Acțiuni asociate machetelor

Pentru a fi mai ușor de înțeles cum funcționează facilitățile de design în zona de acțiuni asociate cu machetele, vom porni de la exemple practice, enumerând enunțuri de probleme și exemplificând soluția de rezolvare.

Tipuri de acțiuni

1 - validare (de exemplu: validare la salvarea facturilor, astfel încât fiecare articol să aibă clasa de articol asociată);
2 - atenționare (de exemplu: atenționare dacă pe factură sunt linii cu Cantitate < 3);
3 - interogare (de exemplu: posibilitatea de a afișa o listă cu valorile facturate pe gestiuni);
4 - interogare (de exemplu: posibilitatea de a afișa o listă cu codurile externe și cele interne pentru fiecare articol din factură);
5 - import date (de exemplu: import rezervare pe factura de ieșire);

Tipuri de SQL

Pentru fiecare caz în parte se înregistrează un SQL. E important să intelegeți ce înseamnă:
a. "**SQL pe baza de date**": acesta folosește sintaxa **Oracle/PosgreSQL** pentru a aduce date tabelele originale (exclus tabelele temporare) din Ora sau PG.
b. "**SQLite pe memorie**": acesta folosește sintaxa **SQLite** aplicată pe tabelele de pe memorie (care încă nu au fost salvate în baza de date), cele ale căror nume și structură le vedeți în optiunea "Tabele WME" pe document.

Mecanismul de lucru cu tabele în WMEnterprise, pentru a întelege diferența între cele două tipuri de SQL, este urmatoarea pentru fiecare document:

- **Oracle/PosgreSQL** se execută fraze SQL care mută date din tabelele originale (ex: IESIRI) în tabelele temporare (ex: TMP_IESIRI);
- **WME** aduce înregistrările din tabelele temporare în niște tabele pe memorie, cu numele de tabelă si de campuri asa cum le vedeti in "Tabele WME";
- tabelele pe memorie au structura identică cu tabelele temporare, dar au alte denumiri, și anume fără prefix TMP_;
- tabelele de pe memorie astfel încărcate, stau în spatele tuturor controalelor vizuale de pe machete și pot fi editate după ce apăsați "Adaug" sau "Modific";
- controalele prin care date sunt introduse la interfața memorează valorile în tabelele de pe memorie;
- la optiunea "Salvez" se rulează algoritmi de atenționare / validare privitor la datele introduse;

- in cazul în care toate verificările de corectitudine a datelor au trecut cu succes, **WME** trimite informațiile din tabelele de pe memorie înapoi în tabelele temporare din **Oracle sau PosgreSQL;**
- după ce au ajuns în temporare (ex: TMP_IESIRI), acestea se codifică unic și ajung în cele din urmă să fie integrate în tabelele originale (ex: IESIRI).

Se înțelege că datele prezente pe memorie înainte de salvare (înainte de a ajunge în **Ora/PG**) nu au cum să fie interogate cu fraze SQL clasice care se aplică doar pentru tabele **Oracle sau PosgreSQL.**
Pentru datele pe memorie se aplică fraze SQL specifice, cu sintaxa de la un motor de baza de date special conceput pentru lucru pe memorie, denumit **SQLite**.

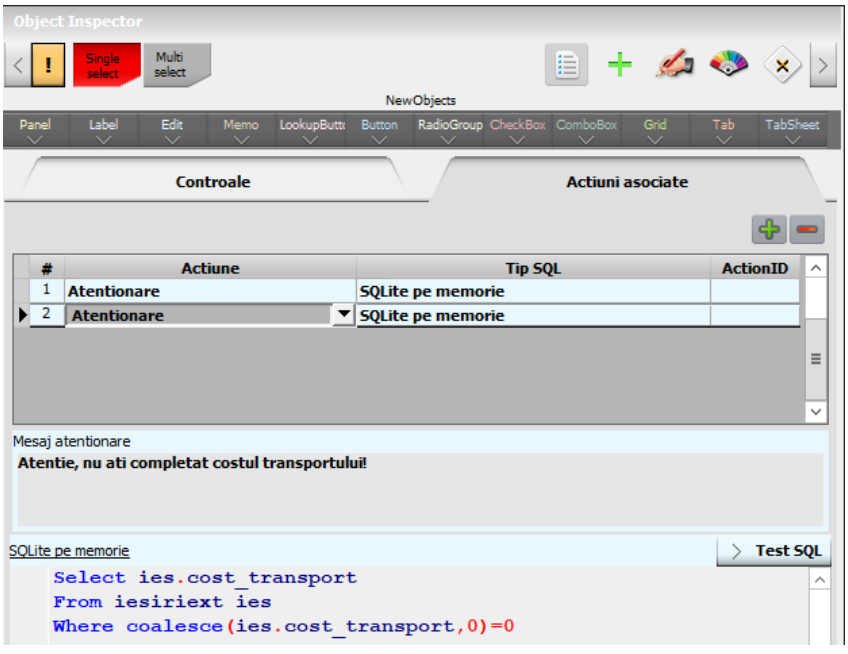
E de înțeles că în SQLite nu au ce cauta tabele din Ora sau PG, acestea nefiind încărcate pe memorie!
SQL pe baza de date folosește excusiv tabele din baza de date Ora/PG, iar SQLite pe memorie folosește exclusiv tabelele din modulul de date (vizualizate prin "Tabele WME").
Cu alte cuvinte vorbim despre 2 dimensiuni paralele, care nu au nicio legătura una cu alta!

(Rev.1.1) Dacă trebuie ceva din baza de date, dar trebuie și o legatură cu datele de pe memorie (pentru ca un SQL care nu are legatura cu documentul tocmai introdus si nepostat nu are nici un sens) se utilizeaza sintaxa **Where Cod = <Tabela.NumeCamp>**

1. SQLite pe memorie

Se foloseste atunci cand informatiile nu sunt inca salvate in baza de date; sql-ul se face pe tabelele de pe memorie, denumirile de campuri si de tabele se iau din tabele WME (iconita de pe doc);

