Rīgas Tehniska universitāte

Datorzinātnes un Informācijas tehnoloģijas fakultāte

Trešais praktiskais darbs mācību priekšmetā

„Datu bāzu vadības sistēmas”

Datu izgūšana no dažādiem datu avotiem,

izmantojot MS ADO tehnoloģiju

Izstrādāja: Sergejs Terentjevs

2. kurss, 9.grupa

Apl.nr. 061RDB140

Pārbaudīja: lekt. V. Vinogradova

2007./2008. māc. g.

# ANOTĀCIJA

Praktiska darbā izpildes gaita tika izstrādāts lietojums - „SQL vaicājumu redaktors”, kas realizē datu avotu pieslēgšanas mehānismu un ļauj definēt datu atlases kritērijus. Darba pamatā tika izmantoti ActiveX objektu modeļi (ADO jeb ActiveX Data Objects), kuri nodrošina piekļūšanu jebkura tipa datu avotiem un nodrošina ērtu objektorientētu interfeisu.

Darba pārskatā ir apskatīts praktiskā darba izpildes process, apskatītas ADO objektu pielietošanas iespējas piesaistot datu avotus MS Access lietojumam un ir uzskatāmi iztirzātas procedūras un funkcijas, ADO moduļu objektu īpašības un metodes.

Praktiskais darbs tika izstrādāts MS Access 2007 uz *Pentium III*  tipā procesorā *Windows XP Proffesional* vidē. Tajā tika izstrādātas 4 formas, 1 modulis, kurš glabā publiski pieejamas procedūras vai funkcijas, un mainīgos, vairākas notikumu procedūras.

Laboratorijas darba pārskata apjoms: 27 lappuses izklāsta tekstā, tajā ietilpst 7 nodaļas, 3 apakšnodaļas un literatūras saraksts, 21 attēls.

# SATURS

[ANOTĀCIJA 2](#_Toc197820268)

[SATURS 3](#_Toc197820269)

[UZDEVUMA NOSTĀDNE 4](#_Toc197820270)

[IZMANTOTAS TABULAS 5](#_Toc197820271)

[MODEĻA MultipleConnection APRAKSTS 7](#_Toc197820272)

[LIETOJUMA APRAKSTS 8](#_Toc197820273)

[DATU AVOTU SARAKSTA FORMA 8](#_Toc197820274)

[JAUNA SAVIENOJUMA FORMA 12](#_Toc197820275)

[SQL vaicājuma redaktora forma 16](#_Toc197820276)

[SECINĀJUMI 26](#_Toc197820277)

[IZMANTOTA LITERATŪRA 27](#_Toc197820278)

# UZDEVUMA NOSTĀDNE

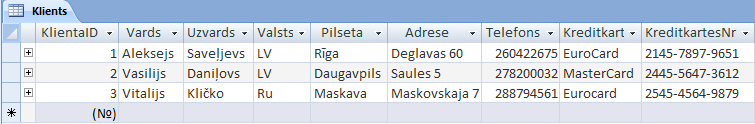
Izveidot lietojumu, kas ļautu lietotājam pieslēgties ārējiem datu avotiem un definēt datu atlases kritērijus (rakstīt SQL pieprasījumus), dinamiski attēlot iegūtus rezultātus. Paredzēt iespēju lietotājam izgūt avota meta datus, kas varētu būt izmantoti datu atlases kritēriju definēšanai.

Darba aprakstā iekļaut elementu veidošanas paskaidrojumu, norādot būtiskās īpašības (tai skaitā izmantotus objektus). Paredzēt iespēju pieslēgties vismaz diviem datu avota veidiem (Access, Excel, ODBC, ...).Ieteicams darba izpildīšanai pielietot ADO tehnoloģiju.

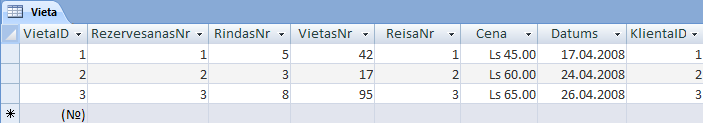
# IZMANTOTAS TABULAS

Praktiskā darba tika izstrādātas trīs tabulas lietojuma testēšanai, no kurām divas tika izveidotas MS Access un viena MS Excel lietotāj programmatūrā.

Datu bāzes pamatideja ir glabāt informāciju par avioreisiem, kurus klienti jau iepriekš rezerve, proti informācija par klientu tiek glabāta Access tabulā *Klients*, savukārt, visa informācija par klienta rezervēto vietu avioreisam tiek glabāta tabula *Vieta*.



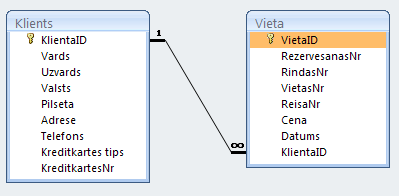
1. att. Tabula *Klients*.



1. att. Tabula *Vieta*.

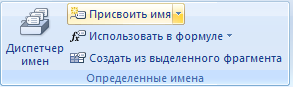
Tabulu veidošanai tika izmantota rīkjosla *Create*,kura piedāvā veidot gan tabulas, gan

pārskatus, gan formas un vaicājumus Access 2007 vidē. Tabulas tika izveidotas *Design View* skatā.Lai definētu saites starp tabulām, ir jāizvēlas *Relationship* komandpoga un jāpārvelk tabulas lauku uz citas tabulas lauku, kuri būs savstarpēji saistīti (mana gadījumā abi šādi lauki ir nosaukti *KlientaID* ). Starp tabulu saite satur viens-pret-daudziem attiecību.

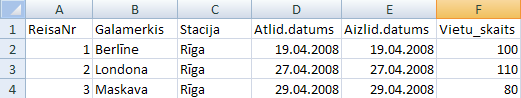


1. att. Relāciju saite starp tabulām.

Savukārt cita tabula ar nosaukums *Reiss*  ir izstrādāta MS Excel un glabā informāciju par pašiem reisiem, t.i., vietu skaitu lidmašīna, atlidošanas un aizlidošanas datumus, atlidošanas staciju un galamērķi. Excel paredz tabulas lauku definēšanu, kas ir ērti tabulas datu atlasīšanai izmantojot SQL vaicājumu (skat. att. 4).



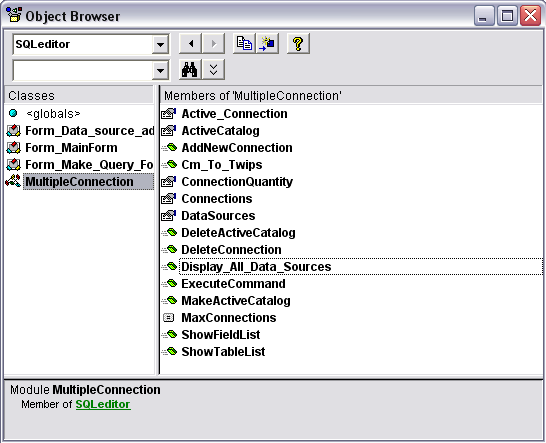
1. att. Excel tabulas lauku nosaukuma *Reiss* definēšana.



1. att. Tabula *Reiss*.

# MODEĻA MultipleConnection APRAKSTS

Visas izveidotas lietojuma procedūras un funkcijas ir uzrakstītas Visual Basic for Application (VBA) valodā un novietotās *MultipleConnecton* modulī. Dotais modelis satur procedūras, kas ir saistītas ar savienojuma izveidošanu, komandas izpildīšanu un rezultātu izvadi.



