Pokročilé programování ve 4D

Obsah

4th Dimension

Příručka školení

Pokročilé programování ve 4D

Verze 6.0.3

Kurz vytvořen:

Jméno	E-Mail
Kent Wilbur	kent@acius.com

Kurz upraven:

Jméno	E-Mail
Jaroslav Macháček	inforce@mbox.vol.cz

ACI Univerzita

1. Úvod

1.1. Vítejte

Tento kurs je zaměřen na ty, kteří již mají nějaké zkušenosti s programováním ve 4th Dimension (4D) a kteří chtějí přidat své databázi nové funkce a rysy. Seznámíte se zde také s novými funkcemi 4D.

1.2. Jak budete v kursu pracovat

Délka trvání kursu je tři dny. Způsob práce je následující:

- Instruktor diskutuje některé rysy a demonstruje je s pomocí promítání na plátno, vy provádíte cvičení a příklady z této příručky, které procvičují diskutované rysy.
- Vy provádíte cvičení a příklady z této příručky, které procvičují diskutované rysy.
- Instruktor při cvičeních odpovídá na dotazy a pomáhá těm, kteří to potřebují.

1.3. Průběh školení

Při práci v tomto kursu budete upravovat a zjednodušovat databázi pro distributora videokazet, je to ta samá databáze, kterou jsme používali v předchozích školeních. Ukládá údaje o zákaznících, fakturách, položkách faktur a produktech.

Databáze je pojmenovaná po fiktivní společnosti „ACI Video“. Na konci tohoto kursu bude struktura zahrnovat níže uvedené tabulky a pole.

[zDialogy]

Název pole	Typ	Vlastnosti
Povinné	Logické	

[Zákazníci]

Název pole	Typ	Vlastnosti
IDZákazníka	Alfa 10	Nutný vstup; Pouze zobrazit; Indexované; Jedinečné
Jméno	Alfa 20	Indexované
Příjmení	Alfa 20	Indexované
Firma	Alfa 25	Indexované; Nutný vstup
Adresa	Alfa 25	
Město	Alfa 20	Indexované
Stát	Alfa 2	Indexované
PSČ	Alfa 10	Indexované
Telefon	Alfa 10	
PlátceDaně	Logické	
Důležitý	Logické	
CelkovéProdeje	Real	
DatumVytvoření	Datum	Pouze zobrazit; Neviditelné
ČasVytvoření	Čas	Pouze zobrazit; Neviditelné
DatumÚpravy	Datum	Pouze zobrazit; Neviditelné
ČasÚpravy	Čas	Pouze zobrazit; Neviditelné

[ObecnéInformace]

Název pole	Typ	Vlastnosti
NázevFirmy	Alfa 40	Nutný vstup, Indexované, Jedinečné
Adresa	Text	
Město	Alfa 22	
Stát	Alfa 2	
PSČ	Alfa 9	
Telefon	Alfa 10	

[Faktury]

Název pole	Typ	Vlastnosti
ČísloFaktury	Long Integer	Nutný vstup, Neměnné; Indexované; Jedinečné
IDZákazníka	Alfa 10	Indexované; Jedinečné
DatumFaktury	Datum	Indexované
FakturaCelkem	Real	
Placeno	Logické	
ZpůsobPlatby	Alfa 15	
DatumVytvoření	Datum	Pouze zobrazit; Neviditelné
ČasVytvoření	Čas	Pouze zobrazit; Neviditelné

DatumUpravení	Datum	Pouze zobrazit; Neviditelné
ČasUpravení	Čas	Pouze zobrazit; Neviditelné

[PoložkyFaktur]

Název pole	Typ	Vlastnosti
ČísloFaktury	Long Integer	Indexované
IDZbožíProdukty	Long Integer	Indexované
Neužito	Logické	Neviditelné
Množství	Integer	
JednCena	Real	
CenaCelkem	Real	

[Produkty]

Název pole	Typ	Vlastnosti
IDZboží	Alfa 10	Nutný vstup; Neměnné; Indexované; Jedinečné
Název	Alfa 30	Indexované
Cena	Real	Indexované
Rok	Integer	Indexované
Kategorie	Alfa 15	Indexované
DatumVytvoření	Datum	Pouze zobrazit; Neviditelné
ČasVytvoření	Čas	Pouze zobrazit; Neviditelné
DatumUpravení	Datum	Pouze zobrazit; Neviditelné
ČasUpravení	Čas	Pouze zobrazit; Neviditelné

[zNavracenáČísla]

Název pole	Typ	Vlastnosti
NázevSekvence	Alfa 20	Indexované
Hodnota	Longint	

[zSekvence]

Název pole	Typ	Vlastnosti
NázevSekvence	Alfa 20	Indexované
Hodnota	Longint	

[zStruktura]

Název pole	Typ	Vlastnosti
ČísloTabulky	Integer	Nutný vstup, Pouze zobrazit, Indexované
ČísloPole	Integer	Nutný vstup, Pouze zobrazit, Indexované
PlatnýNázev	Alfa 31	
NázevZástupce	Alfa 31	
PořadíZástupce	Integer	

VložitVlastníHledání	Logické
VložitVlastníTřídění	Logické

[zUživatelé]

Název pole	Typ	Vlastnosti
IDUživatele	Longint	Nutný vstup, Indexované, Jedinečné
JménoUživatele	Alfa 31	Nutný vstup
PlatnéJméno	Alfa 42	
Adresa	Text	
Město	Alfa 22	
Stát	Alfa 2	
PSČ	Alfa 9	
Telefon	Alfa 10	
DatumPosledníhoPřipojení	Datum	
ČasPosledníhoPřipojení	Čas	
Trvání	Čas	
NejdelšíDobaPřipojení	Čas	
ČítačTikůPřipojení	Longint	Neviditelné
Připojení	Integer	
Připojen	Logické	
OdmítnoutPřístup	Logické	
PředvolbyDotazuTabulky	Text	Neviditelné
PředvolbyTříděníTabulky	Text	Neviditelné
VíceZobrazení	Text	Neviditelné

1.4. Konvence

- "Programování" a "Kód" jsou synonyma pro instrukce, která budete vkládat.
- To co budete zapisovat je v tomto tvaru.

  Příklad: Napište Johnson do pole Příjmení.
- Speciální klávesy na klávesnici jsou uvedeny následovně:

  Stiskněte Enter.
- Jestliže je požadováno, abyste stiskli více než jednu klávesu současně je tento požadavek zapsán následovně:

  Stiskněte CTRL + Shift + 3.
  Vysvětlení: Stiskněte současně klávesy CTRL, Shift a 3.

Macintosh™ ekvivalent bude okamžitě následovat po Windows™ v kurzívě, jak je ukázáno níže.
Stiskněte: Ctrl + Shift + 3 *(Command + Shift + 3)*.

- Volba položky z nabídky je popsána následovně:

  Zvolte Soubor ¬ Otevřít (Ctrl + O) *(□ + O)*
  Vysvětlení: Z nabídky Soubor, zvolte položku Otevřít. Na Windows™ stiskněte Control + O.
  Na Macintosh™ stiskněte Command + O.

- Položky 4D kódu jsou zobrazovány stejně jako jsou v 4D Editoru metod. Například:

[Zákazníci]Stát	Pole
ALERT	Příkaz
MyProcedure	Metoda (procedura)

- Sekce označené "Extra Credit" jsou výběrové. Jestliže skončíte svůj příklad dříve můžete pracovat v sekci "Extra Credit" a naučit se více.

1.5. Příkazy

V tomto kursu se dozvíte o následujících příkazech jazyka 4D:

- APPEND TO LIST
- Application type
- BLOB TO VARIABLE
- CLEAR LIST
- COMPRESS BLOB
- Copy list
- COPY SET
- Count tables
- Current form page
- DELETE DOCUMENT
- DELETE LIST ITEM
- DISABLE MENU ITEM
- DISPLAY RECORD
- DRAG AND DROP PROPERTIES
- Drop position
- ENABLE MENU ITEM
- EXPAND BLOB
- Field
- FILTER KEYSTROKE
- GET FIELD PROPERTIES
- GET HIGHLIGHT
- GET LIST ITEM
- GET LIST PROPERTIES
- GET PICTURE RESOURCE
- GET PROCESS VARIABLE
- GOTO AREA
- GOTO PAGE
- INSERT LIST ITEM
- In transaction
- Is a list
- Keystroke
- Load list
- New list
- OLD
- ON ERR CALL
- PAGE BREAK
- PAGE SETUP
- PRINT FORM
- PRINT SETTINGS
- REDRAW LIST
- REDRAW WINDOW
- RELATE MANY SELECTION
- RELATE ONE SELECTION
- SELECT LIST ITEM
- Selected list item
- SET BLOB SIZE
- SET FIELD TITLES
- SET LIST ITEM PROPERTIES

- SET QUERY DESTINATION
- SET QUERY LIMIT
- SET TABLE TITLES
- Trigger level
- TRIGGER PROPERTIES
- VARIABLE TO BLOB
- VARIABLE TO VARIABLE

1.6. Konvence názvů pro proměnné

Pro přehlednost kódu je důležitá konvence názvů proměnných pro metody a funkce. ACI nezavedla žádnou povinnou konvenci, prezentujeme zde však jednu, která je poměrně široce používána v různých publikacích a kurzech. Všechna písmena jsou malá, kromě L označující LongInteger, kde se může plést 1 a malé l..

Všechny array (pole v paměti) začínají“a”. Dvoudimenzionální array začínají “a2” nebo “aa“.

Proměnné mají následující předpony, které jsou rovněž používány pro array po písmenu “a”.

Typ	Proměnná	1D Array	2D Array
String	s	as	a2s
Text	t	at	a2t
Integer	není	ai	a2i
Longint	L	aL	a2L
Real	r	ar	a2r
Date	d	ad	a2d
Time	h	není	není
Boolean	f	af	a2f
Picture	g	ag	a2g
Pointer	p	ap	a2p
BLOB	o	není	není

1.7. Aktivní objekty

Aktivní objekty jsou téměř vždy číselné. Pro odlišení však pro ně používáme vlastní předpony.

Objekt	Předpona
Tlačítko/Button	b
Zaškrtávací políčko/Checkbox	ck
Teploměr/Thermometer	th
Přepínač/Radio Button	rb1; rb2; rb3; sb1; sb2; sb3, etc.

1.8. Konvence názvů pro metody

Vestavěné příkazy jazyka 4D jsou uváděny následovně :

- Vše velkými písmeny – pro příkazy vykonávající konkrétní akci jako CANCEL (provádí akci storno).
- Smíšeně velká a malá písmena – pro příkazy, které vrací hodnotu jako např. Uppercase.

Příkazy se smíšenou konvencí jsou vždy vpravo od operátoru přiřazení.

V žargonu programátorů je tato metoda navracející hodnotu nazývána funkce

Většina programátorů volí následující konvenci. Metody jsou uváděny s velkými písmeny na počátku každého slova „MojeProcedura“. Speciální metody nejčastěji používané začínají „aa“, aby v seznamu metod byly na počátku. Další dělení je pro metody zvláštního účelu:

- M_” pro metodu spouštěnou z nabídek
- “P_” pro metodu začínající nový proces
- “E_” pro metodu, která je navržena k ručnímu spouštění pomocí příkazu nebo nabídky Execute.

Kromě toho jsme vytvořili následující konvenci pro segmentaci kódu. Vše přímo přiřazené některým účelům (moduly, proměnné, názvy metod, formulářů) začíná následujícími předponami:

- “ALIAS_” pro komponenty, které zacházejí s tabulkami, nebo zastupují pole.
- “CHO_” pro metody, které jsou pro výběrové seznamy
- “GEN_” pro metody, které jsou plně generické
- “IMPEXP” pro komponenty účastné při exportu a importu
- “INITIALIZE” pro komponenty, které inicializují promměnné nebo procesy
- “KEY_” pro komponenty, které zachytávají klávesy
- “LOCK_” pro komponenty, které obsahují zamykání a odemykání záznamů
- “PROCESS_” pro komponenty účastné při řízení procesů
- “SPE_” pro komponenty zacházející s uloženými procedurami
- “WIN_” pro komponenty provádějícími manipulaci s okny

Příklady:
- M_Customers
- P_Customers

- sConstants
- GEN_MyMethod

Když píšete vaše vlastní metody, které jsou funkcemi, je často praktické začínat název písmenem, které reprezentuje typ navracené hodnoty. Tato písmena jsou pak stejná jako v tabulce výše.

1.9. Zvláštní poznámka: Užití konstant definovaných návrhářem

Tato databáze používá specielní konstanty definované uživatelem, tak aby byl kód pochopitelnější a čitelnější. Do své databáze tento kód nemůžete převzít přímo, bez přenesení i dodatečných konstant. Tyto konstanty jsou umístěny v zdroji 4DK# a mohou být přeneseny programem ResEdit na Macintosh, nebo specielním programem od ACI - Rebrocerer. Další možností je úprava kódu bez použití těchto konstant.

2. Vítejte ve 4D V6

4D V6 je zcela nový svět, vše je zcela nové, i když povědomé. Jesliže nebudete schopni s pomocí 4D V6 vytvořit ve své oblasti nejlepší databázi na trhu, měli by jste se poohlédnout po jiném způsobu obživy. Je 4D V6 Oracle ? Ne. Je 4D V6 Sybase ? Ne. Na trhu není nic co by se mohlo vyrovnat možnostem kompletního prostředí 4D. 4D se ve svém současném stavu i s budoucími plány snaží zaujmout na databázovém trhu vedoucí pozici.

Nejdříve, ve verzi 6 je symbolika, která byla změněna od verze 6. Změny byly provedeny, aby byla sladěno názvosloví a symbolika na takovou, která je dnes vývojáři přijímána jako univerzální .

Verze 3	**Verze 6**
Soubor	Tabulka
Formát	Formulář
Vstupní formát	Vstupní formulář
Výstupní formát	Výstupní formulář
Procedura	Metoda
	Metoda databáze
Globální procedura	Metoda projektu
Procedura formátu	Metoda formuláře
Procedura souboru	
Skript	Metoda objektu
Attributes	Properties (Not all)
Object Text	Object Caption (i.e. pole, tlačítko...)
Prostředí aplikace	Prostředí vlastních nabídek
Proces uživatele/aplikace	Proces nabídek uživatele/aplikace
Hledání (Search)	Dotazování (Query by)
Třídění (Sort)	Třídění (Order by)
» nebo >>	->
Sady (sety) na klientu	Sady (sety) na serveru

Search a Sort jsou pouze dva z mnoha dalších příkazů, které změnily názvy. Při konverzi v verze 3 na verzi 6 se provede automatická konverze, pokud tyto příkazy nepoužíváte v příkaze EXECUTE jako textové řetězce.

Za druhé, ve verzi 6 byly provedeny změny v některých velikostech a délkách objektů.

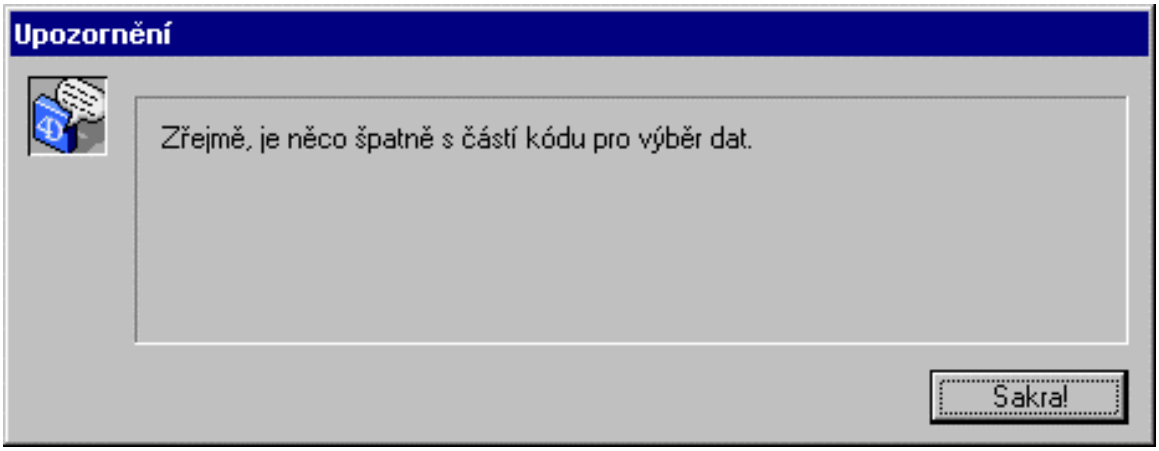
Popis	Verze 3	Verze 6
Názvy tabulek a polí	15	31
Názvy proměnných	11	31
Názvy formulářů	15	31
Názvy externích metod/funkcí	15	31

3. Technika ladění a debugger

Debugger se kompletně změnil od základu a některé techniky byly přidány ještě do verze 6.5. 4D zavádí nový standard do této oblasti. Nevíme o nástroji či jazyce, který by měl účinnější nástroj pro ladění aplikace.

3.0.1. Test ladění a debuggeru.

1. Spusťte databázi debugger.
2. Zvolte Dema ¬ Debugger….



Zřejmě, je něco špatně s částí kódu pro výběr dat.

3.1. Okno přerušení .

Okno se seznamem přerušení je jeden z rysů debuggeru. Dovolí uživateli přiřadit, nebo odstranit místa v kódu, kde bude aktivováno okno ladění. Můžete nastavit přerušení na určitém příkazu, nebo příkaze, který používá určitou sledovanou hodnotu, nebo dokonce na proměnnou, pokud obdrží určitou hodnotu. Toto okno zobrazí všechna přerušení v kódu a to jak ty, které byly nastaveny v kódu v editoru metod, tak i ty, které byly nastaveny přímo v okně ladění.

3.1.1. Nastavení zachytávání k vytvoření přerušení

1. Přejděte do Prostředí návrháře
2. Vyberte Nástroje ¬ Průzkumník provádění….
3. Přejděte na stranu Zachytit a klepněte na tlačítko Přidat zachytávání.

Když přidáte zachytávání je do okna zachytávání vložen příkaz ALERT označený červenou tečkou. Můžete ji buď přepsat, nebo levým tlačítkem myši (CTRL – klepnout na Mac) zobrazit seznam příkazů v hierarchickém seznamu a přepsat jej výběrem ze seznamu.

4. Zobrazte seznam příkazů, jak je popsáno výše.
5. Ze seznamu vyberte příkaz C_DATE.

Když je zachytávání nastaveno, je prováděno, dokud jej neodstraníte, nebo nepozastavíte klepnutím na červenou tečku. Nyní jakýkoliv výskyt příkazu C_DATE v kódu bude zachycen a program bude přerušen a zobrazeno oko ladění

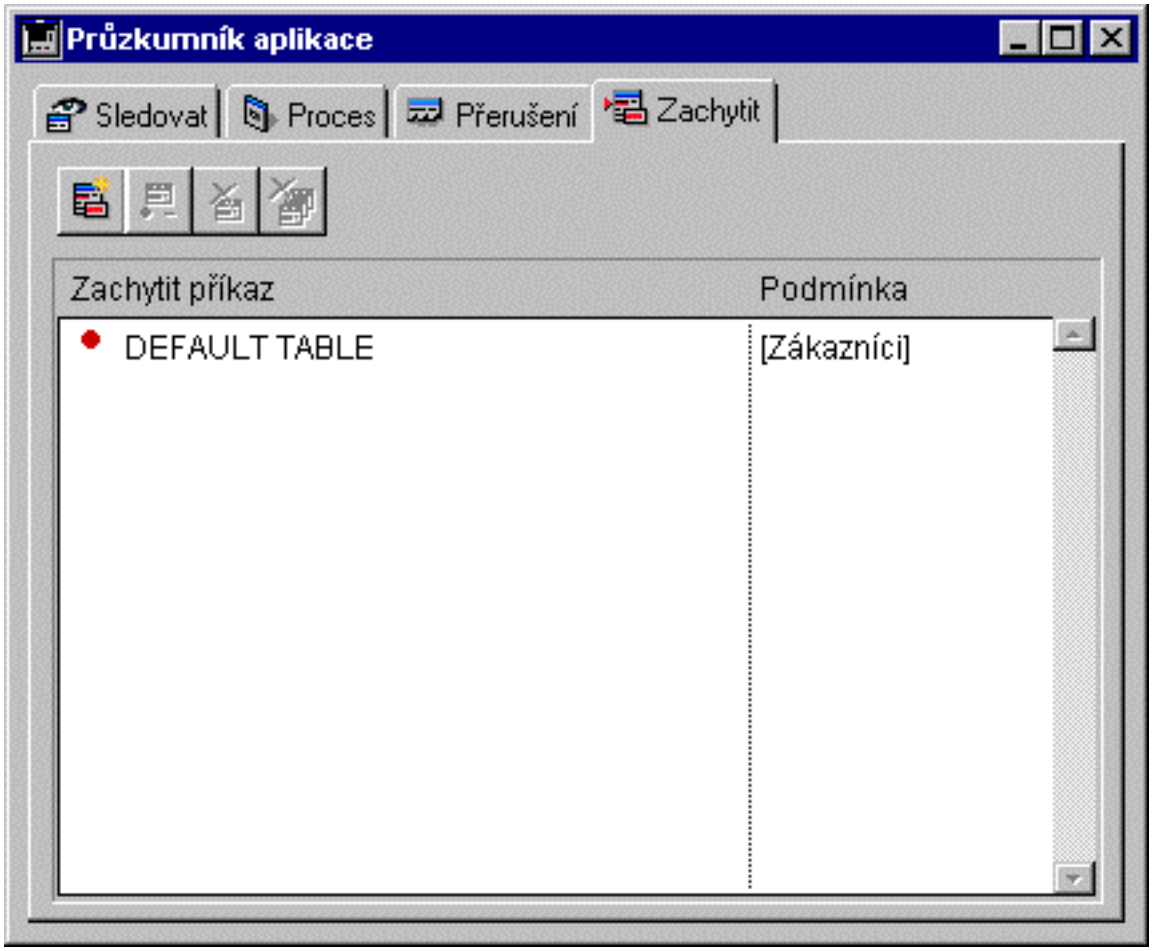
3.1.2. Odstranění Zachytávání

Aby jste odstranili zachytávání musíte klepnout na tlačítko Vymazat z menu a nebo označit zachytávání a stisknout klávesu Delete. Pozastavení zachytávání lze provést klepnutím na tlačítko Umožnit / znemožnit.

1. Vyberte zachytávání C_Date.
2. Stiskněte klávesu Delete.

3.1.3. Definice Zachytávání s podmínkami

1. (Ctrl + click) (Left click) v okně zachytávání a vyberte Přidat nové zachytávání pro.
2. Vyberte ze seznamu příkaz Tabulka ¬ DEFAULT TABLE.
3. Klepněte ve sloupci Podmínky.
4. Napište [Zákazníci].



Kdekoliv se objeví v kódu příkaz DEFAULT TABLE ([Zákazníci]) program se automaticky přepne do okna ladění.

5. Vraťte se do prostředí Vlastní nabídky
6. Zvolte Dema ¬ Debugger….

3.2. Oblast metod

Spodní část okna ladění obsahuje krokovanou metodu. Šipka vpravo označuje řádek kódu, který bude prováděn.

```
  <>f_Version6x20:=True
  <>fJ_Steinman:=True
End if

DEFAULT TABLE([Zákazníci])
C_LONGINT($Lpid)

$Lpid:=PROCESS_LSpawnProcess ("P_Customers";32;"Zákazníci";True;True;True)

 `Konec metody
```

3.3. Oblast výrazů 4D

Oblast výrazů vlevo v okně ladění zobrazuje většinu informací, které potřebujete znát při ladění programu.

```
If ($LUserLimit<2)
  If (Count parameters>4)
    If ($LUserLimit=0)
      $fAllowMultipleViews:=$5
    End if
    If (Count parameters>5)
      $fReclaimPausedProcess:=$6
    End if
  End if

  $Lpid:=Process number($sProcessName)
```

3.3.1. Objekty v řadě / řádce

Tato položka obsahuje hodnoty proměnných a array užívaných v předchozí a právě prováděné řádce kódu krokované metody. Takže můžete vidět hodnoty objektů před a po provedení řádky kódu.

1. Klepněte na znaménko vedle nápisu Objekty v řadě a rozšiřte seznam.
2. Čtyřikrát klepněte na tlačítko Přeskočit a pozorujte změny hodnot v oblasti Objekty v řadě.

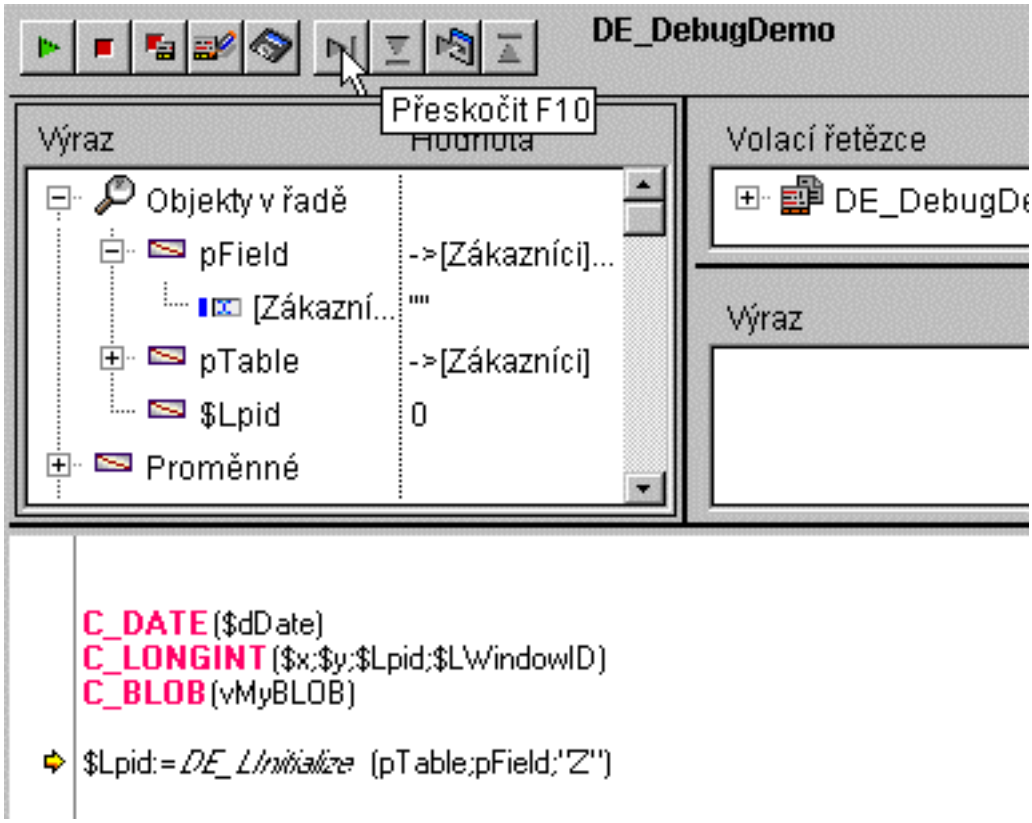
Oblast výrazu budeme také probírat dále spolu s dalšími rysy okna ladění. Nyní, ale probereme základní tlačítka.

3.3.2. Tlačítko Přeskočit

Ve starém debuggeru jste museli projít přes všechny procedury a nebo se pomocí Shift-Click vyhnout krokování ve volané proceduře. Nyní pouze stisknete klávesu Enter a nebo klepnete na tlačítko Přeskočit.

Toto tlačítko vám umožní krokovat metodu krok po kroku. Jestliže je ve krokované metodě volána jiná metoda, toto tlačítko způsobí, že tato metoda bude provedena, ale toto provádění nebude zobrazeno v okně ladění. Stejného výsledku lze dosáhnout klepnutím na tlačítko Enter (Enter na numerické klávesnici pro Windows).

1. Klepněte na tlačítko Přeskočit nebo stiskněte klávesu Enter.



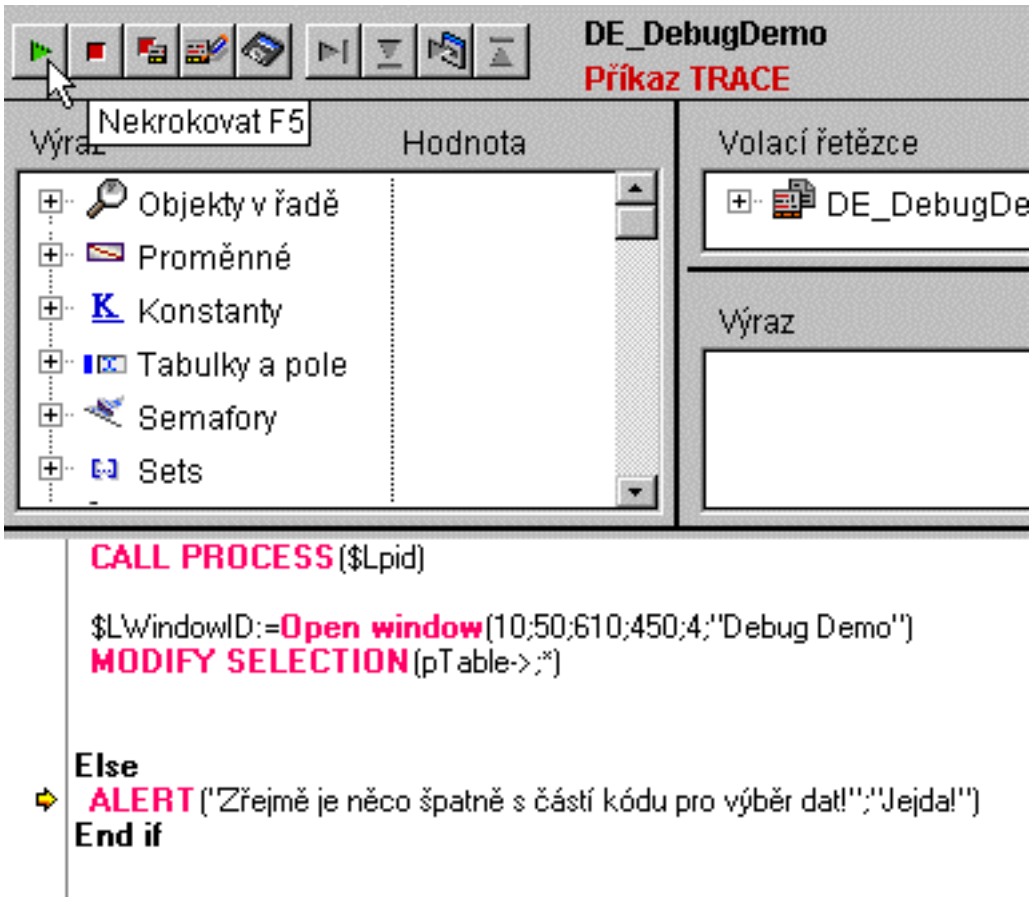
2. Všimněte si, že se objekt v řádce $Lpid = -1
3. Klepněte na tlačítko Přeskočit nebo stiskněte klávesu Enter.
4. Všimněte si, že řádek s příkazem ALERT bude proveden.

Problém je tedy viditelně v metodě, kterou jsme přeskočili.

3.3.3. Tlačítko Nekrokovat

Toto tlačítko vám dovolí provádět metodu bez krokování dokud není program přerušen na dalším bodu přerušení nebo příkazem TRACE.

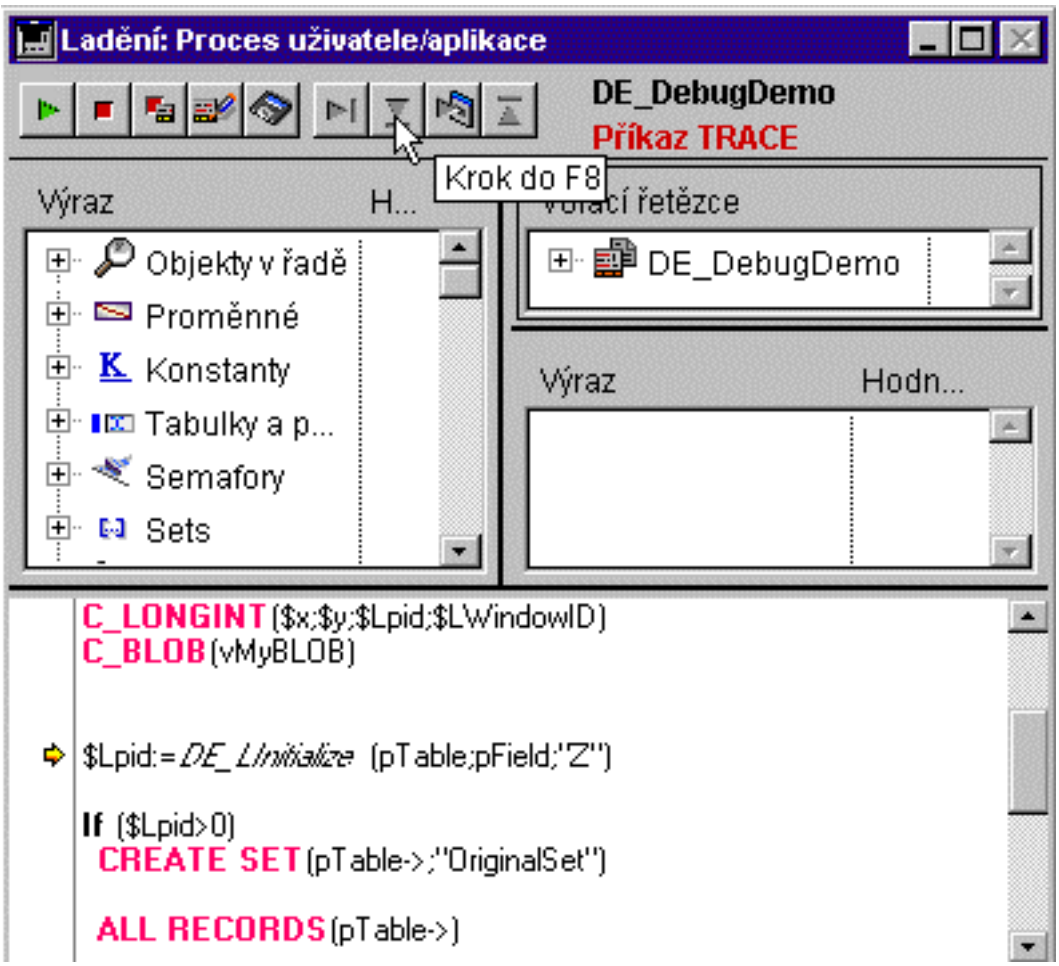
1. Klepněte na tlačítko Nekrokovat a ukončete tak krokování a pokračujte v provádění kódu.



2. Klepněte na tlačítko „Jejda“.
3. Zvolte Dema ¬ Debugger….
4. Postupně čtyřikrát klepněte na tlačítko Přeskočit.

3.3.4. Tlačítko Krok do

Toto tlačítko vám rovněž dovolí provádět metodu krok po kroku. Když je však v krokované metodě volána jiná metoda, toto tlačítko způsobí, že i tato metoda bude krokována. Toto tlačítko je rovněž aktivováno pomocí klávesy Return (klávesa Enter na abecední části klávesnice).

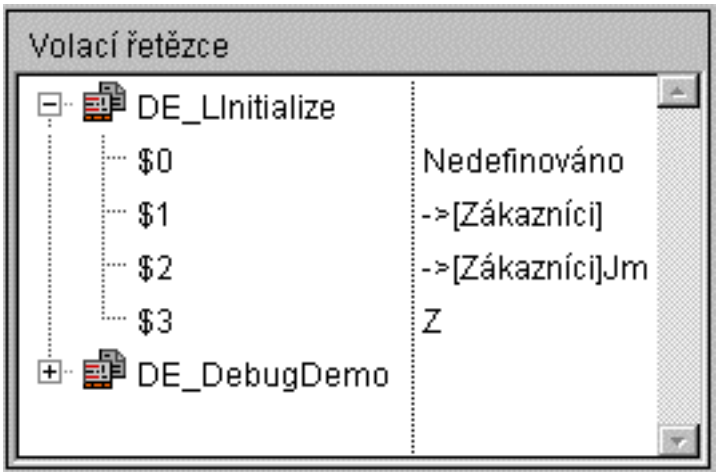


1. Vstupte do metody pomocí tlačítko Krok do.

3.4. Volací řetězce

Oblast Volací řetězce určuje jak byla metoda volána a rovněž hierarchii předchozích volání. Kromě toho Volací řetězec může být rozšířen a zobrazí hodnoty všech parametrů použitých v metodě.

1. Rozšiřte DE_Initialize k zobrazení seznamu parametrů.



Rovněž můžete na metodu poklepat a otevřít ji tak ze seznamu přerušení.

2. Poklepejte na DE_DebugDemo a otevřete tak metodu v Editoru metod.

Poznámka: Tento způsob rovněž funguje při poklepání na název metody v Průzkumníku provádění/přerušení.

3. Poklepejte na DE_LIntitalize a vraťte se tak do metody.
4. Krokujte přes komentáře.

3.5. Proměnné a array v oblasti výrazů

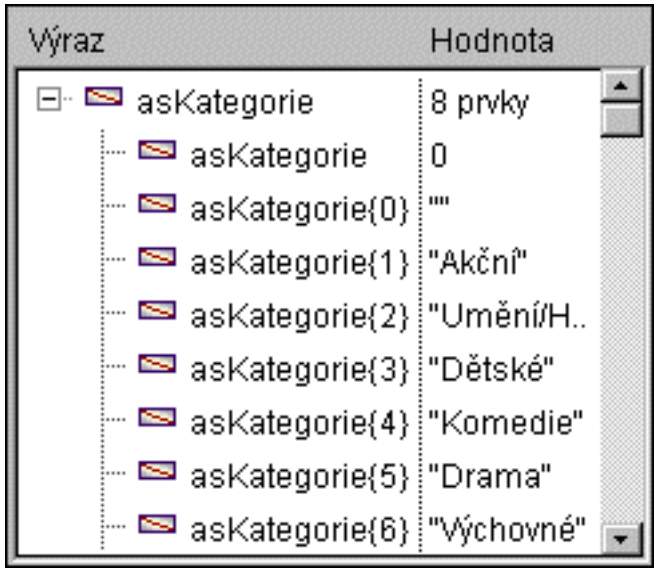
1. Krokujte až na řádek, který začíná příkazem QUERY
2. Rozšiřte oblast proměnných v okně výrazů.

Tato položka obsahuje hodnoty přidělené následujícím položkám:

Proměnné	**Rozsah**
Meziprocessní	Hodnota platná v databázi.

Proces	Hodnota platná v tomto procesu.
Místní	Hodnota platná v této metodě.
Parametry	Hodnota platná v této metodě.
Samoukazatel	Ukazatel na platný objekt (jsme-li v metodě objektu)

3. Rozšiřte a prohlédněte si každý z typů proměnných.
4. Určitě si rozšiřte a prohlédněte proměnnou procesu pTable.
5. Určite si rozšiřte a prohlédněte proměnnou procesu asKategorie.



Poznámka: V seznamu proměnný je zobrazeno pouze prvních 100 prvků array. Jestliže si chcete prohlédnout hodnotu prvku většího než sto musíte přesunout jeden z prvků do pravé části oblasti výrazu a změnit číslo prvku array na požadovaný prvek.

6. Pokračujte v prověřování místních proměnných a parametrů..
7. Krokujte za $LRecordsInSelection

Sláva! Nic nebylo nalezeno! Není divu, hledáme záznamy, které začínají na „Z“.

3.6. Pravá část oblasti výrazů

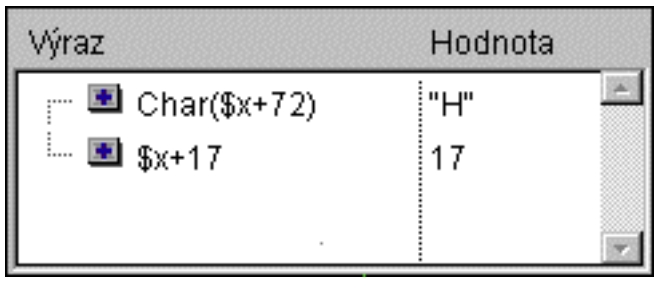
Pravá část oblasti výrazů je v pravé střední části okna ladění. Pokud chceme vložit položku do pravé části výrazů, máme k dispozici několik způsobů:

- Se stisknutou klávesou Ctrl, klepnout na objekt v oblasti metod.
- Rozšířit levou oblast výrazu tak, aby byl vidět požadovaný objekt a buď jej lze přetáhnout do pravé části a nebo na něj poklepat.
- Poklepat myší v pravé části oblasti výrazu a vepsat název objektu z klávesnice.
- Klepnout v pravé části oblasti výrazu pravým tlačítkem myši (Ctrl-klepnout) a ze zobrazené nabídky vybrat Nový výraz nebo Vložit příkaz.

3.6.1. Napsání vlastního výrazu v pravé části oblasti výrazů.

Poklepání v pravé části oblasti výrazů přidá prázdnou položku.

1. (CTRL + klepnout) (klepnout pravým tlačítkem) Přidat příkaz ¬ Řetězce ¬ Char
2. Pokračujte v zadávání nové řádky sledování a napište: ($x+72).
3. Poklepejte v pravé části oblasti výrazů.
4. Do nové řádky napište $x+17.



3.6.2. Odstranění výrazu z pravé části oblasti výrazů

K odstranění výrazu musíme klepnout na patřičný výraz a stisknout klávesu Delete. Pro ty, kteří jsou orientování na myš lze klepnout levým tlačítkem myši v pravé oblasti výrazu a odstranit všechny položky.

1. Odstraňte všechny položky z pravé oblasti výrazu.

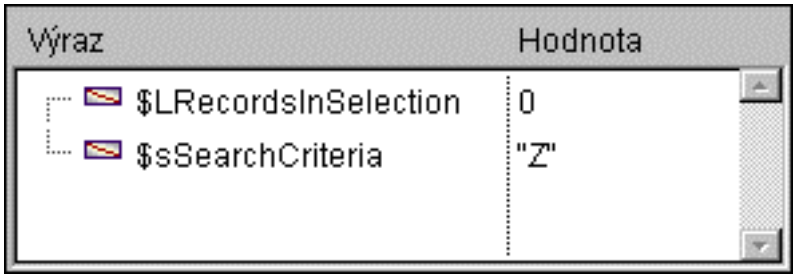
3.6.3. Přetažení položky z levé do pravé části oblasti výrazů

1. Rozšiřte místní proměnné v oblasti výrazů a přetáhněte $sSearchCriteria do pravé oblasti výrazů.

3.6.3. Poklepnutí na položku k přenesení do pravé části oblasti výrazů

Soustřeďme se na proměnnou $LRecordsInSelection.

1. Se stisknutou klávesou Ctrl poklepejte na $LRecordsInSelection v oblasti metody a přidejte ji tak do pravé části oblasti výrazů.



3.7. Změny hodnot objektů v okně ladění

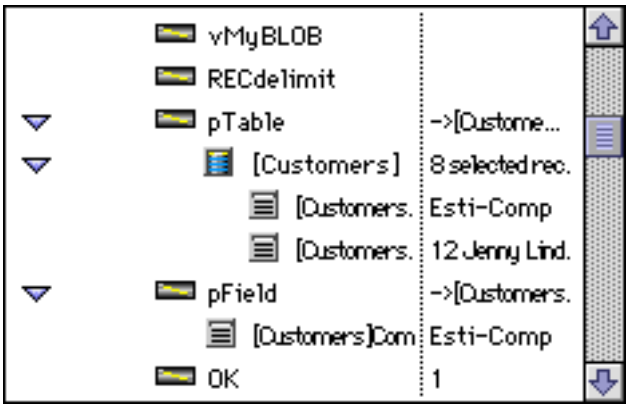
3.7.1. Změna hodnoty

Nebylo by pěkné, kdybychom mohli řídit běh programu aniž bychom museli neustále metody přepisovat? To je otevřít metodu v Editoru metod, změnit kód, uzavřít metodu a spustit znovu tuto část programu od samého začátku. Takto složitě to opravdu provádět nemusíme a změny hodnot proměnných můžeme provést přímo v okně ladění. Pro krokovanou metodu lze měnit hodnoty proměnných a parametrů buď v pravé části oblasti výrazů nebo v levé části oblasti výrazů.

1. Dokrokujme na řádek před příkaz QUERY.
2. Změňte místní proměnnou ***$sSearchCriteria*** na "E"
3. Proveďte krok kódu za řádek s $LRecordsInSelection line.

Proměnná $LRecordsInSelection je nyní větší než 0, protože byly nalezeny záznamy.

4. Prověřte hodnoty proměnných procesu pTable & pField v oblasti výrazů.



3.8. Krokování nového procesu

Když spustíte nový proces na témže stroji je možné otevřít nové další okno ladění pro tento proces současně se spuštěním procesu.

3.8.1. Tlačítko Krok do procesu

Toto tlačítko vám dovolí vstoupit s krokováním do nového procesu.

1. Dokrokujte na řádek kódu s funkcí New process.
2. Klepněte na tlačítko Krok do procesu a spusťte tak druhý proces.
3. Krokujte tento proces po několik řádek a pak klepněte na tlačítko Nekrokovat.

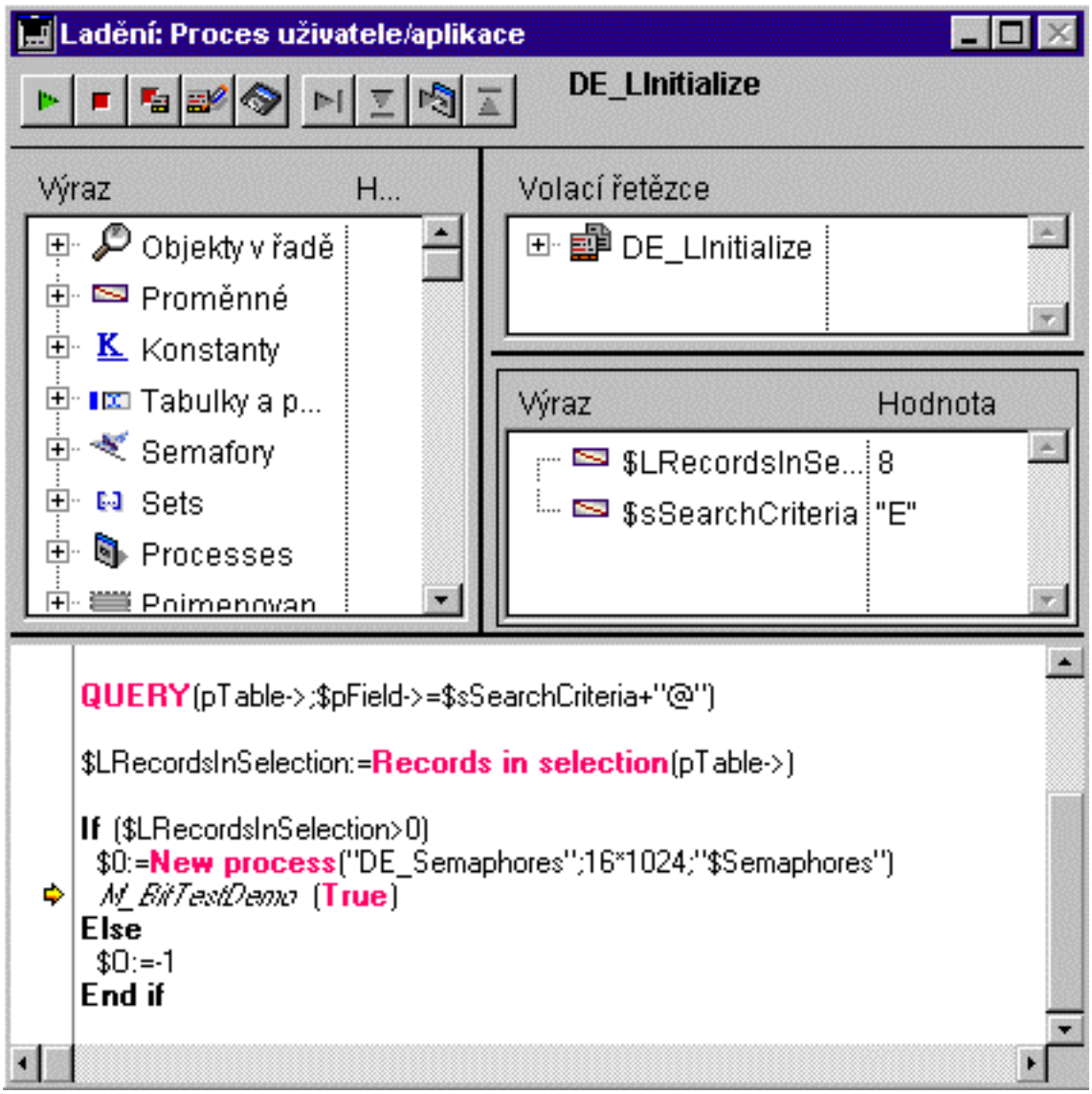
3.9. Návrat do volající metody.

Často je potřebné se vrátit do metody, která volala metodu, kterou právě krokujete, protože jste se přesvědčili, že není důvod pro pokračování krokování metody, kterou právě krokujete, ale nejste si tak jisti s metodou ze které jste přišli.

3.9.1. Tlačítko Vyskočit

Toto tlačítko ukončí krokování právě prováděné metody a když se provádění navrátí do předchozí úrovně kódu, obnoví se krokování na této úrovni.

1. Klepněte na tlačítko Vyskočit a opusťte tak krokování DE_Initialize.



2. Přesuňte tento dialog z hlavní části obrazovky.

3.9.2. Konstanty

Tato položka obsahuje hodnoty konstant používaných v databázi:

1. Prověřte konstanty v oblasti výrazů.

3.9.3. Pole

Tato položka vám umožní přístup k hierarchickému seznamu polí, který obsahuje platný záznam pro každou tabulku v prováděném procesu. Hodnoty zobrazené pro každou tabulku jsou počtem záznamů v platném výběru pro tuto tabulku.

1. Prověřte pole v oblasti výrazů.

3.9.4. Semafory

Tato položka obsahuje semafory, které jsou právě viditelné v prováděném procesu. Zahrnuje globální semafory pro celou databázi a lokální semafory pro tuto jednu pracovní stanici. V seznamu není uveden pouze název semaforu, ale rovněž proces, který semafor vytvořil.

1. V oblasti výrazů prověřte semafory.
2. Rozšiřte a zužujte poznámky semaforu a sledujte jeho změnu.

3.9.5. Sady / Sety

Tato položka obsahuje sady a uvádí tabulku ke které sada patří a rovněž počet záznamů v sadě.

1. Krokujte přes řádek kódu CREATE SET.
2. Prověřte v oblasti výrazů sety.

3.9.6. Procesy

Tato položka ukazuje všechny právě běžící procesy na této pracovní stanici včetně procesu Uživatele/Aplikace, procesu Návrháře, Cache Manager a Web server atd. Stav každého procesu je zobrazen ve sloupci hodnot.

1. V oblasti výrazu prověřte procesy.
2. Přejděte za řádku kódu ALL RECORDS.
3. Opět si prohlédněte část pole a všimněte si změny hodnot.

3.9.7. Pojmenovaný výběr

Tato položka obsahuje všechny pojmenované výběry a ukazuje ke které tabulce patří a rovněž počet záznamů ve výběru.

1. Přejděte za řádek CUT NAMED SELECTION.
2. V oblasti výrazů prověřte výběry.

3.9.8. Informace

Tento seznam obsahuje položky obecných informací a to platnou výchozí tabulku, událost formuláře, zda jste v transakci, zavedenou paměť, velikost vyrovnávací paměti, volnou paměť Stack, stav automatických vztahů ke skupině a k jedinci, směr dotazu (cílové uložení výběru), limit pro ukončení dotazu, úroveň triggeru a soubor k odeslání přes Web.

1. Prověřte v oblasti výrazu informace.

Zavedená paměť – slovní vyjádření stavu dostatku paměti

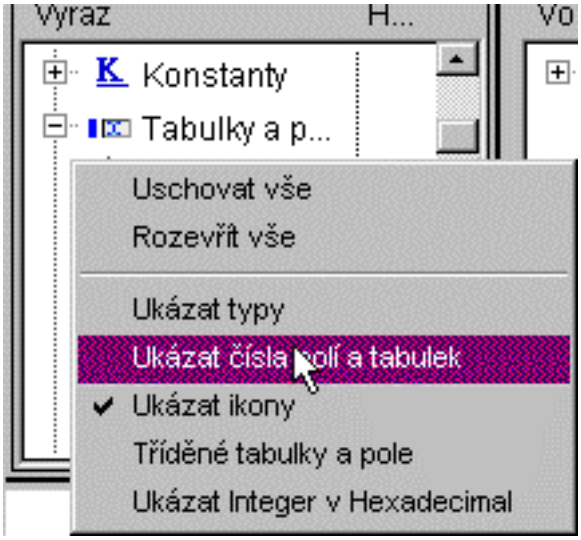
Vel. vyrovn. paměti - kolik paměti stack zbývá

Minimální velikost vyrovnávací paměti – celkem spotřebovaná velikost paměti stack v tomto procesu

3.9.9. Možnosti zobrazení a vkládání

Pro oblast výrazů můžete nastavit způsob a tvar zobrazení z nabídky dostupné pomocí Ctrl + klepnout (pravé tlačítko myši). Možnosti zobrazení jsou:

- Ukázat typy
- Ukázat čísla polí a tabulek
- Ukázat ikony
- Třídit pole a tabulky



1. Prověřte různé možnosti zobrazení a jejich účinek na různé části v oblasti výrazů.
2. Přesuňte $i & $x do pravé části oblasti výrazů.
3. Změňte hodnotu v $x na menší hodnotu.
4. Krokujte smyčkou dokud se opět neobjeví dialog.
5. Klepněte na OK a uzavřete dialog.

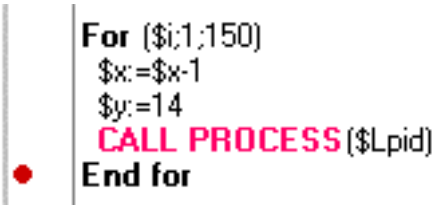
3.10. Nastavení přerušení při krokování

V průběhu krokování můžete přidávat dodatečná přerušení tak, že klepnete myší za levou svislou čáru v řádku metody kam chcete přerušení umístit.

K dočasnému znemožnění existujícího přerušení lze klepnout na červenou tečku znázorňující přerušení. Opětovné umožnění takto znemožněného přerušení se děje opět pomocí klepnutí do patřičného řádku.

3.10.1. Zavedení přerušení

1. Nastavte přerušení v řádce kódu End for.



2. Klepněte na tlačítko Nekrokovat.
3. Pozorujte hodnoty v pravé části oblasti výrazů.

3.11. Definování podmínek pro přerušení

Pokud klepnete (Option+klepnout), (Alt+klepnout) na tečku znázorňující přerušení, můžete v dialogovém okně, které se zobrazí po tomto klepnutí nastavit následující vlastnosti

- Umístění

  Ukazuje název metody a číslo řádky, kde je přerušení umístěno. Tato oblast se nedá upravovat, ale je nastavena již pomocí klepnutí myší, kterým přerušení zavádíme.

- Typ

  Dočasné - Přerušení, které bude odstraněno po ukončení metody. Dočasná přerušení jsou znázorňována zelenou tečkou.

  Trvalé - Přerušení, které je spolu s celou databází uloženo a je zobrazeno v průzkumníku provádění v seznamu přerušení. Výchozí nastavení pro toto přerušení je červená tečka. Při práci více vývojářů může být toto zobrazení svázáno s každým vývojářem, který jej vytvořil.

- Přerušit je-li následující podmínka pravda

  Tento řádek vám dovolí upravit svou podmínku. Syntaxy vašeho výrazu si můžete ověřit klepnutím na tlačítko Kontrola syntaxe.

- Kolikrát přeskočit před přerušením

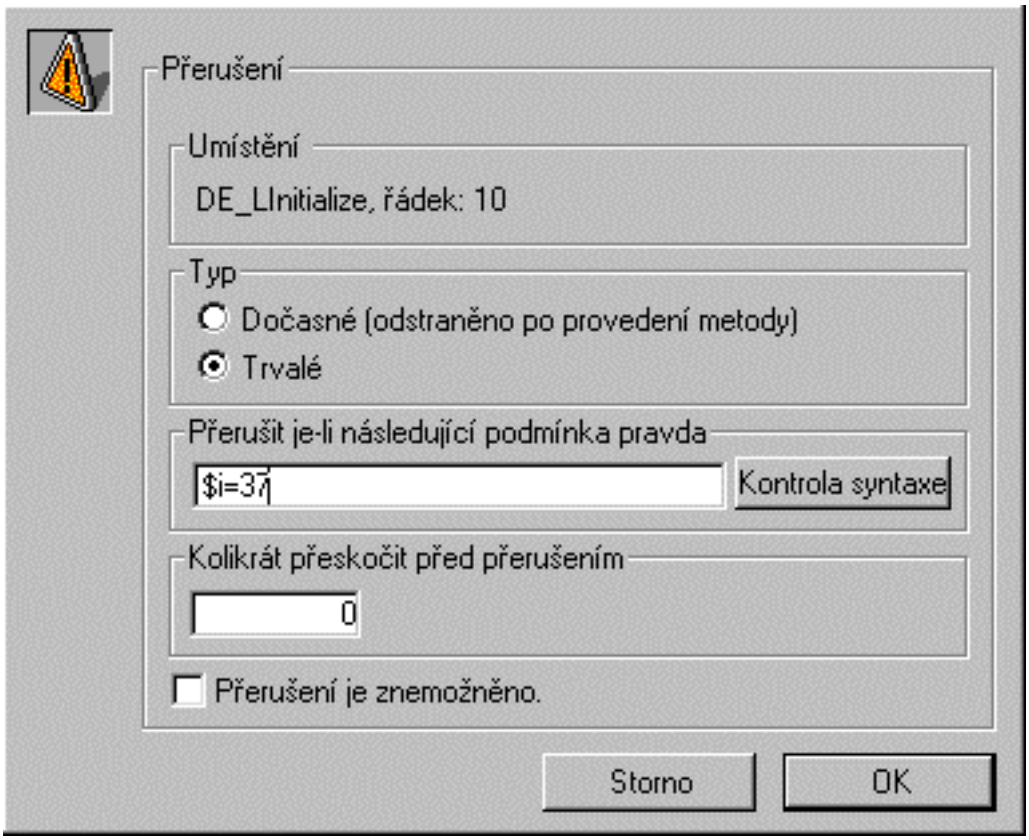
  Toto políčko vám dovolí určit kolikrát bude metoda spuštěna před provedením přerušení. Pokaždé, když je metoda spuštěna je tato hodnota snížena o 1 a při dosažení 0 a bodu přerušení se objeví okno ladění. Tato hodnota se automaticky nevrací k původní zadané hodnotě.

- Přerušení je znemožněno

  Toto zaškrtávací políčko vám dovolí znemožnit provedení přerušení. Znemožněné přerušení se zobrazí jako pomlčka jak v okně ladění tak v seznamu přerušení.

3.11.1. Nastavení podmínky přerušení

1. (Option + klepnout) (Alt + klepnout) na tečku znázorňující přerušení.
2. Nastavte podmínku $i=37

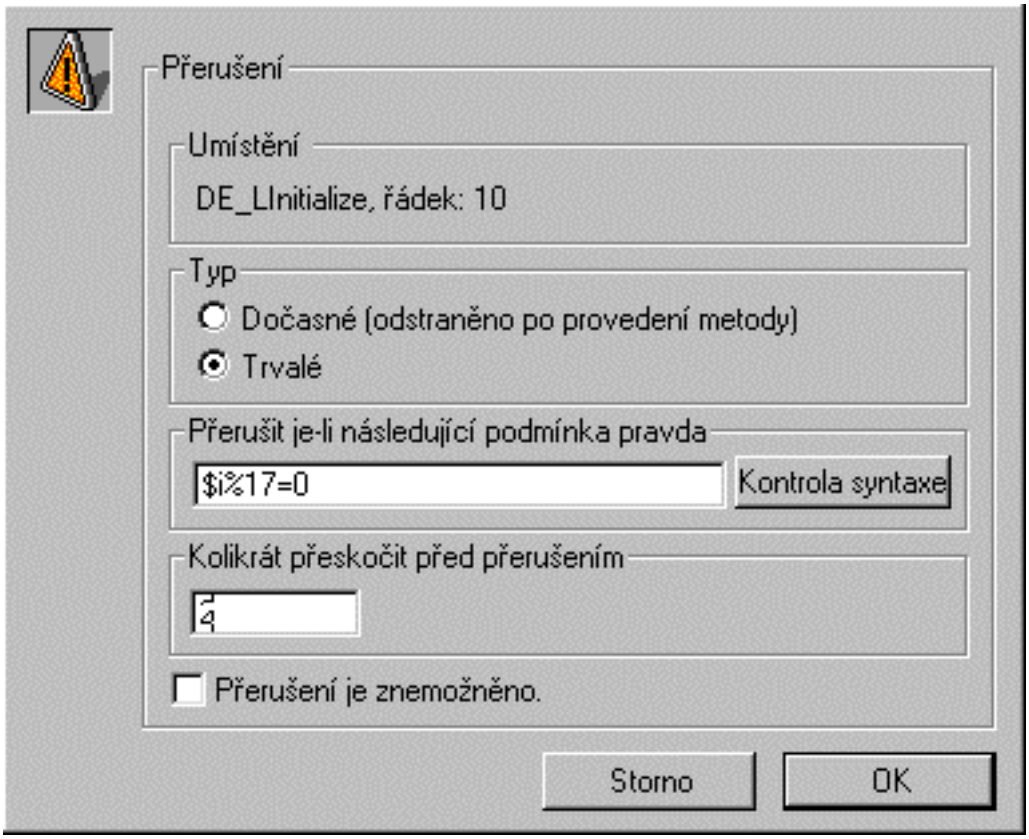


3. Klepněte na tlačítko OK.

4. Klepněte na tlačítko Nekrokovat**.**
5. Prohlédněte si změny hodnot v pravé části oblasti přerušení.
6. (Option + klepnout) (Alt + klepnout) na tečku znázorňující přerušení.

3.11.1. Nastavení podmínky pro přeskočení

1. Nastavte podmínku $i%17=0.
2. Klepněte na tlačítko Kontrola syntaxe.
3. Nastavte počet přeskočení na 2.

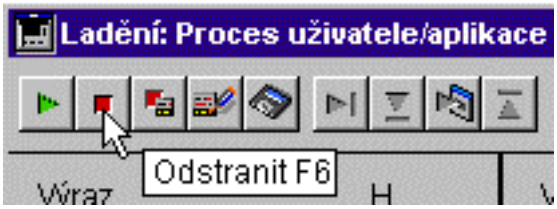


4. Klepněte na tlačítko OK.
5. Klepněte na tlačítko Nekrokovat**.**
6. Prověřte změny hodnot v pravé oblasti výrazů.
7. Nastavte přerušení před
   <>fDebugger:=False.

3.12. Další aktivní objekty okna ladění.

3.12.1. Odstranit

Toto tlačítko zastaví provádění metody.

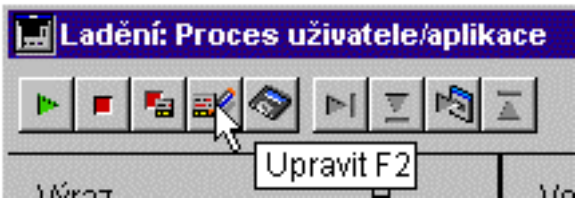


3.12.2. Odstranit a upravit

Toto tlačítko ukončí provádění metody a otevře metodu k úpravě v Editoru metod.

3.12.3. Upravit

Toto tlačítko otevře metodu k úpravě v Editoru metod aniž ukončí provádění metody.

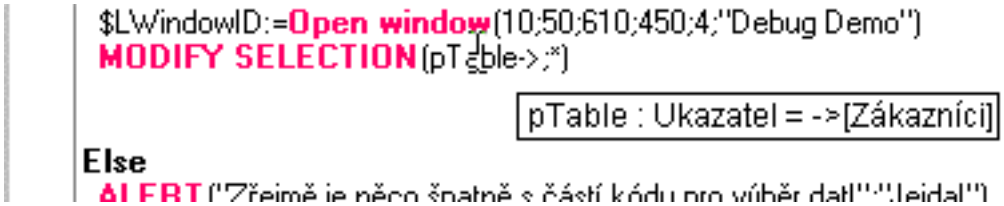


3.12.4. Zobrazení hodnot pod ukazatelem myši

Přemístěte ukazatel myši nad libovolnou proměnnou v oblasti metod. Pod ukazatelem myši se objeví současná hodnota této proměnné.

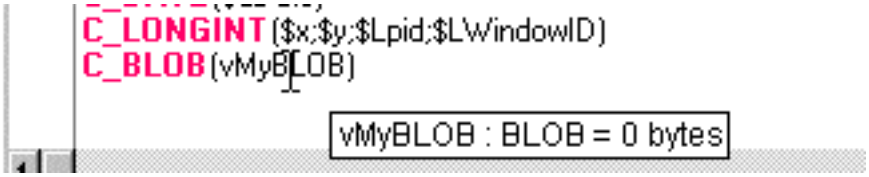
1. Klepněte na tlačítko Nekrokovat**.**
2. Prověřte hodnoty v oblasti výrazů a všimněte si, že se hodnota zvýšila sedmnáctkrát tím, že jsme dvakrát přeskočili přerušení. Hodnota přeskočení se nyní neobnovila.

3. Pokračujte v klepání na tlačítko Nekrokovat dokud nedosáhnete konečného bodu přerušení.



3.13. BLOBy a ladění

Tato verze okna ladění obsahuje mnoho dalších rysů, které vám napomohou. Např. jestliže přesunete ukazatel myši nad proměnnou či pole BLOB ve zdrojovém kódu, dozvíte se velikost BLOB:



Protože Debugger se nestará o to co ukládáte v BLOB musíte to zjistit jiným způsobem. Můžete si napsat určité vlastní metody, které mohou být použity v pravé části výrazů okna ladění. Tyto metody budou používány pro zobrazení dat a to všech nebo části uložených v BLOB.

1. V daném BLOB je uloženo pomocí kódu array. Prověřte obsah BLOB a to napsáním dvou řádek v oblasti výrazů.

   ```
   tBLOB2Text(->vMyBLOB;"TEXT";0;50)
   tBLOB2Text(->vMyBLOB;"HEXA";0;10)
   ```

2. Klepněte na tlačítko Nekrokovat a ukončete tak kód.
3. Vraťte se do Prostředí návrháře a prověřte všechna přerušení.

Metoda tBLOB2Text uvedená níže, vám umožní zobrazit v textu nebo v hexadecimální formě obsah BLOB.

```
If (False)
    ` Metoda: tBLOB2Text ( Pointer ; String { ; Long { ; Long } } )
    ` Kurz programovani ACI University
    ` Genericke metody z nastroju ACI
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Navrací hodnotu v BLOB buď jak  HEX nebo TEXT

  <>fGeneric :=True
  <>f_Version6x30 :=True
  <>fK_Wilbur :=True
```

```
End if

    ` Deklarace parametrů a přiřazení lokálních proměnných
C_TEXT($0)              ` Navracený text
C_POINTER($1)           ` Ukazatel na BLOB
C_STRING(4;$2;$sFormat) ` Format k návratu HEXA nebo TEXT
C_LONGINT($3;$LStart)   ` Pozice v BLOB k navracení
C_LONGINT($4)           ` Navracená délka

    ` Declarace místních proměnných
C_LONGINT($LEnd)        ` Poslední navracená pozice
C_LONGINT($LBlobSize)   ` Velikost BLOB
C_LONGINT($LLength)         ` Navracená délka
C_LONGINT($LParameters) ` počet parametrů
C_LONGINT($iByte)       ` Čílač smyčky

C_BLOB($vxHexDigits)

$LBlobSize:=BLOB Size($1->)-1   ` Zjisti velikost BLOB

If ($LBlobSize<0)
  $0 = "## Není BLOB nebo prázdný BLOB"
Else
                                ` Initializace výchozích hodnot parametrů
  $sFormat:="TEXT"
  $LStart:=0
  $LEnd:=$LBlobSize
  $LParameters:= Count parameters
                                ` Kopírování parametrů, jsou-li
  If ($LParameters >1)
    $sFormat:=$2
    If ($LParameters >2)
      $LStart:=$3
      If ($LParameters >3)
        $LEnd:=$LStart+$4-1
      End if
    End if
  End if
                                ` Kontrola konzistence parametrů
  If ($LStart<0)
    $LStart:=0
  Else
    If ($LStart>($LBlobSize-1))
      $LStart:=$LBlobSize-1
    End if
  End if

  If ($LEnd<0)
    $LEnd:=0
  Else
    If ($LEnd>($LBlobSize-1))
```

```
      $LEnd:=$LBlobSize-1
    End if
  End if

                                    ` V závislosti na formátu:
  Case of
    ś ($sFormat="TEXT")
      $LLength:=$LEnd-$LStart+1      ` Délka výsledného textu
      If ($LLength>32000)
        $LEnd:=32000+$LStart-1  ` Příliš velké, oříznout
        $LLength:=32000
      End if
      $0:=" "*$LLength              ` Initializace výsledného textu
      For ($iByte;0;$LEnd-$LStart-1)
        $0[[$iByte+1]]:=Char($1->{$LStart+$iByte})  ` Copírovat byty jedem po
druhém
      End for

    ś ($sFormat="HEXA")
      SET BLOB SIZE($vxHexDigits;16)  ` Vytvoř lokální BLOB pro Hexa čísla
                                                `01234546789ABCDEF
      For ($iByte;0;9)
        $vxHexDigits{$iByte}:=48+$iByte
      End for

      For ($iByte;10;15)
        $vxHexDigits{$iByte}:=65+$iByte-10
      End for

      $LLength:=($LEnd-$LStart+1)*3          ` Délka výsledného textu
      If ($LLength>32000)
        $LEnd:=10666+$LStart-1               ` Příliš velké, oříznout
        $LEnd:=32000
      End if

      $0:=" "*$LLength                    ` Initializace výsledného textu
      For ($iByte;0;$LEnd-$LStart-1)
                                          ` Calculate High nibble pro byte
        $0[[($iByte*3)+1]]:=Char($vxHexDigits{$1->{$LStart+$iByte}\16})
          ` Calculate Low nibble pro byte
        $0[[($iByte*3)+2]]:=Char($vxHexDigits{$1->{$LStart+$iByte}%16})

    Else
      $0:="## Chyba: Formát zobrazení musí být TEXT nebo HEXA"
  End case
End if
    ` konec metody
```

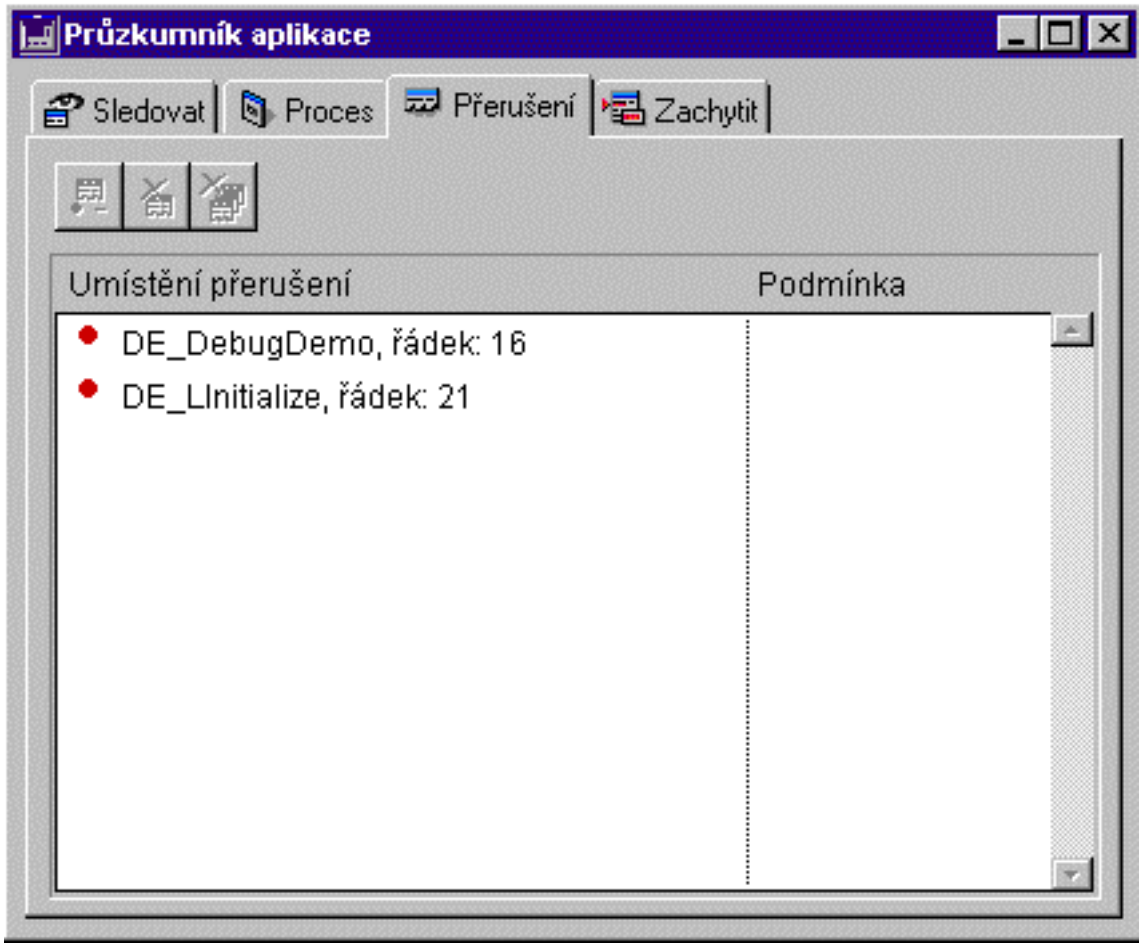
Po přidání metody do vašeho projektu ji můžete používat nejenom k vytváření textových výrazů, ale rovněž přímo v oblasti výrazů v okně ladění.

3.14. Prověření umístění přerušení a metod

Všechna zavedená přerušení během ladění můžete prověřit tím, že se vrátíte do okna Průzkumníka provádění a seznamu přerušení.

3.13.1. Použití okna průzkumníka aplikace k prověření metod.

1. Otevřete Průzkumník provádění na stránce Přerušení



2. Poklepejte na určité přerušení.
3. Po prověření metody odstraňte nové přerušení.

4. Formuláře a objekty formulářů

Formuláře byly oproti v3 podstatně vylepšeny. Zabírají méně místa v paměti, jsou rychlejší, byl upraven jejich vzhled a byly přidány nové rysy. Zopakujeme si některé údaje z předchozích kursů, pro ty kteří se teprve seznamují s v6 to nebude tedy nic nového.

4.1. Vzhled objektů formuláře.

Proti předchozí verzi 4D se vzhled objektu stává vlastností objektu. Existuje šest základních vzhledů pro všechny objekty formuláře.

