

1. Listar las deudas de aquellos contribuyentes que tienen deudas por concepto de arbitrios, estos son aquellos cuyo código de tributo (cCod\_Trib) empieza con 007, 008, 022 y que viven en Asentamientos Humanos (nombre de lugar empieza con A.H.). No debe considerar aquellos que sólo tienen deudas de gastos. Utilice la tabla tDeudas y las adicionales que crea necesarias. Debe mostrar, código, nombre de contribuyente, nombre de lugar y total de deuda.

```
SELECT c.cCod_Cont Codigo,
       c.cNombre Nombre,
       l.cNombre Lugar,
       (d.nTributo+d.nReajuste+d.nInteres+d.nGasto) Deuda
FROM tDeudas d
INNER JOIN tContribuyente c ON d.cCod_cont = c.cCod_Cont
INNER JOIN tLugares l ON c.cCod_Lug = l.cCod_Lug
WHERE (cCod_Trib like '007%'
       OR cCod_Trib like '008%'
       OR cCod_Trib like '022%')
AND l.cNombre like 'A.H.%'
AND NOT (d.nTributo+d.nReajuste+d.nInteres) = 0
```

2. Agregar a la tabla tDeudas la columna nAjuste del tipo decimal (9,2), luego actualizarla poniendo el 0.2% de la suma de nTributo, nReajuste, nInteres, nGasto de aquellos en los que esta suma es mayor a 500.

```
ALTER TABLE tdeudas ADD nAjuste decimal(9, 2)
UPDATE tDeudas
SET nAjuste = (nTributo+nReajuste+nInteres+nGasto)*0.2/100
WHERE (nTributo+nReajuste+nInteres+nGasto) > 500
SELECT *
FROM tDeudas
```

3. En la tabla tImpuesto listar los datos de aquellos cuya calle corresponde a avenidas (empiezan con AV.) cuyo valor afecto (nVal\_Afecto) es menor al valor total (nVal\_Tot) que tienen menos de 3 predios (nNum\_Pred), que corresponden al año 2009 y que el lugar es urbanización, agrupación vecinal o conjunto habitacional (el nombre de lugar empieza con URB., AG.V. o C.H.).

```
SELECT *
FROM tImpuesto i
INNER JOIN tCalles ca ON i.cCod_Calle = ca.cCod_Calle
INNER JOIN tLugares l ON i.cCod_Lug = l.cCod_Lug
WHERE ca.cNombre like 'AV.%'
AND i.nVal_Afecto < i.nVal_Tot
AND i.nNum_Pred < 3
AND i.cAño = '2009'
AND (l.cNombre like 'URB.%'
     OR l.cNombre like 'AG.V.%'
     OR l.cNombre like 'C.H.%')
```

4. Listar el nombre de los contribuyentes y el número de predios de aquellos contribuyentes que en el año 2009 tuvieron más de un predio, esto es más de una fila en la tabla tValuos en ese año.

```
SELECT c.cNombre,  
       count(*) NumPredios  
FROM tValuos v  
INNER JOIN tContribuyente c ON v.cCod_Cont = c.cCod_Cont  
WHERE cAño = '2009'  
GROUP BY c.cNombre  
HAVING count(*) > 1
```

5. Listar los nombres de lugares de los contribuyentes que tienen deudas el año 2009, pero que no tuvieron impuesto (tImpuesto) en ese año.

```
SELECT l.cNombre Lugar  
FROM tDeudas d  
INNER JOIN tContribuyente c ON d.cCod_cont = c.cCod_Cont  
INNER JOIN tLugares l ON c.cCod_Lug = l.cCod_Lug  
WHERE d.cAño = '2009'  
      AND d.cCod_cont not in  
      (SELECT cCod_cont  
       FROM tImpuesto  
       WHERE cAño = '2009' )  
GROUP BY l.cNombre  
ORDER BY 1
```