Exemplu: validarea completarii unui camp din extensii la factura de iesire

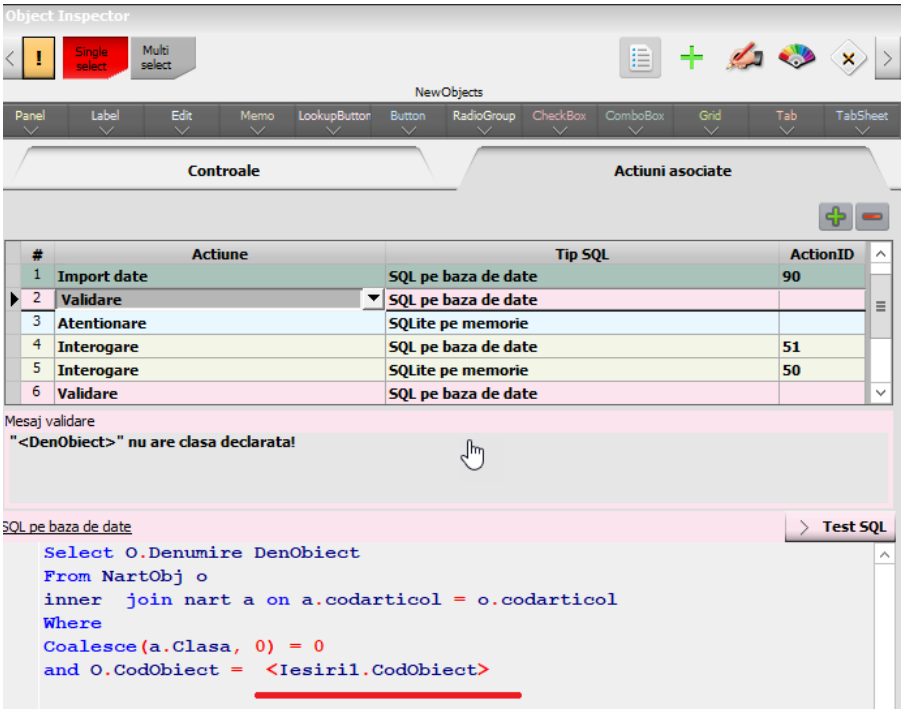


2. Sql pe baza de date

Se foloseste de informatii deja salvate in baza de date; pentru a face referire la campuri de pe document care sunt inca pe memorie, in clauza WHERE a SQL se folosesc valori de pe memorie pe care le accesati asa: <Tabela.NumeCamp>

Inainte de a se executa SQL pe baza de date, in clauza Where Cod = <Tabela.NumeCamp> WME inlocuieste textul <Tabela.NumeCamp> cu valoarea campului NumeCamp din tabla Tabela, asa incat daca Tabela.NumeCamp = 100, atunci clauza devine: Where Cod = 100;

Exemplu: validare la salvarea facturilor, astfel incat fiecare articol sa aiba clasa de articol asociata



Atentie!.
SQLPeBazaDeDate musai contine legatura cu "ceva" de pe memorie, de forma <Tabela.NumeCamp>.
Daca WME constata ca **<Tabela.** din clauza where e tabla Detaliu a documentului, SQL se executa pentru fiecare inregistrare din detaliu, desi in SQL nu se specifica asta. Prin exceptie, cand WME constata ca **<Tabela.** din clauza where e tabla Master a documentului, SQL se executa o singura data.

Despre <Tabela.NumeCamp>, voi exemplifica pe o situatie concreta, facturi iesire. Tabela poate fi IESIRI (master document) sau extensia ei (IesiriEXT) sau IESIRI1 (detaliu pricipal document) sau extensia ei (Iesiri1EXT). Atat, **NU** alte tabele!
Adica:
- poti livra valori din <IESIRI.Camp>, <IesiriEXT.Camp>, caz in care SQL se executa o singura data, inlocuind textul <...> cu valoare campului specificat;

- poti livra valori din <IESIRI1.Camp>, <Iesiri1EXT.Camp>, caz in care SQL se executa pentru fiecare linie din detaliul principal, inlocuind textul <...> cu valoare campului specificat.

Inserare SQLite în SQL pe baza de date

Incepand cu versiunea 25.021 se poate rezolva situatia in care aveti nevoie de un camp din alta tabela de pe memorie:

Dupa cum explicam, de obicei trebuie sa legam in SQLPeBazaDeDate un cod de pe memorie, cam asa **Where Cod = <Tabela.NumeCamp>**, care devine de exemplu **Where Cod = 100**.
Va dau o situatie in care valoarea din conditia WHERE trebuie sa vina musai din alta tabela decat master/extensie master, sau Detaliu/Extensie detaliu.

Enuntul problemei e asa: Pe factura de iesire se aduc linii din comenzi client. Se doreste sa se afiseze termenele de livrare din Comanda1 pentru care se face livrarea.

Asadar, atata vreme cat vreau valori din Comanda1 care e salvata in BD, trebuie sa fac un SQL pe baza de date:

Select C1.Termen from Comanda1 C1 where C1.CodComanda1 = CEVA.

De unde aduc acel CEVA, care e cod de comanda1? Pentru ca nu am in IESIRI1 asa un cod... Ei, acest codcomanda1 se afla in IesLink, care are in structura **CodIes1, CodLinieComanda,** ....Ca sa aflu CodLinieComanda, avand in vedere ca toate valorile sunt pe memorie, trebuie folosit SQLite si SQL ar arata cam asa:

Select CodLinieComanda from IesLink where CodIes1 = X.

Revin: **Select C1.Termen from Comanda1 C1 where C1.CodComanda1 = CEVA**.
Daca **CEVA** este inlocuit cu un SQL intre paranteze patrate, il consider ca e SQLite, se executa in prealabil si va rezulta un CodComanda1, arata deci astfel:

Select C1.Termen from Comanda1 C1 where C1.CodComanda1
= [Select CodLinieComanda from IesLink where CodIes1 = X].

Ei, daca X e <Iesiri1.CodIes1>, atunci **Select CodLinieComanda from IesLink where CodIes1 =** <Iesiri1.CodIes1> va aduce un CodComanda1, iar pentru ca se identifica IESIRI1 intre <...>, SQL de mai jos se executa pentru fiecare linie din Detaliu Iesiri1.

Select C1.Termen from Comanda1 C1 where C1.CodComanda1
= [Select CodLinieComanda from IesLink where CodIes1 = <Iesiri1.CodIes1>].

In concluzie, aceasta fraza SQLpeBazaDeDate face urmatoarele:
Pentru fiecare linie de pe factura, gaseste CodLinieComada la care se leaga, si cu acest cod aduce din Comanda1 campul TERMEN.
In final, la salvare, va aparea un mesaj care afiseaza acest termen de livrare.

Exemple acțiuni pe tipuri

Vă solicit toată ingăduința pentru exemplele care sunt uneori cam puerile, dar important e să intelegeți funcționarea. Pe baza acestor exemple veți putea dezvolta singuri proceduri proprii, pe caz real, după nevoi.