- Žádné

  Všechny objekty se objeví ve svém přirozeném grafickém nastavení podle vybrané platformy a volby uživatele.

- Normální

  Neohraničené objekty se objeví ve svém běžném grafickém nastavení podle vybrané platformy.

  Ohraničené objekty se objeví s orámováním šířky jednoho bodu.

- Tečkovaný

  Neohraničené objekty se objeví ve svém normálním grafickém nastavení podle platformy. Tečkované ohraničení přepíše vybraný vzor. Ohraničené objekty se objeví s tečkovaným orámováním šířky jednoho bodu..

- Vypouklý

  Všechny objekty se objeví s třídimenzionálním vypouklým efektem.

- Ponořený

  Všechny objekty se objeví s třídimenzionálním ponořeným efektem.

- Dvojitý

  Na počítačích Macintosh se objekty objeví s ohraničením dvojitou čarou: tj. dvě plné 1bodové čáry oddělené jedním bodem.

  Na Windows se objekty zobrazí s ohraničením jedné černé a jedné bílé čáry posunuté o jeden bod.

4.2. Vzhled dle rozhraní pro platformu

Verze 4D 3.5.x byla první verzí 4D s možností automatického přizpůsobení aplikace dle systému spuštěného na počítači. 4D automaticky rozpozná systém na kterém je aplikace spuštěna (Macintosh, Windows 95, Windows NT, Copland) a vzhled každého objektu se přizpůsobí zvyklostem na daném systému. Pro přizpůsobení vzhledu objektu můžete zvolit několik úrovní:

- přizpůsobení všech objektů databáze
- přizpůsobení jednoho formuláře
- přizpůsobení konkrétního objektu ve formuláři

4.3. Nové aktivní objekty formuláře

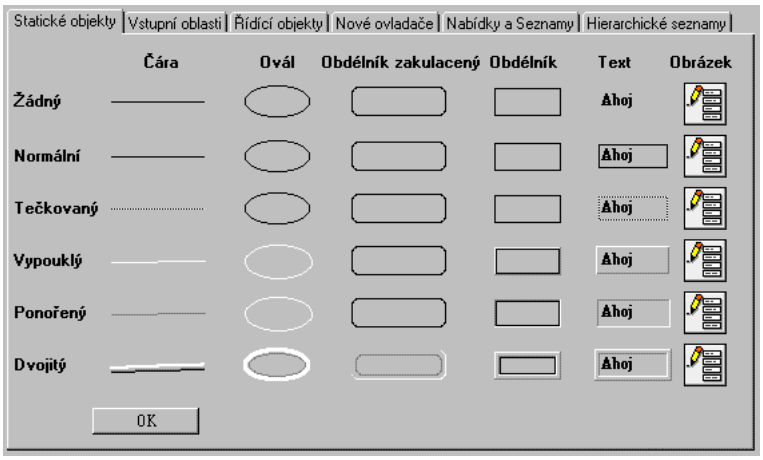
Škála objektů 4D se podstatně rozšířila. Počet objektů se proti v3 zdvojnásobil.

4.3.1. Přehled nových objektů a jejich vzhledu

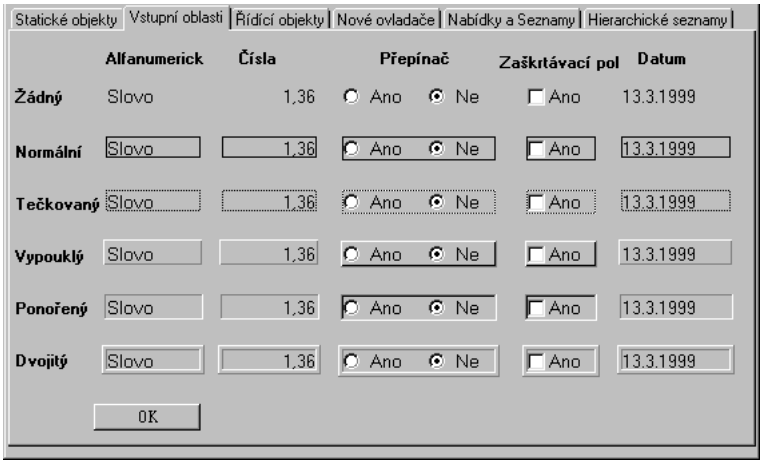
Tato další databáze byla navržena, aby poskytla rychlý přehled všech jednotlivých objektů a jejich nastavení vzhledu pro různé platformy.

1. Spusťte databázi Objekty demo.
2. Listujte stránkami a zkoušejte různé objekty a jejich nastavení pro platformy.

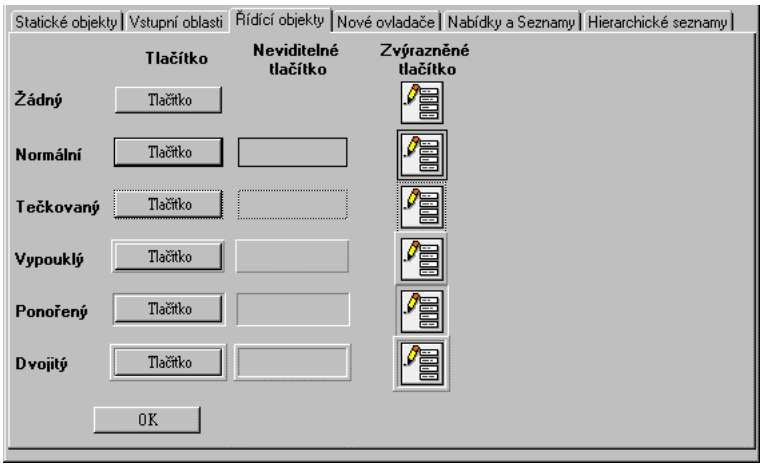
Statické objekty



Oblasti s možností zadávání

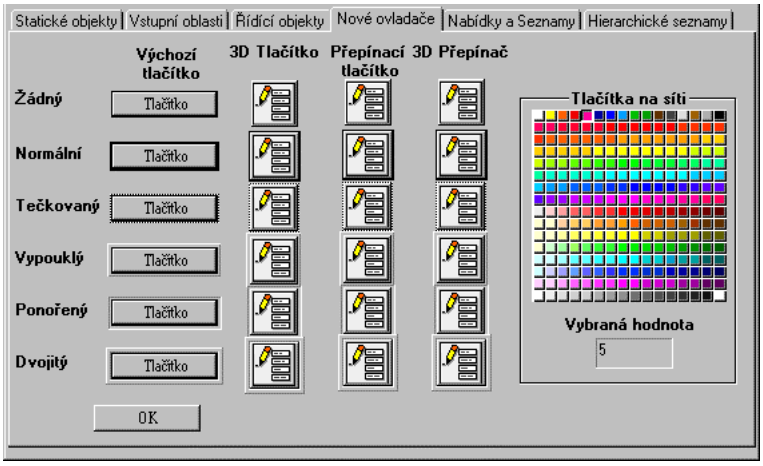


Staré řídící prvky



Neviditelná tlačítka (nejsou příliš neviditelná pokud použijete špatný typ vzhledu)

Nové řídící prvky

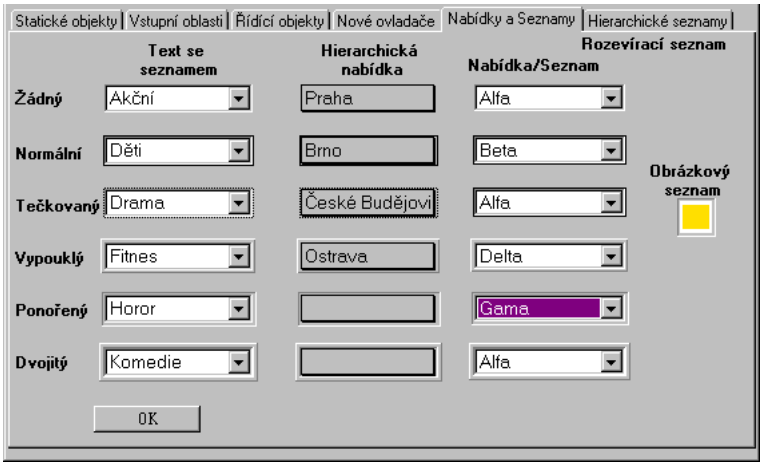


Statický objekt Tlačítka na síti je zde nazván Síť tlačítek.

Síť tlačítek

> Je to jedna průhledná proměnná, která je umístěna nad obrázkem. Tato jedna proměnná se chová jako více nezávislých tlačítek. V předchozí verzi by jste museli vytvořit 255 tlačítek, aby jste mohli dosáhnout téhož matematického efektu a samozřejmě spoustu práce, aby jste dosáhli téhož vizuálního efektu. Při použití sítě tlačítek je navracená hodnota rovna souřadnici xy stisknutého tlačítka na síti.

Nabídky a seznamy



Řízení karty

Tento ovládací prvek je zde používán k přesunům z jedné stránky formuláře na druhou, jako listování v kartotéce. Tento ovládací prvek je vhodný způsob k prezentování všeho co je dostupné v uživatelském formuláři nebo dialogu.

Nabídka/Seznam

Tento seznam je naplňován pomocí array a jeho chování je částečně závislé na platformě. Vybraný prvek je zobrazen v okně seznamu a proměnná seznamu je nastavena na číslo zobrazeného prvku. Hodnota zobrazeného prvku je přesunuta do prvku 0.

Text se seznamem

Text se seznamem je vzhledově podobný objektu Nabídka/seznam kromě toho, že políčko zobrazující prvek je dostupné a lze do něj zapisovat. Úpravy v dostupné oblasti se promítají do prvku 0 připojeného array. Proto je k ovládání tohoto objektu potřeba testovat prvek 0 připojeného array.

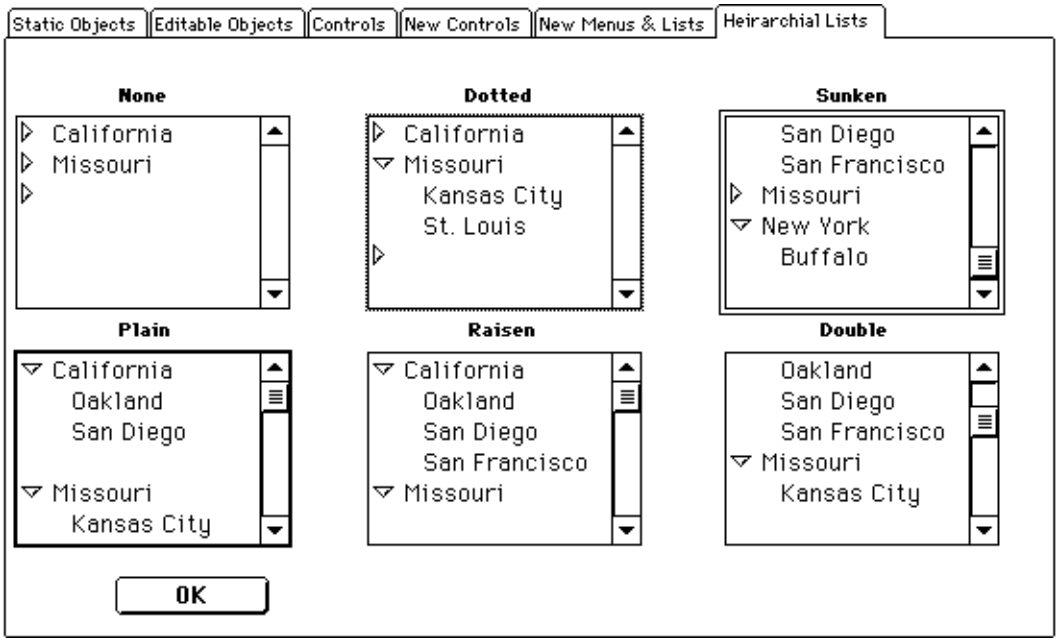
Hierarchická nabídka

Hierarchická nabídka je podobná Nabídce/seznam s tím rozdílem, že je hierarchická. To znamená, že dovoluje zobrazení více úrovní nabídky a ke každému prvku může existovat připojená podnabídka.

Obrázkový seznam

Tento typ Nabídky/seznam je nabídkou obsahující obrázky. K použití tohoto objektu musíte použít obrázky a rovněž musíte určit kolik řádek a sloupců nabídky obsahuje. Číslo prvku, které bylo vybráno je vráceno při výběru prvku.

Hierarchický seznam



Hierarchický seznam je podobný Posuvné oblasti je však hierarchický. Dovolí vám zobrazit podseznamy připojené ke každému zobrazenému prvku seznamu.

4.4. Strana pozadí

Každý formulář obsahuje svou Stranu pozadí. Vše co vložíte na tuto stránku se objeví a funguje na každé další stránce formuláře. Stejný efekt lze dosáhnout několikanásobným umístěním, vložením objektu na každou stránku formuláře. Vše co umístíte na stránku pozadí musíte doopravdy chtít na každé stránce a funkce těchto objektů na Straně pozadí musí být na každé stránce formuláře požadovány.

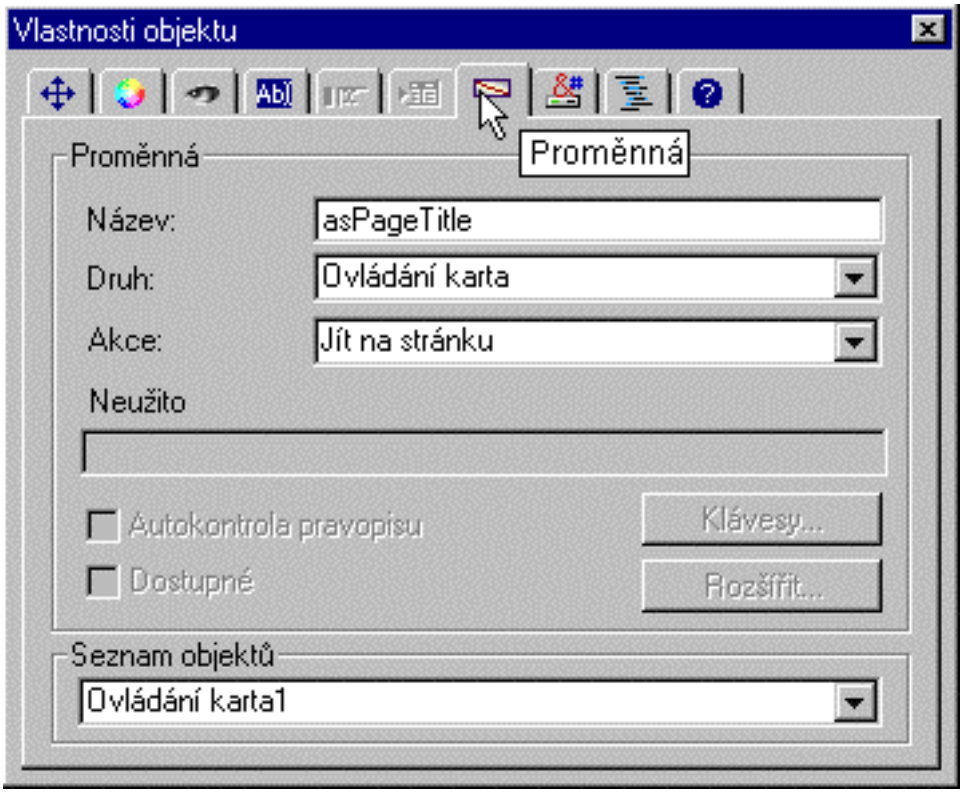
Stránka pozadí je umístěna na stránce číslo 0.

4.4.1. Vyzkoušení požadavků nových aktivních objektů

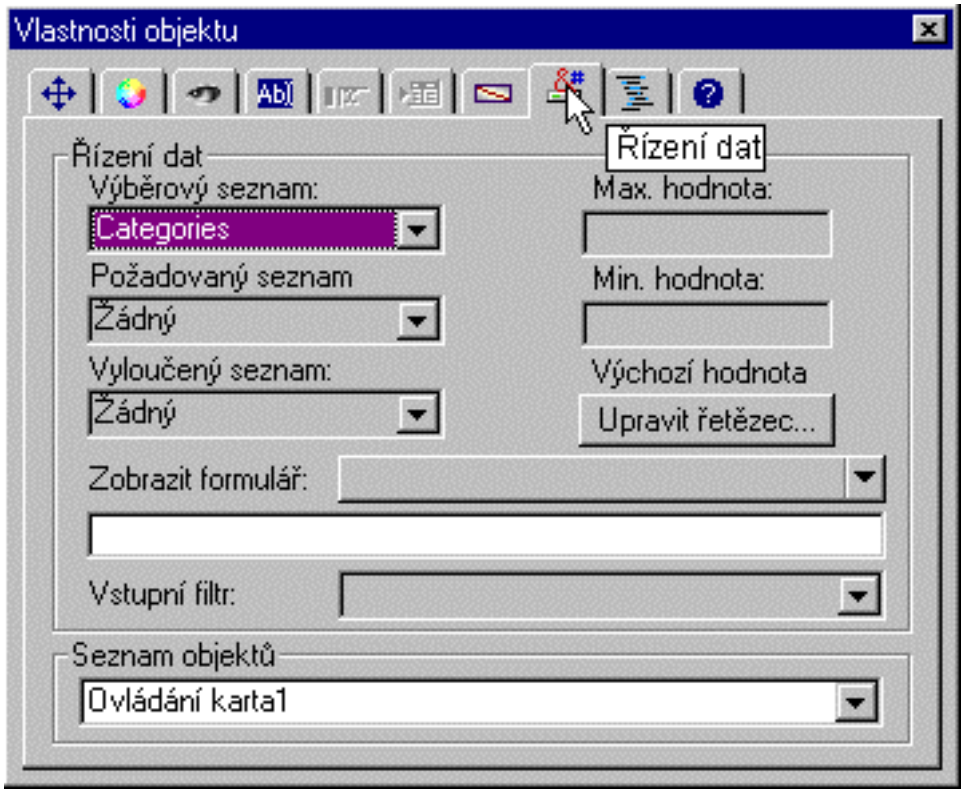
Mnoho z nových objektů vyžaduje speciální zacházení. Předtím než začnete tyto objekty používat vyzkoušejte si co požadují a jak jsou používány.

1. Otevřete formulář [zDialogy]FormObjects.
2. Přesuňte se na Stránku pozadí tohoto formuláře.
3. Otevřete dialog vlastností objektu pro objekt Řízení karta.

   Řízení karta je schopné zobrazovat hodnoty obsažené v array, hierarchický seznam, který nemá podseznamy nebo speciální seznam řetězců. Ovládání jednotlivých stránek může mít automatickou akci Jít na stránku.

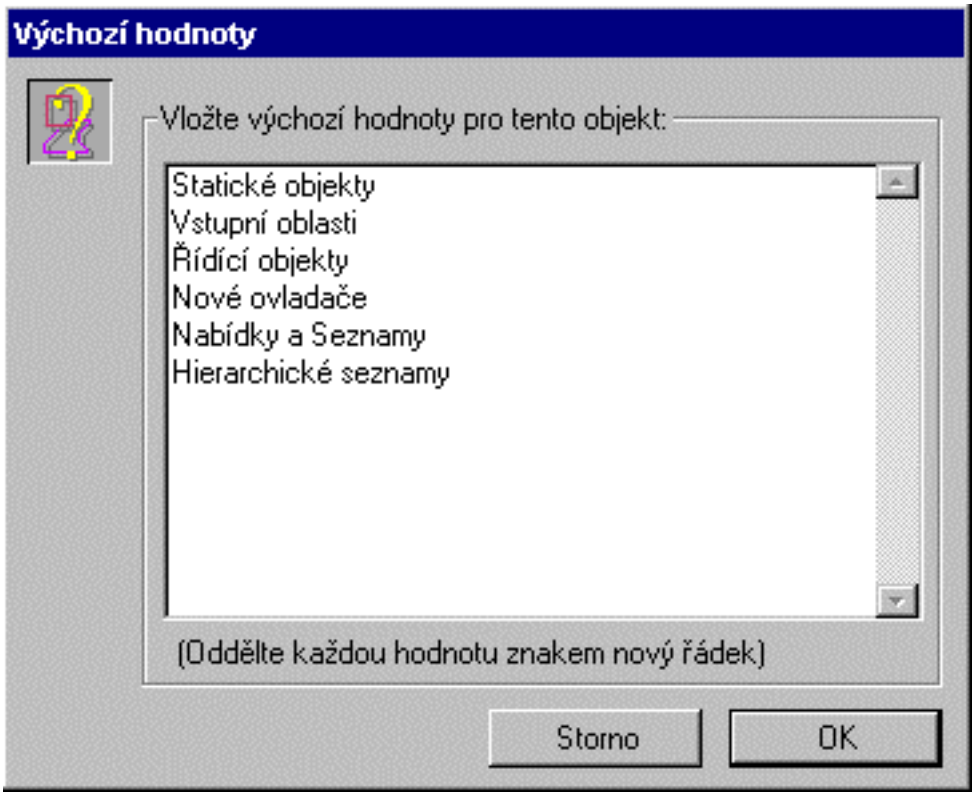


4. Přejděte na stránku Řízení v okně Vlastnosti objektu.



5. Stiskněte tlačítko Upravit řetězec.

   Tato speciální oblast je použita pro vkládání textu ovládacích prvků. Když dokončíte vkládání jednotlivých prvků, uvidíte v Prostředí návrháře náhled všech ovládacích prvků.

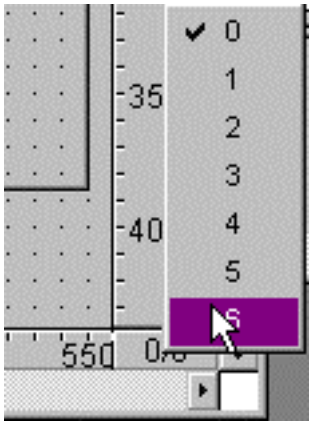


6. Otevřete metodu databáze Při spuštění.

   Tato metoda ukazuje jiný způsob definování řídících prvků ovládání karty.

```
` Tento komentovaný kód definuje ovládací prvky karty.
` ARRAY STRING(31;asPageTitle;6)
` asPageTitle{1}:="Statické objeky"
` asPageTitle{2}:="Vstupní oblasti"
` asPageTitle{3}:="Řídící objekty"
` asPageTitle{4}:="Nové ovladače"
` asPageTitle{5}:="Nabídky a seznamy"
` asPageTitle{6}:="Hierarchické seznamy"

ARRAY STRING(20;asSelectPage;3)
asSelectPage{1}:="Look & Feel"
asSelectPage{2}:="Fun Stuff"
asSelectPage{3}:="Drag & Drop"
```

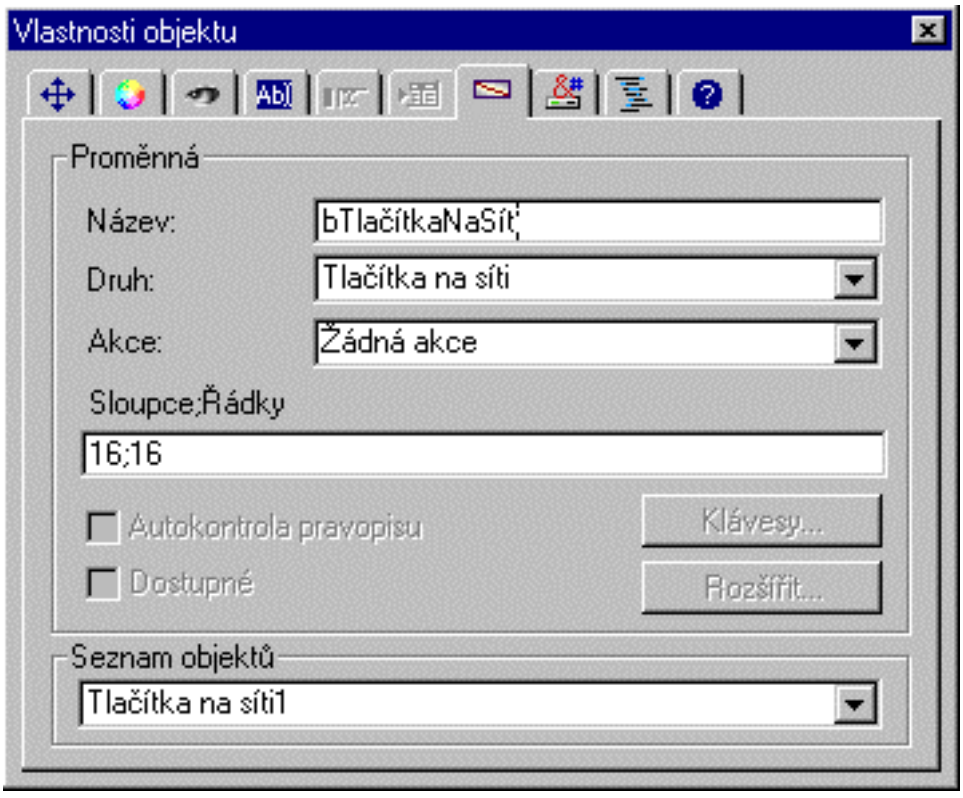
7. Klepněte na nabídku 0/6 a přejděte na stránku 4.



8. Pokud je objekt Síť tlačítek umístěn na pozadí, přeneste si jej na popředí.

   Tento objekt se sítí tlačítek je umístěn vždy na vrchu grafického obrázku. Používá se k určení na kterou část obrázku uživatel klepl..

9. Poklepejte na objekt tlačítka na síti a otevřete dialogové okno definic objektů.



Tlačítka na síti jsou číslována z levého horního rohu doprava a dolů. V našem příkladu je síť 16 sloupců a 16 řádek dolů. Vrchní levé tlačítko navrátí při klepnutí hodnotu 1.

Další dvě číslice určují horizontální a vertikální mezery mezi prvky na síti. Pokud zde použijete nenulovou hodnotu definujete tak bílý rámeček kolem každého tlačítka na síti.

10. Otevřete metodu objektu pro tlačítka na síti.

```
bTlačítkaNaSíti:=String(bTlačítkaNaSíti)
```

Nijak zvláštní kód, ale lze z něj získat představu jak tlačítka na síti pracují.

11. Přesuňte se na stránku 5 formuláře.

Text se seznamem je podobný Nabídce/seznam kromě toho, že objekt dovoluje přijímat text zadávaný z klávesnice. Text se seznamem se inicializuje pomocí array. Jestliže uživatel napíše do objektu text naplnil tak prvek 0 patřičného array. Jestliže chcete, aby vstup provedený uživatelem byl dostupný z připojeného seznamu musíte řídit programem prvek 0 připojeného array.

12. Vytvořte novou metodu projektu GEN_ManageComboBox a napište následující nebo importujte pomocí textového editoru.

```
If (False)
      ` Metoda: GEN_ManageComboBox(pointer;longint;{boolean;{string}})
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97
      ` Účel: Manages changes in the combo box array

   <>fGeneric:= True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definovat parametry
C_POINTER($1;$pArrayToCheck)       ` Ukazatel k array
C_LONGINT($2;$LDataLength)         ` Délka dat array
C_BOOLEAN($3;$fSortArray)          ` Třídit výsledek (výchozý = True)
C_STRING(31;$4;$sListName)         ` Název seznamu k obnovení (výchozí =  není)

      ` Deklarace lokalnich promennych
C_LONGINT($LElementLocation)` umístění nových dat a array
C_LONGINT($LNewSizeOfArray)` Novávelikost array
C_LONGINT($LParameters)        ` Počet parametrů
C_STRING(15;$sValueEntered)    ` Hodnota vložená uživatelem

      ` Přiřazení parametrů k místním proměnným
$pArrayToCheck:=$1
$LDataLength:=$2
$fSortArray:=True
$sListName:=""                     ` Nastavit jako výchozí bez názvu
$LParameters:= Count parameters
If ($LParameters  > 2)
   $fSortArray:=$3
   If ($LParameters  > 3)
      $sListName:= $4
   End if
End if

If (Form event = On Data Change)  ` Uživatel změnil hodnotu textu se seznamem
   $sValueEntered:=$pArrayToCheck->{0}
   $LElementLocation:=Find in array($pArrayToCheck->;$sValueEntered)
   If ($LElementLocation<=0)
      $LNewSizeOfArray:=Size of array($pArrayToCheck->)+1  ` Vzít platnou velikost
array plus jedna
      ARRAY STRING($LDataLength ;$pArrayToCheck->;$LNewSizeOfArray)  `
Změnit velůikost array
```

```
      $pArrayToCheck->{$LNewSizeOfArray}:=$sValueEntered    ` Piřadit novou
hodnotu k array

      If ($fSortArray) ` Otestoa třídění
         SORT ARRAY($pArrayToCheck->) ` Přetřídit array
      End if


      If (Length($sListName )>0)  ` Otestovat jestli seznam k obnovení
         ARRAY TO LIST($pArrayToCheck->;$sListName )
      End if
   End if
End if
      ` Konec metody
```

13. Otevřete metodu objektu pro text se seznamem cbCategory a přidejte následující kód:

```
If (False)
      ` Metoda: cbCategory
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97
      ` Účel: Řídit změny v array text se seznamem

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

GEN_ManageComboBox(Self;15)
      ` Konec metody
```

14. Otevřete metodu objektu pro cbCategory1 a přidejte následující kód:

```
If (False)
      ` Metoda: cbCategory1
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97
      ` Účel: Řídit změny v array text se seznamem

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

GEN_ManageComboBox(Self;15;False)
      ` Konec metody
```

15. Otevřete metodu objektu pro text se seznamem cbCategory2 a přidejte následující kód:

```
If (False)
      ` Metoda: cbCategory2
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97
      ` Účel: Řídit změny v array text se seznamem

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

GEN_ManageComboBox(Self;15;True;"Categories")
      ` Konec metody
```

16. Vraťte se do procesu Vlastní nabídky a testujte chování prvních tří textů se seznamem.

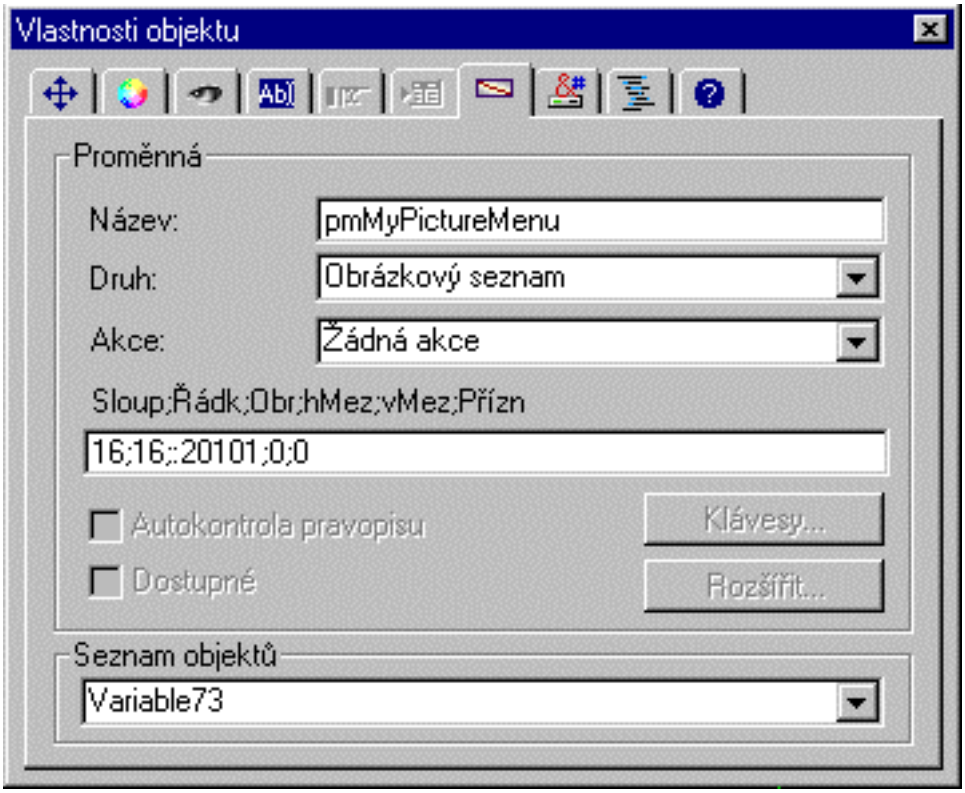
    První text se seznamem upravuje array připojený k seznamu a vložené prvky jsou tříděny. Druhý seznam netřídí prvky array. Poslední nejenom dovoluje úpravy a třídí array, ale rovněž ukládá změny do seznamu ve struktuře.

17. Vraťte se do Prostředí návrháře na stránku 5 formuláře [zDialogy]FormObjects.

    Hierarchické nabídky a Hierarchické seznamy (stránka 6) jsou dvěma různými způsoby zobrazení seznamů, kde prvky seznamu mají či nemají podnabídky nebo podseznamy připojené ke každému prvku. Těmito seznamy se budeme do větších podrobností zabývat později v tomto kurzu.

18. Poklepejte na obrázkový seznam a otevřete dialog definic objektů.

    Obrázkový seznam je nabídka, která zobrazuje dvoudimenzionální array obrázků. Koncept je velice podobný tlačítkům na síti kromě toho, že grafika je skutečně součástí nabídky a není pouze obrázkem pod objektem.



Opět v definici vidíme, že musíme určit počet sloupců a řádek a rovněž mezery v hMargin & vMargin.

Zde však třetí položkou je odkaz na grafiku použitou v obrázkovém seznamu. Jsou tři způsoby jak definovat grafiku pro obrázkový seznam.

- Grafika je proměnná. Jednoduše použijte název proměnné jako třetí parametr.
- Grafika je zdroj PICT umístěný v souboru zdrojů (.res) buď na úrovni vaší struktury nebo v samotné 4D. K odkazu na tento typ grafiky umístěný ve zdroji PICT ID# musíte použít číslo zdroje PICT. Toto číslo zdroje musí být uvedeno dvojtečkou. V našem příkladu používáme PICT 20101 proto je náš třetí parametr „:20101“.
- Grafika je obrázek typu PICT umístěný v knihovně obrázků (více později) Abychom mohli použít grafiku umístěnou v knihovně obrázků musíme opět použít číslo odkazu na obrázek v knihovně plus 65536. Tento odkaz musí být rovněž uveden „:“ (obrázek s číslem 1 z knihovny obrázků odkazujeme „:65537“).

4.5. Styly

Editor stylů vám umožní nastavit různé vlastnosti písma – písmo, velikost písma a styl – a rovněž tuto celou sadu stylu pojmenovat. Styly mohou být pak použity k určení vlastností písma pro jednotlivé objekty na stránce Styly v návrháři formulářů nebo na stránce Písma v okně definic objektů.

Každý pojmenovaný styl obsahuje oddělenou sadu vlastností písma pro každou ze tří platforem podporovaných 4th Dimension. Např. platforma Macintosh může používat písmo Geneva CE zatímco Windows 95 a Windows NT mohou používat písmo MS Sans Serif. Rovněž je pro tyto platformy možné použít rozdílnou velikost písma.

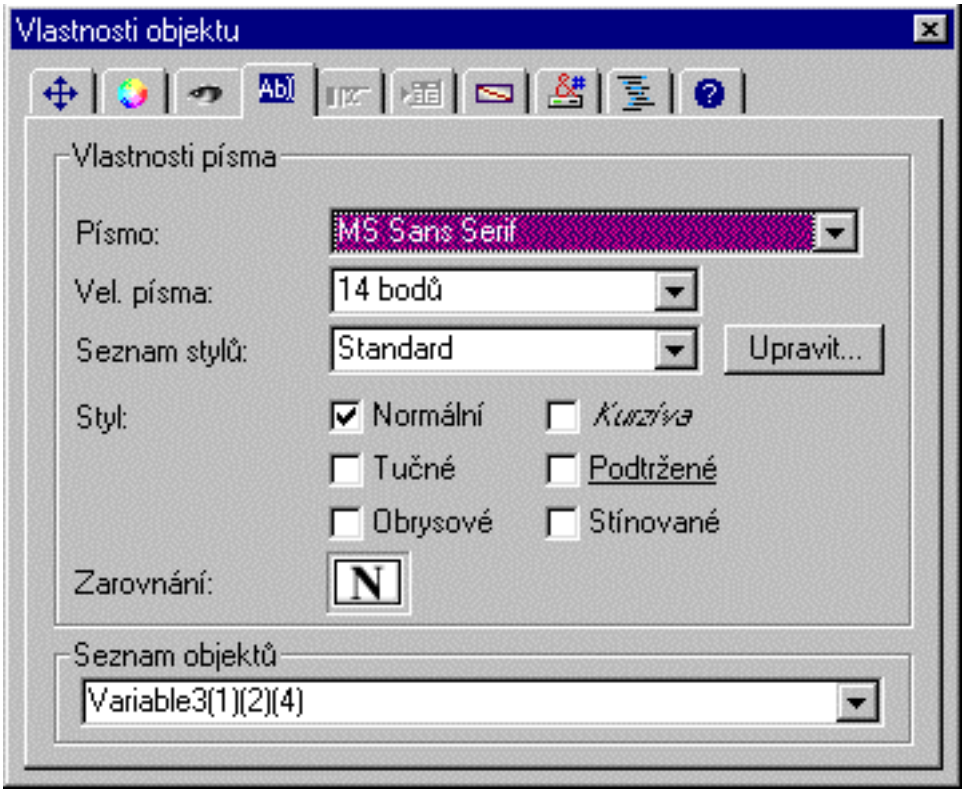
Důrazně vám doporučujeme, aby jste vždy když vytváříte databázi pro použití na různých platformách používali a přiřazovali styly ke každému textovému či číselnému objektu ve všech formulářích.

4.5.1. Vyzkoušení stylů.

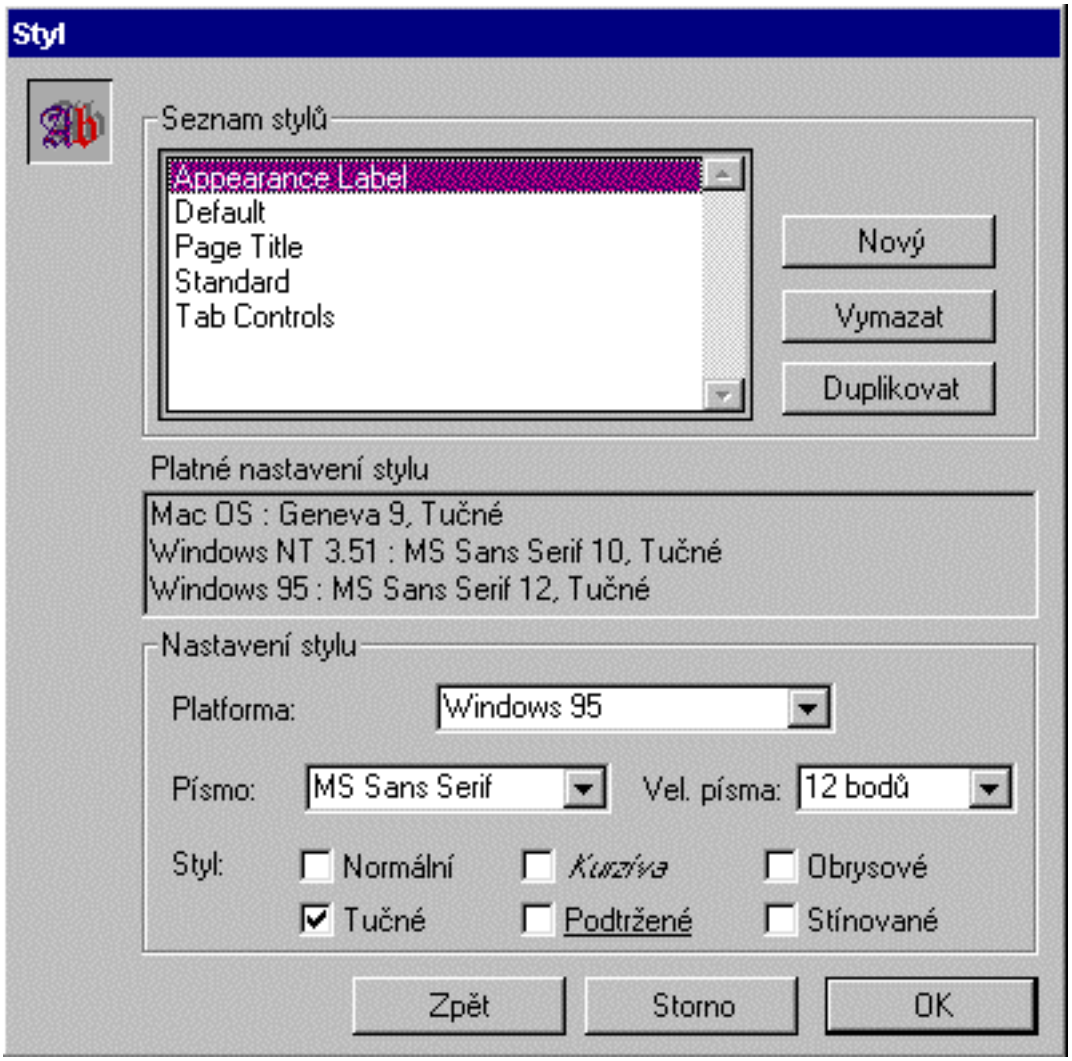
1. Klepněte na některý aktivní objekt na stránce 5 formuláře.

Všimněte si jak se dialog vlastností objektů přepne na nový objekt.

2. Přejděte na stránku Písmo v dialogu.



3. Klepněte na tlačítko Upravit… v řádce Styly.

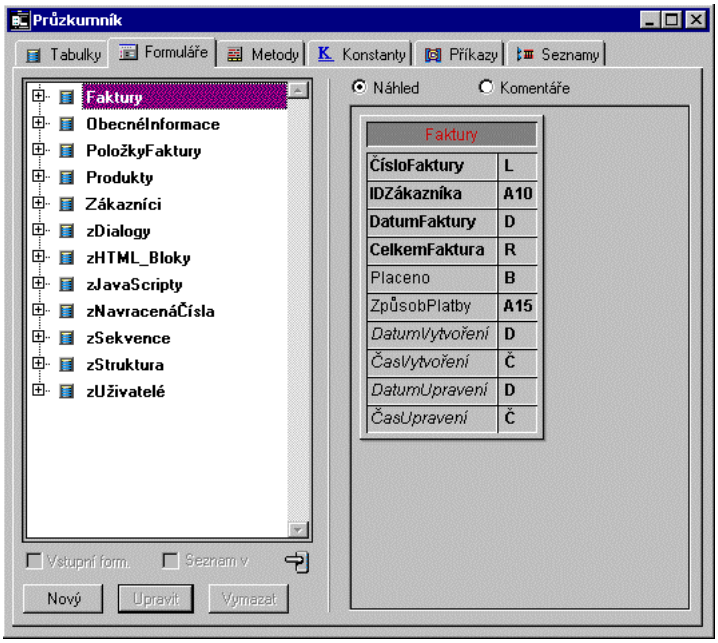


4. Klepněte na tlačítko OK.
5. Klepněte se stisknutým Shift a vyberte tak všechny objekty.
6. Změňtě styl na Tab Controls.

Poznámka: Najednou můžete vybrat více objektů a změnit současně vlastnosti těchto objektů najednou.

5. Začínáme s hierarchickými seznamy

Jedna z věcí, které je dobré si uvědomovat, že 4D byla použita k vytvoření většiny formulářů, které můžete vidět v samotné 4D. Největší vyjímkou je okno ladění. Podívejme se tedy na okno průzkumníku, který byl vytvořen ve 4D. Jak toto provést ?



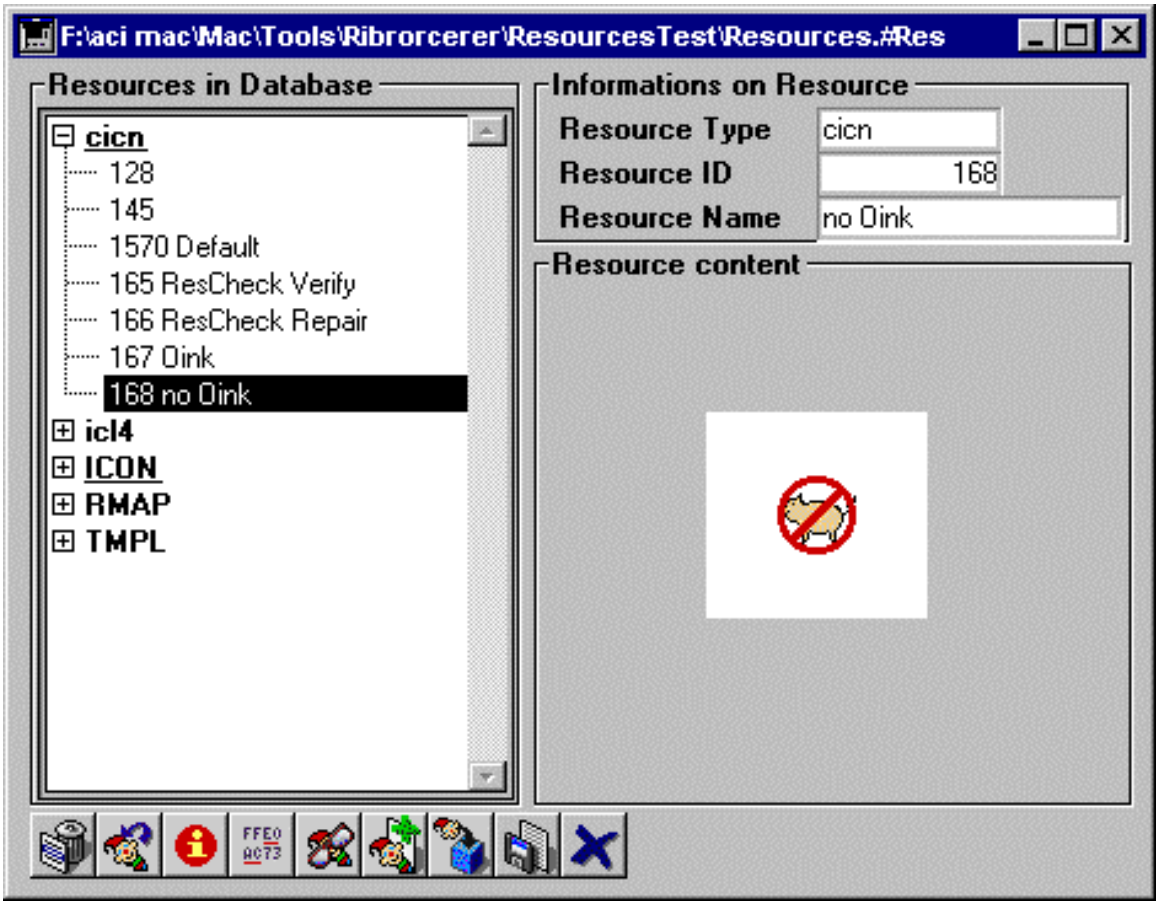
V poslední kapitole jsme hovořili o ovládání karty a viděli jsme, jak vytvořit toto ovládání pomocí array. Toto výše má text a obrázky. Je to array obrázků a array textu v kombinaci ? Ne tak úplně. Podívejme se blíže na okno průzkumníka. Je zde rovněž seznam tabulek, metod a další co je potřeba. Koncepce hierarchických seznamů je k pochopení jednoduchá. Je zde text a obrázek, jako ovládací prvek k rozšíření a zůžení seznamu. Povšimněme si obdoby ovládacího obrázku a textu v hierarchickém seznamu a obrázku a textu v jednom prvku ovládání karty. Obdoba zde je, ve skutečnosti jsou to věci o stejné podstatě, oba objekty jsou hierarchické seznamy textu s přiřazenými obrázky ke každému prvku seznamu. Jsou zde pouze dvě odlišnosti. Ovládání karty nemá podseznamy a grafická reprezentace zobrazení je jiná.

5.1. Obrázky v hierarchických seznamech

Obrázky používané v hierarchickém seznamu mohou mít původ ze třech zdrojů :

- 'cicn' zdroje původem z MacOS
- 65536 + 'PICT' zdroj z indexovaného souboru zdrojů z MacOS
- 131072 + číslo odkazu z knihovny obrázků

'cicn' nabízí výhodu pro automatický výběr ze zdrojů barevných a černobílých s patřičnou maskou pro zobrazení. Je však nutno mít zkušenosti s programy umožňujícími tento typ zdrojů vytvářet (ResEdit (Mac). Rebrocerer (Mac,Win).



Další dvě možnosti jsou vhodné i pro začátečníky. Je nutno si pamatovat, že zdroje musí mít číslo větší než 15000. Při nižších číslech by jste mohli přesměrovat zdroje ze samotné 4D nebo systému MacOS.

5.1.1. Pravidla pro zobrazení prvků karty s obrázky.

Jestliže je okno dost široké budou zobrazemy popisky i obrázky. Jestliže se okno zúží mohou být zobrazeny v řadě jen obrázy, pokud se nevejdou ani ty budou zobrazeny pouze záložky. Důrazně vám doporučujeme, vyhnout se takovémuto typu zobrazení (možnosti zůžení okna), protože je pro uživatele nepřijatelné.

1. Experimentujte s pravidly zobrazení, pomocí okna průzkumníka.

5.1.2. Znedostupnění záložky.

Vytvoření vzhledu záložky tak, aby tato indikovala, že daná stránka je nedostupná pro uživatele, je jednoduché. V tomto případě pouze nastavíte vlastnost položky hierarchického seznamu, připijeného k ovládání karta, jako nedostupné, tj. False.

5.2. Příkazy hierarchických seznamů

Bdeme používat několik příkazů z ovládání hierarchických seznamů pro náš úkol vytvořit ovládání karty.

5.2.1. New list -> ListRef

Příkaz New list vytvoří v paměti nový hierarchický seznam a vrátí jeho jedinečné číslo odkazu.

Upozornění : Hierarchický seznam je vždy obsažen pouze v paměti. Když skončíte s užíváním hierarchického seznamu použijte příkaz CLEAR LIST. Tímto příkazem uvolníte paměť zabranou již nepotřebným hierarchickým seznamem.

5.2.2. APPEND TO LIST (list; položkaText; PoložkaRef{; podlist{; rozšiř}})

Parametr	Typ	Popis
list	ListRef	číslo odkazu na seznam
položkaText	řetězec	Text nové položky seznamu (max. 31 znaků)
PoložkaRef	číslo	Jedinečné číslo odkazu na položku
podlist	ListRef	Možný hierarchický podseznam pro novou položku
rozšiř	logické	Indikuje, zda seznam bude rozšířen či zavřen

Příkaz APPEND TO LIST přidá novou položku do hierarchického seznamu, na nějž jste se odkázali jeho číslem.

Předáváte text položky. Můžete předat textový výraz do 31 znaků, jestliže předáte více, text bude oříznut.

Předáváte číslo odkazu na položku. I když by toto číslo mělo být jedinečné, pokud chcete aby fungovalu, ve skutečnosti můžete předat libovolné číslo.

Jestliže chcete, aby tato položka měla potomky, t.j další navázaný seznam, předejte platné číslo odkazu na tento seznam, který se zde takto stane podseznamem a v rozšiř parametru příznak, zda tento seznam bude rozšířen, nebo zúžen.

Předávané číslo odkazu musí být odkazem na existující seznam. Tento může být prázdný, obsahovat jednoúrovňový seznam, nebo může sám být seznam s podseznamy. Jestliže k nové položce nechcete přiřadit podseznam, tento parametr vynechejte, nebo předejte 0. Jestliže předáte podlist a vynecháte parametr rozšiř bude výchozí zobrazení nerozšířené.

5.2.3. SET LIST ITEM PROPERTIES (list; položkaref; dostup; styl; ikona)

Změna vzhledu položky seznamu a přidání obrázku se provede příkazem SET LIST ITEM PROPERTIES. Je jednodušší provést tak okamžitě po přidání nové položky seznamu, protože na právě přidanou položku se odkážeme nejjednodušeji.

Parametr	Typ	Popis
list	ListRef	číslo odkazu na seznam
položkaref	číslo	č. odkazu na položku nebo 0, poslední přidanou do seznamu
dostup	logické	TRUE = dostupné, FALSE = nedostupné
styl	číslo	styl písma položky
ikona	číslo	'cicn' zdroj 65536 + 'PICT' zdroj 131072 + číslo obrázku v knihovně

5.2.4. Selected list item (list) -> číslo

Příkaz vrátí pozici vybrané položky seznamu v seznamu, jehož číslo jsme předali.

Budete používat tento příkaz na seznam zobrazený ve formuláři k určení, která položka byla uživatelem vybrána.

Jestiže seznam má podseznamy, použijte tento příkaz na hlavní seznam (ten, kerý je skutečně uveden ve formuláři) a ne na některý podseznam. Pozice je vyjádřena relativně k vrchu seznamu a do úvahy se berou viditelné položky podseznamů.

5.2.5. CLEAR LIST (list{; *})

Parametr	Typ	Popis
list	ListRef	číslo odkazu na seznam
*		Je-li uvedena, podseznamy jsou rovněž mazány

Příkaz CLEAR LIST vymaže seznam, jehož odkaz byl předán.

Obvykle budete předávat i volitelný parametr * tak, aby byly vymazány i všechny podseznamy, přiřazené seznamu.

5.2.6. Is a list (list) -> logické

Parametr	Typ	Popis
list	ListRef	číslo odkazu na seznam
Výsledek funkce	Logické	zda je seznam právě platný hierarchický seznam

Vytváření hierarchických seznamů je z hlediska paměti nebezpečná věc. Jestiže se chystáte vytvořit hierarchický seznam, musíte mít jistotu, že jste pro předchozí výskyt tohoto seznamu použili CLEAR LIST, než znovu použijete New List. Jestliže to neprovedete můžete se octnout v problémech s pamětí. Na druhou stranu, pokud se snažíte použít CLEAR LIST na neexistující seznam, způsobíte krach aplikace dokonce i v kompilované verzi.

Problémům se vyhnete použijete-li CLEAR LIST vždy před New list a vždy před CLER LIST požijete Is a List.

```
If (Is a list( MyList))
   CLEAR LIST(MyList)
End if
```

5.3. Hierarchické seznamy a paměť

Abyste dobře rozuměli co se dějr se seznamy a pamětí budeme diskutovat několik možností v Pascalu a C.

Mějme následující kód

```
myhandle:=New handle (10000);
```

Tento kód nalezne místo v paměti a nastaví nový referenci na tento paměťový blok. Handle je odkazován přes ukazatel v sekci hlavních ukazatelů pměti, takže se vytvoří dvě úrovně odkazů na paměťový blok. Aby byl tento objekt (blok) vždy dostupný označí příkaz New handle tuto část paměti jako uzamčenou a tato část nemůže být přemístěna, kdež je paměť přesouvána manažerem paměti.

Jestliže to můžeme provést jednou, proč ne dvakrát?

```
myhandle:=New handle (10000);
myhandle:=New handle (10000);
```

Výsledek těchto kroků je, že i druhá část paměti je uzamčena pomocí příkazu New handle. Původní blok paměti zůstává uzamčen a všechny odkazy na tuto část paměti jsou ztraceny. Tento problém vyústí v nedostatek paměti, a je-li takováto část programu volána stále znovu program na nedostatek paměti zkrachuje.

Před novým použitím odkazu handle musíme uvolnit handle, aby paměť byla odemčena a uvolněna pro nové použití, nebo přemístění.

```
myhandle:=New handle (10000);
Dispose handle(myhandle );
myhandle = New handle (10000);
```

Pokračujme vytvořením jiného handle duplikováním odkazu na původní handle.

```
myhandle:=New handle (10000);
Dispose handle(myhandle );
myhandle:=New handle (10000);
myhandle2:=myhandle ;
Dispose handle(myhandle );
Dispose handle(myhandle2);
```

Jestliže musíme uvolnit handle co je špatného na posledním řádku kódu ? Jestliže jste si odpověděli, že paměť byla již vyčištěna máte pravdu. Nemůžeme dvakrát odemknout tentýž blok paměti.

Co má všechno toto společné s hierarchickými seznamy? Hierarchické seznamy jsou takové odkazy (handle) na paměťové bloky. Nyní poprvé ve 4D tímto způsobem můžete vytvořit vlastní nedostatek paměti. Nevolejte v tomto případě ACI technickou podporu, aby jste ohlásili chybu 4D, i když si budete absolutně jisti tato chyba nebude naše!

Následující kód vytváří všechny výše zmíněné problémy:

```
hlCustomerTabs:=New list
hlCustomerTabs:=New list
hlCustomerTabs2:=hlCustomerTabs
```

```
CLEAR LIST(hlCustomerTabs )
CLEAR LIST(hlCustomerTabs 2)
```

Ačkoliv si jste zcela jisti, že seznam neexistuje, měli by jste vždy prověřit, že to co se snažíte přiřadit k seznamu není již seznam a vždy kontrolovat, že to co chcete mazat je stále seznam.

Vytvoření seznamu by mělo být napsáno následovně:

```
If(Is a list(hlCustomerTabs ))
   CLEAR LIST(hlCustomerTabs )
End if
hlCustomerTabs:=New list
```

Vymazání seznamu by mělo vypadat následovně:

```
If(Is a list(hlCustomerTabs ))
   CLEAR LIST(hlCustomerTabs )
End if
```

Poznámka: Příkaz Is a list zkrachuje 4D jestliže položka, kterou použijete ke kontrole nebyla nikdy užita (není definována). Proto na počátku procesu zahrňte následující kód:

```
C_LONGINT(hlCustomerTabs )
hlCustomerTabs :=0;
```

5.4. Konstanty

4D V6 má část, která se nazývá konstanty. Konstanty jsou rezervovaná slova, která zastupují určité vlastní neměnné číselné hodnoty ve 4D. Když jsou použita v kódu zobrazí se podtrženě.

Příklad:

```
Blue            ` value is 6     Black           ` value is 15
4D Client       ` value is 4     Carriage return ` value is 13
```

Konstanty můžete použít namísto číselných hodnot, tak aby jste vytvořili čitelnější kód.

5.5. Vytvoření ovládání karty s obrázky

Nyní je čas kombinovat výše zmíněné tak, abychom vytvořili pro uživatel příjemnější rozhraní. Vytvořili jsme pro vás určité zdroje 'cicn' :

Customer data = 15001

Invoices = 15002

Poznámka: Chystáme se vložit zdroje pomocí ResEdit nebo jiným nástrojem umožňujícím zacházet se zdroji. Je to jediný způsob jak vyplnit tuto úlohu.

5.5.1. Vytvoření ovládání karty s obrázky.

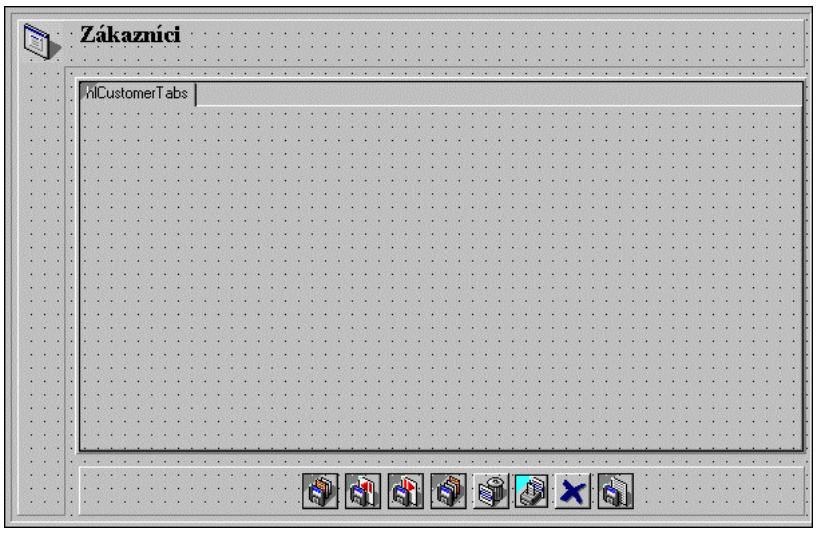
1. Předtím než spustíte databázi ACI Video, vložte zdroje CICN a další zdroje pomocí zvláštního programu umožňujícícho zacházet se zdroji do souboru struktury.
2. Spusťte databázi ACI Video.
3. Vytvořte novou metodu projektu CUSTOMER_Initialize následujícím způsobem nebo importujte pomocí textového editoru.

```
If (False)
     ` Metoda: CUSTOMER_Initialize
     ` Kurz programovani ACI University
     ` Autor: Kent Wilbur
     ` Datum: 3/17/97

     ` Účel: Inicializuje položky potřebné v procesu Zákazníci

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

hlCustomerTabs:=New list          ` Je to nový proces tak vytvoř seznam
APPEND TO LIST(hlCustomerTabs;"Customer Data";1)
SET LIST ITEM PROPERTIES(hlCustomerTabs;0;True;Plain;15001)
APPEND TO LIST(hlCustomerTabs;"Invoices";2)
SET LIST ITEM PROPERTIES(hlCustomerTabs;0;True;Plain;15002)
     ` Konec metody
```

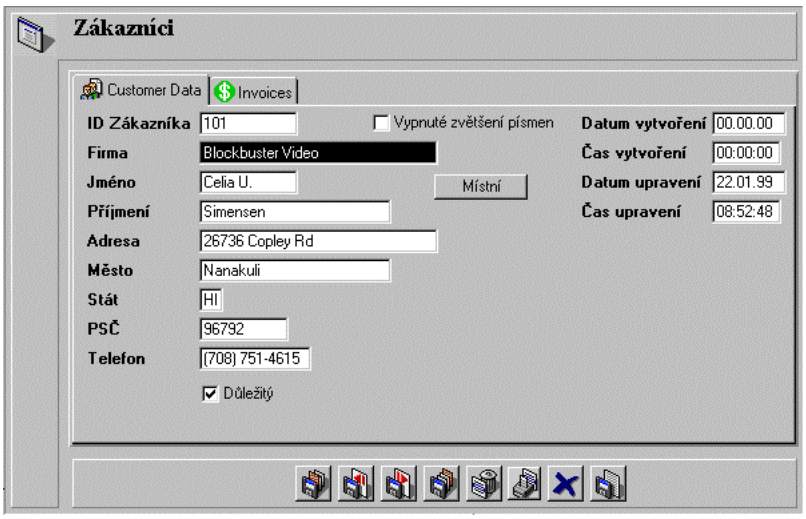
4. Upravte metodu projektu P_Customers následujícím způsobem:

```
…
❖   CUSTOMER_Initialize

    INITIALIZE_Process(->[Zákazníci])
    LOCK_Tables2ReadWrite (pTable)


    GEN_ModifySelection

❖   If (Is a list(hlCustomerTabs))
❖     CLEAR LIST(hlCustomerTabs)
❖   End if
    `Konec metody
```

5. Otevřete formulář [Zákazníci]Vstupní.
6. Přejděte na Stránku pozadí formuláře (Stránka 0).
7. Poklepejte na objekt Řízení karta `hlCustomerTabs`. (Poznámka: začíná HL jako hierarchical list)
8. Přejděte na stránku Řízení dat.
9. Vymažte všechny výchozí hodnoty pod tlačítkem Upravit řetězec.



10. Přejděte do Prostředí uživatele a otestujte změny.



5.6. Předvybraná záložka ovládání karty

Po krátkém experimentování s Řízením karta objevíte, že při přechodech ze stránky na stránku pomocí příkazů je důležité nastavovat v Ovládání karta správnou záložku podle zobrazované stránky formuláře. Zajistit správné vybrání záložky lze v metodě formuláře předtím než se záznam objeví na obrazovce.

5.6.1. SELECT LIST ITEM (list; PoložkaPozice)

Parametr	Typ	Popis
list	ListRef	Číslo odkazu na seznam
PoložkaPozice	Číslo	Pozice položky v rozšířeném seznamu

Příkaz SELECT LIST ITEM vybere položku jejíž pozici předáme v parametru PoložkaPozice v seznamu jehož číslo jsme předali.

Parametr PoložkaPozice je pozice vztažená k současnému stavu rozšíření/zůžení seznamu a jeho podseznamu. Toto číslo je hodnota mezi 1 a hodnotou navrácenou příkazem Count list items jež vyjadřuje počet viditelných položek seznamu. Jestliže předáte hodnotu vně tohoto rozsahu, bude vybrána první položka.

5.6.2. Current form page -> Longint

Funkce Current form page navrátí číslo stránky formuláře, která je právě zobrazována.

5.6.3. Udržování synchronizace objektu Řízení karta.

1. Upravte metodu formuláře [Zákazníci]Vstupní následujícím způsobem:

```
If (False)
      ` Metoda formuláře.: Input
      ` Kurz programovani ACI University
      ` Datum: 1/15/97
      ` By: Jim Steinman

      ` `Purpose: Causes the [Faktury] to be loaded

   <>f_Version6x10:=True
❖  <>f_Version6x30:=True
   <>fJ_Steinman:=True

❖     ` Upraveno:  3/17/97
❖     ` Ovládací karta synchronizovaná s aktuální zobrazenou stranou
❖  <>fK_Wilbur:=True

End if

$FormEvent:=Form event

Case of
  ś ($FormEvent = On Load)
    If (pTable = (->[Faktury])) ` Přichází z tabulky faktury z tlačítka Upravit zákazníka
      SET VISIBLE(*; INVOICEButton@; False)    ` Nelze provádět funkce na faktury
      SET WINDOW TITLE("Modifying "+[Zákazníci]Firma)
    Else
      WIN_InputWindowTitle

    End if

      If (Record number([Zákazníci]) = -3)
        [Zákazníci]FirmaID:=String( LNextSequence ("CustomerID"))
        GEN_TimeStamp (->[Zákazníci]DatumVytvoření;->[Zákazníci]ČasVytvoření)
      End if

❖   SELECT LIST ITEM (hlCustomerTabs;Current form page)

    ckCapatilizeOff :=0

End case
    `Konec metody
```

2. Přepněte se do prostředí Vlastní nabídky a otestujte zda je ovládání karta synchronizováno s tlačítky přepínání stránek.

5.7. Použití hierarchických seznamů v řízení karty pro vztažené tabulky.

Hierarchické seznamy mají další rys, který musíte pečlivě sledovat. Nemůžete zobrazit tentýž hierarchický seznam na více než jednom formuláři současně. Hierarchické seznamy mají vestavěný rys, který pomáhá zobrazit seznam ve správném rozšířeném stavu. Jestliže zobrazíte přesně tentýž hierarchický seznam na dvou stejných formulářích a uživatel změní vzhled jednoho ze seznamů, druhé zobrazení seznamu způsobí, že 4D zkrachuje. V prostředí více procesů na jednom stroji je proto nezbytné zobrazit seznam na každém formuláři jako oddělený. Jestliže přecházíme na formulář ze vztažené tabulky tj. z faktury chceme upravit vztažený záznam zákazníka, potřebujeme odlišnou verzi hierarchického seznamu pro proces faktury. Pravděpodobnost, že budeme tento seznam potřebovat je malá takže jej můžeme vytvořit a okamžitě vymazat po konci používání formuláře zákazníků ze vstupního formuláře faktury.

5.7.1. Použití hierarchických seznamů odlišných podle použitého formuláře

1. Otevřete formulář [Faktury];"Vstupní".
2. Nahraďte metodu objektu bModify importem z textového souboru nebo napište následující:

```
If (False)
    ` Metoda objektu: bModify [Faktury]; "Vstupní"
    ` Kurz programovani ACI University
    ` Created By: Jim Steinman
    ` Datum: 1/15/97

    ` Účel: Dovolí upravit záznam [Zákazníci]

  <>f_Version6x00 :=True
  <>f_Version6x10 :=True
  <>f_Version6x20 :=True
  <>f_Version6x30 :=True
  <>fJ_Steinman :=True

    ` Upraveno: 1/16/97
  <>fK_Wilbur :=True
    ` Upraveno: 1/17/97
  <>fK_Wilbur :=True
    ` Upraveno: 3/17/97
  <>fK_Wilbur :=True

End if

C_LONGINT ($LTableState)
$LTableState:=LOCK_LSet2ReadWrite (->[Zákazníci]; True)

If ($LTableState # -1)
```

```
    CUSTOMER_Initialize                    ` Vytvoří hierarchický seznam
    MODIFY RECORD ([Zákazníci];*)
    LOCK_Set2ReadOnly (->[Zákazníci]; $LTableState)

    If (Is a list(hlCustomerTabs))      ` Vymaže seznam
      CLEAR LIST(hlCustomerTabs)
    End if
  End if

  WIN_InputWindowTitle

    ` Konec metody
```

3. Přepněte se do prostředí Vlastní nabídky a otestujte, že jsou ve formuláři dostupném z faktury obrázky.

5.8. Použití hierarchických seznamů vytvořených v Editoru seznamů

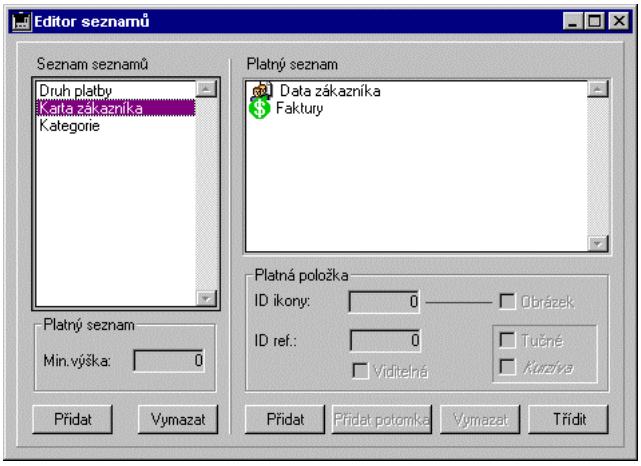
Nyní, když jsme se seznámili s náročnějším způsobem vytváření hierarchických seznamů, řekněme si něco i o jednodušším způsobu vytvoření.

5.8.1. Load list (NázevSeznamu) -> ListRef

Do příkazu Load list předáváme název seznamu vytvořeného pomocí Editoru seznamů v Prostředí návrháře a funkce navrátí číslo odkazu na seznam podobně jako příkaz New list. Tento seznam bude mít všechny vlastnosti nastavené v Editoru seznamů takže tím vyloučíme téměř všechno programování, které jsme prováděli výše.

5.8.2. Úprava kódu k použití v seznamu vytvořeného editorem

1. V Prostředí návrháře otevřete Editor seznamů a vytvořte nový seznam nazvaný CustomerTabs.
2. Vytvořte položky seznamu Data zákazníka a Faktury.
3. Položcce Data zákazníka přiřaďte ID ikony 15001.
4. Do čísla ID ref. přiřaďte číslo 1.



5. Do položky Faktury přiřaďte do ID ikony 15002.
6. Do čísla ID ref. Přiřaďte 2.
7. Upravte metodu CUSTOMER_Initialize následujícím způsobem:

```
If (False)
    ` Metoda: CUSTOMER_Initialize
    ` Kurz programovani ACI University
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Inicializuje položky potřebné pro proces zákazníka

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

If (False)
    ` Toto je způsob jak vytvořit ovládání programem.
  hlCustomerTabs:=New list          ` Je to nový proces pak vytvoř seznam.
  APPEND TO LIST(hlCustomerTabs;"Data zákazníka";1)
  SET LIST ITEM PROPERTIES(hlCustomerTabs;0;True;Plain;15001)
  APPEND TO LIST(hlCustomerTabs;"Faktury";2)
  SET LIST ITEM PROPERTIES(hlCustomerTabs;0;True;Plain;15002)
End if

    ` Toto je způsob využití Editoru seznamů.
  hlCustomerTabs:=Load list ("Karta zákazníka")      ` Je to nový proces pak vytvoř
seznam

    ` Konec metody
```

5.9. Znepřístupnění záložky používající Hierarchický seznam

Nemá smysl přecházet na druhou stránku vstupního formuláře zákazníků, jestliže přicházíme ze vstupního formuláře faktury. Druhá stránka formuláře zákazníků v tomto případě zobrazuje prázdný podformulář, který nemůže zobrazit faktury zákazníka.

Řešení: Je jím znepřístupnění záložky tak, aby uživatel nemohl na tuto záložku klepnout. Vlastnost dostupu je vše co potřebujeme nastavit v hierarchickém seznamu. Jestliže ji nastavíme na True, na záložku půjde klepnout a jestliže na False záložka bude znepřístupněna.

