Conve1.prg  
cone = sqlconnect("pres")

\*for i = 1 to vari

\* cade = "replace "+ alltrim(mat(i,1)) + " with ' ' for isnull(" + alltrim(mat(i,1)) + ") = .t."

\* &cade

\*endfor

\*select cod\_client as id\_cliente, "1" as id\_actividad, "1" as id\_tipo, nom\_client as descripcion,;

\*direccion as calle, 0 as numero, "0" as dpto, barrio as barrio, provincia as provincia, ;

\*cod\_postal as codigo\_postal, pais as pais, telefonos as telefono, fax as fax, " " as email,;

\*" " as web, cuit as cuit, iva as iva, ing\_brutos as ing\_brutos, " " as observaciones, ;

\*" " as contacto, " " as cargo\_prof, "1" as id\_forma\_contacto, " " as ref\_comerciales, ;

\*" " as ref\_bancarias, "01/01/2000" as fecha\_alta from bor into cursor cursal

\*use bor

\*go top

\*do while !eof()

\* cadena = "insert into clientes values ('"+cod\_client+"','1','1','" + ;

\* nom\_client+"','"+direccion+"',0,' ','"+barrio+"','"+;

\* provincia+"',"+alltrim(str(cod\_postal))+",'"+pais+"','"+ telefonos;

\* +"','"+fax+"',' ', ' ','" + cuit+"','"+iva+"','"+;

\* ing\_brutos+"',' ',' ',' ','1',' ',' ','01/01/2000')"

\* vari = sqlexec(cone,cadena)

\* sele bor

\* skip

\*enddo

con = sqlconnect("pres")

clear

sele fletes

go top

do while !eof()

cadena = "insert into fletes values (" + alltrim(str(fletes.kmdesde,10,5)) + ", " + ;

alltrim(str(fletes.kmhasta,10,5)) + ", " + alltrim(str(fletes.menores12m,10,5)) + ", " + ;

alltrim(str(fletes.menores12m,10,5)) + ")"

? sqlexec(con,cadena)

skip

enddo   
  
Main.prg  
set exclusive off

set sysmenu automatic

set delete on

set century on

set date to dmy

set decimals to 2

set procedure to programas.prg

DO FORM principal

READ EVENTS  
  
presupuestos.prg  
  
procedure buscarpresupuestos

cadena = "consu\_presupuestos " + alltrim(str(bpresupuestos.id\_obra)) + ", " + ;

alltrim(str(bpresupuestos.version)) + ", 'E'"

tablerevert(.t.,"cpresupuestos")

sqlexec(conpresupuestos,cadena,"cpresupuestos")

cadena = "consu\_clientes " + alltrim(str(cpresupuestos.id\_cliente)) + ", 1, 'E'"

tablerevert(.t.,"cclientes")

sqlexec(conpresupuestos,cadena,"cclientes")

sele cclientes

cadena = "consu\_obras " + alltrim(str(cpresupuestos.id\_obra)) + ", 1, 'E'"

tablerevert(.t.,"cobras")

sqlexec(conpresupuestos,cadena,"cobras")

sele cobras

if thisform.bempresas = .f. && ya esta listo el cursor de empresas

sele cempresas

seek cpresupuestos.id\_empresa

thisform.frame1.page1.comboempresas.value = cempresas.descripcion

endif

if thisform.busuarios = .f. && ya esta listo el cursor de usuarios

sele cusuarios

seek cpresupuestos.id\_usuario

thisform.frame1.page1.combousuarios.value = cusuarios.descripcion

sele cvendedores

seek cpresupuestos.id\_usuario

thisform.frame1.page1.combovendedores.value = cvendedores.descripcion

endif

return

procedure calculoflete

parameters mat

for i=1 to 2

for j=1 to 12

mat(i,j) = 0

endfor

endfor

sfamaco = 0

scamaco = 0

sremaco = 0

sfameco = 0

scameco = 0

sremeco = 0

sfamaba = 0

scamaba = 0

sremaba = 0

sfameba = 0

scameba = 0

sremeba = 0

sfamavm = 0

scamavm = 0

sremavm = 0

sfamevm = 0

scamevm = 0

sremevm = 0

ant1 = 0

ant2 = 0

ant3 = 0

ant4 = 0

ant5 = 0

ant6 = 0

sele ccomputo

or = order()

set order to flete

go top

do while !eof() and ccomputo.individual = .t.