1. Con la tabla timpuesto, listar los contribuyentes y su dirección, la misma que estará en el siguiente formato: Nombre de Lugar + Nombre de Calle + Número: + cNumero + Mz: + cManzana + Lote:+ clote (ejemplo A.H. LOPEZ ALBUJAR CALLE EGIPTO Mz: H Lote 45), tener en cuenta que Número, Mz y Lote se mostrarán en la dirección siempre y cuando sus respectivos valores no sean null ni estén vacíos. En el ejemplo que se ha puesto se puede apreciar que no se pone número porque o está vacío o es null, también debe mostrar el total de impuesto (nImp\_Pred) y el total de limpieza (nLimp\_Publ). Además, deberá filtrar esta información para que sólo aparezcan aquellos que sólo tienen deudas de tributos (nTributo) y que NO tengan su domicilio en Urbanización (que son los lugares cuyo nombre empieza con URB.). Para ello crear una vista (vimpuestos)

```
CREATE VIEW vImpuestos AS
SELECT c.cNombre,
       rtrim(l.cNombre)+' '+rtrim(ca.cNombre)+ ' ' +
       CASE WHEN (c.cNumero IS NOT NULL AND c.cNumero<> '')
       THEN ' Numero: '+ c.cNumero ELSE '' END +
       CASE WHEN (c.cManzana IS NOT NULL AND c.cManzana<> '')
       THEN ' Mz : ' + c.cManzana ELSE '' END +
       CASE WHEN (c.cLote IS NOT NULL AND c.cLote<> '')
       THEN ' Lote: '+ c.cLote ELSE '' END Direccion,
       sum(i.nImp_Pred) Impuesto, sum(i.nLimp_Publ) Limpieza
FROM timpuesto i
INNER JOIN tContribuyente c ON i.cCod_Cont = c.cCod_Cont
INNER JOIN tLugares l ON i.cCod_Lug = l.cCod_Lug
INNER JOIN tCalles ca ON c.cCod_Calle = ca.cCod_Calle
WHERE i.cCod_Cont in
      (SELECT DISTINCT cCod_cont
       FROM tDeudas
       WHERE nTributo > 0)
      AND NOT left(l.cNombre, 4) = 'URB.'
GROUP BY c.cNombre,
         l.cNombre,
         ca.cNombre,
         c.cNumero,
         c.cManzana,
         c.cLote
```

```
SELECT *
FROM vImpuestos
```

--Nota: No se usa join con la tabla tDeudas, ya que ello haría que salgan más filas por cada fila de deuda, por ello se usa subconsulta

2. Se quiere determinar un listado de los nombres de contribuyentes a los cuales no se les calculó valores (no tienen registros en tValue) pero si tuvieron cálculos de Impuestos (timpuesto). El listado debe mostrar el código del contribuyente y su nombre, si hubieran repetidos deberá mostrar sólo un registro. Usar algún tipo de Join para la solución. No usar subconsultas

```
SELECT DISTINCT c.cCod_Cont,
               c.cNombre
FROM timpuesto i
INNER JOIN tContribuyente c ON i.cCod_Cont = c.cCod_Cont
LEFT JOIN tValuos v ON i.cCod_Cont = v.cCod_Cont
WHERE v.cCod_Cont IS NULL
--Otra forma (usando EXCEPT), aunque menos eficiente
SELECT DISTINCT c.cCod_Cont,
               c.cNombre
FROM timpuesto i
INNER JOIN tContribuyente c ON i.cCod_Cont = c.cCod_Cont
EXCEPT
SELECT c.cCod_Cont,
       c.cNombre
FROM tContribuyente c
INNER JOIN tValuos v ON c.cCod_Cont = v.cCod_Cont
```

3. En la siguiente sentencia: `Select t1.codpersona from tabla 1 t1 Left join tabla2 t2 ON t1.codpersona = t2.codpersona`  
Daría el mismo resultado si en lugar de poner `Select t1.codpersona` ponemos `Select t2.codpersona`? Explique

--Es posible que no dé el mismo resultado si es que no hay integridad y puede que exist a un codpersona que esté en la tabla1 no esté en la tabla 2, con lo cual t2.codpersona devolvería NULL en el caso de que hubiera integridad referencial, devolvería mismo

4. Crear una tabla `tImpuestoValuo`, que contenga las siguientes columnas: `cCod_Cont`, `cNombre`, `impuesto`, `valuo`. Luego llenar esta tabla con todos los datos de códigos y nombres de `tContribuyente`, `impuesto` con `nImp_Pred` (`tImpuesto`), `valuo` con `nVal_Tot` (`tValuo`), estos dos últimos valores que corresponden al año 2008. En los casos en donde no haya impuesto ni valúo deberá colocar cero.

```
CREATE TABLE tImpuestoValuo (cCod_cont char(11), cNombre varchar(100),  
                             Impuesto numeric(15, 2), valuo numeric(15, 2))
```

```
INSERT INTO tImpuestoValuo  
SELECT c.cCod_Cont,  
       c.cNombre,  
       isnull(i.nImp_Pred, 0.00), isnull(v.nVal_Tot, 0.00)  
FROM tContribuyente c  
LEFT JOIN tImpuesto i ON c.cCod_Cont = i.cCod_Cont  
AND i.cAño = '2008'  
LEFT JOIN tValuos v ON c.cCod_Cont = v.cCod_Cont  
AND v.cAño = '2008'
```

--Nota: En el año 2008 no hay registros, si prueban con el 2009 sí salen  
--Si no usan isnull, pueden luego hacer update cambiando a 0.00 todos  
--los registros que tengan valor null

