POGLAVLJE 2

Razvijanje višekorisničkih aplikacija u Accessu

- Zaključavanje na nivou stranica i zapisa
- Poređenje između optimističkog i pesimističkog zaključavanja
- Obrada grešaka u višekorisničkom okruženju pomoću događaja Error obrasca
- Pravljenje petlje za ponovno pokušavanje zaključavanja
- Utvrđivanje ko je sve prijavljen u bazu podataka
- Upravljanje pridruženim tabelama

Razvijanje Accessovih aplikacija za višekorisničko okruženje zahteva dodatno planiranje i drugačiji način razmišljanja u poređenju sa jednokorisničkim aplikacijama, ali ga nije teško naučiti. U ovom poglavlju istražujemo mogućnosti Accessa za višekorisnički rad i najbolje načine u Accessu 2002 za planiranje i projektovanje višekorisničkih sistema za rad s bazama podataka.

NAPOMENA Ovom poglavlju pridružene su tri baze podataka. Ch02app.mdb je „aplikativna" baza podataka – ona sadrži objekte korisničkog interfejsa i njima pripadajući programski kôd, uključujući i kôd koji upravlja vezama sa objektima baze podataka Ch02dat.mdb. Ch02dat.mdb je baza podataka „za podatke" – ona sadrži isključivo tabele. U bazi podataka Ch02auto.mdb čuva se i poslednja upotrebljena vrednost tipa AutoNumber, koju generiše namenska rutina AutoNumber, a koju koristi obrazac frmMenu u bazi podataka ch02app.mdb.

Poređenje između servera za datoteke i sistema klijent/server

U višekorisničkom okruženju, podaci se u Accessovim aplikacijama mogu deliti na tri načina:

Korišćenjem servera za datoteke U ovoj konfiguraciji, na svakoj radnoj stanici radi po jedan primerak mašine Jet, koji šalje zahteve bazi podataka koja se nalazi na centralnom serveru.

Replikovanjem U ovoj konfiguraciji, replikovane kopije, ili replike, baze podataka distribuiraju se širom organizacije. Podaci se dele između replika sinhronizovanjem u redovnim vremenskim razmacima.

Korišćenjem sistema klijent/server U ovoj konfiguraciji, Access se koristi kao čeona komponenta koja sarađuje sa serverom sistema za upravljanje bazama podataka koji podržava SQL naredbe, kao što su SQL Server ili Oracle. Svim podacima upravlja server baze podataka.

U ovom poglavlju govorićemo o tome kako više korisnika može da deli iste podatke pristupanjem serveru za datoteke kojim upravlja mašina baze podataka Jet. Replikovanje je detaljnije objašnjeno u poglavlju 9, *Replikovanje*. Pristup podacima u sistemima klijent/server obrađen je u više poglavlja, počev od poglavlja 3, *Projektovanje klijent/server aplikacija*.

Razdvajanje baza podataka

Ako drugačije ne zadate, Access smešta sve objekte aplikacije u istu .MDB datoteku. To loše utiče na performanse u višekorisničkom okruženju zato što mašina Jet mora putem mreže da pošalje objekat korisniku kad god vaša aplikacija treba da upotrebi neki objekat (npr. obrazac). U produkcionom okruženju, u kome se ništa ne menja osim podataka, veći deo saobraćaja u mreži koji na ovaj način nastaje nepotreban je.

Nepotreban saobraćaj u mreži možete da eliminišete razdvajanjem baze podataka na dva dela: na „aplikacionu" bazu podataka i na bazu podataka „za podatke". Na server za datoteke instalirajte ovu drugu bazu podataka (koja sadrži samo

tabele), a aplikacionu bazu podataka (koja sadrži sve druge objekte) kopirajte na svaku radnu stanicu. Svakoj kopiji aplikacione baze podataka komandom File Get External Tables Link Tables pridružite tabele iz baze podataka „za podatke“.

Ovaj pristup pruža sledeće prednosti:

- Bolje performanse (naročito korisničkog interfejsa).
- U lokalnim aplikacionim bazama podataka, koje se nalaze na radnim stanicama, možete da pravite privremene tabele ne brinući da li ćete time možda izazvati sukobe usled dupliranja imena ili zaključavanja privremenih objekata.
- Razdvajanje baze podataka olakšava ažuriranje aplikacija jer su podaci odvojeni od aplikacije. Izmene u aplikacijama možete da unesete u svoju kopiju aplikacione baze podataka i da ih zatim prenesete u ostale kopije ne ometajući pri tome same podatke.

Glavni nedostatak ovog pristupa jeste to što putanje ka pridruženim tabelama Access upisuje u obliku fiksnih (apsolutnih) podataka. To znači da, ukoliko premestite bazu podataka „za podatke“, morate da obnovite veze s pridruženim tabelama.

SAVET

U Access je ugrađen dodatak Database Splitter, koji pojednostavljuje razdvajanje baze podataka na dve baze, jednu za podatke i jednu za aplikaciju.

Upravljanje pridruženim tabelama

Pošto Access čuva apsolutne putanje ka pridruženim tabelama, upotreba takvih tabela zahteva dodatno održavanje. Kada premestite bazu podataka „za podatke“, prekinute veze možete ponovo da uspostavite na jedan od sledećih načina:

- Izbrišite, pa ponovo uspostavite vezu od početka.
- Pomoću Accessove alatke Linked Table Manager popravite reference i osvežite veze (izaberite Tools Database Utilities Linked Table Manager).
- Napišite VBA kôd za upravljanje vezama.

SAVET

Ukoliko primenjujete univerzalnu konvenciju za imenovanje objekata (Universal Naming Convention, UNC), kada uspostavljate veze između baza podataka (na primer, \\ImeServera\PutanjaDoDeljenogDirektorijuma\Podaci.Mdb), nećete morati da ponovo uspostavljate veze kada aplikacionu bazu podataka premestite s jednog računara na drugi unutar svoje lokalne mreže.

Baza podataka koja je pridružena drugom poglavlju, ch02app.mdb, sadrži modul basLinkedTables, u kome se nalazi kôd za upravljanje pridruženim tabelama. Ulazna tačka u taj deo koda je funkcija adhVerifyLinks, koja je prikazana u listingu 2.1. (Ova funkcija se pri pokretanju baze podataka poziva iz funkcije AutoExec, koja se, pak, poziva iz makroa AutoExec baze podataka.)

NAPOMENA

Da biste funkciju adhVerifyLinks koristili u svojim aplikacijama, treba da uvezete tri modula: basLinkedTables, basFileOpen i CommonDialog.

Listing 2.1

```
Function adhVerifyLinks(strDataDatabase As String, _
 strSampleTable As String) As Boolean

    ' Proverava stanje veza sa pridruženim tabelama.
    ' Ako otkrije prekinutu vezu, prvo pretražuje tekući direktorijum.
    ' Ako u njemu ne nađe traženu bazu podataka, korisniku prikazuje
    ' okvir za dijalog za otvaranje datoteka.
    ' Polazna pretpostavka: sve veze vode ka istoj ciljnoj .MDB datoteci.

    On Error GoTo adhVerifyLinksErr

    Dim varReturn As Variant
    Dim strDBDir As String
    Dim strMsg As String
    Dim varFileName As Variant
    Dim intI As Integer
    Dim intNumTables As Integer
    Dim strProcName As String
    Dim strFilter As String
    Dim lngFlags As Long
#If USEDAO Then
    Dim db As DAO.Database
    Dim tdf As DAO.TableDef
#Else
    Dim cnn As ADODB.Connection
    Dim cat As ADOX.Catalog
    Dim tbl As ADOX.Table
#End If

    strProcName = "adhVerifyLinks"

   ' Ispitujemo stanje veze sa  tabelom čije je ime zadato
   ' u parametru strSampleTable.
    varReturn = CheckLink(strSampleTable)

    If varReturn Then
        adhVerifyLinks = True
        GoTo adhVerifyLinksDone
    End If

#If USEDAO Then
    ' Učitavamo ime direktorijuma u kome se nalazi aplikaciona baza
    ' podataka.
    strDBDir = adhCurrentDBPath()
#Else
    strDBDir = CurrentProject.Path & "\"
#End If
    ' Ime baze podataka „za podatke" zadato je
    ' u parametru strDataDatabase.
```

```
    If (Dir$(strDBDir & strDataDatabase) <> "") Then
        ' Baza podataka za podatke pronađena je u tekućem direktorijumu
        varFileName = strDBDir & strDataDatabase
    Else
        ' Korisnik će pomoću okvira za dijalog sam pronaći
        ' bazu podataka za podatke.
        strMsg = "Neophodna datoteka '" & _
         strDataDatabase & _
         "' nije pronađena." & _
         " Pomoću sledećeg okvira za dijalog" & _
         " možete pokušati da je pronađete na svom računaru." & _
         " Ukoliko ne možete da je pronađete ili" & _
         " niste sigurni šta treba da uradite, u sledećem " & _
         " prozoru pritisnite CANCEL i pozovite" & _
         " administratora baze podataka."
        MsgBox strMsg, vbOKOnly + vbCritical, strProcName

        ' Prikazujemo okvir za dijalog Open File pozivanjem funkcije
        ' adhCommonFileOpenSave iz modula basFileOpen.
        strFilter = adhAddFilterItem( _
         strFilter, "Access (*.mdb)", "*.mdb")

        varFileName = adhCommonFileOpenSave( _
            OpenFile:=True, _
            Filter:=strFilter, _
            Flags:=cdlOFNHideReadOnly + cdlOFNNoChangeDir, _
            InitDir:=strDBDir, _
            DialogTitle:="Pronalaženje datoteke baze podataka")

        If Len(varFileName & "")=0 Then
            ' Korisnik je pritisnuo Cancel.
            strMsg = "Ovu bazu podataka ne možete da koristite " & _
             "dok ne pronađete datoteku '" & strDataDatabase & "'."
            MsgBox strMsg, _
             vbOKOnly + vbCritical, strProcName
            adhVerifyLinks = False
            GoTo adhVerifyLinksDone
        Else
            varFileName = adhTrimNull(varFileName)
        End If
    End If

    'Ponovno uspostavljanje veza. Najpre utvrđujemo ukupan broj tabela.
#If USEDAO Then
    Set db = CurrentDb
    intNumTables = db.TableDefs.Count
#Else
    Set cnn = CurrentProject.Connection
    Set cat = New ADOX.Catalog
    cat.ActiveConnection = cnn
    intNumTables = cat.Tables.Count
```

```
#End If
    varReturn = SysCmd(acSysCmdInitMeter, _
     "Uspostavljam veze s tabelama", intNumTables)

    ' Petlja za proveravanje svih tabela. Ponovo se povezuju
    ' samo one čije svojstvo Connect ne sadrži prazan znakovni niz.
    intI = 0
#If USEDAO Then
    For Each tdf In db.TableDefs
        ' Ako je vrednost svojstva Connect prazan niz,
        ' onda to nije pridružena tabela.
        If Len(tdf.Connect) > 0 Then
            intI = intI + 1
            tdf.Connect = ";DATABASE=" & varFileName

            ' Pošto pri izvršavanju metode RefreshLink može da
            ' nastane greška ukoliko nova putanja nije u redu,
            ' grešku obrađujemo u sledećem redu.
            On Error Resume Next
            tdf.RefreshLink
            'Ako je veza prekinuta, funkcija daje povratnu vrednost
            'False.
            If Err.Number <> 0 Then
                adhVerifyLinks = False
                GoTo adhVerifyLinksDone
            End If
        End If

        varReturn = SysCmd(acSysCmdUpdateMeter, intI + 1)
    Next tdf
#Else
    For Each tbl In cat.Tables
        ' Ako je svojstvo Type = "LINK", radi se o pridruženoj tabeli.
        If tbl.Type = "LINK" Then
            intI = intI + 1
            On Error Resume Next
            ' U sledećem redu veza se ponovo uspostavlja i osvežava.
            ' Ukoliko nova putanja nije u redu, nastaje greška koju
            ' obrađujemo u sledećem redu.
            tbl.Properties("Jet OLEDB:Link Datasource") = _
             varFileName
            'Ako je veza prekinuta, funkcija daje povratnu vrednost
            'False.
            If Err.Number <> 0 Then
                adhVerifyLinks = False
                GoTo adhVerifyLinksDone
            End If
        End If

        varReturn = SysCmd(acSysCmdUpdateMeter, intI + 1)
    Next tbl
```

```
#End If
    adhVerifyLinks = True

adhVerifyLinksDone:
    On Error Resume Next
    varReturn = SysCmd(acSysCmdRemoveMeter)
#If USEDAO Then
    Set tdf = Nothing
    Set db = Nothing
#Else
    Set tbl = Nothing
    Set cat = Nothing
    Set cnn = Nothing
#End If
    Exit Function

adhVerifyLinksErr:
    Select Case Err.Number
    Case Else
        Err.Raise Err.Number, Err.Source, _
         Err.Description, Err.HelpFile, Err.HelpContext
    End Select
    Resume adhVerifyLinksDone
End Function
```

SAVET

Funkciju smo napisali tako da radi i u DAO i u ADO objektnom modelu. Kôd u ovom poglavlju podešen je za ADO. Ukoliko umesto njega želite da koristite ADO, zadajte True kao vrednost konstante za uslovno prevođenje modula USEDAO.

Funkcija adhVerifyLinks prihvata dva parametra: strDataDatabase, koji sadrži ime baze podataka sa podacima u kojoj se nalaze pridružene tabele i strSampleTable, koji sadrži ime jedne od pridruženih tabela. Funkcija adhVerifyLinks počinje tako što proverava stanje veze sa zadatom tabelom. Pretpostavlja se da, ukoliko je ta veza u redu, onda to važi i za veze sa svim ostalim tabelama. (Ako želite, možete da izmenite kôd tako da proverava ispravnost svake veze pojedinačno.) Funkcija proverava stanje veze tako što poziva privatnu funkciju CheckLink, koja je prikazana u listingu 2.2.

Listing 2.2

```
Private Function CheckLink(strTable As String) As Boolean

    ' Proverava stanje veze sa zadatom tabelom.
    ' (Zapravo, daje False i kada tabela ne postoji.)

      On Error Resume Next
```

```
#If USEDAO Then
    Dim varRet As Variant

    ' Ako ne možemo da utvrdimo ime prvog polja tabele,
    ' veza je verovatno prekinuta.
    varRet = CurrentDb.TableDefs(strTable).Fields(0).Name
    If Err.Number <> 0 Then
        CheckLink = False
    Else
        CheckLink = True
    End If
#Else
    Dim cnn As ADODB.Connection
    Dim rst As ADODB.Recordset

    Set cnn = CurrentProject.Connection
    ' Metodom OpenSchema popunjavamo objekat tipa Recordset
    ' podacima o kolonama tabele, zadate u parametru strTable.
    ' Ako je skup podataka prazan, to znači da Jet nije mogao
    ' da pronađe tabelu jer je veza verovatno prekinuta.
    Set rst = cnn.OpenSchema(adSchemaColumns, _
     Array(Empty, Empty, strTable, Empty))
    CheckLink = Not rst.EOF

    rst.Close
    Set rst = Nothing
    Set cnn = Nothing
#End If

End Function
```

Kada koristite model DAO, funkcija CheckLink proverava ispravnost veze tako što pokušava da učita ime prvog polja tabele. Ako se ta operacija završi uspehom, veza je ispravna; u suprotnom, smatra se da je veza prekinuta, pa funkcija daje False kao povratnu vrednost. Kada koristite model ADO, funkcija proverava ispravnost veze tako što poziva posebnu metodu OpenSchema objekta Connection. Metoda OpenSchema popunjava objekat tipa Recordset raznim podacima o šemi baze podataka. (Tako se brže utvrđuje da li je veza s tabelom prekinuta, nego kada se koristi ADOX objekat tipa Table.)

Ako funkcija CheckLink kao povratnu vrednost daje False, onda funkcija adhVerifyLinks traži bazu podataka „za podatke“ u istom direktorijumu u kome se nalazi i aplikaciona baza podataka. Ako je nađe, funkcija ponovo uspostavlja veze s njenim tabelama. Ako se baza podataka „za podatke“ ne nalazi u istom direktorijumu kao aplikacija, funkcija prikazuje korisniku standardni Windowsov okvir za dijalog Open File.

Ako je baza potvrđena, funkcija adhVerifyLinks pokušava da ponovo uspostavi veze s pridruženim tabelama u toj bazi podataka. Kada se koristi model DAO, funkcija utvrđuje da li je određena tabela pridruženog tipa tako što najpre ispituje da li svojstvo Connect objekta TableDef sadrži znakovni niz koji nije prazan. Ako nije prazan, funkcija ponovo uspostavlja vezu s tabelom tako što menja vrednost

njenog svojstva Connect, a zatim poziva metodu RefreshLinks, kao u sledećem delu koda funkcije adhVerifyLinks:

```
tdf.Connect = ";DATABASE=" & varFileName
tdf.RefreshLink
```

Kada se koristi model ADO, funkcija utvrđuje da li je određena tabela pridruženog tipa tako što najpre ispituje da li svojstvo Type objekta Table sadrži vrednost „LINK“. Ako je tako, funkcija ponovo uspostavlja vezu s tabelom tako što koristi svojstvo ADOX objekta Table, koje je specifično za dobavljača Jet podataka, kao u sledećem delu koda funkcije adhVerifyLinks:

```
tbl.Properties("Jet OLEDB:Link Datasource") = _
 varImeDatoteke
```

Uklapanje pridruženih tabela u aplikacije

Kada jedinstvenu bazu podataka postojeće aplikacije razdvojite na bazu podataka „za podatke“ i na aplikacionu bazu podataka, vrlo je verovatno da ćete morati da izmenite određene delove VBA koda. Kada radite s pridruženim tabelama, ne možete direktno da koristite objekte Recordset tipa Table (tabela), niti metodu Seek, ali umesto njih možete da koristite neku od sledećih alternativnih strategija:

- Pravite objekte Recordset čiji tip nije Table i koristite sporiju metodu FindFirst (DAO), ili Find (ADO).
- Koristite metodu OpenDatabase objekta Workspace (DAO), ili metodu Open objekta Connection (ADO) da biste direktno otvorili bazu podataka „za podatke“. Zatim možete da formirate objekte Recordset tipa Table i da koristite metodu Seek, isto kao kada bi tabele bile lokalne.

Koju god metodu da odaberete, verovatno ćete morati da unesete određene izmene u kôd aplikacije. Međutim, moguće je napisati kôd koji koristi metodu Seek, bez obzira na to da li je tabela lokalna ili pridružena, što je prikazano u primeru koda u listingu 2.3 (DAO) i listingu 2.4 (ADO). Oba primera su iz modula basLinkedTables baze podataka ch02app.mdb.