Pe factura de ieșire vom asocia, ca cerințe, următoarele acțiuni (acceptați enunțurile banale fără să judecați!):

1- **validare la salvarea facturilor, astfel încât fiecare articol să aibă clasa de articol asociată;**

2- **atenționare dacă pe factură sunt linii cu Cantitate < 3;**

3- **posibilitatea de a afișa o listă cu valorile facturate pe gestiuni;**

4- **posibilitatea de a afișa o listă cu codurile externe și cele interne pentru fiecare articol din factură.**

5- **import rezervare pe factura de ieșire.**

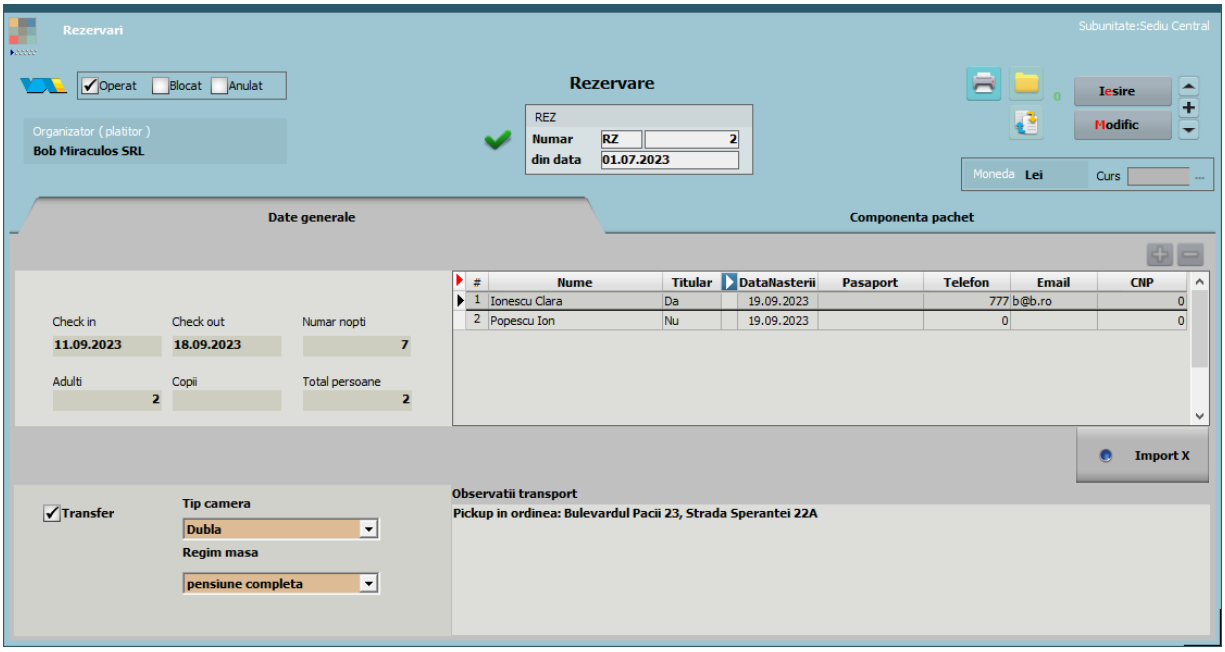
Pentru ultima problemă sunt necesare niște explicații suplimentare.

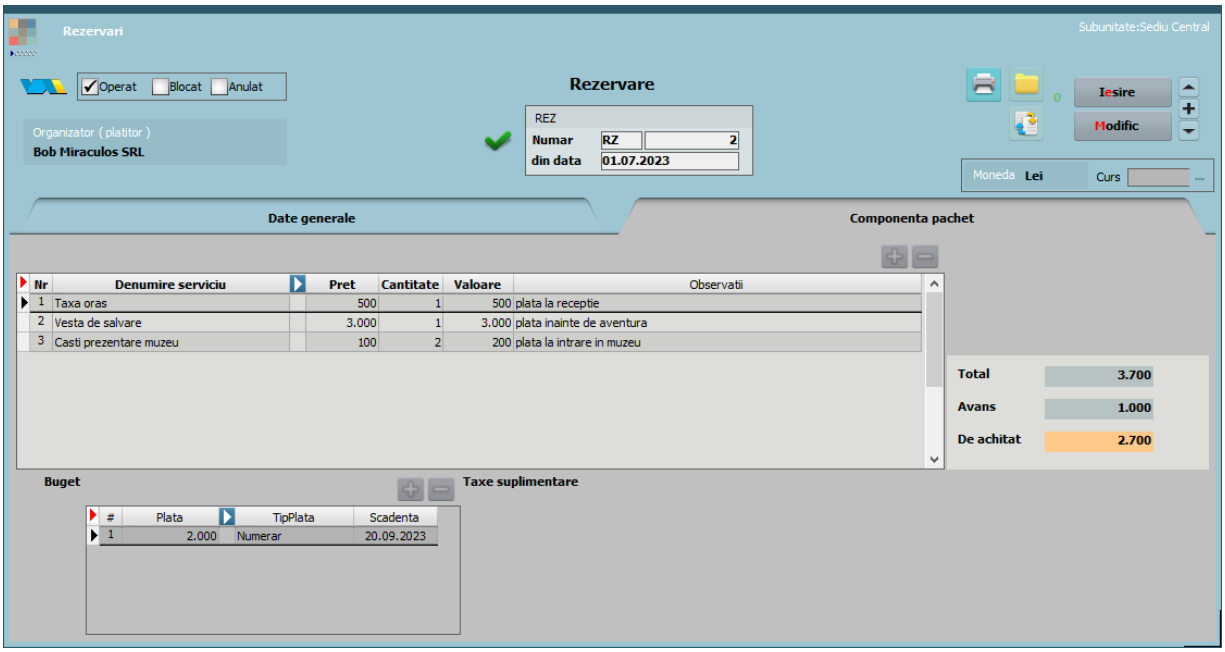
Presupunem că firma lucrează cu rezervări în domeniul turismului.

Rezervările conțin informații legate de servicii care urmează a fi prestate și despre pachetele de turism oferite.

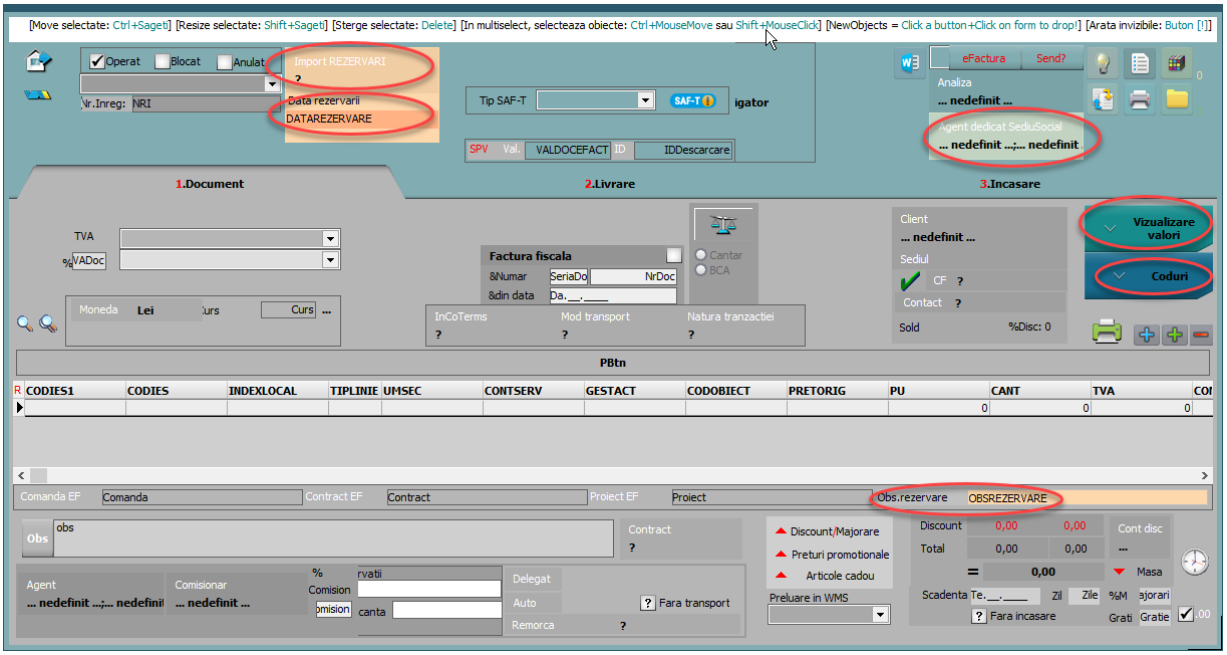
Tabelele în care se vor salva datele sunt deschise la opțiunea "Machete proprii" și nu le voi explica aici.