5. En la tabla `tImpuesto` se quiere actualizar la columna `nVal_Tot` en el año 2008 con la suma de valores (`nVal_Tot`) de la tabla `tValuos` para cada código de contribuyente en ese año, tener en cuenta que en `tValuos` un contribuyente puede tener más de un registro en un mismo año. Escriba un script que permita hacer eso.

```
--Creo primero una tabla temporal  
SELECT cCod_Cont, sum(nVal_Tot) valuo  
INTO #valuos  
FROM tValuos  
WHERE cAño = '2008'  
GROUP BY cCod_Cont
```

```
--Actualizo los datos de la tabla usando la tabla temporal  
UPDATE tImpuesto  
SET nVal_Tot = #valuos.valuo  
FROM #valuos  
WHERE tImpuesto.cCod_Cont = #valuos.cCod_Cont  
AND tImpuesto.cAño = '2008'
```

```
--Otra forma  
UPDATE tImpuesto  
SET nVal_Tot =  
    (SELECT sum(nVal_Tot)  
     FROM tValuos  
     WHERE tValuos.cCod_Cont = tImpuesto.cCod_Cont  
     AND tValuos.cAño = '2008')  
WHERE tImpuesto.cAño = '2008'
```

--Nota: No hay registros en la BD con el año 2008, pueden probar con el 2009

1. Crear una función fncEstado que muestre el estado de un contribuyente de acuerdo a lo siguiente: concatenar los textos VALUO, IMPUESTO, DEUDAS para indicar si tienen o no registros en las tablas tValuos, tImpuesto y tDeudas, respectivamente y en un año determinado. Por ejemplo, si un contribuyente tiene registros en las tres tablas, la función devolverá VALUO, IMPUESTO, DEUDA, si sólo tiene en tValuos y tDeudas devuelve VALUO, DEUDA y de forma similar con las otras posibles combinaciones. Evidentemente, si no tiene registro en ninguna tabla devolverá una cadena vacía. A la función se le pasa como parámetros el código del contribuyente y el año. Luego con un select mostrar los nombres de los contribuyentes y su estado en el año 2010, usando la función fncEstado.

```
CREATE FUNCTION [dbo].[fncEstado] (  
    -- Add the parameters for the function here  
    @codigo char(11), @año char(4))  
RETURNS varchar(25)  
AS  
BEGIN  
    -- Declare the return variable here  
    DECLARE @cadena varchar(25)  
    SET @cadena = '' IF  
        (SELECT count(*)  
         FROM tValuos  
         WHERE cCod_cont = @codigo  
               AND cAño = @año) > 0  
    SET @cadena = @cadena + ' VALUO,' IF  
        (SELECT count(*)  
         FROM tImpuesto  
         WHERE cCod_cont = @codigo  
               AND cAño = @año) > 0  
    SET @cadena = @cadena + ' IMPUESTO,' IF  
        (SELECT count(*)  
         FROM tDeudas  
         WHERE cCod_cont = @codigo  
               AND cAño = @año) > 0  
    SET @cadena = @cadena + ' DEUDA'  
    RETURN @cadena  
END  
  
--Probamos con un contribuyente  
SELECT cNombre, dbo.fncEstado(cCod_Cont, '2010') Estado  
FROM tContribuyente  
WHERE cCod_Cont = '0002112'
```

2. En la Base de Datos Tributos se quiere implementar la posibilidad de que algunos contribuyentes puedan efectuar algún reclamo sobre las deudas que tienen pendientes. Estos reclamos son presentados por tipo de tributo (3 primeros caracteres de cCod\_Trib) y por año. Para ello se le solicita a usted crear una tabla (tReclamos), que permita insertar los registros de reclamos. Crear el procedimiento almacenado paInsertaReclamos, que permita efectuar la inserción de registros, validando además que la deuda que se está reclamando exista en tDeudas, si no existe simplemente no hace la inserción. A este procedimiento se le pasará como parámetro de entrada el código del contribuyente, el año y el código de tributo.

```
CREATE TABLE tReclamos (codigo char(11), codTrib char(3), año char(4), dFecReclamo date)
CREATE PROCEDURE paInsertaReclamos @codigo char(11), @codtrib char(3),
                                   @año char(4), @fecReclamo date
AS
BEGIN
    IF (SELECT count(*)
        FROM tDeudas
        WHERE cCod_cont = @codigo
        AND left(cCod_Trib, 3) = @codtrib
        AND cAño = @año) > 0

        INSERT INTO tReclamos (codigo, codTrib, año, dFecReclamo)
        VALUES (@codigo,
                @codtrib,
                @año,
                @fecReclamo)
END
```