Listing 2.3

```
Sub SeekLocalOrLinkedDAO(ByVal strTable As String, _
 ByVal strCompare As String, _
 Optional ByVal strIndex As String = "PrimaryKey")

    ' Izvršava DAO Seek metodu nad tabelom koristeći
    ' zadati indeks i uslove pretraživanja. Radi i sa lokalnim
    ' i sa pridruženim Accessovim tabelama.
    '
    ' Ulazni parametri:
    '     strTable: Ime tabele
    '     strCompare: Niz vrednosti koje treba pronaći, razdvojene
    '     zarezima
    '     strIndex: Ime indeksa. Podrazumeva se "PrimaryKey"
    ' Izlazni parametri:
```

```
'    U prozoru Immediate ispisuje listu vrednosti polja
'    ili poruku 'Zadata vrednost nije pronađena'.
' Primer:
'    Call SeekLocalOrLinkedDAO("tblCustomer",3)

Dim db As DAO.Database
Dim rst As DAO.Recordset
Dim fld As DAO.Field
Dim strConnect As String
Dim strDB As String
Dim intDBStart As Integer
Dim intDBEnd As Integer

Const adhcDB = "DATABASE="

Set db = CurrentDb
' Iz definicije tabele učitavamo vrednost niza sa parametrima
' za uspostavljanje veze
strConnect = db.TableDefs(strTable).Connect

' Ako je niz parametara jednak "", radi se o lokalnoj tabeli.
' U suprotnom, izdvajamo iz niza parametara deo koji
' se odnosi na bazu podataka.
strDb = ""
If Len(strConnect) > 0 Then
    intDBStart = InStr(strConnect, adhcDB)
    intDBEnd = InStr(intDBStart + Len(adhcDB), _
     strConnect, ";")
    If intDBEnd = 0 Then intDBEnd = Len(strConnect) + 1
    strDB = Mid(strConnect, intDBStart + Len(adhcDB), _
     intDBEnd - intDBStart)

    ' Otvaramo spoljnu bazu podataka.
    Set db = DBEngine.Workspaces(0).OpenDatabase(strDB)
End If

' Da bismo mogli da koristimo metodu Seek,
' treba da otvorimo Recordset objekat tipa Table.
Set rst = db.OpenRecordset(strTable, dbOpenTable)
rst.Index = strIndex

rst.Seek "=", strCompare

If Not rst.NoMatch Then
    ' Ovaj primer samo ispisuje u prozoru Immediate
    ' vrednosti iz svih polja pronađenog zapisa,
    ' ali je to dovoljno da shvatite princip...
    For Each fld In rst.Fields
        Debug.Print fld.Name & ": " & fld.Value
    Next
```

```
    Else
        Debug.Print "Zadata vrednost nije pronađena."
    End If

    Set fld = Nothing
    rst.Close
    Set rst = Nothing
    If Len(strDB) > 0 Then
        db.Close
    End If
    Set db = Nothing
End Sub
```

Listing 2.4

```
Sub SeekLocalOrLinkedADO(ByVal strTable As String, _
 ByVal varCompare As Variant, _
 Optional ByVal strIndex As String = "PrimaryKey")

    ' Izvršava ADO metodu Seek nad tabelom koristeći
    ' zadati indeks i uslove pretraživanja. Radi i sa lokalnim
    ' i sa pridruženim Accessovim tabelama.
    '
    ' Ulazni parametri:
    '     strTable: Ime tabele
    '     varCompare: Niz vrednosti koje treba pronaći
    '     strIndex: Ime indeksa. Podrazumeva se "PrimaryKey"
    ' Izlazni parametri:
    '    U prozoru Immediate ispisuje listu vrednosti polja
    '    ili poruku ' Zadata vrednost nije pronađena '.
    ' Primer:
    '    Call SeekLocalOrLinkedADO("tblCustomer",3)

    Dim cnn As ADODB.Connection
    Dim cat As ADOX.Catalog
    Dim rst As ADODB.Recordset
    Dim fld As ADODB.Field
    Dim strDB As String

    Set cnn = CurrentProject.Connection
    Set cat = New ADOX.Catalog
    cat.ActiveConnection = cnn

    ' Ako je ovo pridružena tabela, strDB će sadržati
    ' ime izvorne baze podataka; u suprotnom,
    ' strDB će sadržati prazan znakovni niz.
    strDB = cat.Tables(strTable). _
     Properties("Jet OLEDB:Link Datasource")
```

```
    If Len(strDB) > 0 Then
        ' Uspostavljamo vezu sa spoljnom bazom podataka.
        Set cnn = New ADODB.Connection
        cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
         "Data Source=" & strDB & ";"
    End If

    Set rst = New ADODB.Recordset
    ' Da bismo mogli da koristimo metodu Seek,
    ' treba da otvorimo Recordset objekat tipa Table.
    rst.Open strTable, cnn, adOpenKeyset, _
     adLockOptimistic, adCmdTableDirect
    rst.Index = strIndex

    rst.Seek varCompare, adSeekFirstEQ

    ' Ako tražena vrednost nije pronađena,
    ' svojtsvo EOF ima vrednost True.
    If Not rst.EOF Then
        ' Ovaj primer samo ispisuje u prozoru Immediate
        ' vrednosti iz svih polja pronađenog zapisa,
        ' ali je to dovoljno da shvatite princip...
        For Each fld In rst.Fields
            Debug.Print fld.Name & ": " & fld.Value
        Next
    Else
        Debug.Print "Zadata vrednost nije pronađena."
    End If

    Set fld = Nothing
    rst.Close
    Set rst = Nothing
    Set cat = Nothing
    If Len(strDB) > 0 Then
        cnn.Close
    End If
    Set cnn = Nothing
End Sub
```

Ova procedura radi tako što najpre iz šeme tabele učitava ime baze podataka „za podatke“. DAO verzija ove funkcije (SeekLocalOrLinkedDAO, listing 2.3) izdvaja taj podatak iz stavke DATABASE svojstva Connect objekta TableDef. ADO verzija (SeekLocalOrLinkedADO, listing 2.4) je jednostavnija jer ne morate da izdvajate ime baze podataka iz vrednosti svojstva Jet OLEDB: Link Datasource ADOX objekta tipa Table jer to svojstvo sadrži samo ime baze podataka.

Podešavanje baze podataka za rad u višekorisničkom okruženju

U Accessu postoji više opcija i parametara koji utiču na ponašanje aplikacija u višekorisničkom okruženju, što je opisano u narednim odeljcima.

Zadavanje režima u kome se baza podataka otvara

Režim rada u kome će Access otvoriti bazu podataka možete zadati na tri načina:

- Kada pokrećete Access sa komandne linije, možete zadati ime baze podataka i jedan od parametara /Excl, ili /Ro da biste bazu podataka otvorili u režimu isključivog korišćenja, odnosno samo za čitanje.
- U okviru za dijalog File Open, bazu podataka možete da otvorite u režimu isključivog korišćenja, samo za čitanje ili u oba.
- Standardni režim u kome se otvara baza podataka možete da zadate na sledeći način: izaberite Tools Options, a zatim na kartici Advanced izmenite vrednost parametra Default Open Mode. Ako želite da dozvolite samo jednokorisnički pristup, kao vrednost ovog parametra izaberite Exclusive, a ako želite višekorisnički pristup bazi podataka, izaberite vrednost Shared.

SAVET

Ako određenog korisnika želite da sprečite da bazu podataka otvori u režimu isključivog korišćenja, izaberite Tools Security User and Group Permissions, a zatim za tog korisnika uklonite znak potvrde ispred ovlašćenja OpenExclusive. Više detalja o tome naći ćete u poglavlju 8, *Zaštita aplikacije*.

Interval osvežavanja

Interval osvežavanja sadržaja baze podataka možete da podesite na kartici Advanced okvira za dijalog Options. Na kraju svakog intervala koji zadate u svojstvu Refresh Interval, Access automatski proverava u otvorenim skupovima podataka i u tabelarnim prikazima da li je došlo do izmena. Standardni interval osvežavanja je 60 sekundi, što za neke aplikacije može da bude presporo. Međutim, ukoliko interval osvežavanja postavite na prenisku vrednost, možete da generišete prevelik saobraćaj u mreži. Da biste utvrdili koja vrednost odgovara vašoj aplikaciji, možda ćete morati malo da eksperimentišete. Opšte pravilo glasi da što je mreža manja, kraći može da bude i interval osvežavanja bez štetnih posledica po saobraćaj u mreži.

Postupak osvežavanja možete da pokrenete i pomoću VBA koda. Da biste osvežili sadržaj tekućeg zapisa koji je prikazan na obrascu, upišite:

```
Me.Refresh
```

Da biste ponovo učitali sadržaj tekućeg zapisa koji je prikazan na obrascu, upišite:

```
Me.Requery
```

DAO i ADO objekti Recordset nemaju metode Refresh; međutim, možete da zahtevate „prisilno“ osvežavanje sadržaja tekućeg zapisa izjednačavanjem svojstva Bookmark objekta Recordset sa samim sobom, na ovaj način:

```
rst.Bookmark = rst.Bookmark
```

Da biste ponovo DAO ili ADO skup podataka popunili podacima, zadajte sledeću komandu:

```
rst.Requery
```

Osvežavanje sadržaja tekućeg zapisa – automatsko, koje Access obavlja na kraju svakog intervala osvežavanja, ili ručno, pozivanjem metode Refresh – brže je od izvršavanja metode Requery. Međutim, novi zapisi koje su drugi korisnici dodali pojavljuju se u skupu podataka tek pošto pozovete metodu Requery, dok zapisi koje su drugi korisnici izbrisali nestaju iz skupa podataka takođe tek kada pozovete metodu Requery (osim ako koristite dinamički ADO objekat Recordset).

SAVET

Čak i kada interval osvežavanja podesite na dužu vrednost, Access automatski osvežava sadržaj tekućeg zapisa svaki put kada korisnik pokuša da unese neku izmenu. Prednost kraćeg intervala osvežavanja sastoji se prevashodno u tome što se na taj način brže obezbeđuje vizuelna povratna informacija da je neki drugi korisnik zaključao ili izmenio sadržaj zapisa koji vi trenutno imate na ekranu.

Nivo zaključavanja

Da bi više korisnika moglo da istovremeno pristupa istom zapisu, mašina baze podataka Jet zaključava (blokira) i otključava zapise. Pre Accessa 2000, Jet je zaključavao stranice sa više zapisa, a ne pojedinačne zapise. Veličina stranice bila je 2 kilobajta (2048 bajta). To znači da je Jet najčešće zaključavao više od jednog zapisa. Koliko njih? To je zavisilo od toga koliko je zapisa Jet mogao da uklopi u stranicu. U zavisnosti od veličine zapisa, to je moglo da bude bilo šta u opsegu od jednog do tridesetak zapisa.

Na sreću, Access 2000 je uveo pravo zaključavanje na nivou zapisa. To ćete omogućiti ako u okviru za dijalog Options, na kartici Advanced, potvrdite polje Open Databases Using Record-Level Locking (slika 2.1). Zapravo, uključivanje/isključivanje ovog polja omogućava zaključavanje na nivou zapisa/stranice. Za većinu (ali ne i sve) operacija nad podacima ovaj parametar znači pravo zaključavanje na nivou zapisa (izuzeci su navedeni u narednom odeljku). S druge strane, ako iz polja Open Databases Using Record-Level Locking uklonite znak potvrde, Jet će zaključavati cele stranice zapisa. Standardno važi zaključavanje na nivou pojedinačnog zapisa.

Nivo zaključavanja određuje prvi korisnik koji otvori bazu podataka. Pošto to prvi korisnik uradi, više ne možete da menjate nivo zaključavanja dok se svi korisnici ne odjave iz baze podataka.

NAPOMENA

Veličina stranice u bazama podataka Accessa 2000 i Accessa 2002 povećana je sa 2 na 4 kilobajta. To je bilo neophodno da bi se obezbedila podrška za Unicode.

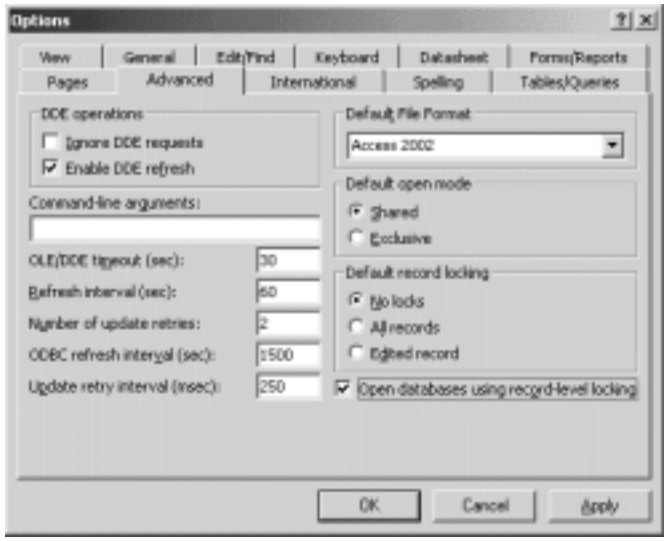
Zaključavanje zapisa/stranice

Kada zadate zaključavanje zapisa/stranica, u nekim operacijama Jet zaključava pojedinačne zapise, dok u drugim zaključava cele stranice.

Jet primenjuje zaključavanje na nivou *zapisa* kada se za učitavanje/upisivanje podataka koriste:

- Accessovi obrasci
- DAO objekti Recordset
- ADO objekti Recordset, osim kada pomoću svojstva „Jet OLE DB: Locking Granularity“ drugačije zadate.

SLIKA 2.1
Zadavanje načina zaključavanja zapisa.



Jet primenjuje zaključavanje na nivou *stranica* u sledećim slučajevima učitavanja/upisivanja podataka:

- Ažuriranje podataka pomoću SQL iskaza koji deluju na grupe zapisa
- Ažuriranje stranica indeksa
- Ažuriranje podataka tipa Memo
- Ažuriranje pomoću ADO objekata Recordset čije svojstvo „Jet OLE DB: Locking Granularity“ ima vrednost 1.

NAPOMENA Mana zaključavanja na nivou zapisa jeste što u tom slučaju Jet ne može dati podatke o imenu mašine i imenu korisnika koji je zapis zaključao. (Pri zaključavanju na nivou stranica zapisa, te podatke Jet može da prikaže u porukama o greškama.)

Zaključavanje na nivou stranica

Kada u bazi podataka zadate zaključavanje na nivou stranica, Jet uvek zaključava cele stranice zapisa. To je isto ponašanje kao u prethodnim verzijama mašine Jet.

Izbor odgovarajućeg nivoa zaključavanja

Zaključavanje na nivou zapisa pruža znatno bolje mogućnosti višekorisničkog pristupa podacima nego zaključavanje na nivou stranice. To znači da, kada se primenjuje zaključavanje na nivou zapisa, više korisnika može istovremeno da ažurira zapise u istoj tabeli. Međutim, to ne znači uvek i bolje performanse. Kada se istovremeno ažurira veliki broj zapisa, zaključavanje na nivou stranice može da obezbedi bolje performanse od zaključavanja na nivou zapisa.

SAVET

U većini slučajeva, bolje mogućnosti višekorisničkog pristupa podacima koje pruža zaključavanje na nivou zapisa, nadoknađuju nešto slabije performanse u poređenju sa zaključavanjem na nivou stranica.

Trenutak zaključavanja

Osim izbora odgovarajućeg nivoa zaključavanja (na nivou zapisa ili na nivou stranice), možete da zadate i trenutak zaključavanja, odnosno kada Jet treba da zaključa podatke. Objekte Recordset možete da otvarate u jednom od sledećih režima rada (svaki od njih detaljnije je opisan u narednim odeljcima):

No Locks Ovaj režim se često naziva *optimističko zaključavanje* (engl. *optimistic locking*). Ako zadate No Locks, zapis (ili stranica koja sadrži zapis koji se trenutno ažurira ukoliko se primenjuje zaključavanje na nivou stranice) zaključan je samo u trenutku kada se upisuju izmene u bazu podataka, ali ne i dok ga korisnik ažurira.

Edited Record Čim korisnik počne da ažurira sadržaj zapisa, zapis (ili stranica sa zapisom koji se trenutno ažurira ukoliko se primenjuje zaključavanje na nivou stranice) zaključava se i ostaje zaključan dok izmene ne budu upisane u bazu podataka. To je poznato i kao *pesimističko zaključavanje* (engl. *pessimistic locking*).

All Records Ova vrednost čini da svi zapisi u celom objektu Recordset postanu zaključani. Ova opcija nije naročito korisna, osim za grupna ažuriranja ili za administrativno održavanje tabela.

Opcije koje određuju način zaključavanja objekata baze podataka, koji rade sa objektima Recordset, možete podešavati. Tabela 2.1 prikazuje koje su sve opcije zaključavanja na raspolaganju za pojedine vrste objekata baze podataka, kao i trenutak kada se zapisi zaključavaju. Parameter RecordLocks većine ovih objekata preuzima vrednost opcije Default Record Locking, koja se zadaje na kartici Advanced okvira za dijalog Options (slika 2.1).

TABELA 2.1: Vrednosti svojstva RecordLocks raznih Accessovih objekata

Accessov objekat	No Locks	Edited Record	All Records	Podrazumeva se	Kada se zapis zaključava
Tabelarni prikaz sadržaja tabele	Da[1]	Da[1]	Da[1]	DRL	Prilikom izmene sadržaja tabelarnog prikaza
Tabelarni prikaz rezultata upita tipa Select	Da	Da	Da	DRL	Prilikom izmene sadržaja tabelarnog prikaza
Tabelarni prikaz rezultata upita tipa Crosstab	Da	Da	Da	DRL	Pri izvršavanju upita
Tabelarni prikaz rezultata upita tipa Union	Da	Da	Da	DRL	Pri izvršavanju upita
Upiti za brisanje i ažuriranje podataka	Ne	Da	Da	DRL	Pri izvršavanju upita
Upiti za pravljenje tabela i dodavanje podataka	Ne	Da	Da	DRL	Pri izvršavanju upita[2]

TABELA 2.1: Vrednosti svojstva RecordLocks raznih Accessovih objekata *(nastavak)*

Accessov objekat	No Locks	Edited Record	All Records	Podrazumeva se	Kada se zapis zaključava
Upiti za definisanje podataka	Ne	Ne	Da	All Records[3]	Pri izvršavanju upita.
Obrasci	Da	Da	Da	DRL	Prikazi Form i Datasheet obrasca.
Izveštaji	Da	Ne	Da	DRL	Izvršavanje izveštaja, njegovo prikazivanje i štampanje.
DAO objekat Recordset	Da	Da	Da	Zapis koji se ažurira[4]	Između poziva metoda Edit i Update.
ADO objekat Recordset	Da	Da	Da	Samo pravo čitanja[5]	Između trenutka početka ažuriranja i pozivanja metode Update.