Folosind opțiunea de design, macheta pentru introducerea datelor la rezervări este deja configurată și arată ca în imaginile de mai jos:



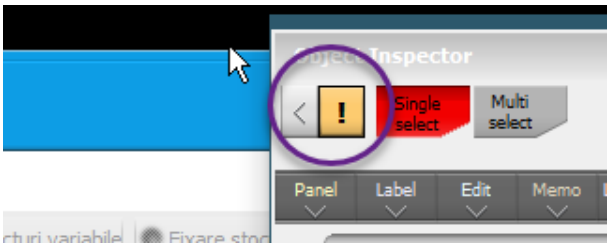


Pornim sa aplicăm aceste cerințe formulate mai sus pe factura de ieșire.
Iată cum arată macheta de factură, cu toate elementele care rezolvă cerințele afișate la design:

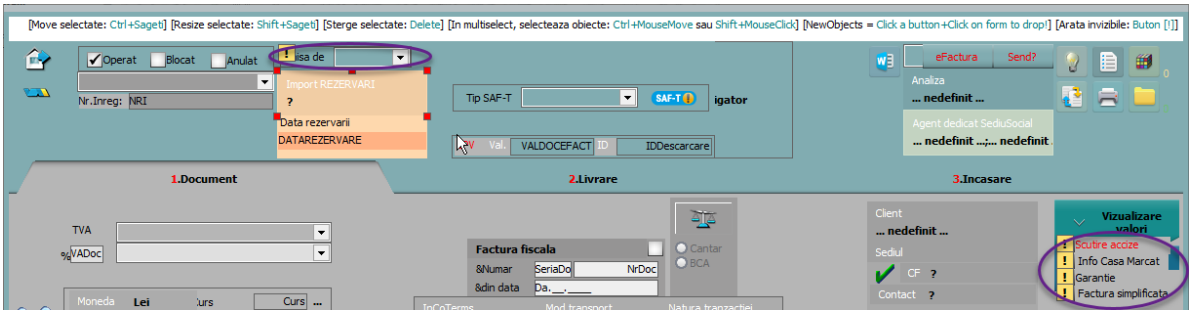


Elementele încercuite sunt controalele noi adăugate pe macheta originală.
Pentru a le face loc, am ascuns alte controale care presupunem că sunt neutilizate de firmă.

Pentru a vizualiza ce controale am ascuns, apăs butonul "Show/Hide invisible", cel încercuit pe *Object inspector* în imaginea de mai jos:

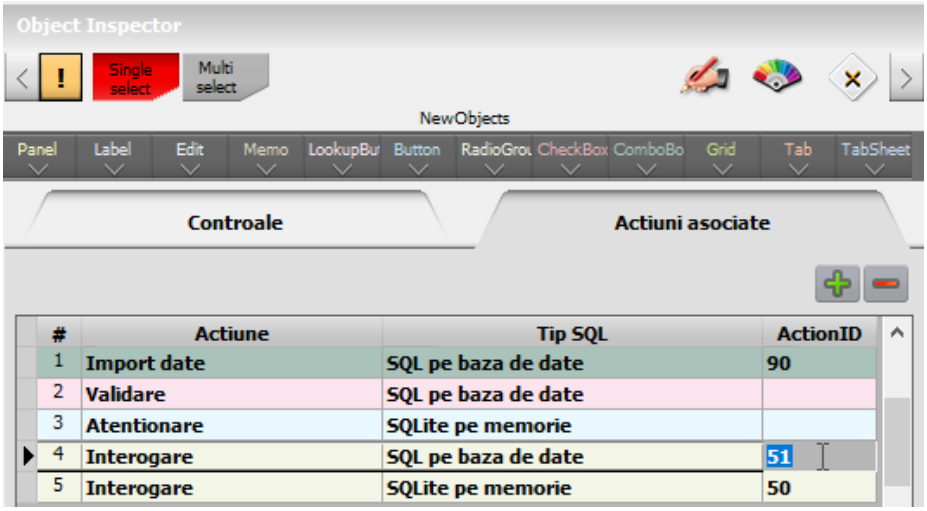


Efectul este următorul:

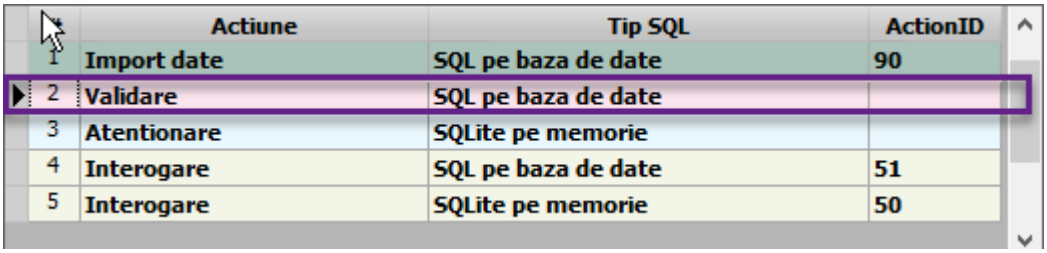


Observați controalele marcate cu pătrat galben la care s-a setat proprietatea *Visible = False.*

Rezolvarea punctelor 1..5 se face din design (F2 + buton "Design" pe o înregistrare nouă de factură), pagina "Actiuni asociate".



1. Validare la salvarea facturilor, astfel incat fiecare articol sa aiba clasa de articol asociata.



	Actiune	Tip SQL	ActionID
1	Import date	SQL pe baza de date	90
2	Validare	SQL pe baza de date	
3	Atentionare	SQLite pe memorie	
4	Interogare	SQL pe baza de date	51
5	Interogare	SQLite pe memorie	50

Mesaj validare:
<DenObiect> nu are clasa declarată!

<DenObiect> reprezintă câmpul pentru denumirea articolului care nu trece validarea, așa cum e el pe memorie în înregistarea din tabela IESIRI1.

