Vistas

```
--Crear Vista
CREATE VIEW usuario_hobbie as
Select tu.rut_usuario ,tu.nombre_usuario,th.nombre_hobbie
From tbl_usuario tu,tbl_usuario_hobbie tuh, tbl_hobbie th
Where tu.rut_usuario=tuh.rut_usuario and th.id_hobbie=tuh.id_hobbie

--Mostrar Vista
Select uh.nombre_usuario,uh.nombre_hobbie
From usuario_hobbie uh
```

Triggers

```
--Crear Gatillador
--INSERT--
Create Trigger gatillo_log
On tbl_usuario
After insert
AS
--Analogia a llaves de inicio y cierre (begin,end)
Begin

-- Se declaran variables
Declare @rut varchar(10)
Declare @nombre varchar(50)

--Asignar valor a las variables desde la tabla clon
correspondiente(inserted,deleted)
Set @rut    =(Select rut_usuario From inserted)
Set @nombre =(Select nombre_usuario From inserted)

--Insertamos los valores en el
log(accion=I(Insert),D(Delete),U(Update))
Insert tbl_log (rut_usuario,fecha_log,nombre_nuevo,accion)
Values(@rut,GETDATE(),@nombre,'I')

End
```

```
--DELETE--
Create Trigger gatillo_log2
On tbl_usuario
After delete
AS
--Analogia a llaves de inicio y cierre (begin,end)
Begin

-- Se declaran variables
Declare @rut varchar(10)
Declare @nombre varchar(50)

--Asignar valor a las variables desde la tabla clon
correspondiente(inserted,deleted)
Set @rut    =(Select rut_usuario From deleted)
Set @nombre =(Select nombre_usuario From deleted)

--Insertamos los valores en el
log(accion=I(Insert),D(Delete),U(Update))
Insert tbl_log (rut_usuario,fecha_log,nombre_nuevo,accion)
Values(@rut,GETDATE(),@nombre,'D')

End
```

--UPDATE--

```
Create Trigger gatillo_log3
On tbl_usuario
After Update
AS
--Analogia a llaves de inicio y cierre (begin,end)
Begin

-- Se declaran variables
Declare @rut varchar(10)
Declare @nombre_ant varchar(50)
Declare @nombre_nuev varchar(50)

--Asignar valor a las variables desde la tabla clon
correspondiente(inserted,deleted)
Set @rut               =(Select rut_usuario From inserted)
Set @nombre_ant        =(Select nombre_usuario From deleted)
Set @nombre_nuev =(Select nombre_usuario From inserted)

--Insertamos los valores en el
log(accion=I(Insert),D(Delete),U(Update))
Insert tbl_log
(rut_usuario,fecha_log,nombre_anterior,nombre_nuevo,accion)
Values(@rut,GETDATE(),@nombre_ant,@nombre_nuev,'U')

End



--Insercion de Datos
Insert into tbl_usuario (rut_usuario,nombre_usuario,fecha_nac_usuario)
Values('1-1','juan','1988-07-30')

--Borrado de Datos
Delete  tbl_usuario Where rut_usuario='1-2'

--Actualizacion de Datos
Update tbl_usuario set nombre_usuario='mark' where rut_usuario='1-1'

--Mostrar log
Select * from tbl_log

--Mostrar usuario
Select * from tbl_usuario
```

Procedimiento Almacenado

```
-- crear Create Procedure
alter Procedure sp_muestra_cliente @rut varchar(10) as
select rut_usuario,nombre_usuario  from tbl_usuario where
rut_usuario=@rut

--executar procedimiento
execute  sp_muestra_cliente '1-1'

Create Procedure sp_inserta_cliente @rut varchar(10),@nomb
varchar(50),@fecha date as

Insert into tbl_usuario (rut_usuario,nombre_usuario,fecha_nac_usuario)
values(@rut,@nomb,@fecha)

execute  sp_inserta_cliente '1-5','juan','2018-02-02'

--exists true o false si existe o no

Create Procedure sp_inserta_exist @rut varchar(10),@nomb
varchar(50),@fecha date as
if not(exists (Select rut_usuario from tbl_usuario where
rut_usuario=@rut))
begin
Insert into tbl_usuario (rut_usuario,nombre_usuario,fecha_nac_usuario)
values(@rut,@nomb,@fecha)
end

--borrar usuario siempre q tenga mas de un hobbie
--cast(@rut)
--print para imprimir datos al diseñador


--crear prodecimiento que elimina mas de un hobbie
Create Procedure sp_e_mult_hobbie as
declare @rut varchar(10)
set @rut=(select top 1 rut_usuario from tbl_usuario_hobbie
             group by rut_usuario having count(rut_usuario)>1 )

delete tbl_usuario_hobbie where rut_usuario=@rut
delete tbl_usuario where rut_usuario=@rut

execute sp_e_mult_hobbie
```