3. Crear una vista vDeudasReclamadas que permita listar el código de contribuyente, su nombre, el tipo de tributo (3 primeros caracteres), el año y el total de deuda, además una columna Reclamada, que indique SI, para las que estén reclamadas y NO, en caso contrario. Para esto último se sugiere crear una función que valide si encuentra datos en la tabla tReclamos creada anteriormente.

```
CREATE FUNCTION [dbo].[fncTieneReclamo]
    (@codigo char(11), @codtrib char(3), @año char(4))
RETURNS char(2)
AS
BEGIN
    -- Declare the return variable here
    DECLARE @rspta char(2)
    SET @rspta = 'NO'
    IF (SELECT count(*)
        FROM tReclamos
        WHERE cCod_cont = @codigo
        AND cAño = @año) > 0
        SET @rspta = 'NO' RETURN @rspta
END

CREATE VIEW vDeudasReclamadas
AS
SELECT d.cCod_cont, c.Nombre, LEFT(d.cCod_Trib, 3) TipoTrib,
       d.cAño, sum(d.nTributo+d.nInteres+d.nReajuste+d.nGasto) Total,
       dbo.fncTieneReclamo(d.cCod_cont, left(d.cCod_Trib, 3), d.cAño)
FROM tDeudas d
INNER JOIN tContribuyente c
```

## Examen Final 2023

1. Elaborar un procedimiento almacenado spCalculaReajuste que permita calcular el reajuste de deudas de los contribuyentes desde la tabla tDeudas, y actualizar la columna nReajuste. Este reajuste se calcula multiplicando al tributo (nTributo) el factor indicado en la tabla siguiente:

Antigüedad\Tributo	001	007	008	022	Otros
Mayor o igual a 5 años	0.15%	0.14%	0.13%	0.12%	0.11%
Mayor a 2 y menor a 5 años	0.11%	0.10%	0.09%	0.08%	0.07%
Menor o igual a 2 años	0.06%	0.05%	0.04%	0.03%	0.02%

Tributo corresponde a los 3 primeros caracteres de la columna cCod\_Trib. Al procedimiento le debe pasar el código de contribuyente y la fecha de cálculo a partir de la cual se determinará la antigüedad de la deuda, considerando el año(cAño) de la deuda.

```
CREATE PROCEDURE spCalculaReajuste @codigo char(11),
                                   @fecCalculo date
AS
BEGIN
    DECLARE @codtrib char(7), @tributo numeric(9, 2), @año char(4)
    DECLARE @antig smallint, @factor decimal(5, 2)
    --Se requiere el uso de un cursor ya que las deudas pueden ser
    --de diferentes tributos y antigüedades
    DECLARE CURSOR_REC
    CURSOR
    FOR
    SELECT cCod_Trib,
           nTributo,
           cAño
    FROM tDeudas
    WHERE cCod_cont = @codigo OPEN CURSOR_REC FETCH NEXT
    FROM CURSOR_REC INTO @codtrib, @tributo,
                          @año WHILE @@FETCH_STATUS=0
    BEGIN
        SET @factor = 100
        SET @antig = YEAR(@fecCalculo) - CAST(@año AS smallint)
        SET @factor =
        CASE LEFT(@codtrib, 3)
            WHEN '001' THEN
                CASE
                    WHEN @antig >= 5 THEN 0.15
                    WHEN @antig > 2 AND @antig < 5 THEN 0.11
                    WHEN @antig > 0 AND @antig <= 2 THEN 0.06
                END
            WHEN '007' THEN
                CASE
                    WHEN @antig >= 5 THEN 0.14
                    WHEN @antig > 2 AND @antig < 5 THEN 0.10
                    WHEN @antig > 0 AND @antig <= 2 THEN 0.05
                END
            WHEN '008' THEN
                CASE
                    WHEN @antig >= 5 THEN 0.13
                    WHEN @antig > 2 AND @antig < 5 THEN 0.09
                    WHEN @antig > 0 AND @antig <= 2 THEN 0.04
                END
            WHEN '022' THEN
                CASE
                    WHEN @antig >= 5 THEN 0.12
                    WHEN @antig > 2 AND @antig < 5 THEN 0.08
                    WHEN @antig > 0 AND @antig <= 2 THEN 0.03
                END
            ELSE
                CASE
                    WHEN @antig >= 5 THEN 0.11
                    WHEN @antig > 2 AND @antig < 5 THEN 0.07
                    WHEN @antig > 0 AND @antig <= 2 THEN 0.02
                END
        END
        --Hacemos la actualización
        UPDATE tDeudas
        SET nReajuste = @tributo * @factor / 100
        WHERE cCod_cont = @codigo
        AND cCod_Trib = @codtrib
        AND cAño = @año
        FETCH NEXT FROM CURSOR_REC INTO @codtrib, @tributo, @año
    END
    CLOSE CURSOR_REC
    DEALLOCATE CURSOR_REC
END

/*
SELECT * FROM tDeudas where cCod_cont = '0000143'
EXEC spCalculaReajuste '0000143', '2012-01-01'*/
```

2. Se tiene la tabla **tPrecios**, con la estructura indicada abajo. Se pide crear una tabla **tAuditoriaPrecios**, que contenga las columnas iCodProducto int, cMes char(2), cAnio char(4), ValorOld numeric(8,2), ValorNew numeric(8,2), dFecha datetime, cTipo char(1).