5.9.1. Znepřístupnění záložky

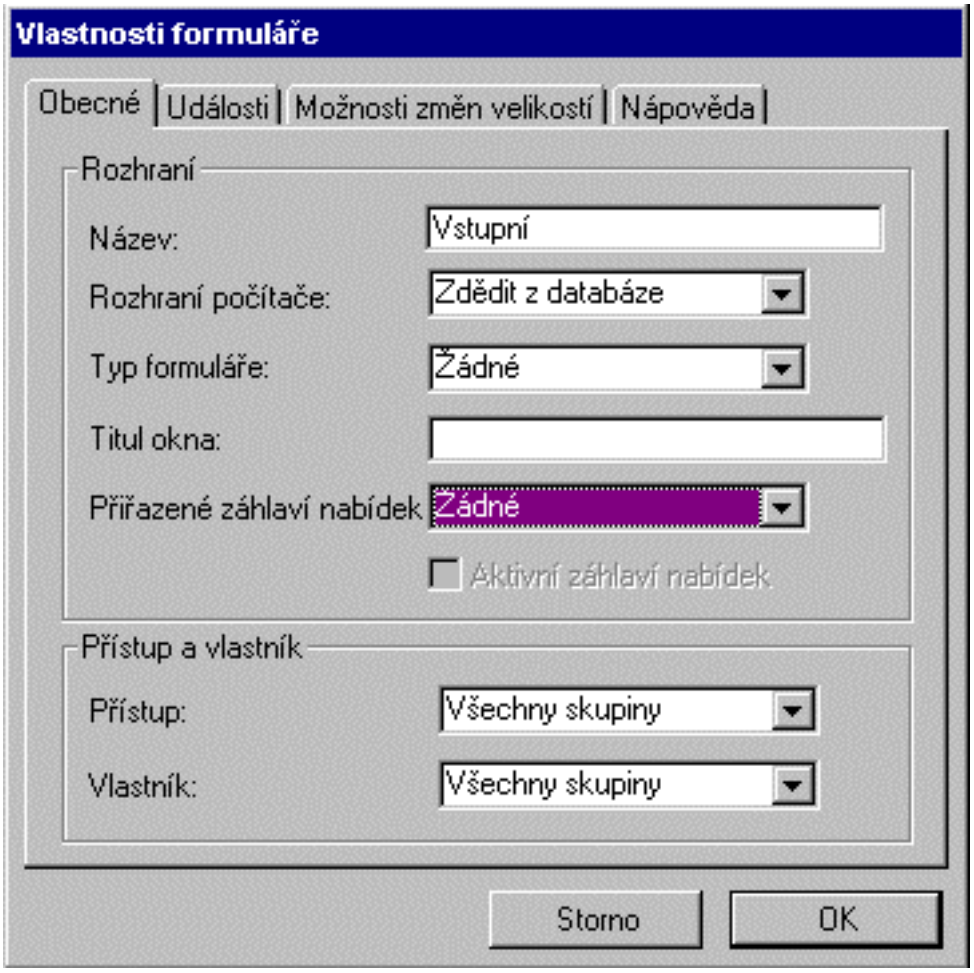
1. Upravte metodu formuláře [Zákazníci]Vstupní následovně:

```
If (False)
      ` Metoda formuláře.: Vstupní
      ` Kurz programovani ACI University
      ` Datum: 1/15/97
      ` Autor: Jim Steinman

      `Účel: Umožnit načíst  [Faktury]

   <>f_Version6x10:=True
   <>f_Version6x30:=True
   <>fJ_Steinman:=True

      ` Upraveno:  3/17/97
      ` karty zobrazit synchonizovaně podle stránek formuláře
   <>fK_Wilbur:=True

End if

$FormEvent:=Form event

Case of
   ś ($FormEvent = On Load)
      If (pTable = (->[Faktury]))       ` Přichází z tlačítka Upravit zákazníka
         SET VISIBLE(*; INVOICEButton@; False)    ` Nemůže dělat funkce na faktuře
         SET WINDOW TITLE("Upravování "+[Zákazníci]Firma)
❖        SET LIST ITEM PROPERTIES(hlCustomerTabs;2;False;Plain;15002)
      Else
         WIN_InputWindowTitle

      End if
```

6. Vlastnosti formuláře

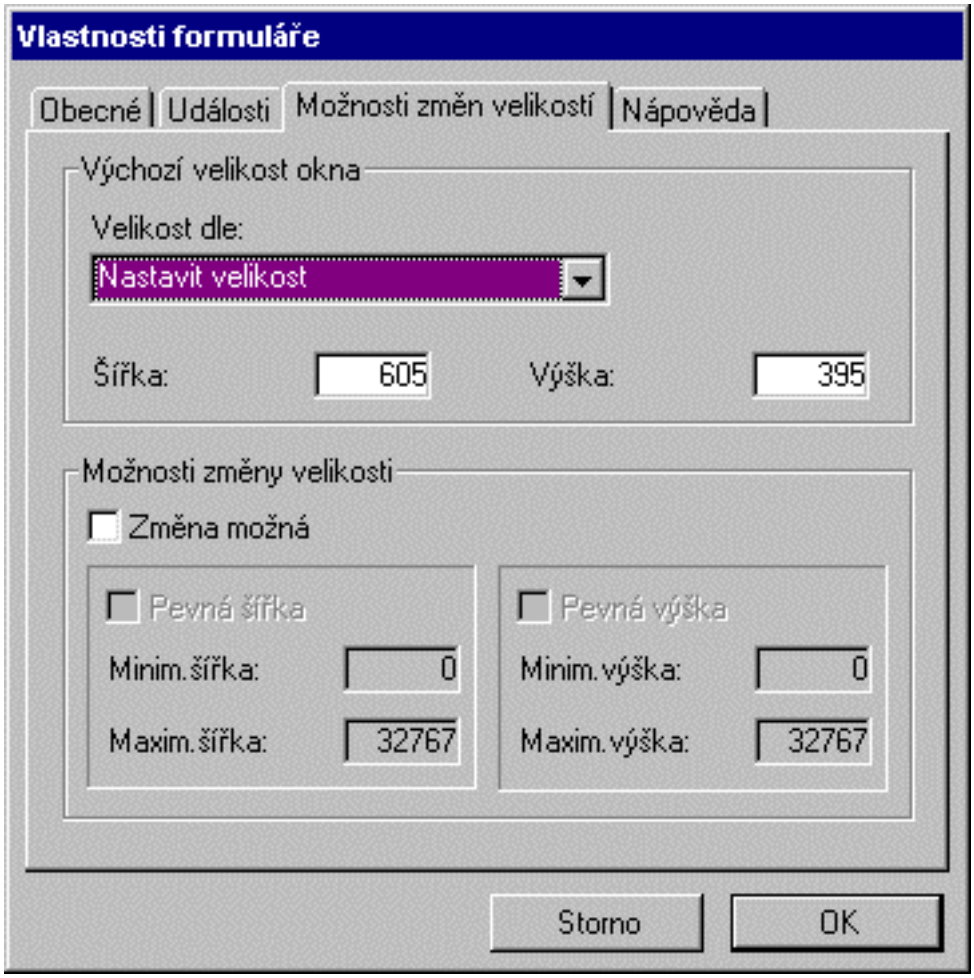
Okno vlastností formuláře vám dovolí nastavit různé aspekty použití formuláře.

6.1. Obecné



Z tohoto okna můžete pojmenovat formulář, přiřadit vzhled, zadat ve formuláři výchozí titulek okna, přiřadit záhlaví nabídky a určit skupiny pro přístup k tomuto formuláři.

To co nás nyní zajímá jsou možnosti změn velikostí zobrazení a změn velikostí uživatelem.



6.2. Možnosti změn velikostí

Při zobrazování vstupního formuláře ve vlastní aplikaci, budete obvykle otevírat formulář s pomocí funkce Open window. Open window vám dovolí určit koordináty rohů okna a rovněž typ okna. Jestliže nepoužijete žádné možnosti změny velikostí, bude schopnost uživatele měnit velikost okna závislá na typu zvoleného okna. Typ okna je určen jako parametr v příkaze Open window. Možnosti určení velikosti a změn velikostí dostupné v okně Vlastnosti formuláře, vám dají větší možnost řízení velikosti okna a jeho možností změn při zobrazení.

6.3. Výchozí velikost okna

Výchozí velikost okna vám umožní řídit počáteční velikost okna. Váš výběr je možný z následujících variant:

- Automaticky velikost: velikost je vypočítána 4th Dimension v závislosti na velikosti všech objektů zobrazovaných na formuláři.

- Nastavit velikost: velikost je závislá na dalších možnostech, které zadáte buď zadáte čísla pevné výšky a šířky nebo zadáte výchozí velikost a možnosti změny formulář od do v parametrech výšky i šířky.

- Velikost závislá na vybraném objektu: v tomto případě použije 4th Dimension nejmenší velikost nezbytnou k zobrazení vybraného objektu. Např. jestliže vyberete objekt, který je umístěn ve spodním pravém rohu zobrazované oblasti, 4th Dimension otevře okno veliké pouze tak, aby byl tento objekt do okna zahrnut. Tuto možnost vybírejte tehdy jestliže chcete umístit některé aktivní objekty do oblasti mimo obrazovku (tzn. vně otevřeného okna). Tyto objekty pak nebudou mít vliv na velikost otevřeného okna.

Když vyberete Automatickou velikost nebo velikost založenou na vybraném objektu můžete určit horizontální a vertikální šířku změny. V tomto případě můžete určit šířku (v bodech), která definuje oblast hranice tak, že okraje objektu na pravém spodním rohu formuláře nejsou uvolněny proti okraji okna.

6.4. Funkce Open window

Funkce Open window navrací nyní číslo odkazu na okno. To co je při této syntaxi důležité je, že nyní můžete použít výchozí velikost okna k otevření okna, které používá velikost vašeho formuláře k určení své velikosti.

Prvním krokem je použití příkazu INPUT FORM s volitelným parametrem hvězdička. Takto použitý příkaz instruuje další volání funkce Open window, aby použila výchozí velikost určeného formuláře jestliže budou pro volání použity rovněž správné parametry, které instruují 4D, aby použila velikost formuláře.

Příklad:

```
INPUT FORM([myTable];"myForm";*)
$LWindowID:=Open window(50;50;-1;-1;0) ` Je potřeba pouze určit vrchní levý roh.
DIALOG([myTable];"myForm")
CLOSE WINDOW
```

Poznámka: Nezapomeňte znovu přiřadit pomocí příkazu INPUT FORM správný vstupní formulář, jestliže pro velikost okna používáte nějaký jiný formulář než normální vstupní formulář.

6.4.1. Vyzkoušení použití výchozích velikostí dle formuláře

1. Vyzkoušejte metodu projektu GEN_ModifySelection.

2. Vyzkoušejte metodu projektu WIN_LNewWindow.
3. Přejděte do Prostředí uživatele a vyzkoušejte jejich chování.

6.5. Formuláře s možnou změnou velikosti

Možnost změn velikostí ve Vlastnostech formuláře vám dovolí použít okna, u kterých lze měnit velikost akcí uživatele a dovolí vám nastavit minimální a maximální dovolené velikosti oken. Nastavení minimální velikosti je způsob jak zabránit uživateli ve změně velikosti formuláře tak, že nebude vidět panel tlačítek a další důležité objekty.

6.6. Objekty a formuláře se změnou velikosti

Práce s formuláři umožňujícími změnu velikostí se podobá výstupu na ledovou horu. Musíte rozhodnout co se bude dít s každým jednotlivým objektem na formuláři jestliže bude formulář měnit velikost.

Zde je několik věcí, které je potřeba si pamatovat:

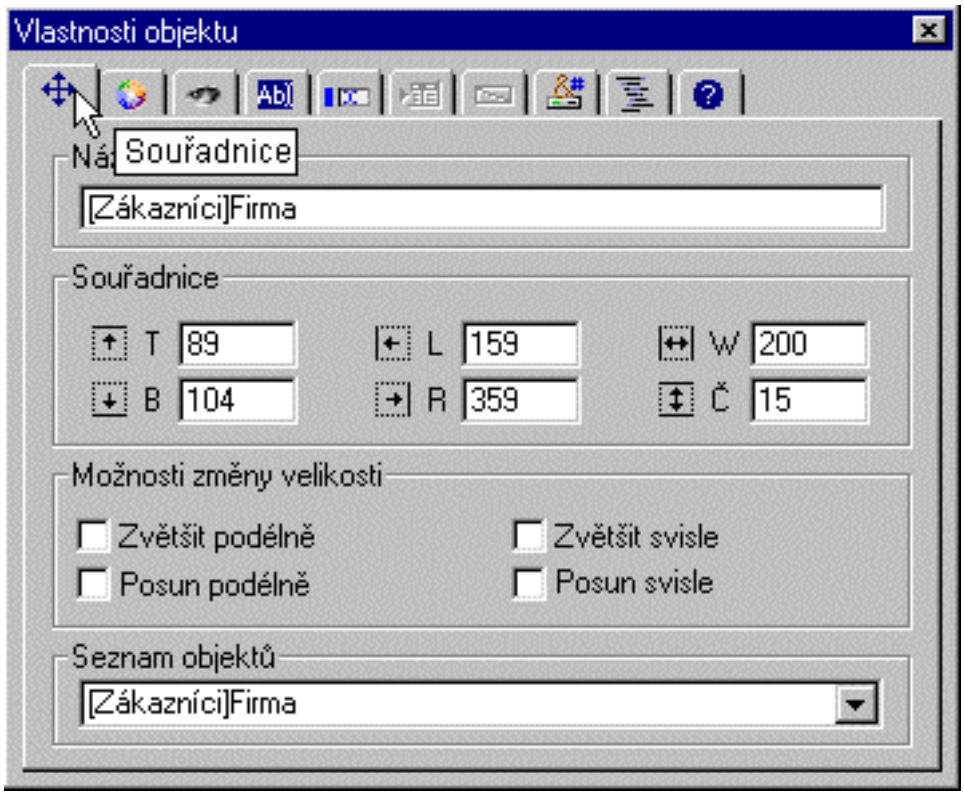
Vstupní formuláře:

- Rozhodněte se, které objekty chcete aby měnily velikost. A to jak horizontálně tak vertikálně.
- Ujistěte se, že všechny objekty vpravo od objektu s možností horizontální změny či horizontálního pohybu se horizontálně přemisťují.
- Ujistěte se, že všechny objekty pod objektem s možnou vertikální změnou či posunem se pohybují rovněž vertikálně.
- Obrázková tlačítka na spodu formuláře se musí pohybovat podobně.
- Nezapomeňte na všechny objekty pozadí, které jste vytvořili.
- Ujistěte se, že všechny objekty, které chcete aby zůstaly skryty mimo obrazovku se rovněž pohybují tak, aby zůstaly skryty mimo obrazovku.
- Testujte formulář, testujte formulář, testujte formulář. Ujistěte se, že jste dobře otestovali minimální velikost.

Výstupní formuláře:

- Horizontální změny velikostí jsou jediné změny velikostí, které je zde potřeba brát v úvahu, protože vertikální změna velikosti změní pouze velikost oblasti a umožní zobrazit více záznamů.
- Po určení minimální a maximální velikosti vašeho vstupního formuláře proveďte totéž u výstupního formuláře.

- Položky v zápatí by se neměli pohybovat a mít velikost.
- Položky pod zápatím by se neměli pohybovat a mít velikost.



6.6.1. Provedení změny formulářů se změnou velikosti

1. Upravte metodu projektu GEN_ModifySelection v místě, kde volá WIN_LnewWindow následovně:

   $LWindowID:=WIN_LNewWindow (-1;-1;<>Cascading window;Plain no zoom box window)

2. Upravte všechny vstupní a výstupní formuláře tak, aby mohly měnit velikost a stejně tak všechny objekty na těchto formulářích.

Kvíz — Okna se změnou velikostí

- Jakou velikost navrhujete pro okna zákazníků?
- Jakou velikost navrhujete pro okna faktur?

- Jakou velikost navrhujete pro okna produktů?

7. Události formuláře

4D v6 používá pro ovládání formulářů a jejich objektů tzv. události formuláře. Jsou to logické události při zpracování formuláře či objektu a nebo zachycují aktivity uživatele.

7.1. Popis událostí formuláře

4th Dimension poskytuej následující předdefinované konstanty:

Konstanta	Hodnota	Popis
On Load	1	Formulář se chystá zobrazit nebo tisknout
On Unload	24	Formulář bude opuštěn a uvolněn
On Validate	3	Vstup do záznamu je ověřován
On Clicked	4	Nastalo klepnutí na objekt
On Double Clicked	13	Nastalo poklepání na objekt
On Keystroke	7	Do objektu, který má fokus je vkládán znak
On Getting Focus	15	Objekt formuláře získává fokus
On Losing Focus	14	Objekt formuláře ztrácí fokus
On Activate	11	Okno formuláře se stává oknem na popředí
On Deactivate	12	Okno formuláře přestává být oknem na popředí
On Outside Call	10	Formulář obdržel volání příkazem CALL PROCESS
On Drop	16	Data jsou puštěna do objektu
On Drag Over	2	Data jsou přetahována přes objekt a mohou být puštěna do objektu
On Menu Selected	18	Položka nabídky byla vybrána
On Data Change	20	Data objektu byla změněna
On Plug-in Area	19	Externí oblast požaduje, aby byla spuštěna jeho metoda objektu
On Printing Header	5	Oblast záhlaví formuláře bude tištěna
On Printing Details	23	Oblast obsahu formuláře bude tištěna
On Printing Break	6	Jedna z oblastí zlomu formuláře bude tištěna
On Printing Footer	7	Oblast zápatí formuláře bude tištěna
On Close Box	22	Uzavírací okénko okna bylo klepnuto
On Display Detail	8	Záznam se chystá zobrazit v seznamu
On Open Detail	25	Záznam byl poklepán a chystá se zobrazit ve vstupním formuláři

On Close Detail	26	Je opouštěn vstupní formulář a přechází se zpět do formuláře seznamu

7.1.1. Obecné události

Při zavedení

Událost Při zavedení (On Load) je vždy první události formuláře a objektu. Předtím než je formulář zobrazen na obrazovku nebo tištěn. Obvykle při této události inicializujete objekty (proměnné a pole) a nebo svázaná data.

Při vyvedení

Události Při vyvedení (On Unload) je vždy poslední událostí, které formulář a objekty obdrží těsně předtím než je formulář používaný pro vstup dat opuštěn a vyveden z paměti. Tato událost se děje pouze ve formulářích používaných pro vstup dat tj. pořizování a úpravy záznamů či dialogy. Tato událost se neděje pro formuláře používané pro zobrazení seznamu záznamů, pro podformuláře či pro tisk. Obvykle v této události budete mazat objekty, proměnné a pole a nebo svázaná data.

Při potvrzení

Událost Při potvrzení (On Validate) se děje, když vstup dat pro záznam (přidání nového záznamu) nebo úprava existujícícho záznamu je ověřována. Obvykle během této události budete provádět akce, které je potřeba provádět pouze tehdy, když je vstup dat přijmut.

Při zobrazení obsahu

Události Při zobrazení obsahu (On Display Detail) se děje pokaždé, když se záznam chystá zobrazit ve výstupním formuláři seznamu záznamů.

7.1.2. Speciální události

Při aktivaci

Události Při aktivaci (On Activate) se děje, když je okno poprvé vytvářeno a stává se aktivním oknem a rovněž pokaždé, když je okno přenášeno na popředí zpoza ostatních oken nebo bylo předtím uschováno.

Při deaktivaci

Události Při deaktivaci (On Deactivate) se děje vždy, když je aktivní okno posíláno na pozadí a jiné okno se stává oknem na popředí. Při dvou střídajících se oknech na popředí se událost Při deaktivaci děje před událostí Při aktivaci v druhém okně. Kromě toho okno, které je uzavíráno neprovádí akci Při deaktivaci, to je odlišné od události Při aktivaci, která nastane při zobrazování okna.

Při vnějším volání

Když proces (formulář procesu) obdrží volání z jiného procesu příkazem OUTSIDE CALL, stane se ve formuláři volaného procesu událost Při vnějším volání (On Outside Call). Jestliže je proces zaneprázdněn, odložen nebo zpožděn událost nenastane.

Při zavření okna

Událost Při zavření okna (On Close Box) nastane, když bylo klepnuto na uzavírací políčko okna.

Při volbě z nabídky

Událost Při volbě z nabídky (On Menu Selected) nastane, když uživatel vybere položku nabídky.

Při otevření obsahu

Událost Při otevření obsahu (On Open Detail) nastane při přechodu z výstupního formuláře seznamu záznamů do vstupního formuláře záznamu.

Při uzavření obsahu

Událost Při uzavření obsahu (On Close Detail) nastane těsně předtím než se znovu objeví na obrazovce výstupní formulář seznamu záznamů při návratu ze vstupního formuláře záznamu.

Při časování

Tato událost nastane vždy po stanovené době, kterou ovlivníme příkazem ON TIMER.

7.1.3. Události svázané s akcí uživatele

Při klepnutí

Uživatel klepl na objekt, který umožňuje klepnutí (viz klepnutelné objekty níže).

Při poklepnutí

Uživatel poklepal na objekt, který umožňuje klepnutí (viz klepnutelné objekty níže).

Před úhozem na klávesu

Uživatel uhodil na klávesu předtím než 4D zobrazí výsledek tohoto úhozu v dostupném objektu (viz dostupné objekty níže).

Po úhozu na klávesu

Uživatel uhodil na klávesu potom, kdy 4D zobrazila výsledek tohoto úhozu v dostupném objektu (viz dostupné objekty níže).

Při získání fokusu

Uživatel vstoupil do dostupného objektu buď pomocí klepnutí tabelátorem nebo pomocí kódu (viz dostupné objekty níže).

Při ztrátě fokusu

Uživatel opouští dostupný objekt buď klepnutím nebo tabelátorem (viz dostupné objekty níže).

Při aktualizaci

Uživatel upravil data v dostupném objektu a opouští tento objekt pomocí klepnutí nebo tabelátorem (viz dostupné objekty níže).

V oblasti Plug-in

Uživatel je v oblasti zásuvného modulu (plug-in) a dokončil něco co způsobí, že tato oblast předá řízení do 4D.

7.1.4. Události potáhnout a pustit

Při vsazení

Uživatel pustil něco do objektu, který je vsaditelný.

Při odtažení

Uživatel přesunuje něco přes objekt, který je vsaditelný.

7.1.5. Události tisku

V záhlaví

Při tisku obsahu

Při tisku zlomu

Při tisku zápatí

7.2. Form event -> Událost číslo

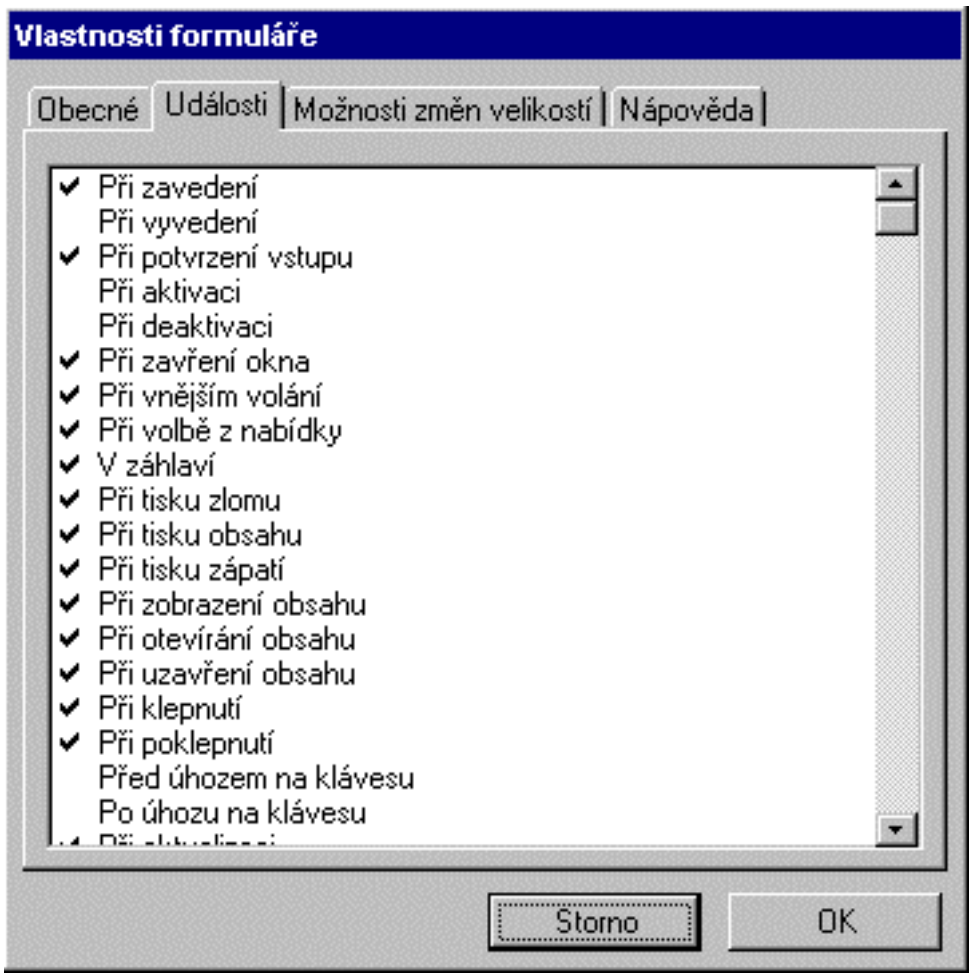
Funkce Form event navrátí číselnou hodnotu, která určuje typ události formuláře, která právě nastala. Obvykle budete používat Form event z metody formuláře nebo objektu a nebo metody projektu volané v metodě formuláře či objektu.

7.3. Zapnutí a vypnutí událostí formuláře

Události formuláře lze zapnout nebo vypnout. Zapnout či vypnout lze rovněž provádění událostí na úrovni objektů. Pokud jsou události vypnuty metoda, která je zaměřena na tuto událost se neprovede, proto je potřeba kontrolovat na úrovni vlastností formuláře i objektů zda jste dané události aktivovali. Pokud je vypnuta kontrola událostí na úrovni formuláře a objektů, provádění kódu je rychlejší.

7.3.1. Vlastnosti událostí na úrovni metod formulářů

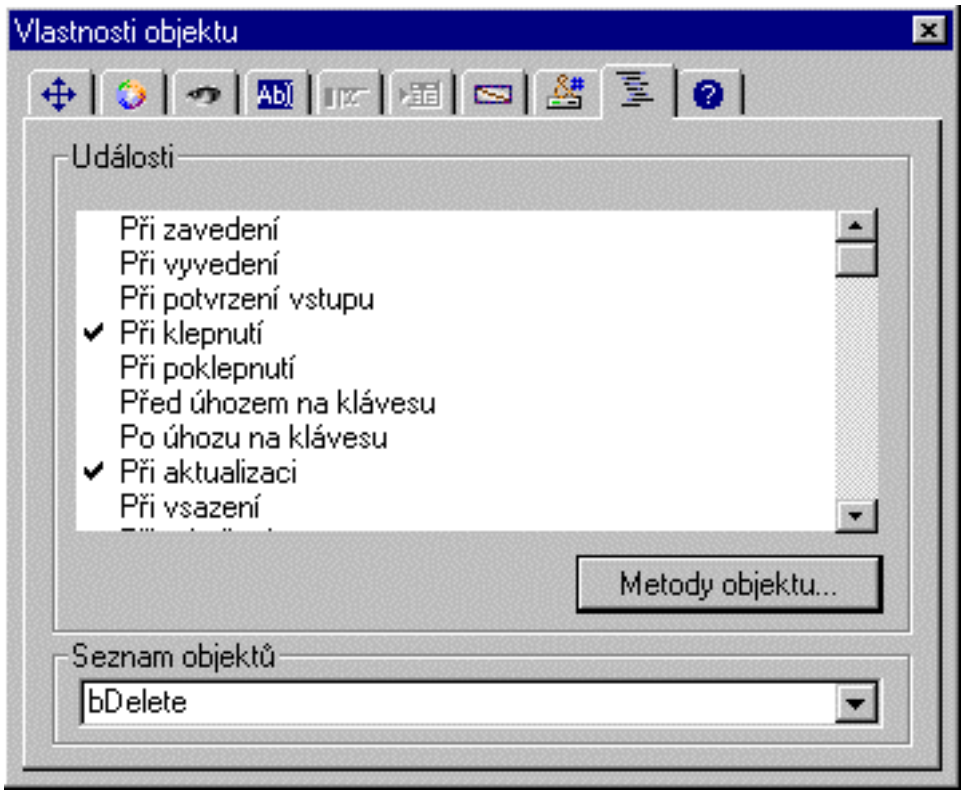
1. Otevřete formulář v Prostředí návrháře.
2. Otevřete okno Vlastnosti formuláře.
3. Přejděte na stránku Události.



Toto je místo, kde můžete ovlivňovat aktivaci a deaktivaci událostí pro určitý formulář. Pro urychlení provádění vaší databáze, vypněte všechny události, které nemáte ošetřeny v kódu metod formuláře a objektů a které na dané události nebudou tedy odpovídat.

7.3.2. Vlastnosti událostí na úrovni metod objektů

1. Otevřete okno definic objektů pro libovolný objekt.
2. Přejděte na stránku Události.



Toto je místo, kde budete ovlivňovat aktivaci a deaktivaci událostí pro určitý objekt. K urychlení provádění vaší databáze, vypněte všechny události, které nemáte v kódu metody objektu a pro které tedy metody objektů nebudou odpovídat.

Poznámka: Při psaní kódu a testování. Jestliže váš kód neprovádí co jste očekávali, zkontrolujte nejdříve zda událost na kterou chcete odpovídat je zapnuta v objektu či formuláři, kde kód existuje.

7.4. Dědění událostí

Události nepracují přesně tak jak by jste si mohli myslet tj. že událost je děděna do nižší úrovně pouze tehdy jestliže je událost zapnuta pro vyšší úroveň. Pro účely této diskuse vezmeme dva základní typy událostí: události zaměřené na formuláře a události zaměřené na objekty. Jestliže je událost zaměřena na formulář (např. Při aktivaci) neobdrží žádný

objekt tohoto formuláře tuto událost i když je zapnuta rovněž událost na úrovni formuláře. Jestliže je událost zaměřena na objekt (např. Při klepnutí) neobdrží metoda formuláře tuto událost ani když je zapnuta tato událost pro metody objektu.

7.5. Události a metody

Když nastane událost formuláře 4th Dimension provede následující akce:

- Nejdříve prohledá všechny objekty formuláře a zavolá všechny metody objektu pro objekty, pro které byla odpovídající událost zaškrtnuta a které jsou tak do události zahrnuty.

- Za druhé, zavolá metodu formuláře jestliže byla vybrána tato událost na úrovni formuláře.

Nepředpokládejme, že metody objektu jsou-li nějaké budou volány v určitém pořadí. Jediné co je pravidlem je, že metody objektů jsou vždy volány před metodou formuláře. Jestliže je objekt podformulář jsou nejdříve volány metody objektu pro podformulář a pak metoda objektu podformuláře. Nakonec pokračuje 4D ve volání metod objektu rodičovského formuláře. Jinými slovy, když je objekt podformulář 4D používá totéž pravidlo pro objekty a metody formuláře jako v rodičovském formuláři.

Jestliže není pro určitou událost zaškrtnuta tato událost na úrovni formuláře neznamená to, že nebudou volány metody objektů pro ty objekty, kde byla tato událost vybrána. Jinými slovy zaškrtnutí či odškrtnutí události na úrovni formuláře nezabrání provedení události objektu byla-li tato na úrovni objektu zaškrtnuta.

Počet objektů zahrnutých do události závisí na druhu a zaměření události. Např.:

- Pro událost Při zavedení budou aktivovány všechny objekty formuláře (z libovolné stránky) u nichž byla zaškrtnuta ve vlastnostech objektu událost Při zavedení a mají metody objektu. Pak pokud byla zaškrtnuta ve vlastnostech formuláře událost Při zavedení, bude provedena metoda formuláře.

- Pro událost Při aktivaci nebude provedena žádná metoda objektu, protože je tato událost zaměřena na formulář jako celek a ne na jednotlivé objekty. Takže jestliže byla tato událost Při aktivaci vybrána na úrovni formuláře bude provedena pouze metoda formuláře.

- Pro událost Při odtažení. bude aktivována pouze metoda objektu, který má vlastnost vsaditelný a má zaškrtnutu událost Při odtažení. Metoda formuláře nebude nikdy volána.

Tabulka uvedená níže shrnuje pro každý typ události jak jsou volány metody objektu a formulářů:

Událost	Metoda objektu	Metoda formuláře	Které objekty
Při zavedení	Ano	Ano	Všechny objekty
Při vyvedení	Ano	Ano	Všechny objekty
Při potvrzení	Ano	Ano	Všechny objekty
Při klepnutí	Ano (klepnutelný) (*)	Ano	Pouze dotčené
Při poklepnutí	Ano (klepnutelný) (*)	Ano	Pouze dotčené
Před úhozem			
Po úhozu	Ano (dostupné z klávesnice) (*)	Ano	Pouze dotčené
Při získání fokusu	Ano (dostupný tab) (*)	Ano	Pouze dotčené
Při ztrátě fokusu	Ano (dostupný tab) (*)	Ano	Pouze dotčené
Při aktivaci	Nikdy	Ano	Žádné
Při deaktivaci	Nikdy	Ano	Žádné
Při vnějším volání	Nikdy	Ano	Žádné
Při vsazení	Ano (pustitelné) (*)	Ano	Pouze dotčené
Při odtažení	Ano (pustitelné) (*)	Never	Pouze dotčené
Při volbě z nabídky	Nikdy	Ano	Žádné
Při aktualizaci	Ano (měnitelný) (*)	Ano	Pouze dotčené
V oblasti plug-in	Ano	Ano	Pouze dotčené
Při tisku záhlaví	Ano	Ano	Všechny objekty
Při tisku obsahu	Ano	Ano	Všechny objekty
Při tisku zlomu	Ano	Ano	Všechny objekty
Při tisku zápatí	Ano	Ano	Všechny objekty
Při zavření okna	Nikdy	Ano	Žádné
Při zobrazení obsahu	Ano	Ano	Všechny objekty
Při otevření obsahu	Nikdy	Ano	Žádné
Při uzavření obsahu	Nikdy	Ano	Žádné

(*) Viz níže část Události, objekty a vlastnosti..

DŮLEŽITÉ: Vždy musíte mít na paměti, že pro libovolnou událost bude metoda objektu či formulář volána pouze tehdy jestliže patřičná událost byla zaškrtnuta ve vlastnostech formuláře či objektu. Výhodou možnosti odškrtnout událost v Prostředí návrháře je, že tím můžete významně snížit počet volání metod a proto významně zvýšit rychlost provádění formulářů. Z tohoto důvodu všechny objekty výše v tabulce znamená pouze ty pro něž je patřičná událost zaškrtnuta.

7.6. Události, objekty a vlastnosti

Metoda objektu je volána jestliže se daná událost skutečně pro daný objekt uděje v závislosti na jejím druhu a vlastnostech objektu. Diskuse uvedená níže podrobněji rozebírá, které události je možno použít pro ovládání různých typů objektů.

7.6.1. Klepnutelné objekty

Klepnutelné objekty jsou objekty, které jsou především ovládány pomocí myši, zahrnují následující objekty:

- Logická dostupná pole či proměnné
- Obrázková pole a proměnné, jejichž formát zobrazení byl nastaven na formát Na pozadí
- Tlačítka, Voliče, Zaškrtávací políčka a Tlačítka na síti
- 3D Tlačítka, 3D Voliče, 3D Zaškrtávací políčka
- Nabídky/seznamy, Hierarchické nabídky, Obrázkové seznamy
- Posuvné oblasti, Hierarchické seznamy
- Neviditelná tlačítka, Zvýrazněná tlačítka, Obrázková tlačítka
- Teploměry, Výseče, Pravítka
- Ovládání karta

Když byla pro některý z výše zmíněných objektů vybrána událost Při klepnutí nebo Při poklepnutí můžete tak ovládat a řídit klepnutí do či poklepání na objekt pomocí příkazu Form event, který vrátí On Clicked nebo On Double Clicked.

Pro všechny tyto objekty se událost Při klepnutí stane při uvolnění tlačítka myši. Existují pouze dvě vyjímky:

- Pro Neviditelná tlačítka nastane událost Při klepnutí jakmile provede klepnutí na objekt a nečeká se na uvolnění tlačítka.

- Pro dělené objekty (Teploměr, Pravítko, Výseč) nastane událost rovněž okamžitě při klepnutí myší do objektu.

Poznámka: Některé z těchto objektů mohou být aktivovány pomocí klávesnice. Např. Zaškrtávací políčko pokud získá fokus může být aktivováno pomocí mezerníku. I v tomto případě je pak aktivována událost Při klepnutí, protože je tím simulováno klepnutí myší.

UPOZORNĚNÍ: Objekt Text se seznamem není považován za klepnutelný objekt. Text se seznamem musí být považován za textovou oblast se vstupem pro kterou přiřazený

seznam poskytuje hodnoty. Jako důsledek musíte tento objekt ovládat pomocí událostí Před úhozem na klávesu , Po úhozu na klávesu a Při aktualizaci.

7.6.2. Objekty dostupné z klávesnice

Objekty dostupné z klávesnice jsou ty objekty do kterých lze vložit data pomocí klávesnice a pro které můžete chtít filtrovat vstup dat na nejnižší úrovni pomocí zachycení události Před úhozem na klávesu a Po úhozu na klávesu. Jsou to následující objekty:

- Všechny dostupné objekty polí (kromě obrázků, podtabulek a BLOB)
- Všechny dostupné proměnné (kromě obrázků, BLOB, ukazatelů a array)
- Texty se seznamem
- Hierarchické seznamy

Při výběru události Před úhozem na klávesu a Po úhozu na klávesu, můžete rozeznávat a ovládat úhoz na klávesu uvnitř objektu a pomocí příkazu Form event, který vrátí On Before Keystroke a On After Keystroke.

7.6.3. Měnitelné objekty

Měnitelné objekty jsou objekty, které mají zdroj dat jejichž hodnota může být měněna pomocí myši nebo klávesnice a které nejsou považovány za prvky řízení rozhraní (ovládané událostí Při klepnutí) Tyto objekty jsou následující:

- Všechny dostupné objekty polí (kromě podtabulek a BLOB)
- Všechny dostupné proměnné (kromě BLOB, Ukazatele a Array)
- Texty se seznamem
- Externí oblasti (pro které je vstup dat zajišťován pomocí 4D Extension)

Tyto objekty jsou ty, které přijímají událost Při aktualizaci. Když je pro tyto objekty vybrána ve vlastnostech událost Při aktualizaci, můžete pro tyto objekty zachytit a ovládat změny hodnoty zdroje dat pomocí příkazu Form event, který navrátí On Data Change .

7.6.4. Objekty dostupné tabelátorem

Objekty dostupné tabelátorem jsou objekty, které získají fokus při přecházení z objektu na objekt pomocí klepnutí na klávesu tabelátor nebo při klepnutí myší do objektu. Objekt získávající fokus je objekt, který přijme znak (znaky) napsané na klávesnici a které

nejsou akcelerátory (Windows) nebo klávesové zkratky (MacOS) na položky nabídky či objekty jako tlačítka.

Všechny objekty mohou být dostupné tabelátorem kromě objektů uvedených níže:

- Nedostupná pole či proměnné
- Tlačítka (pouze na platformě MacOS)
- Síť tlačítek
- 3D Tlačítka, 3D Voliče, 3D Zaškrtávací políčka
- Nabídky (pouze na platformě MacOS), Hierarchické nabídky,
- Obrázkové nabídky
- Posuvné oblasti
- Neviditelná tlačítka, Zvýrazněná tlačítka, Voliče
- Diagramy
- Externí oblasti (pro které není úplnný vstup dat řízen pomocí 4D Extension)
- Ovládání karta

Pokud pro byly pro objekt v jeho vlastnostech vybrány události Při získání fokusu a Při ztrátě fokusu a byla vybrána vlastnost dostupné tabelátorem, pak můžete pro daný objekt zachytit a ovládat změny fokusu pomocí příkazů Form event, který vrátí On Getting Focus nebo On Losing Focus v závislosti na případu.

Once the On Getting Focus and/or On Losing Focus object event properties have been selected for a tabbable object, you can detect and handle the change of focus using the command Form event that will return On Getting Focus or On Losing Focus depending on the case.

8. Ošetřování chyb

Když váš kód narazí na určitý druh problému, který nebyl očekáván objeví se chyba. Váš kód okamžitě zastaví provádění a uživateli je zobrazena zpráva. Chyby mohou být generovány:

- Databázovým strojem 4th Dimension; např. při ukládání záznamu pokud bude snaha uložit zdvojený jedinečný indexovaný klíč.
- Prostředím 4th Dimension; např. když nemáte dostatek paměti pro umístění array.
- Operačním systémem na němž je databáze spuštěna; např. při zaplnění disku nebo chybách vstupu/výstupu.

Není to vždy špatně. Avšak co když chyba byla očekávána, konečným důsledkem bude tak či onak zastavení provádění kódu a uživatel se může ocitnout v situaci téměř tak špatné jak sama chyba.

8.1. ON ERR CALL (MetodaChyby)

Příkaz ON ERR CALL instaluje metodu s názvem MetodaChyby jako metodu pro ovládání chyb v daném procesu. Jestliže metoda chyby je prázdný řetězec vrátí se ošetřování chyb do 4th Dimension. Po úspěšné instalaci metody MetodaChyby 4th Dimension volá vždy tuto metodu, když se vyskytne chyba.

Identifikovat chybu můžete přečtením systémových proměnných chyby. Ošetřování chyby obvykle spočívá v přijetí řešení dle určité chyby a zobrazením zprávy uživateli. Čísla chyb samotné 4th Dimension jsou uvedena v elektronické dokumentaci 4th Dimension.

Příkaz ABORT může být použit k ukončení zpracování. Jestliže nevoláte ABORT v instalované metodě 4th Dimension se navrátí do přerušené metody, která chybu vyvolala a pokračuje v jejím provádění.

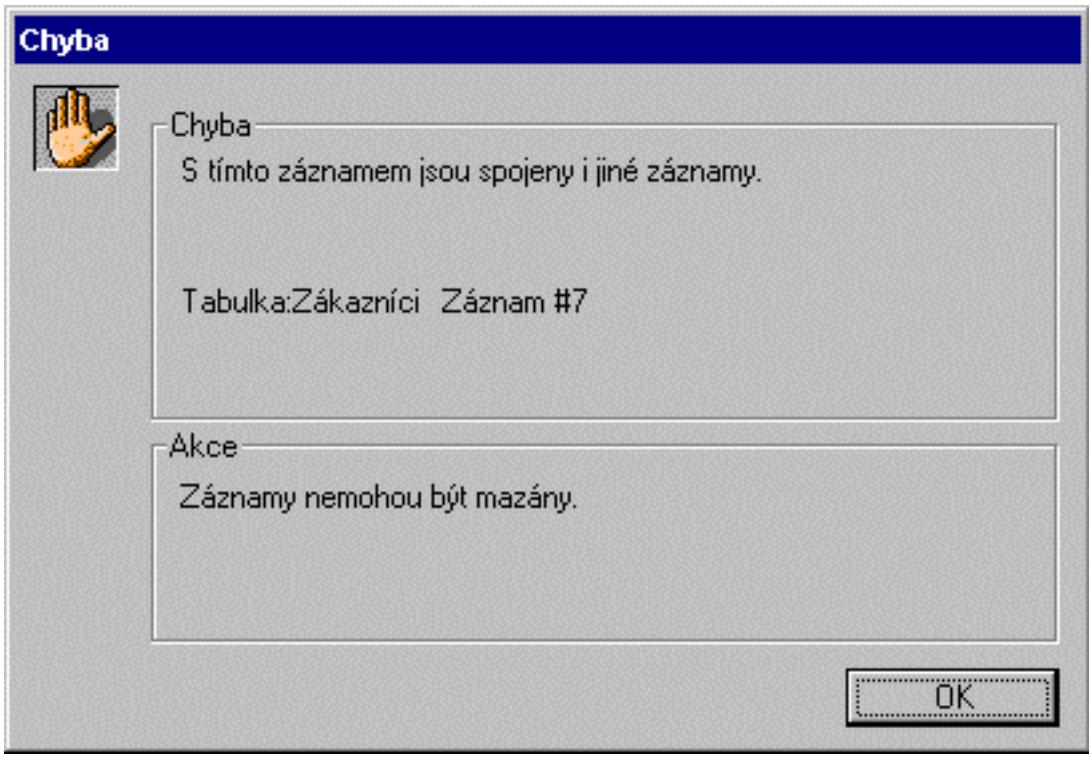
Jestliže se vyskytne chyba v samotné metodě instalované pomocí ON ERR CALL k ošetřování chyby převezme řízení chyb samotná 4th Dimension, proto by jste si měli být jisti, že instalovaná metoda pro ošetřování chyb nemůže generovat chyby. Také samozřejmě nemůžete použít příkaz ON ERR CALL uvnitř metody ošetřující chyby.

Příkaz ON ERR CALL je někdy používán v metodě Při spuštění v některých zákaznických databázích k ošetřování chyb pro tyto aplikace. ON ERR CALL může rovněž být volán na začátku určité metody k ošetření chyb očekávaných v této metodě.

Když je instalována metoda ošetřování chyb pomocí ON ERR CALL není možné krokovat metodu, protože každé klepnutí na klávesnici pomocí modifikátoru generuje chybový kód, který okamžitě aktivuje metodu instalovanou ON ERR CALL.

8.2. Řízení mazání generuje chyby

Jeden z důvodů proč návrháři používají řízení mazání je, aby zabránili vytváření sirotků. Jinými slovy v tomto případě nelze mazat záznam jedinců jestliže existují k tomuto záznamu záznamy v tabulce skupin.



To je dobré místo pro vyzkoušení MetodyChyby.

8.2.1. Vyzkoušení metody chyby na řízení mazání

1. Je-li to nezbytné spusťte databázi ACI Video.

2. Spusťte proces Zákazníci.
3. Vyberte a vymažte tři záznamy.
4. Objeví se chybové hlášení.

Je toto hlášení dostatečně jasné pro běžného uživatele? Co se stane, když uživatel klepne na tlačítko OK?

Zde skutečně potřebujeme metodu, která ošetří tuto chybu a vyřeší vzniklé otázky.

8.2.2. Vytvoření metody řídící mazání záznamů

1. Vytvořte novou metodu projektu ERROR_DeletingRecords následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: ERROR_DeletingRecords
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Ošetřuje chyby vzniklé při mazání záznamů

   <>fGeneric:= True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

C_STRING(255;$sMessage)

$sMessage:=""

Case of
   :(Error = -9987)
      $sMessage:="Jiné záznamy v databázi závisí na informacích ze záznamů" +
                  " které se snažíte vymazat!" + Char(Carriage return)+
                  Char(Carriage return)) + "Ukončuje vymazávání záznamů!"
Else
      $sMessage:="Nastala neočekávaná chyba #  " + String(Error) +
                    Char(Carriage return)) + "Prosím poznamenejte si číslo a
zkonzultujte svého administrátora pro nápravu!“
End case

If( Length ($sMessage) >0)
   ALERT($sMessage)
End if
      ` Konec metody
```

2. Nahraďte metodu projektu M_GEN_DeleteUserSet následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: M_GEN_DeleteUserSet
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Created By: Jim Steinman
      ` Datum: 1/15/97

      ` Účel: Trvale vymaže právě vybrané záznamy

   <>fGeneric:=True
   <>f_Version6x10:=True
   <>f_Version6x20 :=True
      ` Přidán kód k ošetření uzamčených záznamů
   <>f_Version6x30 :=True
      ` Přidán kód k ošetření chyb během mazání záznamů
   <>fJ_Steinman :=True

      ` Upraveno: 1/17/97
   <>fK_Wilbur :=True
      ` Upraveno: 3/17/97
   <>fK_Wilbur :=True

End if

      `Definuje lokální proměnné
C_LONGINT ($OK)                              ` Lokální příznak OK
C_LONGINT ( $LNumberOfRecordsInSet)   ` Počet záznamů v Locked set
C_LONGINT ($LProcessID)                ` ID procesu, který uzamkl záznamy
C_LONGINT ($i)                               ` Čítač záznamů
C_STRING (40; $sUserName)             ` Jméno uživatele, který uzamkl
C_STRING (40; $sMachine)                  ` Název stroje, který uzamkl
C_STRING (40; $sProcessName)    ` Název procesu, který uzamkl

$LNumberOfRecordsInSet:=Records in set ("UserSet")

If ($LNumberOfRecordsInSet > 0)
   CONFIRM ("Vymazat " + String ($LNumberOfRecordsInSet) + " vybraných
záznamů?")

   If (OK = 1)
      ON ERR CALL("ERROR_DeletingRecords")
      CREATE SET (pTable->; "OriginalSelectionSet")          ` Sada k obnově
      USE SET ("UserSet")                                    ` User set použit jako platný výběr

      $OK:=1

      Repeat
```

```
            DELETE SELECTION (pTable->)

            If (Records in set ("LockedSet") > 0)
               $LNumberOfRecordsInSet:=Records in set ("LockedSet")
               CREATE EMPTY SET (pTable->; "NonDeletedSet")
               USE SET ("LockedSet")

               For ($i; 1; $LNumberOfRecordsInSet)
                  LOCKED ATTRIBUTES (pTable->; $LProcessID; $sUserName; $sMachine ;
                        $sProcessName)

                  If ($LProcessID # -1)                  ` Záznam je stále přítomen
                     ADD TO SET (pTable->; "NonDeletedSet")
                  End if

                  NEXT RECORD (pTable->)
               End for

               $LNumberOfRecordsInSet:=Records in set ("NonDeletedSet")

               If ($LNumberOfRecordsInSet > 0)
                  CONFIRM (String ($LNumberOfRecordsInSet) + " záznamů bylo používáno a
není vymazáno!" +
                        Char(Carriage Return) + "Pokračovat v
mazání?";"Vymazat";"Ponechat ")

                  If (OK = 0)
                     ALERT ("Nevymazané záznamy budou ukázány!")
                     $OK:=0
                  End if

                  USE SET ("NonDeletedSet")
               End if
               CLEAR SET ("NonDeletedSet")

            Else

               USE SET ("OriginalSelectionSet")      ` Obnova nevymazaných do platného
výběru
               $OK:=0
            End if                                   ` LockedSet

         Until ($OK = 0)

❖        ON ERR CALL("")
         CLEAR SET ("OriginalSelectionSet")
         WIN_OutputWindowTitle
         End if                                      ` OK = 1

      Else
         ALERT ("Nejdříve musíte záznamy vybrat!")
```

```
End if                                                    ` $LNumberOfRecordsInSet
   ` Konec metody
```

3. Přepněte se do prostředí Vlastní nabídky a testujte vaše nové ovládání chyb.

8.3. Jiné chyby mazání

Existují ještě další chyby, které mohou nastat při pokusu o vymazání záznamů.

Kvíz — Další chyby mazání

- Můžete je vyjmenovat?
- Co se stane s vaším kódem, když tyto chyby nastanou při pokusu o vymazání záznamů?
- Můžeme vylepšit náš kód tak, aby ovládal rovněž některé z těchto chyb?
- Můžeme poskytnout uživateli informaci o záznamu, který způsobil problém?

8.3.1. Vylepšení ošetření chyb při mazání záznamů

1. Napište dodatečná chybová hlášení do vaší části Case v metodě projektu ERROR_DeleteingRecords.
2. Testujte vaše hlášení.

9. Triggery

4th Dimension poskytuje řadu mechanizmů a rysů pro řízení akcí, které mohou být v databázi prováděny např.:

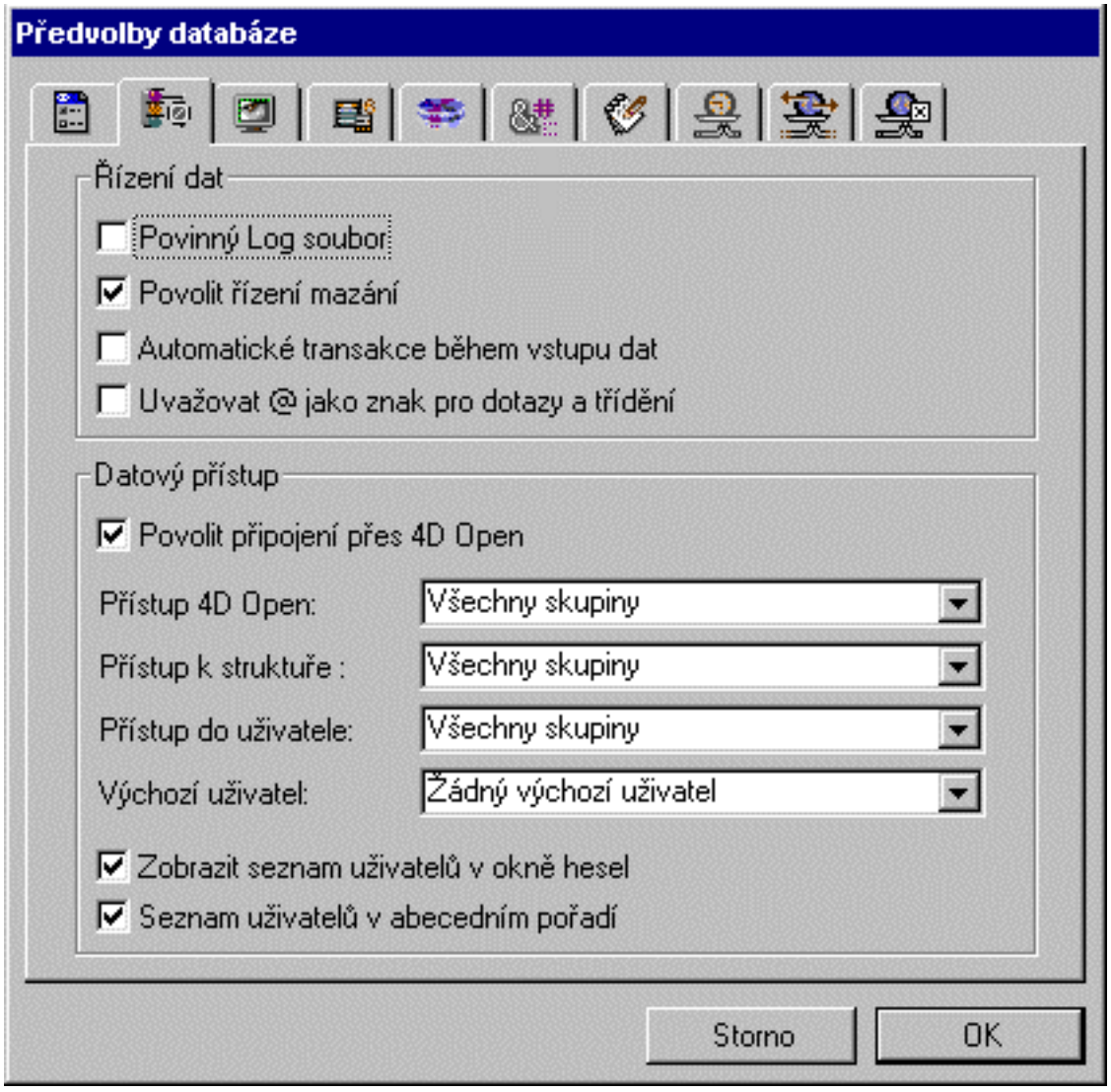
- Na úrovni databázového stroje 4D může být nastavena vlastnost pole **Jedinečné** tak, aby bylo zabráněno vytvoření několika záznamů obsahujících tutéž hodnotu v tomto poli a tabulce. Na stejné úrovni můžete použit vlastnost **Řízení mazání** při mazání záznamů ve vztahů tak, aby bylo zabráněno mazání záznamů vztažených k jiným záznamům.

- Na úrovni uživatelského rozhraní 4D, můžete nastavit vlastnosti objektu tak, že zajistíte normalizaci dat vkládaných konečným uživatelem.

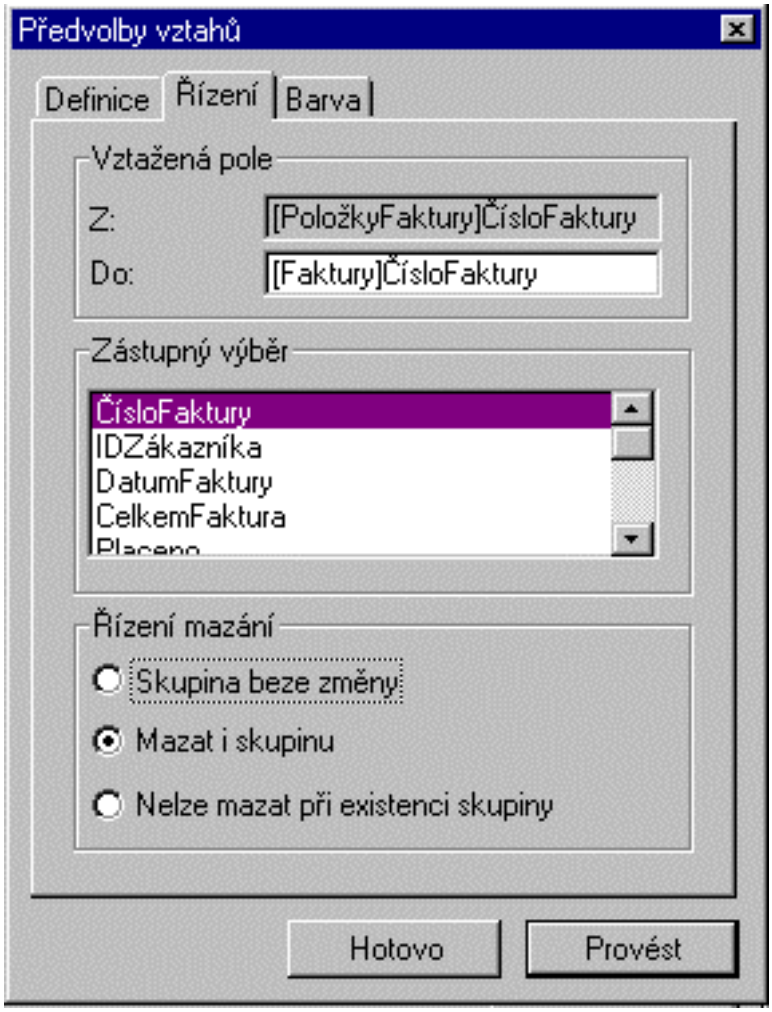
- Na úrovni jazyka 4D vám nyní více než šest set standartních příkazů dovolí zavést vlastní pravidla v databázi v závislosti na skutečných pravidlech ve vaší oblasti působení.

9.1. Řízení mazání na úrovni databázového stroje

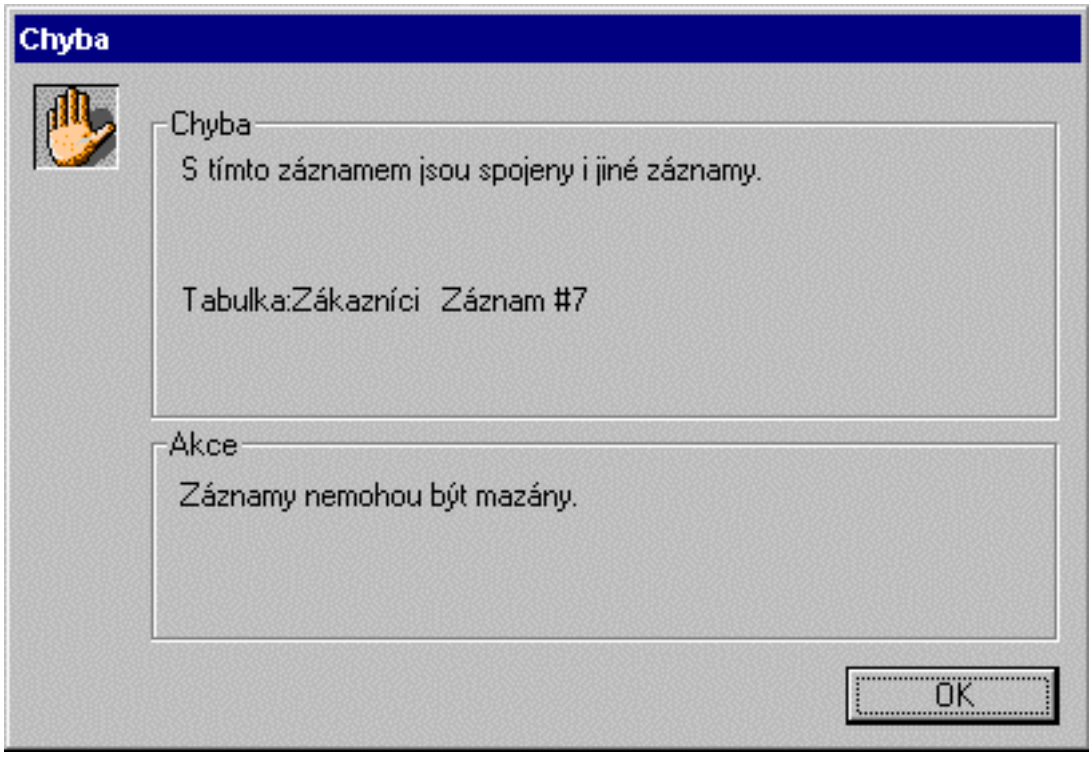
S pomocí předvoleb databáze 4th Dimension je možné zapnout řízení mazání dat pomocí Zaškrtávacího políčka povolit řízení mazání:



Pokud zvolíte tuto možnost je možné pomocí vlastnosti **Povolit řízení mazání** pro určitý vztah stanovit způsob jak databázový stroj bude rozhodovat při pokusu o vymazání záznamu, který je ve vztahu k jiným záznamům :



Tak jak byla vytvořena databáze ACI Video bude uživatelům fakturačního systému zabráněno vymazat záznam zákazníka jestliže k němu již existují záznamy faktur. Když takovýto případ nastane, uživatel je upozorněn databázovým strojem hlášením podobným tomu ukázanému níže:



Samozřejmě nyní jsme toto hlášení nahradili našim vlastním hlášení, ale koncept chování databázového stroje je tentýž jako při řízení mazání.

9.2. Pravidla databáze a omezení

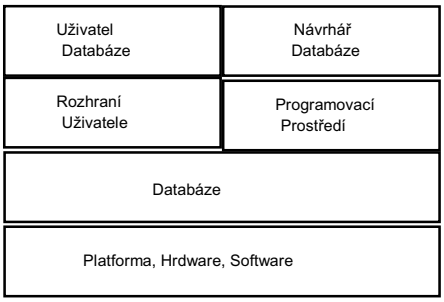
Pravidla a omezení databáze jsou vždy přímo svázány s reálným životem a aspekty oblasti, kterou řídíte pomocí databáze. Zde máme dva příklady:

- Uživatel nemůže vložit záznam pro [Zaměstnance] bez vložení rodného čísla zaměstnance, kromě toho toto číslo musí být dělitelné 11 a musí obsahovat 10 číslic. Toto číslo musí být také v dané tabulce jedinečné.

- Uživatel nemůže potvrdit záznam [Faktury] bez určení názvu nebo čísla ID pro zákazníka na kterého je faktura vystavena. Kromě toho řádek Zálohy zákazníka musí být zkontrolován před tím než bude faktura přijata.

Navrhování databáze 4D se podstatně řídí pravidly reálného života je nutné je nejdříve rozpoznat a pak implementovat do návrhu databáze pomocí nástrojů poskytovaných databázovým strojem, uživatelským rozhraním a jazykem 4th Dimension.

V širším rozsahu mohou být celkové nástroje znázorněny obrázkem níže:

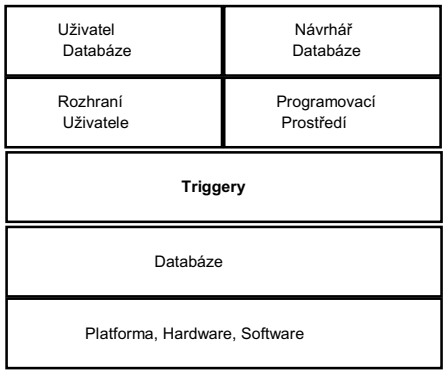


Člověk rychle pochopí, že využití 4th Dimension i když tato poskytuje sofistikované nástroje pro zajištění integrity dat je velice závislé na programátorských schopnostech návrháře databáze.

Typicky ošetření pravidel musí být prováděno obvykle několikanásobně a simultánně jak na úrovni vlastností tak v kódu.

Aby bylo možno zredukovat či dokonce vyloučit pravděpodobnost porušení pravidel databáze na všech úrovních databázového stroje, 4th Dimension poskytuje **triggery**.

Obrázek níže ukazuje kam jsou triggery umístěny v architektuře 4D:



Triggery jsou částí kódu 4D přiřazeného přímo tabulkám a spouštěného pokaždé, když je přistoupeno k záznamu a když nastane některá z databázových událostí/akci nezávisle na zdroji akce:

- Vstup dat
- Standartní nástroje uživatelského prostředí (Import, Užít výraz, Vymazat atd.)
- Provedení určité akce v metodě projektu (SAVE RECORD, DELETE SELECTION,...)
- 4D Extensions

- Aplikace založené na 4D Open
- Připojení WEB

9.3. Triggery a události databáze

Triggery jako metody přiřazené tabulkám se spouštění automaticky, když nastane určitá událost na úrovni databázového stroje, tyto události jsou:

- Uložení nového záznamu
- Uložení existujícího záznamu
- Vymazání záznamu
- Zavedení záznamu

Triggery mohou být spuštěny kdykoliv nastane některá z těchto událostí, bez ohledu na zdroj akce a případně formulář, který je dotčen. Např. trigger tabulky bude spuštěn při importu dat (za předpokladu, že je patřičná událost zaškrtnuta na stránce Triggery ve vlastnostech tabulky).

9.4. Commands for working with triggers

9.4.1. Database event -> číslo

Je-li volána zevnitř triggeru, vrátí příkaz Database event číselnou hodnotu, která označuje typ události databáze; jinými slovy důvod, pro který byl trigger spuštěn.

0 vně jakéhoholiv prováděcího cyklu triggeru
1 ukládání nového záznamu
2 ukládání existujícího záznamu
3 mazání záznamu
4 zavádění záznamu

Jestliže je událost databáze Ukládání existujícího záznamu (Save Existing Record Event), můžete zjistit, jetli bylo změněno určité pole polocí funkce Modified. V tomto případě, můžete pokud je potřeba obnovit hodnotu v poli (byl již v předchzím uložen na disk) pomocí příkazu Old.

Poznámka: Tyto příkazy jsou užitečné pouze při aplikaci na jednoduchá pole, alfanumerická a číselná (jejich výsledek nemá význam s poli typu Picture, BLOB nebo Podtabulka).

9.4.2. In transaction -> Logické

Jestliže, uvnitř triggeru, provedete operaci na více záznamů (např. úprava více záznamů v tabulce Produkty, je-li záznam přidáván do tabulky Faktury), můžete narazit na podmínku (obvykle uzamčení záznamu), kerá způsobí, že trigger nemůže provést korektně, to co jste zamýšleli. V tomto případě, potřebujete zastavit další provádění, vrátit chybu databáze, aby proces věděl, že operaci nebyla dokončena a je potřeba se vrátit k původnímu stavu databáze. V tomto případě volající proces musí být schopen zrušit operace, které trigger provedl nekompletně. Jinými slovy před tím než trigger vůbec něco provede potřebujete znát, zda se původní proces může navrátit k původnímu stavu dataváze, tj. byla-li vyvolána transakce. K tomuto slouží příkaz In transaction.

9.4.3. Trigger Level -> Číslo

4th Dimension nemá žádný jiný limit než dostupnost paměti, při možném kaskádovitém volání triggerů. Kromě toho 4th Dimension plně podporuje rekurzivní kaskádovité volání triggerů (je-li vaším programem vyvoláno). K optimalizaci chování triggerů, pak můžete chtít napsat kód, kdy chování vašeho triggeru závisí na úrovni volání (je-li triggger volán z normální metody, nebo jiného triggeru, třeba jiné tabulky) a ne pouze na události databáze. Například, během události databáze Mazání záznamu tabulky Faktury, můžete chtít přeskočit obnovu pole [Zákazníci]Celkové prodeje, jestliže mazání jedné faktury je součástí mazání všech záznamů faktur jednoho zákazníka spolu s vymazáním i záznamu tohoto zákazníka. Pro tento účel pak použijete příkaz Trigger level nebo TRIGGER PROPERTIES.

9.4.4. TRIGGER PROPERTIES (úroveňTriggeru; událostdb; čísloTab; číslozázn)

Parametr	Typ	Popis
úroveňTriggeru	číslo	úroveň prováděcího cyklu triggeru
událostdb	číslo	událost databáze
číslotab	číslo	pro tabulku číslo
číslozázn	číslo	pro záznam číslo

9.5. K čemu jsou triggery?

Triggery vám dovolí zavést a kontrolovat následující pravidla a omezení databáze:

- Integritu dat tabulky
- Ovládání složených klíčů (složených z více polí)
- řídit BLOBy
- integritu dat přes vztahy

- normalizace vkládání dat
- normalizace výstupních dat (zde pozor)
- přístupová práva
- auditování databáze
- disktibuované systémy, replikace a synchronizace dat

9.6. Na co nejsou triggery?

Nesprávné použití triggerů může snížit chování databáze. Jestliže máte trigger, který upravuje data ve vztahu, může se tento zacyklovat při čekání na odemčení zamčeného vztaženého záznamu.

- obnova vztažených tabulek (pokud velmi opatrně)
- tisk záznamů

9.7. Vyzkoušení existujících triggerů v naší databázi.

V předchozím kurzu jsme zavedli triggery pro ovládání polí DatumÚpravy a ČasÚpravy pro již existující záznamy.

9.7.1. Vyzkoušení existujících triggerů.

1. Vyzkoušejte existující triggery pro tabulku [Zákazníci].

9.8. Triggery navrací výsledek!

Všechny triggery navrací hodnotu. Tato hodnota musí být typ Longinteger. A proto musíte přiřadit parametru $0 hodnotu. Jestliže bude přiřazena jiná hodnota než 0, způsobí to přerušení operace. Hodnoty „chyby“, kterou navracíte by měli být menší než –15000.

I když trigger nevrátí chybu ($0:=0), neznamená to, že operace bude úspěšně dokončena, může se vyskytnout porušení jedinečnosti hodnoty, záznam může být uzamčen a může nastat chyba vstupu a výstupu atd.

Z pohledu vyšší úrovně z procesu jsou však chyby navrácené databázovým strojem nebo triggerem téže podstaty. Pro tuto úroveň jsou chyby triggeru chybami databázového stroje.

9.9. Řízení mazání požaduje automatický vztah skupiny k jedinci

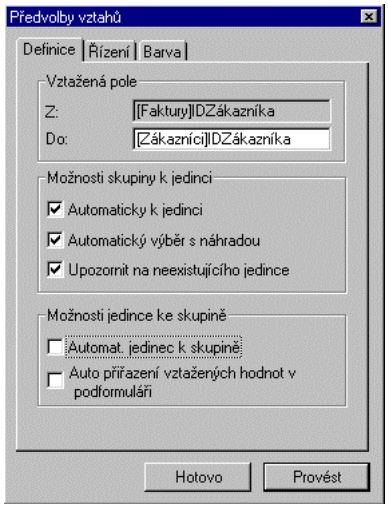
Řízení mazání ačkoli je to dobré věc má i své špatné stránky. Jestliže je v prostředí klient/server zapnuta automatická relace skupiny k jedinci se vždy, když se zavádí

záznam rodičovské tabulky zobrazí i všechny záznamy z tabulky potomků. Tj. je provedeno vyhledání záznamů potomků a záznamy jsou odeslány na stroj klienta pro zobrazení. To nemusí být vhodné chování. Konec konců kolikrát chce uživatel při zobrazení záznamu zákazníka skutečně vidět i všechny jeho faktury.

Toto chování bylo nezbytné pro řízení mazání. 4D v6 nabízí i vaše vlastní řešení. Napište své vlastní řízení mazání v triggeru.

9.9.1. Provedení vztahu [Zákazníci] - [Faktury] neautomatickým.

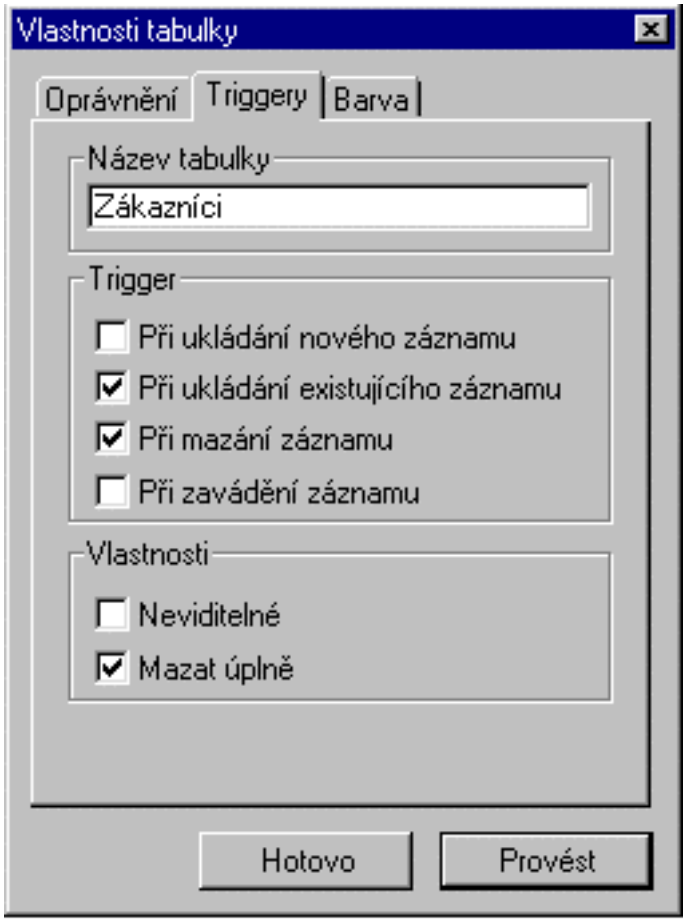
1. Otevřete poklepáním na čáru vztahu [Zákazníci] & [Faktury] dialogové okno vztahu.
2. Odškrtněte políčko Automaticky jedinec ke skupině.



3. Klepněte na tlačítko Provést a Hotovo.

9.9.2. Vytvoření triggeru k řízení mazání.

1. Otevřete Vlastnosti tabulky pro tabulku [Zákazníci].
2. Přejděte na stránku Triggery.
3. Zaškrtněte trigger Při mazání záznamu.
4. Klepněte na tlačítko Provést a Hotovo.



5. Upravte trigger [Zákazníci] následujícím způsobem.

```
If (False)
   ` Trigger: Zákazníci
   ` Kurz programovani ACI University
   ` Autor: Jim Steinman
   ` Datum: 1/15/97

   ` Účel: Trigger k ovládání událostí databáze pro [Zákazníci]

  <>f_Version6x10:=True
❖ <>f_Version6x30:=True
  <>fJ_Steinman:=True

❖    ` Upraveno: 3/17/97
❖ <>fK_Wilbur:=True
❖    ` Přidány rysy řízení mazání

End if
```

```
C_LONGINT($LDatabaseEvent)    ` Testovaná databázová událost
C_LONGINT($0)                 ` Navracený kód chyby je-li nějaký

$LDatabaseEvent:=Database event
$0:=0                         ` Přiřazení výchozího kódu chyby

Case of
  ś ($LDatabaseEvent = On Saving Existing Record Event)
    GEN_TimeStamp (->[Zákazníci]DatumUpravení;->[Zákazníci]ČasUpravení)

  ś ($LDatabaseEvent = On Deleteting Record Event)
    RELATE MANY([Zákazníci]IDZákazníka)
    If (Records in selection([Faktury])>0)
      $0:= -16001             ` Existují záznamy ve vztahu navrať kód chyby
    End if

End case
  `Konec triggeru
```

5. Přepněte se do prostředí Vlastní nabídky a testujte své nové řízení mazání.

9.9.3. Zachycení chyb triggeru

Kódy chyby navracené triggery jsou detekovatelné úplnně stejným způsobem jako kódy chyby navracené 4D. Své vlastní kódy chyby můžete zachytávat a ošetřovat je svým vlastním způsobem stejně jako zachytávané chyby ve 4D.

1. Nahraďte metodu projektu ERROR_DeletingRecords následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
    ` Metoda: ERROR_DeletingRecords
    ` Kurz programovani ACI University
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Řízení chyb vznilkých při mazání záznamů

  <>fGeneric:= True
  <>f_Version6x30:=True
  <>fK_Wilbur:=True
End if

C_STRING(255;$sMessage)

$sMessage:=""

Case of
```

```
   :(Error = -9987)
      $sMessage:="Jiné záznamy v databázi závisí na informacích ze záznamů"+" které se
snažíte vymazat!"+Char(Carriage return )+Char(Carriage return ))+"Ukončuje
vymazávání záznamů!"
❖  :(Error = -16001)
❖     $sMessage:="Jeden ze zákazníků, které se snažíte vymazat má faktury." +
❖                Char(Carriage return)+"Záznam zákazníka nemůže být vymazán!"

   :(Error = -9984)
      $sMessage:="Záznab byl již vymazán!" + Char(Carriage return)+Char(Carriage
return)) + "Zastavuji mazání záznamů!"

   :(Error = -9986)  ` záznam zamčen, nedělat nic

   :(Error = -9991)
      $sMessage:="Nemáte oprávnění k této akci!"