Da = opcija je na raspolaganju.

Ne = opcija nije na raspolaganju za ovaj objekat

DRL = vrednost opcije Default Record Locking baze podataka.

1 Tabelarni prikazi tabela nemaju opciju RecordLocks, već za njih važi opcija Record Locking baze podataka.

2 Pri izvršavanju upita za pravljenje tabela ili za dodavanje podataka postojećim tabelama, zaključava se cela ciljna tabela.

3 Upiti za definisanje podataka nemaju svojstvo RecordLocks. Kada se oni izvršavaju, Access zaključava celu tabelu.

4 Ovakvo ponašanje može se izmeniti zadavanjem svojstva LockEdits DAO objekta Recordset. Standardno se zaključava samo zapis koji se ažurira, osim kada pri pozivanju metode OpenRecordset zadate opciju dbDeny-Write ili dbDenyRead; u tom slučaju se zaključava cela tabela.

5 Može se drugačije zadati pomoću svojstva LockType ADO objekta Recordset ili pomoću argumenta LockType metode Recordset.Open.

Optimističko zaključavanje

Optimističko zaključavanje (vrednost svojstva RecordLocks je No Locks) omogućava da više korisnika istovremeno ažurira isti zapis uz manju verovatnoću sukoba pri zaključavanju zapisa. Međutim, time se povećava rizik nastajanja sukoba pri upisivanju podataka. *Sukob pri upisivanju* (engl. *write conflict*) nastaje kada:

1. Prvi korisnik počinje da ažurira sadržaj zapisa.

2. Drugi korisnik upisuje u bazu podataka izmene zapisa koje je on uneo.

3. Prvi korisnik pokuša da u tom trenutku upiše svoje izmene.

Sukob pri upisivanju je štetan jer on znači da prvi korisnik ažurira drugačiji zapis od zapisa s kojim je počeo da radi.

Pesimističko zaključavanje

Kada primenjujete pesimističko zaključavanje (svojstvo RecordLocks = Edited Record), u svakom datom trenutku samo jedan korisnik može da menja sadržaj zapisa. To je veliki problem kada primenjujete zaključavanje na nivou stranice jer u svakom trenutku samo jedan korisnik može da ažurira bilo koji zapis koji se nalazi na zaključanoj stranici.

Izbor odgovarajućeg trenutka zaključavanja

U većini slučajeva ne biste želeli da dva korisnika menjaju sadržaj istog zapisa u isto vreme. Na osnovu ove činjenice zaključili biste da treba da primenjujte pesimističko zaključavanje. Međutim, u verzijama Accessa pre Accessa 2000 to je značilo zaključavanje cele stranice zapisa, a ne samo određenog zapisa, što je često bilo neprihvatljivo. Zbog toga smo u prethodnim izdanjima ove knjige preporučivali optimističko zaključavanje, osim u aplikacijama u kojima bi sukobi pri upisivanju bili neprihvatljivi. Međutim, pošto Access 2002 omogućava zaključavanje pojedinačnih zapisa, smatramo da je pesimističko zaključavanje bolji izbor za većinu aplikacija. Ipak, ponekad ćete naići i na slučajeve u kojima bi trebalo da razmotrite i primenu optimističkog zaključavanja. Na primer, ako korisnici imaju običaj da zapise zaključavaju duže vreme, može biti pogodnije da se opredelite za optimističko zaključavanje. Tu vrstu zaključavanja možete da primenjujete kada zaključavate cele stranice jer, na primer, vaša aplikacija sadrži veze ka bazi podataka u formatu Accessa 97.

SAVET

Ukoliko postoji mogućnost zaključavanja pojedinačnih zapisa, preporučujemo da primenjujete pesimističko zaključavanje u većini Access 2002 aplikacija koje koriste mašinu baze podataka Jet 4.

Zaključavanje i obrasci

Kada koristite vezane obrasce, Access se automatski stara o zaključavanju zapisa. U narednih nekoliko odeljaka opisano je kako se to odvija i kako možete da izmenite standardno ponašanje Accessa kada se radi o zaključavanju.

Optimističko zaključavanje u obrascima

Kao što je već bilo pomenuto u prethodnom delu ovog poglavlja, najozbiljniji problem sa optimističkim zaključavanjem jeste mogućnost nastajanja sukoba pri upisivanju. Kada u vezanom obrascu dođe do takvog sukoba, Access prikazuje okvir za dijalog Write Conflict, kao u primeru na slici 2.2, gde je prikazan sukob u obrascu frmCustomerOptimistic1 iz baze podataka ch02app.mdb.

Ovaj okvir za dijalog nudi korisniku tri mogućnosti:

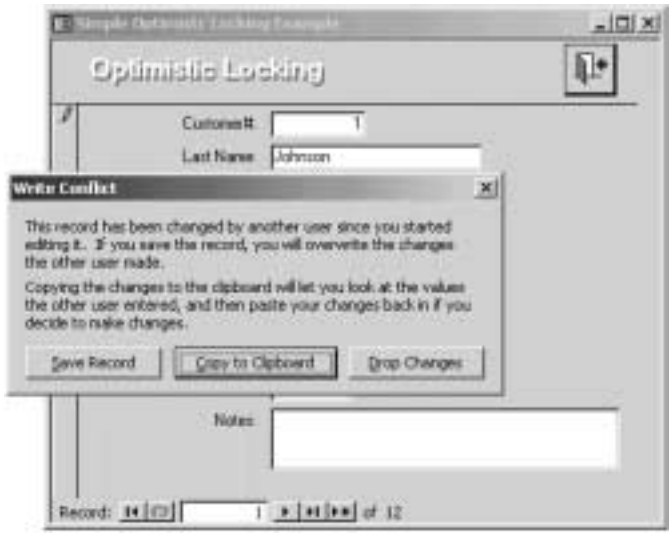
Save Record Ako se korisnik opredeli za ovu opciju, izmene koje je on uneo poništavaju izmene koje je drugi korisnik uneo. Zbog toga u većini slučajeva treba izbegavati ovu opciju; ona „po kratkom postupku“ odbacuje izmene koje je uneo drugi korisnik.

Copy to Clipboard Ova opcija kopira na Clipboard izmene koje je tekući korisnik uneo i osvežava sadržaj zapisa izmenama koje je uneo drugi korisnik. To je dobar izbor za korisnike koji shvataju u čemu je problem, ali zahteva znanje kojim prosečan korisnik ne raspolaže.

Drop Changes Kada korisnik izabere ovu opciju, odbacuju se izmene koje je on uneo, a zapis se ažurira izmenama koje je uneo drugi korisnik.

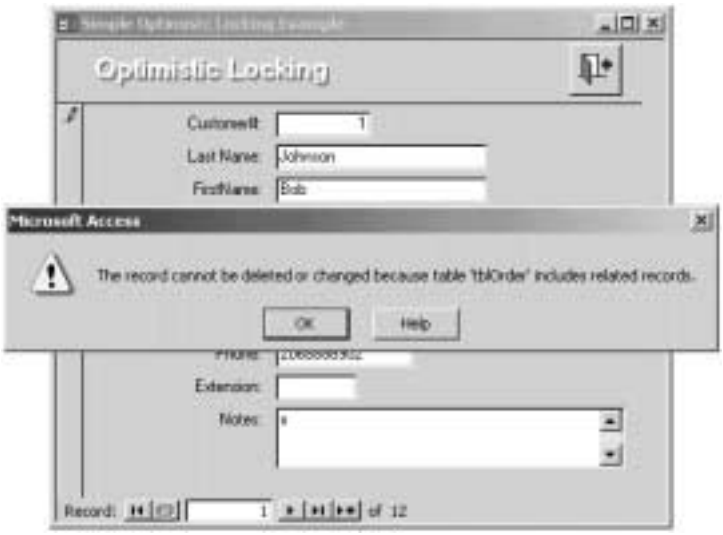
U Accessu 2002 postoji greška (koja je tu još od vremena Accessa 2 – neke od njih se zaista sporo otklanjaju) koja se pojavljuje kada između dve tabele imate vezu tipa „jedan prema više“ u kojoj je *isključeno* lančano ažuriranje, u okviru za dijalog Write Conflict izaberete opciju Save Record za zapis koji se nalazi na strani

SLIKA 2.2
Kada se primenjuje optimističko zaključavanje, korisnik može naići na okvir za dijalog Write Conflict kada pokuša da u bazu podataka snimi zapis čiji je sadržaj izmenio drugi korisnik.



„jedan" veze. U takvim slučajevima pojavljuju se pogrešne poruke o narušavanju referencijalnog integriteta čak i kada ne menjate vrednost primarnog ključa (slika 2.3). Ta poruka se pojavljuje zato što Access u sva polja kopira vaše vrednosti preko onih koje je drugi korisnik upisao, a da prethodno ne proverava da li je sadržaj svakog polja zaista bio izmenjen. Pošto je jedno od polja (ili više njih) primarni ključ, Jet smatra da je izmenjena i njegova vrednost, pa šalje poruku o nepostojećoj grešci.

SLIKA 2.3
Poruka o nepostojećoj grešci može da se pojavi kada u okviru za dijalog Write Conflict izaberete opciju Save Record.



Optimističko zaključavanje i obrada grešaka u kodu

Događaj Error u obrascu možete da iskoristite za presretanje grešaka koje su nastale usled sukoba pri upisivanju tako što ćete ih obraditi pomoću VBA koda u proceduri za obradu događaja Error. Vašoj proceduri Access prosleđuje parametre DataErr i Response. Dve najčešće vrednosti parametra DataErr pri optimističkom zaključavanju opisane su u tabeli 2.2.

TABELA 2.2: Greške koje nastaju u obrascima u kojima se primenjuje optimističko zaključavanje

Broj greške	Tekst poruke o grešci	Napomena
7787	Write Conflict: This record has been changed by another user since you started editing it … (Sadržaj ovoga zapisa je izmenjen od trenutka kada ste vi počeli da ga ažurirate …)	Drugi korisnik je snimio svoje izmene dok je prvi korisnik još unosio svoje.
7878	The data has been changed … (Podaci su bili izmenjeni …)	Drugi korisnik je uneo izmene dok je prvi pregledao sadržaj zapisa.

Greška broj 7787 uzrok je pojavljivanja standardnog okvira za dijalog Write Conflict. Ona nastaje kada korisnik pokuša da *snimi izmene* zapisa čiji je sadržaj već izmenjen. Greška broj 7878 nastaje kada korisnik *počne da menja* zapis čiji je sadržaj drugi korisnik izmenio nakon što mu je prvi korisnik pristupio; nastajanje ove greške je verovatnije u slučaju dugih intervala osvežavanja.

NAPOMENA Pošto događaj Error nastaje kad god se pojavi neka greška pri pristupanju podacima, u svakoj proceduri za obradu događaja koju napišete treba da vodite računa i o drugim greškama u vezi s pristupanjem podacima čiji uzrok nije zaključavanje.

Kao vrednost parametra Response možete zadati jednu od sledećih ugrađenih konstanti:

Vrednost parametra Response	Kada se koristi
acDataErrContinue	Da biste naložili Accessu da nastavi obradu, a da ne prikazuje poruku o grešci; u slučaju optimističkog zaključavanja, ova vrednost čini da se sadržaj zapisa osvežava s tim da se odbacuju izmene koje je uneo tekući korisnik.
acDataErrDisplay	Da biste naložili Accessu da prikaže standardnu poruku o grešci, zadajte ovu konstantu za slučaj da se pojave greške koje ne obrađujete posebno.

Zapazićete da ne postoji način da naložite Jetu da zapis popuni „vašim“ izmenama. Osim što možete zadati standardnu poruku o grešci, jedina preostala mogućnost jeste da dopustite Jetu da, umesto izmena koje je uneo tekući korisnik, upiše izmene koje je drugi korisnik već uneo. Postupak koji se pokreće kada izaberete

opciju Save Record, deo je Accessovog korisničkog interfejsa koji ne može da se menja u kodu. Međutim, osmislili smo zamenu koja daje isti rezultat, ali bez greške pomenute u prethodnom odeljku.

Procedura za obradu događaja Error u obrascu frmCustomerOptimistic2 u bazi podataka ch02app.mdb ilustruje tu zamenu. Listing 2.5 sadrži proceduru za obradu događaja i prateći kôd u proceduri za obradu događaja Current u obrascu frmCustomerOptimistic2.

NAPOMENA U ovom obrascu koristi se DAO kôd za rad sa „pozadinskim" objektom Recordset, ali se lako može izmeniti tako da se umesto njega koristi ADO skup podataka.

Listing 2.5

```
Dim mvarCustomerId As Variant

Const adhcErrWriteConflict = 7787
Const adhcErrDataChanged = 7878

Private Sub Form_Error(DataErr As Integer, _
 Response As Integer)
    ' Obrađuje greške koje se pojavljuju u obrascu

    On Error GoTo Form_ErrorErr

    Dim strMsg As String
    Dim intResp As Integer
    Dim rst As DAO.Recordset
    Dim fld As DAO.Field
    Dim db As DAO.Database

    ' Grananje u zavisnosti od broja greške
    Select Case DataErr

    Case adhcErrWriteConflict
        ' Greška usled sukoba pri upisivanju podataka
        strMsg = "Drugi korisnik je izmenio sadržaj ovog zapisa " & _
         "nakon što ste počeli da ga ažurirate." & vbCrLf & vbCrLf & _
         "Želite li da preuzmete izmene koje je drugi korisnik uneo?" & _
          vbCrLf & vbCrLf & _
         "Izaberite Yes da biste ih preuzeli, " & vbCrLf & _
         "ili No da biste upisali svoje izmene."
        intResp = MsgBox(strMsg, _
         vbYesNo + vbDefaultButton1 + vbQuestion, _
         "Sukob pri upisivanju podataka")
```

```
' Pošto Jet prihvata samo upisivanje izmena
' koje je drugi korisnik uneo, moramo ga prevariti
' tako da prihvati naše izmene. To postižemo
' tako što naše izmene upisujemo u dopunski zapis
' čiji će sadržaj Access zatim upisati preko naših
' izmena.
If intResp = vbNo Then

    Set db = CurrentDb

    ' Formiramo objekat Recordset koji sadrži samo jedan
    ' zapis u kome se nalazi kopija vrednosti primarnog
    ' ključa u trenutku kada smo započeli ažuriranje
    ' zapisa. Ta vrednost je bila uneta u proceduri za
    ' obradu događaja Current. To je neophodno zato što
    ' može da se promeni i vrednost primarnog ključa.
    Set rst = db.OpenRecordset("SELECT * FROM " & _
     "tblCustomer WHERE [CustomerId] = " _
     & mvarCustomerId)

    ' Proveravamo da li je drugi korisnik
    ' promenio vrednost primarnog ključa.
    If rst.RecordCount = 0 Then
        strMsg = "Drugi korisnik je izmenio " & _
         "sadržaj polja Customer# u ovom zapisu. " & _
         "Trebalo bi da osvežite sadržaj zapisa " & _
         "pre nego što nastavite."
        MsgBox strMsg, vbOKOnly + vbInformation, _
         "Sukob pri upisivanju"
    Else
        ' Ažuriramo sadržaj dopunskog zapisa
        ' izmenjenim vrednostima sa obrasca.
        DoCmd.Hourglass True
        For Each fld In rst.Fields
            rst.Edit
                If (fld <> Me(fld.Name)) Or _
                 (IsNull(fld) <> _
                 IsNull(Me(fld.Name))) _
                 Then
                    fld.Value = _
                     Me(fld.Name).Value
                End If
            rst.Update
        Next fld
    End If
End If

' Ovim nalažemo osvežavanje sadržaja zapisa
Response = acDataErrContinue
```

```
        Case adhcErrDataChanged
            ' Ova greška nastaje kada Access otkrije da je
            ' drugi korisnik izmenio zapis nakon što smo mu mi
            ' pristupili da bismo uneli izmene u njega. Nema razloga
            ' za brigu jer još nismo uneli nikakvu izmenu.
            strMsg = "Drugi korisnik je izmenio sadržaj ovog " & _
            "zapisa od trenutka kada ste počeli da radite s njim." _
             & vbCrLf & vbCrLf & _
             "Pre nego što nastavite, zapis će biti osvežen izmenama " & _
             "koje je drugi korisnik uneo."
            MsgBox strMsg, vbOKOnly + vbInformation, _
             "Osvežavanje zapisa"

            ' Ovim nalažemo osvežavanje zapisa
            Response = acDataErrContinue
        Case Else
            ' U ostalim slučajevima Access treba
            ' da prikazuje standardnu poruku o grešci.
            Response = acDataErrDisplay
            Debug.Print DataErr, Error(DataErr)
        End Select

        DoCmd.Hourglass False

    Form_ErrorEnd:
        If Not rst Is Nothing Then
            Set rst = Nothing
        End If
        If Not fld Is Nothing Then
            Set fld = Nothing
        End If
        If Not db Is Nothing Then
            Set db = Nothing
        End If
        Exit Sub

    Form_ErrorErr:
        ' Može se dogoditi da sami izazovemo grešku dok
        ' obrađujemo grešku u vezi s podacima. Na primer,
        ' neko bi mogao da pesimistički zaključa zapis dok
        ' mi pokušavamo da ga ažuriramo.
        ' U tom slučaju, korisniku javljamo da postoji greška
        ' i napuštamo proceduru.
        MsgBox "Error " & Err.Number & ": " & Err.Description, _
         vbOKOnly + vbCritical, "Error Handler Error"
    End Sub
```

Kada nastane greška broj 7787, obrazac frmCustomerOptimistic2 prikazuje poruku o sukobu pri upisivanju podataka (slika 2.4). Ako korisnik izabere opciju Yes, parametru Response procedura dodeljuje vrednost acDataErrContinue i prekida rad. Ako korisnik izabere opciju No, procedura preuzima vrednosti iz tekućeg

zapisa čiji je sadržaj prikazan na obrascu (tekući zapis) i kopira ih u nov objekat Recordset, koji sadrži kopiju zapisa u kome je nastao sukob. Zatim, kada procedura za obradu događaja postavi vrednost parametra Response na acDataErrContinue i završi s radom, Jet kopira „naše" vrednosti preko onih koje sadrži tekući zapis, a to je jednako ponašanje kao kada u okviru za dijalog Write Conflict izaberemo opciju Save Record. Međutim, pošto kopiramo samo sadržaje polja koji se razlikuju od vrednosti koje je drugi korisnik upisao, izbegavamo poruku o narušavanju referencijalnog integriteta:

```
For Each fld In rst.Fields
      rst.Edit
           If (fld <> Me(fld.Name)) Or _
            (IsNull(fld) <> _
            IsNull(Me(fld.Name))) _
            Then
               fld.Value = _
                Me(fld.Name).Value
           End If
     rst.Update
Next fld
```

SLIKA 2.4
Obrazac frmCustomer-Optimistic2 prikazuje namensku poruku o sukobu pri upisivanju podataka.