do case

case ccomputo.origen = "CO" and ccomputo.longitud > 12

do individuales with sfamaco, scamaco, sremaco, 1, 24, 27, mat, ant1

case ccomputo.origen = "CO" and ccomputo.longitud <= 12

do individuales with sfameco, scameco, sremeco, 3, 21, 25, mat, ant2

case ccomputo.origen = "BA" and ccomputo.longitud > 12

do individuales with sfamaba, scamaba, sremaba, 5, 24, 27, mat, ant3

case ccomputo.origen = "BA" and ccomputo.longitud <= 12

do individuales with sfameba, scameba, sremeba, 7, 21, 25, mat, ant4

case ccomputo.origen = "VM" and ccomputo.longitud > 12

do individuales with sfamavm, scamavm, sremaba, 9, 24, 27, mat, ant5

case ccomputo.origen = "VM" and ccomputo.longitud <= 12

do individuales with sfamevm, scamevm, sremeba, 11, 21, 25, mat, ant6

endcase

sele ccomputo

skip

enddo

do comunes with mat

return

procedure individuales

parameters sfa, sca, sre, i, tmin, tmax, mat, ant

if ccomputo.unidad\_medida = "M3" or ccomputo.unidad\_medida = "CU"

pu = ccomputo.peso

else

pu = ccomputo.peso \* ccomputo.longitud

endif

cant2 = int(tmax / pu)

aux = ccomputo.cantidad

if pu > tmax && aqui va la carga individual de piezas que se exeden de peso

mat(2,i) = mat(2,i) + ccomputo.totton

else

do while aux > 0

if ant = id\_pieza and pu <= sre and sca >= 1 && puedo seguir cargando el camion anterior

carga = min(aux, int(sre/pu), sca) && carga es lo menos que puedo cargar

mat(2,i) = mat(2,i) + carga \* pu

sre = sre - carga \* pu

mat(2,i+1) = mat(2,i+1) - min(sfa, carga \* pu)

sfa = min(sfa, carga \* pu)

sca = sca - carga

else && tengo que cargar en un nuevo camion

carga = min(aux,cant2,ccomputo.cantxcamion)

mat(2,i) = mat(2,i) + carga \* pu

sre = tmax - (carga \* pu)

sca = ccomputo.cantxcamion - carga

if (carga \* pu) < tmin && hay que computar toneladas falsas

mat(2,i+1) = mat(2,i+1) + (tmin - (carga \* pu))

sfa = tmin - (carga \* pu)

else

sfa = 0

endif

endif

aux = aux - carga

enddo

endif

ant = ccomputo.id\_pieza

return

procedure comunes

parameters mat

c1 = 0

c2 = 0

c3 = 0

c4 = 0

c5 = 0

c6 = 0

sele ccomputo

do while !eof() and ccomputo.individual = .f.

do case

case ccomputo.origen = "CO" and ccomputo.longitud > 12

c1 = c1 + ccomputo.totton

case ccomputo.origen = "CO" and ccomputo.longitud <= 12

c2 = c2 + ccomputo.totton

case ccomputo.origen = "BA" and ccomputo.longitud > 12

c3 = c3 + ccomputo.totton

case ccomputo.origen = "BA" and ccomputo.longitud <= 12

c4 = c4 + ccomputo.totton

case ccomputo.origen = "VM" and ccomputo.longitud > 12

c5 = c5 + ccomputo.totton

case ccomputo.origen = "VM" and ccomputo.longitud <= 12

c6 = c6 + ccomputo.totton

endcase

skip

enddo

mat(1,1) = mat(1,1) + c1

mat(1,3) = mat(1,3) + c2

mat(1,5) = mat(1,5) + c3

mat(1,7) = mat(1,7) + c4

mat(1,9) = mat(1,9) + c5

mat(1,11) = mat(1,11) + c6

resto1 = c1 - (int(c1/27)\*27)

resto2 = c2 - (int(c2/25)\*25)

resto3 = c3 - (int(c3/27)\*27)

resto4 = c4 - (int(c4/25)\*25)

resto5 = c5 - (int(c5/27)\*27)

resto6 = c6 - (int(c6/25)\*25)

if resto1 < 24 and resto1 > 0

mat(1,2) = mat(1,2) + (24 - resto1)

endif

if resto2 < 21 and resto2 > 0

mat(1,4) = mat(1,4) + (21 - resto2)

endif

if resto3 < 24 and resto3 > 0

mat(1,6) = mat(1,6) + (24 - resto3)

endif

if resto4 < 21 and resto4 > 0

mat(1,8) = mat(1,8) + (21 - resto4)

endif

if resto5 < 24 and resto5 > 0

mat(1,10) = mat(1,10) + (21 - resto5)