1. att. Modulis *MultipleConnection*.

Saites starp dažādam procedūrām ir nodrošinātas izmantojot procedūru parametrus un globālos mainīgos. Modelī ir definēti sekojoši globālie mainīgie:

Public Const MaxConnections = 10

Public Connections(1 To MaxConnections) As ADODB.Connection

Public DataSources(1 To MaxConnections) As String

Public Active\_Connection As ADODB.Connection

Public ActiveCatalog As ADOX.Catalog

Public ConnectionQuantity As Integer

*MaxConnections* – maksimāli vienlaikus atvērto savienojumu skaits;

*Connections* – savienojumu masīvs;

*DataSources* – datu avotu adrešu masīvs;

*Active\_Connection* – aktīvais savienojums, t.i., savienojums ar kurā palīdzību dotā momentā ir izpildīta vaicājuma komanda;

*ActiveCatolog* - aktīvais katalogs, t.i., katalogs kurām tiks veidots vaicājums;

*ConnectionQuantity* – atvērto savienojumu skaits.

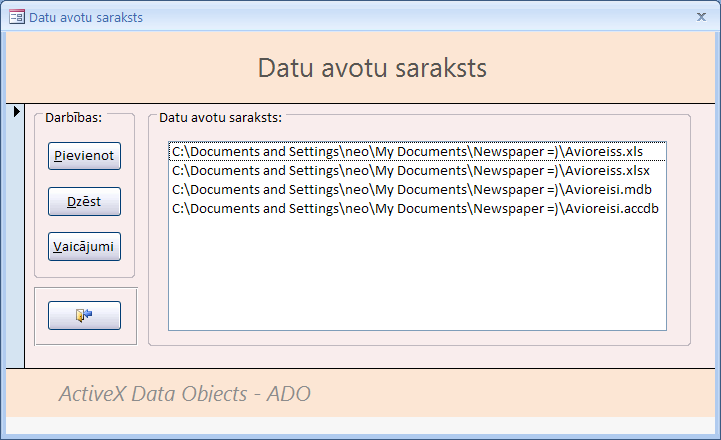
Modulī esošas procedūras lietojuma izpildīšanas laikā tiks izsauktas formas elementu notikumu procedūras. Šīs procedūras tiks iztirzātas nākamajā nodaļa.

# LIETOJUMA APRAKSTS

Praktiskā darba laikā tika izveidots lietojums, kurš nodrošina savienojuma izveidošanu ar jebkuru tipa datu avotu un spēj vienlaicīgi izveidot pat līdz desmit savienojumiem. Lietojums ļauj definēt vaicājumus datu avotam, kuru ir iespējams veidot pēc meta datiem ( tiek izvadītas datu avota visas tabulas un katrai tabulai attiecīgi tiek izvadīts lauku saraksts). Vaicājuma izpildes rezultāts tiek saglābts jauna tabulā, kurai automātiski tiek ģenerēts nosaukums pēc noklusēšanas. Atkarība no lietotāja izvēles, lietojums izvada vaicājuma izpildes rezultātu vai nu tabula, vai arī forma, kura izveidota uz izveidotas tabulas pamatā. Rezultātu formas ir pilnīgi neatkarīgas viena no otrās tādējādi ir iespējams vienlaikus strādāt ar vairākām rezultātu formām. Lietojuma ir nodrošināta vienkārša kļūdu novēršanas koncepcija, t.i., gadījumā ja programmas izpildes laikā ir notikusi kļūda, lietotājam par to tiek paziņots ar Windows dialoga paziņojumu.

## DATU AVOTU SARAKSTA FORMA

Pēc lietojuma palaišanas uzreiz tiek ielādēta *Datu avotu saraksta* forma (*MainForm*) (skat. att. 6), kuru veido vairākas kontroles: pogas, uzraksts, grupas un saraksts, ar to notikumu procedūrām un īpašībām. Formas galvenais uzdevums ir nodrošināt pilno darbības klāsta izvēlni, pateicoties kurai lietotājs var atvērt vairākas formas, kas paredzētas individuālu uzdevumu risināšanai.



1. att. Datu avotu saraksta forma.

Tātad lietotājs var atvērt *Jauna savienojuma* formu, lai veiktu jauna datu avota pievienošanu, šādu darbību nodrošina „Pievienot” poga, kuras notikuma procedūras izskatās sekojoši:

Private Sub btnNewCnn\_Click()

DoCmd.Hourglass HourglassOn

DoCmd.OpenForm "Data\_source\_addition", acNormal

DoCmd.Close acForm, "Mainform", acSaveNo

End Sub

Var pamanīt ka uzreiz pēc pogas nospiešanas, tiek aktivizēts smilšu pulkstenis peles parasta kursora vietā, tiek atvērta *Jauna savienojuma* forma (*Data\_source\_addition*) un aizvērta tekoša forma bez izmaiņu saglabāšanas.

Pēc jauna datu avota pievienošanas, jaunais datu avots un tas atrašanas ceļš tiek paradīti *Datu avotu* sarakstā. Šīs saraksts tiek atjaunots katru reizi, kad forma tiek ielādēta. Attiecīga notikumu procedūra izskatās sekojoši:

Private Sub Form\_Load()

DoCmd.Hourglass HourglassOn

Display\_All\_Data\_Sources()

End Sub

*Display\_All\_Data\_Sources* ir publiski pieejama procedūra, kura ir atrodas *MultipleConnection* modulī; dotas procedūras pirmteksts izskatās šādi:

Public Sub Display\_All\_Data\_Sources()

If Application.CurrentProject.AllForms("MainForm").IsLoaded Then

Dim lst As ListBox

Dim count As Integer

Set lst = Forms("MainForm").Controls("lstConnections")

'Izdesim visas eksistejosas vertiibas

Do While lst.ListCount > 0

lst.RemoveItem(0)

Loop

'Izvadisim visus avotus

For count = 1 To ConnectionQuantity

lst.AddItem(DataSources(count))

Next

lst.DefaultValue = ConnectionQuantity

End If

End Sub

Šajā procedūrā mēs vispirms izveidojam saraksta objektu *lst* un piešķīrām tām *Datu avotu saraksta* formas *lstConnections* saraksta kontroles vērtību, tāda veida mēs varēsim operēt ar šo sarakstu un atjaunot to. Tad attīrām sarakstu, izdzēšot visas eksistējošas vērtības un pievienojam klāt jaunas rindas ar metodes *AddItem* palīdzību. Tā kā datu avotu adreses glabājas masīvā *DataSource*, ir jāpievieno datu avotus tik reizes cik lietotājs ir izveidojis atvērto savienojumu (*ConnectionQuantity*).

Forma ļauj aizvērt kādu no sarakstā atvērtiem savienojumiem, šādu darbība tiek izpildīta lietotājam nospiežot „Dzēst” pogu.

Private Sub btnDeleteConnection\_Click()

Dim SelectedCnn As Integer

SelectedCnn = Me.lstConnections.ListIndex + 1

DeleteConnection(SelectedCnn)

End Sub

Procedūrā mainīgajām *SelectCnn* tiek piešķirta sarakstā izvēlēta savienojuma vērtība un tā tiek nodota moduļa *MultipleConnection* procedūrai *DeleteConnection*, kuras pirmteksts paradīts zemāk.