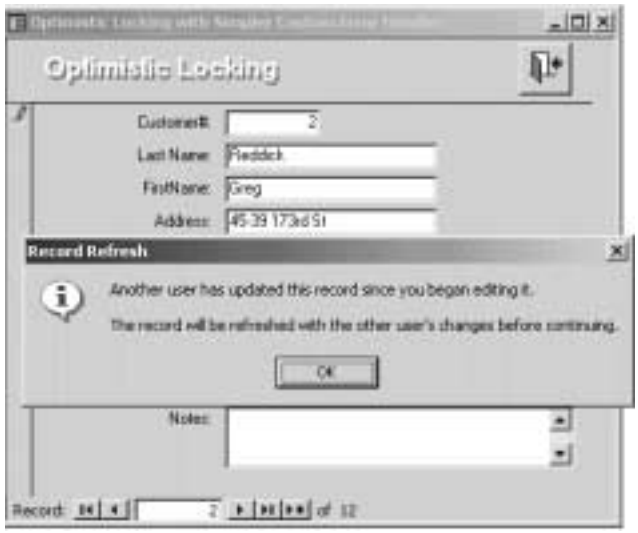
Suština ove zamene je u tome što se u proceduri za obradu događaja Current vrednost primarnog ključa upisuje u promenljivu mvarCustomerId, koja je globalna na nivou modula. To moramo da uradimo zato što treba da otvorimo nov objekat Recordset, s podacima iz tekućeg zapisa, ali ništa nam ne garantuje da korisnik nije izmenio vrednost primarnog ključa. To je razlog zbog kojeg koristimo vrednost primarnog ključa koju smo zabeležili u događaju Current. Druga mogućnost je da dozvolimo samo čitanje sadržaja polja primarnog ključa, ili da koristimo polje tipa AutoNumber, koje samo po sebi ne dozvoljava upisivanje.

Ako u obrascu koristite kombinovan primarni ključ ili upit koji obuhvata više tabela, moraćete da zabeležite vrednosti više polja.

NAPOMENA

U mnogim aplikacijama verovatno nećete želeti da korisniku ponudite mogućnost da upiše svoje izmene pre nego što prethodno pregleda one koje je uneo drugi korisnik. U takvim slučajevima je možda bolje da koristite jednostavniju proceduru za obradu grešaka pri optimističkom zaključavanju koja najpre obaveštava korisnika da je drugi korisnik uneo izmene u zapis, a zatim osvežava sadržaj zapisa. Primer takve jednostavnije procedure za obradu grešaka nalazi se u bazi podataka ch02app.mdb, a pridružen je obrascu frmCustomerOptimistic3 (slika 2.5).

SLIKA 2.5
Obrazac frmCustomer-Optimistic3 prikazuje jednostavniju poruku o sukobu pri upisivanju podataka.

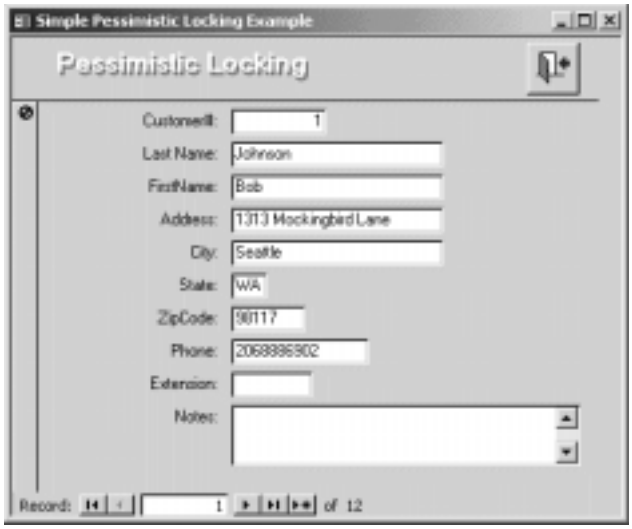


Pesimističko zaključavanje u obrascima

Obrasci u kojima se primenjuje pesimističko zaključavanje eliminišu sukobe pri upisivanju podataka jer je, u svakom trenutku, samo jednom korisniku dozvoljeno da menja sadržaj zapisa. Pojava ikonice sa precrtanim slovom O (Ø) obaveštava druge korisnike da je zapis zaključan, kao što je to pokazano na primeru obrasca frmCustomerPessimistic1 iz baze podataka koja je pridružena ovom poglavlju, ch02app.mdb (slika 2.6).

SAVET

Ako svojstvu RecordSelector obrasca dodelite vrednost No, ikonica s precrtanim slovom O neće se pojaviti kada je zapis pesimistički zaključan. Access će aktivirati zvučni signal, ali osim zvučne, korisnici neće imati nikakvu drugu naznaku o razlogu zbog koga ne mogu da menjaju vrednosti u poljima zapisa. Osim toga, ne generiše se ni greška koju biste mogli da obradite jer, u suštini, greške i nema. To znači da je važno da vrednost svojstva RecordSelector ostane Yes kada primenjujete pesimističko zaključavanje, osim ako osmislite neki svoj mehanizam obaveštavanja korisnika da je zapis zaključan.



SLIKA 2.6
Kada primenjujete pesimističko zaključavanje, ikonica sa precrtanim slovom O na biraču zapisa obaveštava korisnika da je tekući zapis zaključan.

Pesimističko zaključavanje s vremenskim ograničenjem

Pošto Access 2002 podržava pravo zaključavanje na nivou zapisa, pesimističko zaključavanje (važi opcija Edited Record) znatno je prihvatljivija opcija nego što je bila za vreme Accessa 97. Pa ipak, čak i kada korisnici zaključavaju pojedinačne zapise, ne postoji ništa što bi ih sprečilo da određeni zapis blokiraju preterano dugo. Svi smo čuli priču o korisniku koji je počeo da ažurira zapis, a zatim je otišao na ručak, ili, još gore, na godišnji odmor!

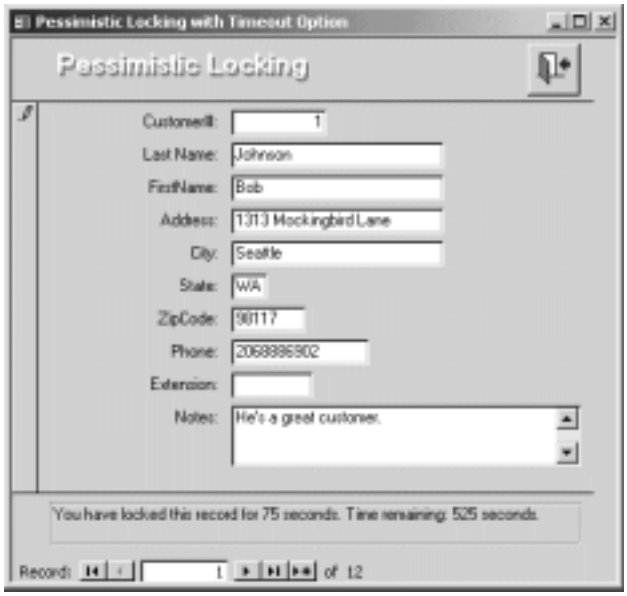
Klasa LockTimeout

Napisali smo modul klase, LockTimeout, koji možete da iskoristite da biste obrascu dodali mogućnost vremenskog ograničavanja. U obrascu frmCustomerPessimistic2, koji je prikazan na slikama 2.7 i 2.8, upotreba klase LockTimeout omogućava da se izmene koje korisnik nije snimio posle deset minuta automatski poništavaju.

Programski kôd pridružen obrascu frmCustomerPessimistic2 prikazan je u listingu 2.6. U proceduri za obradu događaja Load obrasca, pravimo primerak objekta mltoCustomer i pozivamo metodu BindForm da bismo obrazac povezali s klasom LockTimeout. Metoda BindForm prihvata četiri argumenta:

- Referencu na tekući obrazac.
- Referencu na objekat tipa natpis (Label) na obrascu koji će se koristiti za ispisivanje poruke o statusu.
- Interval (u sekundama) ažuriranja poruke o statusu.
- Dužina (u sekundama) intervala posle kog se poništavaju izmene koje je korisnik uneo.

SLIKA 2.7
Obrazac frmCustomerPessimistic2 obaveštava korisnika koliko dugo drži zapis zaključan i koliko mu vremena još preostaje pre nego što izmene koje je uneo budu poništene.



SLIKA 2.8
Obrazac frmCustomerPessimistic2 poništava izmene koje je korisnik uneo ukoliko je zapis bio zaključan deset minuta.

Listing 2.6

```
' Objektna promenljiva tipa LockTimeout
Private mltoCustomer As LockTimeout

Private Sub cmdClose_Click()
    DoCmd.Close acForm, Me.Name
End Sub

Private Sub Form_AfterUpdate()
    ' Odbrojavanje vremena prekidamo kada korisnik
    ' snimi zapis u bazu podataka
    mltoCustomer.StopTimer
End Sub

Private Sub Form_Dirty(Cancel As Integer)
    ' Započinjemo odbrojavanje vremena od trenutka kada
    ' korisnik izmeni sadržaj nekog od polja zapisa
    mltoCustomer.StartTimer
End Sub

Private Sub Form_Load()
    ' Formiramo objekat tipa LockTimeout
    Set mltoCustomer = New LockTimeout
    ' objekat LockTimeout povezujemo s tekućim obrascem
    mltoCustomer.BindForm _
      FormRef:=Me, LabelRef:=lblLockStatus, _
      CheckInterval:=1, TimeoutPeriod:=10 * 60
End Sub

Private Sub Form_Timer()
    ' Ispitujemo šta treba da se uradi
    mltoCustomer.CheckTimer
End Sub

Private Sub Form_Undo(Cancel As Integer)
    ' Prekidamo odbrojavanje vremena
    ' kada korisnik poništi izmene
    mltoCustomer.StopTimer
End Sub
```

Iskoristili smo događaj Undo, koji je nov u Accessu 2002. Ovaj događaj se pokreće kada korisnik poništi izmene unete u obrazac.

Modul klase LockTimeout prikazan je u listingu 2.7. U njemu se pomoću Windowsove API funkcije timeGetTime odbrojava vreme. Klasa LockTimeout koristi pet privatnih promenljivih za evidentiranje sledećih podataka: vreme u kome korisnik drži zapis zaključan, referencu na „povezan“ obrazac, referencu na objekat tipa natpis koji treba ažurirati, interval objekta tipa Timer i trajanje ograničenja.

Metoda BindForm povezuje instancu klase sa zadatim obrascem tako što dodeljuje vrednosti svim privatnim promenljivama na nivou modula, ali osim toga ne radi ništa drugo.

Pozivanjem API funkcije timeGetTime, metoda StartTimer beleži tačno vreme početka odbrojavanja i izaziva događaj Timer u obrascu.

Metoda StopTimer isključuje događaj Timer u obrascu i dodeljuje vrednost 0 promenljivoj mlngLockStart, u kojoj se čuva vreme početka odbrojavanja. Osim toga, ova metoda prazni i tekst natpisa o statusu.

Metoda CheckTimer odrađuje veći deo posla koji klasa obavlja. Pošto se ne generiše nikakav događaj kada se ponište izmene unete u obrazac, ova metoda najpre ispituje svojstvo Dirty obrasca da bi utvrdila da li je bilo izmena. Ako nije, promenljivoj mlngLockStart metoda dodeljuje vrednost 0.

Ako je bilo izmena, procedura CheckTimer izračunava koliko vremena korisnik drži obrazac (zapis) zaključan i koliko još vremena preostaje pre nego što izmene koje je korisnik uneo budu „prisilno“ poništene. Metoda zatim izvršava jednu od sledeće tri akcije:

- Ako je vreme isteklo, pozivanjem metode Undo obrasca poništavaju se izmene koje je korisnik uneo.
- Ako je isteklo više od 90% intervala čekanja, korisnik se na to upozorava.
- Ako je isteklo manje od 90% intervala čekanja, korisnik se o tome samo obaveštava.

Listing 2.7

```
' Koristi se za obračunavanje vremena
Private Declare Function timeGetTime Lib "winmm.dll" () _
 As Long

Private mlngLockStart As Long ' Vreme kada je korisnik zaključao zapis
Private mfrmBound As Form ' Obrazac koji je povezan sa objektom
Private mlngTimerInterval As Long ' Koliko često treba da ispitujemo
Private mlblStatus As Label ' Objekat tipa Label za ispisivanje statusa
Private mlngTimeout As Long ' Koliko dugo čekamo

Public Sub BindForm(FormRef As Form, LabelRef As Label, _
 CheckInterval As Long, TimeoutPeriod As Long)
    ' Ova procedura povezuje instancu klase LockTimeout sa obrascem

    mlngLockStart = 0
    Set mfrmBound = FormRef
    Set mlblStatus = LabelRef
    mlngTimerInterval = CheckInterval
    mlngTimeout = TimeoutPeriod
End Sub

Public Sub StartTimer()
    ' Započinjemo odbrojavanje; zapis je zaključan.

    mlngLockStart = timeGetTime()
    mfrmBound.TimerInterval = mlngTimerInterval * 1000
End Sub
```

```
Public Sub StopTimer()
    ' Prekidamo odbrojavanje jer zapis više nije zaključan.

    mlngLockStart = 0
    mfrmBound.TimerInterval = 0
    mlblStatus.Caption = ""
End Sub

Public Sub CheckTimer()
    ' Ova procedura ispituje stanje brojača vremena
    ' i preduzima odgovarajuće mere koje zavise od
    ' dužine isteklog vremena.

    Dim lngLockDuration  As Long ' Koliko dugo je zapis zaključan.
    Dim lngTimetoTimeout As Long ' Koliko je vremena preostalo.

    ' Ako je zapis izmenjen, izvršavamo jednu od sledećih akcija:
    ' 1. Poništavamo izmene (čime otključavamo zapis) ako je
    '    vreme isteklo.
    ' 2. Upozoravamo korisnika ako je isteklo 90%
    '    vremena.
    ' 3. Inače (ako je isteklo manje od 90% vremena), samo
    '    obaveštavamo korisnika koliko mu je vremena preostalo.
    If mlngLockStart > 0 Then
        ' Pošto izmene još nisu snimljene, ima još da se radi.
        lngLockDuration = _
         (timeGetTime() - mlngLockStart) / 1000
        lngTimetoTimeout = _
         mlngTimeout - lngLockDuration
        If lngLockDuration >= mlngTimeout Then
            ' Vreme je isteklo
            mfrmBound.Undo
            mlngLockStart = 0
            mfrmBound.TimerInterval = 0
            mlblStatus.ForeColor = vbRed
            mlblStatus.Caption = _
             "Vreme čekanja je isteklo! " & _
             " Izmene ovog zapisa koje" & _
             "niste snimili, izgubljene su."
    ElseIf lngLockDuration >= 0.9 * mlngTimeout _
         Then
            ' Upozoravajuća poruka
            mlblStatus.ForeColor = vbRed
            mlblStatus.Caption = _
             "Ovaj zapis ste predugo držali  " & _
             "zaključan. " & _
             "Ako izmene koje ste uneli ne snimite na disk" & _
             " u roku od " & lngTimetoTimeout & _
             " sekundi, biće izgubljene!"
        Else
```

```
            ' Obaveštenje o proteklom vremenu
            mlblStatus.ForeColor = vbBlack
            mlblStatus.Caption = _
             "Ovaj zapis držite zaključan " & _
             lngLockDuration & " sekundi. " & _
             "Preostaje vam još: " & _
             lngTimetoTimeout & " sekundi."
        End If
        ' Sledeća naredba je neophodna da bi
        ' obrazac ažurirao tekst natpisa.
        mfrmBound.Repaint
    Else
        ' Izmene su snimljene; prekidamo odbrojavanje vremena.
        Me.StopTimer
    End If
End Sub
```

Ugradnja vremenskog ograničenja u obrazac

Da biste klasu LockTimeout ugradili u jedan od svojih obrazaca, uradite sledeće:

1. U .MBD bazi podataka napravite vezan obrazac.
2. Iz baze podataka ch02app.mdb uvezite modul klase LockTimeout.
3. U zaglavlje ili u podnožje obrasca postavite objekat tipa natpis (Label). Trebalo bi da bude dovoljno širok da u celini prikazuje poruke o statusu koje će dobijati od klase LockTimeout. Ustanovili smo da odgovara natpis širine oko 4 inča (10 cm) i visine oko 0,3 inča (0,8 cm).
4. Dodajte objektnu promenljivu tipa LockTimeout koja će biti vidljiva na nivou modula. Na primer, u obrazac frmEmployee možete da uvedete sledeću objektnu promenljivu na nivou modula:

   ```
   Private mltoEmployee As LockTimeout
   ```

5. Napišite procedure za obradu sledećih događaja u obrascu: AfterUpdate, Dirty, Undo i Timer. U proceduri za obradu događaja AfterUpdate i Undo pozovite metodu StopTimer klase; u proceduri za obradu događaja Dirty pozovite metodu StartTimer klase, a u proceduri za obradu događaja Timer pozovite metodu CheckTimer klase. Na primer:

   ```
   Private Sub Form_AfterUpdate()
       mltoEmployee.StopTimer
   End Sub

   Private Sub Form_Undo()
       mltoEmployee.StopTimer
   End Sub

   Private Sub Form_Dirty(Cancel As Integer)
       mltoEmployee.StartTimer
   End Sub

   Private Sub Form_Timer()
       mltoEmployee.CheckTimer
   End Sub
   ```

6. Napišite proceduru za obradu događaja Load u obrascu. U toj proceduri treba da napravite primerak objekta LockTimeout i da pozovete njegovu metodu BindForm kojoj ćete proslediti sledeće parametre: referencu na obrazac, referencu na objekat tipa natpis koji će prikazivati poruke o statusu, interval (u sekundama) ažuriranja statusa i vreme (u sekundama) koje treba da istekne pre nego što se korisnikove izmene ponište. Na primer, sledeći kôd zadaje interval ažuriranja statusa od jedne sekunde i vreme čekanja od 10 minuta:

```
Private Sub Form_Load()
    Set mltoCustomer = New LockTimeout()
    mltoEmployee.BindForm _
        FormRef:=Me, LabelRef:=lblStatus, _
        CheckInterval:=1, TimeoutPeriod:=10 * 60
End Sub
```

NAPOMENA Bazi podataka ch02app.mdb dodata je verzija modula klase LockTimeout (pod imenom LockTimeout2000), koja ne zavisi od događaja Undo, zahvaljujući čemu radi i u Accessu 2000. Napravili smo i obrazac, frmCustomerPessimistic2 2000, koji ilustruje način upotrebe klase LockTimeout2000. Ovu verziju koda za vremensko ograničenje zaključavanja možete da upotrebite i u Accessu 2002. Pošto ovaj kôd koristi događaj Timer, videćete da je zbog toga nešto sporiji.