endif

if resto6 < 21 and resto6 > 0

mat(1,12) = mat(1,12) + (21 - resto6)

endif

return

procedure grabarpresupuesto

parameters nuevo, versionp, err, cone

sele cpresupuestos

replace id\_obra with cobras.id\_obra

replace id\_cliente with cclientes.id\_cliente

replace id\_empresa with cempresas.id\_empresa

replace id\_vendedor with cvendedores.id\_usuario

replace id\_usuario with cusuarios.id\_usuario

sele ccomputo

replace all id\_obra with cpresupuestos.id\_obra

sele ccomputo2

replace all id\_obra with cpresupuestos.id\_obra

sele ccomputoref

replace all id\_obra with cpresupuestos.id\_obra

sele ccomputoadic

replace all id\_obra with cpresupuestos.id\_obra

sele ccomputoadicref

replace all id\_obra with cpresupuestos.id\_obra

err = sqlexec(cone,"begin transaction")

if err != -1 && se pudo comenzar la transaccion

if versionp = .t. && esta haciendo una version

nuever = 0

cadena = "execute nueva\_version " + alltrim(str(cpresupuestos.id\_obra)) + ", ?@nuever "

err = sqlexec(cone,cadena) && con esto nuever queda cargado con la nueva version

sele cpresupuestos

versionant = cpresupuestos.version

replace cpresupuestos.version with nuever

sqln = afields(sqlmatriz)

do sqlgrabar with sqlmatriz, sqln, "cpresupuestos","presupuestos",.t.,"",cone,err

if err != -1 and nuever > 0 && se puedo generar la nueva version

err = grabarcomputos(cone)

else

do traterror with err, "G"

sele cpresupuestos

replace cpresupuestos.vercion with versionant

endif

else

if nuevo = .t. && se esta cargando un nuevo presupuesto

sele cpresupuestos

replace version with 1

sqln = afields(sqlmatriz)

do sqlgrabar with sqlmatriz, sqln, "cpresupuestos","presupuestos",.t.,"",cone,err

if err != -1 && se pudo grabar el presupuesto

err = grabarcomputos(cone)

else

do traterror with err, "G"

sele cpresupuestos

replace cpresupuestos.version with 0

endif

else && esta modificando un presupuesto

condicion = "id\_obra = " + alltrim(str(cpresupuestos.id\_obra)) + " and version = " + ;

alltrim(str(cpresupuestos.version))

sele cpresupuestos

sqln = afields(sqlmatriz)

do sqlgrabar with sqlmatriz, sqln, "cpresupuestos","presupuestos",.f.,condicion,cone,err

if err != -1 && se pudo modificar el presupuesto

err = borrarcomputos(cone)

if err != -1 && se puedieron borrar los computos anteriores

err = grabarcomputos(cone)

endif

else

do traterror with err, "G"

endif

endif

endif

if err != -1 && no fallaron las operaciones

err = sqlexec(cone,"commit transaction")

if err = -1

do traterror with err, "G"

endif

endif

endif

if err = -1 && fallo alguna operacion

sqlexec(cone,"rollback")

else

messagebox("Los datos se han grabado correctamente",48,"Resultado de la operacion")

endif

return

function grabarcomputos

parameters cone

err = 0

sele ccomputoadic

go top

sqln = afields(sqlmatriz)

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"DESCRIPCION"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"PRECIOREF"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"TOTAL"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"ORDEN"),1))

sqln = sqln - 4

do while !eof() and err != -1

if ccomputoadic.orden != "z"

replace id\_obra with cpresupuestos.id\_obra

replace version with cpresupuestos.version

do sqlgrabar with sqlmatriz, sqln, "ccomputoadic", "computo\_adicionales",.t., "",cone,err

endif

sele ccomputoadic

skip

enddo

if err != -1 && se pudo grabar el computo de adicionales

sele ccomputo

go top

sqln = afields(sqlmatriz)

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"DESCRIPCION"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"UNIDAD\_MEDIDA"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"PERMITE\_OPCIONAL"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"ANCHO"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"ESPESOR"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"ALTURA"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"VOLUMEN"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"PESO"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"SUP\_CUBIERTA"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"SUP\_ENTREPISO"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"INDIVIDUAL"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"CANTXCAMION"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"CERRAMIENTO"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"REF\_BSAS"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"REF\_CBA"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"REF\_VM"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"TOTTON"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"TOTLONG"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"TOTAL"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"ID"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"ORDEN2"),1))

adel(sqlmatriz,asubscrip(sqlmatriz,ascan(sqlmatriz,"LONGITUD1"),1))

sqln = sqln - 23

do while !eof() and err != -1

if ccomputo.orden2 != "z"

replace id\_obra with cpresupuestos.id\_obra

replace version with cpresupuestos.version

do sqlgrabar with sqlmatriz, sqln, "ccomputo", "computos",.t., "",cone,err

endif

sele ccomputo

skip

enddo

endif

if err = -1

do traterror with err, "G"

endif

return err

function borrarcomputos

parameters cone

err = 0

cadena = "delete from computo\_adicionales where id\_obra = " + ;