Public Sub DeleteConnection(SelectedCnn As Integer)

If ConnectionQuantity > 0 Then

Dim i As Integer

Connections(SelectedCnn).Close()

Set Connections(SelectedCnn) = Nothing

For i = SelectedCnn To ConnectionQuantity - 1

Connections(i) = Connections(i + 1)

DataSources(i) = DataSources(i + 1)

Next

Set Connections(ConnectionQuantity) = Nothing

ConnectionQuantity = ConnectionQuantity - 1

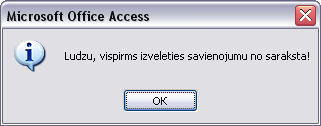
Display\_All\_Data\_Sources()

End If

End Sub

Tātad *DeleteConnections* procedūra aizver izvēlēto savienojumu ar Close metodes palīdzību no savienojuma masīva *Connections* ( šajā masīva glabājas *ConnectionString* vērtības) un pārkārto masīvu, iznicinot aizvērto savienojumu. Rezultāta savienojumu skaits paliek par vienu reizi mazāks (*ConnectionQuantity* - 1).

Trešā formas poga „Vaicājumi” atver SQL vaicājuma redaktora formu, kurā ir iespējams definēt vaicājumus kādam no *Datu avotu saraksta* izvēlētajām avotam. Tātad pirms nospiest šo pogu, lietotājam obligāti ir jāizvēlas avotu no sarakstā, citādi tiks izvadīts attiecīgs paziņojums (skat. att. 8).



1. att. Brīdinājuma paziņojums.

Pogas notikuma procedūra izskatās sekojoši:

Private Sub btnOpenSQLQueryForm\_Click()

Dim CnnNum As Integer

CnnNum = Me.lstConnections.ListIndex + 1

If CnnNum > 0 Then

DoCmd.OpenForm "Make\_Query\_Form", acNormal

MakeActiveCatalog(CnnNum)

DoCmd.Close acForm, "MainForm", acSaveNo

ShowTableList()

Else

MsgBox "Ludzu, vispirms izveleties savienojumu no saraksta!", \_ vbInformation

End If

End Sub

Citādi, ja lietotājs ir izvēlējies kādu no avotiem, formas atvēršanas brīdī notiek aktīva kataloga izveidošana – publiski pieejama procedūra *MakeActiveCatalog*, kurai tiek padots izvēlēta avota identifikators un tiek izsaukta *ShowTableList* procedūra, kura izvada visas avota tabulas.

Procedūras *MakeActiveCatalog* pirmteksts:

Public Sub MakeActiveCatalog(num As Integer)

If Not IsNull(ActiveCatalog) Then

Set ActiveCatalog = New ADOX.Catalog

Set Active\_Connection = Connections(num)

Set ActiveCatalog.ActiveConnection = Active\_Connection

End If

End Sub

Procedūra saņem avota identifikatoru, vadoties pēc kura no savienojumu masīva tiek atlasīts atbilstošs savienojums un tas tiek padarīts par aktīvu - tiek izveidots šī datu avota katalogs, kas satur avota tabulas, lietotājus, skatus u.tml. Tālāka darbība notiks tikai ar šo šī kataloga savienojumu.

Procedūras *ShowTableList* pirmteksts:

Public Sub ShowTableList()

If Application.CurrentProject.AllForms("Make\_Query\_Form").IsLoaded Then

Dim cmb As ComboBox

Dim tbl As ADOX.Table

Set cmb = Forms("Make\_Query\_Form").Controls("cmbTables")

'Izdesim visas eksistejosas vertibas

Do While cmb.ListCount > 0

cmb.RemoveItem(0)

Loop

'Izvadisim visus tabulu nosaukumus

For Each tbl In ActiveCatalog.Tables

If tbl.Type = "Table" Then

cmb.AddItem(tbl.Name)

End If

Next tbl

End If

End Sub

Uzreiz pēc *SQL vaicājuma redaktora* formas atvēršanas, izpildās *ShowTableList* procedūra, kura tikko atvērtas formas izvēlnes (*ComboBox*) kontrolei *cmbTables* piešķir izvēlēta datu avota visu tabulu nosaukumu vērtības.

Pēdēja neapskatīta formas darbība ir aizvērt lietojumu, beidzot darbu ar Access. Šādu darbību nodrošina 7. att. redzama poga ar attēlu, kas ir atdalīta ar taisnstūri no citām formas kontrolēm. Pogas notikuma *OnClick* procedūra izvada paziņojumu vai lietotājs tiešam vēlas iziet no Access un jā tas ir tā, tad Access tiks aizvērts.

Private Sub btnClose\_Click()

If MsgBox("Vai tiesam velaties iziet no Access?", vbYesNo) = 6 Then

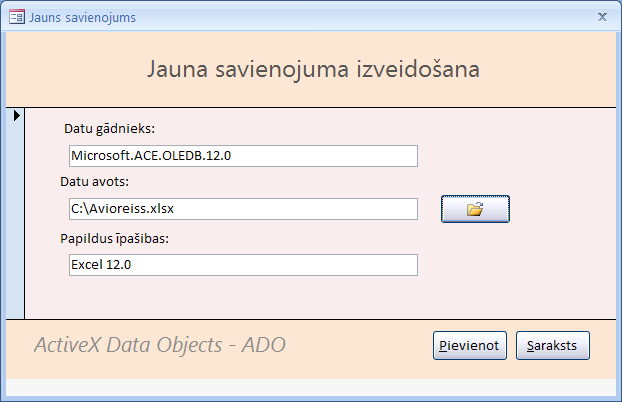
DoCmd.Quit()

End If

End Sub

## JAUNA SAVIENOJUMA FORMA

*Jauna savienojuma* *izveidošanas* forma (*Data\_source\_addition*) nodrošina jauna datu avotu pievienošanu, tā tiek atvērta pēc pogas „Pievienot” nospiešanas (skat. 8. lpp.). Forma ir paradīta 9. attēlā.



1. att. *Jauna savienojuma* forma.

Forma ir pieejami trīs teksta lauki:

1. Datu gādnieks (*Provider*) – šeit ir jānorāda datu gādnieka saīsināts nosaukums, kurš spēs izveidot savienojumu ar specifisku objektu, t.i., piemēram, Microsoft Jet datu bāzes - Microsoft.Jet.OLEDB.4.0 - datu gādnieks var izveidot savienojumu, ka ar Access datu bāzēm, tā arī ar Excel elektronisko darba tabulu lapām u.tml. Teksta lauka noklusēta vērtība ir Microsoft.Jet.OLEDB.4.0;
2. Datu avots (*Data Source*) – šeit ir jānorāda datu avots un tā atrašanas vieta. Pretim teksta laukam ir paredzēta poga, kas ļauj lietotājam, izmantojot Windows pārlūku noradīt datu avotu;
3. Papildus īpašības (*Extended Properties*) - šeit ir jānorāda specifiska īpašība, kas ļautu datu gādniekam precizēt datu avotu. Īpašība tiek automātiski piešķirta laukam, kad lietotājs, izmantojot pārlūku, norāda datu avotu.