Else
      $sMessage:="Nastala neočekávaná chyba #  " + String(Error) + "!" +Char(Carriage
return)) + "Prosím poznamenejte si číslo a zkonzultujte svého administrátora pro
nápravu!"
End case

If( Length ($sMessage) >0)
   ALERT($sMessage)
End if
      ` Konec metody
```

2. Přepněte se do prostředí Vlastní nabídky a otestujte své řízení mazání.

9.9.4. Úpravy vyvolané ve formuláři [Zákazníci]Vstupní.

Nyní když jste vypnuli automatickou relaci jedince ke skupině se na stránce 2 formuláře [Zákazníci]Vstupní neobjeví [Faktury]. Pro tento účel musíte přidat kód, který zobrazení faktur zajistí.

1. Otevřete formulář [Zákazníci]Vstupní.
2. Vytvořte novou metodu projektu hlCustomerTabs pro objekt Ovládací karta následujícím způsobem:

```
If (False)
      ` Metoda objektu: hlCustomerTabs  [Zákazníci];"Input"
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` `Účel: Způsobí, že budou  zavedeny [Faktury]

   <>f_Version6x30:=True
```

```
   <>fK_Wilbur :=True

End if

If(Selected list item(hlCustomerTabs)=2)
   RELATE MANY([Zákazníci]IDZákazníka)
End if
      `Konec metody
```

3. Upravte metodu formuláře [Zákazníci]Vstupní následujícím způsobem:

```
If (False)
      ` Metoda formuláře.: Vstupní
      ` Kurz programovani ACI University
      ` Datum: 1/15/97
      ` By: Jim Steinman

      `Účel: Způsobí, že budou zavedeny [Faktury]

   <>f_Version6x10:=True
   <>f_Version6x30:=True
   <>fJ_Steinman:=True

      ` Upraveno:  3/17/97
      ` Ovladací karta bude synchronizovaná s aktuální stranou formuláře
❖     ` Použije relate many při procházení záznamů na straně 2
<>fK_Wilbur:=True

End if

$FormEvent:=Form event

Case of
   ś ($FormEvent = On Load)
      If (pTable = (->[Faktury]))       ` Přichází z faktur přes tlačítko Upravit zákazníka
         SET VISIBLE(*; INVOICE_Button@; False)  ` Nemůže dělat funkce na faktuře
         SET WINDOW TITLE("Modifying "+[Zákazníci]Firma)
         SET LIST ITEM PROPERTIES(hlCustomerTabs;2;False;Plain;15002)
      Else
         WIN_InputWindowTitle

❖        If (Current form page = 2)
❖           RELATE MANY([Zákazníci]IDZákazníka)
❖        End if

      End if

      If (Record number([Zákazníci]) = -3)
         [Zákazníci]FirmaID:=String( LNextSequence ("CustomerID"))
         GEN_TimeStamp (->[Zákazníci]DatumVytvoření;->[Zákazníci]ČasVytvoření)
      End if

      SELECT LIST ITEM (hlCustomerTabs;Current form page)

      ckCapatilizeOff :=0

End case
      `Konec metody
```

4. Přepněte se prostředí Vlastní nabídky a otestujte přechod na stránku 2 a procházení záznamů na stránce 2.

Extra Kredit Diskuse

Automatický vztah ke skupině nemusí být vždy špatný. Např. co když chcete nalézt zákazníky, kteří mají fakturu větší než Kč 2000? S automatickým vztahem ke skupině to bude jednoduché, přejdete pouze do Editoru dotazů a napíšete podmínku dotazu „přes vztah“ a tím naleznete správné záznamy. S vypnutým automatickým vztahem to však není možné. Dotaz v tomto případě nevrátí žádný výsledek. My však potřebujeme nalézt možnost alespoň dočasně zapnout automatické vztahy i kdyby jen pro dotazy a pak je opět vypnout.

9.10. AUTOMATIC RELATIONS (jedinec; skupina)

Parametr	Typ	Popis
jedinec	Logické	vztahy skupiny k jedinci
skupina	Logické	vztahy jedinec ke skupině

Příkaz AUTOMATIC RELATIONS dočasně změní všechny manuální vztahy na automatické vztahy a to v celé databázi. Vztahy zůstanou automatické dokud je nezmění následné volání AUTOMATIC RELATIONS.

Jestliže je parametr jedinec True pak se automatickými stanou všechny manuální vztahy skupiny k jedinci. Jestliže je parametr jedinec False všechny vztahy skupiny k jedinci definované ve struktuře jako manuální se vrátí do tohoto vztahu.

Totéž platí pro parametr skupina, samozřejmě s tím, že dotčeny jsou vztahy jedinec ke skupině.

Vztahy, které jsou v Prostředí návrháře nastaveny jako automatické nejsou tímto příkazem dotčeny.

Jestliže máme všechny vztahy definovány v Prostředí návrháře jako manuální, tento příkaz umožní jejich změnu na automatické těsně před provedením operací, které potřebují, aby vztahy byly automatické (jako jsou Dotazy přes relace, Třídění přes relace a Rychlé zprávy). Po dokončení operace mohou být vztahy opět přepnuty jako manuální.

Extra Kredit Cvičení

9.9.5. Dočasné nastavení automatických vztahů

1. Upravte metodu projektu M_GEN_QueryEditor následujícím způsobem:

```
If (False)
```

```
                ` Metoda: M_GEN_QueryEditor
                ` Kurz programovani ACI University
                ` Genericke metody z nastroju ACI
                ` Created By: Jim Steinman
                ` Datum: 1/15/97

                ` Účel: Zobrazí okno dotazu pro uživatele

              <>fGeneric:= True
              <>f_Version6x10:=True
❖             <>f_Version6x30:=True
              <>fJ_Steinman:=True

❖               ` Upraveno:  3/17/97
❖               ` Použije automatické vztahy během dotazu
❖             <>fK_Wilbur:=True

            End if

❖           AUTOMATIC RELATIONS (True;True)
            QUERY (pTable->)
❖           AUTOMATIC RELATIONS (False;False)
            WIN_OutputWindowTitle
                `Konec metody
```

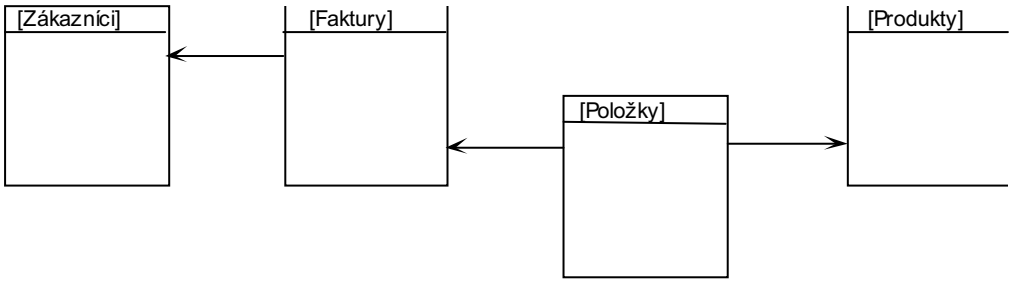
2. Upravte metodu projektu M_GEN_OrderByEditor následujícím způsobem:

```
            If (False)
                ` Metoda: M_GEN_OrderByEditor
                ` Kurz programovani ACI University
                ` Genericke metody z nastroju ACI
                ` Created By: Jim Steinman
                ` Datum: 1/15/97

                ` Účel: Zobrazý vestavěný editor třídění

              <>fGeneric:= True
              <>f_Version6x10:=True
❖             <>f_Version6x30:=True
              <>fJ_Steinman:=True

❖               ` Upraveno:  3/17/97
❖               ` Použije automatické vztahy během třídění
❖             <>fK_Wilbur:=True

            End if

❖           AUTOMATIC RELATIONS (True;True)
            ORDER BY (pTable->)
❖           AUTOMATIC RELATIONS (False;False)
                `Konec metody
```

3. Otestujte tyto změny.

9.11. Kaskádování triggerů

Vezměme do úvahy následující hypotetickou databázi. Fakt, že vám připomíná ACI Video je pouze náhodný.



Každá z tabulek má triggery, která prováděj následující akce:

[Zákazníci]

Při mazání záznamu
- Maže vztažené [Faktury]

[Faktury]

Při ukládání nového záznamu
- Obnovuje ve vztaženém záznamu [Zákazníci] částku celkové prodeje.

Při mazání záznamu
- Maže vztažené [Položky]
- Obnovuje ve vztaženém záznamu [Zákazníci] částku celkové prodeje.

[Položky]

Při ukládání nového záznamu
- Obnovuje vztažený záznam [Produkty] v údaji množství na skladě.

Při ukládání existujícícho záznamu
- Obnovuje vztažený záznam [Produkty] v údaji množství na skladě založeném na porovnání staré hodnoty v položce s novou hodnotou.

Při mazání záznamu
- Obnovuje vztažený záznam [Produkty] v údaji množství na skladě.

Tento mechanismus, kde triggery manipulují s jinými vztaženými tabulkami, což vyvolává spuštění triggerů v těchto tabulkách se nazývá Kaskádování triggerů. Problém nastává, když mažete záznam [Zákazníci]. Záznam [Zákazníci] vyvolává mazání z

[Faktury], které se snaží obnovit údaje v [Zákazníci] , což je zbytečné úsilí. V triggeru [Faktury] při mazání záznamů by jste měli testovat úroveň triggeru. Jestliže je tato úroveň 2 pak použijte příkaz TRIGGER PROPERTIES a to pro kontrolu zda je úroveň triggeru 1 mazání záznamu [Zákazníci] pokud ano tak neobnovujte údaje v záznamu [Zákazníci].

Předchozí příklad samozřejmě nemůže být považován za nejlepší způsob vyplnění zadané úlohy. Velice pečlivě je potřeba sledovat všechny aspekty úlohy zamčení záznamů, transakce atd. Pouze po pečlivém vyhodnocení se můžete rozhodnout o nejlepším způsobu vyplnění úlohy. Předchozí považujte pouze za příklad pro testování úrovně triggeru a jeho vlastností.

9.12. Speciální úvahy o triggerech

Triggery jsou prováděny na stroji, kde je skutečně umístěn databázový stroj. To je zřejmé u jednouživatelské verze 4D. Na 4D Server jsou triggery prováděny v patřičném procesu na stroji serveru a ne na stroji klienta. Když je vyvolán trigger je proveden v kontextu procesu, který způsobil databázovou operaci. Tento proces, který vyvolal provedení triggeru je nazýván Vyzývající proces.

Triggery obzvláště pracují s platnými výběry, záznamy, stavy číst či psát a stavem uzamčenosti záznamu z vyzývajícího procesu.

UPOZORNĚNÍ: Triggery nemohou a nesmí změnit platný záznam tabulky, pro kterou jsou vyvolány. Jestliže uvnitř triggeru potřebujete zkontrolovat jedinečnost hodnoty v několika polích, použijte příkaz SET QUERY DESTINATION, který vám dovolí hledat v tabulce bez změny platného výběru a platného záznamu tabulky.

Především si dejte pozor na použití jiných objektů jazyka nebo databáze z prostředí 4D, protože trigger může být spuštěn na úplnně jiném stroji než běží vyzývající proces tj. v případě 4D Server!

- Meziprocessní proměnné: Trigger má přístup k meziprocessním proměnným stroje na kterém je prováděn. V případě 4D Server je to jiný stroj než vyzývající proces.

- Proměnné procesu: Všemi triggery je sdílena nezávislá tabulka proměnných procesu. Trigger nemá přístup k proměnným vyzývajícího procesu. V triggeru by jste neměli používat proměnné procesu.

- Místní (lokální) proměnné: V triggeru můžete používat lokální proměnné. Jejich rozsah je provádění triggeru, jsou vytvářeny/mazány při každém provedení.

- Semafory: Triggery mohou testovat či nastavovat globální semafory rovněž tak i lokální semafory (na stroji kde jsou spouštěny). Triggery se však musí provádět rychle, takže musíte být opatrní při testování a nastavování semaforů z triggeru.

- Sady a pojmenované výběry: Jestliže používáte sady a pojmenované výběry z triggeru, pracujete na stroji, kde jsou triggery prováděny.

- Uživatelské rozhraní: Nepoužívejte prvky uživatelského rozhraní v triggeru (tj. žádná upozornění, zprávy a dialogová okna). Shodně by jste měli zcela omezit krokování triggerů v okně ladění. Připomeňme si, že na klient/server jsou triggery vykonávány na stroji 4D Server. Zpráva upozornění na stroji serveru asi uživateli na pracovní stanici mnoho nepomůže. Takže ponechte ovládání uživatelského rozhraní pouze pro vyzývající proces.

9.13. Trigger a Transakce

Transakce musí být ovládány na úrovni vyzývajícího procesu, neměli by jste ovládat transakce ne úrovni triggeru. Během jednoho provádění triggeru jestliže musíte přidat, upravit či vymazat několik záznamů (viz případovou studii níže), musíte nejdříve použít příkaz IN TRANSACTION uvnitř triggeru k otestování zda vyzývající proces není právě v transakci. Jestliže tomu tak není trigger může potenciálně narazit na uzamčený záznam. Proto jestliže vyzývající proces není v transakci ani nezačínejte operaci se záznamy, pouze navraťte chybu v $ 0, abyjste signalizovali vyzývajícímu procesu, že databázová operace, kterou zkusil provést musí být provedena v transakci. Jinak jestliže je dotčen uzamčený záznam vyzývající proces nebude mít žádný prostředek k navrácení akce provedené triggerem částečně.

Speciální poznámka: V žádném případě by jste neměli startovat transakci uvnitř triggeru. Můžete kontrolovat jestli jste v transakci (Vyzývající proces), ale nikdy by jste neměli začít transakci v triggeru.

10. Použití triggerů k podpoře složených primárních klíčů

Jeden z nejčastěji kritizovaných nedostatků 4th Dimension je, že neexistuje žádný způsob pro podporu složených primárních klíčů. Tj. zajištění jedinečnosti hodnot v kombinaci polí bez vytvoření složeného pole z výchozích hodnot.

Podívejme se na příklad naší tabulky [PoložkyFaktury] s poli ČísloFaktury a ProduktČíslo:

V naší současné databázi může uživatel vytvořit tutéž položku se stejným produktem ve faktuře kolikrát chce. Proto v ideálním případě můžete chtít zakázat vytvoření dvou záznamů položky faktury s identickými hodnotami v těchto dvou polích.

Abychom to mohli provést provedeme si několik příkazů:

10.1. SET QUERY LIMIT (číslo)

Příkaz SET QUERY LIMIT vám umožní říci 4th Dimension, aby ukončila dotazování pro platný proces jakmile dosáhne vyhledání počtu záznamů, které jste předali jako limit v parametru číslo.

Jestliže například předáte limit rovný 1, zastaví se prohledávání indexu či datového souboru jakmile bude nalezen první záznam vyhovující podmínce dotazu.

K obnovení vyhledávání bez limitu vyvolejte opět příkaz SET QUERY LIMIT s parametrem 0.

UPOZORNĚNÍ: SET QUERY LIMIT působí na všechny následné dotazy provedené uvnitř platného procesu. Nezapomeňte vždy znovu volat příkaz SET QUERY LIMIT (0) po nastavení limitu příkazem SET QUERY LIMIT (limit) (kde limit>0)

SET QUERY LIMIT změní chování pouze příkazu dotazu. Na druhou stranu SET QUERY LIMIT na další příkazy, které mohou změnit platný výběr tabulky jako např. ALL RECORDS, RELATE MANY atd.

10.2. SET QUERY DESTINATION (SměrTyp{; CílObjekt})

Parametr	Typ	Popis
SměrTyp	Číslo	
		0 platný výběr 1 sada/set 2 pojmenovaný výběr 3 proměnná typu BLOB
CílObjket	Řetězec | Proměnná	
		Název Sady, Pojmenovaného výběru Proměnné

Příkaz SET QUERY DESTINATION vám dovolí říci 4th Dimension kam má umístit výsledek jakéhokoliv následného dotazu v platném procesu.

Samotný cíl pro uložení výsledku dotazu určíte parametrem CílObjekt ve shodě s následující tabulkou:

SměrlTyp	CílObjekt
0 (platný výběr)	Parametr vynechán. Příklad: SET QUERY DESTINATION(platný výběr) Záznamy nalezené jakýmkoliv následným dotazem přepíší platný výběr tabulky pro kterou je dotaz proveden.
1 (sada)	Musíte předat název sady (existující nebo k vytvoření) Příklad: SET QUERY DESTINATION(do sady;"můjSet")

Nalezené záznamu v jakémkoliv následném dotazu budou umístěny do sady („můjSet“). Platný výběr a platný záznam pro dotazem dotčenou tabulku zůstanou nezměněny.

2 (pojmenovaný výběr) Musíte předat název pojmenovaného výběru (existujícího nebo k vytvoření)
Příklad: SET QUERY DESTINATION(do pojmenovaného výběru;"můjPojmVýb")

Záznamy nalezené v jakémkoliv následném dotazu budou umístěny do pojmenovaného výběru „můjPojmVýb“. Platný výběr a platný záznam tabulky dotčené dotazem zůstanou nezměněny.

3 (proměnná) Musíte předat číselnou proměnnou (existující nebo k vytvoření)
Příklad: SET QUERY DESTINATION (do proměnné;$vlQueryResult)
Údaj o počtu záznamů nalezených v jakémkoliv následném dotazu bude umístěn do proměnné $vlQueryResult: Platný výběr a platný záznam pro tabulku dotčenou dotazem zůstane nezměněn..

UPOZORNĚNÍ: Příkaz SET QUERY DESTINATION působí na všechny následné dotazy provedené uvnitř platného výběru. Nezapomeňte vždy navráti směr voláním SET QUERY DESTINATION (0) po předchozím volání příkazu SET QUERY DESTINATION (kde typ#0).

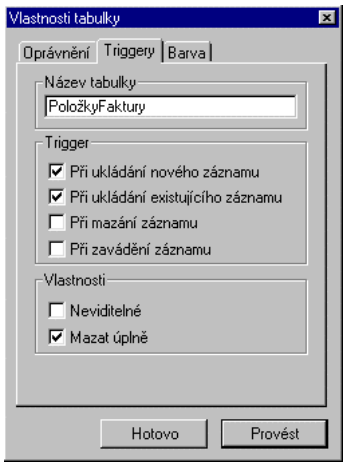
SET QUERY DESTINATION změní pouze chování příkazů dotazů.

10.3. Udržování jedinečných hodnot pro kombinaci polí

První metoda, kterou použijeme je jednoduché hledání k nalezení dotčených záznamů.

10.3.1. Zavedení složeného primárního klíče

1. Otevřete okno Vlastnosti tabulky pro tabulku [PoložkyFaktury].
2. Zapněte triggery Při ukládání nového záznamu a Při ukládání existujícího záznamu.



3. Vytvořte trigger pro tabulku [PoložkyFaktury] následovně nebo importujte pomocí textového editoru:

```
If (False)
      ` Trigger: LineItems
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Ovládá chyby při mazání záznamů

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definici lokálních proměnných
C_LONGINT($0)                  ` Navracená proměnná
C_LONGINT($LDatabaseEvent)     ` Událost databáze
C_LONGINT($LQueryResult)       ` Výsledek dotazu

$0:=0                          ` Předpokládáme dobrý výsledek
$LDatabaseEvent:=Database event

Case of
      ` Jestliže je záznam ukládán (nový nebo existující)
   ś (($LDatabaseEvent= On Saving New Record Event ) | ($LDatabaseEvent=On Saving
Existing Record Event ))
            ` Potřebujeme nalézt pouze jeden dodatečný záznam
      If (Length([PoložkyFaktury]IDZbožíPoložky)>0)

         SET QUERY LIMIT(1)
```

```
            ` Nechceme upravovat platný výběr
            ` Nemůžeme měnit platný záznam
            ` Pouze potřebujeme znát jestli není další záznam
        SET QUERY DESTINATION(Into variable; $LQueryResult)

            ` Provedení dotazu
        QUERY([PoložkyFaktury];[PoložkyFaktury]ČísloFaktury =
[PoložkyFaktury]ČísloFaktury;*)
        QUERY([PoložkyFaktury]; & ;[PoložkyFaktury]IDZbožíPoložky =
[PoložkyFaktury]IDZbožíPoložky)

            ` Jestliže byl nalezen alespoň jeden další záznam nelze záznam přijmout
        If ($LQueryResult>0)
          $0:=-19001
        End if

          ` Návrat do dotazu bez limitu
        SET QUERY LIMIT(0)
        SET QUERY DESTINATION(Into current selection)
      End if
End case
    `Konec triggeru
```

Nezáleží na způsobu jakým se vyzývající proces pokouší uložit záznam. Uložení bude přerušeno s chybou databáze -19001 jestliže budou porušena pravidla stanovená výše.

4. Přepněte se do prostředí Vlastní nabídky a proveďte testování.

10.3.2. Přidání vlastních zpráv o chybách k zobrazení zdvojených záznamů

V této chvíli může být ve vyzývajícím procesu chyba zachycena pomocí metody ON ERR CALL.

1. Vytvořte novou metodu projektu ERROR_Invoices následujícím způsobem:

```
If (False)
    ` Metoda: ERROR_Invoices
    ` Kurz programovani ACI University
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Ošetřuje chyby při zdvojeném záznamu  [PoložkyFaktury]

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if
```

```
C_STRING(255;$sMessage)

$sMessage:=""

Case of
   :(Error = -19001)
      $sMessage:="Již existuje jiná položka této faktury s tímto produktem!"

Else
      $sMessage:="Vyskytla se neočekávaná chyba #  " + String(Error) +
                        Char(Carriage return)) + "Poznamenejte si prosím číslo a
zkonzultujte svého administrátora!"
End case

If( Length ($sMessage) >0)
   ALERT($sMessage)
End if
      ` Konec metody
```

2. Upravte metodu formuláře [Faktury]Vstupní následujícím způsobem:

```
If (False)
      ` Metoda formuláře:[Faktury] Vstupní
      ` Kurz programovani ACI University
      ` Datum: 1/15/97
      ` By: Jim Steinman

      ` Účel: Metoda vstupního formuláře pro [Faktury]; "Vstupní"

   <>f_Version6x10:=True
   <>f_Version6x20:=True
❖  <>f_Version6x30:=True
   <>fJ_Steinman:=True

      ` Upraveno:  1/17/97
   <>fK_Wilbur:=True

❖     ` Upraveno:  3/17/97
❖     ` Přidává ošetřování chyb ke kontrole zdvojených položek
❖  <>fK_Wilbur:=True

End if

C_LONGINT($LFormEvent)

$LFormEvent:=Form event

Case of
```

```
      ś ($LFormEvent= On Load)
…

START TRANSACTION

❖       ON ERR CALL("ERROR_Invoices")

❖       ś ($LFormEvent= On Unload)
❖       ON ERR CALL("")

…
```

3. Zkontrolujte zda událost formuláře Při vyvedení je zaškrtnuta pro formulář [Faktury];"Vstupní ".
4. Přepněte se do prostředí Vlastní nabídky a proveďte testování.

Poznámka: Rychlost dotazu pro více polí v databázi s velkým počtem záznamů nebo pro kombinaci více než dvou polí, může být zlepšena přesměrováním dotazu do sad a dotazem na každé pole zvlášť. Nakonec bude proveden průnik těchto sad.

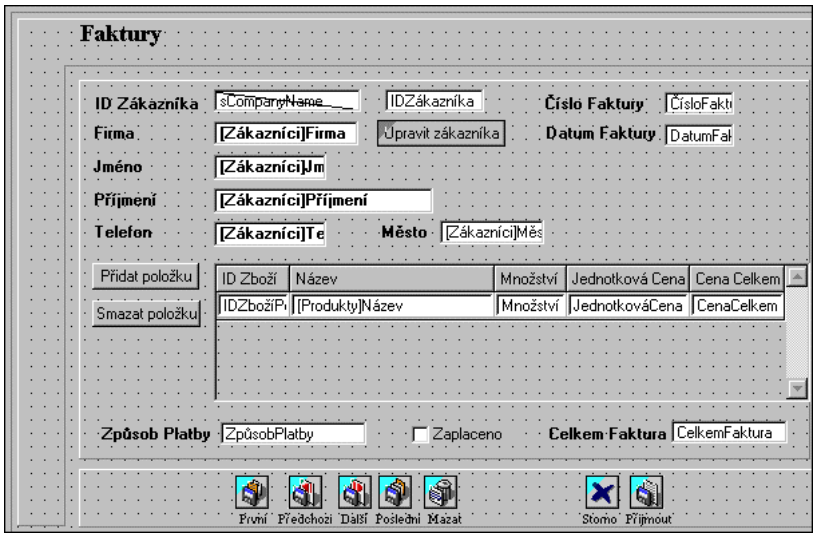
11. Úpravy databáze, které sníží závislost uživatele na umělých klíčích

Velice často zadavatel a uživatel databáze nemá nejmešní představu o tom jak relační databáze musí pracovat. Vlastník ACI Video zachází pouze s názvy zákazníků. Sám nikdy nepoužívá čísla zákazníků. Takže když mu dodáme databázi, která používá čísla ID zákazníka bude zmaten. Na druhé straně vám, ale řekne, že názvy firem zákazníků se často mění. Z tohoto důvodu vám vznikne dilema. Co je potřeba udělat proto, aby jste do konečného řešení vložili to nejlepší z obou těchto rozdílných světů. Je potřeba vytvořit systém, který dovolí uživateli používat záměnným způsobem buď ID zákazníka nebo název firmy.

Nyní když uživatel zakládá novou fakturu musí znát ID zákazníka. Vytvoříme systém, který umožní zadat obojí. Znamená to však, že ve skutečnosti nebudeme zobrazovat na formuláři uložená data. Místo toho zobrazíme proměnné kam uživatel může napsat data a pak my provedeme všechnu ostatní práci v kódu. To nabídne uživateli větší pružnost než vestavěné automatické rysy 4D, ale vyžaduje to více kódu.

11.0.1. Příprava formuláře [Faktury]Vstupní

1. Otevřete formulář [Faktury]Vstupní.
2. Vytvořte dostupnou proměnnou `sCompanyName` o stejné velikost jakou má současně zobrazené pole Společnost a také se stejnými vlastnostmi.
3. Nastavte pole IDZákazníka jako nedostupné.
4. Uspořádejte formulář stejně jako na obrázku níže. Můžete dokonce formulář zmenšit což vám umožní lepší překrývání oken.



5. Změňte pořadí vstupu tak, že proměnná sCompanyName bude první v pořadí.
6. Upravte metodu formuláře [Faktury]Vstupní následovně.

```
If (False)
    ` Metoda formuláře:[Faktury] Input
    ` Kurz programovani ACI University
    ` Datum: 1/15/97
    ` By: Jim Steinman

    ` Účel: Input form method for [Faktury]; "Input"

  <>f_Version6x10:=True
  <>f_Version6x20:=True
  <>f_Version6x30:=True
  <>fJ_Steinman:=True

    ` Upraveno:  1/17/97
  <>fK_Wilbur:=True

    ` Upraveno:  3/17/97
    ` Added Error handling to check for duplicate line items
❖   ` Změněno pro užití a zobrazení sCompanyName
  <>fK_Wilbur:=True

End if

C_LONGINT($LFormEvent)

$LFormEvent :=Form event
```

```
Case of

  ś ($LFormEvent= On Load)
    fNewInvoice:=False
❖   sCompanyName:=[Zákazníci]Firma          ` Přiřazení názvu do proměnné

    If (pTable = (->[Zákazníci]))          ` Coming from the Customers file with bOpen
button
      SET VISIBLE(bModify; False)   ` Can't go back to modify customers
      SET WINDOW TITLE("Modifying Invoice "+String([Faktury]ČísloFaktury))
    Else
      If (Record number([Faktury] = -3)
        fNewInvoice:=True
        [Faktury]ČísloFaktury:=LNextSequence ("InvoiceNo")
❖       SET ENTERABLE(sCompanyName;True)
❖       sCompanyName:=""
        GEN_TimeStamp (->[Faktury]DatumVytvoření;->[Faktury]ČasVytvoření)
      End if
    WIN_InputWindowTitle
    End if

❖   If (Not(fNewInvoice ))          ` Existující záznam
❖     SET ENTERABLE(sCompanyName;False)
❖   End if

START TRANSACTION
…
```

7. Přepněte se do prostředí Vlastní nabídky a otestujte změny ve formuláři faktury.

11.1. Více array sloučených do skupiny

Nyní když jsme změnili IDZákazníka na nedostupné nemáme žádný způsob jak přiřadit fakturu k zákazníkovi. Automatické rysy výběru náhradou byly svázány s polem IDZákazníka, takže tyto automatické rysy jsou nyní nedostupné. Okno výběru s náhradou ukazuje pouze dvě pole. Jeden sloupec pro primární klíč z tabulky jedinců a jiný sloupec pro pole, které jste vybrali. Ale co když primární klíč není důležitý nebo dva sloupce informace nejsou dost? Proto si nyní vytvoříme náš vlastní systém výběru.

11.1.1. Vytvoření posuvné oblasti se skupinou array

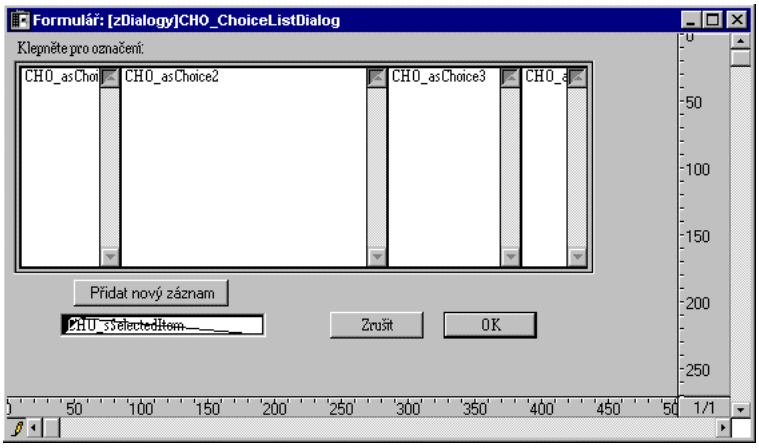
1. Vytvořte nový formulář [zDialogy] ;"CHO_ChoiceListDialog".

2. Na formuláři vytvořte následující objekty:

Typ objektu	Název objektu	Velikost	Vlastnosti	Události
Statický text	"Vybrat položku"	Žádné	Styl: Popisky vstupní	Žádné
Posuvná oblast	CHO_asChoice 1	W 75 x H 150	Zvětšit svisle Styl: Pole vstupní	Při klepnutí
Posuvná oblast	CHO_asChoice 2	W 200 x H 150	Zvětšit podélně Styl: Pole vstupní	Při klepnutí
Posuvná oblast	CHO_asChoice 3	W 100 x H 150	Zvětšit podélně Styl: Pole vstupní	Při klepnutí
Posuvná oblast	CHO_asChoice 4	W 50 x H 150	Zvětšit podélně Zvětšit svisle Styl: Pole vstupní	Při klepnutí

Výchozí tlačítko	CHO_bOK Button Text: "OK"	Rozhodněte	Akce: Přijmout Klávesa: Enter Přesunout svisle Přesunout podélně Styl: Tlačítka	Při klepnutí
Tlačítko	CHO_bCancel Button Text "Cancel"	Rozhodněte	Akce: Zrušit Key: Ctrl+ . Posunout svisle Posunout podélně Styl: Tlačítka	Při klepnutí
Tlačítko	CHO_bEscape Button Text: Escape	Pod zápatím	Akce: Zrušit Key: Escape Posunout svisle Posunout podélně Styl: Tlačítka Nedostupné tab	Při klepnutí

Nedostupná oblast	CHO_sSelectedItem	Rozhodněte	Styl: Pole vstupní Ponořené Zvětšit podélně Posunout svisle	Žádné



Z posuvných oblastí můžete ve formuláři vytvořit jednu skupinu. Tato skupina se bude chovat jako jedna posuvná oblast. Každá posuvná skupina může mít vlastní písmo a styl. Když je během vstupu dat zobrazována posuvná oblast pouze na popředí bude mít posuvník.

Při vytváření posuvných oblastí ve skupině je několik důležitých věcí, které si musít pamatovat:

- Používejte stejnou velikost písma pro každou oblast
- Všechny oblasti vytvořte o stejné výšce
- Zarovnejte stejně vrchní část všech oblastí
- Ujistěte se, že se oblasti vzájemně dotýkají, ale nepřekrývají
- Oblast zcela vpravo přesuňte na popředí tak, aby se zde objevil posuvník

- Ze všech posuvných oblastí vytvořte skupinu (pomocí položky nabídky Vytvořit skupinu), tak aby pracovaly všechny jako jedna posuvná oblast
- Nepoužívejte oblasti ve skupině pro Web.

3. Vytvořte metodu objektu pro CHO_asChoice1 následujícím způsobem:

```
If (False)
      ` Metoda objektu:CHO_asChoice1 [zDialogy];"CHO_ChoiceListDialog"
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Rozezná výběr uživatele

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

If(CHO_asChoice1 > 0)
   CHO_sSelectedItem:=CHO_asChoice2{CHO_asChoice1}
End if
      ` Konec metody
```

4. Podobný kód umístěte pro CHO_asChoice2, CHO_asChoice3 & CHO_asChoice4.
5. Vyberte všechna čtyři array (posuvné oblasti) a vytvořte z nich skupinu.
6. Vytvořte rámeček kolem skupiny 1 bod široký a 1 bod vzdálený od výsledné oblasti.
7. Vyberte rámeček a zvětšete jej doprava o 16 bodů (velikost posuvníku).
8. Nastavte vlastnost rámečku Zvětšit svisle a Zvětšít podélně.
9. Vytvořte druhý rámeček okolo prvního, ale zvětšete jej o 5 bodů na všech stranách.
10. Nastavte vlastnost druhého rámečku na Ponořený vzhled, Zvětšit podélně a Zvětšit svisle.
11. Nastavte vlastnost formuláře na Změna možná a dejte formuláři vhodnou výchozí velikost.
12. Nastavte minimální velikost o 30 bodů doprava od konce CHO_asChoice3 s tímto parametram můžete experimentovat, aby jste porozuměli co zde nastavujete.

11.1.2. Třídění skupiny array

Často je nezbytné třídit všechna array ve skupině. Musí však být tříděna společně tak, aby každý prvek určitého sloupce stále odpovídal témuž prvku jiného sloupce.

SORT ARRAY (array{; array2; ...; arrayN}{; směr})

Parametr	Typ	Popis
array	Array	Array(e) k třídění
směr	> nebo <	> k třídění v vzestupném směru < k třídění v sestupném směru nebo vzestupný směr je-li vynecháno

SORT ARRAY třídí jedno nebo více array v vzestupném nebo sestupném pořadí. Typ takovýchto array může být libovolný kromě ukazatelů či obrázků. Array rovněž nemůže být první rozměr dvourozměrného array. Parametr směru určuje zda bude tříděno ve vzestupném nebo sestupném pořadí. Jestliže je směr větší než (>) třídění je vzestupné, jestliže směr je menší než (<) třídění je sestupné. Jestliže směr není určen pak je třídění vzestupné.

Jestiže je určeno více než jedno array jsou následující tříděna v pořadí prvního array tak, aby si odpovídali prvky. Tato další array nejsou tříděna nezávisle. To je přesně to co potřebujeme pro array ve skupině.

11.1.3. Použití posuvných oblastí array ve skupině

Vše co nyní potřebujeme je napsat metodu objektu pro sCompanyName ve formuláři [Faktury]Vstupní, která bude provádět následující:

- Přidá znak "@" k čemukoliv co vloží uživatel.
- Vyhledá v polích IDZákazníka a Firma vyhovující záznamy.
- Jestliže nalezne více než jeden záznam, vytvoří a naplní čtyři array.
  1) IDZákazníka
  2) Firma
  3) Spojené Jméno a Příjmení
  4) Formátovaný Telefon
- Zobrazí výsledná array v abecedním pořadí podle názvu firmy na formuláři, který jsme právě vytvořili

- Zaznamená výběr uživatele a naplní jej do pole [Faktuy]IDZákazníka a proměnné sCompanyName na formuláři [Faktury]Vstupní.

1. Vytvořte novou metodu projektu CHO_FillCustomerArrays následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: CHO_FillCustomerArrays
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vytvoří array pro výběr zákazníka

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definice lokálních proměnných
C_LONGINT($CHO_LNumberElements)    ` Počet záznamů v array
C_LONGINT($i)                      ` Čítač smyčky

   ` Arrays pro tabulku zákazníků
ARRAY STRING(11;CHO_asChoice1;0)
ARRAY STRING(31;CHO_asChoice2;0)
ARRAY STRING(10;$CHO_asFirstName;0)
ARRAY STRING(20;$CHO_asLastName;0)
ARRAY STRING(10;$CHO_asPhone;0)

SELECTION TO ARRAY([Zákazníci]IDZákazníka;CHO_asChoice1;[Zákazníci]Firma;
                   CHO_asChoice2;[Zákazníci]Jméno;$CHO_asFirstName;

   [Zákazníci]Příjmení;$CHO_asLastName;[Zákazníci]Telefon;$CHO_asPhone)

$CHO_LNumberElements:=Size of array(CHO_asChoice1)
ARRAY STRING(33;CHO_asChoice3;$CHO_LNumberElements)
ARRAY STRING(25;CHO_asChoice4;$CHO_LNumberElements)

For ($i;1;$CHO_LNumberElements)
   CHO_asChoice3{$i}:=$CHO_asFirstName{$i}+" "+$CHO_asLastName{$i}
   CHO_asChoice4{$i}:=String(Num($CHO_asPhone{$i});"|Phone Display Format")
End for

SORT ARRAY(CHO_asChoice2;CHO_asChoice1;CHO_asChoice3;CHO_asChoice4)

      `Konec metody
```

2. Vytvořte novou metodu projektu CHO_DisplayChoiceList následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: CHO_DisplayChoiceList
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Zobrazí výběr zákazníků

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($LWindowID)

INPUT FORM([zDialogy];"CHO_ChoiceListDialog";*)

$LWindowID:=WIN_LNewWindow (-1;-1;Upper; centered;Plain no zoom box window)
DIALOG([zDialogy];"CHO_ChoiceListDialog")
CLOSE WINDOW
      `Konec metody
```

3. Vytvořte metodu objektu sCompanyName následovně nebo importujte verzi 1 z textového editoru:

```
If (False)
      ` Metoda: sCompanyName
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Najde záznam [Zákazíci] a zobrazí výběr je-li potřeba

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

If (Length(sCompanyName)>0)

   MESSAGES OFF
   QUERY([Zákazníci];[Zákazníci]IDZákazníka=sCompanyName+"@";*)
```

```
QUERY([Zákazníci]; | ;[Zákazníci]Firma=sCompanyName+"@")
MESSAGES ON

If (Records in selection([Zákazníci])>0)
   If (Records in selection([Zákazníci])>1)
      CHO_FillCustomerArrays
      CHO_DisplayChoiceList
      If (OK = 1)
         [Faktury]IDZákazníka:=CHO_asChoice1{CHO_asChoice1}
         RELATE ONE([Faktury]IDZákazníka)
         sCompanyName:=[Zákazníci]Firma
      End if
   Else
      sCompanyName:=[Zákazníci]Firma
      [Faktury]IDZákazníka:=[Zákazníci]IDZákazníka
   End if
   ARRAY STRING(11;CHO_asChoice1;0)
   ARRAY STRING(31;CHO_asChoice2;0)
   ARRAY STRING(33;CHO_asChoice3;0)
   ARRAY STRING(25;CHO_asChoice4;0)
Else
   BEEP
   ALERT("Nebyly nalezeny žádné záznamy!";"Zkuste znova")
End if
End if
 `Konec metody
```

4. Přejděte do prostředí Vlastní nabídky a proveďte testování.

11.1.4. Přidání výběru poklepáním v posuvné oblasti

Jiným způsobem jak uživatel pravděpodobně očekává je výběr záznamu zákazníka při poklepání na jeho řádek v posuvné oblasti. Tento rys je velmi jednoduché zahrnout, je k tomu však potřeba upravit kód v každém ze všech array ve skupině. To může být poněkud nudné budete-li to provádět často při každé změně chování skupiny array. Je mnohem lepší vytvořit jednu metodu projektu a uvnitř jednotlivých array odesílat ukazatel na objekt array do této metody. Pak potřebujete-li provést změny jediným místem, které musíte změnit je metoda a ne několik metod objektů.

1. Zrušte skupinu array.
2. Pro každé array přidejte ve vlastnostech událost Při poklepnutí.
3. Vytvořte novou metodu projektu CHO_Lookup následujícím způsobem:

```
If (False)
      ` Metoda: CHO_Lookup(pointer)
      ` Kurz programovani ACI University
```

```
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Ošetřuje výběr v array

   <>fGeneric:= True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

` Deklarace parametrů
C_POINTER($1)` Ukazatel na vybrané array

` Deklarace lokálních proměnných
C_LONGINT($LElementChosen)

$LElementChosen:=$1->
If($LElementChosen > 0)
   CHO_sSelectedItem:=CHO_asChoice2{$LElementChosen }
   If (Form event = On Double Clicked)
      ACCEPT
   End if
End if
   `Konec metody
```

5. Upravte metodu objektu pro všechna array a to následovně:

```
If (False)
      ` Object Method:CHO_asChoice1
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Detect the users choice

   <>f_Version6x30:=True
   <>fJ_Steinman:=True

End if

~~If(CHO_asChoice1 > 0)~~
~~   CHO_sSelectedItem:=CHO_asChoice2{CHO_asChoice1}~~
~~End if~~
CHO_Lookup(Self)
      ` Konec metody
```

6. Vytvořte znovu skupinu z array.

7. Přejděte do prostředí Vlastní nabídky a testujte změny.

11.2. Vyhledání záznamu při psaní na klávesnici

Jiné produkty dovolují uživateli ještě další komfort při vyhledávání záznamů. Uživatel začne psát do klíčového pole záznamu a píše dokud program nevyhledá odpovídající záznam. Naším dalším krokem bude přidání podobného rysu do formuláře, který jsme právě vytvořili pro výběr. K tomu trochu teorie.

11.2.1. Ascii (znak) -> Číslo

Příkaz ASCII vrátí Ascii kód zadaného znaku.

Jestliže je v předávaném řetězci více než jeden znak, vrátí funkce kód pro první znak.

Funkce Char je opakem funkce Ascii. Tato funkce navrací znak na zadaný kód Ascii.

Při porovnávání ve 4D jsou velká a malá písmena považována za shodná. Proto pokud potřebujete rozlišit mezi malými a velkými písmeny, můžete použít funkci Ascii k testování zda je znak malé či velké písmeno.

My budeme používat funkce Ascii pro testování úhozů na speciální klávesy jako jsou klávesy šipek či klávesa Delete..

11.2.2. Keystroke -> znak

Funkce Keystroke navrací znak vložený uživatelem přes klávesnici do pole nebo jiné dostupné oblasti. Obvykle budete volat Keystroke uvnitř metody formuláře či objektu při událostech Před úhozem na klávesu nebo Po úhozu na klávesu. K rozeznání těchto událostí používejte příkaz Form event.

DŮLEŽITÁ POZNÁMKA: Jestliže chcete provádět určité operace „za běhu“ závislé na momentální hodnotě, právě upravované dostupné oblasti a rovněž zachytit nově vložený znak, musíte si pamatovat, že text který vidíte na obrazovce není ještě hodnota v zdrojovém poli či proměnné pro tuto oblast. Vložená hodnota bude datovému zdroji (poli či proměnné) přiřazena při potvrzení dané oblasti (přechodu na jinou oblast, klepnutí na tlačítko atd.- ve fázi Při aktualizaci). Je to podobné jak když simulujete vkládání do pole pomocí proměnné a do příslušného pole ukládáte až zcela prověřenou hodnotu proměnné.

Příkaz Keystroke budete používat pro:

- Filtrování znaků svým vlastním způsobem
- Filtrování vstupu dat způsobem, který nemůžete zajistit např. použitím vstupních filtrů pro data
- Zavedení rysů doplnění uživatelských vstupů při psaní z klávesnice

11.2.3. FILTER KEYSTROKE (filtrovaný znak)

Příkaz FILTER KEYSTROKE vám dovolí nahradit znak napsaný uživatelem do pole či dostupné oblasti prvním znakem řetězce filtrovaný znak, který jste předali.

Jestliže předáte prázdný řetězec je úhoz na klávesnici zrušen a ignorován.

Obvykle budete volat filtr FILTER KEYSTROKE uvnitř metody formuláře či objektu při ovládání událostí Před úhozem na klávesu a Po úhozu na klávesu. K získání skutečného úhozu na klávesu použijte příkaz KEYSTROKE.

Příkaz FILTER KEYSTROKE budete používat pro:

- Filtrování znaků svým vlastním způsobem
- Filtrování vstupu dat způsobem, který nemůžete zajistit např. použitím vstupních filtrů pro data
- Zavedení rysů doplnění uživatelských vstupů při psaní z klávesnice

UPOZORNĚNÍ: Jestliže voláte příkaz KEYSTROKE po zavolání FILTER KEYSTROKE , je znak, který předáváte tomuto příkazu ten, který byl již odfiltrován místo toho, který byl skutečně z klávesnice zadán.

11.2.4. GET HIGHLIGHT (textObjekt; první; poslední)

Parametr	Typ	Popis
textObjekt	text/řetězec	argument textového objektu či proměnné
první	proměnná	první pozice vybrání
poslední	proměnná	poslední pozice vybrání

Příkaz GET HIGHLIGHT je používán k nalezení toho jaký text je právě vybrán (vysvícen) Text může být vybrán buď uživatelem nebo pomocí příkazu HIGHLIGHT TEXT. Tento příkaz nemůže být použit s poli v oblasti podformulářů.

Proměnná První vrací pozici prvního vybraného znaku a proměnná Poslední vrací pozici posledního vybraného znaku plus 1. Jestliže jsou proměnné první a poslední rovny, uživatel nevybral žádný text a kurzor je před znakem určeným proměnnou první.

11.2.5. Type (pole/proměnná) -> číslo

Funkce Type vrací číslo typu dat pole nebo proměnné. Type je obvykle používána k testování zda je parametr pro prováděnou operaci správného typu.

My budeme tento příkaz používat k větvení kódu pro lepší práci s proměnnými řetězce nebo textu.

11.2.6. Přidání rysu jasnozřivosti

1. Vytvořte novou metodu projektu KEY_tDeleteText následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: KEY_tDeleteText (Text; Long; Long) -> Text
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vymaže vysvícenou oblast textu

   <>fGeneric:= True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definice parametrů:
C_TEXT($0)              ` Konečný text po vymazání
C_TEXT($1;$tOriginalText)       ` Originální text
C_LONGINT($2;$LStart)           ` Počáteční pozice vymazání
C_LONGINT($3; $LEnd)                ` Konečná pozice vymazání

      ` Přiřazení parametrů do lokálních proměnných
$tOriginalText:=$1
$LStart:=$2
$LEnd:=$3

$0:=Substring($tOriginalText ;1;$LStart-1-Num($LStart = $LEnd))+
Substring($tOriginalText ;$LEnd)
  `Konec metody
```

2. Vytvořte novou metodu projektu KEY_sDeleteSubstring následovně nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: KEY_sDeleteSubstring (String; Long; Long) -> String
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vymaže vysvícenou oblast řetězce

   <>fGeneric:= True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definice paramterů
C_STRING(255;$ 0)                        ` Konečný řetězec po vymazání
C_STRING(255;$1;$sOriginalString)    ` Originální řetězec
C_LONGINT($2;$LStart)                ` Počáteční pozice vymazání
C_LONGINT($3; $LEnd)                     ` Konečná pozice vymazání

      ` Přiřazení parametrů do lokálních proměnných
$sOriginalString:= $1
$LStart:=$2
$LEnd:=$3

$0:=Substring($sOriginalString;1;$LStart-1-Num($LStart = $LEnd))+
Substring($sOriginalString;$LEnd)
  `Konec metody
```

3. Vytvořte novou metodu projektu KEY_tInsertText následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: KEY_tInsertText (Text; Long; Long; Text) -> Text
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vloží text do textové oblasti

   <>fGeneric:= True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True
```

```
End if

      ` Definice paramterů:
C_TEXT($0)              ` Konečný text po vložení
C_TEXT($1;$tOriginalText)          ` Originální text
C_LONGINT($2;$LStart)              ` Počáteční pozice vymazání
C_LONGINT($3; $LEnd)                    ` Konečná  pozice vymazání
C_TEXT($4;$tInsertText)                                    ` Text pro vložení

      ` Přiřazení parametrů do lokálních proměnných
$tOriginalText:=$1
$LStart:=$2
$LEnd:=$3
$tInsertText:=$4

If ($LStart # $LEnd)
   $tOriginalText:=Substring($tOriginalText;1;$LStart-1) + $tInsertText +
Substring($tOriginalText;$LEnd)
Else
   Case of
      ś ($LStart <= 1)
         $tOriginalText:=$tInsertText + $tOriginalText
      ś ($LStart > Length($tOriginalText))
         $tOriginalText:=$tOriginalText+$tInsertText
      Else
         $tOriginalText:=Substring($tOriginalText;1;$LStart-1) + $tInsertText +
Substring($tOriginalText;$LStart)
   End case
End if
$0:=$tOriginalText
 `Konec metody
```

4. Vytvořte novou metodu projektu KEY_sInsertSubstring následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: KEY_sInsertSubstring (String; Long; Long; String) -> String
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vloží řetězec do řetězce

   <>fGeneric:= True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True
```

```
End if

      ` Definice parametrů:
C_STRING(255;$0)                        ` Konečný řetězec po vložení
C_STRING(255;$1;$sOriginalString)    ` Originální řetězec
C_LONGINT($2;$LStart)                ` Počáteční pozice vymazání
C_LONGINT($3; $LEnd)                    ` Konečná pozice vymazání
C_STRING(255;$4;$sInsertString)      ` Řetězec pro vložení

      ` Přiřazení parametrů do lokálních proměnných
$sOriginalString:= $1
$LStart:=$2
$LEnd:=$3
$sInsertString:=$4

If ($LStart # $LEnd)
   $sOriginalString:=Substring($sOriginalString;1;$LStart-1) + $sInsertString  +
Substring($sOriginalString;$LEnd)
Else
   Case of
      ś ($LStart <= 1)
         $sOriginalString:=$sInsertString  + $sOriginalString
      ś ($LStart > Length($sOriginalString))
     $sOriginalString:=$sOriginalString+$sInsertString
      Else
         $sOriginalString:=Substring($sOriginalString;1;$LStart-1) + $sInsertString  +
Substring($sOriginalString;$LStart)
   End case
End if
 $0:=$sOriginalString
 `Konec metody
```

5. Vytvořte novou metodu projektu KEY_sShadowKeystroke následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: KEY_sShadowKeystroke ( Pointer ; Pointer ; String ) -> String
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Udržuje duplikovanou oblast textu či řetězce při rozeznání úhozu

   <>fGeneric:= True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if
```

```
      ` Definice parametrů:
C_STRING(1;$0)                        ` Navracený úhoz
C_POINTER($1)                  ` Ukazatel na zdrojový text/řetězec
C_POINTER($2)                  ` Ukazatel na duplikovaný text/řetězec
C_STRING(255;$3;$sKeystrokeFilter)  ` Filtr úhozu

  ` Definice lokálních proměnných
C_STRING(255;$sRevisedString)
C_TEXT($tRevisedText)
C_LONGINT($LType)                     ` Typ proměnné
C_LONGINT($LStart)                    ` Počáteční pozice
C_LONGINT($LEnd)               ` Konečná pozice

      ` Přiřazení parametrů do lokálních proměnných
$sKeystrokeFilter:=$3

      ` Nalezne typ proměnné pro větvení
$LType:=Type($1->)

      ` Vrátí původní úhoz
$0:=Keystroke

      ` Určí vybranou část v dostupné oblasti
GET HIGHLIGHT($1->;$LStart;$LEnd)

      ` Začátek práce s platnou hodnotou
If ($LType = Is Text)
   $tRevisedText:=$2->
Else
   $sRevisedString:=$2->
End if

      ` V závislosti na klepnuté klávese či vloženém znaku,
      ` Proveď patřičnou akci
Case of
            ` Byla stisknuta klávesa  Backspace (Delete)
   ś (Ascii($0)=Backspace)
            ` Vymazat vybrané znaky nebo znak vlevo od kursoru
      If ($LType=Is Text)
         $tRevisedText:=KEY_tDeleteText ($tRevisedText;$LStart;$LEnd)
      Else
         $sRevisedString:=KEY_sDeleteSubstring ($sRevisedString;$LStart;$LEnd)
      End if

            ` Byla stisknuta klávesa šipky
            ` Nedělej nic, ale přijmi úhoz
   ś (Ascii($0) = Left Arrow Key)
   ś (Ascii($0) = Right Arrow Key)
```

```
ś (Ascii($0) = Up Arrow Key)
ś (Ascii($0) = Down Arrow Key)

      ` Byl vložen přijatelný znak
ś (Position($0;$sKeystrokeFilter) = 0)
  If ($LType=Is Text)
    $tRevisedText:=KEY_tInsertText ($tRevisedText;$LStart;$LEnd;$0)
  Else
    $sRevisedString:=KEY_sInsertSubstring ($sRevisedString;$LStart;$LEnd;$0)
  End if

Else
      ` Znak je nepřijatelný
  FILTER KEYSTROKE("")
End case

    ` Navrácení hodnoty pro ošetření dalšího úhozu
If ($LType=Is Text)
  $2->:=$tRevisedText
Else
  $2->:=$sRevisedString
End if
 `Konec metody
```

6. Zkontrolujte ve formuláři dostupnou oblast CHO_sSelectedItem.

Typ objektu	Název objektu	Vlastnosti	Události
Dostupná oblast	CHO_sSelectedItem	Styl: Pole vstupní Ponořené Zvětšit podélně Posunout svisle	Po úhozu na klávesu

7. Nastavte pořadí vstupu tak, že objekt CHO_sSelectedItem bude první v pořadí vstupu. Je to nezbytné proto, že by tlačítka měly být vždy na konci pořadí.

8. Vytvořte novou metodu objektu CHO_sSelectedItem následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
    ` Metoda objektu: CHO_sSelectedItem [zDialogy];"CHO_ChoiceListDialog"
    ` Kurz programovani ACI University
    ` Genericke metody z nastroju ACI
```

```
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Oblast pro vkládání textu pro jasnozřivost výběru

   <>fGeneric:= True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definice lokálních proměnných:
C_LONGINT($LAscii)                  ` Ascii kód
C_LONGINT($LFound)                  ` První nalezený prvek array
C_BOOLEAN($fOK)               ` Příznak že vše je OK
C_STRING(255;$sQuery)         ` Řetězec dotazu
C_STRING(1;$sKeystroke)       ` Úhoz

$sKeystroke:=KEY_sShadowKeystroke (->CHO_sSelectedItem;-
>CHO_sQueryString;"")
$fOK:=True

Case of
   ś (Ascii($sKeystroke) = Down Arrow Key)
      If (CHO_asChoice1 < Size of array(CHO_asChoice1))
         CHO_asChoice1:=CHO_asChoice1+1
      End if
      $fOK:=False

   ś (Ascii($sKeystroke)=Up Arrow Key)
      If (CHO_asChoice1>1)
         CHO_asChoice1:=CHO_asChoice1-1
      End if
      $fOK:=False
End case

If ($fOK)
   $sQuery:=CHO_sQueryString+"@"

   $LFound:=Find in array(CHO_asChoice2;$sQuery)
   If ($LFound > 0)
      CHO_asChoice2:=$LFound
   End if
End if
   ` Konec metody
```

9. Zkontrolujte vlastnosti formuláře [zDialogy]CHO_ChoiceListDialog a ujistěte se, že jediná vybraná událost je Při zavedení.

10. Vytvořte novou metodu formuláře pro [zDialogy]CHO_ChoiceListDialog následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda formuláře: [zDialogy];"CHO_ChoiceListDialog"
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vymaže použité proměnné úhozu

   <>fGeneric:= True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

C_LONGINT($LFormEvent)

$LFormEvent:=Form event

Case of
   :($LFormEvent = On Load)
      CHO_sSelectedItem:=""
      CHO_sQueryString:=""
      CHO_asChoice1:=1

End case
      `Konec metody
```

11. Upravte metodu objektu pro CHO_Lookup následovně:

```
If (False)
      ` Metoda: CHO_Lookup(pointer)
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Ošetřuje výběr v array

   <>fGeneric:= True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if
```

```
` Definovat parametry
C_POINTER($1)` Ukazatel na vybrané array

` Deklarace lokalnich promennych
C_LONGINT($LElementChosen)

$LElementChosen:=$1->
If($LElementChosen > 0)
   CHO_sSelectedItem:=CHO_asChoice2{$LElementChosen }
   CHO_sQueryString:=CHO_sSelectedItem
   If (Form event = On Double Clicked)
      ACCEPT
   End if
End if
`Konec metody
```

❖

12. Jděte do prostředí Vlastní nabídky a testujte změny.

12. Vytvoření vlastního okna výběru

Jestliže použijete vestavěné výběrové seznamy 4D, platí následující tvrzení:

- Tyto seznamy fungují automaticky při vstupu do dotčené oblasti.
- Otevře se malé okno ve vrchní části obrazovky.
- Jste omezeni na jeden konkrétní hierarchický seznam.

Jedna z nejčastějších otázek je jak mohu změnit před zobrazením seznamu jednu či dvě položky aniž bych ztratil ostatní. Ve standartních seznamech 4D nemáte příliš na výběr, ale díky události formuláře Při získání fokusu jste schopni udělat téměř cokoliv. Jedinou vyjímkou je Editor dotazů.

Protože již máme vyzkoušené a fungující pole array ve skupině bude úprava existujícího kódu poměrně malá..

12.0.1. Vytvoření okna s vlastním výběrovým seznamem

1. Jestliže chcete zachovat svůj existující kód pro objekt sCompanyName, vytvořte novou metodu projektu a zkopírujte do ní kód z této metody.
2. Upravte vlastnosti objektu sCompanyName tak, aby reagoval pouze na událost Při získání fokusu a zobrazte si všechny dostupné záznamy zákazníků.

12.0.2. Přidání více záznamů zákazníků

Abychom vyzkoušeli jak tento nový rys pracuje, potřebujeme ve fázi zavádění více záznamů zákazníků.

1. Vytvořte novou metodu projektu E_Set2ReadWrite následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
    ` Metoda: E_Set2ReadWrite
    ` Kurz programovani ACI University
    ` Generické metody z nastroju ACI
    ` Datum: 3/17/97
    ` By: Jim Steinman

    ` Účel: Nastaví tabulky do stavu číst psát pro údržbu v Prostředí uživatele

<>fGeneric:= True
<>f_Version6x30:=True
<>fJ_Steinman:= True
```

```
End if

READ WRITE(*)
    `Konec metody
```

2. Nahraďte metodu objektu sCompanyName na formuláři [Faktury]Vstupní textovým souborem sCompanyName Verze2 :

```
If (False)
      ` Metoda: sCompanyName
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Zobrazí záznamy [Zákazníci] ve vlastním výběrovém seznamu

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

ALL RECORDS([Zákazníci])

CHO_FillCustomerArray
CHO_DisplayChoiceList

If (OK=1)
   [Faktury]IDZákazníka:=CHO_asChoice1{CHO_asChoice1}
   RELATE ONE([Faktury]IDZákazníka)
   sCompanyName:=[Zákazníci]Firma
End if

ARRAY STRING(11;CHO_asChoice1;0)
ARRAY STRING(31;CHO_asChoice2;0)
ARRAY STRING(33;CHO_asChoice3;0)
ARRAY STRING(25;CHO_asChoice4;0)

POST KEY(Tab;0)  `Tab to the next field
  `Konec metody
```

3. Přejděte do Prostředí uživatele a proveďte metodu E_Set2ReadWrite.
4. Přepněte se do tabulky [Zákazníci] a naimportujte soubor Customer Import Data.
5. Přepněte se do tabulky [zNavracenáČísla] a vymažte všechny záznamy začínající IDZákazníka.
6. Přepněte se do tabulky [zSekvence] a upravte hodnotu IDZákazníka na 1203.
7. Přepněte se do prostředí Vlastní nabídky a zkuste přidat nový záznam faktury.

12.1. Přednastavení nejčastěji používaných array

To bylo poněkud pomalé. Jak můžete urychlit tento proces?

Můžeme dopředu vytvořit meziprocessní array v paměti a pak je pouze zkopírovat do zobrazovacích array pokaždé, když je budeme potřebovat použít. Protože použijeme meziprocessní array, budou tyto dostupné libovolnému procesu. Nakonec přidáme několik řádek kódu do metody Při spuštění a vytvoříme je již na počátku používání databáze.

12.1.1 Naplnění array do vlastního výběrového seznamu z meziprocessních array v paměti

1. Vytvořte novou metodu projektu CHO_InitializeCustomerIPArrays následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: CHO_InitializeCustomerIPArrays
      ` Kurz programovani ACI University
      ` Generic metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Naplní meziprocessní array

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definice lokálních proměnných:
C_LONGINT($CHO_LNumberElements)          ` Počet záznamů v array
C_LONGINT($i)                            ` Čítač smyčky

      ` Array pro tabulku Zákazníci
ARRAY STRING(11;<>CHO_asCustomerID;0)
ARRAY STRING(31;<>CHO_asCompany;0)
ARRAY STRING(10;$CHO_asFirstName;0)
ARRAY STRING(20;$CHO_asLastName;0)
ARRAY STRING(10;$CHO_asPhone;0)

ALL RECORDS([Zákazníci])

SELECTION TO
ARRAY([Zákazníci]IDZákazníka;<>CHO_asCustomerID;[Zákazníci]Firma;
<>CHO_asCompany;[Zákazníci]Jméno;$CHO_asFirstName;[Zákazníci]Příjmení;
$CHO_asLastName;[Zákazníci]Telefon;$CHO_asPhone)
```

```
REDUCE SELECTION([Zákazníci];0)


$CHO_LNumberElements:=Size of array(<>CHO_asCustomerID)
ARRAY STRING(33;<>CHO_asContactName;$CHO_LNumberElements)
ARRAY STRING(25;<>CHO_asPhone;$CHO_LNumberElements)

For ($i;1;$CHO_LNumberElements)
   <>CHO_asContactName{$i}:=$CHO_asFirstName{$i}+" "+$CHO_asLastName{$i}
   <>CHO_asPhone{$i}:=String(Num($CHO_asPhone{$i});"|FormátTelefon")
End for

SORT
ARRAY(<>CHO_asCompany;<>CHO_asCustomerID;<>CHO_asContactName;<>CHO
_asPhone)
      `Konec metody
```

2. Do metody projektu MyStartup přidejte následující řádku kódu:

```
❖                $Lpid:=PROCESS_LSpawnProcess
("CHO_InitializeCustomerIPArrays";32;"Initialize";True;False)
```

3. Nahraďte metodu projektu CHO_FillCustomerArray následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: CHO_FillCustomerArrays
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Naplní všechna array výběrového seznamu

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

COPY ARRAY(<>CHO_asCustomerID;CHO_asChoice1)
COPY ARRAY(<>CHO_asCompany;CHO_asChoice2)
COPY ARRAY(<>CHO_asContactName;CHO_asChoice3)
COPY ARRAY(<>CHO_asPhone;CHO_asChoice4)
      `Konec metody
```

4. Upravte metodu objektu sCompanyName z formuláře [Faktury]Vstupní následovně:

```
If (False)
      ` Metoda: sCompanyName
```

```
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Zobrazí záznamy [Zákazníci] ve vlastním výběrovém seznamu

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

❖  ~~ALL RECORDS([Zákazníci])~~
❖
CHO_FillCustomerArray
CHO_DisplayChoiceList

If (OK=1)
   [Faktury]IDZákazníka:=CHO_asChoice1{CHO_asChoice1}
   RELATE ONE([Faktury]IDZákazníka)
   sCompanyName:=[Zákazníci]Firma
End if

ARRAY STRING(11;CHO_asChoice1;0)
ARRAY STRING(31;CHO_asChoice2;0)
ARRAY STRING(33;CHO_asChoice3;0)
ARRAY STRING(25;CHO_asChoice4;0)

POST KEY(Tab;0)  `Tab to the next field
  `Konec metody
```

5. Restartujte databázi a proveďte testování změn.

12.2. Otevírání dialogů někdy způsobuje problémy s obnovou obrazovky

Přerušením normálního běhu 4D si můžete někdy způsobit problémy s překreslením obrazovky. Problém jednoduše vyřešíte voláním příkazu REDRAW WINDOW, který způsobí překreslení obrazovky.

12.2.1. Vyřešení problému překreslení obrazovky

1. Upravte metodu objektu sCompanyName následovně:

```
…
REDRAW WINDOW
POST KEY(Tab;0)  `Tabelátorem na další pole
  `Konec metody
```

2. Přejděte do prostředí Vlastní nabídky a proveďte testování.

12.3. Udržování meziprocessních array pomocí triggerů

Nyní máme problém, že se naše data zákazníků neustále mění. Přidáme zákazníka, změníme kontaktní osobu, změníme telefon, vymažeme zákazníka atd. V tomto případě pokud neprovedeme další úpravy bude stav přednaplněných meziprocessních array neaktuální. Protože však máme možnost programovat triggery, problém lze vyřešit.
V triggeru můžeme:

- Přidat nový prvek do MP array, když přidáme nový záznam
- Vymazat prvek z MP array, když záznam vymažeme
- Upravit prvek v MP array jestliže se změní data

12.3.1. Old(pole) -> Stará hodnota pole

Pro platný záznam vrací funkce Old hodnotu pole před zavedením záznamu tj. před jeho úpravami. Jinými slovy vrátí hodnotu pole, která je uložena na disku. Funkce Old pracuje pro pole, která byla změněna metodou nebo akcí uživatele.

Jestliže je záznam nový, navrací Old prázdný řetězec či prázdnou hodnotu pro pole. Např. jestliže je záznam nový a pole je Alfa, Old navrátí prázdný řetězec, jestliže je pole číselné, Old navrací 0. Jestliže je pole Datum Old navrací !00/00/00!. Jestliže je pole typu čas navrací †00:00:00†. Jestliže je pole Logické Old navrací False.

Old nemůže být aplikována na pole typu text, obrázek nebo BLOB. Může být použit na všechny ostatní typy polí včetně podpolí, ale nemá smysl jestliže je použit na celou podtabulku.

Vytvoříme trigger, který bude udržovat MP array aktuální. Protože jsou data dobrá pouze tehdy jestliže jsou aktuální můžeme použít trigger v každé události při úpravě, přidávání a mazání. .

Kvíz — Obnova array

- Co je rychlejší úprava prvku array nebo kontrola zda se změnila hodnota pole pomocí Old?
- Kdy potřebujeme třídit array?
- Jak můžete zajistit, aby změna array probíhla pouze v jednom procesu současně?

- Může být funkce Old použita v triggeru, který je prováděn na serveru?

12.4. Vytvoření triggeru k obnově MP array

Potřebujeme vytvořit trigger pro tabulku [Zákazníci], který zajistí synchronizaci MP array s aktuálními daty. Tato array potřebují být změněna, když jsou data ukládána nebo mazána. A navíc, protože máme víceprocesovou databázi array mohou být upravovány pouze jedním procesem současně.

Kde jsou spouštěny triggery? Na klientu či na serveru?

Triggery běží na serveru. Protože triggery běží na serveru počkáme s úpravami až na diskusi pro uložené procedury a teprve potom vytvoříme kód, který uskuteční ovládání obnovy MP array. A navíc jestliže triggery běží na server a MP array žije na serveru jak může stroj klienta získat tuto obnovu.

Jestliže chcete, aby databáze pracovala se 4D Server nemůžete použít funkci Old, protože funkce Old nepracuje na 4D Server, ale pouze v jednouživatelské verzi či klientu. Takže potřebujeme ještě využít znalostí z dalších kapitol

12.5. Vyloučení automatického výběru s náhradou znamená, že můžete vlastní výběr použít i pro číselná pole

Automatický výběr s náhradou vyžaduje pro vztahy pouze pole Alfa. S mechanismem, který jsme právě vytvořili nebudeme již omezeni pouze na pole Alfa.

14. BLOBy

4th Dimension v6 zavedla datový typ BLOB (Binary Large OBjects). Můžete definovat pole BLOB a proměnné BLOB. Pro BLOB neexistují array.

Uvnitř 4th Dimension je BLOB nepřerušenou řadou bytů s různou délkou této řady a může být ovládán jako jeden objekt jehož jednotlivé byty mohou být adresovány odděleně. BLOB může být prázdný (nulové délky) nebo může obsahovat do 2 147 483 647 bytů (2 GB).

Na BLOB nemohou být použity žádné operátory tj. neexistují žádné výrazy typu BLOB.

14.1. BLOBy a paměť

BLOB zaváděn do paměti jako celek. Proměnné typu BLOB jsou udržovány a existují pouze v paměti. Pole BLOB je zaváděno do paměti z disku tak jako zbytek záznamu, ke kterému patří. Tak jako jiné typy polí může obsahovat velké množství dat (typ obrázek a podtabulka). Pole BLOB nejsou při úpravách záznamu v paměti duplikována, proto výsledek navracený příkazy Old a Modified nemá význam pokud je použijeme na pole BLOB.

Proměnné typu BLOB uzamykají paměť, když jsou vytvářeny, aby ochránily tuto oblast paměti. Když skončíte práci s proměnnou BLOB měli by jste vždy změni velikost BLOB na 0 bytů pomocí příkazu SET BLOB SIZE (myBlob;0).

UPOZORNĚNÍ: Jestliže nemáte dost paměti pro provedení operace s BLOB může váš počítač často krachovat.

Protože BLOB je nepřerušená část paměti nástroje Windows musí být váš paměťový blok nastaven jako větší než váš největší BLOB.

14.2. Zobrazení BLOBů

BLOB může obdržet libovolný typ dat takže na obrazovku nemá žádné výchozí zobrazení. Jestliže na formuláři zobrazujete přímo pole BLOB nebo proměnou, objeví se tato oblast jako prázdná ať již BLOB obsahuje cokoliv.

14.3. Pole BLOB

Pole BLOB můžete použít pro uložení libovolného druhu dat do velikost 2 GB. Pole BLOB nelze indexovat takže v případě vyhledávání záznamů podle obsahu BLOB musíte použít výraz. Nepoužívejte pole BLOB k ukládání dat, která chcete vybavovat velmi

rychle pomocí operací dotazů. Např. neukládejte do pole BLOB klíčová slova, místo toho pro klíčová slova raději použijte podsoubor, který můžete indexovat pro podpole.

14.4. Předávání parametrů, Ukazatele a Výsledky funkcí

BLOBy 4th Dimension mohou být předávány jako parametr do příkazů 4D nebo ruti 4D Extension, které očekávají parametry typu BLOB. Na druhé straně nemohou být předávány jako parametry do vlastních metod. Rovněž BLOB nemůže být navrácen jako výsledek funkce.

K předání BLOB do vaší vlastní metody definujte ukazatel na BLOB a předejte jej jako parametr.

BLOBy mohou být předávány do procesů v době jejich spouštění, ale neměli by jste pro tento účel vytvářet ukazatele. Ukazatele nemohou být předávány jako parametry do jiných procesů.

14.5. Adresování obsahu BLOB

Každý byte BLOBu můžete adresovat individuálně pomocí složených {...}. Uvnitř BLOB jsou byty od 0 do N-1, kde N je velikost BLOB. Příklad:

Protože můžete adresovat všechny byty BLOB individuálně, můžete skutečně do BLOB ukládat cokoliv co chcete uložit v proměnné či poli typu BLOB.

14.6. VARIABLE TO BLOB (proměnná; blob{; posun | *})

Příkaz VARIABLE TO BLOB uloží proměnnou do BLBO s názvem blob.

Jestliže určíte volitelný parametr * je proměnná přidána na konec BLOB a velikost BLOB se úměrně zvětší. S použitím volitelného parametru *. Můžete postupně ukládat libovolný počet proměnných či seznamu do BLOB dokud se BLOB vejde do paměti.

Jestliže neurčíte volitelný parametr * nebo parametr posun, bude proměnná uložena na počátek BLOB a přepíše jakýkoliv předchozí obsah. Velikost BLOB se upraví podle výsledku zápisu.

Jestliže předáte parametr posun bude proměnná do BLOB zapsána od bytu jehož číslo jste předali parametrem posun (začínáme od nuly). Nezáleží na tom kam proměnnou zapíšete, velikost BLOB se zvýší podle místa kam proměnnou zapisujete (existující byty jsou přepsány a je-li potřeba jsou přidány nové). Pokud jsou alokovány nové byty např. před počátek nově vkládané proměnné jsou inicializovány na 0.

Po provedení příkazu parametr posun navrací hodnotu. Tato hodnota je počáteční posun zvětšený o počet bytů, které byly zapsány. Proto proměnnou posun můžete okamžitě použít pro další zápis do téhož BLOB.

VARIABLE TO BLOB přijme libovolný typ proměnné (včetně jiného BLOB) vyjma následujících:

- Ukazatele
- Array ukazatelů
- Dvoudimenzionální array
- Long Integer odkazů (např. na okna a hierarchické seznamy)

UPOZORNĚNÍ: Jestliže používáte pro ukládání proměnných BLOB musíte později použít příkaz BLOB TO VARIABLE pro zpětné obnovení proměnných z obsahu BLOB. Jiný způsob obnovení proměnných z BLOB nedoporučujeme, protože proměnné jsou v BLOB uloženy v interním formátu 4D.

Po provedení příkazu jestliže byla proměnná úspěšně napsána je nastavena systémová proměnná OK na 1. Jestliže operace nemohla být provedena je proměnná OK nastavena na 0 (např. pokud nebyl dostatek paměti).

14.7. BLOB TO VARIABLE (blob; proměnná{; posun})

Příkaz BLOB TO VARIABLE přepíše obsah proměnné daty uloženými uvnitř BLOB začínaje na bytu posun (čísluje se od nuly).

Data v BLOB musí odpovídat cílové proměnné. V typickém případu se používá BLOB, který byl v předchozím naplněn pomocí příkazu VARIABLE TO BLOB.

Jestliže neurčíte volitelný parametr posun jsou data do proměnné čtena od počátku BLOB. Jestliže zacházíte s BLOB do kterého bylo zapsáno několik proměnných musíte povinně určit posun pro proměnné zapsané doprostřed BLOB. Pokud si chcete nebo musíte pamatovat počáteční pozici první zapsané proměnné použijte pro parametr posun jinou proměnnou, protože po přečtení z BLOB je parametr posun zvětšen o počet přečtených bytů. Lze jej tedy s výhodou použít pro čtení další proměnné zapsané v BLOB.

Po provedení příkazu jestliže byla proměnná úspěšně přečtena a přepsána obsahem BLOB nastaví se systémová proměnná na 1. Jestliže operace nemohla být provedena, nastaví se proměnná OK na 0.

14.8. SET BLOB SIZE (blob; velikost{; výplň})

Příkaz SET BLOB SIZE změní velikost BLOB na hodnotu předanou parametrem velikost.

Pokud neurčíte volitelný parametr výplň, budou nově alokované byty (pokud existují) inicializovány na 0 (0x00). Jestliže chcete tyto byty inicializovat na jinou hodnotu, předejte v parametru výplň hodnotu 0…255 (0x00…0xFF).

Když jste skončili práci s BLOB je dobré uvolnit zabranou paměť. Toto provedete nastavením velikosti BLOB na nula.

14.9. Vyzkoušení BLOBů předávaných do jiných procesů

14.9.1. Export záznamů

1. Podívejte se na existující kód pro export záznamů.

   IMPEXP_ImportExportDialog
   IMPEXP_ExportData

15. Uložené procedury

15.1. Přehled o uložených procedurách

Uložené procedury pro 4D Server jsou metody projektu ve 4D, které jsou vývojářem speciálně určeny pro provádění na stroji serveru a nikoliv na stroji klienta.

4D Server podobně jako 4th Dimension a 4D Client je schopný spustit souběžně několik procesů díky schopnostem plánování a dělení času.

4D Server kromě uložených procedur má následující architekturu:

- Procesy jádra spouštějí různé služby databázového stroje
- Uživatelské procesy spouštějí úlohy připojených 4D Client a aplikací založených na 4D Open
- Procesy uložených procedur spouštějí vývojářem definované metody projektu 4D na počítači serveru

Procesy uložených procedur sdílejí následující prostředí:

- Procedura databáze Při spuštění serveru může být implementována vývojářem databáze.
- Příkazy ovládání procesů jako je New process jsou prováděny uvnitř 4D Server: volání New process z uložené procedury vytvoří jinou uloženou proceduru.
- Příkazy procesu spojené s ovládáním rozhraní jako je BRING TO FRONT nemají žádný význam na server, protože procesy uložených procedur nemají uživatelské rozhraní. Je možné otevřít na serveru okno ke zobrazení informace, ale toto okno je samozřejmě trochu neoperativní.
- Procesy uložených procedur sdílejí tutéž tabulku meziprocessních proměnných, tato tabulka je identická s tabulkou definovanou na 4D Client, ale na stroji serveru má každá meziprocessní proměnná jinou hodnotu zrovna tak jako má jinou hodnotu na každém 4D Client.
- Procesy uložených procedur sdílejí uvnitř 4D Server obsah tj. tytéž meziprocessní sady a pojmenované výběry. Dosah meziprocessních sad a pojmenovaných výběrů je počítač serveru.

- Lokální semafory použité v uložených procedurách jsou lokální na stroji 4D Server.

- Mechanismus CALL PROCESS a Outside call nemá na počítači serveru žádný význam, protože uložené procedury nemají uživatelské rozhraní.

15.2. Výměna dat mezi procesy klienta a serveru

Do této doby jsme se zabývali komunikací mezi procesy na počítačích klienta případně jednouživatelské 4D.

Komunikace mezi procesy může být zavedena s použitím následujících objektů a příkazů prostředí 4D:

- Příkazy ovládání procesů jako New process

- Příkazy ovládání rozhraní procesů jako je BRING TO FRONT

- Záznamy(*)

- Meziprocessní proměnné, sady a pojmenované výběry

- Lokální semafory

- Globální semafory (*)

- Mechanismy volání CALL PROCESS a Outside call

- Metoda databáze Při spuštění je obvykle použita pro nastavení struktury dat (tj. proměnných a array) používaných k zavedení mechanismů meziprocessní komunikace.

(*) S použitím těchto objektů byla komunikace mezi procesy možná již ve v3 a to jak procesy běžícími na tomtéž stroji tak procesy běžícími mezi libovolnými stroji systému 4D klient/server.

Se zavedením uložených procedur běžících na 4D Server se do naší architektury 4D klient/server přidávají další rysy:

- Komunikace mezi procesy, tak jak jsme ji probrali pro stanici 4D Client, je umožněna (s určitými omezeními) na stroji 4D Server.

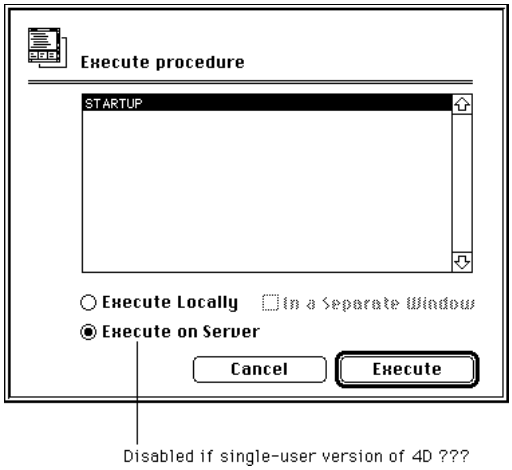
- Pro komunikaci mezi procesy 4D Server a 4D Client existují další mechanismy.

Těmito novými mechanismy jsou:

- Zasílání a přijímání proměnných z jiných procesů pomocí příkazů SET PROCESS VARIABLE a GET PROCESS VARIABLE.
- Zasílání a přijímání dokumentů na či ze stroje serveru pomocí příkazů SET DOCUMENT a GET DOCUMENT.
- Výměna sad a pojmenovaných výběrů.
- Předávání parametrů do procesů v době jejich spouštění.

15.3. Jak spustit uloženou proceduru?

15.3.1. Výběr voliče Provést na serveru v uživatelském prostředí pod položkou nabídky Provést metodu



15.3.2. Execute on server (Metoda ; VelikostStack { ; NázevProcesu { ; Parametr1 { ; Parametr2 ;... } } } { ; * }) -> ČísloProcesu

Příkaz Execute on server spustí nový proces na stroji serveru pod názvem *Název procesu* a navrátí odkaz na tento proces.

Metoda je název metody projektu, která bude na serveru spuštěna.

VelikostStack je množství paměti přidělené stack procesu. Tato část paměti je používána pro uložení objektů procesu jako volání metod, lokální proměnné, parametry k metodám, záznamy, speciálně uložené příkazy do stack a opětovně volané rutiny. Rozsah od 16K do 32K je dostatečný pro většinu procesů. Minimální velikost stack je 16K.

Voliteltný parametr *Název procesu* je název tohoto procesu jak se objeví v okně 4D Server v sekci Stored Procedures.

Důležitá poznámka: Jestliže chcete předat parametry do uložené procedury musíte povinně předat parametr *Název procesu*.

Parametry *Parametr 1……Parametr N* jsou parametry předávané uložené proceduře. Jestliže předáte parametry do uložené procedury jsou v této proceduře obdrženy jako $1,...,$N.

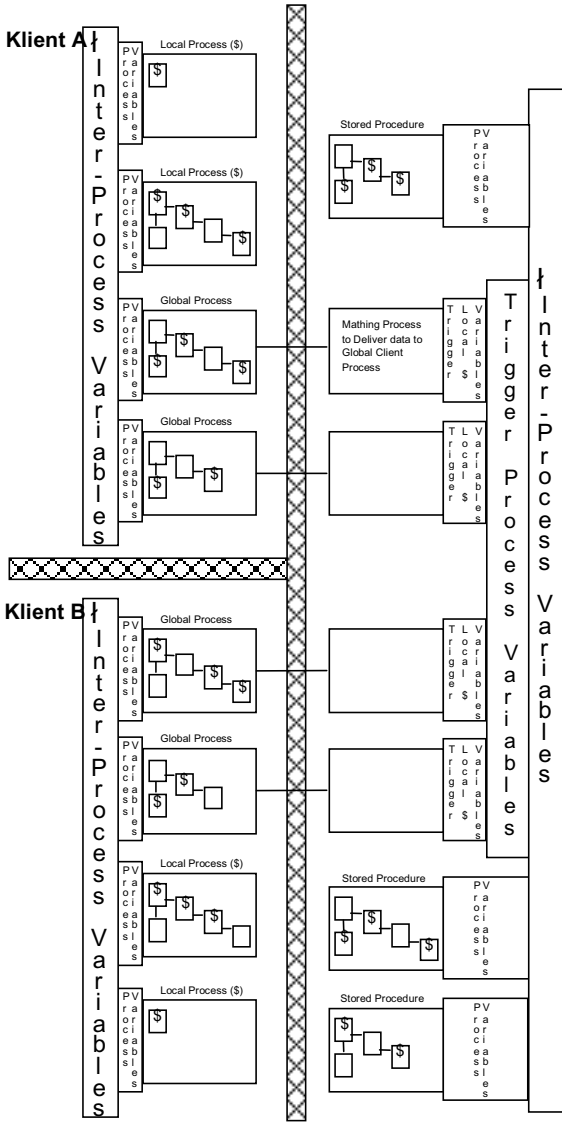
Volitelný parametr * vám dovolí určit jestli proces má být jedinečný. Jedinečnost procesu je založena na názvu procesu. Jestliže proces tohoto jména na serveru již běží vrátí příkaz Execute on server pouze číslo procesu běžící uložené procedury..

Poznámka: Aby bylo od sebe možno rozlišit procesy spuštěné na stroji klienta a serveru vrací příkazy Execute on server a New process spuštěný na serveru záporná čísla procesů.

15.4. Rozsah proměnných na serveru a klientu

Nyní by jste měli být již obeznámeni s rozsahem proměnných na klientu a jednouživatelské stanici. Rozsah proměnných na serveru je maličko jiný. Server má tabulku meziprocessních proměnných, která je sdílena všemi procesy na serveru. Každá uložená procedura má rovněž svou vlastní tabulku lokálních proměnných a proměnných procesu stejně jako klient. Rozdíl je v rozsahu proměnných používaných triggery. Ačkoliv triggery nejsou uložené procedury musíte být velice opatrní, protože triggery sdílejí tabulku proměnných procesu pro všechny klienty. Proto by jste neměli v triggerech používat proměnné procesu.

Víceuživatel
Proměnné a jejich rozsah



15.5. Komunikace mezi procesy

15.5.1. SET PROCESS VARIABLE (IDProcesu ; CílProm1 ; Výraz1 { ; CílProm2 ; Výraz2; ... })

Příkaz SET PROCESS VARIABLE změní hodnoty proměnných CílPromx v procesu určeném hodnotou IDProcesu na hodnoty určené ve VýrazX. Číslo IDProcesu musí být záporné číslo, když nastavujeme proměnné na serveru z procesu na klientu. IDProcesu musí být kladné číslo, když nastavujeme proměnné v jiném procesu na stejném stroji.CílPromX musí být proměnná procesu nebo meziprocessní proměnná.

Poznámka: Uložené procedury (procesy na Serveru) nemohou nastavovat proměnné v procesech klienta. Tato komunikace je jednosměrná.

15.5.2. GET PROCESS VARIABLE (IDProcesu ; ZdrojProm1 ; CílProm1 { ; ZdrojProm2 ; CílProm2 ; ... })

Příkaz GET PROCESS VARIABLE přečte hodnotu proměnných ZdrojPromX z určeného procesu v IDProcesu a nastaví hodnoty proměnných CílPromX ve volajícím procesu. IDProcesu musí být záporné číslo, když čteme proměnné z procesu na serveru. IDProcesu musí být kladné číslo, když čteme proměnné z jiného procesu na stejném stroji. ZdrojPromX musí být proměnné procesu nebo meziprocessní proměnné.

Poznámka: Uložené procedury (procesy na serveru) nemohou získávat proměnné z procesů klienta, je to komunikace jedním směrem.

15.5.3. VARIABLE TO VARIABLE (IDProcesu; CílProm1; ZdrojProm1 {; CílProm2; ZdrojProm2; ...; CílPromN; ZdrojPromN})

Příkaz VARIABLE TO VARIABLE zapíše do proměnných CílPromX v cílovém procesu určeném IDProcesu hodnoty z proměnných ZdrojPromX ve volajícím procesu.

VARIABLE TO VARIABLE provádí totéž jako příkaz SET PROCESS VARIABLE s následujícími odlišnostmi:

- Zatímco příkazem SET PROCESS VARIABLE předáváme výrazy (a proto nemůžeme předat array jako celek) do příkazu VARIABLE TO VARIABLE předáváme výhradně proměnné a proto můžeme předat array jako celek.
- V příkazu SET PROCESS VARIABLE musí být každá cílová proměnná proměnná nebo prvek array. V příkaze VARIABLE TO VARIABLE může být každá cílová proměnná, proměnná nebo prvek array nebo array.

Pro každý odpovídající pár CílProm, ZdrojProm nebo CílProm Výraz musí být typ cílové a zdrojové proměnné kompatibilní jinak se můžeme dočkat nesmyslnné hodnoty proměnné. V interpretovaném módu, jestliže cílová proměnná neexistuje, je vytvořena a je jí přiřazen typ a hodnota zdrojové proměnné.

Volající proces přiřazuje hodnoty do proměnných cílového procesu (cílový proces není žádným způsobem upozorňován, že jiný proces přepisuje jeho hodnoty).

Omezení

VARIABLE TO VARIABLE nepřijímá lokální proměnné jako cílové proměnné.

VARIABLE TO VARIABLE jakýkoliv typ cílového procesu nebo meziprocessní proměnné kromě:

- Ukazatelů
- Array ukazatelů
- Dvoudimenzionálních array

Cílový proces musí být uživatelský proces, nemůže to být proces jádra. Jestliže cílový proces neexistuje je generována chyba.

15.6. Správné užití uložených procedur

- Užít výraz/Apply formula
- Dotazy a třídění dle výrazů/ Query, Order by formula
- Komunikace mezi klientem a serverem
- Složené dotazy, které mohou být urychleny manipulací se sety nebo clustery
- Import záznamů do databáze
- Export záznamů do diskového souboru

15.6.1. Techniky, které pracují na Server, ale musí být používány jen vyjímečně

- ALERT
- CONFIRM
- Request
- DISPLAY RECORD
- Debugger

15.7. Na co nepoužívat uložené procedury?

- Tisky
- Změny nabídek
- Složité rozhraní
- Dialogy, které potřebují vstup

15.7.1. Položky, které je potřeba používat opatrně

Jsou to všechny položky, které vyvolají prvky rozhraní a vyžadují potvrzení u některých z nich především manipulace s disky, může celý 4D Server čekat na odklepnutí hlášení než bude opět pokračovat v práci, takže celá síť v tuto chvíli nepracuje .

15.8. Rychlý přehled

Uložené procedury, které jsou již zahrnuty v databázi

Podívejme se na již existující uložené procedury z kurzu Více procesů a více uživatelů.

15.9. Triggery prováděné na serveru

Všechny triggery se provádějí na server, ale my chceme použít triggery k obnově array, které jsou používány na klientu. Jak můžeme provést to, že klient získá tato obnovená data a zobrazí je ve svých array?

Procesy spuštěné na serveru ze serveru jsou rovněž uloženými procedurami.

Spustíme proces na serveru k vytvoření array na serveru. Server bude udržovat svou kopii meziprocessní array, které bude aktualizovat. Pokaždé když klient bude potřebovat array požádá server o kopii uloženou v BLOB a získá ji pomocí GET PROCESS VARIABLE.

Nakonec potřebujeme určit na daném stroji uživatele zda je to 4D Client nebo jednouživatelská aplikace, abychom mohli správně větvit kód..

15.9.1. Application type - > Číslo

Příkaz Application navrací číselnou hodnotu, která označuje typ 4D prostředí, které je spuštěno (klient, jednouživatelská verze).

15.9.2. Process number(NázevProcesu {;*}) -> IDProcesu

Příkaz Proces number navrátí IDProcesu pro proces určený názvem. Volitelná hvězdička způsobí, že daný proces bude hledán na serveru. Jestliže proces neexistuje je navrácena 0.

15.9.3. Vytvoření array na serveru.

Pokud máte 4D Insider můžete přenést následující kód tímto produktem místo vytváření každé metody individuálně (poznámka: stále budete muset překopírovat krok 4 textovým způsobem, protože 4D Insider neumí překopírovat trigger bez kopírování celé tabulky). Uživatelé 4D Insider si mohou přemístit celý kód z knihovny insider CustomerSP a pak provést pouze kroky 4,7 a 8.

1. Upravte metodu projektu CHO_InitializeCustomerIPArrays následovně:

```
…
SORT
ARRAY(<>CHO_asCompany;<>CHO_asCustomerID;<>CHO_asContactName;<>CHO
_asPhone)

If(Application type = 4D Server)
   VARIABLE TO BLOB(<>CHO_asCustomerID;<>CHO_oCustomerID)
   VARIABLE TO BLOB(<>CHO_asCompany;<>CHO_oCompany)
   VARIABLE TO BLOB(<>CHO_asContactName;<>CHO_oContactName)
   VARIABLE TO BLOB(<>CHO_asPhone;<>CHO_oPhone)
End if
      `Konec metody
```

2. Vytvořte novou metodu projektu P_InitializeIPArrays následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: P_InitializeIPArrays
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Naplní meziprocessní array serveru
      ` Jestliže je na serveru pak ponechá proces v paměti pro dostup klienta k array
      ` přes GET PROCESS VARIABLE
      ` Pracuje v procesu  Initialize Server

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

CHO_InitializeCustomerIPArrays

If( Application type = 4D Server)
   PAUSE PROCESS(Current process)
End if
      `Konec metody
```

3. Nahraďte metodu databáze Při spuštění serveru následujícím způsobem nebo importuje pomocí textového editoru:

```
      ` Database Method: On Server Startup
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Inicializuje informace na server

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

      ` Deklarace lokálních proměnných
C_LONGINT($Lpid)

<>fQuit:=False
<>hTimeDifference:=?00:00:00?
<>MainProcess:=Current process

$Lpid:=PROCESS_LSpawnProcess
("P_InitializeIPArrays";32;"InitializeServer";False;False)

      ` Konec metody
```

4. Nahraďte trigger pro tabulku Zákazníci následujícím textem nebo importujte pomocí textového editoru:

```
If (False)
      ` Trigger: Customers
      ` Kurz programovani ACI University
      ` By: Jim Steinman
      ` Datum: 1/15/97

      ` Účel: Trigger k ovládání událostí databáze pro [Zákazníci]

   <>f_Version6x10:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

      ` Upraveno: 3/17/97
   <>fK_Wilbur:=True
      ` Přidány rysy kontroly mazání
      ` Přidány rysy obnovy array

End if

      ` Definice parametrů
C_LONGINT($0)                    ` Navracený kód chyby
```

```
      ` Definice lokálních proměnných
C_LONGINT($LDatabaseEvent)    ` Testovaná událost databáze
C_LONGINT($LSelectedElement) ` Odpovídající číslo prvku
C_BOOLEAN($fCompanyModified)    ` Změna názvu společnosti
C_BOOLEAN($fNameModified)   ` Změna kontaktní osoby
C_BOOLEAN($fPhoneModified)   ` Změna telefonu

      ` Výchozí přiřazení
$0:=0
$fCompanyModified:=False
$fNameModified:= False
$fPhoneModified:= False

$LDatabaseEvent:=Database event

While (Semaphore("ModCustomerArrays"))
   PROCESS_MyDelay(Current process;1)
End while

Case of
   ś ($LDatabaseEvent = On Saving Existing Record Event)
      GEN_TimeStamp (->[Zákazníci]DatumUpravení;->[Zákazníci]ČasUpravení)
      $LSelectedElement:=Find in array(<>CHO_asCustomerID;[Zákazníci]IDZákazníka)
      If ($LSelectedElement > 0)
         If (<>CHO_asCompany{$LSelectedElement} # [Zákazníci]Firma)
            <>CHO_asCompany{$LSelectedElement}:=[Zákazníci]Firma)
            $fCompanyModified:=True
         End if
         If (<>CHO_asContactName{$LSelectedElement} # ([Zákazníci]Jméno + " " +
[Zákazníci]Příjmení))
            <>CHO_asContactName{$LSelectedElement}:=[Zákazníci]Jméno + " " +
[Zákazníci]Příjmení)
            $fNameModified:= True
         End if
         If (<>CHO_asPhone{$LSelectedElement} #  String(Num([Zákazníci]Telefon);
"|FormátTelefony")
            <>CHO_asPhone{$LSelectedElement}:=String(Num([Zákazníci]Telefon);
"|FormátTelefony")
            $fPhoneModified:= True
         End if
      End if

   ś ($LDatabaseEvent = On Saving New Record Event)
      $LSelectedElement:=Size of array(<>CHO_asCustomerID)+1
      ARRAY STRING(11;<>CHO_asCustomerID;$LSelectedElement)
      ARRAY STRING(31;<>CHO_asCompany;$LSelectedElement)
      ARRAY STRING(33;<>CHO_asContactName;$LSelectedElement)
      ARRAY STRING(25;<>CHO_asPhone;$LSelectedElement)
      <>CHO_asCustomerID{$LSelectedElement}:=[Zákazníci]IDZákazníka
      <>CHO_asCompany{$LSelectedElement}:=[Zákazníci]Firma
```

```
      <>CHO_asContactName{$LSelectedElement}:=[Zákazníci]Jméno + " " +
[Zákazníci]Příjmení
      <>CHO_asPhone{$LSelectedElement}:=String(Num([Zákazníci]Telefon);
"|FormátTelefony")
      $fCompanyModified:=True

   ś ($LDatabaseEvent=On Deleting Record Event)
      RELATE MANY([Zákazníci]IDZákazníka)
      If (Records in selection([Faktury]) = 0)
         $LSelectedElement:=Find in
array(<>CHO_asCustomerID;[Zákazníci]IDZákazníka)
         If ($LSelectedElement > 0)
            DELETE ELEMENT(<>CHO_asCustomerID;$LSelectedElement)
            DELETE ELEMENT(<>CHO_asCompany;$LSelectedElement)
            DELETE ELEMENT(<>CHO_asContactName;$LSelectedElement)
            DELETE ELEMENT(<>CHO_asPhone;$LSelectedElement)
            $fCompanyModified:=True
         End if
      Else
         $0:=-16001                    ` Existují vztažené záznamy
      End if

End case

If ($LDatabaseEvent > 0)
   If ($0 = 0)
      If (Application type # 4D Server)
         If ($fCompanyModified )
            SORT
ARRAY(<>CHO_asCompany;<>CHO_asCustomerID;<>CHO_asContactName;
<>CHO_asPhone)
         End if
      Else
         Case of
            ś ($fCompanyModified )
               SORT ARRAY(<>CHO_asCompany;<>CHO_asCustomerID;
<>CHO_asContactName; <>CHO_asPhone)
               SET BLOB SIZE(<>CHO_oCustomerID;0)
               SET BLOB SIZE(<>CHO_oCompany;0)
               SET BLOB SIZE(<>CHO_oContactName;0)
               SET BLOB SIZE(<>CHO_oPhone;0)
               VARIABLE TO BLOB(<>CHO_asCustomerID;<>CHO_oCustomerID)
               VARIABLE TO BLOB(<>CHO_asCompany;<>CHO_oCompany)
               VARIABLE TO BLOB(<>CHO_asContactName;<>CHO_oContactName)
               VARIABLE TO BLOB(<>CHO_asPhone;<>CHO_oPhone)

            ś ($fNameModified )
               SET BLOB SIZE(<>CHO_oContactName;0)
               VARIABLE TO BLOB(<>CHO_asContactName;<>CHO_oContactName)

            ś ($fPhoneModified )
```

```
                        SET BLOB SIZE(<>CHO_oPhone;0)
                        VARIABLE TO BLOB(<>CHO_asPhone;<>CHO_oPhone)
                    End case
                End if
            End if
        End if

        CLEAR SEMAPHORE("ModCustomerArrays")
            `Konec triggeru
```

5. Upravte metodu projektu MyStartup následovně:

```
❖                   $Lpid:=PROCESS_LSpawnProcess
("CHO_InitializeCustomerIPArrays";32;"Initialize";True;False)

❖               <>fIsClient:=(Application type = 4D Client)
❖               If (Not(<>fIsClient))                          ` Neběžíme na klientu
                   $Lpid:=PROCESS_LSpawnProcess ("P_InitializeIPArrays";32;"Initialize";True;False)
❖               Else
❖                  <>LServerArraysPid:=Process number("InitializeServer";*)
❖                  If (<>LServerArraysPid = 0)
❖                     <>LServerArraysPid:=Execute on server("P_InitializeIPArrays";32*1024
;"InitializeServer")  ` Vytvořit pokud neexistuje
❖                  End if
❖               End if
```

6. Nahraďte metodu projektu CHO_FillCustomerArray následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: CHO_FillCustomerArray
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vytvoří meziprocessní array zákazníků

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

  ` Deklarace lokálních proměnných
C_LONGINT($LWindowID)

While (Semaphore("ModCustomerArrays"))
   PROCESS_MyDelay(Current process;1)
End while

If (<>fIsClient)                              ` Jsme spuštěn na klientu
   $LWindowID:=WIN_LNewWindow (280;40;Upper centered;Modal dialog box)
   MESSAGE("Kopírují se array")

      `Překopíruj array z meziprocessních
   GET PROCESS
VARIABLE(<>LServerArraysPid;<>CHO_oCustomerID;$oTempBlob)
   BLOB TO VARIABLE($oTempBlob;CHO_asChoice1)
   SET BLOB SIZE($oTempBlob;0)
   GET PROCESS VARIABLE(<>LServerArraysPid;<>CHO_oCompany;$oTempBlob)
   BLOB TO VARIABLE($oTempBlob;CHO_asChoice2)
   SET BLOB SIZE($oTempBlob;0)
   GET PROCESS
VARIABLE(<>LServerArraysPid;<>CHO_oContactName;$oTempBlob)
   BLOB TO VARIABLE($oTempBlob;CHO_asChoice3)
   SET BLOB SIZE($oTempBlob;0)
   GET PROCESS VARIABLE(<>LServerArraysPid;<>CHO_oPhone;$oTempBlob)
   BLOB TO VARIABLE($oTempBlob;CHO_asChoice4)
   SET BLOB SIZE($oTempBlob;0)

   CLOSE WINDOW

Else
      ` Get the arrays from the IP arrays
   COPY ARRAY(<>CHO_asCustomerID;CHO_asChoice1)
```

```
   COPY ARRAY(<>CHO_asCompany;CHO_asChoice2)
   COPY ARRAY(<>CHO_asContactName;CHO_asChoice3)
   COPY ARRAY(<>CHO_asPhone;CHO_asChoice4)
End if

CLEAR SEMAPHORE("ModCustomerArrays")
    `Konec metody
```

7. Otevřete okno vlastností tabulky pro tabulku [Zákazníci] zaškrtněte trigger Při uložení nového záznamu.
8. Testujte v módu klient/server.

16. Rozšíření vlastního výběrového seznamu k zahrnutí produktů

Nyní, když jsme si vše základní připravili můžeme naše řešení rozšířit tak, aby zahrnovalo i další okna výběrových seznamů. Když přidáváme položky faktury do faktury můžeme nahradit výběrový seznam 4D pro čísla produktů a názvy naším vlastním.

Zde máme co budeme zobrazovat ve čtyřech array:

1) IDProduktu
2) Titul
3) Rok & Kategorie spojené v jedno
4) Cena

Potřebujete vytvořit metody podobné CHO_FillCustomerArrays, CHO_InitializeCustomerIPArrays a trigger zákazníků a metody objektu na IDProduktu ve formuláři [PoložkyFaktury]Vložený, který bude nahrazovat funkci sCompanyName.

Kdy by jste měli naplnit array je diskutabilní. V případě názvu firmy jsme naplňovali array pokaždé, když probíhal vstup do proměnné sCompanyName. Tento způsob může vyvolat velký pohyb na síti, protože položky faktur se často zadávají (náš původní automatický výběr nyní již nefunguje, protože jsme přešli na manuální vztahy). Jestliže budeme předpokládat, že když jsme zahájili práci na faktuře tak produkty zůstanou nebo měly by zůstat stabilní můžeme zavést array se zavedením faktury a vymazat je z paměti při opuštění faktury.

Musíte upravit metodu projektu P_InitializeIPArrays tak, aby zahrnovala inicializaci meziprocessní array produktů.

Kontrola mazání pro produkty by byla vhodná pokud k němu existují položky faktur. Ale nezapomeňte upravit metodu projektu ERROR_DeletingRecords tak, aby zahrnovala toto nové chybové hlášení.

16.1. COMPRESS BLOB (blob{; komprese})

Příkaz COMPRESS BLOB zkomprimuje předaný BLOB s pomocí vnitřního algoritmu komprese 4D. Volitelný parametr komprese vám dovolí nastavit způsob jakým bude BLOB zkomprimován:

Jestliže předáte 1, BLOB bude komprimován největším možným způsobem bez ohledu na rychlost komprese a dekomprese.

Jestliže předáte 2, BLOB bude komprimován a dekomprimován jak je nejrychleji možné bez ohledu na kompresní poměr (komprimovaný BLOB bude větší než u předchozí možnosti).

Jestliže předáte jinou hodnotu nebo jestliže parametr vynecháte BLOB bude zkomprimován jak to jde nejvíce tj. jako by jste použili v tomto parametru 1.

Po provedení příkazu bude systémová proměnná OK nastavena na1 v případě, že BLOB byl zkomprimován úspěšně. Jestliže se operace neprovedla, systémová proměnná OK bude nastavena na 0 (případ kdy není dost paměti pro provedení komprese).

UPOZORNĚNÍ: Zkomprimovaný BLOB je stále BLOB takže vám nic nebrání upravit jeho obsah. Pokud tak však učiníte příkaz EXPAND BLOB nebude schopen dekomprimovat BLOB správně.

16.2. EXPAND BLOB (blob)

Příkaz EXPAND BLOB dekomprimuje určený BLOB, který byl v předchozím zkomprimován pomocí příkazu COMPRESS BLOB.

Po provedení příkazu bude systémová proměnná OK nastavena na 1, jestliže BLOB byl dekomprimován, jestliže dekomprese nemohla být provedena je systémová proměnná OK nastavena na 0.

16.3. Použití BLOB tak, aby obsahoval více array

Jestliže předpokládáme že, se data produktů budou měnit pouze zřídka, můžeme podstatně získat na chování tím, že sestavíme naše array, umístíme je do jednoho BLOB a ten pak zkompaktujeme. Tento způsob sníží provoz na síti. Příklady v této databázi ukazují, že zkompaktovaný BLOB velký přibližně 22K zatímco dekomprimovaný BLOB má 106K.

16.3.1 Vytvoření výběrového seznamu pro produtky.

Jestliže máte 4D Insider můžete přemístit následující kód s použitím tohoto produktu místo vytváření metod jednu po druhé. Uživatelé 4D Insider mohou přemístit celý kód z knihovny insider ProductsSP, provést kroky 8 a 9 a pokračovat krokem 13.

1. Vytvořte novou metodu projektu CHO_PackProductBLOB následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: CHO_PackProductBLOB
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vytvoří a zkomprimuje BLOB z array

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

If (Application type = 4D Server)                  ` Vytvoří jeden BLOB ze 4 array
   VARIABLE TO BLOB(<>CHO_asTitle;<>CHO_oProducts;*)
   VARIABLE TO BLOB(<>CHO_asProductPartNo;<>CHO_oProducts;*)
   VARIABLE TO BLOB(<>CHO_asYearCategory;<>CHO_oProducts;*)
   VARIABLE TO BLOB(<>CHO_asPrice;<>CHO_oProducts;*)
   COMPRESS BLOB(<>CHO_oProducts)
   ARRAY STRING(11;<>CHO_asProductPartNo;0)       ` Array není potřeba
   ARRAY STRING(31;<>CHO_asTitle;0)
   ARRAY STRING(33;<>CHO_asYearCategory;0)
   ARRAY STRING(25;<>CHO_asPrice;0)
End if
      `Konec metody
```

2. Vytvořte novou metodu projektu CHO_FillProductArrays následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: CHO_FillProductArrays
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Přemístí obsah BLOB do array výběrového seznamu

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definuje lokální proměnné
C_LONGINT($LWindowID)
C_LONGINT($LOffset)                ` Posun pro čtení z BLOB
```

```
C_BLOB($CHO_oTempBLOB)     ` Dočasný název BLOB

While (Semaphore("ModProductArrays"))
   PROCESS_MyDelay(Current process;1)
End while

If (<>fIsClient)                              ` Jsme spuštěni na klientu
   $LWindowID :=WIN_LNewWindow (280;40;Upper centered;Modal dialog box)
   MESSAGE("Čekejte prosím")

      `Získá array pomocí BLOB z procesu serveru
   GET PROCESS
VARIABLE(<>LServerArraysPid;<>CHO_oProducts;$CHO_oTempBLOB)
   EXPAND BLOB($CHO_oTempBLOB)
   $LOffset:=0
   BLOB TO VARIABLE($CHO_oTempBLOB;CHO_asChoice2;$LOffset)
   BLOB TO VARIABLE($CHO_oTempBLOB;CHO_asChoice1;$LOffset)
   BLOB TO VARIABLE($CHO_oTempBLOB;CHO_asChoice3;$LOffset)
   BLOB TO VARIABLE($CHO_oTempBLOB;CHO_asChoice4;$LOffset)

   SET BLOB SIZE($CHO_oTempBLOB;0)  ` Vymaže paměť zabranou BLOB
   CLOSE WINDOW

Else
      `  Překopíruje array z meziprocessních array
   COPY ARRAY(<>CHO_asProductPartNo;CHO_asChoice1)
   COPY ARRAY(<>CHO_asTitle;CHO_asChoice2)
   COPY ARRAY(<>CHO_asYearCategory;CHO_asChoice3)
   COPY ARRAY(<>CHO_asPrice;CHO_asChoice4)
End if

CLEAR SEMAPHORE("ModProductArrays")
      `Konec metody
```

3. Vytvořte novou metodu projektu CHO_InitializeProductIPArrays následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: CHO_InitializeProductIPArrays
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vytvoří meziprocessní array produktů

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if
```

```
      ` Definuje lokální proměnné
C_LONGINT($CHO_LNumberElements)          ` Počet záznamů v array
C_LONGINT($i)                            ` Čítač smyčky

      ` Array pro tabulku Produkty
ARRAY STRING(11;<>CHO_asProductPartNo;0)
ARRAY STRING(31;<>CHO_asTitle;0)
ARRAY INTEGER($CHO_aiYear;0)
ARRAY STRING(15;$CHO_asCategory;0)
ARRAY REAL($CHO_arPrice;0)

ALL RECORDS([Produkty])

SELECTION TO ARRAY([Produkty]IDZboží;<>CHO_asProductPartNo;
[Produkty]Název;<>CHO_asTitle;[Produkty]Rok;$CHO_aiYear;
[Produkty]Kategorie;$CHO_asCategory;[Produkty]Cena;$CHO_arPrice)
REDUCE SELECTION([Produkty];0)

$CHO_LNumberElements:=Size of array(<>CHO_asProductPartNo)
ARRAY STRING(33;<>CHO_asYearCategory;$CHO_LNumberElements)
ARRAY STRING(25;<>CHO_asPrice;$CHO_LNumberElements)

For ($i;1;$CHO_LNumberElements)
   <>CHO_asYearCategory{$i}:=String($CHO_aiYear{$i})+" "+$CHO_asCategory{$i}
   <>CHO_asPrice{$i}:=String($CHO_arPrice{$i};"|Částka")
End for

SORT
ARRAY(<>CHO_asTitle;<>CHO_asProductPartNo;<>CHO_asYearCategory;<>CHO_a
sPrice)

CHO_PackProductBLOB
  `Konec metody
```

4. Upravte metodu projektu P_InitializeIPArrays následujícím způsobem:

```
…
CHO_InitializeCustomerIPArray
❖ CHO_InitializeProductIPArrays

…
```

5. Upravte novou metodu projektu ERROR_DeletingRecords následujícím způsobem:

```
…
❖ :(Error = -16002)
❖ $sMessage:="Jeden z produktů, který se snažíte vymazat byl prodán a je na faktuře."
+Char(Carriage return)+"Produkt nemůže být vymazán!"
…
```

6. Vytvořte novou metodu projektu INITIALIZE_GEN_ModifyVariables následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
    ` Metoda: INITIALIZE_GEN_ModifyVariables
    ` Kurz programovani ACI University
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Inicializuje proměnné pro každou smyčku GEN_ModifySelection

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

fProcessAvailable:=False

ARRAY STRING(11;CHO_asChoice1;0)
ARRAY STRING(31;CHO_asChoice2;0)
ARRAY STRING(33;CHO_asChoice3;0)
ARRAY STRING(25;CHO_asChoice4;0)
  `Konec metody
```

7. Upravte metodu projektu GEN_ModifySelection následovně:

```
…
❖ ~~fProcessAvailable:=False~~
❖ INITIALIZE_GEN_ModifyVariables
```

8. Upravte metodu formuláře [Faktury]Vstupní následovně:

```
…
  ś ($LFormEvent = On Load)
…
((Následující kód umístěte blízko konce událost Při zavedení (On Load))
    If(Not(fNewInvoice ))
      SET ENTERABLE(sCompanyName;False)
❖     CHO_FillProductArray
    End if

…

  ś ($LFormEvent = On Unload)
  ON ERR CALL("")
❖   ARRAY STRING(11;CHO_asChoice1;0)
❖   ARRAY STRING(31;CHO_asChoice2;0)
❖   ARRAY STRING(33;CHO_asChoice3;0)
❖   ARRAY STRING(25;CHO_asChoice4;0)
…
```

9. Upravte metodu objektu sCompanyName ve formuláři [Faktury]Vstupní následovně:

```
…
❖ CHO_FillProductArrays
  POST KEY(Tab;0)  `Tab na další pole
    `Konec metody
```

10. Otevřete podformulář [PoložkyFaktury]Vložený.
11. Ujistěte se, že jediná událost, která je zapnutá pro pole IDZbožíPoložky je Při získání fokusu.
12. Nahraďte metodu objektu pole IDZbožíPoložky následujícím textem nebo importujte pomocí textové editoru:

```
If (False)
    ` Metoda objektuIDZbožíPoložky [PoložkyFaktury];"Vložený"
    ` Kurz programovani ACI University
    ` Autor: Jim Steinman
    ` Datum: 1/5/97

    ` Účel: Počítá ceny

  <>f_Version6x00:=True
  <>f_Version6x30:=True
  <>fK_Wilbur:=True
```

```
      ` Upraveno: 3/17/97
      ` Přidán vlastní výběrový seznam
   <>fK_Wilbur:=True

End if

CHO_DisplayChoiceList

If (OK=1)
   [PoložkyFaktury]IDZbožíPoložky:=CHO_asChoice1{CHO_asChoice1}
   RELATE ONE([PoložkyFaktury]IDZbožíPoložky)

   [PoložkyFaktury]JednotkováCena:=[Produkty]Cena
   [PoložkyFaktury]CenaCelkem:=[PoložkyFaktury]JednotkováCena
*[PoložkyFaktury]Množství
   [Faktury]CelkemFaktura:=Sum([PoložkyFaktury]CenaCelkem)
End if

REDRAW WINDOW
POST KEY(Tab;0)  `Tab na další pole
    `Konec metody
```

13. Nahraďte trigger pro tabulku Produkty následujícím textem nebo importujte:

```
If (False)
      ` Trigger: Products
      ` Kurz programovani ACI University
      ` By: Jim Steinman
      ` Datum: 1/15/97

      ` Účel: Trigger k ovládání událostí databáze pro [Produkty]

   <>f_Version6x10:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

      ` Upraveno: 3/17/97
   <>fK_Wilbur:=True
      ` Přidána obnovat meziprocessní array

End if

  ` Definice lokálních proměnných
C_LONGINT($LDatabaseEvent)    ` Testovaná událost databáze
C_LONGINT($LSelectedElement) ` Číslo odpovídajícího prvku
C_LONGINT($LOffset)              ` Posun BLOB pro čtení
C_LONGINT($LApplicationType) ` Typ spuštěné aplikace
C_LONGINT($0)                    ` Navracený kód chyby
```

```
$LApplicationType:=Application type

$LDatabaseEvent:=Database event
$0:=0                          ` Přiřazení výchozí hodnoty chyby

While (Semaphore("ModProductsArrays"))
   PROCESS_MyDelay(Current process;1)
End while

If (($LApplicationType = 4D Server) & ($LDatabaseEvent > 0))    ` Opětovné obnovení
BLOB
   EXPAND BLOB(<>CHO_oProducts)
   $LOffset:=0
   BLOB TO VARIABLE(<>CHO_oProducts;<>CHO_asTitle;$LOffset)
   BLOB TO VARIABLE(<>CHO_oProducts;<>CHO_asProductPartNo;$LOffset)
   BLOB TO VARIABLE(<>CHO_oProducts;<>CHO_asYearCategory;$LOffset)
   BLOB TO VARIABLE(<>CHO_oProducts;<>CHO_asPrice;$LOffset)

   SET BLOB SIZE(<>CHO_oProducts;0)              ` Uvolnění paměti zabrané BLOB
End if

Case of
   ś ($LDatabaseEvent= On Saving Existing Record Event)
      GEN_TimeStamp (->[Produkty]DatumÚpravy;->[Produkty]ČasÚpravy)
      $LSelectedElement:=Find in array(<>CHO_asProductPartNo;[Produkty]IDZboží)
      If ($LSelectedElement>0)
         <>CHO_asTitle{$LSelectedElement}:=[Produkty]Název
         <>CHO_asYearCategory{$LSelectedElement}:=String([Produkty]Rok)+"
"+[Produkty]Ketegorie
         <>CHO_asPrice{$LSelectedElement}:=String([Produkty]Cena);"|Částka")
      End if

   ś ($LDatabaseEvent= On Saving New Record Event)
      $LSelectedElement:=Size of array(<>CHO_asProductPartNo)+1
      ARRAY STRING(11;<>CHO_asProductPartNo;$LSelectedElement)
      ARRAY STRING(31;<>CHO_asTitle;$LSelectedElement)
      ARRAY STRING(33;<>CHO_asYearCategory;$LSelectedElement)
      ARRAY STRING(25;<>CHO_asPrice;$LSelectedElement)
      <>CHO_asProductPartNo{$LSelectedElement}:=[Produkty]IDZboží
      <>CHO_asTitle{$LSelectedElement}:=[Produkty]Název
      <>CHO_asYearCategory{$LSelectedElement}:=String([Produkty]Rok)+"
"+[Produkty]Ketegorie
      <>CHO_asPrice{$LSelectedElement}: =String([Produkty]Cena);"|Částka")

   ś ($LDatabaseEvent=Delete Record Event)
      RELATE MANY([Produkty]IDZboží)
      If (Records in selection([PoložkyFaktury])=0)
         $LSelectedElement:=Find in array(<>CHO_asProductPartNo;[Produkty]IDZboží)
```

```
      If ($LSelectedElement>0)
        DELETE ELEMENT(<>CHO_asProductPartNo;$LSelectedElement)
        DELETE ELEMENT(<>CHO_asTitle;$LSelectedElement)
        DELETE ELEMENT(<>CHO_asYearCategory;$LSelectedElement)
        DELETE ELEMENT(<>CHO_asPrice;$LSelectedElement)
      End if
    Else
      $0:=-16002  ` Existují záznamy ve vztahu, navrať kód chyby
    End if
End case

If (LDatabaseEvent > 0)
  If ($0 = 0)
    If (<>CHO_asTitle{$LSelectedElement } # [Produkty]Název)
      SORT
ARRAY(<>CHO_asTitle;<>CHO_asProductPartNo;<>CHO_asYearCategory;<>CHO_asPri
ce)
    End if
    If ($LApplicationType = 4D Server)
      CHO_PackProductBLOB
    End if
  End if
End if

CLEAR SEMAPHORE("ModProductArrays")
 `Konec triggeru
```

14. Otevřete okno vlastností tabuky pro tabulku [Produkty] a zapněte triggery Při mazání záznamu a Při uložení nového záznamu.
15. Změňte vztah mezi tabulkami [PoložkyFaktury] a [Produkty] to tak, že vypnete Automaticky jedinec ke skupině (bylo potřeba pro řízení mazání, které nyní ovládá trigger).
16. Restartujte databázi a otestujte váš kód (poznámka: po restartu musíte určitý čas počkat na vytvoření meziprocessních array .

17. Vytváření vztažených záznamů

Při přidávání našich vlastních výběrových seznamů jsme vypnutím automatických vztahů ztratili schopnost automatického dotazu na vytvoření vztažených záznamů. Kam a jak můžeme přidat naše vlastní rysy, které nahrazují tyto automatické?

Od zákazníka jsme se dozvěděli, že jediné místo kam chce přidat tento rys je při vytváření nové faktury tak, aby obsluha byla schopna přidat nového zákazníka. Nechce, aby lidé, kteří vytvářejí nové faktury byly schopni přidávat nové produkty.

17.0.1. Nahrazení vnitřních rysů 4D pro vytváření vztažených záznamů

1. Otevřete formulář [zDialogy];"CHO_ChoiceListDialog".
2. Přidejte tlačítko s následujícími vlastnostmi.

Typ objektu	Název objektu	Velikost	Vlastnosti	Události
Tlačítko	CHO_bAddNew Text tlačítka "Přidat nový záznam"	Rozhodněte	Akce: Přijmout Posunout svisle Posunout podélně Styl: Tlačítka	Při klepnutí

3. Nahraďte metodu objektu sCompanyName ve formuláři [Faktury]Vstupní následujícím textem nebo importujte verzi 3:

```
If (False)
    ` Metoda: sCompanyName
    ` Kurz programovani ACI University
    ` Genericke metody z nastroju ACI
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Zobrazí  záznamy [Zákazníci] ve vlastním výběrovém seznamu

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

ALL RECORDS([Zákazníci])

CHO_FillCustomerArray
CHO_DisplayChoiceList  (True)
```

```
Case of
  ś (CHO_bAddNew = 1)
    CUSTOMER_Initialize          ` Získá hierarchický seznam
    ADD RECORD([Zákazníci])

    If (Is a list(hlCustomerTabs))   ` Vymaže seznam
      CLEAR LIST(hlCustomerTabs)
    End if

    If (OK=1)
      [Faktury]IDZákazníka:=[Zákazníci]IDZákazníka
      sCompanyName:=[Zákazníci]Firma
    End if

  ś (OK=1)
    [Faktury]IDZákazníka:=CHO_asChoice1{CHO_asChoice1}
    RELATE ONE([Faktury]IDZákazníka)
    sCompanyName:=[Zákazníci]Firma
End case

ARRAY STRING(11;CHO_asChoice1;0)
ARRAY STRING(31;CHO_asChoice2;0)
ARRAY STRING(33;CHO_asChoice3;0)
ARRAY STRING(25;CHO_asChoice4;0)

REDRAW WINDOW
CHO_FillProductArrays
POST KEY(Tab;0)  `Tab do dalšího pole
 `Konec metody
```

4. Upravte metodu projektu CHO_DisplayChoiceList následovně:

```
If (False)
    ` Metoda: CHO_DisplayChoiceList (boolean)
    ` Kurz programovani ACI University
    ` Genericke metody z nastroju ACI
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Displays a custom choice list

  <>fGeneric:=True
  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

    ` Deklaruje proměnné
C_BOOLEAN($1)                    ` Přidat nový záznam povoleno
```

```
      ` Deklarovat lokální proměnné
C_LONGINT($LWindowID)

      ` Nastaví výchozí
CHO_fAddNew:=False
If(Count parameters > 0)
   CHO_fAddNew:=$1
End if

INPUT FORM([zDialogy];"CHO_ChoiceListDialog";*)

$LWindowID:=WIN_LNewWindow (-1;-1;Upper centered;Plain no zoom box window)
DIALOG([zDialogy];"CHO_ChoiceListDialog")
CLOSE WINDOW
      `Konec metody
```

5. Upravte metodu formuláře [zDialogy]CHO_ChoiceListDialog následovně:

```
If (False)
      ` Metoda formuláře: [zDialogy];"CHO_ChoiceListDialog"
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vymaže používané proměnné úhozu

   <>fGeneric:= True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

C_LONGINT($LFormEvent)

$LFormEvent:=Form event

Case of
   :($LFormEvent = On Load)
      CHO_sSelectedItem:=""
      CHO_sQueryString:=""
      CHO_asChoice1:=1
      CHO_bAddNew:=0
      If (CHO_fAddNew)
         SET VISIBLE (CHO_bAddNew ; True)
      Else
         SET VISIBLE (CHO_bAddNew ; False)
      End if

End case
      `Konec metody
```

6. Navraťte se do prostředí Vlastní nabídky a proveďte testování.

18. Zástupné názvy tabulek a polí

Velice často se programátor a konečný uživatel nemohou shodnout na názvech, které chtějí používat, programátoři nemají rádi mezery, koneční uživatelé ano, programátoři zacházejí s kódovanými názvy koneční uživatelé ne.

Pole, která jsou přidána později a nejsou blízko ostatních, uživatel nemůže najít.

Ve vestavěných editorech hledání a třídění jsou všechna pole v abecedním pořadí a ne v logickém.

Naštěstí 4D v6 nabízí řešení zástupné názvy pro tabulky a pole. S tímto rysem můžete pojmenovat pole, která uživatel vidí tak jak on bude chtít (limit je 31 znaků). Uživatelé pak při použití zástupných názvů nemusí ani vědět jak jsou pole a tabulky skutečně nazvány..

18.1. SET TABLE TITLES (Názvyťabulek; Čísla tabulek)

Parametr	Typ	Popis
Názvytabulek	Array Řetězců	Názvy tabulek jak chcete aby se zobrazovali
Číslatabulek	Číselné Array	Skutečná čísla tabulek

Příkaz SET TABLE TITLES vám umožní maskovat, přezvat a změnit pořadí tabulek vaší databáze, když se při zobrazení ve standartních dialozích 4th Dimension jako je např. Editor rychlých zpráv a to v Prostředích uživatele a Vlastní nabídky.

Obě array Názvytabulek a Číslatabulek musí být synchronizovány. Array Názvytabulek předáváte názvy jak chcete, aby by ly zobrazovány . Jestliže nechcete zobrazit určitou tabulku jednoduše nezahrnete její název do tohoto array, tabulky se objeví v pořadí, které určíte v tomto array. V každém prvku array Číslatabulek předáváte skutečné číslo tabulky odpovídající číslu z Prostředí návrháře. Číslo tabulky odpovídající určitému názvu musí být ve stejném prvku array jako array Názvytabulek.

Máte např. databázi složenou z tabulek A,B a C vytvořených v tomto pořadí. Chcete, aby se tyto tabulky zobrazovaly jako X,Y a Z. Kromě toho nechcete vůbec ukázat tabulku B. Nakonec chcete, aby se zobrazovaly tabulky v pořadí Z a X. Vytvoříte si pole NázvyTabulek o prvcích Z a Y pole ČíslaTabulek o prvcích 3 a 1.

Příkaz SET TABLE TITLES nemění skutečnou strukturu vaší databáze. Jediný vliv má na zobrazení v standartních dialogových oknech 4th Dimension jako je Editor dotazů. Rozsah příkazu SET TABLE TITLES je na pracovní sekci. Jednou výhodou tohoto faktu v prostředí klient/server je, že rozdílné pracovní stanice 4D Client mohou simultánně

vidět, databázi zcela různým způsobem. Příkaz SET TABLE TITLES můžete použít tolikrát kolikrát chcete tj. změnit způsob zobrazení, kdy chcete. Je potřeba mít pouze na paměti, že se příkaz projeví až v nově zobrazených dialogových oknech 4th Dimension.

Používejte příkaz SET TABLE TITLES pro:

- Dynamickou lokalizaci databáze.
- Zobrazování tabulek způsobem, kterým chcete nezávisle na skutečné definici ve vaší databázi.
- Zobrazování tabulek způsobem, který závisí na přístupových právech uživatele.

UPOZORNĚNÍ: Příkaz SET TABLE TITLES nepřepíše vlastnost tabulky Neviditelná. Jestliže tabulka byla v Prostředí návrháře nastavena jako neviditelná, tak se neobjeví v dialozích 4th Dimension ani když ji zahrnete do příkazu SET TABLE TITLES.

18.2. SET FIELD TITLES (tabulka | podtabulka; Názvypolí; Číslapolí)

Parametr	Typ	Popis
tabulka | podtabulka	Tabulka nebo Podtabulka	Tabulka nebo podtabulka pro kterou nastavujete názvy polí
Názvypolí	Array řetězců	Názvy polí jak se mají zobrazit
Číslapolí	Číselné array	Skutečná čísla polí

Příkaz SET FIELD TITLES vám umožní maskovat, přejmenovat a změnit pořadí polí v tabulce nebo podtabulce, kterou předáváte v parametru tabulka nebo podtabulka. Tyto nové názvy se objeví ve standartních dialogových oknech 4th Dimension jako Editor dotazů v Prostředí uživatele a Vlastní nabídky.

Array Názvypolí a Číslapolí musí být synchronizovány. V array Názvypolí předáte názvy tak jak chcete, aby se zobrazovaly. Jestliže nechcete určité pole ukázat, jednoduše jeho název do array nezahrnete. Pole se objeví v pořadí jak je určíte v tomto array. V každém prvku array Číslapolí, předáváte skutečné číslo pole odpovídající Prostředí návrháře.

Máte např. tabulku nebo podtabulku složenou z polí F,G a H vytvořených v tomto pořadí. Chcete, aby se pole objevovaly jako M, N a O. Kromě toho nechcete ukázat pole N. Nakonec chcete zobrazit pole O a M v tomto pořadí. Předejte proto pole array Názvypolí o dvou prvcích O a M a array Číslapolí s prvky 3 a 1v těchto pořadích.

SET FIELD TITLES nezmění skutečnou strukturu vaší tabulky. Jediný efekt bude mít na standartní dialogová okna 4th Dimension jako Editor dotazů uvnitř Prostředí uživatele a Vlastní nabídky. Rozsah příkazu SET FIELD TITLES je pracovní sekce. Jednou z výhod tohoto způsobu je, že v prostředí klient/server může být na různých pracovních stanicích 4D Client zobrazována tatáž tabulka různými způsoby. Příkaz SET FIELD TITLES můžete volat kolikrát chcete, jediné co se musíte pamatovat je, že příkaz bude mít vliv pouze na nově otevíraná okna ve 4th Dimension.

Použijte příkaz SET FIELD TITLES pro:

- Dynamickou lokalizaci tabulky.
- Zobrazení polí způsobem jakým chcete nezávisle na skutečné definici tabulky.
- Zobrazení polí způsobem, který závisí na konkrétním uživateli a jeho přístupových právech.

UPOZORNĚNÍ: Příkaz SET FIELD TITLES nepřepíše vlastnost pole Neviditelné. Jestliže je pole nastaveno jako Neviditelné na úrovni návrháře, tak se v dialogových oknech neobjeví dokonce i když jej zahrnete do volání SET FIELD TITLES.

18.3. Count tables -> Číslo

Count tables navrátí počet tabulek v databázi. Tabulky jsou číslovány v pořadí, ve kterém byly vytvořeny.

18.4. Table name (Číslotabulky | Ukazateltabulky) -> řetězec

Table name vrátí název tabulky, který odpovídá předanému číslu tabulky či ukazateli tabulky. Název tabulky odpovídá názvu z Prostředí návrháře.

18.5. Field(ukazatel pole) -> Číslo

Příkaz Field v této syntaxi vrátí na předaný ukazatel pole ČísloPole v tabule (pořadí pole v tabulce). Jiná syntaxe, která byla diskutována v předchozím kursu je, že tento příkaz funguje i naopak, na předané číslo pole vrátí jeho ukazatel.

18.6. Čtení struktury

Můžeme nechat 4D, aby většinu práce provedla za nás. Začneme čtením ve vhodných částech struktury, abychom ji zobrazili uživateli. Všechny kursy, kterými jste zatím prošli používali názvovou konvenci, která doposud nebyla vysvětlena tj. že všechny tabulky, které nejsou pro konečného uživatele začínají písmenem „z“. Nyní tuto konvenci

použijeme k okamžitému vyloučení těchto tabulek z zástupných názvů, protože uživatel o těchto tabulkých nepotřebuje vůbec vědět.

18.6.1. Vytvoření tabulky [zStruktura]

1. Vytvořte tabulku s názvem zStruktura.
2. Nastavte vlastnost tabulky na Neviditelné.
3. Přidejte do nové tabulky následující pole:

Název pole	**Typ**	**Vlastnosti**
ČísloTabulky	Integer	Nutný vstup, Pouze zobrazit, Indexované
ČísloPole	Integer	Nutný vstup, Pouze zobrazit, Indexované
PlatnýNázev	Alfa 31	Nutný vstup, Pouze zobrazit
NázevZástupce	Alfa 31	
PořadíZástupce	Integer	
VložitVlastníHledání	Logické	
VložitVlastníTřídění	Logické	

18.6.2. Změny existujícího kódu k využití nových schopností

Bohužel potřebujeme změnit určité části existujícího kódu v databázi

1. Upravte metodu projektu GEN_FieldsToArray následovně:

```
…
      If (($fIncludeInvisible) | (Not($fFieldInvisible))          ` Include if not
invisible or not checking
         $LSizeOfArray:=$LSizeOfArray+1
         ARRAY STRING(31;$asFieldNames;$LSizeOfArray)
❖        $asFieldNames{$LSizeOfArray}:= GEN_sFriendlyName (Field name($pField))
❖        $asFieldNames{$LSizeOfArray}:= Field name($pField)
         ARRAY POINTER($apFieldPointers;$LSizeOfArray)
         $apFieldPointers{$LSizeOfArray}:=$pField
         ARRAY INTEGER($aiFieldNumbers;$LSizeOfArray)
         $aiFieldNumbers{$LSizeOfArray}:=$i
      End if
…
```

18.2.3. Čtení struktury

1. Vytvořte novou metodu projektu ALIAS_fReadStructure následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: ALIAS_fReadStructure
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Čte strukturu a ukládá ji v databázi

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definovat parametry
C_BOOLEAN($0)                  ` Úspěšné čtení

      ` Deklarace lokálních proměnných
C_LONGINT($i)                  ` Čítač smyčky
C_LONGINT($j)                  ` Čítač smyčky
C_LONGINT($LNumberTables)      ` Počet tabulek v databázi
C_LONGINT($LFound)                 ` Pozice v array
C_LONGINT($LReadWriteState)    ` Původní stav ČístPsát tabulky zStruktura
C_LONGINT($LFieldType)         ` Typ pole zástupce
C_LONGINT($LFieldLength)       ` Délka pole zástupce
C_BOOLEAN($fFieldIndexed)      ` Je pole indexované
C_STRING(31;$sTableName)

ARRAY INTEGER($aiTableCount;0)
ARRAY INTEGER($aiTableNumber;0)
ARRAY INTEGER($aiFieldNumber;0)
ARRAY STRING(31;$asActualName;0)
ARRAY STRING(31;ALIAS_asActualName;0)
ARRAY INTEGER(ALIAS_aiField;0)
ARRAY STRING(31;$asAliasName;0)
ARRAY INTEGER($aiAliasOrder;0)
ARRAY BOOLEAN($abIncludeQuery;0)
ARRAY BOOLEAN($abIncludeOrder;0)

If (Not(Semaphore("ModifyingStructureTable")))

   $LReadWriteState:=LOCK_LSet2ReadWrite (->[zStruktura];False)   ` Ujištění, že tabulku
lze upravovat
```

```
      ` Čtení tabulek
  $LNumberTables:=Count tables

  QUERY([zStruktura];[zStruktura]ČísloTabulky = 0)          ` Dotaz v existujících
datech
  SELECTION TO ARRAY([zStruktura]ČísloTabulky;$aiTableCount;
[zStruktura]ČísloPole;   $aiTableNumber;[zStruktura]PlatnýNázev;
ALIAS_asActualName;[zStruktura]NázevZástupce;$asAliasName;
[zStruktura]PořadíZástupce;$aiAliasOrder)

  $LNumberTables:=Count tables
  For ($i;1;$LNumberTables)
    $ sTableName:= Table name($i)
    If (Ascii($sTableName[[1] ]) #Ascii( "z")) ` Vynechat tabulky začínajíci na male z
      $LFound:=Find in array($aiTableNumber;$i)
      If ($LFound>0)
        ALIAS_asActualName{$LFound}:=$sTableName  `Obnova názvu v případě
změn písmen
      Else
          ` Přidali jsme novou tabulku
        $LFound:=Size of array($aiTableCount)+1
        ARRAY INTEGER($aiTableCount;$LFound)
        ARRAY INTEGER($aiTableNumber;$LFound)
        ARRAY STRING(31;ALIAS_asActualName;$LFound)
        ARRAY STRING(31;$asAliasName;$LFound)
        ARRAY INTEGER($aiAliasOrder;$LFound)
        $aiTableCount{$LFound}:=0          ` Nula v čísle tabulky znamená záznam je
tabulka
        $aiTableNumber{$LFound}:=$i
        ALIAS_asActualName{$LFound}:=$sTableName
        $asAliasName{$LFound}:=GEN_sFriendlyName ($sTableName)
        $aiAliasOrder{$LFound}:=$LFound
      End if
    End if
  End for

  ARRAY TO SELECTION($aiTableCount;[zStruktura]ČísloTabulky;$aiTableNumber;
[zStruktura]ČísloPole;ALIAS_asActualName;[zStruktura]PlatnýNázev;
$asAliasName;[zStruktura]NázevZástupce;$aiAliasOrder;[zStruktura]PořadíZástupce)

      ` Číst pole
  COPY ARRAY($aiTableNumber;$aiTableCount)        `Přesun platných čísel
      `Clear everything to make it fresh
  For ($i;1;Size of array($aiTableCount))
      `Clear everything to make it fresh again
    ARRAY INTEGER($aiTableNumber;0)
    ARRAY INTEGER($aiFieldNumber;0)
    ARRAY STRING(31;$asActualName;0)
    ARRAY STRING(31;ALIAS_asActualName;0)
    ARRAY INTEGER(ALIAS_aiField;0)
```

```
      ARRAY STRING(31;$asAliasName;0)
      ARRAY INTEGER($aiAliasOrder;0)
      ARRAY BOOLEAN($abIncludeQuery;0)
      ARRAY BOOLEAN($abIncludeOrder;0)

      QUERY([zStruktura];[zStruktura]ČísloTabulky=$aiTableCount{$i})    ` Hledání
existujících dat
      SELECTION TO
ARRAY([zStruktura]ČísloTabulky;$aiTableNumber;[zStruktura]ČísloPole;
$aiFieldNumber;[zStruktura]PlatnýNázev;$asActualName;[

   zStructure]AliasName;$asAliasName;[zStruktura]PořadíZástupce;$aiAliasOrder;
[zStruktura]VložitVlastníHledání;$abIncludeQuery;[zStruktura]VložitVlastníTřídění;
$abIncludeOrder)

      GEN_FieldsToArray (Table($aiTableCount{$i});->ALIAS_asActualName; ->
ALIAS_aiField;False;False)
                                          `Vynechání neviditelných polí, stejně se
nezobrazí

      For ($j;1;Size of array(ALIAS_asActualName))
         $LFound:=Find in array($aiFieldNumber;ALIAS_aiField{$j})
         If ($LFound>0)
            $asActualName{$LFound}:=ALIAS_asActualName{$j}   `Upravit název v případě
změny písmene
         Else
               ` Přidali jsme nové pole
            $LFound:=Size of array($aiFieldNumber)+1
            ARRAY INTEGER($aiTableNumber;$LFound)
            ARRAY INTEGER($aiFieldNumber;$LFound)
            ARRAY STRING(31;$asActualName;$LFound)
            ARRAY STRING(31;$asAliasName;$LFound)
            ARRAY INTEGER($aiAliasOrder;$LFound)
            ARRAY BOOLEAN($abIncludeQuery;$LFound)
            ARRAY BOOLEAN($abIncludeOrder;$LFound)
            $aiTableNumber{$LFound}:=$aiTableCount{$i}
            $aiFieldNumber{$LFound}:=ALIAS_aiField{$j}
            $asActualName{$LFound}:=ALIAS_asActualName{$j}
            $asAliasName{$LFound}:=GEN_sFriendlyName (ALIAS_asActualName{$j})
            $aiAliasOrder{$LFound}:=$LFound

            GET FIELD
PROPERTIES($aiTableNumber{$LFound};$aiFieldNumber{$LFound};
$LFieldType;$LFieldLength;$fFieldIndexed)
            $abIncludeQuery{$LFound}:=$fFieldIndexed     ` Výchozí stav indexu pro
vlastní dotazy
            $abIncludeOrder{$LFound}:=$fFieldIndexed     ` Výchozí stav indexu pro vlastní
třídění

         End if
      End for
```

```
      ARRAY TO SELECTION($aiTableNumber;[zStruktura]ČísloTabulky;
$aiFieldNumber; [zStruktura]ČísloPole;$asActualName;[zStruktura]PlatnýNázev;
$asAliasName; [zStruktura]NázevZástupce;$aiAliasOrder; [zStruktura]PořadíZástupce;
$abIncludeQuery;[zStruktura]VložitVlastníHledání;$abIncludeOrder;
[zStruktura]VložitVlastníTřídění)
   End for

   LOCK_Set2ReadOnly (->[zStruktura];$LReadWriteState )

   CLEAR SEMAPHORE("ModifyingStructureTable")
   $0:=True

Else
   ALERT("Někdo již upravuje tabulku struktury.")
   $0:=False
End if
      `Konec metody
```

2. Přejděte do Prostředí uživatele a proveďte metodu ALIAS_fReadStructure.

18.3.4. Instalace zástupných jmen

1. Vytvořte novou metodu projektu ALIAS_DeployAliases následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: ALIAS_DeployAliases
      ` Kurz programovani ACI University
      ` Generic metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Instaluje zástupná jména pro tabulky a pole

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokalních proměnných
C_LONGINT($i)                        ` Čítač smyčky

ARRAY INTEGER($aiTableNumber;0)
ARRAY INTEGER($aiFieldNumber;0)
ARRAY STRING(31;$asAliasName;0)
ARRAY INTEGER($aiAliasOrder;0)

QUERY([zStruktura];[zStruktura]ČísloTabulky = 0)        ` Dotaz na existující názvy
tabulek
      ` TableNumber = 0 znamená název tabulky
      ` Skutečná čísla tabulek pro tabulky jsou uložena v  [zStruktura]ČísloPole
If (Records in selection([zStruktura])>0)
   SELECTION TO ARRAY([zStruktura]ČísloPole;$aiTableNumber;
[zStruktura]NázevZástupce;$asAliasName;[zStruktura]PořadíZástupce;$aiAliasOrder)
   SORT ARRAY($aiAliasOrder;$asAliasName;$aiTableNumber)
         ` Kontrola prázdných názvů a jejich vyloučení  z array takže nebudou vůbec
instalovány
   For ($i;Size of array($aiAliasOrder);1;-1)
      If (Length($asAliasName{$i})=0)
         DELETE ELEMENT($asAliasName;$i)
         DELETE ELEMENT($aiAliasOrder;$i)
         DELETE ELEMENT($aiTableNumber;$i)
      End if
   End for
         ` Instaluj názvy tabulek
   SET TABLE TITLES($asAliasName;$aiTableNumber)
End if
```

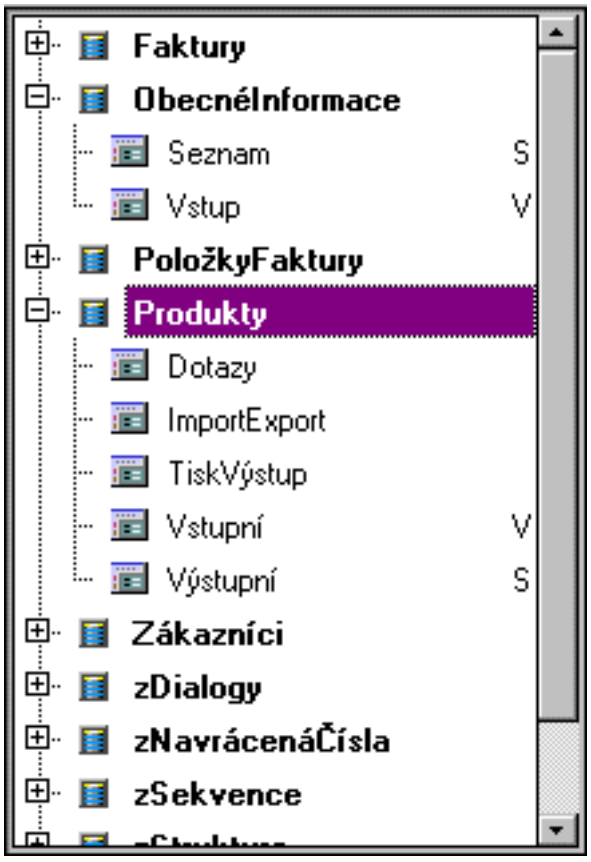
```
For ($i;1;Size of array($aiTableNumber))
   QUERY([zStruktura];[zStruktura]ČísloTabulky = $aiTableNumber{$i};*)
   QUERY([zStruktura]; & ;[zStruktura]NázevZástupce # "")
   If (Records in selection([zStruktura])>0)
      SELECTION TO
ARRAY([zStruktura]ČísloPole;$aiTableNumber;[zStruktura]NázevZástupce;
$asAliasName;[zStruktura]PořadíZástupce;$aiAliasOrder)
      SORT ARRAY($aiAliasOrder;$asAliasName;$aiTableNumber)
         ` Kontrola prázdných názvů a jejich vyloučení z array takže nebudou vůbec
instalovány
      For ($j;Size of array($aiAliasOrder);1;-1)
         If (Length($asAliasName{$j})=0)
            DELETE ELEMENT($asAliasName;$j)
            DELETE ELEMENT($aiAliasOrder;$j)
            DELETE ELEMENT($aiFieldNumber;$j)
         End if
      End for
         ` Instaluj názvy polí
      SET FIELD TITLES(Table($aiTableNumber{$i})->;$asAliasName;
$aiFieldNumber)
   End if
End for
      `Konec metody
```

2. Přejděte do Prostředí uživatele a proveďte nějaké změny v zástupných názvech. Poznámka: Musíte provést metodu E_Set2ReadWrite aby jste mohli změny uskutečnit.
3. Vyzkoušejte různé editory v Prostředí uživatele pro existující tabulky a názvy polí. Poznámka: Položky faktur a Rychlé zprávy vám poskytnou přehled všech změn.
4. Proveďte metodu ALIAS_DeployAliases z Prostředí uživatele.
5. Vyzkoušejte různé editory v Prostředí uživatele pro vámi změněné názvy tabulek a polí.
6. Upravte metodu projektu MyStartup následovně:

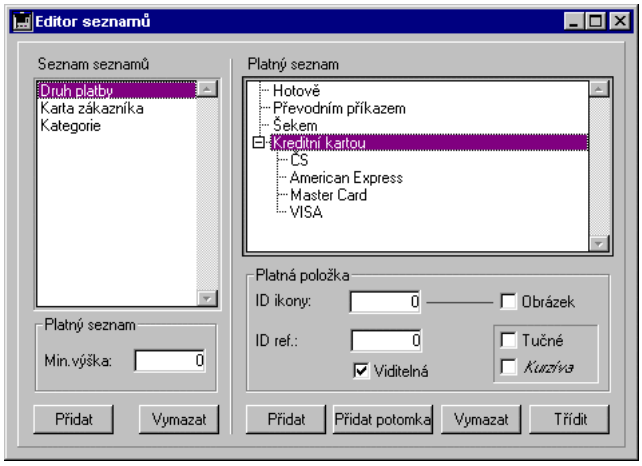
```
…