+ alltrim(str(cpresupuestos.id\_obra)) + " and version = " + ;

alltrim(str(cpresupuestos.version))

err = sqlexec(cone,cadena)

if err != -1 && se pudo borrar el computo de adicionales viejo para bajar el nuevo

cadena = "delete from computos where id\_obra = " + alltrim(str(cpresupuestos.id\_obra)) +;

" and version = " + alltrim(str(cpresupuestos.version))

err = sqlexec(cone,cadena)

endif

if err = -1

do traterror with err, "G"

endif

return err

Programas.prg  
  
function elegir

parameters base

do form pantelegir with base

return recno(base)

\* esta funcion graba un registro en un atabla de sql server, se le pasan: una matriz con

\* la estructura de la tabla, el numero de campos de la tabla, el nombre del cursor fuente,

\* el nombre de la tabla destino, el tipo(.t. = insert, .f. = update), la condicion de where

\* para hacer el update, el controlador de coneccion sobre el que se hace la operacion y una

\* variable de entrada salida que sirve para retornar si hubo o no error

procedure sqlgrabar

parameters sqlmatriz, sqln, sqlcur, sqltabla, sqltipo, sqlclave, sqlconeccion, err

cadena = ""

if sqltipo = .t. && mnuevo hay que hacer in insert

cadena = "insert into " + alltrim(sqltabla) + " ("

for i = 1 to sqln - 1

cadena = cadena + alltrim(sqlmatriz(i,1)) + ", "

endfor

cadena = cadena + alltrim(sqlmatriz(sqln,1)) + ") values ("

for i = 1 to sqln - 1

cadena = cadena + sqlvalor(sqlmatriz(i,2),sqlmatriz(i,1),sqlcur) + ", "

endfor

cadena = cadena + sqlvalor(sqlmatriz(sqln,2),sqlmatriz(sqln,1),sqlcur) + ")"

else

cadena = "update " + alltrim(sqltabla) + " set "

for i = 1 to sqln -1

cadena = cadena + alltrim(sqlmatriz(i,1)) + " = " + sqlvalor(sqlmatriz(i,2),sqlmatriz(i,1),;

sqlcur) + ", "

endfor

cadena = cadena + alltrim(sqlmatriz(sqln,1)) + " = " + sqlvalor(sqlmatriz(sqln,2),;

sqlmatriz(sqln,1), sqlcur)

if len(alltrim(sqlclave)) != 0

cadena = cadena + " where " + sqlclave

endif

endif

err = sqlexec(sqlconeccion,cadena)

return

\*esta funcion arma una cadena insert para generar en nueva nueva version de presupuesto, en

\*el lugar de version va %%%% porque este simbolo es reemplazado luego en el procedimiento

\*almacenado por el numero de la nueva version que se encuentre en el servidor

procedure sqlcadena

parameters sqlmatriz, sqln, sqlcur, sqltabla, cadena2

posversion = 0

cadena = "insert into " + alltrim(sqltabla) + " ("

for i = 1 to sqln - 1

cadena = cadena + alltrim(sqlmatriz(i,1)) + ", "

if alltrim(sqlmatriz(i,1)) = "VERSION"

posversion = i

endif

endfor

cadena = cadena + alltrim(sqlmatriz(sqln,1)) + ") values ("

for i = 1 to sqln - 1

if i = posversion && es el valor de version

cadena = cadena + "%%%%,"

else

cadena = cadena + sqlvalor(sqlmatriz(i,2),sqlmatriz(i,1),sqlcur) + ", "

endif

endfor

cadena = cadena + sqlvalor(sqlmatriz(sqln,2),sqlmatriz(sqln,1),sqlcur) + ")"

cadena2 = cadena

return

function sqlvalor

parameters sqltipo, sqlcampo, sqlcur

sqlretvalor = ""

sqlvervalor = sqlcur + "." + sqlcampo

do case

case alltrim(sqltipo) = "C"

if isnull(&sqlvervalor) = .t.