if Exists

--preocedimiento almacenado
--insertar venta
--borrar venta si existe y si no insertarla
--SUMATORIA DE VENTA
--borrar, recorrer cada registro y actualizar

```
ALTER Procedure insertar  @id_prod int,@cantidad          int AS
declare @id_venta     int

IF NOT EXISTS (SELECT top 1 tv.id_venta FROM tbl_venta tv order by tv.id_venta desc)
BEGIN
SET @id_venta=1;
        IF NOT EXISTS (SELECT td.id_prod FROM tbl_detalle td where
td.id_prod=@id_prod)
        BEGIN
        IF ((select tp.stock from tbl_producto tp where tp.id_prod=@id_prod)>=@cantidad)
                BEGIN
                Insert into tbl_venta    (id_venta,fecha_venta)
values(@id_venta,GETDATE());
                Insert into tbl_detalle (id_prod,cantidad,id_venta)
values(@id_prod,@cantidad,@id_venta);
                UPDATE tbl_producto SET stock=(stock-@cantidad) where
id_prod=@id_prod
                END
                ELSE
                BEGIN
                PRINT 'PRODUCTO SIN STOCK SUFICIENTE'
                END
        END
        ELSE
        BEGIN
        PRINT 'PRODUCTO YA EXISTE EN DETALLE';
        END
END
ELSE
BEGIN
SET @id_venta=(SELECT top 1 tv.id_venta FROM tbl_venta tv order by tv.id_venta
desc)+1;
        IF NOT EXISTS (SELECT td.id_prod FROM tbl_detalle td where
td.id_prod=@id_prod)
        BEGIN
```

```
                IF ((select tp.stock from tbl_producto tp where
tp.id_prod=@id_prod)>=@cantidad)
                BEGIN
                Insert into tbl_venta    (id_venta,fecha_venta)
values(@id_venta,GETDATE());
                Insert into tbl_detalle (id_prod,cantidad,id_venta)
values(@id_prod,@cantidad,@id_venta);
                UPDATE tbl_producto SET stock=(stock-@cantidad) where
id_prod=@id_prod
                END
                ELSE
                BEGIN
                PRINT 'PRODUCTO SIN STOCK SUFICIENTE'
                END
        END
        ELSE
        BEGIN
        PRINT 'PRODUCTO YA EXISTE EN DETALLE';
        END
END

EXEC insertar 4,30;
```

Cursor

```
--crear prodecimiento almacenado SP
--Create
Alter Procedure SP_cursores2 as

--declarar variables iguales a los campos que almacenaremos
declare @rut varchar(10);
declare @nomb varchar(50);
declare @ape varchar(50);

--declarar cursor
declare MFC cursor for
--realizar select a recorrer
Select  tu.rut,nomb,tu.ape
From    tbl_usuario tu,tbl_sueldo ts
Where tu.rut=ts.rut
group by tu.rut,tu.nomb,tu.ape

--abrimos cursor
Open MFC

--indicamos al cursor que avance un espacio (desde nada al primer registro)
--y almacene en las variables
Fetch MFC Into @rut,@nomb,@ape

--Creamos ciclo
While(@@FETCH_STATUS=0)
Begin
--Imprimimos
        Print 'nombre: '+@nomb+' apellido: '+@ape

        --ejecutamos procedimiento almacenadoque solicita rut
        execute SP_cursores3 @rut

--avanza un registro
Fetch MFC Into @rut,@nomb,@ape

End

--cerramos cursor
Close MFC
--destruimos cursor
Deallocate MFC
```

Cursor dependiente a través de un procedimiento

--procedimiento que ira dentro del procedimiento anterior
--(es como un metodo dentro de otro en orientacion a objetos)

```
Alter Procedure SP_cursores3 @rut varchar(10) as

declare @anno int;
declare @sueldoB int;
declare @sueldoL int;
--declarar cursor
declare MFC2 cursor for
--realizar select a recorrer
Select  ts.anno,sum(ts.monto),sum(ts.monto-(ts.monto*0.185))
From    tbl_usuario tu,tbl_sueldo ts
Where tu.rut = ts.rut and tu.rut=@rut
group by ts.anno
--abrir cursor
Open MFC2
--iniciamos cursor en posicion 1
Fetch MFC2 Into @anno,@sueldoB,@sueldoL

While(@@FETCH_STATUS=0)
Begin
Print 'año: '+CAST(@anno as varchar(12))+' sueldo bruto: '+ CAST(@sueldoB as
varchar(12)) +' sueldo liquido :'+CAST(@sueldoL as varchar(12))
Fetch MFC2 Into @anno,@sueldoB,@sueldoL
end
--cerramos cursor2
Close MFC2

Deallocate MFC2


--ejecutamos metodo
--el prodecimiento SP_cursores2contiene al SP_cursores3
Execute SP_cursores2
```