Zaključavanje i vrste objekata Recordset

U nekoliko narednih odeljaka razmatraju se pitanja u vezi sa zaključavanjem DAO i ADO objekata Recordset.

DAO objekti Recordset

Vrednost opcije Default Record Locking nema uticaja kada u kodu otvarate DAO objekte Recordset pozivanjem metode OpenRecordset. U tom slučaju Access primenjuje pesimističko zaključavanje (važi opcija Edited Record), osim ako:

A. Parametar Options metode OpenRecordset postavite na dbDenyWrite ili dbDenyRead

ili

B. Izmenite vrednost svojstva LockEdits.

Kada koristite opciju dbDenyWrite, pri ažuriranju zaključavate sve slogove. Možete ići i korak dalje, tj. da zabranite i menjanje i učitavanje sadržaja objekta Recordset, ali samo objekata tipa Table, tako što ćete zadati opciju dbDenyRead, kojom uvodite najviši stepen ograničenja pristupa podacima. (Ove opcije mogu da se primenjuju samo na Accessove standardne tabele.)

NAPOMENA Na objekte Recordset tipa Snapshot (snimak podataka) ne mogu se primenjivati nikakve vrste zaključavanja jer su oni već po svojoj prirodi takvi da se njihov sadržaj može samo čitati.

Pomoću svojstva LockEdits objekta Recordset možete da zadate način zaključavanja koji će se primenjivati. Standardna vrednost True čini da se primenjuje pesimističko (Edited Record) zaključavanje. Optimističko zaključavanje (No Locks) možete da zadate ako ovom svojstvu zadate vrednost False.

Na primer, procedura LockingPessDAO u modulu basRecordsetLocking baze podataka ch02app.mdb (videti listing 2.8) otvara objekat Recordset na osnovu tabele tblCustomer, pri čemu primenjuje pesimističko zaključavanje.

Listing 2.8

```
Sub LockingPessDAO()
    ' Pesimističko zaključavanje

    Dim db As DAO.Database
    Dim rstPessimistic As DAO.Recordset

    Stop

    Set db = CurrentDb()
    Set rstPessimistic = _
     db.OpenRecordset("tblCustomer", dbOpenDynaset)

    ' Sledeća naredba nalaže Accessu da primenjuje
    ' pesimističko zaključavanje u objektu Recordset
    ' rstPessimistic.
    rstPessimistic.LockEdits = True

    rstPessimistic.MoveFirst
    ' Zapis je zaključan između pozivanja
    ' metoda Edit i Update.
    rstPessimistic.Edit
        rstPessimistic!City = "Detroit"
    rstPessimistic.Update

    rstPessimistic.Close
    Set rstPessimistic = Nothing
    Set db = Nothing
End Sub
```

NAPOMENA

Procedura LockingPessDAO i druge procedure za zaključavanje podataka iz ovog poglavlja sadrže naredbu Stop. To omogućava da eksperimentišete sa zaključavanjem. Pokrenite proceduru; ona će se zaustaviti. Zatim otvorite drugu instancu Accessa i zaključajte prvi zapis tabele tblCustomer. Na primer, u drugoj instanci Accessa možete da otvorite obrazac frmCustomerPessimistic1 i da zatim izmenite sadržaj nekog njegovog polja. Vratite se zatim u proceduru LockingPessDAO i izvršavajte je korak po korak dok ne nastane greška zaključavanja. Broj te greške zavisiće od toga kako ste i u kom ste trenutku zaključali zapis u drugoj instanci Accessa.

Na sličan način, procedura LockingOptDAO u modulu basRecordsetLocking (videti listing 2.9) otvara objekat Recordset na osnovu tabele tblCustomer, ali se ovog puta primenjuje optimističko zaključavanje.

Listing 2.9

```
Sub LockingOptDAO()
    ' Optimističko zaključavanje

    Dim db As DAO.Database
    Dim rstOptimistic As DAO.Recordset

    Set db = CurrentDb()
    Set rstOptimistic = _
     db.OpenRecordset("tblCustomer", dbOpenDynaset)

    Stop

    ' Sledeća naredba nalaže Accessu da u objektu Recordset
    ' primenjuje optimističko zaključavanje
    ' rstOptimistic
    rstOptimistic.LockEdits = False

    rstOptimistic.MoveFirst
    rstOptimistic.Edit
        rstOptimistic!City = "Chicago"
    ' Zapis je zaključan od trenutka izdavanja
    ' naredbe Update do upisivanja podataka
    ' u bazu podataka.
    rstOptimistic.Update

    rstOptimistic.Close
    Set rstOptimistic = Nothing
    Set db = Nothing
End Sub
```

NAPOMENA Isto kao i procedura LockingPessDAO, procedura LockingOptDAO sadrži naredbu Stop. To omogućava da eksperimentišete sa zaključavanjem. Pokrenite proceduru; ona će se zaustaviti. Zatim otvorite drugu instancu Accessa i zaključajte prvi zapis tabele tblCustomer, ili izazovite sukob pri upisivanju tako što ćete pokušati da ažurirate sadržaj zapisa pošto to već započnete u prvoj instanci Accessa. Vratite se u proceduru LockingOptDAO i izvršavajte je korak po korak dok ne nastane greška zaključavanja.

DAO greške u višekorisničkom okruženju

Kada u višekorisničkom okruženju koristite nevezane objekte Recordset, neophodno je da presrećete i da obrađujete sve greške koje u takvom okruženju nastaju.

Najčešće među njima navedene su u tabeli 2.3. Greška broj 3197 ekvivalentna je grešci sukob pri upisivanju broj 7787, koja nastaje u vezanim obrascima.

Pošto pri zaključavanju na nivou zapisa nije poznato ime korisnika koji je zaključao zapis, Microsoft je dodao dve dodatne greške Accessu 2000 i novijim verzijama, broj 3202 i broj 3218. Te greške su ekvivalentne greškama broj 3186 i 3260, ali bez imena korisnika i mašine.

TABELA 2.3: VBA brojevi najčešćih grešaka u višekorisničkom okruženju.

Broj greške	Tekst poruke
3186	Upisivanje podataka nije uspelo; zapis je zaključao korisnik *imekorisnika* na mašini *imemašine*.
3187	Učitavanje podataka nije uspelo; zapis je zaključao korisnik *imekorisnika* na mašini *imemašine*.
3188	Ažuriranje podataka nije uspelo; zapis je zaključan u drugoj sesiji na istoj mašini.
3189	Tabelu *imetabele* je zaključao korisnik *imekorisnika* na mašini *imemašine*.
3197	Microsoftova mašina baze podataka Jet je prekinula postupak jer vi i još jedan korisnik pokušavate da izmenite isti podatak u isto vreme.
3202	Upisivanje podataka nije uspelo; drugi korisnik drži zapis zaključan.
3218	Ažuriranje podataka nije uspelo; drugi korisnik drži zapis zaključan.
3260	Ažuriranje podataka nije uspelo; zapis je zaključao korisnik *imekorisnika* na mašini *imemašine*.

DAO petlja za ponovno pokušavanje

Kada radite sa objektima Recordset, neophodno je da predvidite i obradite greške opisane u tabeli 2.3. Da biste obradili slučaj zaključanog zapisa, u svoju aplikaciju verovatno ćete ugraditi blok za obradu grešaka koji će sadržati određeni oblik petlje za ponovno pokušavanje kako biste više puta pokušali da pristupite zaključanom zapisu. Modul basRecordsetLocking u bazi ch02app.mdb sadrži petlju koja ponavlja pokušaje. Ta procedura, LockingPessDAO2, zajedno sa pripadajućim konstantama na nivou modula, prikazana je u listingu 2.10.

Listing 2.10

```
' Brojevi grešaka koje se pojavljuju
' u višekorisničkom okruženju
Const adhcLockErrCantSave1 = 3186
Const adhcLockErrCantSave2 = 3202
Const adhcLockErrCantRead = 3187
Const adhcLockErrExclusive = 3189
Const adhcLockErrDatChngd = 3197
Const adhcLockErrCantUpdate1 = 3188
Const adhcLockErrCantUpdate2 = 3260
Const adhcLockErrCantUpdate3 = 3218
```

```
' Broj ponovnih pokušaja
Const adhcLockRetries = 5
' Donja granica opsega vrednosti za
' interval čekanja između dva pokušaja.
Const adhcLockLBound = 2
' Gornja granica opsega vrednosti za
' interval čekanja između dva pokušaja.
Const adhcLockUBound = 10

Sub LockingPessDAO2()
   ' Pesimističko zaključavanje sa blokom
   ' za obradu grešaka

    On Error GoTo ProcErr

    Dim db As DAO.Database
    Dim rstPessimistic As DAO.Recordset
    Dim lngWait As Long
    Dim lngW As Long
    Dim intRetryCount As Integer
    Dim lngReturn As Long

    Randomize

    Set db = CurrentDb()
    Set rstPessimistic = _
     db.OpenRecordset("tblCustomer", dbOpenDynaset)

    Stop

    ' Sledeća naredba nalaže Accessu da primenjuje
    ' pesimističko zaključavanje u objektu Recordset
    ' rstPessimistic.
    rstPessimistic.LockEdits = True

    rstPessimistic.MoveFirst
    ' Zapis je zaključan između pozivanja
    ' metoda Edit i Update.
    Debug.Print "Pokušavam da zaključam zapis radi ažuriranja."
    rstPessimistic.Edit
        rstPessimistic!City = "Detroit"
    rstPessimistic.Update
    Debug.Print "Zapis je uspešno ažuriran!"

ProcEnd:
    On Error Resume Next
    rstPessimistic.Close
    Set rstPessimistic = Nothing
    Set db = Nothing
    Exit Sub
```

```
ProcErr:
    Select Case Err.Number
        Case adhcLockErrCantSave1, _
         adhcLockErrCantSave2, _
         adhcLockErrCantRead, _
         adhcLockErrCantUpdate1, _
         adhcLockErrCantUpdate2, _
         adhcLockErrCantUpdate3, _
         adhcLockErrExclusive
            intRetryCount = intRetryCount + 1
            If intRetryCount <= adhcLockRetries Then
                Debug.Print "Neuspešno zaključavanje, pokušaj br." & _
                 intRetryCount & "."
                ' Mašini Jet dajemo priliku
                ' da uhvati korak.
                dao.DBEngine.Idle
                ' Interval do narednog pokušaja izračunavamo
                ' na osnovu broja prethodnih pokušaja kome
                ' dodajemo nasumično odabran broj.
                lngWait = intRetryCount ^ 2 * _
                 Int((adhcLockUBound - adhcLockLBound + 1) _
                 * Rnd() + adhcLockLBound)
                ' Ovde se izvršava prazna petlja, a Windowsu
                ' omogućavamo da za to vreme obavlja druge poslove.
                For lngW = 1 To lngWait
                    DoEvents
                Next lngW
                Resume
            Else
                lngReturn = _
                 MsgBox("Procedura treba da ažurira " & _
                 "zapis koji je drugi korisnik zaključao. " & _
                 &vbCrLf & "Želite li da ponovo " & _
                 "pokušate da ažurirate zapis?" & _
                 vbCrLf & _
                 "Pritisnite Yes da biste ponovili pokušaj, " & _
                 "ili No da biste odustali.", _
                 vbYesNo + vbCritical, _
                 "Premašen maksimalan broj pokušaja")
                If lngReturn = vbYes Then
                    intRetryCount = 0
                    MsgBox _
                     "Procedura će ponovo pokušati " & _
                     "ažuriranje podataka. " & vbCrLf & _
                     "Napomena: Neće biti dodatne poruke " & _
                     "ukoliko ažuriranje bude " & _
                     "uspešno.", vbOKOnly + _
                      vbInformation, "Ponavljanje pokušaja ažuriranja"
                    Debug.Print "Brojač pokušaja postavljen na 0."
                    Resume
```

```
                Else
                    MsgBox "Ažuriranje zapisa nije uspelo!", _
                     vbOKOnly + vbCritical, "Ažuriranje prekinuto"
                    Debug.Print "Ažuriranje prekinuto."
                    Resume ProcEnd
                End If
            End If
        Case adhcLockErrDatChngd
            ' Sadržaj zapisa se promenio dok ga je korisnik ažurirao
            ' uz optimističko zaključavanje.
            MsgBox "Drugi korisnik je izmenio sadržaj zapisa " & _
             "s kojim ste radili." & _
             "Pritisnite OK da biste preuzeli " & _
             "izmene koje je on uneo." & _
             "Možete potom da unesete svoje izmene i da ih snimite.", _
             vbExclamation + vbOKOnly, "Sukob pri upisivanju"
            rstPessimistic.CancelUpdate
            Resume Next
        Case Else
            MsgBox "Greška broj " & Err.Number & ": " & _
             Err.Description, vbOKOnly + vbCritical, _
             "Nepredviđena greška"
            Resume ProcEnd
        End Select
End Sub
```

NAPOMENA

Procedura LockingPessDAO i druge procedure za zaključavanje podataka iz ovog poglavlja sadrže naredbu Stop. To omogućava da eksperimentišete sa zaključavanjem. Pokrenite proceduru; ona će se zaustaviti. Zatim otvorite drugu instancu Accessa i zaključajte prvi zapis tabele tblCustomer (za to možete da iskoristite obrazac frmCustomerPessimistic1). Vratite se zatim u proceduru LockingPessDAO2, otvorite prozor Debug, a zatim u VBA IDE okruženju izaberite Run Continue. Obratite pažnju na to da procedura upisuje u prozor Immediate poruke o tome da pokušava da zaključa zapis. Posle prve grupe neuspešnih pokušaja zaključavanja, zadajte da želite da pokušate ponovo. Onda oslobodite zaključan zapis i videćete da procedura uspešno zaključava zapis.

U bloku za obradu grešaka najpre ispitujemo da li se radi o jednoj od grešaka koje mogu da nastanu pri pokušaju ažuriranja zaključanog zapisa. Ako se dogodila jedna od njih, sadržaj promenljive intRetryCount povećavamo za jedan. Tu promenljivu koristimo kao brojač pokušaja:

```
Select Case Err.Number
    Case adhcLockErrCantSave1, _
     adhcLockErrCantSave2, _
     adhcLockErrCantRead, _
     adhcLockErrCantUpdate1, _
     adhcLockErrCantUpdate2, _
     adhcLockErrCantUpdate3, _
     adhcLockErrExclusive
        ntRetryCount = intRetryCount + 1
```

Dalje, ispitujemo da li je vrednost brojača intRetryCount manja od maksimalnog broja pokušaja (adhcLockRetries). Ako je tako, pozivamo metodu DAO.DBEngine.Idle da bismo Jetu omogućili da „uhvati korak“ i završi poslove koje nije stigao da obavi, poput oslobađanja ranije zaključanih zapisa koje nije imao vremena da uradi.

```
If intRetryCount <= adhcLockRetries Then
    Debug.Print "Neuspešno zaključavanje, pokušaj br." & _
    intRetryCount & "."
    ' Mašini Jet dajemo priliku da uhvati korak.
    dao.DBEngine.Idle
```

Možda ćemo pozivanjem metode DBEngine.Idle uspeti da oslobodimo zaključan zapis, ali je vrlo verovatno da ćemo morati da sačekamo nekoliko sekundi da to učini drugi korisnik ili proces. To je razlog zbog kojeg je u blok za obradu greške ugrađen „pametan“ kôd za čekanje. Verovatno ćete sličan kôd ugraditi u sve svoje procedure u kojima mogu da nastanu greške pri zaključavanju zapisa. Princip je sledeći: promenljivoj lngWait, tipa Long Integer, dodeljujemo vrednost pomoću formule koja najpre izračunava kvadrat prethodnog broja pokušaja, zatim rezultat množi s nasumično odabranom vrednošću koja se nalazi u opsegu između date donje i gornje granice. Zatim petlju For … Next izvršavamo lngWait puta da bismo sačekali određeno vreme. Naredba DoEvents u petlji omogućava Windowsu da obavi druge poslove (ako ih ima) dok čekamo. Na osnovu nasumično izabranog broja i kvadrata ukupnog broja prethodnih pokušaja, petlja pokušava da razdvoji dva korisnika koji istovremeno zahtevaju zaključavanje. Posle petlje For...Next, ponovni pokušaj se ostvaruje pomoću naredbe Resume.