sqlretvalor = "''"

else

sqlretvalor = "'" + alltrim(&sqlvervalor) + "'"

endif

case alltrim(sqltipo) = "I"

if isnull(&sqlvervalor) = .t.

sqlretvalor = "0"

else

sqlretvalor = alltrim(str(&sqlvervalor))

endif

case alltrim(sqltipo) = "N"

if isnull(&sqlvervalor) = .t.

sqlretvalor = "0.0"

else

sqlretvalor = alltrim(str(&sqlvervalor,20,10))

endif

case alltrim(sqltipo) = "F"

if isnull(&sqlvervalor) = .t.

sqlretvalor = "0.0"

else

sqlretvalor = alltrim(str(&sqlvervalor,20,10))

endif

case alltrim(sqltipo) = "B"

if isnull(&sqlvervalor) = .t.

sqlretvalor = "0.0"

else

sqlretvalor = alltrim(str(&sqlvervalor,20,10))

endif

case alltrim(sqltipo) = "M"

if isnull(&sqlvervalor) = .t.

sqlretvalor = "''"

else

sqlretvalor = "'" + alltrim(&sqlvervalor) + "'"

endif

case alltrim(sqltipo) = "T"

if isnull(&sqlvervalor) = .t. or empty(&sqlvervalor)

sqlretvalor = "''"

else

sqlretvalor = "'" + dtoc(&sqlvervalor) + "'"

endif

case alltrim(sqltipo) = "L"

if isnull(&sqlvervalor) = .t.

sqlretvalor = "0"

else

sqlretvalor = iif(&sqlvervalor=.t.,"1","0")

endif

endcase

return sqlretvalor

procedure traterror

parameters enro, etipo

nn = aerror(emat)

ecadena = "ERROR NUMERO " + alltrim(str(emat(1))) + chr(13) + alltrim(emat(2)) + chr(13)

do case

case etipo = "G"

ecadena = ecadena + "NO SE PUDO GRABAR"

case etipo = "B"

ecadena = ecadena + "NO SE PUDO BORRAR"

case etipo = "C"

ecadena = ecadena + "NO SE PUDO CONSULTAR"

case etipo = "N"

ecadena = ecadena + "NO SE PUDO COCRETAR LA OPERACION"

endcase

= messagebox(ecadena,48,"ERROR !!!")

return

function copiarcursor

parameters desde, hasta, agregar

set null on

sele (desde)

copycant = afields(copymat)

for i = 1 to copycant

if alltrim(copymat(i,2)) = "B"

copymat(i,2) = "N"

endif

endfor

cadena = "create cursor " + alltrim(hasta)+ " ("

for i = 1 to copycant - 1

cadena = cadena + alltrim(copymat(i,1)) + " " + alltrim(copymat(i,2)) + "(" + ;

alltrim(str(copymat(i,3))) + "," + alltrim(str(copymat(i,4))) + "), "

endfor

cadena = cadena + alltrim(copymat(copycant,1)) + " " + alltrim(copymat(copycant,2)) + "(" + ;

alltrim(str(copymat(copycant,3))) + "," + alltrim(str(copymat(copycant,4))) + ")"

if len(alltrim(agregar)) = 0

cadena = cadena + ")"

else

cadena = cadena + ", " + agregar + ")"

endif

&cadena

return

function compararhojas

ret = .f.

sele chojas\_indicesref

seek chojas\_indices.clave\_pieza

if found()

if alltrim(nombre\_plano) != alltrim(chojas\_indices.nombre\_plano) or alltrim(numero\_plano) != ;

alltrim(chojas\_indices.numero\_plano) or longitud != chojas\_indices.longitud or ;

ancho != chojas\_indices.ancho or volumen != chojas\_indices.volumen or peso != ;

chojas\_indices.peso or peso\_hierros != chojas\_indices.peso\_hierros ;

or alltrim(seccion) != alltrim(chojas\_indices.seccion) or ;

orden != chojas\_indices.orden or producida != chojas\_indices.producida or ;

alltrim(hormigon) != alltrim(chojas\_indices.hormigon) or alltrim(fecha\_produccion) != ;

alltrim(chojas\_indices.fecha\_produccion) or alltrim(ultimo\_movimiento) != ;

alltrim(chojas\_indices.ultimo\_movimiento) or alltrim(origen) != ;

alltrim(chojas\_indices.origen) or alltrim(stoqueada\_en) != ;

alltrim(chojas\_indices.stoqueada\_en) or movida != chojas\_indices.movida or ;

alltrim(memoinutil) != alltrim(chojas\_indices.memoinutil) or ;

inutilizada != chojas\_indices.inutilizada or alltrim(apto\_armadura) != ;

alltrim(chojas\_indices.apto\_armadura) or alltrim(apto\_fabricacion) != ;

alltrim(chojas\_indices.apto\_fabricacion) or observaciones != chojas\_indices.observaciones ;

or alltrim(tipo\_hormigon) != alltrim(chojas\_indices.tipo\_hormigon)

ret = .t.