Forma satur trīs komandpogas, no kurām poga „Saraksts” ļauj lietotājam aizver tekošo formu un atgriezties uz *Datu avotu saraksta* formu. Pogas nospiešanas notikuma procedūras pirmteksts izskatās šādi:

Private Sub btnExitAddition\_Click()

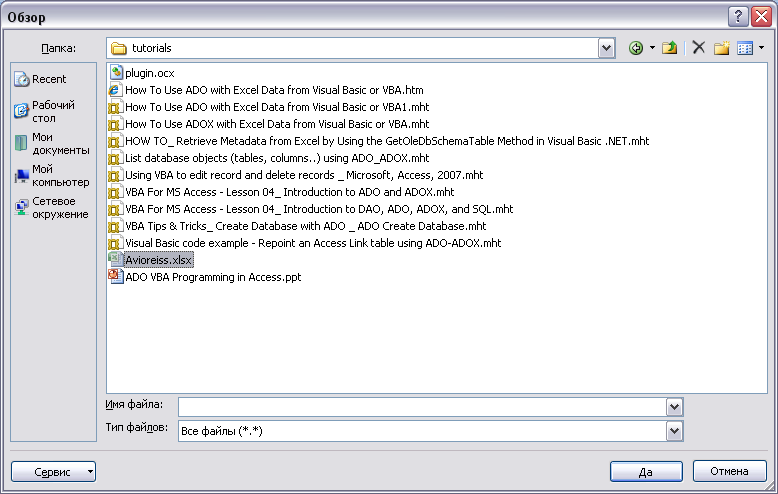
DoCmd.Hourglass HourglassOn

DoCmd.Close acForm, "Data\_Source\_addition", acSaveNo

DoCmd.OpenForm "MainForm", acNormal

End Sub

Pretī *Datu avots* laukam ir novietota poga ar attēlu (), kas paredz Windows pārlūka atvēršanu, kurā lietotājs var ērti sameklēt un atvērt datu avotu (skat. att. 10).



1. att. *Windows* pārlūks.

Pēc datu avota atvēršanas, datu avota atrašanas ceļš un nosaukums tiek automātiski piešķirts *Datu avots* lauka vērtībai un jā ir atvērts īpašs datu avots, tad *Papildus īpašības* laukam tiek piešķirta tā raksturojuša īpašība.

Pogas notikuma nospiešanas procedūra izskatās sekojoši:

Private Sub btnBrowser\_Click()

Dim browser As FileDialog

Dim Item As Object

Set browser = Application.FileDialog(msoFileDialogFilePicker)

With browser

If .Show = -1 Then

For Each Item In .SelectedItems

Me.txtDataSource.Value = Item

'MS Excel 2000 - 2003 gadijuma

If Right(Item, 4) = ".xls" Then

Me.txtDataProvider = "Microsoft.Jet.OLEDB.4.0"

Me.txtProperties.Value = "Excel 8.0"

Else

'MS Excel 2007 gadijuma

If Right(Item, 5) = ".xlsx" Then

Me.txtProperties.Value = "Excel 12.0"

Me.txtDataProvider.Value = "Microsoft.ACE.OLEDB.12.0"

Else

'MS Access 2007 gadijuma

If Right(Item, 6) = ".accdb" Then

Me.txtDataProvider.Value = \_

"Microsoft.ACE.OLEDB.12.0"

Me.txtProperties.Value = ""

Else

Me.txtDataProvider = "Microsoft.Jet.OLEDB.4.0"

Me.txtProperties = ""

End If

End If

End If

Next Item

End If

End With

Set browser = Nothing

End Sub

Tātad procedūras pamata tiek veidots pārlūka objekta *browser*, kuram turpmāk noradām īpašību *msoFileDialogFilePicker*, tāda veida ļaujam lietotājam izvēlēties vienu vai vairākus failus. Papildus tiek izveidots objekts *Item*, kuram piešķiram katra lietotāja izvēlēto faila (objekta) vērtību un vadoties pēc izvēlēta faila, t.i., nosakot tā paplašinājumu, piešķiram attiecīgo datu gādnieku un īpašības ( jā tādas ir nepieciešamas).

Pēc pogas „Pievienot” piespiešanas tiek izsaukta savienojuma izveidošanas procedūra (*AddNewConnection*), kurai tiek padotas formas teksta lauku vērtības, balstoties uz kurām tiek inicializēts un atvērts savienojums.

Private Sub btnAdd\_Click()

AddNewConnection Me.txtDataProvider.Value, Me.txtDataSource.Value & " ",\_

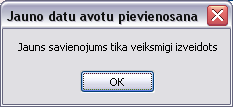
Me.txtProperties.Value & " "

DoCmd.OpenForm "MainForm", acNormal

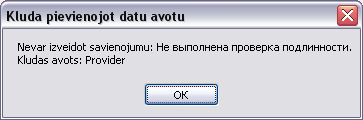
DoCmd.Close acForm, "Data\_source\_addition", acSaveNo

End Sub

Veiksmīgas savienošanas gadījuma, par to tiek paziņots lietotājam (skat. 11. att.) un avots tiek pievienots datu avotu sarakstām (skat. 7. att.), citādi ir paziņots par notikušo kļūdu (skat. 12. att.).



1. att. Paziņojums par veiksmīgu savienojuma izveidi.



1. att. Kļūdas paziņojums.

Modelī *MultipleConnection* pieejamas procedūras *AddNewConnection* pirmteksts:

Public Sub AddNewConnection(Data\_Provider As String, Data\_Source As String, \_ Properties As String)

If ConnectionQuantity < MaxConnections Then

'Ja ir vietas masiva izveidosim jaunu piesliegumu

'Kludas gadijuma izmantosim Error Handler, lai izvadit pazinojumu \_

lietotajam

On Error GoTo Connection\_Error\_Handler

Dim NewConnection As Connection

Dim Connection\_String As String

NewConnection = New ADODB.Connection

Connection\_String = "Provider=" & Data\_Provider & \_

";Data Source=" & Data\_Source & \_

";Extended Properties=" & Properties

NewConnection.ConnectionString = Connection\_String

NewConnection.Mode = adModeReadWrite

NewConnection.Open() 'Atveram jaunu savienojumu

'Pievienojam tikko izveidoto savienojumu masivaam

ConnectionQuantity = ConnectionQuantity + 1

Connections(ConnectionQuantity) = NewConnection

DataSources(ConnectionQuantity) = Data\_Source

MsgBox("Jauns savienojums tika veiksmigi izveidots", , "Jauno" & \_

"datu avotu pievienosana")

Exit\_Connection\_Error\_Handler:

NewConnection = Nothing

End If

Exit Sub

Connection\_Error\_Handler:

MsgBox("Nevar izveidot savienojumu: " & Err.Description & Chr(13) & \_

Chr(10) & "Kludas avots: " & Err.Source, , "Kluda")