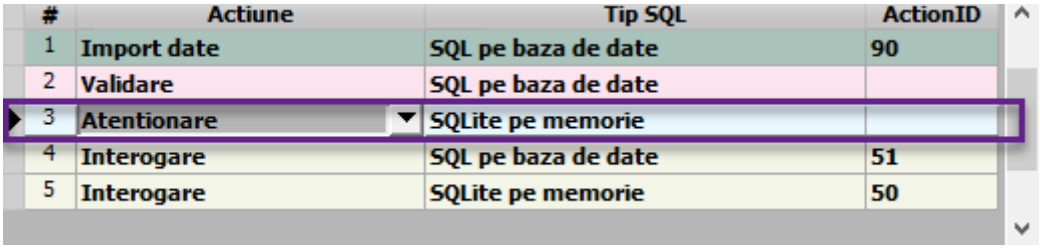
SQL pe baza de date:

```
Select O.Denumire DenObiect
From NartObj o
inner   join NArt a on a.codarticol = o.codarticol
Where Coalesce(a.Clasa, 0) = 0 and O.CodObiect = <Iesiri1.CodObiect>
```

Acest SQL este lansat în Ora/PG, după ce <Iesiri1.CodObiect> este inlocuit cu CodObiect din linia tabelei de pe memorie Iesiri1.
Dacă în SQL pe baza de date este folosită tabela de detaliu principală de pe memorie, în cazul de față Iesiri1, atunci acest SQL se va executa pentru fiecare linie de pe factură, deci pentru fiecare linie din Iesiri1.
Similar dacă în SQL este folosită tabela de EXTENSIE la detaliul principal de pe memorie, în cazul de față Iesiri1EXT.

---

2. Atenționare daca pe factura sunt linii cu Cantitate < 3



#	Actiune	Tip SQL	ActionID
1	Import date	SQL pe baza de date	90
2	Validare	SQL pe baza de date	
3	Atentionare	SQLite pe memorie	
4	Interogare	SQL pe baza de date	51
5	Interogare	SQLite pe memorie	50

Mesaj atentionare:
<DenObiect>: Cantitate facturata < 3 (<Cant>)

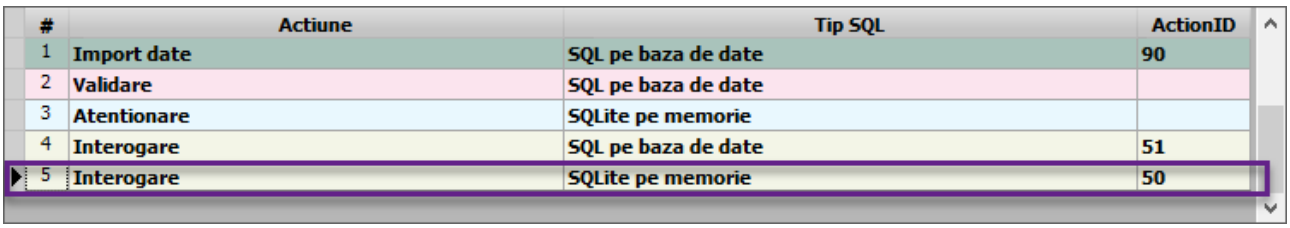
<DenObiect> reprezintă câmpul pentru denumirea articolului care nu trece validarea, așa cum e el pe memorie în inregistarea din tabela IESIRI1.
<Cant> reprezintă câmpul pentru cantitate la articolul care nu trece validarea, așa cum e el pe memorie în inregistarea din tabela IESIRI1.

SQLite pe memorie :

```
Select I1.DenObiect, I1.Cant
From Iesiri1 I1
Where I1.Cant < 3
```

Acest SQL e lansat in SQLite pe tabelele pe memorie care nu sunt încă în baza de date.

3. Afișare listă cu valorile facturate pe gestiuni



#	Actiune	Tip SQL	ActionID
1	Import date	SQL pe baza de date	90
2	Validare	SQL pe baza de date	
3	Atentionare	SQLite pe memorie	
4	Interogare	SQL pe baza de date	51
5	Interogare	SQLite pe memorie	50

Titlu macheta de afișare:
Total valori pe gestiuni

Textul e afișat ca titlu pe macheta care afișează grila cu valorile

SQLite pe memorie :

```
Select Rez.Gestiune, Sum(Rez.V) Valoare
from
(Select Coalesce(Ix1.DenGest, Lt.Gestiune) Gestiune, I1.Cant*I1.PU V
from Iesiri1 I1
left join Iesiri1Ext Ix1 on Ix1.CodIes1 = I1.CodIes1
left join
( select
  Coalesce(x.CodIesInt1, 0) CodIes1,
  Case when x.Gest = 0
    then Null
    else Substr(x.DenGestIoArt, 1, InStr(x.DenGestIoArt, Chr(13))-1)
  end Gestiune
  from LivrTot x
) Lt on Lt.CodIes1 = I1.CodIes1
) Rez
group by Rez.Gestiune
```

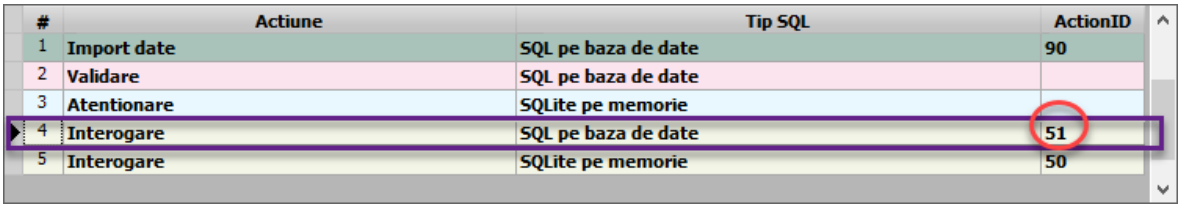
Acest SQL e lansat în SQLite pe tabelele pe memorie care nu sunt încă în baza de date.
De observat cum am extras gestiunea, deoacere aveam următoarele probleme.

Pe Iesiri1 nu exista câmp cu denumirea de gestiune asociată serviciilor!
Mi-am "procurat" singur acest câmp pe extensia la Iesiri1!
Cu alte cuvinte, am adăugat un câmp temporar pe extensia Iesiri1EXT : DenGest.
Iesiri1EXT.DenGest se actualizează la schimbarea valorii Iesiri1.GestAct (vezi structura Iesiri1EXT!).

A doua problemă este faptul ca pe LivrTot (pagina de livrare din factura) există un câmp ce conține Denumirea gestiunii + Denumirea tipului contabil.
Am observat că cele 2 denumiri sunt separate prin caracterul CR, adica #13.
Așadar, pentru liniile de stoc, care au gestiunea pe câmpul LivrTot.DenGestIOArt, am copiat textul pana la #13 (adica Chr(13) in SQL).