En esta tabla deberá almacenar los cambios de la tabla tPrecios , cada vez que se modifiquen el Precio (n014Precio) o el stock mínimo (n014Minimo), para ello crear un disparador trPrecios. Cuando se modifique el stock mínimo se pondrá el valor M, como se puede apreciar de la estructura también debe guardar los valores antiguos y nuevos y los otros datos que se requieren llenar en la tabla tAuditoriaPrecios.

```
tPrecios
i014CodPrecio INT IDENTITY(1,1),
i014CodProducto INT,
c014Mes CHAR(2),
c014Anio CHAR(4),
n014Minimo NUMERIC(8,2),
n014Precio NUMERIC(8,2)
```

```
CREATE TABLE tAuditoriaPrecios
    (icodProducto int, cMes char(2),
    cAnio char(4), ValorOld numeric(8, 2),
    ValorNew numeric(8, 2), dfecha datetime,
    cTipo char(1))

CREATE TRIGGER trPrecios
ON tPrecios
AFTER UPDATE
AS
BEGIN
    IF update(n014Precio)
        INSERT INTO tAuditoriaPrecios (icodProducto, cMes, cAnio, ValorOld, ValorNew, dFecha, cTipo)
        SELECT i.i014CodPrecio, i.c014Mes, i.c014Anio, d.n014Precio, i.n014Precio, getdate(), 'P'
        FROM inserted i
        INNER JOIN deleted d ON i.i014CodPrecio = d.i014CodPrecio

    IF update(n014Minimo)
        INSERT INTO tAuditoriaPrecios (icodProducto, cMes, cAnio, ValorOld, ValorNew, dFecha, cTipo)
        SELECT i.i014CodPrecio, i.c014Mes, i.c014Anio, d.n014Minimo, i.n014Minimo, getdate(), 'M'
        FROM inserted i
        INNER JOIN deleted d ON i.i014CodPrecio = d.i014CodPrecio

END
--select * from tAuditoriaPrecios
```



3. Elaborar un procedimiento almacenado paComparaImpto que muestre un conjunto de resultados en el cual se comparan los impuestos de dos años calculados por lugar. El pa recibe como parámetros de entrada los dos años(año1, año2), que se quieren comparar. Deberá mostrar un conjunto de resultados con las siguientes columnas:

Lugar	Impuesto_Año1	Impuesto_Año2
-------	---------------	---------------

Donde lugar es el nombre del lugar, Impuesto\_Año1 e Impuesto\_Año2 son los impuestos calculados en esos años (nImp\_Pred)

```
CREATE PROCEDURE paComparaImpto @año1 char(4),@año2 char(4)
AS
BEGIN
    --Creamos tablas temporales por año
    SELECT cCod_Lug, sum(nImp_Pred) impuesto
    INTO #tmp1
    FROM tImpuesto
    WHERE cAño = @año1
    GROUP BY cCod_Lug

    SELECT cCod_Lug, sum(nImp_Pred) impuesto
    INTO #tmp2
    FROM tImpuesto
    WHERE cAño = @año2
    GROUP BY cCod_Lug

    --Unimos los codigos de lugar en otra temporal
    --Es necesario porque puede que hayan codigos de lugar
    --que estén en una tabla y no en la otra
    SELECT cCod_Lug
    INTO #tmp3
    FROM #tmp1
    UNION
    SELECT cCod_Lug
    FROM #tmp2

    --Agregamos columnas de impuesto para cada año
    ALTER TABLE #tmp3 ADD imptoanio1 numeric(11, 2), imptoanio2 numeric(11, 2)
    UPDATE #tmp3
    SET imptoanio1 = impuesto
    FROM #tmp1
    WHERE #tmp3.cCod_Lug = #tmp1.cCod_Lug

    UPDATE #tmp3
    SET imptoanio2 = impuesto
    FROM #tmp2 WHERE #tmp3.cCod_Lug = #tmp2.cCod_Lug

    --Mostramos los resultados
    SELECT l.cNombre Lugar,
           isnull(t.imptoanio1, 0) Impuesto_Año1,
           isnull(t.imptoanio2, 0) Impuesto_Año2
    FROM #tmp3 t
    INNER JOIN tLugares l ON t.cCod_Lug = l.cCod_Lug
    ORDER BY 1
END

--exec paComparaImpto '2009', '2010'
```

- Usando la tabla tImpuesto se requieren listar los códigos y nombre de contribuyentes que en el año 2009 tienen deudas en ese mismo año, pero no tienen deudas en el año 2008. Para deudas use la tabla tDeudas.

```
SELECT i.cCod_Cont [codigo], c.cNombre [nombre]
FROM timpuesto i
JOIN tContribuyente c ON i.cCod_Cont = c.cCod_Cont
WHERE year(i.cAño)= 2009
AND i.cCod_Cont in
(SELECT DISTINCT cCod_cont
FROM tDeudas
WHERE year(cAño)= 2009 )
AND i.cCod_Cont not in
(SELECT DISTINCT cCod_cont
FROM tDeudas
WHERE year(cAño)= 2008 )
```

