 'Siete'   
when '8' then 'Ocho '   
when '9' then 'Nueve '   
else ''   
end   
end   
set @resultado=isnull(@resultado,'') +isnull(@resultaux,'')   
   
if @resultado='' set @resultado='Cero.'   
return @resultado   
end

Funciones escalares

Creación:

CREATE FUNCTION fn\_MultiplicaSaldo ( @NumCuenta VARCHAR(20),@Multiplicador DECIMAL(10,2))

RETURNS DECIMAL(10,2)

AS

BEGIN

DECLARE @Saldo DECIMAL(10,2),

@Return DECIMAL(10,2)

SELECT @Saldo = SALDO FROM CUENTAS

WHERE NUMCUENTA = @NumCuenta

SET @Return = @Saldo \* @Multiplicador

RETURN @Return

END

La función ejecutará sus sentencias SELECT una vez por cada fila del conjunto de resultados devuelto por la consulta SELECT principal.

SELECT IDCUENTA, NUMCUENTA, SALDO, FXALTA,

-- Ejecucion de la funcion:

dbo.fn\_MultiplicaSaldo( NUMCUENTA, IDCUENTA) AS RESULTADO

FROM CUENTAS

Ejemplo:

DECLARE @NumCuenta VARCHAR(20),

@Resultado DECIMAL(10,2)

SET @NumCuenta = '200700000001'

SET @Resultado = dbo.fn\_MultiplicaSaldo(@NumCuenta, 30.5)

PRINT @Resultado

Funciones en línea

CREATE FUNCTION fn\_MovimientosCuenta

( @NumCuenta VARCHAR(20) )

RETURNS TABLE

AS

RETURN

(

SELECT MOVIMIENTOS.\*

FROM MOVIMIENTOS

INNER JOIN CUENTAS ON MOVIMIENTOS.IDCUENTA = CUENTAS.IDCUENTA

WHERE CUENTAS.NUMCUENTA = @NumCuenta

)

Ejecución:

SELECT \* FROM fn\_MovimientosCuenta('200700000001')

SELECT \* FROM CUENTAS

Funciones en línea de múltiples sentencias

Ejemplo 1:

CREATE FUNCTION fn\_CuentaMovimietos()

RETURNS @datos TABLE

( -- Estructura de la tabla que devuelve la funcion.

NumCuenta varchar(20),

Saldo decimal(10,2),

Saldo\_anterior decimal(10,2),

Saldo\_posterior decimal(10,2),

Importe\_Movimiento decimal(10,2),

FxMovimiento datetime

) AS

BEGIN

-- Variables necesarias para la lógica de la funcion.

DECLARE @idcuenta int,

@numcuenta varchar(20),

@saldo decimal(10,2)

-- Cursor con las 3 cuentas de mayor saldo

DECLARE CDATOS CURSOR FOR

SELECT TOP 3 IDCUENTA, NUMCUENTA, SALDO

FROM CUENTAS

ORDER BY SALDO DESC

OPEN CDATOS

FETCH CDATOS INTO @idcuenta, @numcuenta, @saldo

-- Recorremos el cursor

WHILE (@@FETCH\_STATUS = 0)

BEGIN

-- Insertamos la cuenta en la variable de salida

INSERT INTO @datos

(NumCuenta, Saldo)

VALUES

(@numcuenta, @saldo)

-- Insertamos los tres últimos movimientos de la cuenta

INSERT INTO @datos (Saldo\_anterior, Saldo\_posterior, Importe\_Movimiento, FxMovimiento )

SELECT TOP 3

SALDO\_ANTERIOR, SALDO\_POSTERIOR,

IMPORTE, FXMOVIMIENTO

FROM MOVIMIENTOS

WHERE IDCUENTA = @idcuenta

ORDER BY FXMOVIMIENTO DESC

-- Vamos a la siguiente cuenta

FETCH CDATOS INTO @idcuenta, @numcuenta, @saldo

END

CLOSE CDATOS;

DEALLOCATE CDATOS;

RETURN

END

Ejecución: select \* from fn\_CuentaMovimietos()