---

4. Afișare lista cu codurile externe și cele interne pentru fiecare articol din factura.



#	Actiune	Tip SQL	ActionID
1	Import date	SQL pe baza de date	90
2	Validare	SQL pe baza de date	
3	Atentionare	SQLite pe memorie	
4	Interogare	SQL pe baza de date	51
5	Interogare	SQLite pe memorie	50

Titlu machetă de afișare:
Coduri articole

Textul e afișat ca titlu pe macheta care afișează grila cu valorile

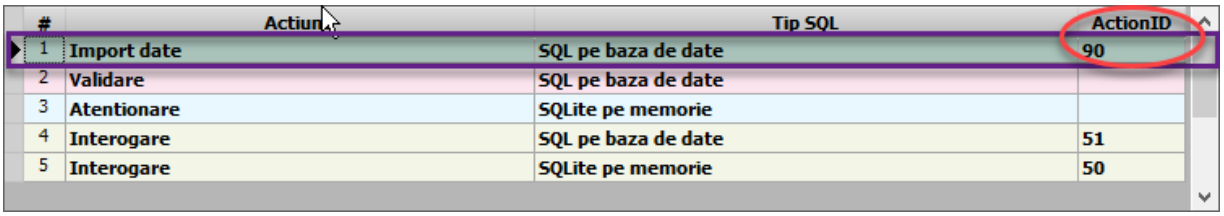
SQL pe baza de date :

Select
distinct O.Denumire, O.CodExtern, O.CodIntern
From NartObj o
Where O.CodObiect = <Iesiri1.CodObiect>

Atenție! SQL pe baza de date are întotdeauna o legatură cu un câmp de pe memorie, acesta făcând legătura cu datele introduse pe machetă care încă nu se găsesc în Ora/PG.
Acest SQL este lansat în Ora/PG, după ce <Iesiri1.CodObiect> este înlocuit cu CodObiect din linia tabelei de pe memorie Iesiri1.
Dacă în SQL pe baza de date este folosită tabela de detaliu principală de pe memorie, în cazul de față Iesiri1, atunci acest SQL se va executa pentru fiecare linie de pe factură, deci pentru fiecare linie din Iesiri1.
Similar, dacă în SQL este folosită tabela de EXTENSIE la detaliul principal de pe memorie, în cazul de fata Iesiri1EXT.

---

5. Import rezervare pe factura de ieșire.



#	Actiun	Tip SQL	ActionID
1	Import date	SQL pe baza de date	90
2	Validare	SQL pe baza de date	
3	Atentionare	SQLite pe memorie	
4	Interogare	SQL pe baza de date	51
5	Interogare	SQLite pe memorie	50

Titlu macheta de import:

neutilizat

SQL pe baza de date :

```
Select R1.CODOBIECT, R1.CANTITATE as Cant, R1.PRET as PU,
      R1.CodRezervare1, R1.Observatii as ObsRezervare,
      R.DataDoc as DataRezervare, R.CodPart as Partener
from rezervare R
inner join Rezervare1 R1 on R1.CodRezervare = R.CodRezervare
where R.CodRezervare = <IesiriExt.CodRezervare>
```

Atenție! SQL pe baza de date are întotdeauna o legătura cu un câmp de pe memorie, acesta făcând legătura cu datele introduse pe macheta care încă nu se gasesc în Ora/PG.

Acest SQL este lansat in Ora/PG, dupa ce <IesiriExt.CodRezervare> este inlocuit cu CodRezervare din linia tabelei extensie la Iesiri de pe memorie, IesiriEXT.
Desigur, pentru a avea acest câmp în IesiriEXT, l-am adaugat în structura extensiei la Iesiri.
El poate fi câmp original (se va și salva pe tabela din Ora/PG IesiriEXT!), sau doar temporar, caz în care este memorat doar pe TMP_IesiriEXT(caz în care nu se salvează pentru această înregistrare în IesiriEXT);
Acest câmp va fi completat folosind un control de alegere, denumit buton Lookup, asa cum o să vedeți mai jos, iar imediat după alegerea rezervării se va executa SQL de import, acesta aducând liniile rezervarii pe Iesiri1.

Observație:

*Cursorul obținut la execuția SQL are următoarele câmpuri: **CODOBIECT, CANT, PU, CodRezervare1, ObsRezervare, DataRezervare, Partener**.*
Algoritmul e suficient de sofisticat și folosește câmpurile astfel:
****CODOBIECT, CANT, PU** *le găsește în structura IESIRI1, drept care adaugă o linie nouă în Iesiri1 (implicit și în Iesiri1EXT care e sincron cu Iesiri1);*
****CodRezervare1, ObsRezervare** *le găsește pe Iesiri1EXT (extensia detaliului principal), drept care le pune acolo;*

CodRezervare1 e camp original, se va salva în Ora/PG pentru a avea evidența liniilor importate;

ObsRezervare e câmp temporar, se incarca doar la interfață pentru a da informații legate de linia importată din rezervări;

**__DataRezervare__ îl găsește în extensia IesiriEXT, fiind un câmp temporar adăugat de mine;*

Se încarcă doar la interfața pentru a da informații legate de rezervare;

**Algoritmul identifică un buton Lookup pe macheta de ieșiri care e legat la câmpul Partener (butonul "Client");*

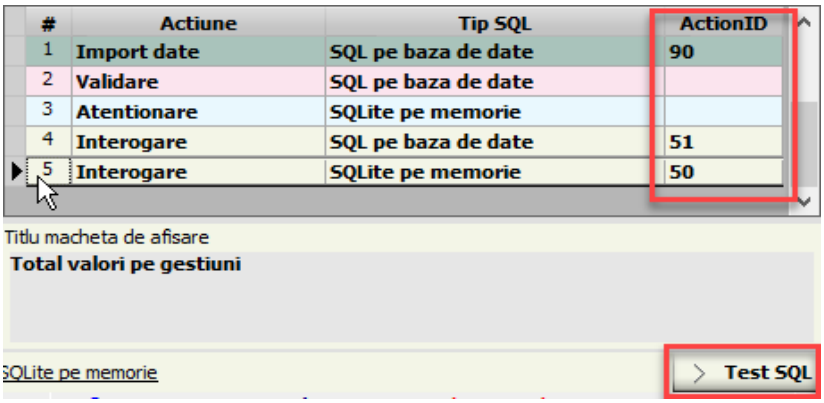
***Partener** e câmpul standard în Ieșiri si odată identificat, algoritmul îl propune pentru alegere pe view mic de parteneri, acționând automat butonul "Client".*

ActionID

De remarcat în grilă coloana ActionID introdusă la import (91) și la interogări (50, 51).

ActionID se vor lega de butoanele care vor executa importul sau interogarea.