Resume Exit\_Connection\_Error\_Handler

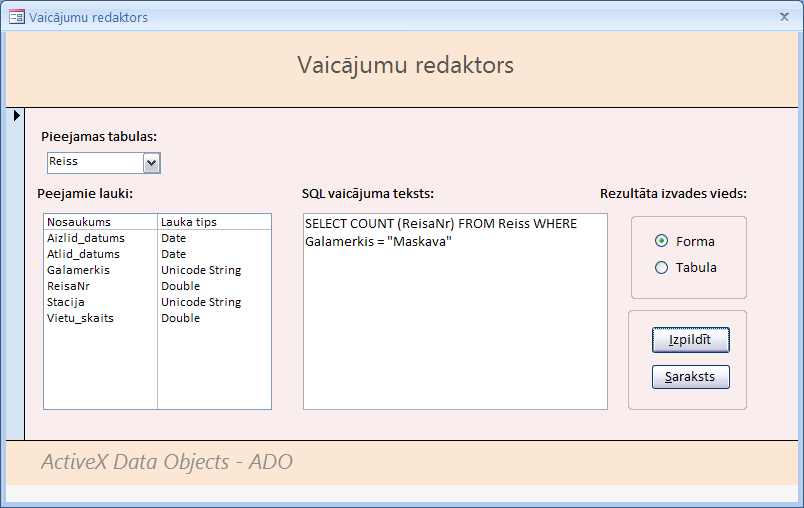
End Sub

Vispirms procedūra pārbauda vai patreizējo savienojumu skaits nepārsniedz maksimāli iespējamo atvērto savienojumu skaitu. Tad izveido jauno ADO savienojuma objektu *NewConnection* un *Connection\_String* mainīgo, kam piešķiram virkni ar datiem, kuri nepieciešami specifiska savienojumu izveidošanai (šie dati ir iegūti lietotājam aizpildot formas laukus) un kurš dublē savienojuma īpašību *ConnectionString* ( tas izveidots vienīgi pirmteksta optimizācijai). Tad *ConnectionString* īpašībai piešķiram *Connection\_String* vērtību, tāda veida jaunais savienojums iegūst savu vērtību un nozīmi. Tagad procedūrai ir jāatver šo specifisku savienojumu un tās tiek izdarīts ar metodes *Open* palīdzību. Pirms atvēršanas ar īpašības *Mode* palīdzību datu gādniekam tiek noradīts, ka savienojums ir jātver rakstīšanas un lasīšanas režīma.

Pēc savienojuma izveidošanas vienīgi atliek pievienot jaunu savienojumu *Connections* savienojumu masīvam, atvērto savienojumu skaitu - *ConnectionQuantity* palielināt par vienu vērtību un datu avotu ar tā atrašanas adresi pievienot datu avotu adrešu masīvam *DataSources*, lai turpmāk to attēlotu *Datu avotu* sarakstā.

## SQL vaicājuma redaktora forma

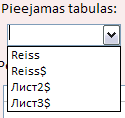
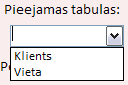
*SQL vaicājuma redaktora* (*Make\_Query\_Form*) galvenais uzdevums ir ļaut lietotājam definēt datu atlasēs kritērijus kādam izvēlētājam datu avotam (veidot SQL vaicājumus) un izvadīt vaicājuma rezultātus vai nu tabulas, vai arī formas veidā (skat. 13. att.).

****

1. att. *SQL vaicājuma redaktora* forma.

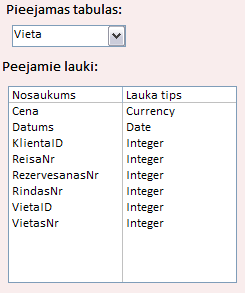
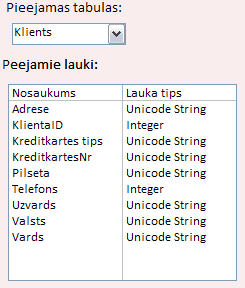
Forma tiek ielādēta uzreiz pēc *Datu avotu saraksta* formas pogas „Vaicājumi” nospiešanas. Pirms pogas nospiešanas, obligāti ir jāizvēlas kāds datu avots no tekošas formas sarakstā (skat. 7. att. un 10. lpp.).

*SQL vaicājuma redaktora* ielādes gaitā notiek aktīva kataloga izveidošana ar publiski pieejamas procedūras *MakeActiveCatalog* līdzdalību un tiek izsaukta *ShowTableList* procedūra, kura *SQL vaicājumu redaktora* formas izvēlnes kontrolei *cmbTables* piešķir izvēlēta datu avota visu tabulu nosaukumu vērtības (skat. 11. lpp.).



1. att. Datu avotu Avioresi.mdb (kreisa pusē) un Avioreisi.xls (laba pusē) tabulu saraksts.

Zem minētas izvēlnes ir izvietots saraksts, kurš, pēc tabulas izvēlēs izvēlnē, parāda šīs tabulas lauku nosaukumu sarakstu un šo lauku datu tipus (skat. att. 13. un 15.).



1. att. Tabulu pieejamie lauki.

Šādu saraksta vērtību atjaunošanu pēc katras jaunas tabulas izvēles nodrošina izvēlnes notikumu procedūra *AfterUpdate*, kura izsauc procedūru *ShowFieldList* un nodod tai izvēlnē izvēlēto vērtību.

Private Sub cmbTables\_AfterUpdate()

ShowFieldList(Me.CmbTables.Value)

End Sub

Procedūras *ShowFieldList* pirmteksts:

Public Sub ShowFieldList(TableName As String)

If Application.CurrentProject.AllForms("Make\_Query\_Form").IsLoaded Then

Dim lst As ListBox

Dim cln As ADOX.Column

Dim strItem As String

Set lst = Forms("Make\_Query\_Form").Controls("lstFields")

'Izdesim visas eksistejosas vertiibas

Do While lst.ListCount > 0

lst.RemoveItem(0)

Loop

'Pievienosim virsrakstu

lst.AddItem("Nosaukums; Lauka tips")

'Izvadisim visus laukus

For Each cln In ActiveCatalog.Tables(TableName).Columns

Select Case cln.Type

Case adBigInt : strItem = cln.Name & "; " & "Big Integer"

Case adBinary : strItem = cln.Name & "; " & "Binary"

Case adBoolean : strItem = cln.Name & "; " & "Boolean"

Case adBSTR : strItem = cln.Name & "; " & "BSTR"

Case adChapter : strItem = cln.Name & "; " & "Chapter"

Case adChar : strItem = cln.Name & "; " & "Char"

Case adCurrency : strItem = cln.Name & "; " & "Currency"

Case adDate : strItem = cln.Name & "; " & "Date"

Case adDBDate : strItem = cln.Name & "; " & "DBDate"

Case adDBTime : strItem = cln.Name & "; " & "DBTime"

Case adDBTimeStamp : strItem = cln.Name & "; " & "DBTiemTamp"

Case adDecimal : strItem = cln.Name & "; " & "Decimal"

Case adDouble : strItem = cln.Name & "; " & "Double"

Case adEmpty : strItem = cln.Name & "; " & "Empty"

Case adError : strItem = cln.Name & "; " & "Error"

Case adFileTime : strItem = cln.Name & "; " & "FileTime"

Case adGUID : strItem = cln.Name & "; " & "GUID"

Case adIDispatch : strItem = cln.Name & "; " & "IDispatch"

Case adInteger : strItem = cln.Name & "; " & "Integer"

Case adIUnknown : strItem = cln.Name & "; " & "Unknown"

Case adLongVarBinary : strItem = cln.Name & "; " & "Long & \_

" Binary"

Case adLongVarChar : strItem = cln.Name & "; " & "Long & \_

" string"

Case adLongVarWChar : strItem = cln.Name & "; " & "Long" & \_ " Unocode string"

Case adNumeric : strItem = cln.Name & "; " & "Numeric"

Case adPropVariant : strItem = cln.Name & "; " & \_ "Automation PropVariant"

Case adSingle : strItem = cln.Name & "; " & "Single"

Case adSmallInt : strItem = cln.Name & "; " & "Small Integer"

Case adTinyInt : strItem = cln.Name & "; " & "One-Byte" & \_

" Integer"

Case adUnsignedBigInt : strItem = cln.Name & "; " & \_ "Unsigned Big Integer"

Case adUnsignedInt : strItem = cln.Name & "; " & "Unsigned" & \_ " Integer"

Case adUnsignedSmallInt : strItem = cln.Name & "; " & \_ "Unsigned Small Integer"

Case adUnsignedTinyInt : strItem = cln.Name & "; " & \_ "Unsigned One-Byte Integer"