Evo koda petlje za ponovno pokušavanje:

```
lngWait = intRetryCount ^ 2 * _
 Int((adhcLockUBound - adhcLockLBound + 1) _
 * Rnd() + adhcLockLBound)
 ' Ovde se izvršava prazna petlja, ali Windowsu
 ' omogućavamo da za to vreme obavlja druge
 ' poslove.
For lngW = 1 To lngWait
       DoEvents
Next lngW
Resume
```

Ako je premašen maksimalan broj pokušaja, blok za obradu grešaka obaveštava korisnika (u nekim situacijama možda to nećete želeti) i pruža mu mogućnost da pokuša ponovo ili da odustane:

```
Else
    lngReturn = _
     MsgBox("Procedura treba da ažurira " & _
     "zapis koji je drugi korisnik zaključao. " & _
     & vbCrLf & "Želite li da ponovo " & _
     "pokušate da ažurirate zapis?" & _
     vbCrLf & _
     "Pritisnite Yes da biste ponovili pokušaj, " & _
     "ili No da biste odustali.", _
     vbYesNo + vbCritical, _
     "Premašen maksimalan broj pokušaja")
```

```
        If lngReturn = vbYes Then
            intRetryCount = 0
            MsgBox _
             "Procedura će ponovo pokušati " & _
             "ažuriranje podataka." & vbCrLf & _
             "Napomena: Neće biti dodatne poruke " & _
             "ukoliko ažuriranje bude " & _
             "uspešno.", vbOKOnly + _
              vbInformation, "Ponavljanje pokušaja ažuriranja"
            Debug.Print "Brojač pokušaja postavljen na 0."
            Resume
        Else
            MsgBox "Ažuriranje zapisa nije uspelo!", _
             vbOKOnly + vbCritical, "Ažuriranje prekinuto"
            Debug.Print "Ažuriranje prekinuto."
            Resume ProcEnd
        End If
    End If
```

Blok za obradu grešaka sadrži i naredbu Case koja obrađuje sukobe pri upisivanju. Kôd ne radi ništa posebno, ali je kao primer dovoljan da shvatite osnovnu ideju:

```
Case adhcLockErrDatChngd
    ' sadržaj zapisa se promenio dok ga je korisnik ažurirao
    ' uz optimističko zaključavanje.
    MsgBox "Drugi korisnik je izmenio sadržaj " & _
     "zapisa s kojim ste radili." & _
     " Pritisnite OK da biste preuzeli " & _
     "izmene koje je on uneo." & _
     "Možete potom da unesete svoje izmene i da ih snimite.", _
     vbExclamation + vbOKOnly, "Sukob pri upisivanju"
    rstPessimistic.CancelUpdate
    Resume Next
```

ADO objekti Recordset

Ako drugačije ne zadate, ADO objekat Recordset se uvek otvara samo za čitanje. Međutim, ako svojstvu LockType objekta Recordset zadate odgovarajuću vrednost (pre pozivanja metode Open), ili ako koristite argument LockType metode Recordset.Open, možete da zadate vrstu zaključavanja koja će se koristiti.

Vrstu i trenutak zaključavanja možete da zadate pomoću jedne od sledećih konstanti:

Konstanta	Vrednost
adLockReadOnly	1
adLockPessimistic	2
adLockOptimistic	3
adLockBatchOptimistic	4

UPOZORENJE

ADO objekti Recordset otvoreni upotrebom vrednosti svojstva CurrentProject.Connection ne podržavaju pesimističko zaključavanje. Iako na prvi pogled izgleda kao da metoda Open ADO Recordset objekta, kada se koristi s tom vrstom veze, ipak podržava pesimističko zaključavanje, to nije tačno. Ovaj problem možete da zaobiđete tako što ćete umesto upotrebe svojstva Connection objekta CurrentProject, napraviti nov objekat tipa Connection.

Kôd prikazan u listingu 2.11 (procedura LockingPessADO u modulu basRecordsetLocking) pokazuje kako se otvara objekat Recordset s pesimističkim zaključavanjem. Obratite pažnju na to da u ovom primeru nismo koristili svojstvo CurrentProject.Connection jer ta posebna vrsta veze s bazom podataka ne omogućava pesimističko zaključavanje.

Listing 2.11

```
Sub LockingPessADO()
    ' Pesimističko zaključavanje

    Dim cnn As ADODB.Connection
    Dem rstPessimistic As ADODB.Recordset

    ' Da biste u Accessu mogli da primenjujete
    ' pesimističko zaključavanje, treba da otvorite novu
    ' vezu sa bazom podataka.
    ' CurrentProject.Connection ne podržava
    ' pesimističko zaključavanje!
    ' Imajte u vidu da svojstvo CurrentProject.BaseConnectionString
    ' sadrži parametre tekuće veze s bazom podataka.
    Set cnn = New ADODB.Connection
    cnn.Open CurrentProject.BaseConnectionString

    Set rstPessimistic = New ADODB.Recordset

     ' Otvaramo objekat Recordset tipa Keyset
     ' sa pesimističkim zaključavanjem
    rstPessimistic.Open "tblCustomer", cnn, adOpenKeyset, _
     adLockPessimistic, adCmdTable
    Stop

    rstPessimistic.MoveFirst
    ' Zapis se zaključava u trenutku
    ' kada je izmenjen.
    rstPessimistic!City = "Chicago"
    rstPessimistic.Update

    rstPessimistic.Close
    Set rstPessimistic = Nothing
    cnn.Close
    Set cnn = Nothing
End Sub
```

NAPOMENA

Procedura LockingPessADO, slično drugim procedurama za zaključavanje podataka u ovom poglavlju, sadrži naredbu Stop. To omogućava da eksperimentišete sa zaključavanjem. Pokrenite proceduru; ona će se zaustaviti. Zatim otvorite drugu instancu Accessa i zaključajte prvi zapis tabele tblCustomer (za to može da vam posluži obrazac frmCustomerPessimistic1). Vratite se zatim u proceduru LockingPessADO i izvršavajte je korak po korak dok ne nastane greška zaključavanja. Broj te greške zavisiće od toga kako ste i u kom ste trenutku zaključali zapis u drugoj instanci Accessa.

Listing 2.12 sadrži ekvivalentan primer, LockingOptADO, ali sa optimističkim zaključavanjem.

Listing 2.12

```
Sub LockingOptADO()
    ' Optimističko zaključavanje

    Dim cnn As ADODB.Connection
    Dim rstOptimistic As ADODB.Recordset

    Set cnn = CurrentProject.Connection
    Set rstOptimistic = New ADODB.Recordset

     ' Otvaramo objekat Recordset tipa Keyset
     ' sa optimističkim zaključavanjem
    rstOptimistic.Open "tblCustomer", cnn, adOpenKeyset, _
     adLockOptimistic, adCmdTable

    Stop

    rstOptimistic.MoveFirst
        rstOptimistic!City = "Chicago"
    ' Zapis se zaključava u trenutku upisivanja
    ' u bazu podataka.
    rstOptimistic.Update

    rstOptimistic.Close
    Set rstOptimistic = Nothing
    cnn.Close
    Set cnn = Nothing
End Sub
```

NAPOMENA

Ispitajte rad procedure LockingOptADO tako što ćete je izvršiti dok ne dođe do naredbe Stop. Otvorite zatim drugu instancu Accessa i zaključajte prvi zapis tabele tblCustomer ili izazovite sukob pri upisivanju tako što ćete u drugoj instanci pokušati da ažurirate zapis pošto u prvoj započnete njegovo ažuriranje. Vratite se zatim u proceduru LockingOptADO i izvršavajte je korak po korak dok ne nastane greška zaključavanja.

ADO petlja za ponovno pokušavanje

Slično kao u DAO modelu, kada radite sa ADO objektima Recordset, treba da predvidite i da obradite sve greške opisane u tabeli 2.3. To najčešće znači da treba da ugradite blok za obradu grešaka koji sadrži petlju za ponovno pokušavanje, isto kao što smo to uradili u DAO primeru. U bazi podataka ch02app.mdb nalazi se primer procedure s takvom petljom za ponovno pokušavanje. To je procedura LockingPessADO2, koja je prikazana u listingu 2.13.

Listing 2.13

```
' Brojevi grešaka koje se pojavljuju
' u višekorisničkom okruženju
Const adhcLockErrCantSave1 = 3186
Const adhcLockErrCantSave2 = 3202
Const adhcLockErrCantRead = 3187
Const adhcLockErrExclusive = 3189
Const adhcLockErrDatChngd = 3197
Const adhcLockErrCantUpdate1 = 3188
Const adhcLockErrCantUpdate2 = 3260
Const adhcLockErrCantUpdate3 = 3218

' Broj ponovnih pokušaja
Const adhcLockRetries = 5
' Donja granica opsega vrednosti za
' interval čekanja između dva pokušaja.
Const adhcLockLBound = 2
' Gornja granica opsega vrednosti za
' interval čekanja između dva pokušaja.
Const adhcLockUBound = 10

Sub LockingPessADO2()

   ' Pesimističko zaključavanje sa blokom
   ' za obradu grešaka

    On Error GoTo ProcErr

    Dim cnn As ADODB.Connection
    Dim rstPessimistic As ADODB.Recordset
    Dim lngWait As Long
    Dim lngW As Long
    Dim intRetryCount As Integer
    Dim lngReturn As Long

    Randomize

    ' Da biste u Accessu mogli da primenjujete
    ' pesimističko zaključavanje, treba da otvorite novu
    ' vezu s bazom podataka.
```

```
    ' CurrentProject.Connection ne podržava
    ' pesimističko zaključavanje!
    ' Imajte u vidu da svojstvo CurrentProject.BaseConnectionString
    ' sadrži parametre tekuće veze s bazom podataka.
    Set cnn = New ADODB.Connection
    cnn.Open CurrentProject.BaseConnectionString

    Set rstPessimistic = New ADODB.Recordset

     ' Otvaramo objekat Recordset tipa Keyset
     ' s pesimističkim zaključavanjem
    rstPessimistic.Open "tblCustomer", cnn, adOpenKeyset, _
     adLockPessimistic, adCmdTable

    Stop

    rstPessimistic.MoveFirst
        rstPessimistic!City = "Chicago"
    ' Zapis se zaključava u trenutku upisivanja u bazu podataka.
    rstPessimistic.Update
    Debug.Print "Zapis je uspešno ažuriran!"

ProcEnd:
    On Error Resume Next
    rstPessimistic.Close
    Set rstPessimistic = Nothing
    cnn.Close
    Set cnn = Nothing
    Exit Sub

ProcErr:
    ' Moramo da koristimo svojstvo SQLState
    ' da bismo dobili izvorni broj Jet greške.
    ' Pretpostavka: generiše se samo jedan
    ' objekat tipa Error.
    Select Case cnn.Errors(0).SQLState
        Case adhcLockErrCantSave1, _
         adhcLockErrCantSave2, _
         adhcLockErrCantRead, _
         adhcLockErrCantUpdate1, _
         adhcLockErrCantUpdate2, _
         adhcLockErrCantUpdate3, _
         adhcLockErrExclusive
            intRetryCount = intRetryCount + 1
            If intRetryCount <= adhcLockRetries Then
                Debug.Print "Greška pri zaključavanju, pokušaj br. " & _
                 intRetryCount & "."
                ' Windowsu i Jetu dajemo priliku da uhvate korak
                DoEvents
                ' Interval između dva pokušaja izračunavamo
                ' na osnovu broja prethodnih pokušaja kome
                ' dodajemo nasumično odabran broj
```

```
            lngWait = intRetryCount ^ 2 * _
             Int((adhcLockUBound - adhcLockLBound + 1) _
             * Rnd() + adhcLockLBound)
            ' Ovde se izvršava prazna petlja, a Windowsu
            ' omogućavamo da za to vreme obavlja druge
            ' poslove.
            For lngW = 1 To lngWait
                DoEvents
            Next lngW
            Resume
        Else
            lngReturn = _
             MsgBox("Procedura treba da ažurira " & _
              "zapis koji je drugi korisnik zaključao. " & _
              &vbCrLf & "Želite li da ponovo " & _
              "pokušate da ažurirate zapis?" & _
              vbCrLf & _
              "Pritisnite Yes da biste ponovili pokušaj, " & _
              "ili No da biste odustali.", _
              vbYesNo + vbCritical, _
              "Premašen maksimalan broj pokušaja")
            If lngReturn = vbYes Then
                intRetryCount = 0
                MsgBox _
                 "Procedura će ponovo pokušati " & _
                 "ažuriranje podataka. " & vbCrLf & _
                 "Napomena: Neće biti dodatne poruke " & _
                 "ukoliko ažuriranje bude " & _
                 "uspešno.", vbOKOnly + _
                  vbInformation, "Ponavljanje pokušaja ažuriranja"
                Debug.Print "Brojač pokušaja postavljen na 0."
                Resume
            Else
                MsgBox "Ažuriranje zapisa nije uspelo!", _
                 vbOKOnly + vbCritical, "Ažuriranje prekinuto"
                Debug.Print "Ažuriranje prekinuto."
                Resume ProcEnd
            End If
        End If
    Case adhcLockErrDatChngd
        ' Sadržaj zapisa se promenio dok ga je korisnik ažurirao
        ' uz pesimističko zaključavanje.
        MsgBox "Drugi korisnik je izmenio sadržaj zapisa " & _
         "s kojim ste radili." & _
         "Pritisnite OK da biste preuzeli " & _
         "izmene koje je on uneo." & _
         "Možete potom da unesete svoje izmene i da ih snimite.", _
         vbExclamation + vbOKOnly, "Sukob pri upisivanju"
        rstPessimistic.CancelUpdate
        Resume Next
    Case Else
```

```
            MsgBox "Greška broj " & cnn.Errors(0).SQLState & _
             ": " & cnn.Errors(0).Description, _
             vbOKOnly + vbCritical, "Nepredviđena greška"
            Resume ProcEnd
    End Select

End Sub
```

Blok za obradu grešaka u listingu 2.13 veoma je sličan DAO bloku za obradu grešaka u listingu 2.10. Više detalja o tome kako on radi i kako da ga testirate naći ćete u objašnjenju listinga 2.10.

SAVET

Kada greške koje su nastale u Jetu obrađujete u svom ADO kodu, treba da ispitujete vrednosti svojstva SQLState objekta Error da biste dobili podatak o tačnom broju greške. (Kao što je opisano u poglavlju 6, *Korišćenje ADO objekata sa podacima na serveru*, kada obrađujete greške koje su nastale u SQL Serveru, umesto svojstva SQLState, treba da ispitujete svojstvo NativeError objekta Error.)

Transakciona obrada

Transakciona obrada (engl. *transaction processing*) je izraz iz oblasti baza podataka koji znači grupisanje izmena sadržaja baze podataka u „paket" koji se potom obrađuje kao neraskidiva celina. „Paket" se uvek obrađuje tako da se uspešno izvrše ili sve *transakcije*, ili nijedna od njih. Na primer, kada u nekoj aplikaciji za bankarsko poslovanje premeštate iznos s jednog računa na drugi, ne biste želeli da stanje na jednom računu povećate, a da ga pri tom na drugom računu ne smanjite. Zato ćete te dve izmene grupisati u jednu transakciju.

Transakciona obrada je korisna u aplikacijama u kojima jedna akcija *mora* da se izvrši u sprezi s jednom ili više drugih akcija. Transakciona obrada je uobičajena u bankarskim, računovodstvenim i mnogim drugim aplikacijama.

DAO transakcije

DAO model podržava transakcionu obradu pomoću sledećih metoda objekta *Workspace*:

- BeginTrans
- CommitTrans
- Rollback

Pozivanjem metode *BeginTrans* označava se početak niza operacija koje čine jednu logičku jedinicu. Metoda *CommitTrans* preuzima sve izmene načinjene od poslednjeg mesta na kome je bila pozvana metoda BeginTrans i upisuje ih na disk. Metoda *Rollback* deluje na suprotan način od CommitTrans: ona poništava sve izmene i vraća stanje kakvo je bilo pre poslednjeg poziva metode CommitTrans. Najjednostavniji oblik DAO transakcione obrade najčešće je sličan sledećem:

```
On Error GoTo Err_Handler

Dim wrkCurrent As DAO.WorkSpace
Dim fInTrans As Boolean
```

```
fInTrans = False
Set wrkCurrent = DAO.DBEngine.Workspaces(0)
' ...
wrkCurrent.BeginTrans
    fInTrans = True
    ' (Izmene podataka počinju odavde)
wrkCurrent.CommitTrans
    fInTrans = False
' ...
Err_Handler:
    if fInTrans Then
        wrkCurrent.Rollback
    End If
    ' (Preostali deo bloka za obradu grešaka)
```

Kada koristite DAO transakcionu obradu, trebalo bi da vodite računa o sledećem:

- Transakcioni način obrade ne podržavaju sve vrste objekata Recordset. Ne podržava je nijedan ISAM format koji nije standardno ugrađen u Access, ali to čini većina ODBC izvora podataka. Da biste saznali da li određeni objekat Recordset podržava transakcionu obradu, ispitajte vrednost njegovog svojstva *Transactions*.
- Transakcije obuhvataju *sve* izmene podataka u radnom prostoru. Sve što izmenite posle pozivanja metode BeginTrans, prenosi se u bazu podataka ili poništava kao jedinstvena celina. To važi čak i za izmene načinjene u više baza podataka koje su otvorene u istom radnom prostoru.
- U Jet bazama podataka moguće je ugnežđivanje jedne transakcije u drugu do najviše pet nivoa dubine. Unutrašnja transakcija mora uvek da bude završena ili poništena pre one koja je okružuje. Ugnežđivanje transakcija sa ODBC tabelama nije moguće. Da biste obavili dve međusobno nezavisne transakcije, možete da koristite dva odvojena radna prostora (objekta tipa Workspace).
- Ako zatvorite radni prostor bez izričitog završavanja svih transakcija koje su u njemu započete, sve nezavršene transakcije automatski se poništavaju.

SAVET

Obrasci Accessa 2002 omogućavaju povezivanje objekta Recordset bilo kog tipa sa obrascem. Ova novina omogućava upotrebu DAO transakcija u vezanim obrascima.

ADO transakcije

ADO model podržava transakcionu obradu pomoću sledećih metoda objekta Connection:

- BeginTrans
- CommitTrans
- RollbackTrans

Pozivanjem metode *BeginTrans* označava se početak niza operacija koje čine jednu logičku jedinicu. Metoda *CommitTrans* preuzima sve izmene načinjene od poslednjeg mesta na kome je bila pozvana metoda BeginTrans i upisuje ih na disk.

Metoda *RollbackTrans* deluje na suprotan način od CommitTrans: ona poništava sve izmene i vraća stanje kakvo je bilo pre poslednjeg poziva metode CommitTrans. U svom najjednostavnijem obliku, ADO transakciona obrada veoma je slična DAO transakcionoj obradi:

```
On Error GoTo Err_Handler

Dim cnn As ADO.Connection
Dim fInTrans As Boolean

fInTrans = False
Set cnn = CurrentProject.Connection
' ...
cnn.BeginTrans
    fInTrans = True
    ' (Izmene podataka počinju odavde)
cnn.CommitTrans
fInTrans = False
' ...
Err_Handler:
    if fInTrans Then
        cnn.RollbackTrans
    End If
    ' (Preostali deo bloka za obradu grešaka)
```

Kao što vidite, kôd je vrlo sličan osim što u ADO modelu metode za transakcionu obradu pripadaju objektu Connection umesto objektu Workspace. Osim toga, DAO metoda Rollback u ADO modelu zove se RollbackTrans.