❖       ALIAS_DeployAliases
      `Konec metody
```

19. Více o hierarchických seznamech

Hierarchické seznamy se používají ve všech oblastech uvnitř 4D Prostředí návrháře a Prostředí uživatele ve vestavěných editorech.



Je velice snadné je používat.



Pokud chcete hierarchické seznamy samy vytvářet a ošetřovat znamená to samozřejmě programování ve vašem kódu.

19.1. Hierarchické seznamy - přehled příkazů

19.1.1. APPEND TO LIST (list; Položkatext; Položkaref{; podlist{; rozšiř}})

Parametr	Typ	Popis
list	ListRef	Číslo odkazu na seznam
Položkatext	Řetězec	Text v nové položce seznamu (max 31 znaků)
Položkaref	Číslo	Jedinečné číslo odkazu na novou položku seznamu
podlist	ListRef	Volitelný hierarchický podseznam pro novou položku seznamu
rozšiř	Logické	Indikuje zda podseznam bude rozšířen či zúžen

19.1.2. SET LIST ITEM PROPERTIES (list; Položkaref; dostup; styl; ikona)

Ke změně vzhledu položky seznamu a přidání obrázku musíte použít tento příkaz, je to jednodušší provést okamžite po přidání položky seznamu, protože se na poslední přidané prvky odkazujete nejjednodušeji (0).

Parametr	Typ	Popis
list	ListRef	Číslo odkazu na seznam
Položkaref	Číslo	Číslo odkazu na položku nebo 0 pro poslední přidanou položku
dostupné	Logické	TRUE = Dostupné, FALSE = Nedostupné
styl	Číslo	Styl písma pro položku

ikona	Číslo	Číslo zdroje 'cicn' nebo 65536 číslo zdroje 131072+ číslo odkazu v knihovně

19.2. Další příkazy hierarchických seznamu

19.2.1. GET LIST ITEM (list; Položkapoz; Položkatext; Položkaref)

Parametr	Typ	Popis
list	ListRef	Číslo odkazu na seznam
Položkapoz	Číslo	Pozice položky v rozšířeném seznamu
Položkatext	Řetězec	Text položky v seznamu
Položkaref	Číslo	Číslo odkazu na položku

19.2.2. GET LIST PROPERTIES (list; vzhled; ikona; Výškařádky)

Parametr	Typ	Popis
list	ListRef	Číslo odkazu na seznam
vzhled	Číslo	1=Mac 2=Win
ikona	Číslo	Číslo zdroje 'cicn' nebo 65536 číslo zdroje 131072+ číslo odkazu v knihovně
Výškařádky	Číslo	Minimální výška řádky vyjádřená v bodech

19.2.3 REDRAW LIST (list)

REDRAW LIST způsobí, že určený seznam bude překreslen.

19.2.4. Selected list item (list) -> Long

Parametr	Typ	Popis
list	ListRef	Číslo odkazu na seznamu
výsledek funkce	Long	Položky seznamu v rozšířeném seznamu

19.2.5. DELETE LIST ITEM (list; PoložkaRef | *{; *})

Parametr	Typ	Popis
list	ListRef	Číslo odkazu na seznam
PoložkaRef | *	Číslo odkazu na položku nebo * pro právě vybranou položku	Číslo | *
*	*	Jestliže je určen rovněž vymaže

podseznamy
Je-li vynechán podseznamy
nejsou
mazány

Příkaz DELETE LIST ITEM vymaže položku ze seznamu jehož čísla odkazů jsme předali. Jestliže předáte * jako druhý parametr vymažete ze seznamu právě vybranou položku. Jinak určujete číslo odkazu na položku, kterou chcete vymazat. Jestliže položka s daným číslem odkazu neexistuje, příkaz neprovede nic.

Jestliže pracujete s čísly odkazů na položky, vytvořte seznam, kde budou mít položky jedinečná čísla odkazů jinak nebudete schopni mezi nimi rozlišovat.

Nezávisle na tom, kterou položku mažete by jste měli určovat volitelný třetí parametr * k automatickému vymazání přiřazených podseznamů k položce (jsou-li nějaké). Jestliže nepředáte jako třetí parametr *, je dobré znát číslo odkazu na seznam přiřazený jako podseznam k položce, protože eventuelně budete muset tento přiřazený podseznam vymazat příkazem CLEAR LIST.

19.2.6. INSERT LIST ITEM (list; předPoložkaRef| *; Položkatext; Položkaref{; podlist{; rozšiř}})

Parametr	Typ	Popis
list	ListRef	Číslo odkazu na seznam
Předpoložkouref | *	Číslo | *	Číslo odkazu na položku kam vkládáme * pro právě vybranou položku seznamu
Položkatext	Řetězec	Text pro nově vkládanou položku (max 31 zn.)
PoložkaRef seznamu	Číslo	Jedinečné číslo odkazu na novou položku
podlist	ListRef	Volitelný podseznam přiřazený nové položce
rozšiř	Logické	Indikuje zda bude podseznam rozšířen či zúžen

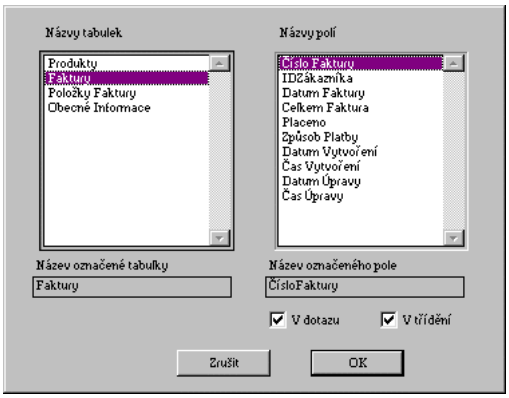
Přikaz INSERT LIST ITEM vloží novou položku do seznamu jehož číslo odkazu jste předali v parametru list. Jestliže předáte jako druhý parametr * položka je vložena před právě vybranou položku seznamu.

V tomto případě se nově vložená položka stane vybranou položkou. Jinak jestliže chcete vložit položku před určenou položku, musíte předat číslo odkazu na tuto položku.
V tomto případě nově vložená položka není automaticky vybrána. Jestliže neexistuje položka s tímto číslem odkazu, příkaz neprovede nic.

Pro nově vytvářenou položku předáváte rovněž text a číslo odkazu na ni v parametrech Položkatext a Položkaref.

19.3. Ovládání zástupných názvů tabulek a polí pomocí hierarchických seznamů

Nyní se chystáme vytvořit naše vlastní okno, ve kterém budeme ovládat pomocí hierarchických seznamů naše zástupné názvy tabulek a polí. Uživatelé často požadují seznamy, ve kterých lze upravovat. Podívejme se jestli toto není právě naše úloha.



19.3.1. Vytvoření systému ovládání zástupných názvů tabulek a polí

1. Vytvořte novou metodu projektu ALIAS_HandleAliasList následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: ALIAS_HandleAliasList (Pointer; Pointer; Pointer; Pointer) -> Boolean
      ` Kurz programovani ACI University
      ` Generické metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Ovládá změny v hierarchickém seznamu zástupců

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      `Deklaruje parametry
C_BOOLEAN($0)                        ` Je akce povolená
C_POINTER($1;$pDestinationObject)    ` Ukazatel na cílové array
C_POINTER($2;$pOrderArray)           ` Ukazatel  na array třídění
C_POINTER($3;$pNameArray)            ` Ukazatel na array názvů
```

```
C_POINTER($4;$pNumberArray)        ` Ukazatel na skutečná čísla

     ` Deklarace lokálních proměnných
C_LONGINT($i)                      ` Čítač smyčky
C_LONGINT($LSourceElement)         ` Zdrojový prvek vybrán
C_LONGINT($LDestinationElement)        ` Cílový prvek
C_LONGINT($LSourcePid)             ` ID procesu
C_LONGINT($LSeledtedOrderNumber)       ` Dočasné zapamatování pro číslo pořadí
C_LONGINT($LSeledtedActualNumber)      ` Dočasné zapamatování pro skutečné číslo
C_LONGINT($LFormEvent)             ` Získaná událost formuláře
C_LONGINT($LAppearance)  ` Vzhled seznamu
C_LONGINT($LCICN)  ` Odkaz na barevnou ikonu
C_LONGINT($LHieght)  ` Velikost čísla písma
C_POINTER($pSourceObject)          ` Ukazatel na zdrojový objekt
C_STRING(31;$sSelectedName)        ` Dočasné zapamatování názvu

ARRAY INTEGER($aiOrderArray;0)

     `Přiřadit  parametry k proměnným
$pDestinationObject:=$1
$pOrderArray:=$2
$pNameArray:=$3
$pNumberArray:=$4

$LFormEvent:=Form event

Case of
  ś ($LFormEvent = On Data Change)
     $LSourceElement:=Selected list item($pDestinationObject->)
     GET LIST ITEM($pDestinationObject->;$LSourceElement;
$LSeledtedActualNumber;$sSelectedName)
     $pNameArray->{$LSourceElement}:=$sSelectedName

  ś ($LFormEvent = On Drop)
          ` Získání informace o zdrojovém objektu potažení
     DRAG AND DROP
PROPERTIES($pSourceObject;$LSourceElement;$LSourcePid)

          ` Vybral uživatel platné číslo prvku?
     If ($LSourceElement>0)
               ` Získání cílového čísla prvku
        $LDestinationElement:=Drop position

           ` Je to přetažení a puštění z téhož array uvnitř téhož procesu?
        $0:=($LSourcePid = Current process) & ($pSourceObject=$pDestinationObject)

        If ($0)
              ` Jestliže ano a prvek nebyl potažen sám na sebe
           If ($LDestinationElement # $LSourceElement)

                 ` Ulož potažený prvek do dočasných proměnných
```

```
            $sSelectedName:=$pNameArray->{$LSourceElement}
            $LSeledtedActualNumber:=$pNumberArray->{$LSourceElement}

                ` Vymaž potažený prvek
            DELETE ELEMENT($pNameArray->;$LSourceElement)
            DELETE ELEMENT($pNumberArray->;$LSourceElement)
            SELECT LIST ITEM($pDestinationObject->;$LSourceElement)
            DELETE LIST ITEM($pDestinationObject->;*)

                ` Jestliže cílový prvek byl za potaženým prvkem
            If ($LDestinationElement > $LSourceElement)

                    ` Snižte číslo cílového prvku
               $LDestinationElement:=$LDestinationElement-1
            End if

            GET LIST PROPERTIES($pDestinationObject-
>;$LAppearance;$LCICN;$LHieght)
                ` Jestliže se potažení uskutečnilo za poslední prvek
            If ($LDestinationElement = -1)

                    ` Nastavte cílové číslo prvku jako nový prvek na konci array
               $LDestinationElement:=Size of array($pNameArray->)+1
               APPEND TO LIST($pDestinationObject-
>;$sSelectedName;$LSeledtedActualNumber)
            Else
               SELECT LIST ITEM($pDestinationObject->;$LDestinationElement)
               INSERT LIST ITEM($pDestinationObject-
>;*;$sSelectedName;$LSeledtedActualNumber)
            End if

                ` Vložte tento nový prvek
            INSERT ELEMENT($pNameArray->;$LDestinationElement)
            INSERT ELEMENT($pNumberArray->;$LDestinationElement)
            SET LIST ITEM PROPERTIES($pDestinationObject-
>;0;True;Plain;$LCICN)

                ` Nastavte jeho hodnotu, která byla v předchozím uložena do prvku 0
array
            $pNameArray->{$LDestinationElement}:=$sSelectedName
            $pNumberArray->{$LDestinationElement}:=$LSeledtedActualNumber

                ` Přesunovaná položka seznamu se stává nově vybraným prvkem
seznamu
            SELECT LIST ITEM($pDestinationObject->;$LDestinationElement)

                ` Změňte třídění array
            COPY ARRAY($pOrderArray->;$aiOrderArray)
            For ($i;1;Size of array($aiOrderArray))
               $aiOrderArray{$i}:=$i
            End for
```

```
            COPY ARRAY($aiOrderArray;$pOrderArray->)

            REDRAW LIST($pDestinationObject->)

          End if

        Else

          If ($LSourcePid=Current process)        `Prvek přichází z jinéhor array

            ALERT("Přesuny mezi seznamy nejsou povoleny")
          End if
        End if

      Else
        ALERT("Nejdříve musíte vybrat prvek k přetažení.")
      End if
End case
    `Konec metody
```

2. Vytvořte novou metodu projektu ALIAS_FieldData následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
    ` Metoda: ALIAS_FieldData(Longint;)
    ` Kurz programovani ACI University
    ` Genericke metody z nastroju ACI
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Vytvoři hierarchický seznam z polí

  <>fGeneric:=True
  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

    `Deklarace parametrů
C_LONGINT($1)                              ` Číslo pole

    ` Deklarace lokálních proměnných
C_LONGINT($LFieldNumber)

    ` Přiřazení parametrů k lokálnim proměnným
$LFieldNumber:=$1

QUERY([zStruktura];[zStruktura]ČísloTabulky = ALIAS_LCurrentTable;*)
QUERY([zStruktura];[zStruktura]ČísloPole = $LFieldNumber)
ALIAS_sActualFieldName:= [zStruktura]PlatnýNázev
```

```
If ([zStruktura]VložitVlastníHledání)
   ALIAS_LIncludeCustomQuery:=1
Else
   ALIAS_LIncludeCustomQuery:=0
End if

If ([zStruktura]VložitVlastníTřídění)
   ALIAS_LIncludeCustomOrder :=1
Else
   ALIAS_LIncludeCustomOrder:=0
End if
      `Konec metody
```

3. Vytvořte novou metodu projektu ALIAS_BuildFieldList následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: ALIAS_BuildFieldList (Longint; Longint; Boolean)
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vytvoří hierarchické seznamy z polí

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      `Deklarace parametrů
C_LONGINT($1;$LOldTableNumber)  ` Staré číslo tabulky
C_LONGINT($2;$LNewTableNumber)` Nové číslo tabulky
C_BOOLEAN($3;$fFieldModified)          ` Pole bylo upraveno

      ` Deklarace lokálních proměnných
C_LONGINT($i)                          ` Čítač smyčky

      ` Přiřazení parametrů k lokálním proměnným
$LOldTableNumber:=$1
$LNewTableNumber:=$2
$fFieldModified:=$3

      ` Ulož změny
If ($fFieldModified)
      ` Hledání existujících názvů polí
   MESSAGES OFF  ` Neukazuj teploměry
   QUERY([zStruktura];[zStruktura]ČísloTabulky = $LOldTableNumber)
```

```
   ORDER BY([zStruktura];[zStruktura]ČísloPole)
   MESSAGES ON
   SORT
ARRAY(ALIAS_aiFieldNumber;ALIAS_asFieldNameAlias;ALIAS_aiFieldAliasOrder)
   ARRAY TO SELECTION(ALIAS_aiFieldNumber;[zStruktura]ČísloPole;
ALIAS_asFieldNameAlias;[zStruktura]NázevZástupce;
ALIAS_aiFieldAliasOrder;[zStruktura]PořadíZástupce)
End if

If (Is a list(hlFieldNameList))
   CLEAR LIST(hlFieldNameList)
End if

hlFieldNameList:=New list
GET LIST PROPERTIES(hlFieldNameList;$LAppearance;$LCICN;$LHieght)

      ` Získání skutečných názvů tabulek
QUERY([zStruktura];[zStruktura]ČísloTabulky = 0;*)
QUERY([zStruktura]; & [zStruktura]ČísloPole = $LNewTableNumber)
ALIAS_sActualTableName:=[zStruktura]PlatnýNázev

      ` Hledání existujících názvů polí
QUERY([zStruktura];[zStruktura]ČísloTabulky = $LNewTableNumber)

      `Vytvoření array z polí
If (Records in selection([zStruktura])>0)
   SELECTION TO ARRAY([zStruktura]ČísloPole;ALIAS_aiFieldNumber;
[zStruktura]NázevZástupce;ALIAS_asFieldNameAlias;
[zStruktura]PořadíZástupce;ALIAS_aiFieldAliasOrder)
   SORT
ARRAY(ALIAS_aiFieldAliasOrder;ALIAS_asFieldNameAlias;ALIAS_aiFieldNumber)
End if

$LNumberFields:=Size of array(ALIAS_asFieldNameAlias)

For ($i;1;$LNumberFields)
   APPEND TO
LIST(hlFieldNameList;ALIAS_asFieldNameAlias{$i};ALIAS_aiFieldNumber{$i})
   SET LIST ITEM PROPERTIES(hlFieldNameList;0;True;Plain;$LCICN)
End for

      ` Nastav hodnoty pro tuto tabulku
ALIAS_LCurrentTable:=$LNewTableNumber
ALIAS_fFieldModified:=False

      ` Ukaž hodnoty polí
ALIAS_FieldData (ALIAS_aiFieldNumber{1})

      ` Překresli seznam
REDRAW LIST(hlFieldNameList)
      `Konec metody
```

4. Vytvořte novou metodu projektu M_ALIAS_ModifyAliases následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: M_ALIAS_ModifyAliases
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vytvoří hierarchický seznam z polí a tabulek

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($i)                  ` Čítač smyčky
C_LONGINT($LReadWriteState)    ` Původní stav ČístPsát tabulky [zStruktura]
C_LONGINT($LWindowID)          ` ID okna
C_BOOLEAN($fUpdated )          ` [zStruktura] bylo upravena

      ` Deklarace proměnných seznamu
C_LONGINT(hlTableNameList)     ` hierarchický seznam pro názvy tabulek
C_LONGINT(hlFieldNameList)     ` hierarchický seznam pro názvy polí

If ((Current user = "Návrhář") | (Current user = "Administrátor"))

        `Nejdřív získem změněné hodnoty
   $LWindowID:=WIN_LNewWindow (350;40;Upper centered;Modal dialog box)

   GOTO XY(2;1)
   MESSAGE("Čekejte prosím! Čtou se tabulky a pole.")
   $fUpdated:=ALIAS_fReadStructure
   CLOSE WINDOW

   If ($fUpdated)     ` Obnova byla úspěšná

      If (Not(Semaphore("ModifyingStructureTable")))

         ARRAY INTEGER(ALIAS_aiTableNumber;0)
         ARRAY STRING(31;ALIAS_asTableNameAlias;0)
         ARRAY INTEGER(ALIAS_aiTableAliasOrder;0)
         ARRAY INTEGER(ALIAS_aiFieldNumber;0)
         ARRAY STRING(31;ALIAS_asFieldNameAlias;0)
         ARRAY INTEGER(ALIAS_aiFieldAliasOrder;0)
```

```
	$LReadWriteState:=LOCK_LSet2ReadWrite (->[zStruktura];False)  ` Ujistěte se, že
tabulku lze upravovat

	QUERY([zStruktura];[zStruktura]ČísloTabulky = 0)		` Hledejte existující
názvy tabulke
		` ČísloTabulky = 0 znamená názvy tabulek
		` Aktuální čísla tabulek pro tabulky jsou uloženy v  [zStruktura]ČísloPole

	If (Is a list(hlTableNameList))
	   CLEAR LIST(hlTableNameList)
	End if

	hlTableNameList:= New list
		` Získej ikonu
	GET LIST PROPERTIES(hlTableNameList;$LAppearance;$LCICN;$LHieght)

		`Vytvoř array tabulek
	If (Records in selection([zStruktura])>0)
	   SELECTION TO ARRAY([zStruktura]ČísloPole;ALIAS_aiTableNumber;
[zStruktura]NázevZástupce;ALIAS_asTableNameAlias;[zStruktura]PořadíZástupce;
ALIAS_aiTableAliasOrder)
	   SORT ARRAY(ALIAS_aiTableAliasOrder;ALIAS_asTableNameAlias;
ALIAS_aiTableNumber)
	End if

	$LNumberTables:=Size of array(ALIAS_asTableNameAlias)

	For ($i;1;$LNumberTables)
	   APPEND TO LIST(hlTableNameList;ALIAS_asTableNameAlias{$i};
ALIAS_aiTableNumber{$i})
	   SET LIST ITEM PROPERTIES(hlTableNameList;0;True;Plain;$LCICN)
	End for

	If (Is a list(hlFieldNameList))
	   CLEAR LIST(hlFieldNameList)
	End if

	hlFieldNameList:=New list
	ALIAS_BuildFieldList (0;ALIAS_aiTableNumber{1};False)

	START TRANSACTION

	$LWidth:=370			` Požadovaná šířka
	$LHeight:=285			` Požadovaná výška.
	$LWindowID:=WIN_LNewWindow ($LWidth;$LHeight;Window centered;Modal
dialog box)
	DIALOG([zDialogy];"ALIAS_ManageAliases")
	CLOSE WINDOW

	If (OK=1)
		` Ulož změny
```

```
            MESSAGES OFF  ` Neukazuj teploměr průběhu
            If (ALIAS_fFieldModified)
               QUERY([zStruktura];[zStruktura]ČísloTabulky = ALIAS_LCurrentTable)
                                                      ` Hledej existující názvy polí
               ORDER BY([zStruktura];[zStruktura]ČísloPole)
               SORT
ARRAY(ALIAS_aiFieldNumber;ALIAS_asFieldNameAlias;ALIAS_aiFieldAliasOrder)
               ARRAY TO SELECTION(ALIAS_aiFieldNumber;[zStruktura]ČísloPole;
ALIAS_asFieldNameAlias;[zStruktura]NázevZástupce;ALIAS_aiFieldAliasOrder;
[zStruktura]PořadíZástupce)
            End if

            QUERY([zStruktura];[zStruktura]ČísloTabulky = 0)  ` Hledej existující názvy
tabulek
            ORDER BY([zStruktura];[zStruktura]ČísloPole)
            SORT
ARRAY(ALIAS_aiTableNumber;ALIAS_asTableNameAlias;ALIAS_aiTableAliasOrder)
            ARRAY TO SELECTION(ALIAS_aiTableNumber;[zStruktura]ČísloPole;
ALIAS_asTableNameAlias;[zStruktura]NázevZástupce;ALIAS_aiTableAliasOrder;
[zStruktura]PořadíZástupce)
            MESSAGES ON
            VALIDATE TRANSACTION
            ALIAS_DeployAliases           ` Instaluj změny
         Else
            CANCEL TRANSACTION
         End if

         LOCK_Set2ReadOnly (->[zStruktura];$LReadWriteState)

            ` Všechno vymaž
         CLEAR SEMAPHORE("ModifyingStructureTable")
         If (Is a list(hlFieldNameList))
            CLEAR LIST(hlFieldNameList)
         End if
         If (Is a list(hlTableNameList))
            CLEAR LIST(hlTableNameList)
         End if

         ARRAY INTEGER(ALIAS_aiTableNumber;0)
         ARRAY STRING(31;ALIAS_asTableNameAlias;0)
         ARRAY INTEGER(ALIAS_aiTableAliasOrder;0)
         ARRAY INTEGER(ALIAS_aiFieldNumber;0)
         ARRAY STRING(31;ALIAS_asFieldNameAlias;0)
         ARRAY INTEGER(ALIAS_aiFieldAliasOrder;0)

      Else
         ALERT("Někdo již upravuje tabulku struktuy.")
      End if
   End if

Else
```

```
   ALERT("Nemáte právo upravovat zástupné názvy, kontaktujte svého administrátora.")
End if
   `Konec metody
```

5. Otevřete záhlaví nabídky #1 a přidejte do nabídky Soubor následující:

Položka nabídky	Metoda
Změnit heslo…	M_GEN_ChangePassword
-	
Vlastnosti tabulek a polí…	M_ALIAS_ModifyAlliases
-	
Konec	M_Quit

6. Určete a zadejte heslo návrháře.
7. Restartujte databázi k načtení těchto nových rysů.

19.4. Další důležité informace o hierarchických seznamech

19.4.1. Omezení hierarchických seznamů

Teoretické omezení (limit) pro jeden hierarchický seznam je 32 000 prvků, každý prvek může být přiřazen k dalšímu hierarchickému seznamu o 32 000 prvcích. Proto teoretický limit je $32{,}000^{32000}$. Maximální počet prvků, které mohou být zobrazeny současně v seznamu je 32000, toto číslo zahrnuje všechny rozšířené seznamy a připojené seznamy.

Skutečný limit je omezení paměti a samozřejmě uživatel a jeho vůle jak dlouhým seznamem chce listovat.

20. Zpřístupňování a znepřístupňování položek nabídek

Příležitostně je vhodné a potřeba zpřístupnit či znepřístupnit určitou položku nabídky. Obvykle je to z některého z těchto důvodů:

- Tato možnost nabídky není dostupná či vhodná pro určitý formulář
- Tato možnost nabídky není dostupná či vhodná pro určitého uživatele
- Podmínky, které se vztahují k právě zobrazeným datum či výběrům dat jsou takové, že daná akce/položka nabídky by měla být znepřístupněna

20.1. DISABLE ITEM (nabídka; Položkanabídky{; proces})

Parametr	Typ	Popis
nabídka	Číslo	Pořadí nabídky záhlaví
Položkanabídky	Číslo	Pořadí položky nabídky v dané nabídce
proces	Číslo	Číslo odkazu na proces, ve kterém je daná nabídka zobrazen, jestliže to není tento spuštěný proces

Příkaz DISABLE ITEM znepřístupní (zešediví) položku nabídky určenou parametry nabídka a položkanabídka. Jestliže je parametr položkanabídky 0 pak budou znepřístupněny všechny položky dané nabídky.

Položka nabídky je zpřístupněna či znepřístupněna pouze dokud není záhlaví nabídky změněno pomocí příkazu MENU BAR.

Volitelný parametr proces je používán k zpřístupnění či znepřístupnění položek nabídky v různých procesech. Poznamenejme však, že se tento příkaz omezuje svým působením pouze na určený proces.

Jako obecné pravidlo platí. Jestliže objevíte, že určitou nabídku či položku nabídky musíte velice často znepřístupňovat, je výhodnější ji rovnou znepřístupnit v Editoru nabídek.

Nabídka Soubor je vždy nabídka 1. Čísla nabídek pro připojená záhlaví začínají číslem 2049. Nabídky Úpravy a Nápověda jsou zahrnovány dodatečně a nejsou součástí číslování nabídek.

20.2. ENABLE ITEM (nabídka; Položkanabídky{; proces})

Parametr	Typ	Popis
nabídka	Číslo	Pořadí nabídky záhlaví
Položkanabídky	Číslo	Pořadí položky nabídky v dané nabídce
proces	Číslo	Číslo odkazu na proces, ve kterém je daná nabídka zobrazen, jestliže to není tento spuštěný proces

Příkaz ENABLE ITEM znepřístupňuje položku nabídky určenou parametry nabídka a položka nabídky. Jestliže je položka nabídky 0. Navrátí se celá nabídka do stavu definovaného Editorem nabídek v Prostředí návrháře.

Příkaz je platný dokud není záhlaví nabídky změněno pomocí příkazu MENU BAR.

20.3. Omezený přístup k položce nabídky Vlastnosti tabulek a polí

V naší databázi nechceme, aby zástupné názvy tabulek mohl měnit každý, takže pokud uživatel není Návrhář nebo Administrátor znepřístupníme položku nabídky..

Všeobecně platí, jestliže často měníme stav určité položky nabídky, změníme jeho stav na častěji používaný přímo v Editoru nabídek.

Protože nebudeme tuto položku nabídky používat příliš často a nechceme, aby se uživatel ani náhodně dostal do této položky nabídky. Znepřístupníme ji přímo v Editoru nabídek. .

20.3.1. Nastavte položku nabídky na Nedostupná

1. Přejděte do Editoru nabídek a znepřístupněte položku nabídky Vlastnosti tabulek a polí.

2. Vytvořte novou metodu projektu MENU_SetMenuBar následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
    ` Metoda: MENU_SetMenuBar(Longint)
    ` Kurz programovani ACI University
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Ovládá stav položky nabídky

  <>f_Version6x30:=True
  <>fK_Wilbur:=True
```

```
End if

     ` Definovat parametry
C_LONGINT($1)                        ` Záhlaví nabídky k nastavneí

MENU BAR($1)
If ((Current user="Návrhář") | (Current user="Administrátor"))
   ENABLE MENU ITEM(1;9)
Else
   DISABLE MENU ITEM(1;9)  ` Znepřístupněte pro jistotu
End if
 `Konec metody
```

3. Upravte metodu projektu MyStartup na posledním řádku následovně:

```
MENU_SetMenuBar(1)
```

21. 4D Insider

Může se stát, že vaše databáze postupně vzrůstá až do fáze, kdy si již nepamatujete vše o každé metodě, tabulce, poli, proměnné atd. Debugger vám nemůže říci vše o tom co se děje, když vaše aplikace běží, rozhodně ne o všech možnostech.

1. Udělejte si kopii celé složky vaší databáze
2. Pracujte pouze na kopii
3. Spusťte 4D Insider a otevřete kopii databáze

Prostřední sloupec ukazuje kolik objektů máte v databázi.

Prolistujte si tento seznam

Tento seznam je filtrovaný takže můžete vybrat k zobrazení jenom určité typy objektů.Vyberte proměnné (variables).

Vyberte All. Klepněte na tabulku [Zákazníci] a podívejte se dva krajní sloupce, kde je používána (Used by) a co sama používá (Uses it).

Rozšiřte tabulku [Zákazníci] a klepněte na jedno z polí. Podívejte se kde je používáno.

Vyberte Project metod, klepněte na jednu z nich a opět se podívejte, kde je používána a co používá.

Vyberte příkazy (Command) a opět zkontrolujte oba krajní sloupce. .

Prověřte si takto postupně i ostatní typy objektů.

Poklepejte na určitou metodu projektu a podívejte se jak je zobrazena ve spodním okně.

Přetáhněte panel Uses (Je používáno) přes střední sloupec a klepněte na nějaký objekt ve středním sloupci. Nyní můžete vidět co používá a čím je používán.

Přetáhněte sloupec Used by zpět do středního sloupce a vyberte metodu.

Vyberte String resource a poklepejte na jeden z nich.

Poklepejte na formulář [Zákazníci]Vstupní, který má stránku 2.

Poklepejte na určité tlačítko, které má metodu objektu.

Vyberte postupně možnosti Design environment, User runtime a Variable name.

V našem kódu jsme důsledně používali konvenci názvů, která nám pomůže provádět úlohy s pomocí 4D Insider.

Dejme vyhledat proměnné <>fGeneric, <>f_Version6x10, <>f_Version6x20, <>f_Version6x30, <>fJ_Steinman, <>fK_Wilbur

Používali jsme modulární konvenci názvů.

Nastavme si filter např. pro názvy začínajíci ALIAS.

Chcete důsledně používat zdroj string k usnadnění přepisování názvů? Indsider vytvoří zdroje string za vás. Vyberte nějaké formuláře a vyzkoušejte si to.

Chcete si vytvořit knihovnu objektů z dané databáze? Vytvořte prázdnou strukturu (nový soubor), upravte velikost oken, aby se vám zobrazovaly i současná databáze i nově otevřený soubor. Postupně přetahujte požadované objekty z databáze do prázdného souboru.

Proveďte pomocí 4D Insider hledání.

Proveďte pomocí 4D Insider nahrazení.

Vyzkoušejte si psaní dokumentace, dokumentace je uložena spolu se strukturou databáze, můžete dokumentovat tabulky, formuláře, metody formulářů, triggery atd.

Tisk pracuje pro střední sloupec. Zredukujte výběr ve středním sloupci, zobrazte si tiskové možnosti a vytiskněte jej.

21.1. Použití 4D Insider k nalezení záhlaví nabídky 1

Potřebujeme znát každé místo, kde voláme příkaz MENU BAR tak, abychom se ujistili, že správně zpřístupníme či znepřístupníme položku Vlastnosti tabulek a polí. Protože jsou naše ostatní nabídky připojeny k téže nabídce Soubor, je pokaždé při použití příkazu MENU BAR znovu obnoven původní stav nabídky Soubor, takže i položka Vlastností tabulek a polí je nyní nepřístupná. Proto po každém použití příkazu MENU BAR ji musíme pro patřičné uživatele zpřístupnit.

21.1.1. Nalezení a nahrazení příkazu MENU BAR

1. Použijte 4D Insider k nalezení všech výskytů příkazu MENU BAR a vytiskněte je
2. Vraťte se do 4D a nahraďte všechna volání MENU BAR (x) voláním MENU_SetMenuBar (x)

- Kde byly umístěny všechny změny?

21.1.2. Použijte 4D Insider k nahrazení všech výskytů MENU BAR

1. Použijte vámi vytvořenou knihovnu 4D Insider k nahrazení metod projektu, které volají příkaz MENU BAR.
2. Změňte ručně metody formuláře výstupní, kde jsou použity příkazy MENU BAR.

22. Ovládání platných výběrů

Platné výběry zabírají v paměti 4 byty na každý záznam ve výběru. Platné výběry jsou uloženy na počítači serveru.

Představte si, že máte databázi se 125 000 záznamy a že uživatel použije na tuto tabulku se 125 000 záznamy příkaz ALL RECORDS. I když vezmeme v úvahu fakt, že příkaz ALL RECORD ve skutečnosti zachází s tabulkou vyjímek, je to pořád dost zabrané paměti a server musí pracovat tak, aby udržoval tento výběr v paměti. Nyní si představme 10 uživatelů a každý z nich volá ALL RECORDS na tutéž tabulku. Jaké plýtvání pamětí a časem Serveru.

Určitě si dokážete představit kolik z těchto uživatelů musí skutečně používat všech 125 000 záznamů, pravděpodobně žádný. Většina uživatelů velmi zřídka, jestli vůbec chce pracovat se všemi záznamy v tabulce. Obvykle je výběr uživatele omezen pouze na jeden záznam nebo na speciální výběr záznamů, který obdržel dotazem. Takže proč nutit Server provádět příkaz ALL RECORDS, když bude nejpravděpodobněji okamžitě následovat dotaz nebo ještě hůře se uživatel bude snažit listováním dojít na záznam, který potřebuje a způsobí tím enormní nárůst provozu na síti a zatížení serveru.

Pro lepší využití paměti zvláště při serverových řešeních, nebudeme již volat ALL RECORDS jako v Prostředí uživatele, ale raději zobrazíme uživatelský dialog tak, že se uživatel může omezit pouze na vlastní výběr záznamů a to těch, které ho skutečně v daný okamžik zajímají.

22.1. Dvojí účel dialogů, pro 4D a WEB

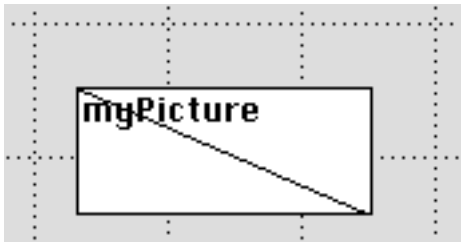
4D dovoluje použít prohlížeče Web jako klienty. I když toto připojení není úplně stejné. Ne všechno co může zpracovat samotná 4D/4D Client, pracuje pro Web. Kromě toho ne každý formulář, který vytvoříte pracuje pro Web.

22.2. Knihovna obrázků

Obrázky zabírají mnoho paměti. Jestliže zobrazujete tentýž obrázek na více než jednom formuláři v databázi a tento obrázek je ve formuláři přímo umístěn, zabere svou část paměti na každém formuláři, který se objeví na obrazovce. Proč takto plýtvat pamětí? Obvyklým způsobem je použití stejného obrázku opakovaně v jedné databázi. Existují dvě základní cesty jak takovýto obrázek umístit v paměti pouze jednou.

Za prvé, jestliže je obrázek umístěn v souboru zdrojů datového souboru, pak můžete použít příkaz GET PICTURE RESOURCE a uložit tento obrázek do proměnné, pak tuto proměnnou umístíte na formulářích a můžete ji použít kde potřebujete. Tento způsob má

své nevýhody, protože když pracujete na formuláři v Prostředí návrháře uvidíte ve formuláři pouze rámeček proměnné, skutečný obrázek nemůžete vidět.



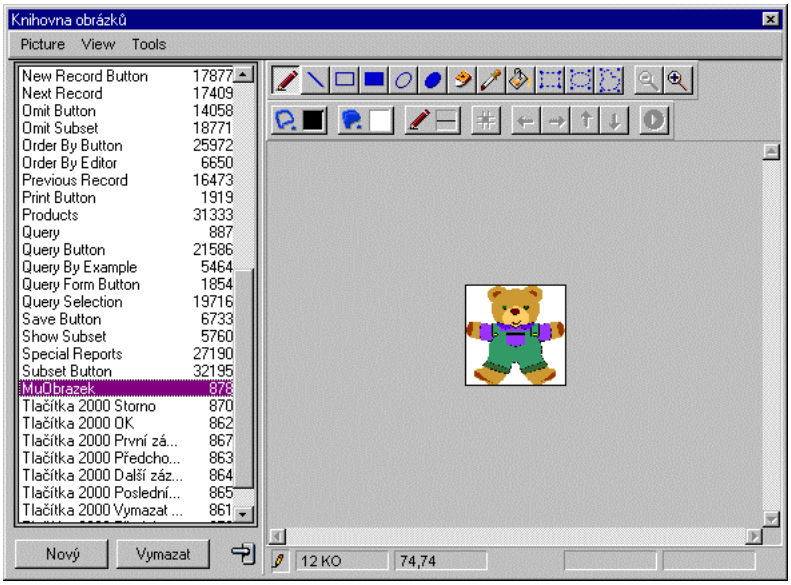
Má však i své výhody. Je velmi rychlý . Obrázky jsou všechny přeneseny na stroj klienta při prvním spuštění databáze (nebo když jsou zdroje obnovovány). V tomto případě vidíte zprávu Přenos zdrojů na tento počítač. Tyto zdroje jsou přeneseny přes síť pouze jednou pak jsou dostupné na klientu pro všechny formuláře. Jestliže používáte více procesů potřebujete k tomu, aby obrázky dostupné ve všech procesech, meziprocessní proměnné naplňované ve fázi Při spuštění. Pak jsou dostupné okamžitě kdekoliv je potřebujete. Z paměti jsou vybavovány velice rychle. Důrazně vám doporučujeme tento způsob pro obrázky, které jsou používány znovu a znovu.

Druhá metoda je pro obrázky, které nejsou používány tak často. A rovněž pro ty, kteří nejsou schopni nebo nerozumějí tomu jak dostat obrázky do PICTzdrojů souboru struktury a také rovněž pro ty, kteří se nejsou schopni vyrovnat s nevýhodami první metody.

4D v6 obsahuje část, která se jmenuje Knihovna obrázků. Obrázky vložené do této knihovny se na stroji klienta vyskytují pouze jednou. Když jsou přeneseny na počítač klienta nejsou ještě zcela známy. Nejsou však uloženy v paměti dokud nejsou potřeba. Tento způsob je pomalejší v přístupu, na druhou stranu se objeví na formuláři kam je přemístíte dokonce i v Prostředí návrháře. To co vidíte na formuláři není skutečně uložený obrázek do formuláře, ale spíše rámeček s ukazatelem na aktuální obrázek.

22.2.1.Používání Knihovny obrázků

1. Vytvořte nový prázdný formulář nazvaný [zDialogy]PRELIM_QueryOrder.
2. Nalezněte na vašem počítači nějaký obrázek v kreslícím programu a zkopírujte jej do schránky.
3. Vyberte Nástroje ¬ Knihovna obrázků.
4. Klepněte na tlačítko Nový.
5. Vložte obrázek do pravé strany okna.
6. Pojmenujte obrázek MůjObrazek.



Všimněte si čísla vpravo od názvu. Toto je číslo odkazu do knihovny obrázků. Tento obrázek můžete programem obdržet v různých příkazech, když použijete toto číslo.

7. Uchopte tento obrázek, potáhněte jej a pusťte do nového formuláře.
8. Vraťte se do systému a nalezněte další obrázek.
9. Nahraďdte obrázek MůjObrazek v pravém okně knihovny obrázků.

Všimněte si, že se obrázek ve vašem formuláři změnil. Obrázek může mít i jinou velikost než starý obrázek a nyní může vypadat dost hrozně (může být z některé strany sražený).

10. Změňte velikost obrázku ve formuláři na jeho originální velikost pomocí (□ + klepnout) (Ctrl + klepnout) na pravý spodní roh obrázku.

11. Vymažte tento obrázek z formuláře.

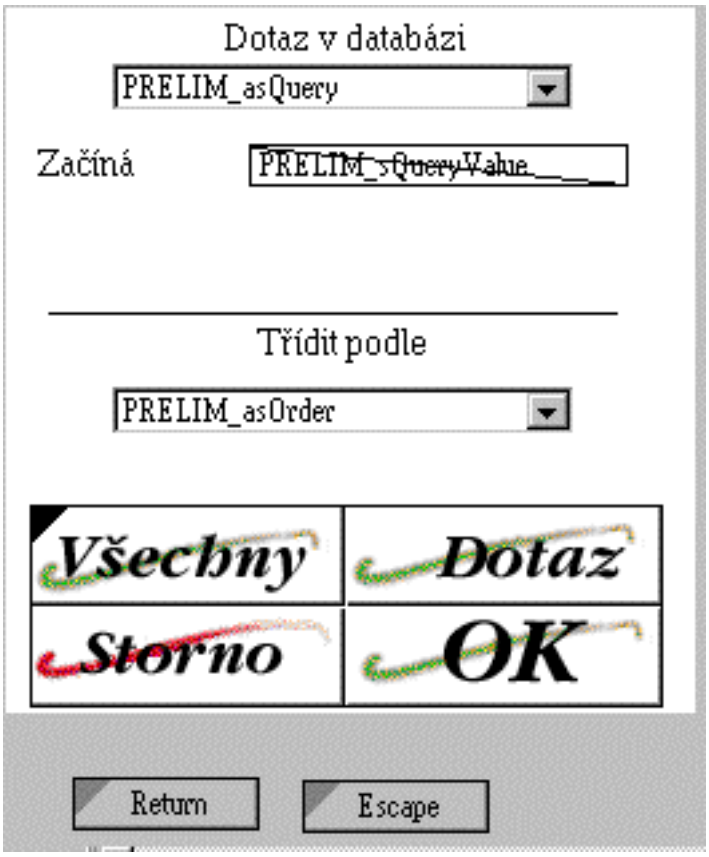
22.2.2. Vytvoření dialogu, který slouží 4D a WEB současně

1. Vložte z knihovny do formuláře z [zDialogy]PRELIM_QueryOrder obrázek CustomQueryPalette.
2. Přidejte následující položky.

(Pozn.: Zkuste je srovnat ve velikost a umístění na obrázky)

Typ objektu	Název	Vlastnosti	Další
Zvýrazněné tlačítko	bAllRecords	Akce: Přijmout	Zkratka: Ctrl G
Zvýrazněné tlačítko	bQueryEditor	Akce: Přijmout	Zkratka: Ctrl S
Zvýrazněné tlačítko	bOK	Akce: Přijmout	Zkratka: Enter
Zvýrazněné tlačítko	bCancel	Akce: Storno	Zkratka: Ctrl .
Nabídka/seznam	PRELIM_asQuery	Akce: No Action Dostupné tab Styl: Popisky vstupní	
Nabídka/seznam	PRELIM_asOrder	Akce: Žádná Dostupné tab Styl: Popisky vstupní	
Statický text	Žádný		Dotaz v databázi pro
Statický text	Žádný		Začíná:
Statický text	Žádný		Třídit podle
Dostupné	PRELIM_sQueryValue	Vzhled: Normální Styl: Popisky vstupní	
Čára	Žádný		Dělící čára (viz příklad)
Tlačítko (Stránka 0) pod zápatím	bReturn	Akce: Žádná Nedostupné tab	Text tlačítka: Return Zkratka: Return
Tlačítko (Stránka 0) pod zápatím	bEscape	Akce: Storno Nedostupné tab	Text tlačítka: Escape Zkratka: Escape

Rámeček (Stránka 0)	Žádný	Barva popředí: Bílá Výplň: Plná	Přesuňte tento objekt na pozadí celého formuláře.



Poznámka: Položky umístěné na stránku 0 se objevují při užití databáze na Web speciálním způsobem. Proto umístěte všechny možné klávesové zkratky užívané v jednouživatelské databázi či na klientu do tlačítek pod zápatí a pod okraj obrazovky na stránce 0. Když bude formulář použit pro Web tlačítka nebudou odeslána. Rovněž pozadí bude na Web odesláno jako pozadí pouze objeví-li se na stránce 0 v levém horním rohu.

3. Vytvořte metodu objektu pro bAllRecords následovně:.

   M_GEN_ShowAllRecords

4. Vytvořte metodu objektu pro bQueryEditor následovně:

   M_GEN_QueryEditor

22.3. Styly

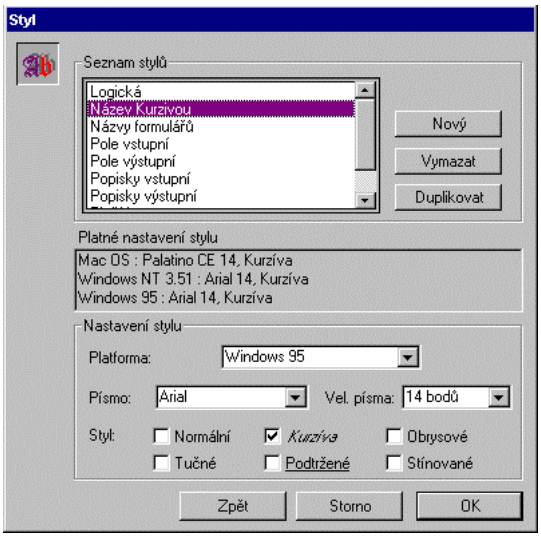
Nyní můžete definovat styly jako v libovolném textovému editoru. Na rozdíl od většiny textových editorů může být styl písma definován v závislosti na vybrané platformě na které bude formulář zobrazován. Kromě toho, když je formulář užíván na různých platformách textové položky mohou měnit i svou fyzickou velikost na formuláři tak, aby vyhověli požadavkům písma na této platformě. Pevným bodem pro objekt je levý horní roh objektu.

K omezení problémů při zobrazování na různých platformách doporučují autoři 4D, aby každý objekt zobrazující text na libovolném formuláři měl vždy přiřazený styl. Vytvoření a použití stylu.

22.3.1. Vytvoření a použití Stylu

1. Poklepejte na objekt statického textu Dotaz v databázi pro.
2. Přemístěte se na stránku Písma v definicích objektu.
3. Klepněte na tlačítko Upravit.
4. Klepněte na tlačítko Nový.
5. Pojmenujte styl `Název kurzívou`.
6. Pro každou platformu nastavte následující styl

Mac OS - Palatino CE - 14 bodů - Kurzíva
Windows 3.1 - Arial CE - 14 bodů - Kurzíva
Windows 95 - Arial CE - 14 bodů - Kurzíva



8. Klepněte na tlačítko OK.
9. Přiřaďte tento nový styl všem objektům statického textu.

22.4. Události formuláře & Metody formuláře

Když vytváříme nový formulář pro optimalizaci chování vždy nastavujeme události formuláře pouze na ty, které budou potřeba. To zabrání metodě formuláře v neustálém spouštění se i když pro nadbytečné události není co provádět.

22.4.1. Sladění událostí formulářů s metodou formuláře

1. Otevřete okno vlastností formuláře a vypněte všechny události kromě Při zavedení

Poznámka: Všechny události můžete najednou zapnout či vypnout pomocí Command + klepnout (Ctrl + klepnout) na libovolnou událost.

2. Vytvořte metodu formuláře pro [zDialogy]PRELIM_QueryOrder následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` From Method: [zDialogy]PRELIM_QueryOrder
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Sestaví vlastní array pro dotazy a třídění

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($LFormEvent)
C_LONGINT($LCurrentTable)

$LFormEvent:=Form event

If ($LFormEvent = On Load)
   $LCurrentTable:=Table(pTable)

         ` Získat pole k hledání pro nabídku
   QUERY([zStruktura];[zStruktura]ČísloTabulky = $LCurrentTable;*)
   QUERY([zStruktura]; & ;[zStruktura]VložitVlastníHledání = True)
   ORDER BY([zStruktura];[zStruktura]PořadíZástupce)
```

```
   SELECTION TO ARRAY([zStruktura]NázevZástupce;PRELIM_asQuery)
      ` Naplnit nabídku předvybranou hodnotou
   PRELIM_asQuery:=1
   PRELIM_QueryValue:=""

      ` Získat pole k třídění pro nabídku
   QUERY([zStruktura];[zStruktura]ČísloTabulky = $LCurrentTable;*)
   QUERY([zStruktura]; & ;[zStruktura]VložitVlastníTřídění = True)
   ORDER BY([zStruktura];[zStruktura]PořadíZástupce)
   SELECTION TO ARRAY([zStruktura]NázevZástupce;PRELIM_asOrder)
      ` Vložit možnost netřídit jako výchozí
   INSERT ELEMENT(PRELIM_asOrder;1;1)
   PRELIM_asOrder{1}:="** Žádné **"
      ` Naplnit nabídku předvybranou hodnotou
   PRELIM_asOrder:=1

End if
   `Konec metody
```

22.5. Finalizace

Nakonec potřebujeme nahradit příkaz ALL RECORDS a místo něj otevřít a zobrazit náš dialog.

22.5.1. Nahrazení příkazu ALL RECORDS

1. Vytvořte metodu projektu PRELIM_CustomQueryOrder následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
   ` Metoda: PRELIM_CustomQueryOrder
   ` Kurz programovani ACI University
   ` Genericke metody z nastroju ACI
   ` Autor: Kent Wilbur
   ` Datum: 3/17/97

   ` Účel: Zobrazí uživateli vlastní dialog hledání

  <>fGeneric:=True
  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

WIN_LNewWindow (265;260;Upper centered;Modal dialog box)

DIALOG([zDialogy];"PRELIM_QueryOrder")
CLOSE WINDOW
  `Konec metody
```

2. Upravte metodu projektu GEN_ModifySelection následovně:

```
If (False)
    ` Metoda: GEN_ModifySelection
    ` Kurz programovani ACI University
    ` Genericke metody z nastroju ACI
    ` Created by: Jim Steinman
    ` Datum: 1/15/97

    ` Účel: Zobrazí výstupní formulář tabulky určené pTable.

  <>fGeneric:=True
  <>f_Version6x10:=True
  <>f_Version6x20:=True
  <>f_Version6x30:=True
  <>fJ_Steinman:=True

    ` Upraveno: 1/17/97
  <>fK_Wilbur:=True
    ` Upraveno: 3/17/97
  <>fK_Wilbur:=True

End if

C_LONGINT($LWindowID)

Repeat
  INITIALIZE_GEN_ModifyVariables

  MENU_SetMenuBar(1)

  PRELIM_CustomQueryOrder

  MENU_SetMenuBar(3)

  If (OK=1)
    INPUT FORM(pTable->;"Vstupní";*)          ` Nastaví velikost okna stanovenou návrhářem
    OUTPUT FORM(pTable->;"Výstupní")

    $LWindowID:=WIN_LNewWindow (-1;-1;Cascading window;Plain no zoom box window)

    ~~ALL RECORDS(pTable ->)~~
    CREATE EMPTY SET(pTable->;"UserSet")

    PROCESS_UpdateWindowArray ("";Current process)

    WIN_OutputWindowTitle
    MODIFY SELECTION(pTable->;*)
```

```
            PROCESS_UpdateWindowArray ("";Current process)

            CLOSE WINDOW

            GEN_ClearSelection

❖           End if

            fProcessAvailable:=True
            PAUSE PROCESS(Current process)
         UNTIL( <>fQuit)
            ` Konec metody
```

3. Otestujte nový dialog v prostředí Vlastní nabídky na tabulce Produkty. (Pozn.: Jediná tlačítka, která nyní pracují jsou Všechny a Dotaz)

22.6. Naplnění všech testovaných hodnot hodnotami

V našem současném formuláři pro dialog dotazů je tento nejdřív zobrazen pak se krátce na obrazovku vrátí dialog třídění a nakonec provádění s teploměrem pokračuje. Je to proto, že kód provádění akce je umístěn pod naše tlačítka stejně jako ve výstupním formuláři. Při spojení Web je to tak jak by uživatel očekával spousta času stráveného překreslováním obrazovek. Ve skutečnosti pro tento příklad můžeme chování vylepšit zobrazením následného dialogu hledání až po uzavření vlastního dialogu.

Práce s Web vyžaduje ještě několik speciálních technik, když obdržíme odpověď od klienta Web víme, že hodnota tlačítka, které bylo stisknuto bude rovna 1. V této chvíli však žádné z dalších tlačítek, které by byly stisknuty nenaplní své hodnoty pro Web klienta. Proto bude nezbytné přiřadit hodnoty pro každou proměnnou, kterou potřebujeme kontrolovat, protože na rozdíl od 4D a 4D Client. Web klient hodnotu nepřiřadí nebo v horším případě přiřadí těmto proměnným své vlastní hodnoty.

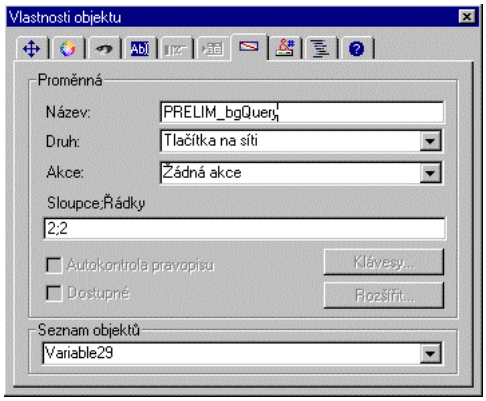
Rovněž v tomto případě můžeme využít jiný objekt 4D a použít jej míst 4 oddělených tlačítek. Tento objekt je síť tlačítek. Můžeme nahradit 4 tlačítka 1 jednoduchým tlačítkem a inicializovat hodnotu snáze pouze nastavením jedné proměnné. Pak můžeme pouze kontrolovat, která oblast sítě tlačítek byla vybrána a zachovat se podle toho.

Kvíz — Nalezení polí

- Jak můžete nalézt skutečné pole, jestliže znáte zástupný název pole?
- Jak se můžete na toto pole dotázat?
- Jak můžete toto pole třídit?

22.6.1. Přemístění akce za fázi uzavření dialogu

1. Vymažte tlačítka bAllRecrods, bOK, bCancel a bQueryEditor.
2. Vytvořte síť tlačítek nazvanou PRELIM_bgQuery, která má dvě řady a dva sloupce a upravte ji na velikost přesně podle obrázku odpovídajícího tlačítku.



3. Vytvořte novou metodu objektu PRELIM_bgQuery následovně:

```
If (False)
      ` Metoda objektu: PRELIM_bgQuery [zDialogy];"PRELIM_QueryOrder"
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Zruší formulář

   <>fGeneric :=   True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

CANCEL
   `Konec metody
```

4. Přejmenujte tlačítko bReturn na bEnter.
5. Upravte tlačítko bEnter tak, aby mělo automatickou akci Storno.

6. Vytvořte novou metodu objektu bEnter následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda objektu: bEnter [zDialogy];"PRELIM_QueryOrder"
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Přesměruje úhoz na klávesu Enter

   <>fGeneric :=   True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

If(Length(PRELIM_sQueryValue)>0)
   PRELIM_bgQuery:=4
Else
   PRELIM_bgQuery:=1
End if
   `Konec metody
```

7. Vytvořte novou metodu objektu bEscape následovně:

```
If (False)
      ` Metoda objektu: bEscape [zDialogy];"PRELIM_QueryOrder"
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Přesměruje úhoz na klávesu Escape

   <>fGeneric:=True
   <>f_Version6x30:= True
   <>fK_Wilbur:= True

End if

PRELIM_bgQuery:=3
   `Konec metody
```

8. Upravte metodu formuláře [zDialogy]PRELIM_QueryOrder následovně:

```
…
   PRELIM_asOrder:=1
```

```
❖                ` Přiřadí výchozí hodnoty pro všechna tlačítka
❖                ` Je to nezbytné pro úspěšné zobrazení dialogu na Web
❖             PRELIM_bgQuery:=0

          End if
               `Konec metody
```

9. Vytvořte novou metodu projektu ALIAS_pFieldFromAliasName následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: ALIAS_pFieldFromAliasName (string {'table number]_ -> pointer
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Navrátí ukazatel na pole určené svým zástupným názvem

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

C_POINTER($0)                    ` Ukazatel na pole dotazu
C_STRING(31;$1;$sFieldName)      ` Název zástupných polí
C_LONGINT($2;$LTableNumber)      ` Číslo tabulky

      ` Přiřazení parametrů k lokálním proměnným
$sFieldName:=$1
If(Count parameters>1)
   $LTableNumber:=$2
   QUERY([zStruktura];[zStruktura]ČísloTabulky = $LTableNumber ;*)
   QUERY([zStruktura]; & ;[zStruktura]NázevZástupce = $sFieldName)
Else
   QUERY([zStruktura];[zStruktura]NázevZástupce = $sFieldName)
   $LTableNumber := [zStruktura]ČísloTabulky
End if

$0:=Field($LTableNumber ;[zStruktura]ČísloPole)
      `Konec metody
```

10. Nahraďte metodu projektu PRELIM_CustomQueryOrder následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: PRELIM_CustomQueryOrder
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
```

```
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Zobrazí pro uživatele náš vlastní dialog hledání

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($LWidth)              ` Šířka dialogu
C_LONGINT($LHeight)                 ` Výška dialogu
C_LONGINT($LCurrentTable)       ` Číslo platné tabulky
C_LONGINT($LFieldType)          ` Typ pole
C_LONGINT($LWindowID)           ` ID okna
C_POINTER($pQueryField)         ` Ukazatel na pole dotazu

$LWidth:=265
$LHeight:=260
$LWindowID:=WIN_LNewWindow ($LWidth;$LHeight;Upper centered;Modal dialog
box)

DIALOG([zDialogy];"PRELIM_QueryOrder")
CLOSE WINDOW

If (PRELIM_bgQuery# 3)
   $LCurrentTable:=Table(pTable)
   Case of
      ś (PRELIM_bgQuery             = 4)

   $pQueryField:=ALIAS_pFieldFromAliasName(PRELIM_asQuery{PRELIM_asQuery
};
                           $LCurrentTable)

         QUERY(pTable->;$pQueryField-> = PRELIM_sQueryValue+"@")

      ś (PRELIM_bgQuery             = 2)
         M_GEN_QueryEditor

      Else
         M_GEN_ShowAllRecords

   End case

         ` Potřebujeme třídit záznamy
   If ((Records in selection(pTable->) >
0) & (PRELIM_asOrder{PRELIM_asOrder} # "** Žádné **"))
            ` Najít pole
```

```
   $pQueryField:=ALIAS_pFieldFromAliasName(PRELIM_asOrder{PRELIM_asOrder}
;$LCurrentTable)
      ORDER BY(pTable->;$pQueryField->)
   End if
End if
      `Konec metody
```

11. Upravte metodu projektu GEN_ModifySelection následovně:

- ~~If (OK= 1)~~
- If (PRELIM_bgQuery# 3)

12. Otestujte tyto změny v prostředí Vlastní nabídky.

22.7. Generický kód a ukazatele na pole mohou způsobovat problémy

Problémem je kód podobný následujícímu:

```
QUERY(pTable->;$pQueryField->=PRELIM_sQueryValue+"@")
```

Tento kód pracuje dobře jestliže je pole, na které se dotazujeme typu Alfa nebo Text. Pokud je však pole jiného typu, kód se přeruší. Na druhou stranu kontrola každého pole v databázi je poněkud nepraktická a téměř nemožná.

```
Case of:
   :(pTable = (->[Zákazníci])
      Case of
         :($pQueryField ->= [Zákazníci]Jméno)
            QUERY(pTable->;$pQueryField->=PRELIM_sQueryValue+"@")
         :($pQueryField ->= [Zákazníci]Příjmení)
            QUERY(pTable->;$pQueryField->=PRELIM_sQueryValue+"@")
…
```

22.8. Type (pole/proměnná) -> Číslo

Funkce Type navrátí číslo určující typ dat, pole či proměnné. Type je obvykle používán k testování zda pole nebo proměnná je správného typu.

22.8.1 Vylepšení kódu pro kontrolu typu dat

1. Nahraďte metodu projektu PRELIM_CustomQueryOrder následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: PRELIM_CustomQueryOrder
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Zobrazí uživateli vlastní dialog dotazů

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($LWidth)              ` Šířka dialogu
C_LONGINT($LHeight)                 ` Výška dialogu
C_LONGINT($LCurrentTable)       ` Číslo platné tabulky
C_LONGINT($LFieldType)          ` Typ pole
C_LONGINT($LWindowID)           ` ID okna
C_POINTER($pQueryField)         ` Ukazatel na pole dotazu

$LWidth:=265
$LHeight:=260
$LWindowID:=WIN_LNewWindow ($LWidth;$LHeight;Upper centered;Modal dialog
box)

DIALOG([zDialogy];"PRELIM_QueryOrder")
CLOSE WINDOW

If (PRELIM_bgQuery# 3)
   $LCurrentTable:=Table(pTable)
   Case of
      ś (PRELIM_bgQuery= 4)
         $pQueryField:=ALIAS_pFieldFromAliasName
(PRELIM_asQuery{PRELIM_asQuery} ; $LCurrentTable)
```

```
            ` Jaký je typ pole
         $LFieldType:= Type($pQueryField->)
         Case of
            ś ($LFieldType = Is Alpha Field) | ($LFieldType= Is Text)
               QUERY(pTable->;$pQueryField->= PRELIM_sQueryValue+"@")

            ś ($LFieldType = Is Real) | ($LFieldType = Is LongInt) | ($LFieldType = Is
Integer)
               QUERY(pTable->;$pQueryField-> = Num(PRELIM_sQueryValue))

            ś ($LFieldType= Is Date)
               QUERY(pTable->;$pQueryField-> = Date(PRELIM_sQueryValue))

            ś ($LFieldType = Is Boolean)
               QUERY(pTable->;$pQueryField->= ($tQueryValue[[1]]="T");*)
               QUERY(pTable->; | ;$pQueryField->= ($tQueryValue[[1]]="F"))

            ś ($LFieldType= Is Time)
               QUERY(pTable->;$pQueryField-> = Time(PRELIM_sQueryValue))
         End case

      ś (PRELIM_bgQuery =2)
         M_GEN_QueryEditor

      Else
         M_GEN_ShowAllRecords

   End case

      ` Potřebujeme třídit záznamy
   If ((Records in selection(pTable->) >
0) & (PRELIM_asOrder{PRELIM_asOrder} # "** Žádné **"))
         ` Nalézt pole
      $pQueryField:=
ALIAS_pFieldFromAliasName(PRELIM_asOrder{PRELIM_asOrder};$LCurrentTable)
      ORDER BY(pTable->;$pQueryField->)
   End if
End if
      `Konec metody
```

22.9. Přidání funkcí pro 4D a 4D Client

Další cvičení přidá dodatečné funkce do formuláře pro 4D. Naneštěstí nebude fungovat vůbec pro Web. Proto musíme udělat později zcela jiný formulář pro Web.

Někdy je vhodné spíše než jednu hodnotu vložit určitý rozsah dat. Rozsahy budou vhodné pro hodnoty Číselné, Datumu nebo Času.

22.9.1. Přidání rysů do vlastního dialogu dotazů

1. Vytvořte nový prázdný formulář nazvaný [zDialogy]PRELIM_QueryOrder.NWA.
2. Zkopírujte vše z původního formuláře do tohoto nového formuláře. Nezapomeňte vše umístit na správnou stránku, rovněž zkopírujte metodu formuláře.
3. Poklepejte na text Začíná: a dejte mu název `StartsWith`.
4. Vyberte text Začíná: a proměnnou PRELIM_sQueryValue variable.
5. Zkopírujte tyto dvě položky, duplikujte je a zarovnejte duplikáty pod originály.
6. Změňte duplikovaný text na Do: a dejte mu název objektu `QueryTo`.
7. Zarovnejte text v objektu doprava
8. Přejmenujte duplikovanou proměnnou na `PRELIM_sQueryValueTo` a dejte jí název objektu `QueryValueTo`.
9. Vyberte text Do: a duplikujte jej.
10. Změňte v duplikovaném textu text na Od: a přidejte název objektu `QueryFrom`.
11. Optimalizujte velikost objektu Od:.
12. Umístěte text Od: na vrch textu Začíná: a zarovnejte je.
13. Vytvořte metodu projektu PRELIM_QueryMethod následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: PRELIM_QueryMethod
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Zobrazí nebo uschová objekty formuláře v závislosti na typu dat.

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($LCurrentTable)
C_LONGINT($LFieldType)
C_POINTER($pQueryField)
```

```
$LCurrentTable:=Table(pTable)

QUERY([zStruktura];[zStruktura]ČísloTabulky = $LCurrentTable;*)
QUERY([zStruktura];[zStruktura]NázevZástupce =
PRELIM_asQuery{PRELIM_asQuery})
$pQueryField:=Field($LCurrentTable;[zStruktura]ČísloPole)

    ` Jaký je typ pole
$LFieldType:=Type($pQueryField->)

If (($LFieldType = Is Alpha Field) | ($LFieldType = Is Text) | ($LFieldType = Is
Boolean))
  SET VISIBLE(*;"StartsWith";True)
  SET VISIBLE(*;"Query@";False)

Else
  SET VISIBLE(*;"StartsWith";False)
  SET VISIBLE(*;"Query@";True)

End if
    `Konec metody
```

14. Vytvořte metodu objektu PRELIM_asQuery na formuláři [zDialogy]PRELIM_QueryOrder.NWA následovně:

```
If (False)
    ` Metoda: PRELIM_asQuery [zDialogy]PRELIM_QueryOrder.NWA
    ` Kurz programovani ACI University
    ` Genericke metody z nastroju ACI
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Zobrazí či uschová objekty formuláře v závislosti na typu pole.

  <>fGeneric:=True
  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

PRELIM_QueryMethod
    `Konec metody
```

15. Upravte metodu formuláře [zDialogy]PRELIM_QueryOrder.NWA následovně:

```
If (False)
    ` Metoda formuláře: [zDialogy]PRELIM_QueryOrder.NWA
    ` Kurz programovani ACI University
```

```
…
        ` Naplní nabídku hodnotami
   PRELIM_asQuery:=1
   PRELIM_sQueryValue:=""
❖  PRELIM_sQueryValueTo:=""
❖  PRELIM_QueryMethod

        `Získá pořadí polí pro nabídku
…
```

16. Nahraďte metodu projektu PRELIM_CustomQueryOrder následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: PRELIM_CustomQueryOrder
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Zobrazí pro uživatele vlastní dialog dotazů

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($LWidth)          ` Šířka dialogu
C_LONGINT($LHeight)             ` Výška dialogu
C_LONGINT($LCurrentTable)   ` Číslo platné tabulky
C_LONGINT($LFieldType)      ` Typ pole
C_LONGINT($LWindowID)       ` ID okna
C_POINTER($pQueryField)     ` Ukazatel na pole dotazů

$LWidth:=265
$LHeight:=260
$LWindowID:=WIN_LNewWindow ($LWidth;$LHeight;Upper centered;Modal dialog box)

DIALOG([zDialogy];"PRELIM_QueryOrder.NWA")
CLOSE WINDOW

If (PRELIM_bgQuery# 3)
   $LCurrentTable:=Table(pTable)
   Case of
      ś (PRELIM_bgQuery          = 4)
```

```
   $pQueryField:=ALIAS_pFieldFromAliasName(PRELIM_asQuery{PRELIM_asQuery
}; $LCurrentTable)

         ` Jaký je typ pole
      $LFieldType:=Type($pQueryField->)
      Case of
         ś ($LFieldType = Is Alpha Field) | ($LFieldType= Is Text)
            QUERY(pTable->;$pQueryField->= PRELIM_sQueryValue+"@")

         ś ($LFieldType = Is Real) | ($LFieldType = Is LongInt) | ($LFieldType = Is
Integer)
            If(Length(PRELIM_sQueryValueTo)>0)
               QUERY(pTable->;$pQueryField->>=Num(PRELIM_sQueryValue);*)
               QUERY(pTable->; & ;$pQueryField-><=Num(PRELIM_sQueryValueTo))
            Else
               QUERY(pTable->;$pQueryField-> = Num(PRELIM_sQueryValue))
            End if

         ś ($LFieldType= Is Date)
            If(Length(PRELIM_sQueryValueTo)>0)
               QUERY(pTable->;$pQueryField->>=Date(PRELIM_sQueryValue);*)
               QUERY(pTable->; & ;$pQueryField-><=Date(PRELIM_sQueryValueTo))
            Else
               QUERY(pTable->;$pQueryField-> = Date(PRELIM_sQueryValue))
            End if

         ś ($LFieldType = Is Boolean)
            QUERY(pTable->;$pQueryField->= ($tQueryValue[[1]]="T");*)
            QUERY(pTable->; | ;$pQueryField->= ($tQueryValue[[1]]="F"))

         ś ($LFieldType= Is Time)
            If(Length(PRELIM_sQueryValueTo)>0)
               QUERY(pTable->;$pQueryField->>=Time(PRELIM_sQueryValue);*)
               QUERY(pTable->; & ;$pQueryField-><=Time(PRELIM_sQueryValueTo))
            Else
               QUERY(pTable->;$pQueryField-> = Time(PRELIM_sQueryValue))
            End if
      End case

   ś (PRELIM_bgQuery =2)
      M_GEN_QueryEditor

   Else
      M_GEN_ShowAllRecords

End case

      ` Potřebujeme třídit záznamy
If ((Records in selection(pTable->) >
0) & (PRELIM_asOrder{PRELIM_asOrder} # "** Žádné **"))
```

```
            ` Nalézt pole

   $pQueryField:=ALIAS_pFieldFromAliasName(PRELIM_asOrder{PRELIM_asOrder}
;
                              $LCurrentTable)
      ORDER BY(pTable->;$pQueryField->)
   End if
End if
      `Konec metody
```

17. Upravte původní metodu formuláře [zDialogy]PRELIM_QueryOrder následovně:

```
…
      ` Naplní nabídku hodnotami
   PRELIM_asQuery:=1
   PRELIM_sQueryValue:=""
   PRELIM_sQueryValueTo:=""

      `Získá pořadí polí pro nabídku
…
```

18. Vraťte se do prostředí Vlastní nabídky a testujte změny.

22.10. Zvětšení počtu polí dostupných pro dotazy a třídění záznamů

Bylo by pěkné kdybychom měli možnost mít větší výběr pro tabulky [Zákazníci] a [Faktury]. Náš dialog pro dotazy a třídění je založen na tom jestli je tabulka indexovaná nebo ne. Abychom zvětšili počet možností. Potřebujeme zvýšit počet indexů.

22.10.1. Zvýšení dostupnosti polí pro dialog

1. Indexujte následující pole:

   [Zákazníci]Firma
   [Zákazníci]Město
   [Zákazníci]Stát
   [Zákazníci]PSČ
   [Zákazníci]Důležitý
   [Zákazníci]CelkovéProdeje
   [Faktury]DatumFaktury
   [Faktury]FakturaCelkem

2. Proveďte metodu E_Set2ReadWrite z Prostředí uživatele
3. Přepněte se do tabulky [zStruktura].
4. Vymažte všechny záznamy této tabulky.
5. Proveďte metodu M_ALIAS_ModifyAliases z Prostředí uživatele nebo z prostředí Vlastní nabídky a obnovte vaše zástupné názvy.
6. Vyzkoušejte různé tabulky z prostředí Vlastní nabídky a zkoušejte výsledky pro výběry z dialogu dotazů a třídění.

23. Konvence názvů formulářů s příponami

Všimli jste si, že každá tabulka má formulář nazvaný Vstupní? Jestliže je máte všechny otevřeny v Prostředí návrháře, jak je rychle rozlišit?

Nyní když jsme se naučili něco více o názvech tabulek a polí, můžeme použít ukazatele na tabulky rovněž ke skládání názvů formulářů.

23.0.1 Vytvoření názvů formulářů jedinečných, ale stále generických

Jestliže máte 4D Insider můžete následující kód přemístit pomocí 4D Insider z knihovny FormName. Jinak:

1. Vytvořte metodu projektu GEN_SetFormName následujícím způsobem nebo importujte pomocí textového editoru:

```
 If (False)
      ` Metoda: GEN_SetFormName (pointer; string {;Asterisk})
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Nastaví názvy formulářů podle parametrů

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definovat parametry
C_POINTER($1)    ` Ukazatel na tabulku
C_STRING(10;$2)     ` Typ formuláře k nastavení

      ` Deklarace lokálních proměnných
C_POINTER($pTable)                ` Ukazatel na tabulku
C_STRING(10;$sFormToSet)              ` Typ formuláře k nastavení
C_STRING(31;$sTableName)

      ` Přiřazení parametrů k lokálním proměnným
$pTable:=$1
$sFormToSet:=$2
$sTableName:=Table name($pTable)

Case of
 ś (Count parameters = 3) & ($sFormToSet = "Vstupní")
      INPUT FORM($pTable->;$sTableName+"_v";*)

   ś (Count parameters = 3)
```

```
      INPUT FORM($pTable->;$sFormToSet;*)

   ś ($sFormToSet = "Vstupní")
      INPUT FORM($pTable->;$sTableName+"_v")

   ś ($sFormToSet = "Dotaz")
      INPUT FORM($pTable->;$sTableName+"_d")

   ś ($sFormToSet = "TiskVýstup")
      OUTPUT FORM($pTable->;$sTableName+"_ts")

   ś ($sFormToSet = "TiskVstupní")
      OUTPUT FORM($pTable->;$sTableName+"_v")

   ś ($sFormToSet = "Seznam")
      OUTPUT FORM($pTable->;$sTableName+"_s")

   ś ($sFormToSet = "ImportExport")
      OUTPUT FORM($pTable->;$sTableName+"_ie")

End case
   `Konec metody
```

2. Vraťte se do 4D a přejmenujte formuláře (Zákazníci, Faktury, Produkty) následovně:

Vstupní	Název tabulky_v	Zákazníci_v
Výstupní	Název tabulky_s	Zákazníci_s
TiskVýstup	Název tabulky_ts	Zákazníci_ts
Dotaz	Název tabulky_d	Zákazníci_d
ImportExport	Název tabulky_ie	Zákazníci _ie

3. S použitím 4D Insider najděte všechny výskyty příkazu INPUT FORM a OUTPUT FORM v databázi. V následujících metodách uvidíte použití těchto příkladů.

IMPEXP_ExportData	
IMPEXP_ExportDataOptimized	
GEN_ModifySelection	
M_GEN_QueryByForm	
M_SpecialReports	Upravte pouze druhý výskyt
P_IMPEXP_ImportData	
SalesByState	Upravte pouze druhý výskyt

4. Upravte všechny výskyty v databázi nalezené 4D Insider podle následujícího příkladu a tabulky v bodě 2:

Nahraďte: INPUT FORM(pTable->;"Vstupní";*)

Za: GEN_SetFormName(pTable;"Vstupní";"*")

Nahraďte: OUTPUT FORM(pTable->;"Výstupní")

Za: GEN_SetFormName(pTable;"Seznam")

5. Otestujte všechny změny.

24. Tisk vstupních formulářů

Tisk ze vstupního formuláře závisí na požadavku řešení a jsou možné dva způsoby. Buď tiskneme speciálně připravený formulář pro jeden záznam např. formulář faktury nebo pro dokumentaci vstupů tiskneme ten formulář, který je vidět na obrazovce. Výše jsme zvolili způsob druhý a pro tisk z jednoho záznamu jsme určili vstupní formulář. Proberme si teď ještě několik příkazů.

24.1. PRINT RECORD ({tabulka}{; }{*})

Parametr	Typ	Popis
tabulka	Tabulka	Tabulka pro kterou tisknout platný záznam nebo výchozí tabulka je-li vynecháno
*		Potlačí dialogová okna tiskárny

Příkaz PRINT RECORD tiskne platný záznam tabulky bez změny platného výběru. Pro tisk je zvolen současně vybraný výstupní formulář.

S pomocí příkazu PRINT RECORD můžete tisknout i vložené formuláře a externí oblasti. To není možné pomocí příkazu PRINT FORM.

POZNÁMKA: Jestliže byly provedeny úpravy do záznamu a tento nebyl uložen, tiskne tento příkaz upravené hodnoty polí a nikoliv hodnoty polí z disku.

24.2. Objekty ze stránky 0 nejsou tištěny

Libovolné objekty ze stránky pozadí formuláře, který tisknete se neobjeví na papíře. Toto musíte mít na paměti pokud plánujete některý formulář současně pro tisk. Jestliže chcete tisknout libovolnou položku umístěnou na stránce 0, musíte ji přemístit na stránku 1.

24.3. Úpravy vstupních formulářů pro tisk

24.3.1. Získání obrázkového tlačítka

1. Přejděte na libovolný výstupní formulář a zkopírujte tlačítko tisku tohoto formuláře.
2. Vložte tiskové tlačítko bPrintInput na dva vstupní formuláře pro Zákazníky a Faktury. Faktura má již svůj vlastní formulář tisku vytvořený v předchozím kurzu.

3. Vytvořte metodu objektu pro bPrintInput ve formuláři [Faktury]Faktury.v následovně:

```
If (False)
      ` Metoda: bPrintInput [Faktury]Invoices.i
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Ovládá tisk vstupního formuláře

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

OUTPUT FORM(pTable->;"Tisk")
PRINT RECORD(pTable->)
GEN_SetFormName(pTable;"Seznam")
   `Konec metody
```

4. Přesuňte objekty formuláře [Faktury]Faktury.v ze stránky 0 na stránku 1.
5. Vytvořte metodu objektu bPrintInput [Zákazníci]Zákazníci.v následovně:

```
If (False)
      ` Metoda: bPrintInput [Zákazníci]Customers.i
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Ovládá tisk vstupního formuláře

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

GEN_SetFormName(pTable;"TiskVstupní")
PRINT RECORD(pTable->)
GEN_SetFormName(pTable;"Seznam")
   `Konec metody
```

24.3. Tisk vícestránkových vstupních formulářů

Nyní jsme již zjistili, že vstupní formulář Zákazníci má sice dvě stránky, ale druhá stránka se netiskne. To je proto, že příkaz PRINT RECORD tiskne pouze první stránku formuláře. Pokud chcete tisknout vícestránkové formuláře musíte mít každou stránku na jejím vlastním tiskovém formuláři a tisknout každý z těchto formulářů.

Protože formuláře rovněž obsahují prvky na pozadí a rozhodně nebudeme chtít je mít duplikovány na každé stránce formuláře je tento koncept vytváření více tiskových formulářů vyhovující.

24.4. Copy List (hList) -> Nový seznam

Nelze zobrazit přesně tentýž hierarchický seznam v tutéž dobu na dvou různých formulářích, jestliže to zkusíte zkrachujete 4D. Jestliže zobrazujeme hierarchický seznam jako ovládání karty musíme proto nejlépe zobrazit vlastní seznam na každém formulář tj. v našem řešení Zákazníci.v.