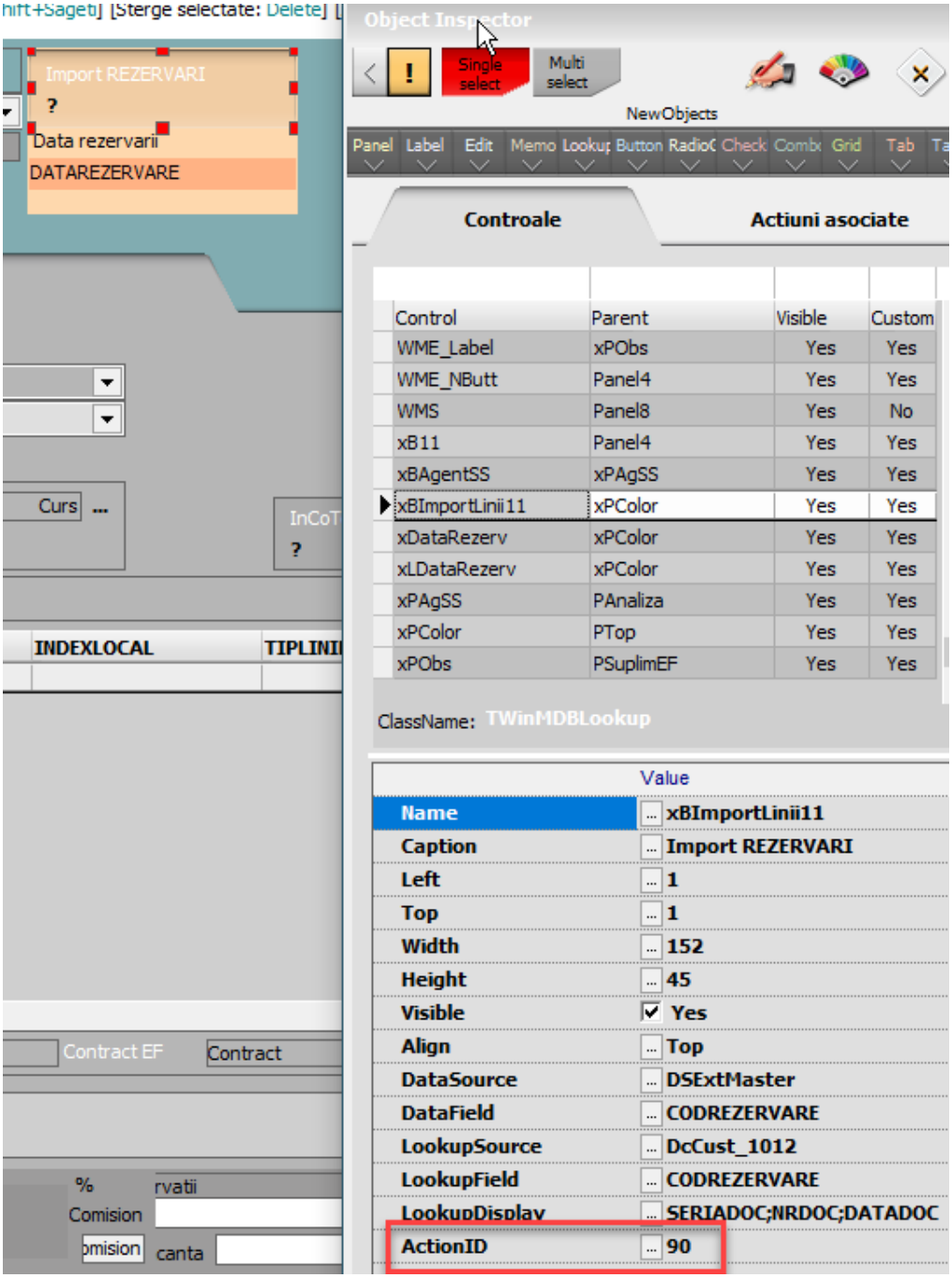
Dacă ActionID este negativ, atunci acțiunea este inactivată. (Rev.1.1)



De asemenea, de remarcat pe pagina de acțiuni asociate un butonul "Test SQL", care desigur va permite să testati fraza SQL pe care ați produs-o!

Acțiunea de import necesită alegerea în prealabil a unui cod, în cazul nostru al unui cod de rezervare.

Fiind vorba de o alegere, vom folosi pentru aceasta controlul potrivit, și anume un *LookupButton*.



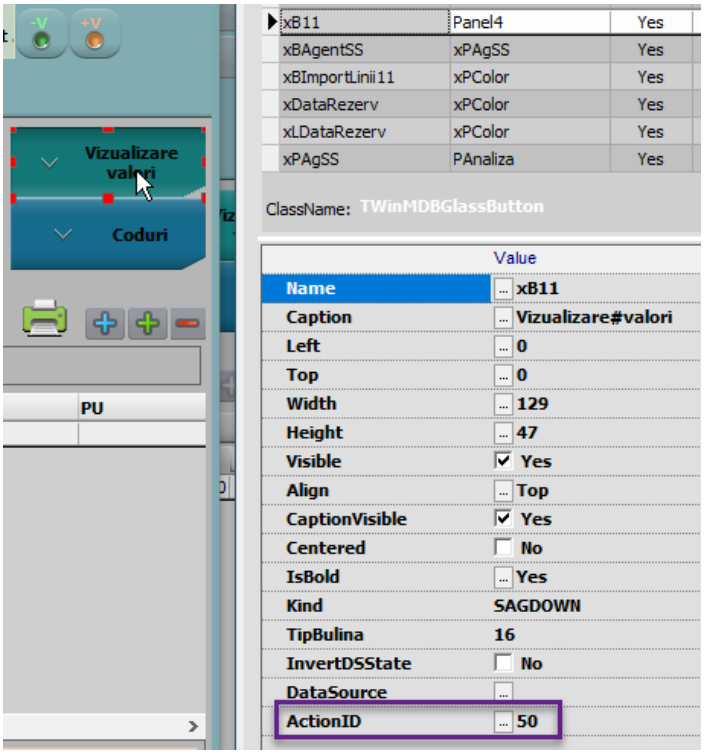
Observați proprietatea ActionID = 90.
Aceasta proprietate va declanșa SQL-ul atasat liniei de import cu ActionID 90 și va aduce datele pe factură.
Butonul e legat la Datasource = DSExtMaster (adica in final la IesiriEXT).
Codul ales se stochează în DataField = CodRezervare , adică IesiriEXT.CodRezervare.

Pentru a afișa pe buton informații legate de rezervare (altfel CodRezervare nu ar avea înțeles pentru operator), se folosește LookupSource = DcCust_1012.
Acest mod de codificare a unui view mic asociat, DcCust_1012, identifică prin 1012 cod de identificare a machetei proprii "rezervări" , care are ID egal cu 1012.