Case adUserDefined : strItem = cln.Name & "; " & "User" & \_ " Defined"

Case adVarBinary : strItem = cln.Name & "; " & "Binary"

Case adVarChar : strItem = cln.Name & "; " & "String"

Case adVariant : strItem = cln.Name & "; " & "Variant"

Case adVarNumeric : strItem = cln.Name & "; " & "Numeric"

Case adVarWChar : strItem = cln.Name & "; " & "Unicode" & " String"

Case adWChar : strItem = cln.Name & "; " & "Unicode String"

End Select

lst.AddItem(strItem)

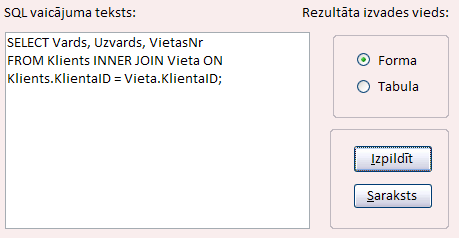
Next cln

End If

End Sub

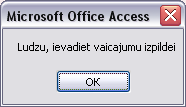
Procedūras izpildes laikā vispirms tiek izveidots saraksta objekts *lst,* kurām turpmāk tiek piešķirta SQL vaicājuma redaktora formas *lstFields* saraksta vērtība (šīs saraksts atrodas zem apskatītas izvēlnes), šāda veida procedūra var veiksmīgi operēt ar esošo sarakstu. Bez dota objekta, tiek izveidots arī *ADOX.Column cln* objekts, kurām turpmāk tiek piešķirta katras izvēlētas tabulas kolonnu jeb atribūtu vērtības. Kad objekts iegūst tekošas tabulas kolonnas vērtību, procedūra nosaka atribūta datu tipu un atkarība no tā izvēlas piemērotu operāciju no *Case* sarakstā. Šādas operācijas pamatā teksta mainīgam *strItem* tiek piešķirtas tekošas tabulas kolonnas nosaukums un datu tips, tad dota mainīga vērtība tiek piešķirta esošam sarakstām. Šāds darbību klāsts notiek tiktāl, kamēr netiks apskatītas visas tabulas kolonnas.

Vaicājuma ievadīšana notiek speciāli tām atvelētājam teksta laukā ar nosaukumu *SQL vaicājuma teksts*, pēc vaicājuma ievadīšanas ir jāizvēlas rezultāta izvades veids – tabula vai forma, kas izdarāms pārslēgu grupā ar attiecīgu nosaukumu un jānospiež poga „Izpildīt” (skat. att. 16).

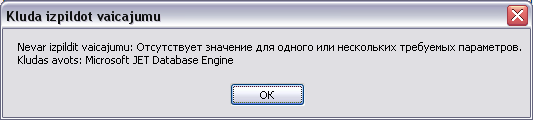


1. att. Vaicājuma izpildes elementi.

Gadījumā jā lietotājs neievadīs vaicājuma tekstu un nospiedis pogu „Izpildīt”, vai arī vaicājuma teksts būs nekorekts, par notikušu kļūdu tiks paziņots (skat. att. 17 un 18).

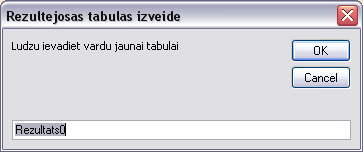
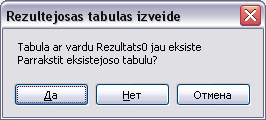


1. att. Kļūdas paziņojums.



1. att. Kļūdas paziņojums.

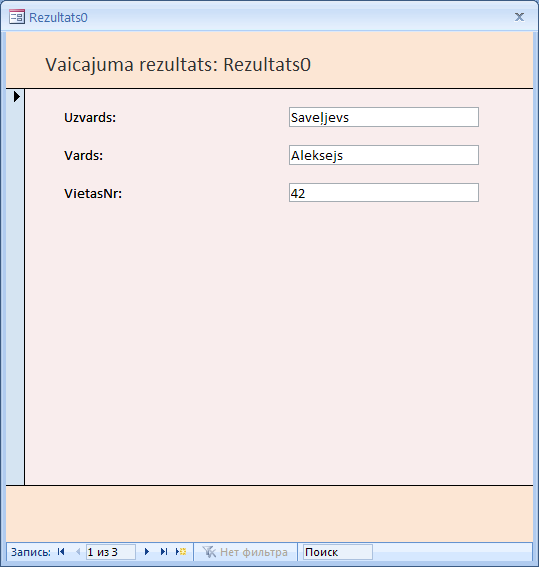
Vaicājumu izpildes rezultāti tiek saglabāti tabulā un pēc lietotāja piekrišanas arī formā. Nosaukums jaunai tabulai un formai tiek ģenerēts automātiski, tomēr lietotājam tiek arī piedāvāta iespēja ievadīt savu nosaukumu rezultātu tabulai (skat. att. 19). Jā ievadītais nosaukums jau eksistē, par to tiek paziņots un lietotājam ir vai nu jāpārraksta tabulu vai jāievada nosaukumu no jaunā vai arī jāatceļ darbību, nospiežot pogu *Cancel*. (skat. att. 20). Forma tiek veidota uz izveidotas tabulas pamatā un lietotājs var izvēlēties vai saglabāt jaunizveidotu formu, vai nē.



1. att. Nosaukuma piešķiršana tabulai. 20. att. Paziņojums par jau esošo tabulas nosaukumu.



1. att. Pēc vaicājuma izpildes izveidota rezultējoša tabula.



1. att. Rezultāts formas veidā.

Tagad apskatīsim funkcijas un procedūras, pateicoties kurām šāds darbību klāsts tiek izpildīts. Pēc pogas „Izpildīt” nospiešanas tiek veikta pārbaude un veiksmīgas pārbaudes gadījumā tiek izsauktā modelī *MultipleConnections* publiski pieejama procedūra *ExecuteCommand*, kurai tiek padota teksta laukā ievadīta vaicājuma vērtība.

Private Sub btnRunQuery\_Click()

If IsNull(Me.txtQuery.Value) Then

MsgBox("Ludzu, ievadiet vaicajumu izpildei")

Exit Sub

End If

ExecuteCommand(Me.txtQuery.Value)

End Sub

Procedūras *ExecuteCommand* pirmteksts:

Public Sub ExecuteCommand(strQuery As String)

On Error GoTo Error\_Handle

Dim NewRecordset, ResultRecordset As ADODB.Recordset

Dim NewCommand As ADODB.command

Dim CurrentCatalog As ADOX.Catalog

Dim NewTable As ADOX.Table

Dim fld, fld1 As ADODB.Field

Dim cln As ADOX.Column

Dim CurrentConnection As ADODB.Connection

Dim NewForm As Access.Form

Dim lbl As Access.Label

Dim res\_tbl\_num, num, Response, left, left\_t, top, width,\_

height As Integer

Dim res\_tbl\_name, def\_tbl\_name As String

Dim already\_exists As Boolean

Pēc visu lokālo mainīgo definēšanas procedūrā notiek pieslēgšana pie tekošas datu bāzes. Uz tekošo datu bāzi noradīs *ADOX.Catalog* objekts *CurrentCatalog*.