endif

endif

return ret

function validaranexo

private all

ret = .f.

if !empty(cobras.calle) and !empty(cobras.localidad) and !empty(cobras.provincia) and;

!empty(cclientes.calle) and !empty(cclientes.localidad) and !empty(cclientes.provincia) and;

!empty(cclientes.telefono) and !empty(cproyectistas.calle) and !empty(cproyectistas.localidad);

and !empty(cproyectistas.provincia) and !empty(cproyectistas.telefono) and ;

!empty(cpresupuestos.id\_empresa) and !empty(cpresupuestos.id\_vendedor) and ;

!empty(cpresupuestos.id\_usuario) and !empty(cpresupuestos.fecha\_venta) and ;

!empty(cpresupuestos.plazo\_entrega)

ret = .t.

endif

return ret

function bloqueo

parameters cone, tabla, idcampo, idvalor

ret = .f.

private all

cadena = "select " + alltrim(idcampo) + " from " + alltrim(tabla) + " with (updlock) " + ;

" where " + alltrim(idcampo) + " = " + alltrim(idvalor)

xret = sqlexec(cone, cadena)

if xret = 1

ret = .t.

endif

return ret  
  
salir.prg  
  
procedure salir

if messagebox("Desea salir del sistema? ",36,"Pregunta al usuario") = 6

principal.release

ENDIF

stock.prg  
  
procedure grabarmov

parameters nuevo, tipo, cone, err

if tipo == 'S' or tipo = 'O'

do grabarsalida with nuevo, cone, err, tipo

else

do grabarentrada with nuevo, cone, err

endif

return

procedure grabarsalida

parameters nuevo, cone, err, tipo

err = sqlexec(cone,"begin transaction")

if err != -1 && se pudo comenzar la transaccion

sele csalidas\_stock

if tipo == 'O' && esta grabando una orden de carga

replace ordencarga with .t.

else

replace ordencarga with .f.

endif

replace id\_obra with cobras.id\_obra

sqln = afields(sqlmatriz)

adel(sqlmatriz,ascan(sqlmatriz,"ID\_MOVIMIENTO"))

sqln = sqln - 1

clave = "id\_movimiento = " + alltrim(str(csalidas\_stock.id\_movimiento))

do sqlgrabar with sqlmatriz, sqln, "csalidas\_stock","salidas\_stock",;

nuevo,clave,cone,err

if err != -1 && se pudo grabar en salidas\_stock

if nuevo = .t.

ncodigo = 0

sqlexec(cone,"execute ver\_codigo ?@ncodigo ")

replace csalidas\_stock.id\_movimiento with ncodigo

endif

do grabarstocksal with cone, nuevo, csalidas\_stock.id\_movimiento, err, tipo

if err != -1 && se pudo grabar la tabla stock

err = sqlexec(cone,"commit transaction")

if err != -1 && la trnsaccion ya esta confirmada

delete from chojas\_indices where elegir = .f.

endif

endif

endif

endif

if err = -1 && fallo alguna operacion

do traterror with err, "G"

sqlexec(cone,"rollback")

endif

return

procedure grabarentrada

parameters nuevo, cone, err

err = sqlexec(cone,"begin transaction")

if err != -1 && se pudo comenzar la transaccion

band = .f.

sele chojas\_indices

go top

do while !eof()

if chojas\_indices.elegir = .t. and alltrim(chojas\_indices.ultimo\_movimiento) == "S"

band = .t.

endif

skip

enddo

go top

if band = .t. && hay algo para grabar

sele centradas\_stock

replace id\_obra with csalidas\_stock.id\_obra

replace id\_salida with csalidas\_stock.id\_movimiento

sqln = afields(sqlmatriz)

adel(sqlmatriz,ascan(sqlmatriz,"ID\_MOVIMIENTO"))

sqln = sqln - 1

clave = "id\_movimiento = " + alltrim(str(centradas\_stock.id\_movimiento))

do sqlgrabar with sqlmatriz, sqln, "centradas\_stock","entradas\_stock",;

nuevo,clave,cone,err

if err != -1 && se pudo grabar en entradas\_stock

if nuevo = .t.