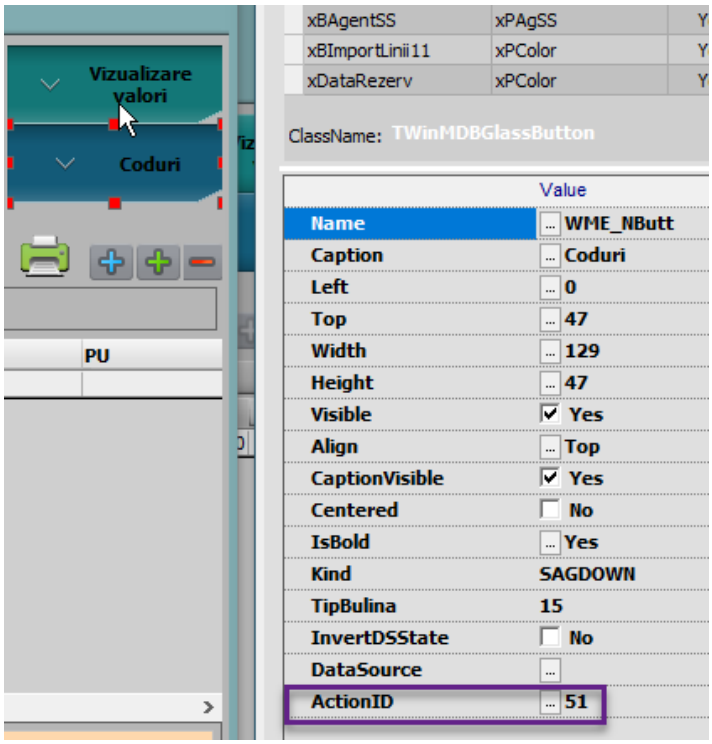
Pentru a aduce informatiile cerute in LookupDisplay, și anume SERIADOC;NrDOC;DataDoc (care sunt din tabela Rezervari!), butonul leagă câmpul IesiriEXT.CodRezervare
cu înregistrarea din DcCust_1012 (care e tabela Rezervari) prin LookupField = CodRezervare (adica Rezervari.CodRezervare).
Rezervari.CodRezervare = IesiriEXT.CodRezervare, unde Rezervari.CodRezervare este **LookupField**, iar IesiriEXT.CodRezervare este **DataField**.

De pe această linie se afișează pe butonul lookup informațiile următoare:

ActionID 50 din grila de "Acțiuni asociate" e legat de un buton simplu, care are și el proprietate ActionID potrivită.

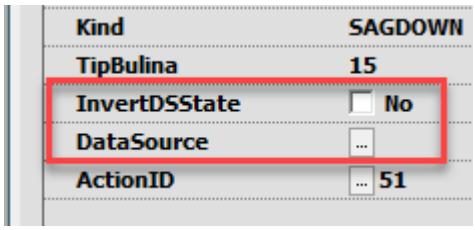


Similar, ActionID 51 din grila de "Acțiuni asociate" e legat de un buton simplu, care are și el proprietate ActionID potrivită.



DataSource și InvertDSState

O ultimă observație legată de butoanele simple.



Există la butoanele simple o proprietate denumită **DataSource.**

*Dacă aici NU e specificată o valoare, atunci butonul e activ indiferent dacă macheta e în editare sau înregistrarea a fost salvată.
*Dacă aici e specificată o valoare, atunci butonul e activ astfel:
- când InvertDSState = No, atunci butonul e activ doar când ești în editate pe machetă;
- când InvertDSState = Yes, atunci butonul e activ doar după salvarea înregistrării;

Alte exemple din practică

Validare la salvare factură ieșire că denumirea carnetului documentului și denumirea carnetului comenzii stinse să aibă același sufix

(Sufix = textul de la finalul denumirii, dupa caracterul ”-”)

Mesaj validare:

<DenObiect> provine de la alta serie de document

SQL pe baza de date:

```
select nob.denumire as DenObiect from nartobj nob
where nob.codobiect = <Iesiri1.codobiect>
and (SELECT upper(SUBSTR(dc.DENUMIRE, INSTR(dc.DENUMIRE, '-', -1) + 1))
      FROM doccarnete dc
      WHERE dc.codcarnet = <Iesiri.Carnetdoc>) !=
        (select upper(SUBSTR(dci.DENUMIRE, INSTR(dci.DENUMIRE, '-', -1) + 1))
        FROM doccarnete dci
         WHERE codcarnet in
                (select carnetdoc
                   FROM comanda
                  WHERE codcomanda in
                        (select codcomanda
                           FROM comanda1
                          WHERE codcomanda1 = [ select codliniecomanda
                                  FROM IesLink
                                 WHERE codies1 = <Iesiri1.Codies1> ])))
```

Validare la salvare bon consum că denumirea carnetului documentului și denumirea carnetului comenzii stinse să aibă același sufix

Mesaj validare:

<DenObiect> provine de la alta serie de document

SQL pe baza de date:

```
select nob.denumire as DenObiect from nartobj nob
where nob.codobiect = <Iesiri1.codobiect>
and (SELECT upper(SUBSTR(dc.DENUMIRE, INSTR(dc.DENUMIRE, '-', -1) + 1))
      FROM doccarnete dc
      WHERE dc.codcarnet = <Iesiri.Carnetdoc>) !=
        (select upper(SUBSTR(dci.DENUMIRE, INSTR(dci.DENUMIRE, '-', -1) + 1))
        FROM doccarnete dci
         WHERE codcarnet in
                (select carnetdoc
                   FROM comanda
                  WHERE codcomanda in
                        (select codcomanda
```

```
                FROM comanda1
                WHERE codcomanda1 = [ select codliniecomanda
                                      FROM IESPRODLINK
                                      WHERE codies1 = <Iesiri1.Codies1> ]))))
```

Validare la salvare notă de predare că denumirea carnetului documentului și denumirea carnetului comenzii stinse să aibă același sufix

Mesaj validare:

<DenObiect> provine de la alta serie de document

SQL pe baza de date:

```
select nob.denumire as DenObiect from nartobj nob
where nob.codobiect = <Intrari1.codobiect>
and (SELECT upper(SUBSTR(dc.DENUMIRE, INSTR(dc.DENUMIRE, '-', -1) + 1))
      FROM doccarnete dc
      WHERE dc.codcarnet = <Intrari.Carnetdoc>) !=
        (select upper(SUBSTR(dci.DENUMIRE, INSTR(dci.DENUMIRE, '-', -1) + 1))
        FROM doccarnete dci
         WHERE codcarnet in
                (select carnetdoc
                    FROM comanda
                   WHERE codcomanda in
                          (select codcomanda
                              FROM comanda1
                             WHERE codcomanda1 = [ select codcomanda1
                                                   FROM IntrPRODLINK
                                                   WHERE codintr1 = <Intrari1.Codintr1> ])))
```

Validare la salvare comandă internă că sufixul clasei web a articolelor este aceeași cu sufixul denumirii carnetului comenzii interne

Mesaj validare:

<DenObiect> este din alta clasa decat documentul

SQL pe baza de date

```
select nob.denumire as DenObiect from nartobj nob
where nob.codobiect = <Comanda1.Codobiect>
and (SELECT upper(SUBSTR(dc.DENUMIRE, INSTR(dc.DENUMIRE, '-', -1) + 1))
      FROM doccarnete dc
      where dc.codcarnet = <Comanda.Carnetdoc>) !=
        (select upper(SUBSTR(cls.den_claseas, INSTR(cls.den_claseas, '-', -1) + 1))
         from nartobj nob
             join nart n on n.codarticol = nob.codarticol
             join nclaseas cls on cls.codclaseas = n.clasaw
        where nob.codobiect = <Comanda1.Codobiect>
)
```

Review-uri document

Rev. 1.0	07.01.2025	Documentație inițială
Rev. 1.1	12.03.2025	Vers. 25.021 - Inserare SQLite in SQL pe baza de date