'Izveidosim katalogu kas noradis uz tekoso projektu

Set CurrentConnection = Application.CurrentProject.Connection

Set CurrentCatalog = New ADOX.Catalog

Set CurrentCatalog.ActiveConnection = CurrentConnection

Tālāk izveidosim *Recordset* objektu, kas laicīgi glabās vaicājuma izpildes rezultātu. Lai to panāktu, vispirms izveidosim *ADODB.Command* objektu *NewCommand*, kurām piešķiram lietotāja ievadīta vaicājuma vērtību un izpildām to ar metodes *Execute* palīdzību. Vaicājuma izpildes rezultātu (rakstu kopu) piešķirsim izveidotājam *NewRecordset*  rakstu kopas objektam. Vaicājuma izpildei obligāti ir jānorāda aktīvais savienojums (savienojums, kuru lietotājs ir izvēlējies no sarakstā un kurām tika izveidots katalogs).

'Vaicajuma izpilde

Set NewCommand = New ADODB.command

NewCommand.CommandText = strQuery

NewCommand.ActiveConnection = Active\_Connection

Set NewRecordset = NewCommand.Execute(Options:=adCmdText)

Nākamais solis ir jaunas tabulas nosaukuma ģenerēšana. Tā kā programma nedzēš iepriekšēja vaicājumā iegūtu tabulu, mums ir jāpanāk, lai procedūra atrod iepriekšējo tabulas nosaukumu un vadoties pēc tā piedāvā jauno nosaukumu, t.i., jā tiks atrasta tabula ar noklusēto nosaukumu RezultatsN, kur N ir rezultāta numurs, tad procedūrai ir jāpiedāvā RezultatsN + 1 nosaukums.

'Meklesim rezultasu tabulai vardu pec noklusesanas:

'rezultatsN, kur N rezultaрu tabulas numurs

res\_tbl\_num = 0

def\_tbl\_name = "Rezultats" & res\_tbl\_num

Do

already\_exists = False

For Each NewTable In CurrentCatalog.Tables

If NewTable.Name = def\_tbl\_name Then

already\_exists = True

res\_tbl\_num = res\_tbl\_num + 1

def\_tbl\_name = "Rezultats" & res\_tbl\_num

Exit For

End If

Next NewTable

Loop Until Not already\_exists

Neskatoties uz to, ka nosaukums tabulai tiek atrasts automātiski, tomēr piedāvāsim lietotājam ievadīt savu nosaukumu. Tātad vispirms izvadām logu ar paziņojumu ievadīt tabulas nosaukumu, tad pārbaudām vai tāds nosaukums jau ir definēts kādai datu bāzes esošai tabulai. Jā tāds nosaukums tika jau izmantots, tad izvadām paziņojumu ar izvēles darbībām – parakstīt atrasto tabulu, vai arī ne, vai nu visu atcelt. Lietotāja izvēles darbību piešķiram mainīgajām *Response*.

'Ludzam lietotaju ievadit tabulas nosaukumu

'Pec noklusesanas tiks piedavats nosaukums def\_tbl\_name

Do

already\_exists = False

res\_tbl\_name = InputBox("Ludzu ievadiet vardu jaunai tabulai", \_

"Rezultejosas tabulas izveide", def\_tbl\_name)

If res\_tbl\_name = "" Then GoTo Resume\_Error\_Handle

For Each NewTable In CurrentCatalog.Tables

'Parbaudam, vai tabulas ar tadu nosaukumu nav datu baze

If NewTable.Name = res\_tbl\_name Then

Response = MsgBox("Tabula ar vardu " & res\_tbl\_name & \_

" jau eksiste" & Chr(13) & Chr(10) & \_

"Parrakstit eksistejoso tabulu? ", vbYesNoCancel, \_ "Rezultejosas tabulas izveide")

Select Case Response

Case 6

'iznicinam tabulas ar nosaukumu res\_tbl\_name

CurrentCatalog.Tables.Delete(res\_tbl\_name)

already\_exists = False

Case 7 : already\_exists = True

Case 2 : GoTo Resume\_Error\_Handle

End Select

Exit For

End If

Next NewTable

'turpinam lidz netiks ievadits unikals nosaukums

Loop Until Not already\_exists

Tagad ir iespējams izveidot jaunu tabulu. Definēsim jaunas tabulas laukus, nosaukumu, lauku īpašības un atribūtus. Beigas pievienosim jaunizveidotu tabulu tekošai datu bāzei.

'Jaunas tabulas izveidoрana

Set NewTable = New ADOX.Table

NewTable.Name = res\_tbl\_name

'Jaunas tabulas struktuuras defineesana

For Each fld In NewRecordset.Fields

NewTable.Columns.Append(fld.Name)

Next fld

'Iestadisim NewTable tabulas katrai kolonnai atributu Required = No

For Each cln In NewTable.Columns

cln.Attributes = adColNullable

Debug.Print(cln.Name)

Next cln

'Tabulas pievienosana catalogam

CurrentCatalog.Tables.Append(NewTable)

Kad jauna tabula ir izveidota, ierakstīsim vaicājumā izpildes rezultātā iegūtu rakstu kopu jaunizveidotajā tabulā. Lai to izdarītu ir nepieciešams izveidot jaunu *ADODB.Recordset* rakstu kopas objektu *ResultRecordset*, kas noradīs uz tikko izveidoto tabulu. Šajā nolūkā izmantosim metodi *Open*, kas *ResultRecordset* objektam piešķirs tekošas datu bāzes mūsu izveidotas tabulas vērtību (citiem vārdiem sakot, piešķirsim tām šīs tabulas rakstu kopu). Opcija *adOpenKeyset* norāda tekošas rindas – kursora režīmu, t.i., paredz rakstu atjaunošanu, dzēšanu un pievienošanu, savukārt, opcija *adLockOptimistic* nosaka datu slēgšanas režīmu, kas varētu būt lietderīga tāda gadījuma, piemēram, kad divi lietotāji vienlaikus izmanto vienus un tos pašus datus. Mūsu gadījumā raksti tiks slēgti tikai tad, kad tie tiks atjaunoti. Izpildām veiktas izmaiņas

'Ierakstam vaicajuma rezultatus NewTable tabula

ResultRecordset = New ADODB.Recordset

ResultRecordset.Open res\_tbl\_name, CurrentConnection, adOpenKeyset, \_

adLockOptimistic

'Vaicajuma rezultatu no rakstu kopas NewRecordset

'ierakstisana tabula RezultatiN (rakstu kopa ResultRecordset)

'Turpinām kāmēr nav sasniegts rakstu kopas beigas

Do Until NewRecordset.EOF

'Izmantojot AddNew metodi,pārlecam uz jaunu rakstu

ResultRecordset.AddNew()

'Apzīmēsim ar num rezultātu tabulas lauka numuru

num = 0

'Katru vaicājuma rakstu kopas tekoša lauka vērtību

For Each fld In NewRecordset.Fields

'Piešķiram mūsu izveidotas tabulas tekošam laukam

ResultRecordset.Fields.Item(num).Value = fld.Value

num = num + 1

Next fld

'Izpildām veiktas izmaiņas

ResultRecordset.Update()

'Parējam uz nākamo rakstu

NewRecordset.MoveNext()

Loop

Pēdējais kas mums ir palicis ir izvadīt rezultātu formas vai tabulās veidā. Gadījumā, ja ir nepieciešams izvadīt rezultātu formas veidā izveidosim jaunu formu ar metodes *CreateForm* palīdzību un izmantojot jau iepriekš izveidotu formu *Result* kā šablonu. Tad jaunizveidotajai formai par datu avotu noradām mūsu izveidotu tabulu un izveidojām tai rakstu, papildus pie formas īpašībām noradām kā forma ir peldoša un modāla. Citādi atvērām tikai tabulu.