Transakciona obrada u višekorisničkom okruženju

Transakciona obrada je izuzetno važna u višekorisničkim aplikacijama. Kada više korisnika istovremeno unosi izmene u bazu podataka, više se ne možete pouzdati u to da će uvek jedna izmena biti trajno upisana u bazu pre nego što započnete narednu, osim ako određenu grupu izmena ne „uokvirite“ u transakciju. Zbog toga bi u višekorisničkom okruženju trebalo da koristite transakcionu obradu.

NAPOMENA

Kada koristite transakcije, povećavate stepen *integriteta* izmena koje korisnik unosi, ali smanjujete *konkurentnost*, odnosno mogućnost da pomoću vaše aplikacije veći broj korisnika istovremeno menja podatke jer ima više zapisa koji ostaju zaključani duže vreme nego bez transakcija. Posledica preterane upotrebe transakcija verovatno će biti povećan broj grešaka pri zaključavanju podataka u vašoj aplikaciji.

Kada koristite transakcije u višekorisničkom okruženju, Jet tretira svaku transakciju kao jednu operaciju pisanja na disk, pri čemu primenjuje način zaključavanja koji ste zadali. To znači da ako ste zadali pesimističko zaključavanje u DAO objektu Recordset, Jet zaključava zapis kada naiđe na metodu Edit. Kada se koristi ADO, Jet zaključava zapis kada izmenite vrednost nekog od polja objekta Recordset. Ako umesto pesimističkog koristite optimističko zaključavanje, Jet zaključava zapis kada naiđe na metodu Update; to važi za oba modela, DAO i ADO. Međutim,

u okviru jedne transakcije, Jet kumulira zaključavanja zapisa, koje ne oslobađa dok transakcija ne bude završena ili poništena.

Ako Jet unutar transakcije naiđe na zaključan zapis, nastaje standardni skup presretljivih grešaka. Kada se to dogodi, treba da pokušate da zapis uspešno zaključate ili da poništite transakciju.

Implicitne transakcije

Osim *izričitih* (*eksplicitnih*) transakcija koje formirate metodama BeginTrans, CommitTrans i Rollback, Jet 4 (i njegovi prethodnici od Jeta 3) formira i *implicitne* transakcije da bi poboljšao performanse pri radu sa objektima Recordset. U bazi podataka koju je korisnik otvorio za isključivu upotrebu, Jet standardno završava implicitne transakcije svake 2 sekunde; kada je baza podataka otvorena za deljenu upotrebu, Jet završava te transakcije svakih 50 milisekundi. U višekorisničkom okruženju ne bi trebalo da standardna vrednost od 50 milisekundi bude uzrok primetnog smanjenja konkurentnosti.

Tačan učinak koji implicitne transakcije imaju na konkurentnost zavisi od vrednosti više parametara u registru. Ti parametri opisani su u tabeli 2.4. Da biste zadali drugačije vrednosti, treba da ih upišete u parametre registra koji se nalaze u sledećoj grani:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Jet\4.0\Engines\Jet 4.0.
```

TABELA 2.4: Parametri registra koji se odnose na transakcije

Ključ	Tip podatka	Učinak	Podrazumevana vrednost
UserCommitSync	String	Određuje da li se eksplicitne transakcije izvršavaju sinhrono (neugnežđene transakcije se izvršavaju jedna za drugom).	Yes
ImplicitCommitSync	String	Određuje da li će Jet započinjati implicitne transakcije prilikom ažuriranja objekta Recordset. Vrednost No nalaže Jetu da koristi implicitne transakcije.	No
ExclusiveAsyncDelay	DWORD	Određuje maksimalnu dužinu intervala (u milisekundama) pre nego što Jet završi implicitnu transakciju kada je baza podataka otvorena u isključivom režimu.	2000
SharedAsyncDelay	DWORD	Određuje maksimalnu dužinu intervala (u milisekundama) pre nego što Jet završi implicitnu transakciju kada je baza podataka otvorena u deljenom režimu.	50

U nekim aplikacijama ćete namerno žrtvovati konkurentnost u korist boljih performansi. U tim slučajevima može biti korisno da povećate vrednost ključa SharedAsyncDelay.

SAVET

Jetove parametre u registru možete privremeno da zamenite drugim vrednostima ako koristite metodu SetOption DAO objekta DBEngine, ili svojstva ADO objekta Connection.

Akcioni upiti i transakcije u višekorisničkom okruženju

Accessov korisnički interfejs (UI), kao i DAO i ADO modeli omogućavaju upravljanje ponašanjem transakcija pri izvršavanju akcionih upita.

Svojstvo UseTransaction

U prozoru svojstava akcionog upita, koji ste snimili u bazu podataka, možete da postavite vrednost svojstva UseTransaction na No da biste sprečili Access da akcioni upit izvrši unutar jedne transakcije. Podrazumevana vrednost ovog svojstva je Yes. Ona nalaže Accessu da svaki akcioni upit izvršava u okviru jedne transakcije. Kada svojstvo UseTransaction ima vrednost No, mogu se poboljšati performanse akcionih upita koji deluju na veliki broj zapisa, ali zato u slučaju greške nije moguće poništavanje transakcije.

Ponašanje pri transakcionoj obradi možete da izmenite i za DAO snimljene upite (QueryDef), ali pošto svojstvo UseTransaction nije standardno Jetovo svojstvo, morate ga najpre dodati zbirci svojstava objekta QueryDef pomoću koda sličnog sledećem:

```
' Upit ne treba umetati u transakciju.
Set prp = qdf.CreateProperty("UseTransaction", _
 dbBoolean, False)
qdf.Properties.Append prp
```

Slično tome, u ADO modelu zadajete da se ne koristi transakcija pri izvršavanju akcionog upita tako što svojstvu Jet OLEDB:Bulk Transactions određenog ADO objekta tipa Command dodelite vrednost 1 (koja znači da se transakcije *ne* koriste; vrednost 2 nalaže Jetu da koristi transakcije):

```
' Upit ne treba umetati u transakciju.
cmd.Properties("JET OLEDB:Bulk Transactions") = 1
```

SAVET

Možete da upotrebite i globalno svojstvo koje se odnosi na sve grupne operacije koje se obavljaju putem određenog objekta Connection. Da biste to uradili, treba da zadate odgovarajuću vrednost svojstvu Jet OLEDB:Global Bulk Transactions ADO objekta Connection.

Svojstvo FailOnError

Upiti za ažuriranje (Update) i brisanje (Delete) podataka imaju svojstvo FailOnError, koje je povezano sa svojstvom UseTransaction. Standardno ponašanje Accessa je takvo da, kada iz njegovog korisničkog interfejsa pokrenete akcioni upit, on dozvoljava delimično izvršavanje upita posle upozorenja korisniku da ažuriranje određenih zapisa neće uspeti. Međutim, ako vrednost svojstva FailOnError podesite na Yes, Access neće dozvoliti da se upit za ažuriranje ili brisanje izvrši samo delimično.

Kada koristite DAO model, možete da izmenite svojstvo FailOnError objekta Recordset, ili da argumentu Options metode QueryDef.Execute dodelite vrednost dbFailOnError, kao u sledećem primeru:

```
' Upit se ne izvršava ako postoje zapisi koji se
' ne mogu ažurirati/brisati.
qdf.Execute dbFailOnError
```

Listing 2.14 prikazuje primer upotrebe opcije dbFailOnError s metodom Execute objekta QueryDef.

Listing 2.14

```
Function DeleteTempOrdersDAO()

    ' Primer upotrebe opcije dbFailOnError
    ' i svojstva RecordsAffected akcionih upita.
    '
    On Error GoTo ProcErr

    Dim db As DAO.Database
    Dim wrk As DAO.Workspace
    Dim qdf As DAO.QueryDef
    Dim lngRecsEstimated As Long
    Dim lngRecsAffected As Long
    Dim intResp As Integer
    Dim strWhere As String
    Dim fInTrans As Boolean

    DeleteTempOrdersDAO = False
    fInTrans = False

    Set db = CurrentDb
    Set wrk = dao.DBEngine.Workspaces(0)

    ' Odredba Where upita
    strWhere = "[OrderId] BETWEEN 1 AND 100"

    ' Utvrđujemo ukupan broj zapisa koje treba izbrisati.

    lngRecsEstimated = _
     DCount("*", "tblTempOrders", strWhere)
    Set qdf = db.CreateQueryDef("", _
     "DELETE * FROM tblTempOrders WHERE " & strWhere)

    ' Odustati u slučaju greške?
    intResp = MsgBox("Pronađeno je " & _
     lngRecsEstimated & " zapisa za brisanje." & vbCrLf & _
     "Odustati ako brisanje nekog od njih ne bude moguće?", _
     vbYesNo + vbInformation + vbDefaultButton1, _
     "Primer akcionog upita")

    wrk.BeginTrans
        fInTrans = True
        If intResp = vbYes Then
            qdf.Execute dbFailOnError
        Else
            qdf.Execute
        End If
```

```
        ' Ovde utvrđujemo ukupan broj zapisa koji
        ' su zaista bili izbrisani.
        lngRecsAffected = qdf.RecordsAffected
        If lngRecsAffected < lngRecsEstimated Then
            intResp = MsgBox("Samo " & lngRecsAffected & _
             " od " & lngRecsEstimated & _
             " zapisa može da se briše." & vbCrLf & _
             "Je li to prihvatljivo?", _
             vbOKCancel + vbInformation, _
             "Primer akcionog upita")
            ' Poništavamo transakciju ako to korisnik zahteva.
            If intResp = vbCancel Then
                MsgBox "Transakcija poništena!", _
                 vbOKOnly + vbCritical, _
                 "Primer akcionog upita"
                wrk.Rollback
                GoTo ProcDone
            End If
        End If
    wrk.CommitTrans
    fInTrans = False

    DeleteTempOrdersDAO = True

ProcDone:
    qdf.Close
    Set qdf = Nothing
    wrk.Close
    Set wrk = Nothing
    Exit Function

ProcErr:
    If fInTrans Then
        wrk.Rollback
        MsgBox "Došlo je do greške. Brisanje poništeno.", _
         vbOKOnly + vbCritical, "Primer akcionog upita"
    Else
        MsgBox "Greška broj " & Err.Number & ": " & _
         Err.Description, _
         vbOKOnly + vbCritical, "Primer akcionog upita"
    End If
    Resume ProcDone

End Function
```

Kada koristite ADO model, možete postići jednak efekat tako što ćete svojstvu Jet OLEDB:Partial Bulk Ops ADO objekta Command dodeliti vrednost 2 (koja znači da nije dozvoljeno delimično ažuriranje; vrednost 1 znači da je to dozvoljeno):

```
' Upit se ne izvršava ako postoje zapisi koji se
' ne mogu ažurirati/brisati.
cmd.Properties("JET OLEDB:Partial Bulk Ops") = 2
```

SAVET

Možete da koristite i globalno svojstvo koje se odnosi na sve grupne operacije koje se obavljaju putem određenog objekta Connection. Da biste to uradili, treba da zadate odgovarajuću vrednost svojstvu Jet OLEDB:Global Partial Bulk Ops ADO objekta Command.

Listing 2.15 prikazuje primer upotrebe svojstva Jet OLEDB:Global Partial Bulk Ops ADO objekta Command.

Listing 2.15

```
Function DeleteTempOrdersADO()

    ' Primer upotrebe svojstva "Jet OLEDB:Partial Bulk Ops"
    ' ADO objekta Command.
    '
     On Error GoTo ProcErr

    Dim cnn As ADODB.Connection
    Dim cmd As ADODB.Command
    Dim lngRecsEstimated As Long
    Dim lngRecsAffected As Long
    Dim intResp As Integer
    Dim strWhere As String
    Dim fInTrans As Boolean

    DeleteTempOrdersADO = False
    fInTrans = False

    Set cnn = CurrentProject.Connection

    ' Opcija Where odredbe
    strWhere = "[OrderId] BETWEEN 1 AND 100"

    ' Utvrđujemo broj zapisa koje treba izbrisati.

    lngRecsEstimated = _
     DCount("*", "tblTempOrders", strWhere)
    Set cmd = New ADODB.Command
    cmd.ActiveConnection = cnn
    cmd.CommandText = _
     "DELETE * FROM tblTempOrders WHERE " & strWhere
    cmd.CommandType = adCmdText

    ' Odustajemo u slučaju greške?
    intResp = MsgBox("Pronađeno je " & _
     lngRecsEstimated & " zapisa za brisanje." & vbCrLf & _
     " Odustati ako brisanje nekog od njih ne bude moguće?", _
     vbYesNo + vbInformation + vbDefaultButton1, _
     "Action Query Example")
```

```
    cnn.BeginTrans
        fInTrans = True
        If intResp = vbYes Then
            cmd.Properties("JET OLEDB:Partial Bulk Ops") = 2
        Else
            cmd.Properties("JET OLEDB:Partial Bulk Ops") = 1
        End If
        cmd.Execute lngRecsAffected
        If lngRecsAffected < lngRecsEstimated Then
            intResp = MsgBox("Samo " & lngRecsAffected & _
             " od " & lngRecsEstimated & _
             " zapisa može da se briše." & vbCrLf & _
             "Je li to prihvatljivo?", _
             vbOKCancel + vbInformation, _
             "Primer akcionog upita")
            ' Poništavamo transakciju.
            If intResp = vbCancel Then
                MsgBox "Transakcija poništena!", _
                 vbOKOnly + vbCritical, _
                 "Primer akcionog upita"
                cnn.RollbackTrans
                GoTo ProcDone
            End If
        End If
    cnn.CommitTrans
    fInTrans = False

    DeleteTempOrdersADO = True

ProcDone:
    Set cmd = Nothing
    cnn.Close
    Set cnn = Nothing
    Exit Function

ProcErr:
    If fInTrans Then
        cnn.RollbackTrans
        MsgBox "Došlo je do greške. Brisanje poništeno.", _
         vbOKOnly + vbCritical, "Primer akcionog upita"
    Else
        MsgBox "Greška broj" & cnn.Errors(0).SQLState & ": " & _
         cnn.Errors(0).Description, _
         vbOKOnly + vbCritical, "Primer akcionog upita"
    End If
    Resume ProcDone

End Function
```

Upotreba namenske procedure za generisanje vrednosti u polju tipa AutoNumber

Mašina baze podataka Jet pruža visok stepen prilagodljivosti pri izboru polja tipa AutoNumber (ovaj tip polja zvao se Counter u verzijama pre Accessa 95). Polje tipa AutoNumber možete da podesite tako da sadrži vrednosti tipa Long Integer koje rastu sekvencijalno, ili koje se generišu nasumično, a možete da koristite i 16-bitni jedinstveni identifikator (GUID). (Više informacija o GUID identifikatorima naći ćete u poglavlju 9.) Međutim, ponekad ćete radije koristiti namensku proceduru za generisanje vrednosti koje rastu po određenom algoritmu. Razlozi za to mogu biti sledeći:

- Neophodan vam je alfanumerički znakovni niz.
- Korak povećanja mora da bude veći od 1. (Iako Accessov korisnički interfejs, DAO i ADOX ne omogućavaju da zadate korak povećanja različit od 1, možete da napravite takvo AutoNumber polje ako upotrebite SQL DDL upit sa iskazom CREATE TABLE.)
- Želite da ponovo upotrebite vrednosti iz odbačenih zapisa.
- Želite da izračunate vrednosti na osnovu sadržaja drugih polja zapisa.
- Imate podatke koji se nalaze u nestandardnoj tabeli koja ne podržava polja tipa AutoNumber.
- Replikujete bazu podataka, ali ne želite da koristite nasumično generisane AutoNumber vrednosti.

SAVET

Važno je da se ne pouzdate previše u prividno jedinstvene primarne ključeve, kao što su oni koji se mogu generisati u poljima tipa AutoNumber (pomoću Accessovog standardnog algoritma ili onog koji sami napišete). Iako ta polja mogu da garantuju jedinstvenost zapisa, morate obezbediti, pomoću dodatnih indeksa ili procedura za obradu događaja da korisnik ne može upisati, na primer, pet zapisa o istom korisniku.

Namenski algoritam za polja tipa AutoNumber

Svoj algoritam za generisanje vrednosti u poljima tipa AutoNumber u Accessovoj bazi podataka možete da implementirate pomoću zasebne tabele u kojoj se čuva naredna AutoNumber vrednost. Tu tabelu morate da zaključavate kada učitavate novu AutoNumber vrednost, kao i da presrećete greške koje mogu da nastanu kada više korisnika istovremeno pokušava da učita novu vrednost iz te tabele.

Napisali smo namensku proceduru za generisanje AutoNumber vrednosti za polje MenuID tabele tblMenu u bazi podataka ch02app.mdb. Procedura, u kojoj se koristi DAO model, poziva se iz obrasca frmMenu; tabela u kojoj se čuva AutoNumber vrednost nalazi se u bazi podataka ch02auto.mdb.

NAPOMENA

Ova procedura može da se implementira i pomoću ADO modela.

Implementiranje namenskog algoritma za generisanje vrednosti za polja tipa AutoNumber

Postupak generisanja nove vrednosti za polje tipa AutoNumber podelili smo na dva dela. Procedura nižeg nivoa obrađuje AutoNumber korak povećanja ili daje -1 ukoliko nova AutoNumber vrednost ne može da se generiše. Druga procedura višeg nivoa dodeljuje AutoNumber vrednost novom zapisu i određuje šta će biti urađeno ako se pojavi greška. Obe procedure nalaze se u modulu basAutoNumber u bazi podataka ch02app.mdb.

Funkcija adhGetNextAutoNumber je procedura nižeg nivoa koja direktno komunicira s bazom podataka u kojoj se nalazi tabela koja sadrži sledeću AutoNumber vrednost. Ime te tabele je parametar procedure. Naredna AutoNumber vrednost čuva se u tabeli (u bazi podataka na koju upućuje konstanta adhcAutoNumDb) čije se ime sastoji od imena ciljne tabele i sufiksa _ID. Na primer, naredna AutoNumber vrednost za tabelu tblMenu čuva se u tabeli tblMenu_ID. Ova tabela ima vrlo jednostavnu strukturu: sastoji se od samo jednog polja tipa Long Integer, koje se zove NextAutoNumber. Funkcija adhGetNextAutoNumber prikazana je u listingu 2.16.

Listing 2.16

```
' Baza podataka u kojoj se čuva naredna AutoNumber vrednost.
Const adhcAutoNumDb = "Ch02Auto.Mdb"

' Broj ponavljanja pokušaja zaključavanja
Const adhcLockRetries = 5
' Donja granica opsega vrednosti za
' interval čekanja između dva pokušaja.
Const adhcLockLBound = 2
' Gornja granica opsega vrednosti za
' interval čekanja između dva pokušaja.
Const adhcLockUBound = 10