ncodigo = 0

sqlexec(cone,"execute ver\_codigo ?@ncodigo ")

replace centradas\_stock.id\_movimiento with ncodigo

endif

do grabarstockent with cone, nuevo, centradas\_stock.id\_movimiento, err

if err != -1 && se pudo grabar la tabla stock

err = sqlexec(cone,"commit transaction")

endif

endif

endif

endif

if err = -1 && fallo alguna operacion

do traterror with err, "G"

sqlexec(cone,"rollback")

endif

return

procedure grabarstockent

parameters cone, nuevo, movi, err

tipo = 'E'

contrario = 'S'

sele chojas\_indices

go top

if nuevo = .f.

do while !eof() and err != -1

if chojas\_indices.inutilizada = .t.

cadena = "update hojas\_indices set inutilizada = " + ;

iif(chojas\_indices.inutilizada = .t., "1","0") + ", memoinutil = '" + ;

alltrim(chojas\_indices.memoinutil) + "' " + "where clave\_pieza = " + ;

alltrim(str(chojas\_indices.clave\_pieza))

err = sqlexec(cone,cadena)

endif

skip

enddo

else

do while !eof() and err != -1

if chojas\_indices.elegir = .t. and alltrim(chojas\_indices.ultimo\_movimiento) == "S"

cadena = "insert into stock (clave\_pieza, id\_movimiento, tipo) values (" + ;

alltrim(str(chojas\_indices.clave\_pieza)) + ", " + alltrim(str(movi)) + ;

", '" + tipo + "')"

err = sqlexec(cone,cadena)

if err != -1

cadena = "update hojas\_indices set ultimo\_movimiento = '" + ;

tipo + "', memoinutil = '" + alltrim(chojas\_indices.memoinutil) +;

"', inutilizada = " + iif(chojas\_indices.inutilizada = .t., "1","0") + ;

", id\_obra = " + alltrim(str(obraorigen)) + ;

" where clave\_pieza = " + alltrim(str(clave\_pieza))

err = sqlexec(cone,cadena)

endif

endif

skip

enddo

endif

return

procedure grabarstocksal

parameters cone, nuevo, movi, err, tipo

contrario = 'E'

if nuevo = .f. && se esta modificando un movimiento

cadena = "update hojas\_indices set ultimo\_movimiento = '" + contrario + ;

"' where clave\_pieza in (select clave\_pieza from stock where id\_movimiento = " + ;

alltrim(str(movi)) + " and tipo = 'S')"

err = sqlexec(cone,cadena)

if err != -1 && se pudieron marcar las piezas

cadena = "delete from stock where id\_movimiento = " + alltrim(str(movi)) + ;

" and tipo = 'S'"

err = sqlexec(cone,cadena)

if err != -1 && se pudo borrar la tabla stock

sele chojas\_indices

go top

do while !eof() and err != -1

if chojas\_indices.elegir = .t.

cadena = "insert into stock (clave\_pieza, id\_movimiento, tipo) values (" + ;

alltrim(str(chojas\_indices.clave\_pieza)) + ", " + alltrim(str(movi)) + ;

", 'S')"

err = sqlexec(cone,cadena)

else

\* es una pieza no elegida, si es de otra hoja, hay que volverla al origen

if chojas\_indices.id\_obra != chojas\_indices.obraorigen

cadena = "update hojas\_indices set id\_obra = " + ;

alltrim(str(chojas\_indices.obraorigen)) + " where clave\_pieza = " + ;

alltrim(str(chojas\_indices.clave\_pieza))

err = sqlexec(cone,cadena)

if err != -1

delete && la pieza se paso a otra obra, con lo que se quita de la hoja indice

endif

endif

endif

skip

enddo

if err != -1 && se pudo grabar todo en la tabla stock

cadena = "update hojas\_indices set ultimo\_movimiento = '" + tipo + ;

"' where clave\_pieza in (select clave\_pieza from stock where id\_movimiento = " + ;

alltrim(str(movi)) + " and tipo = 'S')"

err = sqlexec(cone,cadena)

endif

endif

endif

else && es un movimiento nuevo

sele chojas\_indices

go top

do while !eof() and err != -1

if chojas\_indices.elegir = .t.

cadena = "insert into stock (clave\_pieza, id\_movimiento, tipo) values (" + ;

alltrim(str(chojas\_indices.clave\_pieza)) + ", " + alltrim(str(movi)) + ;