- Listar los codigos de lugares de la tabla tContribuyente que no están en la tabla tLugares y los códigos de lugares de tLugares que no están en tContribuyente. El resultado se debe mostrar en un solo listado.

```
SELECT c.cCod_Lug
FROM tContribuyente c
LEFT JOIN tLugares l ON c.cCod_Lug=l.cCod_Lug
WHERE l.cCod_Lug IS NULL
UNION
SELECT l.cCod_Lug
FROM tContribuyente c
RIGHT JOIN tLugares l ON c.cCod_Lug =l.cCod_Lug
WHERE c.cCod_Lug IS NULL
```

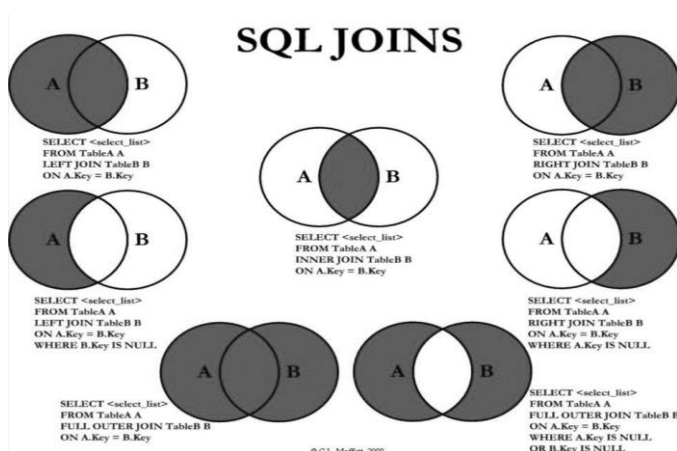


Figure 1 Teoria

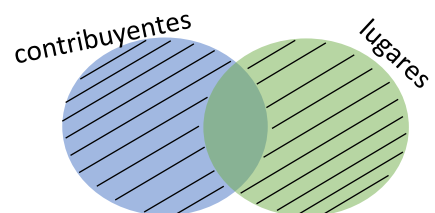


Figure 2 Representacion del Problema

3. Crear una tabla tImpuestoValuo, que contenga las siguientes columnas: cCod\_Cont, cNombre, impuesto, valuo. Luego llenar esta tabla con todos datos de códigos y nombres de tContribuyente, impuesto con nImp\_Pred (tImpuesto), valuo con la suma de nVal\_Tot (tValuo), estos dos últimos valores que correspondan al año 2009. En los casos en donde no haya impuesto o valúo deberá colocar cero.

```
CREATE TABLE tImpuestoValuo(cCod_Cont char(11),
                             cNombre varchar(100),
                             impuesto decimal(15, 2) DEFAULT 0,
                             valuo decimal(15, 2) DEFAULT 0)

INSERT INTO timpuestoValuo(cCod_Cont, cNombre)
SELECT cCod_Cont, cNombre
FROM tContribuyente

SELECT cCod_Cont, nImp_Pred
INTO #impuesto
FROM timpuesto
WHERE year(cAño)=2009

SELECT cCod_Cont, sum(nVal_Tot) total
INTO #valuo
FROM tValuos
WHERE year(cAño)=2009
GROUP BY cCod_Cont

UPDATE i
SET impuesto = nImp_Pred
FROM #impuesto v
JOIN timpuestoValuo i ON i.cCod_Cont=v.cCod_Cont
UPDATE i
SET valuo=total
FROM #valuo v
JOIN timpuestoValuo i ON i.cCod_Cont=v.cCod_Cont
```

4. Usando la tabla tValuos listar el código, nombre y total de valúo de los contribuyentes que en el año 2009 tienen deudas en ese mismo año, pero que no tengan ni intereses ni gastos. Además, sólo debe listar aquellos contribuyentes que tienen más de un predio.

```
SELECT v.cCod_Cont, c.cNombre, sum(v.nVal_Tot) total
FROM tValuos v
JOIN tContribuyente c ON v.cCod_Cont=c.cCod_Cont
WHERE year(v.cAño)= 2009
AND v.cCod_Cont in
(SELECT cCod_cont
FROM tDeudas
WHERE year(cAño)=2009
GROUP BY cCod_cont
HAVING sum(nTributo+nReajuste+nInteres+nGasto)>0)
AND v.cCod_Cont not in
(SELECT cCod_cont
FROM tDeudas
WHERE year(cAño)=2009
GROUP BY cCod_cont
HAVING sum(nInteres+nGasto)=0)
GROUP BY v.cCod_Cont, c.cNombre
HAVING count(*)>1
```