' Konstante za brojeve grešaka
Const adhcErrRI = 3000
Const adhcLockErrCantUpdate2 = 3260
Const adhcLockErrTableInUse = 3262

Function adhGetNextAutoNumber(ByVal strTableName _
 As String) As Long

    ' Daje narednu generisanu AutoNumber vrednost
    ' za zadatu tabelu. Te vrednosti se čuvaju u bazi
    ' podataka na koju upućuje konstanta
    ' adhcAutoNumDb, u tabelama čija se imena sastoje
    ' od imena tabela za koje generišu AutoNumber vrednosti
    ' i od sufiksa _ID.
    ' Povratna vrednost funkcije je -1 ukoliko zbog problema
    ' zaključavanja nije moguće generisanje AutoNumber
    ' vrednosti.
```

```
    On Error GoTo adhGetNextAutoNumber_Err

    Dim wrk As DAO.Workspace
    Dim db As DAO.Database
    Dim rstAutoNum As DAO.Recordset
    Dim lngNextAutoNum As Long
    Dim lngW As Long
    Dim lngX As Long
    Dim intRetryCount As Integer

    Randomize
    DoCmd.Hourglass True
    intRetryCount = 0

    ' Otvaramo objekat Recordset na osnovu odgovarajuće tabele u
    ' bazi podataka sa AutoNumber vrednostima. Dok je ona otvorena,
    ' zabranjujemo čitanje svim ostalim korisnicima.
    Set wrk = DAO.DBEngine.Workspaces(0)
    Set db = wrk.OpenDatabase(adhCurrentDBPath() & _
     adhcAutoNumDb, False)
    Set rstAutoNum = db.OpenRecordset(strTableName _
     & "_ID", dbOpenTable, dbDenyRead)

    ' Povećavamo postojeću AutoNumber vrednost za jedan.
    ' To je i povratna vrednost funkcije.
    rstAutoNum.MoveFirst
    rstAutoNum.Edit
        lngNextAutoNum = rstAutoNum![NextAutoNumber]
        rstAutoNum![NextAutoNumber] = lngNextAutoNum + 1
    rstAutoNum.Update

    adhGetNextAutoNumber = lngNextAutoNum

adhGetNextAutoNumber_Exit:
    DoCmd.Hourglass False
    On Error Resume Next
    rstAutoNum.Close
    Set rstAutoNum = Nothing
    db.Close
    Set db = Nothing
    wrk.Close
    Set wrk = Nothing
    Exit Function

adhGetNextAutoNumber_Err:
    Select Case Err.Number
    Case adhcErrRI, _
     adhcLockErrCantUpdate2, _
     adhcLockErrTableInUse
        ' Tabelu je zaključao drugi korisnik
        intRetryCount = intRetryCount + 1
        ' Broj pokušaja premašio maksimum, odustajemo.
```

```
        If intRetryCount > adhcLockRetries Then
            adhGetNextAutoNumber = -1
            Resume adhGetNextAutoNumber_Exit
        Else
            ' Omogućavamo Windowsu i Jetu da uhvate korak.
            DAO.DBEngine.Idle
            ' Interval između dva pokušaja određujemo
            ' na osnovu broja prethodnih pokušaja kome
            ' dodajemo nasumično odabran broj.
            lngW = intRetryCount ^ 2 * _
             Int((adhcLockUBound - adhcLockLBound + 1) _
              * Rnd() + adhcLockLBound)
            ' Ovde se izvršava prazna petlja, a Windowsu
            ' omogućavamo da za to vreme obavlja druge poslove.
            For lngW = 1 To lngW
                DoEvents
            Next lngW
            Resume
        End If
        Case Else
            ' Nepredviđena greška
            MsgBox "Greška broj " & Err.Number & ": " _
             & Err.Description, _
             vbOKOnly + vbCritical, "adhGetNextAutoNumber"
            adhGetNextAutoNumber = -1
            Resume adhGetNextAutoNumber_Exit
    End Select

End Function
```

Ukoliko nema sukoba oko AutoNumber vrednosti, posao funkcije adhGetNextAutoNumber svodi se na učitavanje i povećavanje vrednosti u polju NextAutoNumber. Međutim, u višekorisničkom okruženju može se dogoditi da jedan korisnik zahteva AutoNumber vrednost iz tabele koju ju je drugi korisnik već zaključao. Funkcija adhGetNextAutoNumber obrađuje tu vrstu grešaka pomoću petlje za ponovno pokušavanje slične onoj koja je opisana u odeljku „DAO petlja za ponovno pokušavanje“, u prethodnom delu ovog poglavlja.

Funkcija adhAssignID, koja se takođe nalazi u modulu basAutoNumbers (prikazana je u listingu 2.17), dodeljuje AutoNumber vrednosti na višem nivou. Ona se poziva u proceduri za obradu događaja BeforeInsert u obrascu, pri čemu joj se prosleđuju tri parametra: referenca na objekat tipa Form, ime tabele u kojoj se čuva generisana AutoNumber vrednost i ime polja u koje se upisuje ta vrednost. Funkcija adhAssignID odbacuje svaki zapis kome se ne može dodeliti AutoNumber vrednost. U zavisnosti od problema koji rešavate, može vam biti neophodna složenija procedura za kontrolu na višem nivou. Na primer, umesto da odbacujete zapise kojima ne možete da dodelite vrednost, možete da ih smeštate u privremenu lokalnu tabelu.

Listing 2.17

```
Function adhAssignID(frm As Form, ByVal _
 strTableName As String, _
 ByVal strAutoNumField As String) As Variant

    ' Poziva se iz događaja BeforeInsert obrasca radi
    ' dodeljivanja nove AutoNumber vrednosti polju.
    ' Odbacuje svaki zapis koji ne može da bude upisan.

    On Error GoTo adhAssignID_Err

    Dim lngNewID As Long

    lngNewID = adhGetNextAutoNumber(strTableName)
    If lngNewID <> -1 Then
        frm(strAutoNumField) = lngNewID
    Else
        MsgBox "Dodavanje zapisa nije moguće jer se ne može " & _
        "dodeliti AutoNumber vrednost", vbOKOnly + vbCritical, _
        "Greška pri upisivanju zapisa"
        frm.Undo
    End If

adhAssignID_Exit:
    Exit Function

adhAssignID_Err:
    frm.Undo
    MsgBox "Greška broj " & Err.Number & ": " & Err.Description, _
     vbOKOnly + vbCritical, "adhAssignID"
    Resume adhAssignID_Exit

End Function
```

Funkciju adhGetNextAutoNumber možete da izmenite tako da generiše vrednosti za AutoNumber polja u nekom posebnom formatu sa korakom povećanja koji je različit od 1, ili u alfanumeričkom formatu.

Liste korisnika i upravljanje vezom

Mašina Jet 4, uvedena prvi put sa Accessom 2000, pruža dve nove mogućnosti za višekorisničko okruženje pomoću kojih se efikasnije upravlja korisnicima. To su lista korisnika i upravljanje vezom.

Lista korisnika

Kada koristite ADO model, možete da napravite objekat Recordset specifičan za određeni izvor podataka (pomoću metode OpenSchema objekta Connection) koji pruža podatke o tome koji su korisnici trenutno aktivni u bazi podataka. Da biste

to uradili, metodi OpenSchema treba da prosledite „magični“ GUID identifikator. Taj identifikator, čija je vrednost {947bb102-5d43-11d1-bdbf-00c04fb92675}, nema nikakvo značenje, osim što je jedini koji vam omogućava da od Jeta dobijete podatke za listu korisnika. (Pomoću metode OpenSchema možete da formirate objekat Recordset sa raznim vrstama korisnih podataka o šemi baze podataka. Ovu metodu koristili smo u prethodnom delu poglavlja, u odeljku „Upravljanje pridruženim tabelama“.)

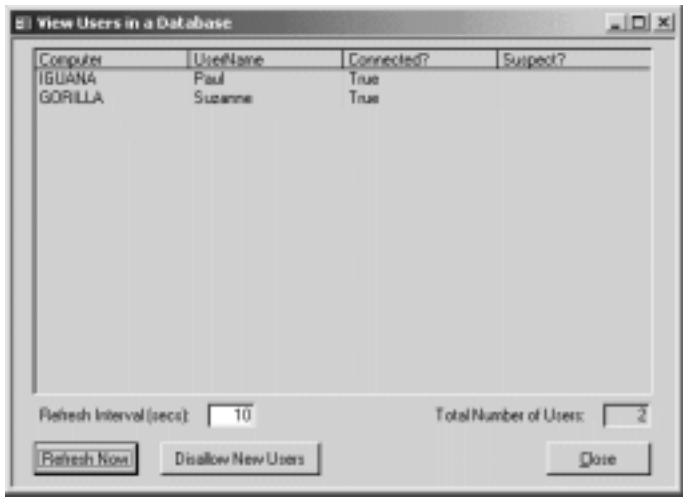
Programski kôd u listingu 2.18, koji ćete naći u obrascu frmViewUsers baze podataka ch02app.mdb, pokazuje kako se upotrebljava lista korisnika.

Listing 2.18

```
' Da biste od šeme baze podataka dobili listu korisnika,
' treba da zadate ovaj magični broj. Zbog čega nije definisana
' konstanta za njega? Ostaće zagonetka...
Const adhcUsers = "{947bb102-5d43-11d1-bdbf-00c04fb92675}"

Sub BuildUserList()

    ' Formira listu tekućih korisnika baze podataka
    ' pomoću metode OpenSchema objekta Connection.

    Dim cnn As ADODB.Connection
    Dim rst As ADODB.Recordset
    Dim fld As ADODB.Field
    Dim intUser As Integer
    Dim strUser As String
    Dim varVal As Variant

    ' Zaglavlja kolona
    strUser = "Computer;UserName;Connected?;Suspect?"

    Set cnn = CurrentProject.Connection

    Set rst = cnn.OpenSchema( _
     Schema:=adSchemaProviderSpecific, _
     SchemaId:=adhcUsers)

    With rst
        Do Until .EOF
            intUser = intUser + 1
            For Each fld In .Fields
                varVal = fld.Value
                ' Pošto su neke od dobijenih vrednosti znakovni nizovi
                ' koji se završavaju znakom Null, uklanjamo te znakove.
```

```
                If InStr(varVal, vbNullChar) > 0 Then
                    varVal = Left(varVal, _
                     InStr(varVal, vbNullChar) - 1)
                End If
                strUser = strUser & ";" & varVal
            Next
            .MoveNext
        Loop
    End With
    txtUsers = intUser
    lboUsers.RowSource = strUser

    ' Završno čišćenje
    rst.Close
    Set rst = Nothing
    Set fld = Nothing
    Set cnn = Nothing
End Sub
```

Kada joj prosledite odgovarajući parametar SchemaID, metoda OpenSchema puni rezultujući objekat Recordset podacima o korisnicima. Međutim, pošto se neke od vrednosti koje metoda OpenSchema daje završavaju znakom Null, procedura BuildUserList odseca sve što se nalazi iza tog znaka pre nego što formira znakovni niz kojim zatim inicijalizuje objekat tipa ListBox na obrascu frmViewUsers (tako što popunjava njegovo svojstvo RowSource). Primer liste sa dva korisnika koja tako nastaje prikazan je na slici 2.9.



SLIKA 2.9
Pomoću metode OpenSchema obrazac frmViewUsers prikazuje listu korisnika koji upravo koriste bazu podataka.

Upravljanje vezom

Jet 4 pruža mogućnost zvanu *upravljanje vezom* (koja je poznata i kao *pasivno blokiranje*) pomoću koje možete da sprečite prijavljivanje korisnika u bazu podataka. To je korisno kada želite da ograničite broj istovremenih korisnika baze podataka ili kada želite da obavite neki administrativni posao, kao što je pravljenje rezervnih kopija baze podataka. Imajte u vidu da je to pasivna mogućnost, što znači da ne postoji način da iz baze podataka prisilno isključite korisnike koji su upravo aktivni.

Vezom upravljate pomoću svojstva Jet OLEDB:Connection Control ADO objekta Connection. Da biste sprečili prijavljivanje novih korisnika u bazu podataka, ovom svojstvu zadajte vrednost 1. Da biste ponovo dozvolili prijavljivanje korisnika, ovom svojstvu zadajte vrednost 2.

Listing 2.19 prikazuje mogućnost pasivnog blokiranja baze podataka na primeru obrasca frmViewUsers.

Listing 2.19

```
Const adhcAllowUsers = "Allow New Users"
Const adhcDisallowUsers = "Disallow New Users"

Private Sub cmdShutdown_Click()
    If cmdShutdown.Caption = adhcDisallowUsers Then
        ' Zadajemo pasivnu blokadu i inicijalizujemo
        ' natpis na dugmetu.
        CurrentProject.Connection. _
         Properties("Jet OLEDB:Connection Control") = 1
        cmdShutdown.Caption = adhcAllowUsers
    Else
        ' Uklanjamo pasivnu blokadu i u skladu s tim
        ' menjamo natpis na dugmetu.
        CurrentProject.Connection. _
         Properties("Jet OLEDB:Connection Control") = 2
        cmdShutdown.Caption = adhcDisallowUsers
    End If
End Sub
```

Slika 2.10 prikazuje poruku o grešci koja se pojavljuje kada pokušate da pristupite bazi podataka koja je pasivno blokirana.

SLIKA 2.10 Kada uvedete pasivnu blokadu baze podataka, korisnici koji pokušaju da se u nju prijave dobijaju ovakvu poruku.

Ostala pitanja

U nekoliko narednih odeljaka opisano je više pitanja koja ćete u nekim slučajevima razmatrati kada projektujete aplikacije za višekorisnička okruženja.

Bezbednost

Iako se pitanja bezbednosti ne postavljaju samo u višekorisničkom okruženju, neosporno je da im u takvom okruženju treba posvetiti više pažnje. Kako raste broj radnih stanica koje koriste vašu aplikaciju, raste i verovatnoća da ćete morati da sprečavate neovlašćene korisnike da pristupaju podacima *ili* samoj aplikaciji.

Ako primenjujete zaključavanje na nivou stranice, bezbednosni sistem omogućava Accessu da korisnicima daje tačne podatke o tome ko je određeni zapis zaključao. Nažalost, od sadašnje verzije mehanizma za zaključavanje na nivou zapisa nije moguće dobiti podatak o imenu korisnika koji je zapis zaključao.

NAPOMENA Više detalja o bezbednosnom sistemu naći ćete u poglavlju 8.

Žica

Kada prelazite s jednokorisničkih aplikacija na višekorisničke, treba da počnete da razmišljate o mogućim uskim grlima pri razmeni podataka. Kad god aplikacija zahteva podatke, oni se putem mreže (koja je poznata i kao *žica*) šalju radnoj stanici. To znači da je neophodno da svedete na minimum slanje velikih količina podataka žicom, kako zbog korisnika koji je zahtevao podatke, tako i zbog opterećenja mreže.

Jedna od oblasti kojima treba posvetiti posebnu pažnju jeste *količina* zapisa koju ćete na obrascu prikazati korisnicima. Iako se u Accessu obrazac veoma jednostavno povezuje sa celom tabelom ili upitom, brzo ćete ustanoviti da aplikacija loše radi u mreži kada dopustite korisnicima da se bez ikakvih ograničenja šetaju po celom sadržaju objekta Recordset (čak i ako su tabele skromne veličine od, npr. 30000 do 50000 zapisa). Mnogo je bolje da korisniku ponudite po jedan zapis u datom trenutku i da u kodu menjate izvor podataka obrasca nego da pomoću metode Find prelazite na ciljni zapis.

Testirati, ponovo testirati i još jedanput testirati!

Neophodno je da svoje aplikacije temeljnije testirate ukoliko one treba da rade u mrežnom okruženju. Pravilo koje u tom slučaju treba da imate na umu glasi: „Ako nešto *može* da pođe naopako, to će *sigurno poći naopako*.“ To znači da u višekorisničkim aplikacijama treba da posvetite znatno više vremena testiranju i otklanjanju grešaka.

Jedna od prednosti razvijanja aplikacija u Windows okruženju jeste to što na istoj mašini možete da projektujete, testirate i otklanjate greške u višekorisničkim aplikacijama tako što ćete pokrenuti dve instance Accessa i prelaziti s jedne na drugu. Iako takav način rada omogućava da otkrijete i otklonite mnoge potencijalne probleme, ipak morate da testirate aplikaciju i na ciljnoj mreži, pod opterećenjem uobičajenog broja korisnika koji predviđate, da biste otkrili sve potencijalne probleme. Drugim rečima, nema zamene za stvarne uslove.

Sažetak

Kada razvijate Jet aplikacije za višekorisničko okruženje, važno je imati na umu da treba predvideti moguće probleme. U višekorisničke aplikacije treba da bude ugrađena mogućnost obrade grešaka, ili oporavka od grešaka koje nastaju usled zaključavanja zapisa. Ne postoji skup savršenih opštih rešenja koja mogu da se primene na sve višekorisničke Jet aplikacije. Moraćete da razvijete odgovarajuće rešenje za određenu bazu podataka uzimajući u obzir sledeće:

- Podesite parametre za višekorisnički rad tako da određena aplikacija obezbeđuje maksimalne performanse.
- Razdvojte bazu podataka na bazu za podatke i bazu za aplikacije da biste obezbedili bolje performanse.
- Koristite VBA programski kôd za upravljanje pridruženim tabelama u arhitekturi sa razdvojenim bazama podataka.
- Koristite zaključavanje na nivou zapisa, osim kada imate ozbiljan razlog za korišćenje zaključavanja na nivou stranice.
- Kada razmatrate strategiju zaključavanja, pokušajte da nađete ravnotežu između jednostavnosti upotrebe, integriteta podataka i jednostavnost programiranja.
- Vodite računa o greškama koje mogu da nastanu kada više korisnika deli iste podatke.
- Koristite posebne namenske blokove za obradu grešaka kada radite s vezanim obrascima u kojima se primenjuje optimističko zaključavanje.
- Razmislite o ugradnji vremenskog ograničenja kada radite s vezanim obrascima u kojima se primenjuje pesimističko zaključavanje.
- Koristite transakcije da biste očuvali integritet podataka kada više operacija treba da se izvrši kao celina.
- Koristite liste korisnika i mogućnosti pasivne blokade koju Jet pruža da biste upravljali vezama koje korisnici uspostavljaju s bazom podataka.
- Minimizujte količine podataka koje se prosleđuju linijama mreže.
- Temeljno testirajte svoje aplikacije.