Příkaz Copy list vytvoří v paměti nový seznam identický starému seznamu. Tento seznam lze pak použít pro druhé zobrazení. Nezapomeňte vymazat tento seznam, když skončíte.

24.4.1. Cvičení na tisk vstupních formulářů s více stránkami

1. Vytvořte pro zákazníky nový formulář nazvaný Stránka 1.
2. Zkopírujte vše ze stránky 1 formulář [Zákazníci]Zákazníci.v do vašeho nového formuláře [Zákazníci]Stránka1.
3. Zkopírujte vše ze stránky 0 formuláře [Zákazníci]Zákazníci.v do vašeho nového formuláře [Zákazníci]Stránka1.
4. Vytvořte nový formulář pro [Zákazníci] nazvaný Stránka2.
5. Zkopírujte vše ze stránky 2 formulář [Zákazníci]Zákazníci.v do vašeho nového formuláře [Zákazníci]Stránka2.
6. Zkopírujte vše ze stránky 0 formuláře [Zákazníci]Zákazníci.v do vašeho nového formuláře [Zákazníci]Stránka2.
7. Nahraďte metodu objektu bPrintInput na formuláři [Zákazníci]Zákazníci.v následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
    ` Metoda: bPrintInput [Zákazníci]Customers.v
    ` Kurz programovani ACI University
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Ovládá tisk vstupních formulářů

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

hlCustomerTabs2:=Copy list (hlCustomerTabs)
```

```
SELECT LIST ITEM (hlCustomerTabs2 ;1)      ` Nastaví záložku na stránku 1

OUTPUT FORM([Zákazníci];"Stránka1")
PRINT RECORD(pTable->)

SELECT LIST ITEM (hlCustomerTabs2 ; 2)      ` Nastaví záložku na stránku 2
RELATE MANY([Zákazníci]IDZákazníka)
OUTPUT FORM([Zákazníci];"Stránka2")
PRINT RECORD(pTable->;*)

GEN_SetFormName(pTable;"Seznam")

CLEAR LIST (hlCustomerTabs2 )
  `Konec metody
```

8. Právě jsme vytvořili rys, který nám dovoluje tisknout hierarchické seznamy nyní tuto část asi odstraníte, ale v budoucnosti lze tento kód využít.
9. Testujte váš nový tisk vstupního formuláře.

25. Tabulka obecných informací

Většinou je dobré vytvořit si určitý druh tabulky k uložení běžných informací o databázi. V mnoha databázích se tato tabulka nazývá buď Uživatel nebo Konstanty. Protože 4D používá název konstanty pro hodnoty, které se nikdy nemění a uživatel pro jednotlivce, který sedí za konkrétním stroje, nazvěme tuto tabulku [ObecnéInformace]. Účelem této tabulky je uložit údaje o databázi a firmě, která ji provozuje a případně využít tyto údaje pro různé druhy zpráv bez nutnosti natvrdo tyto údaje programovat. Jestliže se část těchto informací změní tak jako např. adresa, název, jednatel společnosti atd. tak protože jsou tyto údaje uloženy v databázi lze je měnit operativně s okamžitým efektem na všech dotčených místech.

25.0.1. Vytvoření tabuky [ObecnéInformace].

1. Vytvořte novou tabulku a nazvěte ji [ObecnéInformace].
2. Do této tabulky přidejte následující pole:

Název pole	**Typ**	**Vlastnosti**
NázevFirmy	Alfa 40	Nutný vstup, Indexované, Jedinečné
Adresa	Text	
Město	Alfa 22	
Stát	Alfa 2	
PSČ	Alfa 9	
Telefon	Alfa 10	

3. Vytvořte metodu projektu M_Preferences následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: M_Preferences
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Dovoluje uživateli upravit své předvolby

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
```

```
C_LONGINT($LTableState)

If ((Current user="Návrhář") | (Current user="Administrátor"))
   $LTableState:=LOCK_LSet2ReadWrite (->[ObecnéInformace];True)

   ALL RECORDS([ObecnéInformace])

   START TRANSACTION
   If (Records in selection([ObecnéInformace])=1)
      MODIFY RECORD([ObecnéInformace];*)
   Else
      ADD RECORD([ObecnéInformace])
   End if

   If (OK=1)
      VALIDATE TRANSACTION
   Else
      CANCEL TRANSACTION
   End if

   LOCK_Set2ReadOnly (->[ObecnéInformace];$LTableState)

Else
   ALERT("Nemáte oprávnění měnit předvolby!")
End if
      `Konec metody
```

4. Otevřete Záhlaví #1 a do nabídky Soubor přidejte následující položku:

Položka nabídky	Metoda
Vlastnosti polí a tabulek…	M_ALIAS_ModifyAliases
-	
Předvolby…	M_Preferences
-	
Konec	M_Quit

5. Upravte metodu INITIALIZE_GEN_ModifyVariables následovně:

```
   …
❖  ALL RECORDS([ObecnéInformace])
   fProcessAvailable:=False
   …
```

6. Upravte metodu MyStartup následovně:

```
   …
❖  ALL RECORDS([ObecnéInformace])
```

7. Přejděte do Prostředí uživatele a vytvořte výchozí formuláře.
8. Restartujte databázi a přidejte záznam Obecné informace.

26. Úplnné řízení tisku

Často se stává, že potřebujete tisknout něco co prostě není možné tisknout pomocí příkazů REPORT, PRINT LABEL, PRINT RECORD nebo PRINT SELECTION. Pro tyto případy potřebujete úplnnou kontrolu nad tištěnou stránkou.

Dále si vytvoříme zprávu, která bude zprávou o produktech dělená po jednotlivých kategoriích. Tato zpráva bude na začátku určovat všechny kategorie titulů, které jsme prodali, počet titulů v každé kategorii a procenta z celkového počtu, která reprezentují. Kromě toho počet různých titulů skutečně prodaných a procenta z celkových prodejů, které toto číslo reprezentuje.

V druhé části zprávy budeme rozlišovat podle státu, kategorie, kolik bylo prodáno, počet prodaných jednotek, počet různých zbylých titulů, celkové tržby, procento tržeb za stát na kategorii a procento z celkových tržeb.

ACI Video

Zpráva mix produktů

Tato zpráva je pro faktury od 1.1.1999 do 11.11.1999

Kategorie filmů		Celkové mn.	% z celkem	# Prodáno	% Mn. prodár
Akční		19	1,85	5	23
Děti		84	8,20		00
Drama		618	60,29	9	42
Fitness		5	,49		00
Horor		26	2,54		00
Komedie		73	7,12		00
Sport		10	,98		00
Umění/Hudba		115	11,22		00
Vzdělání		75	7,32	7	33
Kategorie celkem		1025		21	
Stát: CA	**Počet jedn.**	**Počet titulů**	**Částka**	**% ve státě**	**% celkem**
Drama	8	2	234,80 Kč	18,42	15
Vzdělání	5	1	1 040,00 Kč	81,58	69
Totals for CA	13	3	1 274,80 Kč		85
Stát: OR	**Počet jedn.**	**Počet titulů**	**Částka**	**% ve státě**	**% celkem**

- Jak lze toto provést pomocí PRINT SELECTION?

26.1. PRINT FORM ({tabulka; }formulář)

Parametr	Typ	Popis
tabulka	Tabulka	Tabulka ke které patří formulář nebo výchozí tabulka je-li vynechána
formulář	Řetězec	Formulář k tisku

Příkaz PRINT FORM vytiskne formulář s platnými hodnotami polí a proměnných. Tento příkaz tiskne pouze oblast obsahu (oblast mezi čárou záhlaví a čárou obsahu) formuláře. Je obvykle používán k tisku velmi složitých zpráv, které požadují úplnné řízení během tiskového procesu. PRINT FORM neprovádí žádné operace se záznamy, zlomy, uvolňování stránek, záhlaví nebo zápatí. Všechny tyto operace jsou ve vaší zodpovědnosti. Příkaz PRINT FORM tiskne pole a proměnné pouze v oblastech o pevné velikosti.

Protože PRINT FORM neprovádí uvolňování papíru po tisku formuláře je zde jednoduché kombinovat různé formuláře na téže stránce. PRINT FORM je tedy výborným nástrojem pro složité tiskové úlohy, které zahrnují různé tabulky a různé formuláře. Pokud chcete vynutit uvolnění papíru mezi jednotlivými formuláři použijte příkaz PAGE BREAK.

Dialogová okna tiskárny se neobjeví při použití příkazu PRINT FORM. Zpráva nepoužívá nastavení vzhledu stránky, které bylo přiřazeno v Prostředí návrháře, když byl formulář vytvářen. Jestliže chcete, aby se objevili dialogová okna tiskárny, musíte zahrnout do kódu příkazy PRINT SETTING nebo PAGE SETUP před sérii příkazů PRINT FORM.

Příkaz PRINT FORM sestavuje každou tištěnou stránku v paměti. Každá stránka je tištěna, když je v paměti zaplněna. Aby jste zajistili vytištění poslední stránky po použití příkazu PRINT FORM musíte uzavřít sekci příkazem PAGE BREAK. Jinak poslední stránka zůstane v paměti, čeká na zaplnění a nebude tištěna.

Podformuláře a externí oblasti nejsou tisknuty v příkaze PRINT FORM.

Příkaz PRINT FORM provádí pro metodu formuláře pouze události Při zavedení a Při tisku obsahu.

26.2. PAGE BREAK {(* | >)}

Parametr	Typ	Popis
* | > nebo		* Zruší tiskovou úlohu započatou pomocí PRINT FORM
		> Zvýší prioritu této tiskové úlohy

Tento příkaz tiskne data, která byla odeslána na tiskárnu a uvolní stránku z tiskárny. PAGE BREAK je používán spolu s příkazem PRINT FORM a to k zajištění konce stránky a tisku poslední stránky. Nepoužívejte PAGE BREAK spolu s příkazem PRINT SELECTION. Místo toho zde používejte příkazy SUBTOTAL nebo BREAK LEVEL s volitelnými parametry ke generování zlomů stránek (nových stránek).

Parametry * a > jsou oba volitelné.

Parametr * vám dovolí zrušit tiskovou úlohu započatou příkazem PRINT FORM provedení tohoto příkazu okamžitě zastaví běžící tiskovou úlohu (stránky, které byly již odeslány na tiskárnu budou vytištěny).

Parametr > upravuje způsob jakým se příkaz PAGE BREAK chová. Tato syntaxe má dvojí efekt:

- Drží tiskovou úlohu otevřenou dokud není opět volán příkaz PAGE BREAK bez parametru.
- Přidělí tiskové úloze prioritu takže nemůže být provedena žádná další tisková úloha dokud tato tisková úloha neskončí.

Druhý volitelný parametr je užitečný zvláště při užití tiskové úlohy s tiskem na pozadí (spooling). Parametr > zajistí, že tisková úloha bude uložena do jednoho souboru. To samozřejmě sníží čas tisku.

26.3. PAGE SETUP ({tabulka; }formulář)

Parametr	Typ	Popis
tabulka	Tabulka	Tabulky vlastnící formulář nebo výchozí tabulka je-li vynecháno
formulář	Řetězec	Formulář, který má být užit pro nastavení vzhledu stránky

Příkaz PAGE SETUP nastaví vzhled stránky pro tiskárnu na vzhled stránky uložený s formulářem, nastavení vzhledu stránky je uloženo spolu s formulářem při jeho vytvoření v Prostředí návrháře. Příkaz PAGE SETUP může být použit před PRINT SELECTION.

Používejte příkaz PAGE SETUP před PRINT FORM.

26.4. PRINT SETTINGS

Příkaz PRINT SETTINGS zobrazí dialogová okna tiskárny. Nejdříve je zobrazeno dialogové okno vzhledu stránky. Následně je zobrazeno dialogové okno tiskové úlohy. Měli by jste zahrnout příkaz PRINT SETTINGS před každou skupinu příkazů PRINT FORM. Příkaz PRINT SETTINGS nemá žádný efekt na PRINT SELECTION nebo PRINT LABEL.

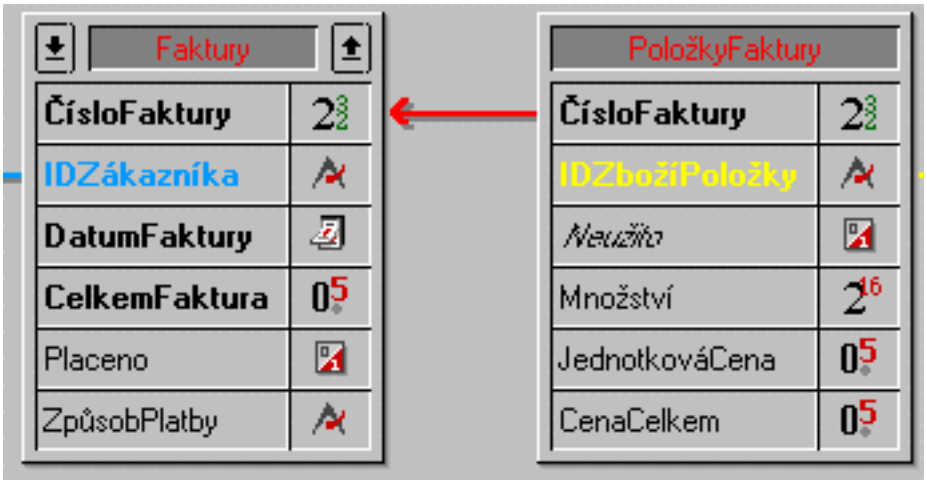
Dialogové okno tiskové úlohy obsahuje zaškrtávací políčko Tisk na obrazovku, které dovolí uživateli určit jestli chce tisknout náhled na obrazovku.

26.5. RELATE MANY SELECTION (pole)

Parametr	Typ	Popis
pole	Pole	Pole ze souboru skupin (ze kterého začíná vztah)

Příkaz RELATE MANY SELECTION vytvoří výběr záznamů v tabulce skupin založený na výběru záznamů v tabulce jedinců.

Uvažujme naši strukturu:



RELATE MANY SELECTION ([PoložkyFaktury]ČísloFaktury)

Pro tabulku [Faktury] pro platný výběr v použitém procesu nalezne příkaz RELATE MANY SELECTION všechny záznamy z tabulky [PoložkyFaktury], které jsou vztažené k záznamům ve výběru tabulky [Faktury].

26.6. RELATE ONE SELECTION (tabulka skupin; tabulka jedinců)

Parametr	Typ	Popis
Tabulka skupin	Tabulka	Název tabulky skupin (z které vztah začíná)
Tabulka jedinců	Tabulka	Název tabulky jedinců (do které vztah vede)

Příkaz RELATE ONE SELECTION vytvoří nový výběr záznamů v tabulce jedinců založený na výběru záznamů v tabulce skupin. Tento příkaz může být použit pouze tehdy existuje-li vztah. Vztahy mohou být manuální nebo automatické.

Tento příkaz může pracovat přes několik úrovní vztahů. Mezi použitou tabulkou jedinců a tabulkou skupin může být několik dalších vložených tabulek. Jediné co je nutné je, aby byl mezi nimi vztah stále ve stejném směru..

26.7. Chování příkazů RELATE MANY SELECTION a RELATE ONE SELECTION .

Existují dva různé způsoby pro vytvoření výběru přes vztahy. Následující dva příklady kódu je dokumentují. Oba ve svém výsledku provádějí totéž pouze jinými metodami (oba pouze předpokládají, že existuje platný výběr v tabulce [PoložkyFaktury]).

```
CREATE EMPTY SET([Produkty];"ProductsSold")
FIRST RECORD([PoložkyFaktury])
While (Not(End selection([PoložkyFaktury])
   RELATE ONE([PoložkyFaktury]IDZbožíProdukty)
   ADD TO SET([Produkty];"ProductsSold")
   NEXT RECORD([PoložkyFaktury])
End while
```

Je funkčně stejné jako:

```
RELATE ONE SELECTION ([PoložkyFaktury]; [Produkty])
```

Následující dva příklady kódu provádějí opět tutéž úlohu (oba předpokládají, že existuje platný výběr v tabulce [Produkty]).

```
CREATE EMPTY SET([PoložkyFaktury];"TotalLineItems")
FIRST RECORD([Produkty])
While (Not(End selection([Produkty])
   RELATE MANY([Produkty]IDZboží)
   CREATE SET([PoložkyFaktury];"TempSet")
   UNION ("TotalLineItems";"TempSet";"TotalLineItems")
   CLEAR SET("TempSet")
   NEXT RECORD([PoložkyFaktury])
End while
```

Je funkčně totéž jako:

```
RELATE MANY SELECTION ([Produkty]IDZboží)
```

Který z nich je rychlejší?

Odpověď není zase tak jednoduchá, aby se mohla omezit na ten nebo onen. Jestliže výběr záznamů v tabulce jedinců (v jednom případě cílový, v druhém zdrojový) je menší než polovina záznamů v tabulce, je RELATE ONE SELECTION a RELATE MANY SELECTION rychlejší.

Na druhou stranu, jestliže výběr záznamů v tabulce jedinců je více než polovina záznamů v tabulce, je část kódu s ADD TO SET rychlejší.

Pokud chcete optimalizovat chování vaší databáze, měli by jste kontrolovat co děláte a větvit svůj kód podle toho.

26.8. Vytvoření systému s PRINT FORM

Klíčem k fungujícímu systému PRINT FORM je mít dopředu sestavené jednotlivé komponenty a pak je vybírat a skládat do konečného výsledku, který potřebujeme. Když již budete mít vytvořeny tyto komponenty každá další úloha PRINT FORM se bude vytvářet jednodušeji než ty předchozí.

26.8.1. Úpravy existujícího kódu ke sdílení

Z předchozích kurzů máme již vytvořen určitý kód v metodě ProdejPoStátech, který můžeme upravit tak, aby byl sdílen k vytvoření části našeho systému. Překopírujme část sdíleného kódu do obecné metody projektu a přepišme metodu ProdejPoStátech tak, aby používala obecné metody. Tato technika nám umožní lehčeji udržovat kód. Jestliže budeme v budoucnu potřebovat něco upravit, bude tato úprava potřebná pouze na jednom místě pro všechny zprávy.

Jestliže máte 4D Insider můžete si následující kód překopírovat z knihovny PrintFrm, pak pokračujte v krocích 24.8.2. a pak 24.8.5.

1. Vytvořte novou metodu projektu QueryInvoices následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: QueryInvoices
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Dovolí uživateli vložit rozsah datumů pro dotaz na faktury

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($LWidth)            ` Šířka okna
C_LONGINT($LHeight)               ` Výška okna
C_LONGINT($LWindowID)         ` ID okna
```

```
$LWidth:=210                        ` Požadovaná šířka otevíraného okna.
$LHeight:=185                       ` Požadovaná výška.
$LWindowID:=WIN_LNewWindow ($LWidth;$LHeight;Upper centered;Modal dialog
box)
DIALOG([zDialogy];"DotazDatumy")
CLOSE WINDOW

If ((OK=1) & (dFrom # !00/00/00!))
   Case of
      ś (rb1=1)                     ` Hledání jednoho datumu
         QUERY([Faktury];[Faktury]DatumFaktury =  dFrom)
         sMessage:="Tato zpráva je pro faktury ze dne: "+String(dFrom;1)

      ś (rb4=1)                        ` Dotaz na datumy od do
         QUERY([Faktury];[Faktury]DatumFaktury >= dFrom;*)     ` Začátek sestavování
dotazu
         QUERY([Faktury]; & ;[Faktury]DatumFaktury <= dTO)     ` Konec sestavení dotazu
a jeho provedení
         sMessage:="Tato zpráva je pro faktury od  "+String(dFrom;1)+" do
"+String(dTo;1)

      ś (rb2=1)                        ` Dotaz na datum a období před
         QUERY([Faktury];[Faktury]DatumFaktury <= dFrom)
         sMessage:="Tato zpráva je pro faktury před dnem včetně"+String(dFrom;1)

      Else                             ` Dotaz na datum a období po
         QUERY([Faktury];[Faktury]DatumFaktury >= dFrom)
         sMessage:="Tato zpráva je pro faktury po dni včetně "+String(dFrom;1)
   End case
End if
     `Konec metody
```

2. Nahraďte metodu ProdejPoStátech následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
    `Procedure: ProdejPoStátech
    `ACI University Programming Classes
    `Created By: Jim Steinman
    `Date: 1/17/97

    `Účel: Spustí zprávu se seznamem faktur po státech

  <>f_Version6x10:=True
  <>f_Version6x30:=True
  <>fJ_Steinman:=True

    ` Modified 3/17/97
  <>fK_Wilbur:=True

End if

C_LONGINT($LRecordsInSelection)
```

❖ QueryInvoices

```
$LRecordsInSelection:=Records in selection([Faktury])
If ($LRecordsInSelection > 0)
   ORDER BY([Faktury];[Zákazníci]Stát;>;[Faktury]DatumFaktury)
…
```

26.8.2. Přidejte novou zprávu do dialogu se speciálními zprávami

1. Upravte metodu formuláře [zDialogy]VybratZprávu následovně:

```
…
    ś (Current process = 1)          `Proces uživatele, aplikace
       DISABLE BUTTON(bReport)    `Tyto zprávy nejsou dostupné z úvodní
obrazovky
       DISABLE BUTTON(bLabel)
       DISABLE BUTTON(bChart)
❖      ARRAY STRING(40;asReports;2)
❖      asReports{1}:="Zprava Mix Produktu…"
❖      asReports{2}:="Prodej Po Státech…"
…
```

26.8.3. Vytvoření nových metod pro ovládání úloh PRINT FORM

1. Vytvořte metodu projektu nazvanou PRINT_PF_Footer, ale neimportujte zatím žádný kód. Tato metoda projektu a další metoda PRINT_PrintForm jsou svou podstatou rekursivní, jestliže by jste sem vložili kód příliš brzo, neproběhlo by správné rozeznání ostatních metod.
2. Vytvořte metodu projektu PRINT_PF_ColumnHeaders následujícím způsobem nebo importujte pomocí textového editoru:

```
   If (False)
      ` Metoda: PRINT_PF_ColumnHeaders(boolean)
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Tiskne záhlaví sloupců pro užití s úlohou PRINT FORM

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

   ` Definovat parametry
C_BOOLEAN($1;$fBigLine)          ` Vytisknout silnou čáru v horní části

   ` Deklarace lokálních proměnných
C_LONGINT($LPixelsToBottom)  ` Počet bodů odspodu stránky pro uvolnění stránky
   ` Přiřazení parametrů k lokálním proměnným
If (Count parameters > 0)
   $fBigLine:=$1
Else
   $fBigLine:=False
End if

If (fTallHeader)
   $LPixelsToBottom:=55
Else
   $LPixelsToBottom:=40
End if

If (LPixelCount+$LPixelsToBottom > 690)
      `Přemístit se na konec stránky
   PRINT_PF_Footer

      `Tisk hlavičky stránky 2
   PRINT FORM([zDialogy];"PRINT_PF_Page2Header")
   $fBigLine:=True
```

```
End if

If ($fBigLine)
   PRINT FORM([zDialogy];"PRINT_PF_BigLine")
Else
   PRINT FORM([zDialogy];"PRINT_PF_Line")
End if

   `Tisk hlavičky sloupců
If (fTallHeader)
   PRINT FORM([zDialogy];"PRINT_PF_ColumnHeaderTall")
Else
   PRINT FORM([zDialogy];"PRINT_PF_ColumnHeaderShort")
End if
PRINT FORM([zDialogy];"PRINT_PF_HairLine")
      `Konec metody
```

3. Vytvořte metodu projektu nazvanou PRINT_PrintForm následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: PRINT_PrintForm(string;longint)
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Tiskne formuláře potřebné pro zprávu pomocí PRINT FORM

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definovat parametry
C_POINTER($1;$pTable)                  ` Ukazatel na tabulku formuláře
C_STRING(31;$2;$sNameofForm)       ` Název tištěného formuláře
C_LONGINT($3;$LPixelsToBottom)          ` Počet bodů odspodu stránky k uvolnění
stránky

      ` Deklarace lokálních proměnných
C_LONGINT($LGapSize)                   ` Potřebný počet prázdných řádek
      ` Přiřazení parametrů k lokálním proměnným
$pTable:=$1
$sNameofForm:=$2
$LPixelsToBottom:=$3

$fNewPage:=False
```

```
If (LPixelCount + $LPixelsToBottom > 690)
      ` Začátek nové stránky
   PRINT_PF_ColumnHeaders (True)

   $fNewPage:=True

End if

      ` Tiskneme něco jiného než prázdné řádky
If ($sNameofForm # "PRINT_PF_Gap")
      PRINT FORM($pTable->;$sNameofForm)
Else
   If (Not($fNewPage))                  ` Začali jsme na nové stránce? Pokud ano,
nevynechat.
      $LGapSize:=$LPixelsToBottom/10
      If ($LGapSize > 0)
         For ($i;1;$LGapSize )               ` Velikost prázdného místa po 10 bodech
dokonce
            PRINT FORM([zDialogy];"PRINT_PF_Gap10")
         End for
      End if
      $LGapSize:=$LPixelsToBottom%10
      If ($LGapSize > 0)
         For ($i;1;$LGapSize )               ` Zbytek prázdného místa místo bodů ke konci
            PRINT FORM([zDialogy];"PRINT_PF_Gap")
         End for
      End if
      LPixelCount:=LPixelCount+$LPixelsToBottom
   End if
End if
      `Konec metody
```

4. Napište do PRINT_PF_Footer následující nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: PRINT_PF_Footer
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Tiskne oblast zápatí pro zprávy vytvářené pomocí PRINT FORM

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      `Přesun na konec stránky
```

```
$LGapSize:=690-LPixelCount
If ($LGapSize>0)
   PRINT_PrintForm (->[zDialogy];"PRINT_PF_Gap";$LGapSize)
End if

      `Tisk konce stránky
PRINT FORM([zDialogy];"PRINT_PF_HairLine")
PRINT FORM([zDialogy];"PRINT_PF_Footer")

PAGE BREAK
      `Konec metody
```

5. Vytvořte metodu projektu ZpravaMixProduktu následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: ZpravaMixProduktu
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Creates a Product Mix Report

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($i)                          `Čítač smyčky
C_LONGINT($j)                          `Čítač smyčky
C_LONGINT($LRecordsInSelection)        `Zaznamů ve výběru
C_LONGINT($LNumberOfProducts)          ` Celkový počet produktů
C_LONGINT($LNumberOfCategories) ` Celkový počet kategorií
C_LONGINT($LQuantityOfProductSold)     ` Celkové množství prodaných produktů
C_LONGINT($LQtySoldPerStateCategory) ` Celkové množství prodané na kategorii na
stát
C_LONGINT($LDifferentTitles)           ` Celkový počet různých titulů prodaných na stát
C_REAL($rTotalSold)                        ` Celkový prodejní obrat
C_REAL($rTotalStateSold)               ` Celkový prodejní obrat na stát
C_REAL($rStateSales)                       ` Kategorie prodané na stát

ARRAY STRING(15;$asCategories;0)
ARRAY STRING(2;$asStates;0)

QueryInvoices

If ((OK=1) & (dFrom # !00/00/00!))
   $LRecordsInSelection:=Records in selection([Faktury])
   If ($LRecordsInSelection>0)
```

```
ALL RECORDS([ObecnéInformace])  ` Zde získáváme název naší společnosti
PRINT SETTINGS                  ` Uživatel volí vzhled stránky
If (OK=1)
   `zvolen vzhled stránky, nyní úvodní stránku
   LPixelCount:=0                       ` Nastavení čítače bodů
   LPageNumber:=0                       ` Nastavení čísla stránky
   fTallHeader:=False
   sReportName:="Zpráva mix produktů"   ` Nastavení názvu zprávy

   CREATE SET([Faktury];"TotalInvoices")` Vytvoření sady ze všech
reportovaných faktur

   ` Získání zákazníků z těchto faktur
   If ($LRecordsInSelection<(Records in table([Faktury])/2))
      RELATE ONE SELECTION([Faktury];[Zákazníci])
   Else
      CREATE EMPTY SET([Zákazníci];"TotalCustomers")
      For ($i;1;$LRecordsInSelection)
         RELATE ONE([Faktury]IDZákazníka)
         ADD TO SET([Zákazníci];"TotalCustomers")")
         NEXT RECORD([Faktury])
      End for
      USE SET("TotalCustomers")
      CLEAR SET("TotalCustomers")
   End if

   DISTINCT VALUES([Zákazníci]Stát;$asStates)

   ` Získání položek faktur pro tyto faktury
   If ($LRecordsInSelection<(Records in table([Faktury])/2))
      RELATE MANY SELECTION([PoložkyFaktury]ČísloFaktury)
      CREATE SET([PoložkyFaktury];"TotalLineItems")
   Else
      FIRST RECORD([Faktury])
      CREATE EMPTY SET([PoložkyFaktury];"TotalLineItems")
      For ($i;1;$LRecordsInSelection)
         RELATE MANY([Faktury]ČísloFaktury)
         CREATE SET([PoložkyFaktury];"TempSet")
         UNION ("TotalLineItems";"TempSet";"TotalLineItems")
         CLEAR SET("TempSet")
         NEXT RECORD([Faktury])
      End for
   End if

   USE SET("TotalLineItems")

   $LQuantityOfProductSold:=Sum([PoložkyFaktury]Množství)      ` Získá
celkové prodané množství
   $rTotalSold:=Sum([PoložkyFaktury]CenaCelkem)

   ALL RECORDS([Produkty])
```

```
DISTINCT VALUES([Produkty]Kategorie;$asCategories)

   ` Vytvoření array kategorie
$LNumberOfCategories:=Size of array($asCategories)
      ` Kolik v této kategorii
ARRAY INTEGER($aiQuantityOfCategory;$LNumberOfCategories)
      ` Procenta z celků
ARRAY REAL($arCategoryPercentOfTotalProduct;$LNumberOfCategories)
      ` Kolik prodáno v této kategorii
ARRAY INTEGER($aiQuantityOfCategorySold;$LNumberOfCategories)
      ` Procenta z celkových prodejů
ARRAY REAL($arCategoryPercentOfTotalSold;$LNumberOfCategories)

$LNumberOfProducts:=Records in selection([Produkty])

For ($i;1;$LNumberOfCategories)
   QUERY([Produkty];[Produkty]Kategorie = $asCategories{$i})          ` Nalézt
každou kategorii

      ` Celkové množství pro kategorii
   $aiQuantityOfCategory{$i}:=Records in selection([Produkty])
      ` Výpočet procent pro všechny produkty
   $arCategoryPercentOfTotalProduct{$i}:=Round($aiQuantityOfCategory{$i}/
$LNumberOfProducts*100;2)

      ` Získání čísla prodáno v kategorii
   If ($arCategoryPercentOfTotalProduct{$i}<50)
      RELATE MANY SELECTION([PoložkyFaktury]IDZbožíPoložky)
   Else
         ` Chceme všechny položky faktur pro typ kategorie
      QUERY([PoložkyFaktury];[Produkty]Kategorie=$asCategories{$i})
   End if
   CREATE SET([PoložkyFaktury];"CategoryLineItems")

      ` Získání jedné kategorie v daném období

  INTERSECTION("CategoryLineItems";"TotalLineItems";"CategoryLineItems"+Strin
g($i))
   USE SET("CategoryLineItems"+String($i))
   CLEAR SET("CategoryLineItems")

      ` Nakonec máme prodané množství
   $aiQuantityOfCategorySold{$i}:=Sum([PoložkyFaktury]Množství)
      ` Výpočet procent z celkových prodejů
   $arCategoryPercentOfTotalSold{$i}:=Round($aiQuantityOfCategorySold{$i}/
$LQuantityOfProductSold*100;2)

End for

   ` Tiska hlavičky první stránky
```

```
PRINT_PrintForm (->[zDialogy];"PRINT_PF_MainHeader";0)

   ` Definice záhlaví sloupců
sDescriptionHead:="Kategorie filmů"
sColumnHead1:=""
sColumnHead2:="Celkové mn."
sColumnHead3:="% z celkem"
sColumnHead4:="# Prodáno"
sColumnHead5:="% Mn. prodáno"

PRINT_PF_ColumnHeaders (True)    ` Tisk začátku s tlustou čárou

   ` Tisk obsahu pro tuto kategori
For ($i;1;$LNumberOfCategories)
   sDescription:=$asCategories{$i}
   sColumn1:=""
   sColumn2:=String($aiQuantityOfCategory{$i};"#####")
   sColumn3:=String($arCategoryPercentOfTotalProduct{$i};"###,00")
   sColumn4:=String($aiQuantityOfCategorySold{$i};"#####")
   sColumn5:=String($arCategoryPercentOfTotalSold{$i};"### 00")

   PRINT_PrintForm (->[zDialogy];"PRINT_PF_Detail";18)
End for

   ` Tisk celkových součtů kategorie
sDescription:="Kategorie celkem"
sColumn1:=""
sColumn2:=String($LNumberOfProducts;"#####")
sColumn3:=""
sColumn4:=String($LQuantityOfProductSold;"#####")
sColumn5:=""
PRINT_PrintForm (->[zDialogy];"PRINT_PF_Totals";24)

For ($i;1;Size of array($asStates))
      ` Tiska hlavičky státu
   sDescriptionHead:="Stát: "+$asStates{$i}
   sColumnHead1:="Počet jedn."
   sColumnHead2:="Počet titulů"
   sColumnHead3:="Částka"
   sColumnHead4:="% ve státě"
   sColumnHead5:="% celkem"

   PRINT_PrintForm (->[zDialogy];"PRINT_PF_Gap";10)

      ` Tisk záhlaví nové sekce
   fTallHeader:=True
PRINT_PF_ColumnHeaders (False)    ` Tisk s normální čárou

   $LDifferentTitles:=0

   USE SET("TotalInvoices")
```

```
QUERY SELECTION([Faktury];[Zákazníci]Stát=$asStates{$i})

   ` Získání položek faktur pro faktury za stát
If ($LRecordsInSelection<(Records in table([Faktury])/2))
   RELATE MANY SELECTION([PoložkyFaktury]ČísloFaktury)
   CREATE SET([PoložkyFaktury];"StateLineItems")
Else
   CREATE EMPTY SET([PoložkyFaktury];"StateLineItems")
   For ($j;1;$LRecordsInSelection)
      RELATE MANY([Faktury]ČísloFaktury)
      CREATE SET([PoložkyFaktury];"TempSet")
      UNION ("StateLineItems";"TempSet";"StateLineItems")
      CLEAR SET("TempSet")
      NEXT RECORD([Faktury])
   End for
End if

USE SET("StateLineItems")

$rTotalStateSold:=Sum([PoložkyFaktury]CenaCelkem)
$LQtySoldPerStateCategory:=Sum([PoložkyFaktury]Množství)

For ($j;1;$LNumberOfCategories)
   INTERSECTION("StateLineItems";"CategoryLineItems"+String($j);
"StateCategoryLineItems")
   USE SET("StateCategoryLineItems")

      ` Jsou zde nějaké kategorie k tisku
   If (Records in selection([PoložkyFaktury])>0)
      sDescription:=$asCategories{$j}
      sColumn1:=String(Sum([PoložkyFaktury]Množství);"#####")
      RELATE ONE SELECTION([PoložkyFaktury];[Produkty])
      sColumn2:=String(Records in selection([Produkty]))
      $LDifferentTitles:=$LDifferentTitles+Num(sColumn2)
      $rStateSales:=Sum([PoložkyFaktury]CenaCelkem)
      sColumn3:=String($rStateSales;"|Částka")
      sColumn4:=String(Round($rStateSales/$rTotalStateSold*100;2);"###,00")
      sColumn5:=String(Round($rStateSales/$rTotalSold*100;2);"###.00")

      PRINT_PrintForm (->[zDialogy];"PRINT_PF_Detail";18)
   End if

   CLEAR SET("StateCategoryLineItems")
End for

   ` Tisk celkem stát
sDescription:="Totals for "+$asStates{$i}
sColumn1:=String($LQtySoldPerStateCategory;"#####")
sColumn2:=String($LDifferentTitles;"#####")
sColumn3:=String($rTotalStateSold;"|Částka")
sColumn4:=""
```

```
      sColumn5:=String(Round($rTotalStateSold/$rTotalSold*100;2);"###.00")
      PRINT_PrintForm (->[zDialogy];"PRINT_PF_Totals";25)

      CLEAR SET("StateLineItems")

   End for

      `Je čas pro součty
   PRINT_PrintForm (->[zDialogy];"PRINT_PF_BigLine";30)
   PRINT_PrintForm (->[zDialogy];"PRINT_PF_Gap";3)
   PRINT_PrintForm (->[zDialogy];"PRINT_PF_BigLine";0)

   USE SET("TotalLineItems")

   $LRecordsInSelection:=Records in selection([PoložkyFaktury])

   If ($LRecordsInSelection<(Records in table([PoložkyFaktury])/2))
      RELATE ONE SELECTION([PoložkyFaktury];[Produkty])
   Else
      CREATE EMPTY SET([Produkty];"ProductsSold")
      For ($i;1;$LRecordsInSelection)
         RELATE ONE([PoložkyFaktury]IDZbožíPoložky)
         ADD TO SET([Produkty];"ProductsSold")
         NEXT RECORD([PoložkyFaktury])
      End for
      USE SET("ProductsSold")
      CLEAR SET("ProductsSold")
   End if

   sDescriptionHead:="Celkový součet"
   sColumnHead1:=String($LQuantityOfProductSold;"#####")
   sColumnHead2:=String(Records in selection([Produkty]);"#####")
   sColumnHead3:=String($rTotalSold;"|Částka")
   sColumnHead4:=""
   sColumnHead5:=""

      ` Tisk součtů kontrolovaný pro konec předchozí stránky
   fTallHeader:=False
   PRINT_PrintForm (->[zDialogy];"PRINT_PF_ColumnHeaderShort";20)

      ` Vysuňte poslední stránku
   PRINT_PF_Footer

      ` Vymazat všechny sady
   CLEAR SET("TotalInvoices")
   CLEAR SET("TotalLineItems")
   For ($i;1;$LNumberOfCategories)
      CLEAR SET("CategoryLineItems"+String($i))
   End for
End if
```

```
    Else
      ALERT("Ve zvoleném období nebyly nalezeny žádné faktury!")
    End if
  End if
      `Konec metody
```

26.8.4. Vyzkoušení jednotlivých komponent PRINT FORM

1. Podívejme se na jednotlivé formuláře, které skládají části naší zprávy PRINT FORM. Existuje několik jednotlivých formulářů zde zahrnutých.

   [zDialogy]PRINT_PF_BigLine
   [zDialogy]PRINT_PF_Line
   [zDialogy]PRINT_PF_HairLine
   [zDialogy]PRINT_PF_ColumnHeaderShort
   [zDialogy]PRINT_PF_ColumnHeaderTall
   [zDialogy]PRINT_PF_Detail
   [zDialogy]PRINT_PF_Footer
   [zDialogy]PRINT_PF_GrandTotal
   [zDialogy]PRINT_PF_Page2Header
   [zDialogy]PRINT_PF_Totals
   [zDialogy]PRINT_PF_Gap

2. Prověřte typickou metodu formuláře pro libovolný z předchozích formulářů.

```
If (False)
      ` Form Method: PRINT_PF_Detail
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Nastaví oblast obsahu pro PRINT FORM

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

If (Form event=On Printing Detail)
   LPixelCount:=LPixelCount+17
End if
      `Konec metody
```

3. Prověřte metodu formuláře [zDialogy]PRINT_PF_Totals.

```
If (False)
      ` Form Method: PRINT_PF_Totals
      ` Kurz programovani ACI University
      ` Generic metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Nastaví oblast součtů pro  PRINT FORM

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

If (Form event=On Printing Detail)
   LPixelCount:=LPixelCount+23
   If (Length(sColumn1)=0)
      SET VISIBLE(*;"Line1";False)
   End if
   If (Length(sColumn2)=0)
      SET VISIBLE(*;"Line2";False)
   End if
   If (Length(sColumn3)=0)
      SET VISIBLE(*;"Line3";False)
   End if
   If (Length(sColumn4)=0)
      SET VISIBLE(*;"Line4";False)
   End if
   If (Length(sColumn5)=0)
      SET VISIBLE(*;"Line5";False)
   End if
End if
      `Konec metody
```

26.8.5. Vytváření formulářů komponent PRINT FORM.

1. Vytvořte nový prázdný formulář nazvaný [zDialogy]PRINT_PF_MainHeader.
2. Přesuňte řídící čáru H na vrch formuláře do bodu 0.
3. Přidejte do oblasti obsahu pole [ObecnéInformace]NázevFirmy .
4. Přidejte proměnnou `sReportName` do oblasti obsahu formuláře.
5. Přidejtě proměnnou `sMessage` do oblasti obsahu formuláře.
6. Přidejte libovolné další objekty, které chcete vidět v hlavičce vaší zprávy.
7. Přesuňte řídící čáry O, Z0 a P na spodní okraj vašeho formuláře hlavičky.
8. Změřte v bodech skutečné umístění řídící čáry O.

9. Vytvořte pro tento celý formulář bílou výplň, bílí rámeček na pozadí a pošlete objekt na pozadí (pozn.: tento krok není nezbytný pro všechny tisky formátů pomocí PRINT FORM. Záleží na ovladači tiskárny a operačním systému).
10. Vytvořte metodu formuláře [zDialogy]PRINT_PF_MainHeader následovně:

```
If (False)
      ` Form Method: PRINT_PF_MainHeader
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Nastaví oblast hlavičky zprávy tištěné  PRINT FORM

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

If (Form event=On Printing Detail)
   LPixelCount:=xxxx `Vaši přesnou velikost formátu hlavičky zde.
End if
      `Konec metody
```

11. Otestujte tisk zprávy Zpráva Mix Produktů.

26.8.6. Vytvořeního vlastního teploměru průběhu.

Dialogy, kde se rychle mění čísla na obrazovce jsou otravné. Uživatel je ale také nervózní, když nemá přehled o průběhu operace.

1. Vytvořte metodu projektu nazvanou GEN_ProgressThermometer následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: GEN_ProgressThermometer  (řetězec; real{; logické})
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Zobrazí vlastní teploměr průběhu
      ` Platná hodnota by měla být mezi 0 - 100.

      ` Pro inicializaci teploměru zahrňte do kódu následující řádek:
      ` GEN_ProgressThermometer("";0;True)

      ` K překreslení teploměru, zahrňte do kódu následující řádek:
      ` GEN_ProgressThermometer("message string";rValue)
```

```
      ` K vymazání teploměru zahrňte do kódu následující řádek:
      ` GEN_ProgressThermometer("";0;False)

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definovat parametry
C_STRING(80;$1)                  ` Zpráva k zobrazení
C_REAL($2;$rPercentComplete)     ` Současná hodnota naplňovaná od  0-100
C_BOOLEAN($3)                    ` Příznak k inicializaci či uzavření okna

      ` Deklarace lokálních proměnných
C_LONGINT($Lwid)    ` ID okna

      `Přiřadit  parametry k proměnným
sProgressMessage:=$1
$rPercentComplete:=$2

If (Count parameters=3)
   If ($3)                       ` Fáze inicializace
      $Lwid:=WIN_LNewWindow (410;70;Window centered;Modal dialog box)
      ALL RECORDS([ObecnéInformace])
      INPUT FORM([ObecnéInformace];"GEN_ProgressThermometer")
      gProgressThermometer:=<>gProgressThermometer-4
      DISPLAY RECORD([ObecnéInformace])
   Else                          ` Fáze vymazání
      CLOSE WINDOW
      INPUT FORM([ObecnéInformace];"Vstupní")
   End if
Else                             ` Fáze zobrazení
   gProgressThermometer:=<>gProgressThermometer+(4*$rPercentComplete)-4
   DISPLAY RECORD([ObecnéInformace])
End if
      `Konec metody
```

2. Nahraďte metodu projektu ZpravaMixProduktu následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: ZpravaMixProduktu
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Vytvoří zprávu Mix produkty

   <>f_Version6x30:=True
```

```
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($i)                        `Čítač smyčky
C_LONGINT($j)                        `Čítač smyčky
C_LONGINT($LRecordsInSelection)          `Zaznamů ve výběru
C_LONGINT($LNumberOfProducts)          ` Celkový počet produktů
C_LONGINT($LNumberOfCategories) ` Celkový počet kategorií
C_LONGINT($LQuantityOfProductSold)    ` Celkové množství prodaných produktů
C_LONGINT($LQtySoldPerStateCategory) ` Celkové množství prodané na kategorii na
stát
C_LONGINT($LNumberOfStates)       ` Celkový počet států
C_LONGINT($LDifferentTitles)       ` Celkový počet různých titulů prodaných na stát
C_REAL($rTotalSold)                    ` Celkový prodejní obrat
C_REAL($rTotalStateSold)             ` Celkový prodejní obrat na stát
C_REAL($rStateSales)                   ` Kategorie prodané na stát
C_REAL($rFractionPerLoop)          ` Zlomek dokončení na smyčku pro teploměr
C_REAL($rPercentComplete)          ` Skutečně dokončeno procent

ARRAY STRING(15;$asCategories;0)
ARRAY STRING(2;$asStates;0)

QueryInvoices

If ((OK=1) & (dFrom # !00/00/00!))
   $LRecordsInSelection:=Records in selection([Faktury])
   If ($LRecordsInSelection>0)

      ALL RECORDS([ObecnéInformace])   ` Zde získáváme název naší společnosti
      PRINT SETTINGS                   ` Uživatel volí vzhled stránky
      If (OK=1)
         `zvolen vzhled stránky, nyní úvodní stránku
        LPixelCount:=0                        ` Nastavení čítače bodů
        LPageNumber:=0                    ` Nastavení čísla stránky
        fTallHeader:=False
        sReportName:="Zpráva mix produktů"     ` Nastavení názvu zprávy

        MESSAGES OFF
        GEN_ProgressThermometer ("Probíhá výpočet počtu států.";0;True)

        CREATE SET([Faktury];"TotalInvoices")` Vytvoření sady ze všech
reportovaných faktur

         ` Získání zákazníků z těchto faktur
        If ($LRecordsInSelection<(Records in table([Faktury])/2))
          RELATE ONE SELECTION([Faktury];[Zákazníci])
        Else
          CREATE EMPTY SET([Zákazníci];"TotalCustomers")
          For ($i;1;$LRecordsInSelection)
```

```
            RELATE ONE([Faktury]IDZákazníka)
            ADD TO SET([Zákazníci];"TotalCustomers")")
            NEXT RECORD([Faktury])
        End for
        USE SET("TotalCustomers")
        CLEAR SET("TotalCustomers")
    End if

    DISTINCT VALUES([Zákazníci]Stát;$asStates)

        ` Získání položek faktur pro tyto faktury
    If ($LRecordsInSelection<(Records in table([Faktury])/2))
        RELATE MANY SELECTION([PoložkyFaktury]ČísloFaktury)
        CREATE SET([PoložkyFaktury];"TotalLineItems")
    Else
        FIRST RECORD([Faktury])
        CREATE EMPTY SET([PoložkyFaktury];"TotalLineItems")
        For ($i;1;$LRecordsInSelection)
            RELATE MANY([Faktury]ČísloFaktury)
            CREATE SET([PoložkyFaktury];"TempSet")
            UNION ("TotalLineItems";"TempSet";"TotalLineItems")
            CLEAR SET("TempSet")
            NEXT RECORD([Faktury])
        End for
    End if

    USE SET("TotalLineItems")

    $LQuantityOfProductSold:=Sum([PoložkyFaktury]Množství)      ` Získá
celkové prodané množství
    $rTotalSold:=Sum([PoložkyFaktury]CenaCelkem)

    ALL RECORDS([Produkty])
    DISTINCT VALUES([Produkty]Kategorie;$asCategories)
❖   GEN_ProgressThermometer ("Probíhá výpočet kategorií.";0)

        ` Vytvoření array kategorie
    $LNumberOfCategories:=Size of array($asCategories)
        ` Kolik v této kategorii
    ARRAY INTEGER($aiQuantityOfCategory;$LNumberOfCategories)
        ` Procenta z celků
    ARRAY REAL($arCategoryPercentOfTotalProduct;$LNumberOfCategories)
        ` Kolik prodáno v této kategorii
    ARRAY INTEGER($aiQuantityOfCategorySold;$LNumberOfCategories)
        ` Procenta z celkových prodejů
    ARRAY REAL($arCategoryPercentOfTotalSold;$LNumberOfCategories)

    $LNumberOfProducts:=Records in selection([Produkty])
❖
❖       ` Příprava dat pro teploměr průběhu
❖   $LNumberOfStates:=Size of array($asStates)
```

```
❖                $rFractionPerLoop:=100/($LNumberOfCategories*($LNumberOfStates+2))
❖                $rPercentComplete:=0

                 For ($i;1;$LNumberOfCategories)
                    QUERY([Produkty];[Produkty]Kategorie = $asCategories{$i})          ` Nalézt
              každou kategorii

                    ` Celkové množství pro kategorii
                   $aiQuantityOfCategory{$i}:=Records in selection([Produkty])
                    ` Výpočet procent pro všechny produkty
                   $arCategoryPercentOfTotalProduct{$i}:=Round($aiQuantityOfCategory{$i}/
                         $LNumberOfProducts*100;2)

                    ` Získání čísla prodáno v kategorii
                   If ($arCategoryPercentOfTotalProduct{$i}<50)
                    RELATE MANY SELECTION([PoložkyFaktury]IDZbožíProdukty)
                   Else
                      ` Chceme všechny položky faktur pro typ kategorie
                    QUERY([PoložkyFaktury];[Produkty]Kategorie=$asCategories{$i})
                   End if
                   CREATE SET([PoložkyFaktury];"CategoryLineItems")

                    ` Získání jedné kategorie v daném období

                 INTERSECTION("CategoryLineItems";"TotalLineItems";"CategoryLineItems"+Strin
              g($i))
                   USE SET("CategoryLineItems"+String($i))
                   CLEAR SET("CategoryLineItems")

                    ` Nakonec máme prodané množství
                   $aiQuantityOfCategorySold{$i}:=Sum([PoložkyFaktury]Množství)
                    ` Výpočet procent z celkových prodejů
                   $arCategoryPercentOfTotalSold{$i}:=Round($aiQuantityOfCategorySold{$i}/
                         $LQuantityOfProductSold*100;2)
❖
❖                    `Obnova teploměru průběhu
❖                  $rPercentComplete:=$rPercentComplete+$rFractionPerLoop
❖                    GEN_ProgressThermometer ("Výpočet součtů pro kategorii"
;$rPercentComplete)

                   End for

                    ` Tisk hlavičky první stránky
❖                    GEN_ProgressThermometer ("Tisk součtů pro kategorii";$rPercentComplete)

                   PRINT_PrintForm (->[zDialogy];"PRINT_PF_MainHeader";0)

                    ` Definice záhlaví sloupců
                  sDescriptionHead:="Kategorie filmů"
                  sColumnHead1:=""
                  sColumnHead2:="Celkové mn."
```

```
                    sColumnHead3:="% z celkem"
                    sColumnHead4:="# Prodáno"
                    sColumnHead5:="% Mn. prodáno"

                    PRINT_PF_ColumnHeaders (True)    ` Tisk začátku s tlustou čárou

                       ` Tisk obsahu pro tuto kategori
                    For ($i;1;$LNumberOfCategories)
                       sDescription:=$asCategories{$i}
                       sColumn1:=""
                       sColumn2:=String($aiQuantityOfCategory{$i};"#####")
                       sColumn3:=String($arCategoryPercentOfTotalProduct{$i};"###,00")
                       sColumn4:=String($aiQuantityOfCategorySold{$i};"#####")
                       sColumn5:=String($arCategoryPercentOfTotalSold{$i};"### 00")

                       PRINT_PrintForm (->[zDialogy];"PRINT_PF_Detail";18)
❖
❖                            `Obnova průběhu teploměru
❖                      $rPercentComplete:=$rPercentComplete+$rFractionPerLoop
❖                      GEN_ProgressThermometer ("Tisk kategorie
"+sDescription;$rPercentComplete)

                    End for

                       ` Tisk celkových součtů kategorie
                    sDescription:="Kategorie celkem"
                    sColumn1:=""
                    sColumn2:=String($LNumberOfProducts;"#####")
                    sColumn3:=""
                    sColumn4:=String($LQuantityOfProductSold;"#####")
                    sColumn5:=""
                    PRINT_PrintForm (->[zDialogy];"PRINT_PF_Totals";24)

                    For ($i;1;Size of array($asStates))
                          ` Tiska hlavičky státu
                       sDescriptionHead:="Stát: "+$asStates{$i}
                       sColumnHead1:="Počet jedn."
                       sColumnHead2:="Počet titulů"
                       sColumnHead3:="Částka"
                       sColumnHead4:="% ve státě"
                       sColumnHead5:="% celkem"

                       PRINT_PrintForm (->[zDialogy];"PRINT_PF_Gap";10)

                          ` Tisk záhlaví nové sekce
                       fTallHeader:=True
                    PRINT_PF_ColumnHeaders (False)    ` Tisk s normální čárou

                       $LDifferentTitles:=0

                       USE SET("TotalInvoices")
```

```
                        QUERY SELECTION([Faktury];[Zákazníci]Stát=$asStates{$i})

                          ` Získání položek faktur pro faktury za stát
                        If ($LRecordsInSelection<(Records in table([Faktury])/2))
                          RELATE MANY SELECTION([PoložkyFaktury]ČísloFaktury)
                          CREATE SET([PoložkyFaktury];"StateLineItems")
                        Else
                          CREATE EMPTY SET([PoložkyFaktury];"StateLineItems")
                          For ($j;1;$LRecordsInSelection)
                            RELATE MANY([Faktury]ČísloFaktury)
                            CREATE SET([PoložkyFaktury];"TempSet")
                            UNION ("StateLineItems";"TempSet";"StateLineItems")
                            CLEAR SET("TempSet")
                            NEXT RECORD([Faktury])
                          End for
                        End if

                        USE SET("StateLineItems")

                        $rTotalStateSold:=Sum([PoložkyFaktury]CenaCelkem)
                        $LQtySoldPerStateCategory:=Sum([PoložkyFaktury]Množství)

                        For ($j;1;$LNumberOfCategories)
                          INTERSECTION("StateLineItems";"CategoryLineItems"+String($j);
                                       "StateCategoryLineItems")
                          USE SET("StateCategoryLineItems")

                            ` Jsou zde nějaké kategorie k tisku
                          If (Records in selection([PoložkyFaktury])>0)
                            sDescription:=$asCategories{$j}
                            sColumn1:=String(Sum([PoložkyFaktury]Množství);"#####")
                            RELATE ONE SELECTION([PoložkyFaktury];[Produkty])
                            sColumn2:=String(Records in selection([Produkty]))
                            $LDifferentTitles:=$LDifferentTitles+Num(sColumn2)
                            $rStateSales:=Sum([PoložkyFaktury]CenaCelkem)
                            sColumn3:=String($rStateSales;"|Částka")
                            sColumn4:=String(Round($rStateSales/$rTotalStateSold*100;2);"###,00")
                            sColumn5:=String(Round($rStateSales/$rTotalSold*100;2);"###.00")

                            PRINT_PrintForm (->[zDialogy];"PRINT_PF_Detail";18)
                          End if

                          CLEAR SET("StateCategoryLineItems")
❖
❖                           `Obnovat teploměr průběhu
❖                       $rPercentComplete:=$rPercentComplete+$rFractionPerLoop
❖                       GEN_ProgressThermometer ("Tisk kategorie "+sDescription+" pro stát  "
+sDescriptionHead;$rPercentComplete)

                        End for
```

```
      ` Tisk celkem stát
   sDescription:="Totals for "+$asStates{$i}
   sColumn1:=String($LQtySoldPerStateCategory;"#####")
   sColumn2:=String($LDifferentTitles;"#####")
   sColumn3:=String($rTotalStateSold;"|Částka")
   sColumn4:=""
   sColumn5:=String(Round($rTotalStateSold/$rTotalSold*100;2);"###.00")
   PRINT_PrintForm (->[zDialogy];"PRINT_PF_Totals";25)

   CLEAR SET("StateLineItems")

End for

   `Je čas pro součty
PRINT_PrintForm (->[zDialogy];"PRINT_PF_BigLine";30)
PRINT_PrintForm (->[zDialogy];"PRINT_PF_Gap";3)
PRINT_PrintForm (->[zDialogy];"PRINT_PF_BigLine";0)

USE SET("TotalLineItems")

$LRecordsInSelection:=Records in selection([PoložkyFaktury])

      `Obnova teploměru průběhu
   GEN_ProgressThermometer ("Dokončování zprávy";100)


If ($LRecordsInSelection<(Records in table([PoložkyFaktury])/2))
   RELATE ONE SELECTION([PoložkyFaktury];[Produkty])
Else
   CREATE EMPTY SET([Produkty];"ProductsSold")
   For ($i;1;$LRecordsInSelection)
      RELATE ONE([PoložkyFaktury]IDZbožíProdukty)
      ADD TO SET([Produkty];"ProductsSold")
      NEXT RECORD([PoložkyFaktury])
   End for
   USE SET("ProductsSold")
   CLEAR SET("ProductsSold")
End if

sDescriptionHead:="Celkový součet"
sColumnHead1:=String($LQuantityOfProductSold;"#####")
sColumnHead2:=String(Records in selection([Produkty]);"#####")
sColumnHead3:=String($rTotalSold;"|Částka")
sColumnHead4:=""
sColumnHead5:=""

   ` Tisk součtů kontrolovaný pro konec předchozí stránky
fTallHeader:=False
PRINT_PrintForm (->[zDialogy];"PRINT_PF_ColumnHeaderShort";20)

   ` Vysuňte poslední stránku
```

```
            PRINT_PF_Footer

               ` Vymazat všechny sady
            CLEAR SET("TotalInvoices")
            CLEAR SET("TotalLineItems")
            For ($i;1;$LNumberOfCategories)
               CLEAR SET("CategoryLineItems"+String($i))
            End for
❖
❖           GEN_ProgressThermometer ("";0;False)          ` Vypnutí teploměru
❖           MESSAGES ON

         End if

      Else
         ALERT("Ve zvoleném období nebyly nalezeny žádné faktury!")
      End if
   End if
      `Konec metody
```

3. Do metody projektu MyStartup přidejte následující řádku kódu.

```
GET PICTURE RESOURCE(15001;<>gProgressThermometer)
```

4. Vytvořte formulář v tabulce [ObecnéInformace] a nazvěte jej `GEN_ProgressThermometer`.

5. Zkopírujte formulář GEN_ProgressThermometer z tabulky [zDialogy] do nově vytvořeného formuláře v tabulce [ObecnéInformace].

Tento krok je nezbytný, protože abychom mohli vidět teploměr musíme mít nějaký platný záznam pro zobrazení formuláře.

6. Otestujte tyto úpravy.

26.8.7. Použití lokálních sad

Sady (sety) žijí na serveru a udržování sad po příliš dlouhou dobu na serveru může být nevýhodné a zatěžující pro paměť serveru. K přesunutí sady na počítač klienta, kde s nimi můžete manipulovat stačí definovat tuto sadu jako lokální (název má předponu $). Vytvoření lokální sady přemístí sadu na počítač klienta.

26.8.8. COPY SET(původnísada;kopiesady)

Příkaz COPY SET kopíruje obsah sady původnísada do sady kopiesady. Obě sady nezávisle na sobě mohou být sady procesu, meziprocessní nebo lokální.

4D Server: v prostředí 4D klient/server jsou procesní a meziprocessní sady udržovány na počítači serveru, zatímco lokální sady jsou udržovány na počítači klienta. Příkaz COPY SET vám dovolí kopírovat sady mezi dvěma stroji.

Následující příklad v prostředí klient/server kopíruje sadu procesu Client "SetA", udržovanou na počítači serveru do lokální sady "$SetB" udržované na počítači klienta:

```
COPY SET("SetA";"$SetB")
```

26.8.9. Přemisťování sad mezi serverem a klientem.

1. Nahraďte metodu projektu ZpravaMixProduktu následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
    ` Metoda: ZpravaMixProduktu
    ` Kurz programovani ACI University
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Creates a Product Mix Report

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

    ` Deklarace lokálních proměnných
C_LONGINT($i)                       `Čítač smyčky
C_LONGINT($j)                       `Čítač smyčky
C_LONGINT($LRecordsInSelection)         `Zaznamů ve výběru
C_LONGINT($LNumberOfProducts)       ` Celkový počet produktů
C_LONGINT($LNumberOfCategories) ` Celkový počet kategorií
C_LONGINT($LQuantityOfProductSold)  ` Celkové množství prodaných produktů
C_LONGINT($LQtySoldPerStateCategory) ` Celkové množství prodané na kategorii na
stát
C_LONGINT($LNumberOfStates)     ` Celkový počet států
C_LONGINT($LDifferentTitles)    ` Celkový počet různých titulů prodaných na stát
C_REAL($rTotalSold)                 ` Celkový prodejní obrat
C_REAL($rTotalStateSold)            ` Celkový prodejní obrat na stát
C_REAL($rStateSales)                ` Kategorie prodané na stát
C_REAL($rFractionPerLoop)           ` Zlomek dokončení na smyčku pro teploměr
C_REAL($rPercentComplete)           ` Skutečně dokončeno procent
```

```
ARRAY STRING(15;$asCategories;0)
ARRAY STRING(2;$asStates;0)

QueryInvoices

If ((OK=1) & (dFrom # !00/00/00!))
   $LRecordsInSelection:=Records in selection([Faktury])
   If ($LRecordsInSelection>0)

      ALL RECORDS([ObecnéInformace])   ` Zde získáváme název naší společnosti
      PRINT SETTINGS                   ` Uživatel volí vzhled stránky
      If (OK=1)
         `zvolen vzhled stránky, nyní úvodní stránku
        LPixelCount:=0                     ` Nastavení čítače bodů
        LPageNumber:=0                     ` Nastavení čísla stránky
        fTallHeader:=False
        sReportName:="Zpráva mix produktů"     ` Nastavení názvu zprávy

        MESSAGES OFF
        GEN_ProgressThermometer ("Probíhá výpočet počtu států.";0;True)

        CREATE SET([Faktury];"TotalInvoices")` Vytvoření sady ze všech
reportovaných faktur

          ` Získání zákazníků z těchto faktur
        If ($LRecordsInSelection<(Records in table([Faktury])/2))
          RELATE ONE SELECTION([Faktury];[Zákazníci])
        Else
          CREATE EMPTY SET([Zákazníci];"TotalCustomers")
          For ($i;1;$LRecordsInSelection)
            RELATE ONE([Faktury]IDZákazníka)
            ADD TO SET([Zákazníci];"TotalCustomers")")
            NEXT RECORD([Faktury])
          End for
          USE SET("TotalCustomers")
          CLEAR SET("TotalCustomers")
        End if

        DISTINCT VALUES([Zákazníci]Stát;$asStates)

          ` Získání položek faktur pro tyto faktury
        If ($LRecordsInSelection<(Records in table([Faktury])/2))
          RELATE MANY SELECTION([PoložkyFaktury]ČísloFaktury)
          CREATE SET([PoložkyFaktury];"TotalLineItems")
        Else
          FIRST RECORD([Faktury])
          CREATE EMPTY SET([PoložkyFaktury];"TotalLineItems")
          For ($i;1;$LRecordsInSelection)
            RELATE MANY([Faktury]ČísloFaktury)
            CREATE SET([PoložkyFaktury];"TempSet")
            UNION ("TotalLineItems";"TempSet";"TotalLineItems")
```

```
            CLEAR SET("TempSet")
            NEXT RECORD([Faktury])
          End for
        End if

        USE SET("TotalLineItems")
❖       COPY SET("TotalLineItems";"$TotalLineItems")
❖       CLEAR SET("TotalLineItems")


        $LQuantityOfProductSold:=Sum([PoložkyFaktury]Množství)          ` Získá
celkové prodané množství
        $rTotalSold:=Sum([PoložkyFaktury]CenaCelkem)

        ALL RECORDS([Produkty])
        DISTINCT VALUES([Produkty]Kategorie;$asCategories)
        GEN_ProgressThermometer ("Probíhá výpočet kategorií.";0)

          ` Vytvoření array kategorie
        $LNumberOfCategories:=Size of array($asCategories)
            ` Kolik v této kategorii
        ARRAY INTEGER($aiQuantityOfCategory;$LNumberOfCategories)
            ` Procenta z celků
        ARRAY REAL($arCategoryPercentOfTotalProduct;$LNumberOfCategories)
            ` Kolik prodáno v této kategorii
        ARRAY INTEGER($aiQuantityOfCategorySold;$LNumberOfCategories)
            ` Procenta z celkových prodejů
        ARRAY REAL($arCategoryPercentOfTotalSold;$LNumberOfCategories)

        $LNumberOfProducts:=Records in selection([Produkty])

            ` Příprava dat pro teploměr průběhu
        $LNumberOfStates:=Size of array($asStates)
        $rFractionPerLoop:=100/($LNumberOfCategories*($LNumberOfStates+2))
        $rPercentComplete:=0

        For ($i;1;$LNumberOfCategories)
          QUERY([Produkty];[Produkty]Kategorie = $asCategories{$i})          ` Nalézt
každou kategorii

            ` Celkové množství pro kategorii
          $aiQuantityOfCategory{$i}:=Records in selection([Produkty])
            ` Výpočet procent pro všechny produkty
          $arCategoryPercentOfTotalProduct{$i}:=Round($aiQuantityOfCategory{$i}/
                $LNumberOfProducts*100;2)

            ` Získání čísla prodáno v kategorii
          If ($arCategoryPercentOfTotalProduct{$i}<50)
            RELATE MANY SELECTION([PoložkyFaktury]IDZbožíProdukty)
          Else
              ` Chceme všechny položky faktur pro typ kategorie
```

```
            QUERY([PoložkyFaktury];[Produkty]Kategorie=$asCategories{$i})
         End if
         CREATE SET([PoložkyFaktury];"CategoryLineItems")

            ` Získání jedné kategorie v daném období
❖
      INTERSECTION("CategoryLineItems";"$TotalLineItems";"$CategoryLineItems"+String(
$i))
❖        USE SET("$CategoryLineItems"+String($i))


            ` Nakonec máme prodané množství
         $aiQuantityOfCategorySold{$i}:=Sum([PoložkyFaktury]Množství)
            ` Výpočet procent z celkových prodejů
         $arCategoryPercentOfTotalSold{$i}:=Round($aiQuantityOfCategorySold{$i}/
               $LQuantityOfProductSold*100;2)

            `Obnova teploměru průběhu
         $rPercentComplete:=$rPercentComplete+$rFractionPerLoop
            GEN_ProgressThermometer ("Výpočet součtů pro
kategorii";$rPercentComplete

      End for

         ` Tisk hlavičky první stránky
            GEN_ProgressThermometer ("Tisk součtů pro kategorii";$rPercentComplete)

      PRINT_PrintForm (->[zDialogy];"PRINT_PF_MainHeader";0)

         ` Definice záhlaví sloupců
      sDescriptionHead:="Kategorie filmů"
      sColumnHead1:=""
      sColumnHead2:="Celkové mn."
      sColumnHead3:="% z celkem"
      sColumnHead4:="# Prodáno"
      sColumnHead5:="% Mn. prodáno"

      PRINT_PF_ColumnHeaders (True)    ` Tisk začátku s tlustou čárou

         ` Tisk obsahu pro tuto kategori
      For ($i;1;$LNumberOfCategories)
         sDescription:=$asCategories{$i}
         sColumn1:=""
         sColumn2:=String($aiQuantityOfCategory{$i};"#####")
         sColumn3:=String($arCategoryPercentOfTotalProduct{$i};"###,00")
         sColumn4:=String($aiQuantityOfCategorySold{$i};"#####")
         sColumn5:=String($arCategoryPercentOfTotalSold{$i};"### 00")

         PRINT_PrintForm (->[zDialogy];"PRINT_PF_Detail";18)

            `Obnova průběhu teploměru
```

```
            $rPercentComplete:=$rPercentComplete+$rFractionPerLoop
            GEN_ProgressThermometer ("Tisk kategorie
"+sDescription;$rPercentComplete)

         End for

            ` Tisk celkových součtů kategorie
         sDescription:="Kategorie celkem"
         sColumn1:=""
         sColumn2:=String($LNumberOfProducts;"#####")
         sColumn3:=""
         sColumn4:=String($LQuantityOfProductSold;"#####")
         sColumn5:=""
         PRINT_PrintForm (->[zDialogy];"PRINT_PF_Totals";24)

         For ($i;1;Size of array($asStates))
               ` Tiska hlavičky státu
            sDescriptionHead:="Stát: "+$asStates{$i}
            sColumnHead1:="Počet jedn."
            sColumnHead2:="Počet titulů"
            sColumnHead3:="Částka"
            sColumnHead4:="% ve státě"
            sColumnHead5:="% celkem"

            PRINT_PrintForm (->[zDialogy];"PRINT_PF_Gap";10)

               ` Tisk záhlaví nové sekce
            fTallHeader:=True
         PRINT_PF_ColumnHeaders (False)   ` Tisk s normální čárou

            $LDifferentTitles:=0

            USE SET("TotalInvoices")
            QUERY SELECTION([Faktury];[Zákazníci]Stát=$asStates{$i})

               ` Získání položek faktur pro faktury za stát
            If ($LRecordsInSelection<(Records in table([Faktury])/2))
               RELATE MANY SELECTION([PoložkyFaktury]ČísloFaktury)
               CREATE SET([PoložkyFaktury];"StateLineItems")
            Else
               CREATE EMPTY SET([PoložkyFaktury];"StateLineItems")
               For ($j;1;$LRecordsInSelection)
                  RELATE MANY([Faktury]ČísloFaktury)
                  CREATE SET([PoložkyFaktury];"TempSet")
                  UNION ("StateLineItems";"TempSet";"StateLineItems")
                  CLEAR SET("TempSet")
                  NEXT RECORD([Faktury])
               End for
            End if

            USE SET("StateLineItems")
```

```
                $rTotalStateSold:=Sum([PoložkyFaktury]CenaCelkem)
                $LQtySoldPerStateCategory:=Sum([PoložkyFaktury]Množství)

                For ($j;1;$LNumberOfCategories)
❖                 INTERSECTION("StateLineItems";"$CategoryLineItems"+String($j);
"StateCategoryLineItems")

                  USE SET("StateCategoryLineItems")

                    ` Jsou zde nějaké kategorie k tisku
                  If (Records in selection([PoložkyFaktury])>0)
                    sDescription:=$asCategories{$j}
                    sColumn1:=String(Sum([PoložkyFaktury]Množství);"#####")
                    RELATE ONE SELECTION([PoložkyFaktury];[Produkty])
                    sColumn2:=String(Records in selection([Produkty]))
                    $LDifferentTitles:=$LDifferentTitles+Num(sColumn2)
                    $rStateSales:=Sum([PoložkyFaktury]CenaCelkem)
                    sColumn3:=String($rStateSales;"|Částka")
                    sColumn4:=String(Round($rStateSales/$rTotalStateSold*100;2);"###,00")
                    sColumn5:=String(Round($rStateSales/$rTotalSold*100;2);"###.00")

                    PRINT_PrintForm (->[zDialogy];"PRINT_PF_Detail";18)
                  End if

                  CLEAR SET("StateCategoryLineItems")

                    `Obnovit teploměr průběhu
                $rPercentComplete:=$rPercentComplete+$rFractionPerLoop
                GEN_ProgressThermometer ("Tisk kategorie "+sDescription+" pro stát  "
                        +sDescriptionHead;$rPercentComplete)

                End for

                  ` Tisk celkem stát
                sDescription:="Totals for "+$asStates{$i}
                sColumn1:=String($LQtySoldPerStateCategory;"#####")
                sColumn2:=String($LDifferentTitles;"#####")
                sColumn3:=String($rTotalStateSold;"|Částka")
                sColumn4:=""
                sColumn5:=String(Round($rTotalStateSold/$rTotalSold*100;2);"###.00")
                PRINT_PrintForm (->[zDialogy];"PRINT_PF_Totals";25)

                CLEAR SET("StateLineItems")

              End for

                `Je čas pro součty
              PRINT_PrintForm (->[zDialogy];"PRINT_PF_BigLine";30)
              PRINT_PrintForm (->[zDialogy];"PRINT_PF_Gap";3)
              PRINT_PrintForm (->[zDialogy];"PRINT_PF_BigLine";0)
```

```
❖                 USE SET("$TotalLineItems")

                  $LRecordsInSelection:=Records in selection([PoložkyFaktury])

                        `Obnova teploměru průběhu
                    GEN_ProgressThermometer ("Dokončování zprávy";100)


                  If ($LRecordsInSelection<(Records in table([PoložkyFaktury])/2))
                    RELATE ONE SELECTION([PoložkyFaktury];[Produkty])
                  Else
                    CREATE EMPTY SET([Produkty];"ProductsSold")
                    For ($i;1;$LRecordsInSelection)
                      RELATE ONE([PoložkyFaktury]IDZbožíProdukty)
                      ADD TO SET([Produkty];"ProductsSold")
                      NEXT RECORD([PoložkyFaktury])
                    End for
                    USE SET("ProductsSold")
                    CLEAR SET("ProductsSold")
                  End if

                  sDescriptionHead:="Celkový součet"
                  sColumnHead1:=String($LQuantityOfProductSold;"#####")
                  sColumnHead2:=String(Records in selection([Produkty]);"#####")
                  sColumnHead3:=String($rTotalSold;"|Částka")
                  sColumnHead4:=""
                  sColumnHead5:=""

                    ` Tisk součtů kontrolovaný pro konec předchozí stránky
                  fTallHeader:=False
                  PRINT_PrintForm (->[zDialogy];"PRINT_PF_ColumnHeaderShort";20)

                    ` Vysuňte poslední stránku
                  PRINT_PF_Footer

                    ` Vymazat všechny sady
❖                CLEAR SET("$TotalInvoices")
❖                 CLEAR SET("$TotalLineItems")
                  For ($i;1;$LNumberOfCategories)
❖                   CLEAR SET("$CategoryLineItems"+String($i))
                  End for

                  GEN_ProgressThermometer ("";0;False)          ` Vypnutí teploměru
                  MESSAGES ON

                End if

              Else
                ALERT("Ve zvoleném období nebyly nalezeny žádné faktury!")
              End if
```

```
End if
    `Konec metody
```

2. Otestujte tyto úpravy.

27. Více o ošetřování chyb

Jak jsme probrali již v předchozích kapitolách lze ve 4D vytvořit vlastní chybová hlášení na základě kritérií, která si vymyslíte. Díky triggerům se vaše akce Uložit záznam, Vymazat záznam atd. neprovedou jestliže je do parametru $0 v triggeru přiřazena jiná hodnota než 0.

Tento fakt můžete využít k vytvoření svých vlastních chyb.