If Application.Forms("Make\_Query\_Form").Controls("cbResult").Value = 1 Then

'Formas izveidosana

Set NewForm = Application.CreateForm(, "Result")

NewForm.RecordSource = res\_tbl\_name

NewForm.Caption = res\_tbl\_name

NewForm.Modal = True

NewForm.PopUp = True

'Izmantojot Cm\_To\_Twips funkciju, lai iegūtu uzskatamākas mērvienības,

'noradīsim formas platumu, garumu u.tml.

NewForm.width = Cm\_To\_Twips(13)

num = 1

width = Cm\_To\_Twips(5)

height = Cm\_To\_Twips(0.5)

left = Cm\_To\_Twips(1)

left\_t = Cm\_To\_Twips(7)

'Izveidosim nepieciešamus teksta laukus, lai izvadītu tabulas laukus

For Each cln In NewTable.Columns

top = Cm\_To\_Twips((0.5 \* num) + (0.5 \* (num - 1)))

'Uzraksts saturēs lauka nosaukumu

Application.CreateControl NewForm.Name, acLabel, acDetail, , \_ (cln.Name & ": "), left, top, width, height

'Teksta lauks noradīs uz tabulas kolonnu cln.Name

Application.CreateControl NewForm.Name, acTextBox, acDetail, , \_ cln.Name, left\_t, top, width, height

num = num + 1

Next cln

width = Cm\_To\_Twips(3)

height = Cm\_To\_Twips(0.83)

top = Cm\_To\_Twips(0.5)

left = Cm\_To\_Twips(0.5)

'Virsraksta izveidošana

Set lbl = Application.CreateControl (NewForm.Name, acLabel, \_

acHeader, ,("Vaicajuma rezultats: " & res\_tbl\_name), left, top)

lbl.FontSize = 16

lbl.SizeToFit()

lbl.ForeColor = RGB(51, 51, 51)

'Formas atvēršana

DoCmd.DoMenuItem(0, 2, 1)

DoCmd.Restore()

Else

'Tabulas atveršana Datasheet View režīmā

DoCmd.OpenTable res\_tbl\_name, acViewNormal, acReadOnly

End If

Procedūras beigu izpildē tiek atbrīvoti visi liekie mainīgie no datora atmiņas.

Resume\_Error\_Handle:

If Not IsNull(NewRecordset) Then

Set NewRecordset = Nothing

End If

If Not IsNull(ResultRecordset) Then

Set ResultRecordset = Nothing

End If

If Not IsNull(CurrentCatalog) Then

Set CurrentCatalog = Nothing

End If

If Not IsNull(NewTable) Then

Set NewTable = Nothing

End If

If Not IsNull(fld) Then

Set fld = Nothing

End If

If Not IsNull(cln) Then

Set cln = Nothing

End If

If Not IsNull(CurrentConnection) Then

Set CurrentConnection = Nothing

End If

If Not IsNull(NewCommand) Then

Set NewCommand = Nothing

End If

If Not IsNull(lbl) Then

Set lbl = Nothing

End If

Exit Sub

Kļūdas gadījumā tiek izvadīts atbilstošs kļūdas apraksts un kļūdas avots.

Error\_Handle:

MsgBox("Nevar izpildit vaicajumu: " & Err.Description & Chr(13) & \_

Chr(10) & "Kludas avots: " & Err.Source, , "Kluda izpildot vaicajumu")

Resume Resume\_Error\_Handle

End Sub

Pēdēja neapskatīta formas poga „Saraksts” ļauj lietotājam atgriezties uz *Datu avotu saraksta* formu. Pēc dotas pogas nospiešanas, bez tekošas formas aizvēršanas un iepriekš minētas formas atvēršanas, tiek izsauktā *DeleteActiveCatalog* procedūra.

Private Sub btnDataSourceList\_Click()

DoCmd.Hourglass HourglassOn

DoCmd.OpenForm "MainForm", acNormal

DoCmd.Close acForm, "Make\_Query\_Form", acSaveNo

DeleteActiveCatalog()

End Sub

Procedūras *DeleteActiveCatalog* pirmteksts:

'Izdzes aktivo katalogu

Public Sub DeleteActiveCatalog()

If Not IsNull(ActiveCatalog) Then

Set ActiveCatalog = Nothing

End If

End Sub

Procedūra nodrošina aktīva kataloga, ka arī aktīva savienojuma objektu dzēšanu no datora atmiņas pēc darba pabeigšanas ar apskatīto formu. Taču pats savienojums paliks atvērts, jo uz to noradīs objekts no *Connections* masīva. Lai aizvērtu savienojumu, *Datu avotu saraksta* formā ir jāizvēlas nepieciešamo avotu un jānospiež poga „Dzēst”.

# SECINĀJUMI

Praktiskā darbā galvenais mērķis bija iepazīties un iegūt praktiskās zināšanas par ADO modeļu objektiem, to hierarhiju, objektu īpašībām un metodēm, izveidojot VBA (Visual Basic for Applications) lietojumu. Pildot praktisku darbu tika izstrādāts lietojums, kas dot iespēju pievienot vairākus datu avotus un izpildīt vaicājumus šiem avotiem. Par darba gandarījumu var atzīmēt jauniegūtas zināšanas, kuras paplašināja manas iemaņas lietojumu izstrāde un ļāva plašāk iepazīties ar datu bāzes datu piekļuves objektiem un to metodēm un īpašībām.

Darba gaitā pārliecinājos, ka VBA objektorientētais interfeiss ir labi piemērots Access, ka arī citu Microsoft programmatūras lietojumu izstrāde. Šāds priekšstats galvenokārt ir skaidrojams ar to, ka Visual Basic (VbScript, VBA, Visual Basic.Net) ir Microsoft plaši atbalstīta programmēšanas valoda un tā pirmkārt ir orientēta šādu lietojumu ērtai, uzskatāmai un kvalitatīvai izstrādei.

Neskatoties uz to, ka ADO objektorientētais interfeiss bija izstrādāts pirmkārt, lai nodrošinātu ērtu saskarni starp OLE DB pielietojumiem un lietojumu, tas, manuprāt, nodrošina arī pietiekamu funkcionalitāti lielāku daļu uzdevumu risināšanai.

Darbā gaitā salīdzināju ADO tehnoloģiju ar jaunāko ADO.Net tehnoloģiju, ar kuru biju iepazinies jau iepriekš. Par galveno atšķiru var uzsvērt, ka ADO (MDAC) galvenokārt paredzēta lietojumu izstrādei, kuri spēj piekļūt gandrīz jebkurām datu avotam, savukārt, ADO.Net (.NET Framework) vēl papildus galvenais uzsvars tiek likts uz drošību, atmiņas pārvaldību un izņēmumu veidošanu.

Darbs izrādījās diezgan apjomīgs un noderīgs, par to liecina liels informācijas apjoms Internetā. Ļoti lietderīga informācija tika uzzināta tieši lietotāju atbalsta Internet lapas, jo pārsvara vairāki lietotāji ir saskārušies ar manis līdzīgam darba izstrādes gaitā rādītam grūtībām.

# IZMANTOTA LITERATŪRA

1. Viktorija Vinogradova. Lekciju konspekti studiju priekšmetā „Datu bāzes vadības sistēma” – Rīga, RTU, 2007.
2. Internets: <http://msdn2.microsoft.com/en-us/library/bb979619.aspx>
3. Internets: <http://support.microsoft.com/kb/257819>