**ORDER BY** total **DESC**

5. Crear una tabla tmpDeudas que contenga el código, nombre y total de deuda de los contribuyentes en el año 2009. Luego agregar un campo Tipo. Ese campo tipo actualizarlo de la siguiente manera: Usando la tabla tImpuesto, si la suma de nImp\_Pred+nLimp\_Publ+nPar\_Jard+nRell\_Sani+nSerenazgo en el año 2009 para ese contribuyente es mayor a 2000 deberá poner PRINCIPAL, si está entre 500 y 2000 poner MEDIANO en caso contrario poner PEQUEÑO.

```
CREATE TABLE tmpDeudas(codigo char(11),
                        nombre varchar(100),
                        totalDeuda decimal(15, 2) DEFAULT 0)

INSERT INTO tmpDeudas(codigo, nombre, totalDeuda)
SELECT c.cCod_Cont,c.cNombre,
       sum(d.nGasto+d.nInteres+d.nReajuste+d.nTributo) [Total Deuda]
FROM tContribuyente c
JOIN tDeudas d ON c.cCod_Cont = d.cCod_cont
GROUP BY c.cCod_Cont,c.cNombre

ALTER TABLE tmpDeudas ADD tipo varchar(20)
```

#### PRIMERA FORMA DE REALIZAR LA ACTUALIZACIÓN

```
UPDATE tmpDeudas
SET tipo = 'PRINCIPAL'
WHERE codigo in
  (SELECT i.cCod_Cont
   FROM timpuesto i
   WHERE year(i.cAño)=2009
   GROUP BY i.cCod_Cont
   HAVING sum(i.nImp_Pred+i.nLimp_Publ+i.nPar_Jard+i.nRell_Sani+i.nSerenazgo)>2000)

UPDATE tmpDeudas
SET tipo = 'MEDIANO' WHERE codigo in
  (SELECT i.cCod_Cont
   FROM timpuesto i
   WHERE year(i.cAño)=2009
   GROUP BY i.cCod_Cont
   HAVING sum(i.nImp_Pred+i.nLimp_Publ+i.nPar_Jard+i.nRell_Sani+i.nSerenazgo)
        BETWEEN 500 AND 2000)

UPDATE tmpDeudas
SET tipo = 'PEQUEÑO' WHERE tipo IS NULL
```

#### SEGUNDA FORMA DE REALIZAR LA ACTUALIZACIÓN

```
UPDATE tmpDeudas
SET tipo= CASE
           WHEN impuestos.total >2000 THEN 'PRINCIPAL'
           WHEN impuestos.total BETWEEN 500 AND 2000 THEN 'MEDIANO'
           ELSE 'PEQUEÑO'
        END
FROM tmpDeudas d
JOIN(
  SELECT i.cCod_Cont,
         sum(nImp_Pred+nLimp_Publ+nPar_Jard+nRell_Sani+nSerenazgo) total
  FROM timpuesto i
  WHERE YEAR(cAño)=2009
  GROUP BY i.cCod_Cont
  HAVING sum(nImp_Pred+nLimp_Publ+nPar_Jard+nRell_Sani+nSerenazgo)>0
) AS impuestos ON d.codigo=impuestos.cCod_Cont

UPDATE tmpDeudas
SET tipo = 'PEQUEÑO'
WHERE tipo IS NULL
```

1. La tabla tValuos, contiene el valor por año de los predios de cada contribuyente, cada predio se identifica por el código catastral (cCod\_Catas) y el valor total del predio está dado por la columna nVal\_Tot. Se requiere crear una vista (vDeudasPredios) que liste por año los códigos de contribuyentes, su nombre, el lugar donde viven, la cantidad de predios y el valor total de los mismos en ese año, estos últimos de la tabla tValuos. Deberá filtrar esta información para que sólo aparezcan aquellos que tienen deudas y que no tengan su domicilio en Asentamientos Humanos.

```
CREATE VIEW vDeudasPredios
([AÑO], [CODIGO], [CONTRIBUYENTE], [LUGAR], [CANTIDAD PREDIOS], [VALOR TOTAL])
AS
SELECT v.cAño, c.cCod_Cont codigo, c.cNombre nombre, l.cNombre lugar,
       count(*) predios,
       sum(v.nVal_Tot) total
FROM tValuos v
JOIN tContribuyente c ON c.cCod_Cont=v.cCod_Cont
JOIN tLugares l ON l.cCod_Lug = c.cCod_Lug
WHERE c.cCod_Cont in
      (SELECT cCod_Cont
       FROM tDeudas
       WHERE (nTributo+nReajuste+nInteres+nGasto)>0)
AND l.cNombre not like 'A.H%'
GROUP BY v.cAño, c.cCod_Cont, c.cNombre, l.cNombre
```