27.0.1. Vytvoření vlastního systému chyb.

1. Vytvořte metodu projektu ERROR_CustomErrors následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: ERROR_CustomErrors
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Ošetřování chyb pro chyby záznamu zákazníků

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_STRING(255;$sMessage)

$sMessage:=""

Case of
   ś (Error = -15997)
      $sMessage:="Tel. číslo potřebuje předčíslí UTO."+Char(Carriage return)+"Prosíme
opravte."
      GOTO PAGE(1)
      GOTO AREA([Zákazníci]Telefon)

   ś (Error = -15998)
      $sMessage:="Bart je skvělý trenér, ale "+"může dělat jenom jednu práci
současně!"+Char(Carriage return)+"Zkuste jinou kontaktní osobu!"

   ś (Error = -15999)
      $sMessage:="Kent se snaží."+Char(Carriage return)+"Dejte, ale příležitost někomu
jinému!"

   ś (Error = -16001)
```

```
      $sMessage:="Jeden ze zákazníků, kterého se snažíte vymazat má faktury."+
Char(Carriage return)+"Zákazník nemůže být vymazán!"

   ś (Error = -16002)
      $sMessage:="Jeden z produktů, které se snažíte vymazat má faktury."+
Char(Carriage return)+"Produkt nemůže být vymazán!"

   ś (Error = -9984)
      $sMessage:="Záznam byl již vymazán!"+Char(Carriage return)+ Char(Carriage
return)+"Ukončuji mazání záznamů!"

   ś (Error = -9987)
      $sMessage:="V databázi existují záznamy, které závisí na některých záznamech, "+
" které se snažíte vymazat!"+Char(Carriage return)+Char(Carriage return)+"Ukončuji
mazání záznamů!"

   :(Error = -9991)
      $sMessage:="Nemáte oprávnění provádět tuto operaci!"

   Else
      $sMessage:="Vyskytla se neočekávaná chyba číslo  # "+String(Error)+
Char(Carriage return)+"Poznamenejte si prosím toto číslo a konzultujte svého
administrátora!"
End case

If( Length ($sMessage) >0)
   ALERT($sMessage)
End if
      `Konec metody
```

2. Vytvořte metodu projektu fValidateCustomers následujícím způsobem nebo importujte pomocí textovéh editoru:

```
If (False)
      ` Metoda: fValidateCustomers
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Volána při kontrole přijímaného záznamu zákazníka

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definovat parametry
C_BOOLEAN($0)

      ` Zapne ošetřování chyb k zachycení chyb generovaných triggerem
```

```
ON ERR CALL("ERROR_CustomErrors")
Error:=0

    ` Ukládá záznam, který bude generovat chybu v triggeru
SAVE RECORD([Zákazníci])

    ` Navrátí při libovolné chybě
$0:=(Error=0)

    ` Vypne ošetřování chyb
ON ERR CALL("")
`Konec metody
```

3. Vytvořte metodu objektu pro tlačítko bValidate button ve formuláři [Zákazníci]Zákazníci.v následovně:

```
If (False)
    ` Metoda: bValidate
    ` Kurz programovani ACI University
    ` Autor: Kent Wilbur
    ` Datum: 3/17/97

    ` Účel: Uloží záznam je-li platný a opustí formulář.

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

If (fValidateCustomers )
  ACCEPT
End if
    `Konec metody
```

Poznámka: Ideálně by to mělo být tlačítko Storno avšak při přidávání nových záznamů by nám takto bylo povoleno přidat pouze jeden záznam současně. Můžete samozřejmě nahradit tento kód větvením s příznakem, že jestliže je záznam nový proveď Accept jinak Cancel.

4. Vytvořte nové metody objektu pro tlačítka bFirst; bPrevious; bNext a bLast ve formuláři [Zákazníci]Zákazníci .v dle vzoru pro tlačítko bFirst uvedeného dále:

```
If (False)
    ` Metoda: bFirst
    ` Kurz programovani ACI University
    ` Autor: Kent Wilbur
    ` Datum: 1/17/97
```

```
      ` Účel: Přemístí se na první záznam tabulky [Zákazníci] jestliže
      ` je současný záznam platný

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

If (fValidateCustomers )
   FIRST RECORD(Current form table->)
End if
      `Konec metody
```

5. Vytvořte metodu projektu LValidateCustomers následujícím způsobem nebo importujte pomocí textového souboru:

```
If (False)
      ` Metoda: LValidateCustomers
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 1/17/97
      ` CB: Trigger zákazníci

      ` Účel: Zkontroluje správnost vstupu zákazníka

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

$0:=0
[Zákazníci]Telefon:=Replace string ([Zákazníci]Telefon ;"?";"")
If (Length([Zákazníci]Telefon) > 0)
   If (Length([Zákazníci]Telefon) # 10)
      $0:=-15997
   End if
End if

If (([Zákazníci]Příjmení="Wilbur") & ([Zákazníci]Jméno="Kent"))
   $0:=-15999
End if
If (([Zákazníci]Příjmení="Scott") & ([Zákazníci]Jméno="Bart"))
   $0:=-15998
End if
  `Konec metody
```

6. Upravte trigger tabulky [Zákazníci] trigger následovně:

```
…
Case of
   ś ($LDatabaseEvent = On Saving Existing Record Event)
❖        $0:= LValidateCustomer
❖        If ($0 = 0)
            GEN_TimeStamp (->[Zákazníci]DatumUpravení;->[Zákazníci]ČasUpravení)
            $LSelectedElement:=Find in
array(<>CHO_asCustomerID;[Zákazníci]IDZákazníka)
            If ($LSelectedElement > 0)
               If (<>CHO_asCompany{$LSelectedElement} # [Zákazníci]Firma)
                  <>CHO_asCompany{$LSelectedElement}:=[Zákazníci]Firma)
                  $fCompanyModified:=True
               End if
```

```
                    If (<>CHO_asContactName{$LSelectedElement} # [Zákazníci]Jméno + " " +
[Zákazníci]Příjmení)
                      <>CHO_asContactName{$LSelectedElement}:=[Zákazníci]Jméno +
                                  " " + [Zákazníci]Příjmení)
                      $fNameModified:= True
                    End if
                    If (<>CHO_asPhone{$LSelectedElement} #  String(Num([Zákazníci]Telefon);
                                                "|Phone Display Format")
                      <>CHO_asPhone{$LSelectedElement}:=String(Num([Zákazníci]Telefon);
                                                "|Phone Display Format")
                      $fPhoneModified:= True
                    End if
                  End if
❖                 End if

                ś ($LDatabaseEvent = On Saving New Record Event)
❖                 $0:=LValidateCustomers
❖                 If ($0 = 0)
                    $LSelectedElement:=Size of array(<>CHO_asCustomerID)+1
                    ARRAY STRING(11;<>CHO_asCustomerID;$LSelectedElement)
                    ARRAY STRING(31;<>CHO_asCompany;$LSelectedElement)
                    ARRAY STRING(33;<>CHO_asContactName;$LSelectedElement)
                    ARRAY STRING(25;<>CHO_asPhone;$LSelectedElement)
                    <>CHO_asCustomerID{$LSelectedElement}:=[Zákazníci]IDZákazníka
                    <>CHO_asCompany{$LSelectedElement}:=[Zákazníci]Firma
                    <>CHO_asContactName{$LSelectedElement}:=[Zákazníci]Jméno +
                                  " " + [Zákazníci]Příjmení
                    <>CHO_asPhone{$LSelectedElement}:=String(Num([Zákazníci]Telefon);
                                  "|Phone Display Format")
                    $fCompanyModified:=True
❖                 End if
              …
```

7.Otestujte tato nová chybová hlášení.

Poznámka: Chybová hlášení budou generována zadáte-li do jmen kontaktních osob Kent Wilbur a Scot Bart či zadáte-li špatně telefonní číslo

27.1. Pojmenované výběry a chyby

Když vytváříte pojmenované výběry pomocí příkazu COPY NAMED SELECTION nebo použijete pojmenovaný výběr, který byl vytvořen pomocí COPY NAMED SELECTION je možné zkrachovat na nedostatku paměti. Jestliže se tak stane obdržíte kód chyby –181. Stojí za to do svých vlastních chybových hlášení zahrnout i ošetření této chyby. Pak by jste měli instalovat metodu na ošetřování chyb před použitím příkazu COPY NAMED SELECTION.

28. Událost Při zavření okna

Když uživatel klepne na uzavírací okénko očekává, že se okno uzavře. Proto stojí za to, touto akcí generovanou událost Při zavření okna (On Close box) ošetřít.

28.0.1. Vytvoření okna, které se uzavře při klepnutí na uzavírací okénko

1. Vytvořte metodu projektu GEN_CloseBox následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
        ` Metoda: GEN_CloseBox
        ` Kurz programovani ACI University
        ` Genericke metody z nastroju ACI
        ` Autor: Kent Wilbur
        ` Datum: 3/17/97

        ` Účel: Poskytuje metodu pro uzavírací okénko
        ` Tato metoda neprovádí nic kromě nastavení příznaku
        ` kód který skutečně odpovídá na danou událost je umístěn v každé metodě
formuláře
        ` takže jsou možné různé druhy odezvy


   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

fCloseBox:=True
     `Konec metody
```

2. Upravte metodu INITIALIZE_GEN_ModifyVariables následovně:

```
ALL RECORDS([ObecnéInformace])
fProcessAvailable:=False
fCloseBox:=False                    ` Nastaví příznak uzavření okna na Ne
```

❖

3. Upravte všechny existující metody vstupních formulářů tak, aby obsahovaly následující kód:

```
ś ($LFormEvent = On Close Box)
   HIDE PROCESS(Current Process)
   POST KEY(Escape;0)
```

4. Upravte všechny existující metody výstupních formulářů tak, aby obsahovaly následující kód jako první položku bloku Case:

```
ś (fCloseBox) | ($LFormEvent = On Close Box)
   POST KEY(Escape;0)
```

5. Upravte metodu projektu GEN_ModifySelection následovně:

```
…
❖ $LWindowID:=WIN_LNewWindow (-1;-1;<>Cascading window;Plain no zoom box window;"";“GEN_CloseBox")
```

6. Otestujte vaši novou technologii uzavírání oken.

29. Vytváření předvoleb uživatele

Uživatelé systému rádi ovlivňují chování svého systému. Takovýto rys, pokud si to přejete, lze do databáze zavést. Kromě toho může být vhodné omezit určitým uživatelům přístup k některým položkám nabídky nebo jim omezit počet otevřených oken, která mohou být otevřena v určité agendě. Náš další příklad vytvoří sekci předvoleb uživatelů pro naši databázi.

29.0.1. Vytvoření pro předvolby uživatelů

1. Vytvořte novou tabulku a nazvěte ji zUživatelé.
2. Přidejte do této tabulky následující pole:

Název pole	Typ	Vlastnosti
IDUživatele	Longint	Nutný vstup, Indexované, Jedinečné
JménoUživatele	Alfa 31	Nutný vstup
PlatnéJméno	Alfa 42	
Adresa	Text	
Město	Alfa 22	
Stát	Alfa 2	
PSČ	Alfa 9	
Telefon	Alfa 10	
DatumPosledníhoPřipojení	Datum	
ČasPosledníhoPřipojení	Čas	
Trvání	Čas	
NejdelšíDobaPřipojení	Čas	
ČítačTikůPřipojení	Longint	Neviditelné
Připojení	Integer	
Připojen	Logické	
OdmítnoutPřístup	Logické	
PředvolbyDotazuTabulky	Text	Neviditelné
PředvolbyTříděníTabulky	Text	Neviditelné
VíceZobrazení	Text	Neviditelné

3. Vytvořte nový výstupní formulář a pojmenujte jej [zUživatelé]zUživatelé_s. s následujícími poli:

NázevUživatele

Připojení

DatumPosledníhoPřipojení
Trvání
OdmítnoutPřístup

4. Nechte 4D vytvořit výchozí vstupní formulář a přejmenujte jej na [zUživatelé]zUživatelé_v.

29.1. Nastavení výchozích hodnot uživatelů

Pro uživatele bude velmi důležité, aby byly schopni používat systém okamžitě. Proto je velice důležité vytvořit pro normální používání systému výchozí hodnoty pro předvolby. V další fázi práce necháte uživateli možnost tyto hodnoty upravovat. Často jsme viděli systémy, které uživatelle předtím než se připojí do systému poprvé ho požádají, aby definoval výchozí hodnoty. Jestliže je vybral nesprávně systém nebyl schopen nebo ochoten pracovat vůbec. Uživatel, který se k systému připojuje poprvé nezná určitě nic o tomto systému. Udělejte to tedy co nejjednodušší a ne nemožné.

29.1.1. Vytvoření nového uživatele v systému.

Naše databáze vytvoří nového uživatele, když tento vstoupí do systému. Vytvoření požadovaných výchozích hodnot proběhne ve stejnou dobu.

1. Vytvořte metodu projektu USER_GetUser následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: USER_GetUser
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Získá záznam uživatele

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokáních proměnných
C_LONGINT($LTableState)

QUERY([zUživatelé];[zUživatelé]JménoUživatele=Current user)
```

```
$LTableState:=LOCK_LSet2ReadWrite (->[zUživatelé];False)
If (Records in selection([zUživatelé])<1)            ` Vytvoření chybějícího uživatele
   CREATE RECORD([zUživatelé])
   [zUživatelé]JménoUživatele:=Current user
   [zUživatelé]PlatnéJméno:=[zUživatelé]JménoUživatele
   [zUživatelé]OdmítnoutPřístup:=False
   [zUživatelé]PředvolbyDotazuTabulky:=" " * 510          ` Potřebné max pro 255
tabulek
   [zUživatelé]PředvolbyTříděníTabulky:= " " * 510
   [zUživatelé]TypPřístupu:=" " * 255
   [zUživatelé]IDUživatele:=LNextSequence ("UserID")
   SAVE RECORD([zUživatelé])
End if

      ` Přístup uživatele nebyl odmítnut
If (Not([zUživatelé]OdmítnoutPřístup))
   If ([zUživatelé]Připojen=False)          ` Uživatel se právě připojuje
      [zUživatelé]Připojen:=True            ` Uživatel je připojen
      GEN_TimeStamp (->[zUživatelé]DatumPosledníhoPřipojení;-
>[zUživatelé]ČasPosledníhoPřipojení)
      [zUživatelé]Připojení:=[zUživatelé]Připojení+1
      [zUživatelé]ČítačTikůPřipojení:=Tickcount
      SAVE RECORD([zUživatelé])
   End if
Else
   ALERT("Vaše přístupová práva jsou neplatná!"+Char(Carriage return)+"Kontaktujte
svého administrátora.")
   QUIT 4D
End if
LOCK_Set2ReadOnly (->[zUživatelé];$LTableState)
      `Konec metody
```

2. Vytvořte metodu projektu GEN_hTicks2Time následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: GEN_hTicks2Time (Longint) -> Time
      ` Kurz programovani ACI University
      ` Genericke metody z nastroju ACI
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Konvertuje tiky na čas.

   <>fGeneric:=True
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

  ` Definovat parametry
```

```
C_LONGINT($1;$LTickcount)      ` Tiky ke konverzi do času
C_TIME($0)

      ` Deklarace lokálních proměnných
C_LONGINT($LHours)
C_LONGINT($LMinutes)
C_LONGINT($LSeconds)

  ` Přiřazení parametrů k lokálním proměnným
$LTickcount:=$1
$LSeconds:=$LTickcount/60
If ($LSeconds > 60)
   $LMinutes:=$LSeconds/60
   $LSeconds:=$LSeconds%60
   If ($LMinutes>60)
      $LHours:=$LMinutes/60
      $LMinutes:=$LMinutes%60
   End if
End if

$0:=Time("!"+String($LHours)+":"+String($LMinutes)+":"+String($LSeconds))
  `Konec metody
```

3. Vytvořte metodu projektu USER_LogoutUser následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: USER_LogoutUser
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Uzavře záznam uživatele

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($LLapsedTicks)
C_LONGINT($LTableState)

QUERY([zUživatelé];[zUživatelé]JménoUživatele=Current user)
$LTableState:=LOCK_LSet2ReadWrite (->[zUživatelé];False)

$LLapsedTicks:=Tickcount-[zUživatelé]ČítačTikůPřipojení
[zUživatelé]Trvání:=GEN_hTicks2Time ($LLapsedTicks)
If ([zUživatelé]Trvání > [zUživatelé]NejdelšíDobaPřipojení )
   [zUživatelé]NejdelšíDobaPřipojení:=[zUživatelé]Trvání
```

```
End if
[zUživatelé]Připojen:=False                ` Odpojení uživatele
SAVE RECORD([zUživatelé])
LOCK_Set2ReadOnly (->[zUživatelé];$LTableState)
    `Konec metody
```

4. Přidejte do metody projektu MyStartup následující řádek.

```
…
<>LMainProcess:=Current process

❖           USER_GetUser
…
```

5. Nahraďte metodu projektu M_GEN_Quit následovně:

```
If (False)
    ` Metoda: M_GEN_Quit
    ` Kurz programovani ACI University
    ` Created by Jim Steinman
    ` Datum: 1/15/97

    ` Účel: Tato metoda ukončí databázi

  <>fGeneric:=True
  <>f_Version6x10:=True
  <>f_Version6x20:=True
  <>f_Version6x30:=True
  <>fJ_Steinman:=True

    ` Upraveno: 1/17/97
  <>fK_Wilbur:=True
    ` Upraveno: 3/17/97
  <>fK_Wilbur:=True
End if

<>fQuit:=True ` Nastaví příznak ukončování takže od tohoto okamžiku nejde nic začít

If (<>LBackground>0)

  If (<>LBackground=1)
    CONFIRM("Je spuštěn proces na pozadí. Ukončit po jeho skončení?")
  Else
    CONFIRM("Jsou spuštěny procesy na pozadí. Ukončit po jejich skončení?")
  End if

  If (OK=1)

    Repeat
      PROCESS_MyDelay(Current process;240)
    Until (<>LBackground=0)
```

```
      USER_LogoutUser
      QUIT 4D
    Else
      <>fQuit:=False ` Návrat do systému
    End if

  Else
    USER_LogoutUser
    QUIT 4D
  End if
      ` Konec metody
```

6. Upravte metodu INITIALIZE_GEN_ModifyVariables následovně:

```
ALL RECORDS([ObecnéInformace])
fProcessAvailable:=False
fCloseBox:=False          ` Nastaví příznak uzavření okna na Ne
USER_GetUser
```

7. Zrestartujte databázi k vytvoření uživatelského záznamu pro Návrháře.

29.2. Vytvoření uživatelsky přívětivého rozhraní k úpravám předvoleb uživatele

V současné době máme metodu pro změny v tabulce [ObecnéInformace] a můžeme ji využít pro tentýž účel k zpřístupnění individuálních předvoleb jednotlivým uživatelům.

29.2.1. Úpravy předvoleb uživatelů

1. Vytvořte metodu projektu USER_ModifyUser následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: USER_ModifyUser
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Umožní uživateli upravovat své předvolby

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($LTableState)

QUERY([zUživatelé];[zUživatelé]JménoUživatele=Current user)
$LTableState:=LOCK_LSet2ReadWrite (->[zUživatelé];True)

If ($LTableState>=0)
   MODIFY RECORD([zUživatelé];*)
   LOCK_Set2ReadOnly (->[zUživatelé];$LTableState)
End if
      `Konec metody
```

2. Upravte metodu projektu M_Preferences následovně:

```
…
C_LONGINT($LTableState)
❖  C_LONGINT($LUserTableState)

If ((Current user="Návrhář") | (Current user="Administrátor"))
   $LTableState:=LOCK_LSet2ReadWrite (->[ObecnéInformace];True)
❖  $LUserTableState:=LOCK_LSet2ReadWrite (->[zUživatelé];True)

   ALL RECORDS([ObecnéInformace])

   START TRANSACTION
   If (Records in selection([ObecnéInformace])=1)
```

```
    MODIFY RECORD([ObecnéInformace];*)
  Else
    ADD RECORD([ObecnéInformace])
  End if

  If (OK=1)
    VALIDATE TRANSACTION
  Else
    CANCEL TRANSACTION
  End if

  LOCK_Set2ReadOnly (->[ObecnéInformace];$LTableState)
❖ LOCK_Set2ReadOnly (->[zUživatelé];$LUserTableState )
❖ QUERY([zUživatelé];[zUživatelé]JménoUživatele=Current user)

Else
  ALERT("You are not authorized to change the Preferences!")
❖ USER_ModifyUser
…
```

3. Přidejte tlačítko nazvané `bModifyUser` na první stránku formuláře [ObecnéInformace]Vstupní.
4. Přidělte tlačítku text Upravit osobní předvolby.
5. Vytvořte metodu objektu pro tlačítko bModifyUser následovně:

```
If (False)
      ` Metoda: bModifyUser
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Dovolí uživateli upravit své předvolby

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

USER_ModifyUser
    `Konec metody
```

6. Otestujte tyto změny v prostředí Vlastní nabídky.

29.2.2. Vytvoření metody k úpravám předvoleb uživatelů pro dotazy a třídění

1. Vytvořte metodu projektu PRELIM_QueryOrderPreferences následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
	` Metoda: PRELIM_QueryOrderPreferences(pointer)
	` Kurz programovani ACI University
	` Autor: Kent Wilbur
	` Datum: 3/17/97

	` Účel: Vytvoří uživatelská array pro dotazy a třídění na základě předvoleb
uživatele

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

	` Definovat parametry
C_POINTER($1)	` Ukazatel na tabulku

	` Deklarace lokálních proměnných
C_LONGINT($LPreferencePosition)
C_LONGINT($LTableNumber)		` Číslo tabulky

$LTableNumber:=Table($1)		` Získá číslo tabulky
$LPreferencePosition:=($LTableNumber*2)-1

	` Získá pole dotazu pro nabídku
QUERY([zStruktura];[zStruktura]ČísloTabulky = $LTableNumber;*)
QUERY([zStruktura]; & ;[zStruktura]VložitVlastníHledání = True)
ORDER BY([zStruktura];[zStruktura]PořadíZástupce)
SELECTION TO ARRAY([zStruktura]NázevZástupce;PRELIM_asQuery)
	` Naplní nabídku hodnotami
PRELIM_asQuery:=Num(Substring([zUživatelé]PředvolbyDotazuTabulky;$LPreference
Position;2))+1

	`Získá pole třídění pro nabídku
QUERY([zStruktura];[zStruktura]ČísloTabulky=$LTableNumber;*)
QUERY([zStruktura]; & ;[zStruktura]VložitVlastníTřídění=True)
ORDER BY([zStruktura];[zStruktura]PořadíZástupce)
SELECTION TO ARRAY([zStruktura]NázevZástupce;PRELIM_asOrder)
	` Vloží možnost netřídit jako výchozí
INSERT ELEMENT(PRELIM_asOrder;1;1)
PRELIM_asOrder{1}:="** Žádné **"
	` Naplní nabídku hodnotou
PRELIM_asOrder:=Num(Substring([zUživatelé]PředvolbyTříděníTabulky;$LPreference
Position;2))+1
	`Konec metody
```

2. Upravte metodu formuláře pro [zDialogy]PRELIM_QueryOrder.NWA následovně:

```
If (False)
```

```
            ` Form Method: [zDialogy];"PRELIM_QueryOrder.NWA"
            ` Kurz programovani ACI University
            ` Genericke metody z nastroju ACI
            ` Autor: Kent Wilbur
            ` Datum: 3/17/97

            ` Účel: Vytvoří vlastní array pro dotazy a třídění

      <>fGeneric:=True
      <>f_Version6x30:=True
      <>fK_Wilbur:=True

   End if

      ` Deklarace lokálních proměnných
   C_LONGINT($LFormEvent)
   C_LONGINT($LCurrentTable)

   $LFormEvent:=Form event

   If ($LFormEvent=On Load)
      $LCurrentTable:=Table(pTable)

❖     PRELIM_QueryOrderPreferences (pTable)
      PRELIM_QueryValue:=""
      PRELIM_QueryValueTo:=""
      PRELIM_QueryMethod

            ` Přiřadí výchozí hodnoty ke všem tlačítkům, které kontrolujeme v metodách
            ` Je nezbytné pro úspěšné zobrazení dialogu na Web
      PRELIM_bgQuery:=0

   End if
      `Konec metody
```

3. Otevřete formulář [zUživatelé]zUživate .v.
4. Přidejte do tohoto formuláře `[zU•ivatelé]zUsers.v` nabídku/seznam a pojmenujte ji `ALIAS_asTableNameAlias`.
5. Vytvořte metodu objektu ALIAS_asTableNameAlias pro nabídku/seznam následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: ALIAS_asTableNameAlias [zUživatelé];"zUživatelé .v"
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Přemístí uživatele z tabukly do tabulky
```

```
   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if


If (Form event=On Clicked)
   PRELIM_QueryOrderPreferences
(Table(ALIAS_aiTableNumber{ALIAS_asTableNameAlias}))
End if
` Konec metody
```

6. Vytvořte metodu projeku USER_UpdateQueryOrderPreference následujícím způsobem nebo importujte pomocí textového editoru:

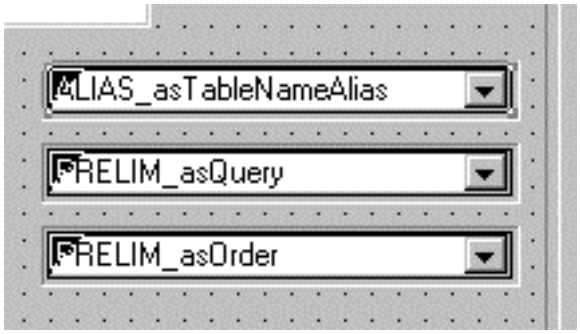
```
If (False)
      ` Metoda: USER_UpdateQueryOrderPreference
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Obnoví předvolby uživatele pro dotazy a třídění

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

   ` Definovat parametry
C_POINTER($1;$pField)            ` Ukazatel na pole k obnově
C_LONGINT($2;$LArrayPosition)    ` Pozice v array, která je výchozí

   ` Deklarace lokálních proměnných
C_LONGINT($LPreferencePosition)
C_LONGINT($LTableNumber)         ` Číslo tabulky
C_TEXT($tBeginning)
C_TEXT($tEnd)

   ` Přiřazení parametrů k lokálním proměnným
$pField:=$1
$LArrayPosition:=$2

$LTableNumber:=ALIAS_aiTableNumber{ALIAS_asTableNameAlias}
   ` Nalézt umístění tabulky
$LPreferencePosition:=($LTableNumber*2)-1
   ` Nahradit předvolby
$tBeginning:=Substring($pField->;1;$LPreferencePosition-1)
$tEnd:=Substring($pField->;$LPreferencePosition+2)
$pField->:=$tBeginning+String(($LArrayPosition-1);"00")+$tEnd
   `Konec metody
```

7. Přidejte objekt nabídka/seznam nazvaný `PRELIM_asQuery`.
8. Vytvořte metodu objektu pro nabídku/seznam nazvanou PRELIM_asQuery následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
        ` Metoda: PRELIM_asQuery [zUživatelé];"Users.i"
        ` Kurz programovani ACI University
        ` Autor: Kent Wilbur
        ` Datum: 3/17/97

        ` Účel: Přiřadí předvolby uživatele pro dotazy

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

If (Form event=On Clicked)
   If (PRELIM_asQuery>0)
      USER_UpdateQueryOrderPreference (-
>[zUživatelé]PředvolbyDotazuTabulky;PRELIM_asQuery)
   End if
End if
      `Konec metody
```

9. Přidejte objekt nabídka/seznam nazvaný `PRELIM_asOrder`.
10. Vytvořte metodu objektu nabídky/seznam PRELIM_asOrder následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
        ` Metoda: PRELIM_asOrder [zUživatelé];"zUsers.i"
        ` Kurz programovani ACI University
        ` Autor: Kent Wilbur
        ` Datum: 3/17/97

        ` Účel: Přiřadí předvolby uživatele pro třídění

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

If (Form event=On Clicked)
   If (PRELIM_asOrder>0)
      USER_UpdateQueryOrderPreference (-
>[zUživatelé]PředvolbyTříděníTabulky;PRELIM_asOrder)
   End if
End if
      `Konec metody
```



11. Vytvořte metodu formuláře pro formulář [zUživatelé]zUživatelé.v následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
        ` Form Method: zUsers.i
        ` Kurz programovani ACI University
        ` Autor: Kent Wilbur
        ` Datum: 3/17/97

        ` Účel: Připraví předvolby uživatele

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

     ` Deklarace lokálních proměnných
C_LONGINT($LFormEvent)

$LFormEvent:=Form event

Case of
  ś ($LFormEvent=On Load)

     If (JménoUživatele=Current user)
        SET ENTERABLE([zUživatelé]OdmítnoutPřístup;False)
     End if

        ` Připraví array tabulek
     ARRAY INTEGER(ALIAS_aiTableNumber;0)
     ARRAY STRING(31;ALIAS_asTableNameAlias;0)
     ARRAY INTEGER(ALIAS_aiTableAliasOrder;0)

     QUERY([zStruktura];[zStruktura]ČísloTabulky=0)          ` Vyhledá existují data
názvů tabulek

     If (Records in selection([zStruktura])>0)
```

```
        SELECTION TO ARRAY([zStruktura]ČísloPole;ALIAS_aiTableNumber;
[zStruktura]NázevZástupce;ALIAS_asTableNameAlias;[zStruktura]PořadíZástupce;
ALIAS_aiTableAliasOrder)
        SORT
ARRAY(ALIAS_aiTableAliasOrder;ALIAS_asTableNameAlias;ALIAS_aiTableNumbe
r)
        ALIAS_asTableNameAlias:=1
        PRELIM_QueryOrderPreferences
(Table(ALIAS_aiTableNumber{ALIAS_asTableNameAlias}))
     End if

End case
     `Konec metody
```

12. Otestujte tyto nové změny v prostředí Vlastní nabídky.

29.2.3. Vytvoření metody pro změnu přístupu uživatele

1. Vytvořte metodu projektu USER_FillUserAccessArrays následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
        ` Metoda: USER_FillUserAccessArrays
        ` Kurz programovani ACI University
        ` Autor: Kent Wilbur
        ` Datum: 3/17/97

        ` Účel: Naplní array přístupu uživatele

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Deklarace lokálních proměnných
C_LONGINT($i)    `Čítač smyčky
C_LONGINT($LStatusValue)          ` Hodnota stavu tabulky
C_STRING(31;$sOldUserName)        ` Staré jméno uživatele

$sOldUserName:=[zUživatelé]JménoUživatele

QUERY([zUživatelé];[zUživatelé]JménoUživatele=USER_asUserNames{USER_asUser
Names})
If (Not(Locked([zUživatelé])))
   For ($i;1;Size of array(USER_asUserTableStatus))

   $LStatusValue:=Num(Substring([zUživatelé]TypPřístupu;USER_aLTableNumber{$i}
;1))

      Case of
```

```
        ś ($LStatusValue=0)
          USER_asUserTableStatus{$i}:=" "
        ś ($LStatusValue=1)
          USER_asUserTableStatus{$i}:="1"
        ś ($LStatusValue=2)
          USER_asUserTableStatus{$i}:="S"
        ś ($LStatusValue=3)
          USER_asUserTableStatus{$i}:="X"
      End case
    End for
Else
      ` Záznam je uzamčen nastaví zpět staré hodnoty
  ALERT("Tento záznam uživatele je uzamčen. Nyní jej nelze upravit!")
  QUERY([zUživatelé];[zUživatelé]JménoUživatele = $sOldUserName)
  USER_asUserNames:=Find in array(USER_asUserNames;$sOldUserName)
End if
    `Konec metody
```

2. Vytvořte metodu projektu USER_SaveUserAccessArrays následovně nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: USER_SaveUserAccessArrays
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Uloží array uživatelského přístupu

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

    ` Deklarace lokálních proměnných
C_LONGINT($i)    `Čítač smyčky
C_LONGINT($LStatusValue)          ` Hodnoty stavu tabulky
C_STRING(255;$sBeginning)
C_STRING(255;$sEnd)

[zUživatelé]TypPřístupu:=[zUživatelé]TypPřístupu+(255-
Length([zUživatelé]TypPřístupu)*" ")

For ($i;1;Size of array(USER_asUserTableStatus))
  Case of
    ś (USER_asUserTableStatus{$i}=" ")
      $LStatusValue:=0
    ś (USER_asUserTableStatus{$i}="1")
      $LStatusValue:=1
    ś (USER_asUserTableStatus{$i}="S")
```

```
      $LStatusValue :=2
    ś (USER_asUserTableStatus{$i}="X")
      $LStatusValue:=3
  End case

      ` Potřebujeme obnovit změny
  If
($LStatusValue # Num(Substring([zUživatelé]TypPřístupu;USER_aLTableNumber{$i};
1)))

    If (USER_aLTableNumber{$i}>1)
      $sBeginning:=Substring([zUživatelé]TypPřístupu;1;USER_aLTableNumber{$i}-
1)
    Else
      $sBeginning:=""
    End if

    If (USER_aLTableNumber{$i}<(Length([zUživatelé]TypPřístupu)-1))
      $sEnd:=Substring([zUživatelé]TypPřístupu;USER_aLTableNumber{$i}+1)
    Else
      $sEnd:=""
    End if

    [zUživatelé]TypPřístupu:=$sBeginning+String($LStatusValue)+$sEnd

  End if

End for

SAVE RECORD([zUživatelé])
    `Konec metody
```

3. Vytvořte metodu projektu USER_ChangeTableAccess následovně nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: USER_ChangeTableAccess
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Dovolí uživateli upravovat své předvolby

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

Case of
  ś (USER_asUserTableStatus{USER_asUserTableStatus}=" ")
```

```
    USER_asUserTableStatus{USER_asUserTableStatus}:="1"
  ś (USER_asUserTableStatus{USER_asUserTableStatus}="1")
    USER_asUserTableStatus{USER_asUserTableStatus}:="S"
  ś (USER_asUserTableStatus{USER_asUserTableStatus}="S")
    USER_asUserTableStatus{USER_asUserTableStatus}:="X"
  ś (USER_asUserTableStatus{USER_asUserTableStatus}="X")
    USER_asUserTableStatus{USER_asUserTableStatus}:=" "
End case
    `Konec metody
```

4. Otevřete formulář [ObecnéInformace] ObecnéInformace.v.
5. Vytvořte stránku dvě v tomto formuláři.
6. Přemístěte všechny obecné položky formuláře na stránku 0, je-li to nezbytné.
7. Přidejte objekt nabídka/seznam `USER_asUserNames` a přiřaďte mu jedinou událost Při klepnutí.
8. Vytvořte metodu objektu pro nabídku/seznam USER_asUserNames následovně nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: USER_asUserNames [ObecnéInformace];" ObecnéInformace.v"
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Uloží změny do přístupu uživatele

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

USER_SaveUserAccessArrays
    `Konec metody
```

9. Přidejte posuvnou oblast `USER_asUserTableStatus` a přiřaďte ji jedinou událost, kterou bude odpovídat Při klepnutí.
10. Vytvořte metodu objektu této posuvné oblasti USER_asUserTableStatus následovně nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: USER_asUserTableStatus [ObecnéInformace];" ObecnéInformace.v"
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Obnoví array přístupů pro vizuální zobrazení

  <>f_Version6x30:=True
```

```
  <>fK_Wilbur:=True

End if

USER_ChangeTableAccess
    `Konec metody
```

11. Přidejte posuvnou oblast pojmenovanou `USER_asUserTableNames` a přiřaďte ji jedinou událost na kterou bude odpovídat Při klepnutí.
12. Vytvořte metodu objektu této posuvné oblasti USER_asUserTableNames následovně nebo importujte pomocí textového editoru:

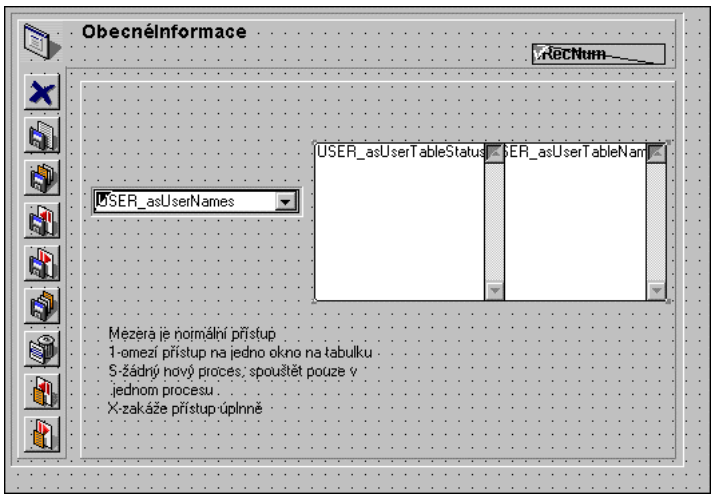
```
If (False)
      ` Metoda: USER_asUserTableNames [ObecnéInformace];" ObecnéInformace.v"
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Obnoví array přístupu uživatele pro vizuální zobrazení

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

USER_ChangeTableAccess
    `Konec metody
```

13. Upravte velikost těchto dvou posuvných oblastí tak, aby byly stejné výšky a vytvořte z nich skupinu.
14. Vytvořte metodu formuláře pro [ObecnéInformace] ObecnéInformace.v následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Form Method: GeneralInformation.i
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Připraví předvolby uživatele pro obecné informace

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

End if

    ` Deklarace lokálních proměnných
C_LONGINT($LFormEvent)
```

```
$LFormEvent:=Form event

Case of
   ś ($LFormEvent=On Load)
      QUERY([zStruktura];[zStruktura]ČísloTabulky=0)
      SELECTION TO ARRAY([zStruktura]PlatnýNázev;USER_asUserTableNames;
                  [zStruktura]ČísloPole;USER_aLTableNumber)
      SORT ARRAY(USER_asUserTableNames;USER_aLTableNumber)
      ARRAY STRING(2;USER_asUserTableStatus;Size of
array(USER_aLTableNumber))

      ALL RECORDS([zUživatelé])
      SELECTION TO ARRAY([zUživatelé]JménoUživatele;USER_asUserNames)
      SORT ARRAY(USER_asUserNames)
      USER_asUserNames:=1

      USER_FillUserAccessArrays

   ś ($LFormEvent=On Validate)
      USER_SaveUserAccesArrays
End case
```

15. Přidejte do formuláře statický text: Mezera je normální přístup
    1-omezí přístup na jedno okno na tabulku
    S-žádný nový proces, spouštět pouze v jednom procesu
    X-zakáže přístup úplnně



16. Vytvořte metodu projektu PROCESS_LUserLimit následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
```

```
      ` Metoda: PROCESS_LUserLimit
      ` Kurz programovani ACI University
      ` Autor: Kent Wilbur
      ` Datum: 3/17/97

      ` Účel: Zkontroluje uživatelské limity na proces

   <>f_Version6x30:=True
   <>fK_Wilbur:=True

End if

      ` Definovat parametry
C_LONGINT($0)                    ` Vrací uživatelský limit
C_STRING(31;$1;$sTableName)

      ` Předefinuje na základě parametrů
$sTableName:=$1

$0:=0
   MESSAGES OFF
   QUERY([zStruktura];[zStruktura]PlatnýNázev=$sProcessName;*)
   QUERY([zStruktura]; & ;[zStruktura]ČísloTabulky=0)
   MESSAGES ON
         ` Získá limit uživatele je-li
   If (Records in selection([zStruktura])>0)
            ` Zapamatuje si čísla tabulek pro tabulky uložené v poli ČísloPole
      $0:=Num(Substring([zUživatelé]TypPřístupu;[zStruktura]ČísloPole;1))
   End if
      `Konec metody
```

17. Nahraďte metodu projektu PROCESS_LSpawnProcess následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
      ` Metoda: PROCESS_LSpawnProcess (string; integer; string; bool; bool;bool) ->
longint
      ` ACI Univerzita kurzy programování
      ` Generic ACI Shell Programming
      ` Vytvořeno: Jim Steinman
      ` Datum: 16/1/97

      ` Returns -> longint - process id, nula pokud nebyl proces vytvořen
      ` ----------------------------------------------------------

      ` Účel: Je místo pouhého volání funkce New Process.
      ` Zabrání vytvoření procesu jestliže přepínač <>fQuit existuje.
      ` Tato metoda vytvoří nový proces řízeným způsobem.
      ` Možnost upozornit uživatele, když proces nemusí být vytvořen
      ` z důvodu nedostatku paměti. Vývojář řídí zda předaná
      ` metoda má být provedena ve více než jednom procesu (více oken).
```

```
      ` umožňuje vyhledat pozastavené procesy a vzbudit je.


   <>fGeneric:=True
   <>f_Version6x20:=True
   <>f_Version6x30:=True
   <>fJ_Steinman:=True

      ` Modified 3/17/97
   <>fK_Wilbur:=True

End if

      ` Definovat parametry a přeřazení místních proměnných
C_LONGINT($0)          ` ID procesu, nula pokud nebyl vytvořen.
C_STRING(31;$1;$sMethod)          ` Metoda ke spuštění v novém procesu.
C_LONGINT($2;$LStack)                  ` Stack velikost (paměť pro procesové
proměnné).
C_STRING(31;$3;$sProcessName)` Název vytvořeného procesu.
C_BOOLEAN($4;$fInformUser)    ` Informuje uživatele, že není dost paměti ke spuštěnbí
procesu.
C_BOOLEAN($5;$fAllowMultipleViews)   ` (Volitelné) Umožní více spuštění.
C_BOOLEAN($6;$fReclaimPausedProcess)` (Volitelné) Žádá navrácení přerušného
procesu


C_LONGINT($LState)                        ` Stav procesu
C_LONGINT($i)                          ` opakovací čítač
C_LONGINT($LLocation)                  ` Pozice názvu procesu v array
C_STRING(255;$sMessage)                ` Zpráva uživateli o nedostatku paměti.
C_LONGINT($LUserLimit)              ` Určitý limit uživatele pro tento proces

If (Not(<>fQuit))     ` Nezačít proces pokud se vypíná databáze
   $sMethod:=$1
   $LStack:=$2*1024
   $sProcessName:=$3
   $fInformUser:=$4
   $fAllowMultipleViews:=False
   $fReclaimPausedProcess:=False

      ` Nejdříve zkontroluj zda je toto proces pro tabulku
   $LUserLimit:=PROCESS_LuserLimit(Table name (pTable))

      ` Je to povolený proces
   If ($LUserLimit<2)
      If (Count parameters>4)
         If ($LUserLimit=0)              ` Žádné omezení uživateli
            $fAllowMultipleViews:=$5
         End if
         If (Count parameters>5)
```

```
            $fReclaimPausedProcess:=$6
         End if
      Else
         $fAllowMultipleViews:=False
      End if

      $Lpid:=Process number($sProcessName)

      If ($Lpid=0)     ` Již neběží žádný proces

         $Lpid:=New process($sMethod;$LStack;$sProcessName)

      Else
         $0:=0
         If ($fReclaimPausedProcess)
            If (PROCESS_fCheckProcessAvailable ($Lpid)) ` Restartovat dostupný proces
               $0:=$Lpid
            End if

            If (($fAllowMultipleViews) & ($0=0))
               $LLocation:=Find in array(<>PROCESS_asViewName;$sProcessName)

               If ($LLocation>0)          ` Máme dodatečné procesy
                  $i:=1
                  While (($0=0) & ($i<= <>PROCESS_aiViewNumber{$LLocation}))
                     $Lpid:=Process number($sProcessName+String($i))
                     If (PROCESS_fCheckProcessAvailable ($Lpid))     ` Restartovat
dostupný proces                                          $0:=$Lpid
                     End if
                     $i:=$i +1
                  End while
            End if                                 ` Dodatečné procesy
         End if                                    ` Více spuštění
      End if                                       ` Navrácení procesů


      If (($fAllowMultipleViews) & ($0=0))  ` Pokud je nastaven flag více spuštění
         $Lpid:=New process($sMethod;$LStack;$sProcessName+
String(PROCESS_LNextView ($sProcessName)))
      Else
         RESUME PROCESS($Lpid)
         SHOW PROCESS($Lpid)
         BRING TO FRONT($Lpid)
      End if

      End if

      If ($Lpid=0)     ` Něco je špatně
         If ($fInformUser)                ` Is the warn the user flag set
         $sMessage:="Nedostatek paměti k splnění akce. "
         $sMessage:=$sMessage + "Zkuste uzavřít nějaká okna k uvolnění paměti."
```

```
                BEEP
                ALERT($sMessage)
              End if

            End if
            $0:=$Lpid

❖         Else      ` Předvolby uživatele oznamují omezení pro tuto tabulku
❖           If ($LUserLimit = 2)                ` Nezačínáme nový proces pouze voláme metodu
❖             EXECUTE($sMethod)
❖           Else
❖             ALERT("V tuto dobu nemáte přístup k této položce nabídky."+ Char(Carriage
return)+"Kontaktujte svého administrátora.")
❖           End if
❖
❖             $0:=0
❖
❖         End if
        Else
          ALERT ("Databáze se vypíná. Nelze začít nový proces.")
          $0:=0
        End if
            ` Konec metody
```

18. Upravte metodu projektu GEN_ModifySelection následovně:

```
        If (False)
            ` Metoda: GEN_ModifySelection
            ` Kurz programovani ACI University
            ` Genericke metody z nastroju ACI
            ` Created by: Jim Steinman
            ` Datum: 1/15/97

            ` Účel: Zobrazí výstupní formulář pro tabulku určenou ukazatelem pTable.

          <>fGeneric:=True
          <>f_Version6x10:=True
          <>f_Version6x20:=True
          <>f_Version6x30:=True
          <>fJ_Steinman:=True

            ` Upraveno: 1/17/97
          <>fK_Wilbur:=True
            ` Upraveno: 3/17/97
          <>fK_Wilbur:=True
        End if

        C_LONGINT($LWindowID)
❖       C_LONGINT($LUserLimit )

❖       $LUserLimit:=PROCESS_LUserLimit(Table name(pTable))
```

```
Repeat
   fProcessAvailable:=False

   MENU_SetMenuBar(3)

   PRELIM_CustomQueryOrder

   If (OK=1)
      INPUT FORM(pTable->;"Input";*)          ` Nastaví na navrženou velikost
      OUTPUT FORM(pTable->;"Output")

      $LWindowID:=WIN_LNewWindow (-1;-1;Cascading window;Plain no zoom box
window)

      CREATE EMPTY SET(pTable->;"UserSet")

      PROCESS_UpdateWindowArray ("";Current process)

      WIN_OutputWindowTitle
      MODIFY SELECTION(pTable->;*)

      PROCESS_UpdateWindowArray ("";Current process)

      CLOSE WINDOW

      GEN_ClearSelection

   End if

   If ($LUserLimit <2)
      fProcessAvailable:=True
      PAUSE PROCESS(Current process)
   End if
UNTIL( <>fQuit)
      ` Konec metody
```

19. Upravte metodu projektu M_Preferences následovně:

```
…
   If (OK=1)
      USER_SaveUserAccessArrays
      VALIDATE TRANSACTION
   Else
      CANCEL TRANSACTION
   End if
…
```

20. Přejděte do prostředí Vlastní nabídky a otestujte tyto nové rysy. Omezte přístup do tabulky Produkty na žádný, do Faktur na pouze jeden a testujte výsledky.

21. Obnovte nastavení na plnný přístup.

30. Export dat pomocí uložených procedur

Při normálním způsobu exportu, kdy datový soubor vzniká na počítači klienta, server tak či onak musí odvést téměř všechnu práci při exportu dat a kromě toho musí přemístit všechna data přes síť na klienta (včetně polí, která nejsou zahrnuta v exportovaném souboru). Na serveru v případě, že server bude sám zapisovat vznikající exportní soubor na disk nedojde k zvýšení zavádění záznamů, ale podstatně se sníží provoz na síti. Protože data včetně nadbytečných dat záznamů nebudou putovat sítí se rovněž výrazně sníží čas operace. Kód z této kapitoli byl testování na 1101 záznamu v tabulce Zákazníci. Všechny exporty byly prováděny v interpretačním módu.

Na klientu	00:08:42
Uložené procedury na serveru	00:01:00

Pro tato řešení opět platí testujte, testujte, testujte na své reálné databázi.

30.1. Možnosti pro přesun textového souboru zpět uživateli

30.1.1. BLOBy

Výhody: Mnoho

Nevýhody: Potenciálně vyžadují velké množství paměti, může být i více než je k dispozici.

30.1.2. Síťová složka sdílená serverem

Výhody: Použití síťové složky z jiného počítače místo serveru je rovněž druh síťového serveru. Toto řešení nevyžaduje dodatečnou paměť na žádném stroji sítě.

Nevýhody: Jestliže stroj se síťovou složkou zkrachuje může to zastavit provádění na 4D Serveru se zobrazením dialogu, který čeká na potvrzení. Uživatel pak musí přejít ke stroji se síťovou složkou, obnovit jeho provoz a na serveru potvrdit dialog.

30.1.3. Použití serveru jako síťové složky

Výhody: Nevyžaduje více paměti na žádném ze strojů. 4D Server se nemůže přerušit jestliže zkrachuje jiný počítač.

Nevýhody: Počítač 4D Server musí rovněž působit jako síťový server a pokud je přetížen může to výrazně zpomalit i celý provoz řešení 4D klient/server.

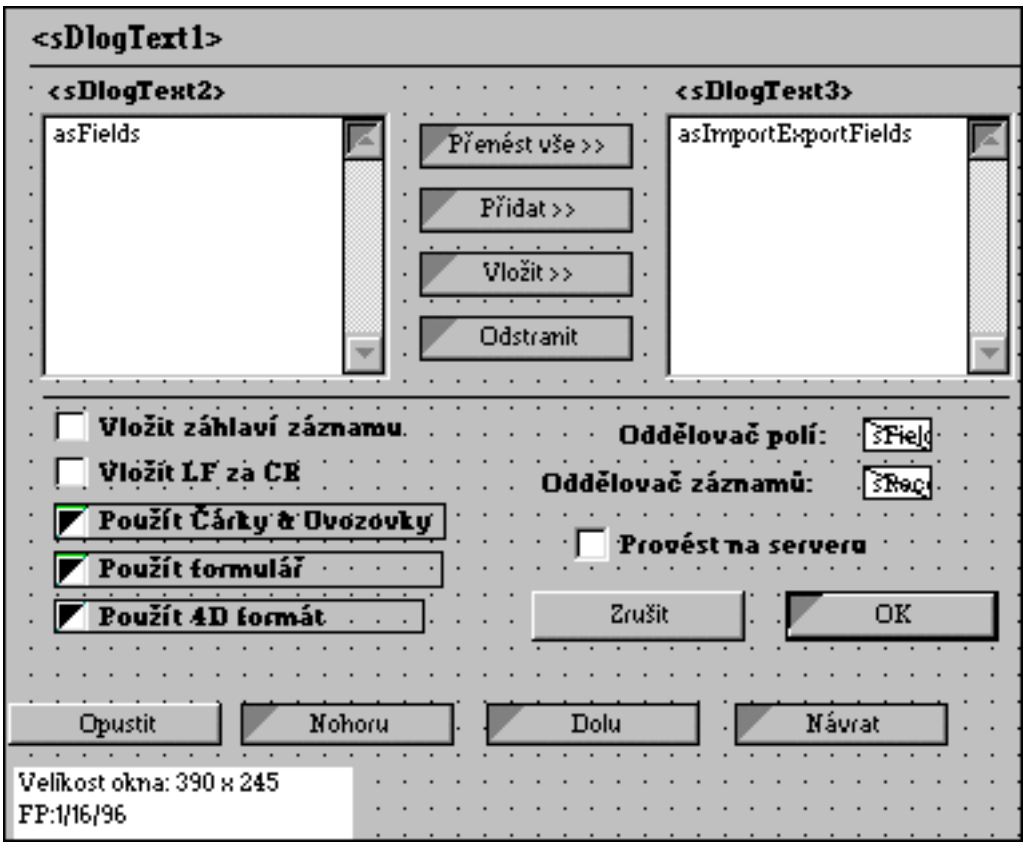
30.2. Příkladový kód pro export s použitím uložených procedur

Protože žádná z metod nenabízí zcela přesvědčivé výhody pro přenos datového souboru zpět uživateli, tento příklad je doveden pouze do fáze vytvoření souboru na disku. Na vás již bude určit, který disk to bude a co dělat, když bude tento diskový soubor dokončen.

30.2.1. Export dat na serveru

1. Otevřete formulář [zDialogy];"IMPEXP_ImportExport".
2. Přidejte nad tlačítka nové zaškrtávací políčko:

Typ objektu	Název objektu	Vlastnosti	Text
Zaškrtávací políčko	IMPEXP_ckOnServer	Styl: Logické	Provést na serveru



3. Upravte metodu formuláře [zDialogy];"IMPEXP_ImportExport" následovně:

```
…
    BUTTON TEXT(bExport;Substring(sDlogText1;1;6)) ` Nastaví tlačítko OK na
"Import" nebo "Export".

❖       If ((Application type # 4D Client) | (sDlogText = "Import"))
❖          SET VISIBLE(IMPEXP_ckOnServer;False)
❖       End if
End case
    `Konec metody
```

4. Nahraďte metodu projektu IMPEXP_ImportExportDialog následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
    ` Metoda: IMPEXP_ImportExportDialog (string)
    ` ACI Univerzita kurzy programování
    ` Generic ACI Shell Programming
    ` Vytvořeno: Jim Steinman
    ` Datum: 16/1/97

    ` Účel: připraví Import Export.
    ` všechny záznamy v platném výběru jsou exportovány.

  <>fGeneric :=True
  <>f_Version6x20 :=True
  <>fJ_Steinman :=True

End if

    ` Přiřazení parametrů
C_STRING(6;$1;$sÚčel)              ` "Import" nebo "Export"

    ` Vytvoření místních proměnných
C_LONGINT($Lpid)                       ` ProcesID
C_LONGINT($LWidth)                     ` Šířka okna
C_LONGINT($LHeight)                        ` Výška okna
C_LONGINT($LWindowID)          ` ID okna
C_LONGINT($LFormat)                        ` Druh formátu použitého při exportu
C_LONGINT($LTableNumber)
C_STRING(31;$sProcessName)     ` Název procesu ke spuštění
C_BLOB($oParameters)                       ` Blob do kterého se přiřadí parametry
exportu

    ` Přiřazení parametrů k proměnným
$sÚčel:=$1

  ` Vytvoření array k uložení názvů polí
ARRAY STRING (31; asFields; 0)
```

```
   ` Vytvoření array k uložení ukazatelů k polím
ARRAY INTEGER (aiFields; 0)

   ` Nastavení array exportních polí
ARRAY STRING (31; asImportExportFields; 0)
ARRAY INTEGER (aiImportExport; 0)
GEN_FieldsToArray (pTable; ->asFields; ->aiFields; False;False)

   ` Výchozí hodnota pro formát dialogu.
if (<>WIN_fIsWindows)
   sFieldDelimiter:="44"
   sRecordDelimiter:="10"
   ckCommasQuote:=1
Else
   sFieldDelimiter:="9"
   sRecordDelimiter:="13"
   ckCommasQuote:=0
End if
ckInclude:=0
sDlogText1:=$sÚčel + " Dat…"
sDlogText2:=Table name (pTable) + " Pole"
sDlogText3:=$sÚčel + " Pořadí"

$LWidth:=390 `   cílová šířka
$LHeight:=245 `  cílová šířka
$LWindowID :=WIN_LNewWindow ($LWidth; $LHeight; <>WIN_LCentered;
<>WIN_LModal)
DIALOG ([zDialogs]; "IMPEXP_ImportExport")
CLOSE WINDOW

If (bExport = 1)

   Case of

      ś (ckForm = 1)                        ` Užít existujíc formulář
         $LFormat:=3
      ARRAY INTEGER (aiImportExport; 0)

      ś (ck4DFormat = 1)                    ` Použít vnitřní formát 4D
         $LFormat:=2
      ARRAY INTEGER (aiImportExport; 0)

      ś (ckInclude = 1)              ` Vložit záhlaví záznamu
         $LFormat:=1

      Else
         $LFormat:=0
   End case

   If ($sÚčel = "Export")  `          Výběr použít pouze pro export
```

```
      SET CHANNEL(12;"")`          Vytvořit nový soubor
      COPY NAMED SELECTION (pTable->; "<>ExportSelection")
   Else
      SET CHANNEL(10;"")`          Otevřít existující soubor pro import
   End if

   If (OK=1)
      SET CHANNEL(11)`                  Pro nyní zavřít soubor

         ` Připravit BLOB pro přiřazení exportovaných polí do nového procesu
      $LTableNumber:=Table(pTable)
      VARIABLE TO BLOB ($LTableNumber;$oParameters;*)
      VARIABLE TO BLOB(aiImportExport;$oParameters;*)
      VARIABLE TO BLOB($LFormat;$oParameters;*)
      VARIABLE TO BLOB(ckCRLF;$oParameters;*)
      VARIABLE TO BLOB(ckCommasQuote;$oParameters;*)
      VARIABLE TO BLOB(sFieldDelimiter;$oParameters;*)
      VARIABLE TO BLOB(sRecordDelimiter;$oParameters;*)
      VARIABLE TO BLOB(Document;$oParameters;*)
❖     If ((IMPEXP_ckOnServer = 1) & ($sÚčel = "Export"))
❖        ARRAY LONGINT($aLRecordNumbers;0)
❖        SELECTION TO ARRAY(pTable->;$aLRecordNumbers)
❖        VARIABLE TO BLOB($aLRecordNumbers;$oParameters;*)
❖        CLEAR NAMED SELECTION("<>ExportSelection")   ` nepotřeba pro server
❖     End if


         ` Začít export v odděleném procesu

      $sProcessName :=$sÚčel+"ing "+Substring(Table name(pTable);1;4)
      $Lpid :=Process number($sProcessName)
      If ($Lpid>0)                              ` Umožnit více procesům pracovat ve
stejnou chvíli
      $sProcessName :=Substring ($sProcessName;1;29)+String(PROCESS_LnextView
($sProcessName))
                                                ` Limit na 29 znaků pro vložení čísla pokud je
potřeba
      End if

❖     If (IMPEXP_ckOnServer=1)
❖        $Lpid:=Execute on server("P_IMPEXP_"+$ sÚčel
+"Data";32*1024;$sProcessName;
                  $oParameters;True)
❖        DELETE DOCUMENT(Document)
❖        Else
      $Lpid :=New process("P_IMPEXP_"+$sÚčel+"Data";32*1024;$sProcessName;
$oFieldPointers;True)
❖        End if
      SET BLOB SIZE ($oParameters;0)

      If ($Lpid=0)
```

```
        If ($sÚčel="Export")                ` Výběr použít pouze pro export
          CLEAR NAMED SELECTION("<>ExportSelection")      ` vymazat jestliže jiné
procesy nebyli vytvořeny
        End if

        ALERT("Není dost paměti pro "+$sÚčel)
      End if

   End if                                   ` OK

End if                                      ` bExport

   ` Konec metody
```

5. Nahraďte metodu projektu IMPEXP_ExportData následujícím způsobem nebo importujte pomocí textového editoru:

```
If (False)
  ` Metoda: P_IMPEXP_ExportData (BLOB; boolean)
  ` ACI Univerzita kurzy programování
  ` Generc ACI Shell Programming
  ` Vytvořeno: Jim Steinman
  ` Datum: 16/1/97

  <>fGeneric:=True
  <>f_Version6x20:=True
  <>fJ_Steinman:=True

End if

` Vytvoření parametrů
C_BLOB($1)  ` BLOB parametrů exportu
C_BOOLEAN($2;$fBackground)  ` Proces na pozadí (volitelné; True = pozadí)

` vytvoření místních proměnnîch
C_POINTER($pTable)  ` ukaztel k tabulce
C_LONGINT($LFieldType)  ` Součet parametrů
C_LONGINT($LFormat)  ` Umístění okna
C_BOOLEAN($fCRLF)  ` Vložit řádek po return
C_BOOLEAN($fCommasQuotes)  `  Použít čárky
C_BOOLEAN($fStoredProcedure)  ` Prohíhá jako stored procedure
C_STRING(2;$sFieldDelimiter)  ` Odděovač polí
C_STRING(2;$sRecordDelimiter)  ` Oddělovač záznamů
C_STRING(2;$sDelimiter)  ` dočasný oddělovač
C_STRING(2;$sQU)  ` Uvedení znaků pro tečky & uvozovky Export
C_LONGINT($Lpid)  ` ID procesu Logoff testu
C_LONGINT($LNumberOfFields)  ` Počet polí k exportu
C_LONGINT($LNumberOfRecords)  ` Počet záznamů k exportu
C_LONGINT($LPosition)  ` Pozice znaků
```

```
C_LONGINT($LWindowID)  ` ID okna pro zprávy
C_LONGINT($LTable)  ` Číslo tabulky
C_LONGINT($LOffset)  ` Offset pro rozluštění BLOB
C_LONGINT($i;$j)  ` opakovcí počítadlo
C_TIME($hDocumentReference)  ` Okdaz k dokumentu
C_TEXT($tBuffer)  ` buffer dat
C_TEXT($tField)  ` možná data polí
ARRAY INTEGER($aiFieldNumbers;0)

` definovat výchozí hodnoty
$fBackground:=False
$fStoredProcedure:=False

` přiřazení BLOB parametrů k proměnným
$LOffset:=0
BLOB TO VARIABLE($1;$LTable;$LOffset)
BLOB TO VARIABLE($1;$aiFieldNumbers;$LOffset)
BLOB TO VARIABLE($1;$LFormat;$LOffset)
BLOB TO VARIABLE($1;$i;$LOffset)
$fCRLF:=($i=1)  ` 1 značí ano
BLOB TO VARIABLE($1;$i;$LOffset)
$fCommasQuotes:=($i=1)  ` 1 značí ano
BLOB TO VARIABLE($1;$sFieldDelimiter;$LOffset)
$sFieldDelimiter:=Char(Num($sFieldDelimiter))  ` převedení čísel řetězce na char
BLOB TO VARIABLE($1;$sRecordDelimiter;$LOffset)
$sRecordDelimiter:=Char(Num($sRecordDelimiter))  ` převedení čísel řetězce na char
BLOB TO VARIABLE($1;Document;$LOffset)

If (Application type=4D Server )
 $fStoredProcedure:=True
 BLOB TO VARIABLE($1;$aLRecordNumbers;$LOffset)
 $LPosition:=Position("\";Document)
 If ($LPosition>0)  ` Stáhnout oddělovače polí
  $sDelimiter:="\"
 Else
  $sDelimiter:=":"
 End if
 Repeat
  $LPosition:=Position($sDelimiter;Document)
  If ($LPosition>0)
   Document:=Substring(Document;$LPosition+1)
  End if
 Until ($LPosition=0)
 If (<>WIN_fIsWindows)  ` vložit windws extension pokud to bude nutné
  If (Position(".TXT";Document)=0)
   Document:=Document+".TXT"
  End if
 End if
End if

SET BLOB SIZE($1;0)
```

```
If (Count parameters=2)
  $fBackground:=$2
End if

$LNumberOfFields:=Size of array($aiFieldNumbers)
ARRAY POINTER($pFields;$LNumberOfFields)
For ($i;1;$LNumberOfFields)
  $pFields{$i}:=Field($LTable;$aiFieldNumbers{$i})  ` ukazatele k polím
End for

If ($fCRLF)
  $sRecordDelimiter:=$sRecordDelimiter+Char(Line feed )
End if
$pTable:=Table($LTable)  `Ukazatel k exportované tabulce

If ($fBackground)  ` ExportData běží ve vlastním procesu

  MENU_SetMenuBar (1)  ` nastavit nabídku pro proces tak, že nabídka bude prázdná
  `             když bude zobrazené okno exportu.

  <>LBackground:=<>LBackground+1
  If (Not($fStoredProcedure))
    USE NAMED SELECTION("<>ExportSelection")
    CLEAR NAMED SELECTION("<>ExportSelection")
  End if
  SET CHANNEL(10;Document)

Else   ` ExportDat byl volán procesem který otevřel soubor

  SET CHANNEL(12;"")

End if

If (OK=1)

  If ($fBackground)
    HIDE PROCESS(Current process)  ` Přesunout napozadí
  End if

  If ($fStoredProcedure)
    $LNumberOfRecords:=Size of array($aLRecordNumbers)
    GOTO RECORD($pTable->;$aLRecordNumbers{1})
  Else
    FIRST RECORD($pTable->)
    $LNumberOfRecords:=Records in selection($pTable->)
  End if

  $sTableName:=Table name($pTable)

  $LWindowID:=WIN_LNewWindow
(200;30;<>WIN_LCentered;<>WIN_LStandardNoResize;"Export")
```

```
If ($fBackground)
  PROCESS_UpdateWindowArray ("";Current process;True)
End if

$Lpid:=Process number("$Logoff")  ` je zde test logoff
If ($Lpid>0)
  ON EVENT CALL("")  `vypnout zachytávání událostí
  PAUSE PROCESS($Lpid)
End if


GOTO XY(6;0)
MESSAGE("Export "+$sTableName+"... ")

Case of

  ś (($LFormat=0) | ($LFormat=1))  ` Text neboMerge formát

    ` Vytvořit array typů polí
    ARRAY LONGINT(aLFieldTypes;$LNumberOfFields)

    For ($i;1;$LNumberOfFields)
      GET FIELD PROPERTIES($pFields{$i};$LType)
      aLFieldTypes{$i}:=$LType
    End for

    If ($fCommasQuotes)  ` Děláme tečky a uvozovky
      $sQU:=<>sQU
    Else
      $sQU:=""
    End if

    If ($LFormat=1)  ` Merge Format. Nejdříve exportovat názvy polí
      $sDelimiter:=$sFieldDelimiter
      $tBuffer:=""

      For ($i;1;$LNumberOfFields)

        If ($i=$LNumberOfFields)  ` exportujeme poslední název pole
          $sDelimiter:=$sRecordDelimiter
        End if

        $tBuffer:=$tBuffer+$sQU+GEN_sFriendlyName (Field
name($pFields{$i}))+$sQU+$sDelimiter
      End for

      SEND PACKET($tBuffer)
    End if

    ` nyní export záznamů
```

```
$tBuffer:=""

For ($i;1;$LNumberOfRecords)
 $sDelimiter:=$sFieldDelimiter

 For ($j;1;$LNumberOfFields)

  If ($j=$LNumberOfFields)  ` exportujeme poslední pole v záznamu
   $sDelimiter:=$sRecordDelimiter
  End if

  Case of
   ś ((aLFieldTypes{$j}#Is Alpha Field ) & (aLFieldTypes{$j}#Is Text ) &
    (aLFieldTypes{$j}#Is Boolean ))  ` překonvertovat na řetězec

    If ((aLFieldTypes{$j}=Is Real ) | (aLFieldTypes{$j}=Is Integer ) |
     (aLFieldTypes{$j}=Is LongInt ))  ` jestliže to je číslo
     $tData:=(String(($pFields{$j})->)+$sDelimiter)
    Else
     $tData:=$sQU+(String(($pFields{$j})->)+$sQU+$sDelimiter)
    End if

   ś (aLFieldTypes{$j}=Is Boolean )  ` je to logické

    If ($pFields{$j}->)
     $tData:=$sQU+"True"+$sQU+$sDelimiter
    Else
     $tData:=$sQU+"False"+$sQU+$sDelimiter
    End if

   Else

    $tData:=$sQU+(($pFields{$j}->)+$sQU+$sDelimiter)
  End case

  If ((Length($tBuffer)+Length($tData))>16000)
   SEND PACKET($tBuffer)
   $tBuffer:=$tData
   $tData:=""
  Else
   $tBuffer:=$tBuffer+$tData
  End if

 End for


 If ($fStoredProcedure)
  GOTO RECORD($pTable->;$aLRecordNumbers{$i})
 Else
  NEXT RECORD($pTable->)
 End if
```

```
      If ($i%25=0)  ` Nehrnout dopředu věci pouze k zobrazení
        GOTO XY(6;1)
        MESSAGE("Zznam "+String($i)+" z "+String($LNumberOfRecords))
      End if

    End for

    If ($tBuffer#$tData)
      SEND PACKET($tBuffer)
    End if

  ś ($LFormat=2)  ` 4D Formát

    For ($i;1;$LNumberOfRecords)

      SEND RECORD($pTable->)
      If ($fStoredProcedure)
        GOTO RECORD($pTable->;$aLRecordNumbers{$i})
      Else
        NEXT RECORD($pTable->)
      End if

      If ($i%25=0)  ` Nehrnout dopředu věci pouze k zobrazení
        GOTO XY(6;1)
        MESSAGE("záznam "+String($i)+" z "+String($LNumberOfRecords))
      End if

    End for

  ś ($LFormat=3)  ` Použít formulář ImportExport
    ` Účel: Umožnit exportovaným záznamům do platné tabulky použít formulář
    ` Poznámka: Vývojář musí mít vytvořený formulář s názvem ImportExport pro tuto
pr
    FldDelimit:=Num($sFieldDelimiter)
    RecDelimit:=Num($sRecordDelimiter)

    GEN_SetFormName ($pTable;"ImportExport")
    EXPORT TEXT($pTable->;"")
    GEN_SetFormName ($pTable;"Seznam")

  End case

  CLOSE WINDOW
  SET CHANNEL(11)  ` Zavřít dokument

  If ($fBackground)  ` Vymazat čítač procesů na pozadí
    PROCESS_UpdateWindowArray ("";Current
process;True)<>LBackground:=<>LBackground-1
  End if
```

```
If (($Lpid>0) & (<>LBackground=0))
  <>fUserPresent:=True    ` nastavit pokud je uživatel zde
  RESUME PROCESS($Lpid)
End if


End if  `    OK=1

` Konec metody
```

6. Nahraďte metodu projektu IMPEXP_ExportDataOptimized importem pomocí textového editoru:

Kód pro tuto metodu zde není uveden, je podobný tomu v předchozím bodě, ale obsahuje kód optimalizovaný pro rychlejší export v kompilované databázi. Provádí se však mnohem pomaleji jestliže databáze běží v interpretovaném módu.

7. Nahraďte metodu databáze On Server Startup následujícím způsobem nebo importujte pomocí textového editoru:

```
   ` Database Method: On Server Startup
   ` Kurz programovani ACI University
   ` Genericke metody z nastroju ACI
   ` Autor: Kent Wilbur
   ` Datum: 3/17/97

   ` Účel: Inicializuje informace na serveru

  <>f_Version6x30:=True
  <>fK_Wilbur:=True

    ` Deklarace lokálních proměnných
C_LONGINT($Lpid)

<>fQuit:=False
<>hTimeDifference:=†00:00:00†
<>MainProcess:=Current process
<>LBackground:=0

WIN_IsWindows
WIN_InitializeWindowVariables
INITIALIZE_Characters

ARRAY STRING(35;<>PROCESS_asWindowName;0)
ARRAY INTEGER(<>PROCESS_aiWindows;0)
<>LWindowPaletteUpdater:=0

$Lpid:=PROCESS_LSpawnProcess
("P_InitializeIPArrays";32;"InitializeServer";False;False)
```

```
` Konec metody
```

8. Otevřete databázi pomocí 4D Server a 4D Client a otestujte tyto nové techniky.

31. Závěr

Pokusili jsme se vám v tomto kurzu ukázat škálu různých technik programování. Všechno co zde bylo probíráno rozhodně nemůže být považováno za pouze jediný způsob k vyplnění zadaných úloh či dokonce nejlepší způsob k vyplnění úlohy. Je potřeba mít stále na paměti, že každá databáze je jiná a potřebuje vyladění podle své podstaty.

Příkazy a funkce probírané v tomto kurzu mají ještě mnoho dalších způsobů použíti než pouze ty, které jsme probrali. Vyzkoušejte si tedy jejich různé další použití, vyzkoušejte kód jiných databázi a otestujte jak pracuje.

Nejdůležitější, ale je: Nebojte se zkusit vše co vás napadne, pouze chybami se člověk učí.

Odpovědi Kvíz

Okna se změnou velikosti

Tabulka	Minimum Podélně	Minimum Svisle
Zákazníci	600	390
Faktury	600	395
Produkty	350	350

Další chyby mazání

- Můžete je vyjmenovat?

  -9974 Záznam již vymazán
  -9986 Záznam uzamčen během automatické akce mazání
  -9991 Chyba oprávnění

- Co se stane s vaším kódem, když tyto chyby nastanou při pokusu o vymazání záznamů?

  Dostanete chybové hlášení a číslo chyby spolu s tímto hlášením bude zobrazeno na obrazovku.

- Můžeme vylepšit náš kód tak, aby ovládal rovněž některé z těchto chyb?

  Ano a to poskytnutím vlastních rozšířených hlášení uživateli, když nastane tato chyba.

- Můžeme poskytnout uživateli informaci o záznamu, který způsobil problém.

  Ne, protože vymazávané záznamy jsou z platného výběru odstraněny a proto, když nastane chyba platný výběr je prázdný.

Obnovat array

- Co je rychlejší úprava prvku array nebo kontrola zda se změnila hodnota pole pomocí Old?

Konečná odpověď je Záleží na. V našem případě potřebujeme zkontrolovat zda se změnilo 6 polí . Funkce Old kontroluje pole v paměti (změněná pole) proti polím v paměti (stará hodnota). Musíme to však udělat šestkrát a jestliže se libovolné pole změnilo, musíme najít pomocí Find in array správný prvek v array rovněž v paměti a pak nahradit hodnoty v array. Použijeme Find in array jednou. Nahrazení hodnoty prvku pokud již byl nalezen není podstatné. V každém případě mluvíme o mikrosekundách a to co je skutečně lepší bychom mohli určit až na milionech iterací ne pouze na šesti.

- Kdy potřebujeme třídit array?

  Máme zde použito ukládání v reálném čase. Array potřebujeme třídit jestliže se změnila hodnota v poli Firma, třídění array zabere určitý čas a pokud není nezbytné měli bychom se mu vyhnout

- Jak můžete zajistit, aby změna array probíhala pouze v jednom procesu současně?

  Semaforem. Buď lokálním nebo globálním v závislosti na rozsahu array umístěném buď na klientu či na serveru.

Příkaz MENU BAR

- Kde byly umístěny všechny změny?

GEN_ModifySelection
IMPEXP_ExportData
IMPEXP_ExportDataOptimized
P_DailySales
P_IMPEXP_ImportData
P_LogoffChecker
P_PROCESS_TablesPalette
P_PROCESS_WindowsPalette
WIN_InputWindowTitle
Metoda formuláře [Zákazníci]Výstupní
Metoda formuláře [Faktury]Výstupní
Metoda formuláře [PoložkyFaktury]Výstupní
Metoda formuláře [zDialogy]OutputButtons
Metoda formuláře [zDialogy]OutputButtons.rev

Nalezení polí

- Jak můžete nalézt skutečné pole, jestliže znáte zástupný název pole?
- Jak se můžete na toto pole dotázat?
- Jak můžete toto pole třídit?

Nejdříve se dotážete na zástupný název pole v tabulce zStruktra a tím zjistíte jeho skutečný název či jeho číslo v tabulce.

Pokročilé programování ve 4D

Rejstřík

Pokročilé programování ve 4D

Osobní poznámky

Pokročilé programování ve 4D

Osobní poznámky

ACI Univerzita