", 'S')"

err = sqlexec(cone,cadena)

else

\* es una pieza no elegida, si es de otra hoja, hay que volverla al origen

if chojas\_indices.id\_obra != chojas\_indices.obraorigen

cadena = "update hojas\_indices set id\_obra = " + ;

alltrim(str(chojas\_indices.obraorigen)) + " where clave\_pieza = " + ;

alltrim(str(chojas\_indices.clave\_pieza))

err = sqlexec(cone,cadena)

if err != -1

delete && la pieza se paso a otra obra, con lo que se quita de la hoja indice

endif

endif

endif

skip

enddo

if err != -1 && se pudo grabar todo en la tabla stock

cadena = "update hojas\_indices set ultimo\_movimiento = '" + tipo + ;

"' where clave\_pieza in (select clave\_pieza from stock where id\_movimiento = " + ;

alltrim(str(movi)) + " and tipo = 'S')"

err = sqlexec(cone,cadena)

endif

endif

return

**Funcionalidades principales:**

**1. Archivo Conve1.prg**

**Objetivo: Migrar datos de clientes y fletes a una base de datos.**

**Partes Clave:**

* **cone = sqlconnect("pres")**:
  + **Qué hace**: Conecta a una base de datos llamada "pres" (como abrir una carpeta de Excel en línea).
  + **Para qué**: Para poder leer/escribir datos en ella.
* **Sección Comentada (Migrar Clientes)**:
  + **Qué hace**: Copia datos de una tabla llamada bor a una nueva tabla clientes, ajustando formatos (ej: dirección, CUIT).
* **Inserción de Fletes**:
  + **Qué hace**: Recorre una tabla fletes y genera comandos SQL para copiar sus registros a otra tabla fletes.
  + **Variables**:
  + cadena: Crea una instrucción SQL como INSERT INTO fletes VALUES (100, 200, 50, 50).
  + sqlexec(con,cadena): Ejecuta ese comando en la base de datos.
  + **Para qué**: Actualizar la tabla de fletes con nuevos rangos de kilómetros y costos.

**2. Archivo Main.prg**

**Objetivo: Configurar el entorno de la aplicación y lanzarla.**

**3. Archivo Presupuestos.prg**

**Objetivo: Gestionar presupuestos (crear, modificar, calcular costos).**

**Procedimientos Clave:**

1. **buscarpresupuestos**:
   * **Qué hace**: Busca información de presupuestos, clientes y obras en la base de datos.

 **calculoflete**:

* **Qué hace**: Calcula el costo de transporte (flete) de materiales.
* **Variables**:
  + mat(i,j): Matriz para guardar costos según tipo de material (ej: acero, madera).
* **Lógica**:
  + Si un material mide más de 12 metros, usa una tarifa; si es menor, otra.

 **grabarpresupuesto**:

* **Qué hace**: Guarda o actualiza un presupuesto en la base de datos.
* **Transacciones**:
  + BEGIN TRANSACTION: Inicia una operación segura (como guardar un borrador).
  + COMMIT: Confirma los cambios si todo sale bien.
  + ROLLBACK: Cancela todo si hay un error.
* **Para qué**: Evita que se guarden datos corruptos.

**4. Archivo Programas.prg**

**Objetivo: Funciones genéricas reutilizables.**

**Funciones Clave:**

1. **sqlgrabar**:
   * **Qué hace**: Construye automáticamente consultas SQL para insertar o actualizar registros.

 **sqlvalor**:

* **Qué hace**: Formatea valores para que sean compatibles con SQL (ej: convierte True a 1).
* **Para qué**: Evitar errores al escribir fechas, números o textos en la base de datos.

 **traterror**:

* **Qué hace**: Muestra un mensaje de error comprensible si algo falla (ej: "No se pudo guardar el presupuesto").
* **Variables**:
  + ecadena: Texto del error (ej: "ERROR: Conexión a la base de datos perdida").

**5. Archivo Salir.prg**

**Objetivo: Cerrar la aplicación de forma segura.**

** grabarsalida:**

* **Qué hace: Registra una salida de materiales y actualiza el inventario.**
* **Variables:**
  + **csalidas\_stock: Tabla temporal con los datos de la salida.**
* **Transacciones:**
  + **Si todo es correcto, actualiza stock y marca los materiales como "enviados".**

** grabarentrada:**

* **Qué hace: Registra una devolución de materiales al inventario.**

**6. Archivo Stock.prg**

**Objetivo: Gestionar movimientos de inventario (entradas y salidas).**

**Procedimientos Clave:**

1. **grabarmov**:
   * **Qué hace**: Decide si registrar una entrada (devolución) o salida (envío) de stock.