2. Crear la tabla tDeudaAño, que contenga las siguientes columnas: cCod\_Cont char(11), cNombre varchar(100), nDeuda2008, nDeuda2009, nDeuda2010, estas tres de tipo decimal(9,2). Luego deberá actualizar esta tabla para colocar los datos correspondientes usando la tabla tDeudas y tContribuyente, en las columnas nDeuda2008, nDeuda2009, nDeuda2010, deberá colocar las deudas totales correspondientes a esos años. No debe usar PIVOT, sino utilizar tablas temporales en las cuales vaya colocando las deudas de cada año y luego actualizar en la tabla creada. Además debe actualizar la tabla de forma tal que no quede ninguna fila en la que hayan columnas con valores NULL.

```
CREATE TABLE tDeudaAño(
    cCod_Cont char(11), cNombre varchar(100), nDeudas2008 decimal(9, 2) DEFAULT 0,
    nDeudas2009 decimal(9, 2) DEFAULT 0, nDeudas2010 decimal(9, 2) DEFAULT 0)

SELECT d.cCod_Cont,
       sum(d.nTributo+d.nReajuste+d.nInteres+d.nGasto) total
INTO #2008
FROM tDeudas d
WHERE year(d.cAño)=2008
GROUP BY d.cCod_Cont

SELECT d.cCod_Cont,
       sum(d.nTributo+d.nReajuste+d.nInteres+d.nGasto) total INTO #2009
FROM tDeudas d
WHERE year(d.cAño)=2009
GROUP BY d.cCod_Cont

SELECT d.cCod_Cont,
       sum(d.nTributo+d.nReajuste+d.nInteres+d.nGasto) total INTO #2010
FROM tDeudas d
WHERE year(d.cAño)=2010
GROUP BY d.cCod_Cont

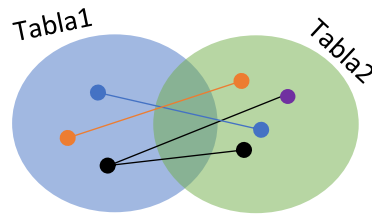
INSERT INTO tDeudaAño(cCod_Cont)
SELECT cCod_Cont
FROM #2008
UNION
SELECT cCod_Cont
FROM #2009
UNION
SELECT cCod_Cont
FROM #2010

UPDATE tDeudaAño
SET cNombre =c.cNombre
FROM tDeudaAño d
JOIN tContribuyente c ON c.cCod_Cont=d.cCod_Cont

UPDATE tDeudaAño
SET nDeudas2008 =a.total
FROM tDeudaAño d
JOIN #2008 a ON d.cCod_Cont=a.cCod_Cont

UPDATE tDeudaAño
SET nDeudas2009 =a.total
FROM tDeudaAño d
JOIN #2009 a ON d.cCod_Cont=a.cCod_Cont
UPDATE tDeudaAño
SET nDeudas2010 =a.total
FROM tDeudaAño d
JOIN #2010 a ON d.cCod_Cont=a.cCod_Cont
```

3. En la siguiente sentencia : `Select t1.codpersona from tabla1 t1 left join tabla2 t2 on t1.codpersona=t2.codpersona`. Daría el mismo resultado si en lugar de poner `Select t1.codpersona` ponemos `Select t2.codPersona` ? Explique



Depende primero de la integridad referencial, ya que el **LEFT JOIN** puede darnos nulos en caso de que no encuentre referencia con ciertos registros en la tabla donde se le referencia, en caso de que un registro no este referenciado en por otro entonces este dara nulo, y asi en viceversa ademas en las consultas con `t1.codpersona` y `t2.codPersona` puede ser diferente, ya que cada una de ellas muestra datos de una tabla diferente en el contexto de la consulta **LEFT JOIN**.

4. Se requiere implementar la posibilidad de que algunos contribuyentes puedan efectuar algún reclamo sobre las deudas que tienen pendientes. Estos reclamos son presentados por tipo de tributo (3 primeros caracteres de `cCod_Trib`) y por año. Para ello se le solicita a usted crear una tabla (`tReclamos`), que permita insertar los registros de reclamos. Crear el procedimiento almacenado `paInsertaReclamos`, que permita efectuar la inserción de registros, validando además que la deuda que se está reclamando exista en `tDeudas`, si no existe simplemente no hace la inserción. A este procedimiento se le pasará como parámetro de entrada el código del contribuyente, el año, el código de tributo y el motivo de reclamo.

```
CREATE TABLE tReclamos(cCod_Cont char(11), año char(4), cCod_Trib char(3), motivo text)
SELECT *
FROM tReclamos
```

```
CREATE PROCEDURE paInsertaReclamos
(@cCod_Cont char(11), @año char(4), @cCod_Trib char(3), @motivo text)
AS
BEGIN
    IF EXISTS(SELECT * FROM tDeudas
              WHERE year(cAño)=@año AND cCod_Cont=@cCod_Cont
              AND left(trim(cCod_Trib),3)=@cCod_Trib)
    BEGIN
        INSERT INTO tReclamos
        VALUES (@cCod_Cont,@año,@cCod_Trib,@motivo)
        PRINT 'INSERCIÓN EXITOSA'
    END
    ELSE
    BEGIN
        PRINT 'FALLO LA INSERCIÓN'
    END
END
```