Obrada transakcija u ADO.NET-u

U ovoj lekciji naučićete kako da:

- ✔ *Napravite transakciju*
- ✔ *Napravite ugneženu transakciju*
- ✔ *Potvrdite transakciju*
- ✔ *Poništite transakciju*

U nekoliko poslednjih poglavlja videli smo kako prilikom menjanja i ažuriranja podataka, objekti ADO.NET snabdevača podataka međusobno utiču jedan na drugog. U ovom poglavlju završićemo izlaganje o ADO.NET snabdevačima podataka tako što ćemo istražiti kako se obrađuju transakcije.

Transakcije

Transakcija je serija akcija koje se moraju posmatrati kao jedna celina—akcije moraju ili da sve uspeju, ili da ne uspe nijedna. Klasičan primer transakcije predstavlja prenos novca sa jednog bankovnog računa na drugi. Da bi se novac preneo, željeni iznos, od recimo 100 dinara, podigne se sa jednog računa i položi na drugi. Kad bi se desilo da podizanje novca uspe, a polaganje ne uspe, novac bi zauvek nestao. Kad bi uspelo polaganje, ali ne i podizanje, novac bi bio izmišljen. Jasno je da u slučaju da jedna od akcija ne uspe, moraju da ne uspeju obe akcije.

ADO.NET podržava transakcije korišćenjem objekta Transaction, koji se pravi na otvorenoj konekciji. Komande koje se na transakciji izvrše dok je transakcija u toku moraju se upisati u transakciju tako što će se njihovom svojstvu Transaction dodeliti referenca na objekat Transac-

tion. Komande se ne mogu izvršavati na konekciji izvan transakcije u trenutku dok je neka transakcija otvorena.

Ako se transakcija potvrdi, sve komande koje čine deo te transakcije trajno će se upisati u izvor podataka. Ako se transakcija poništi, sve komande će biti odbačene u izvoru podataka.

Pravljenje transakcija

Objekat Transaction implementiran je kao deo snabdevača podataka. Postoji posebna verzija za svaki od postojećih snabdevača podataka: OleDbTransaction u prostoru imena System.Data.OleDb i SqlTransaction u prostoru imena System.Data.SqlClient.

Objekat SqlTransaction implementiran je korišćenjem Microsoft SQL Serverovih transakcija —pravljenje objekta SqlTransaction preslikava se direktno na naredbu *BeginTransaction*. Objekat OleDbTransaction implementiran je unutar OLE DB-a. Bez obzira na to koji snabdevač podataka koristite, nikada ne bi trebalo da na bazi podataka izvršavate naredbu *BeginTransaction*.

Pravljenje novih transakcija

Transakcije se prave pozivom metoda *BeginTransaction* objekta Connection, koji vraća referencu na objekat Transaction. Metod *BeginTransaction* je preopterećen, čime se dozvoljava da opciono specifikujete svojstvo IsolationLevel, kao što je prikazano u tabeli 5-1. Konekcija mora biti valjana i otvorena prilikom poziva metoda *BeginTransaction*.

Metod	Opis
BeginTransaction()	Počinje transakciju.
BeginTransaction(IsolationLevel)	Počinje transakciju sa specifikovanim nivoom izolovanja IsolationLevel.

Tabela 5-1 Metodi *BeginTransaction* objekta Connection

Pošto SQL Server podržava imenovane transakcije, snabdevač podataka SqlClient sadrži dve dodatne verzije metoda *BeginTransaction*, kao što je prikazano u tabeli 5-2.

Metod	Opis
BeginTransaction(TransactionName)	Počinje transakciju sa imenom specifikovanim znakovnim nizom TransactionName.
BeginTransaction(IsolationLevel, TransactionName)	Počinje transakciju sa specifikovanim nivoom izolovanja IsolationLevel i imenom specifikovanim znakovnim nizom TransactionName.

Tabela 5-2 Dodatni metodi *BeginTransaction* za SQL Server

ADO

Za razliku od ADO-a, metodi *Commit* i *RollBack* su u ADO.NET-u smešteni u objekat Transaction, a ne u objekat Command.

Opcioni parametar IsolationLevel metoda *BeginTransaction* određuje ponašanje konekcije u odnosu na zaključavanje (engl. *lock*). Moguće vrednosti parametra IsolationLevel prikazane su u tabeli 5-3.

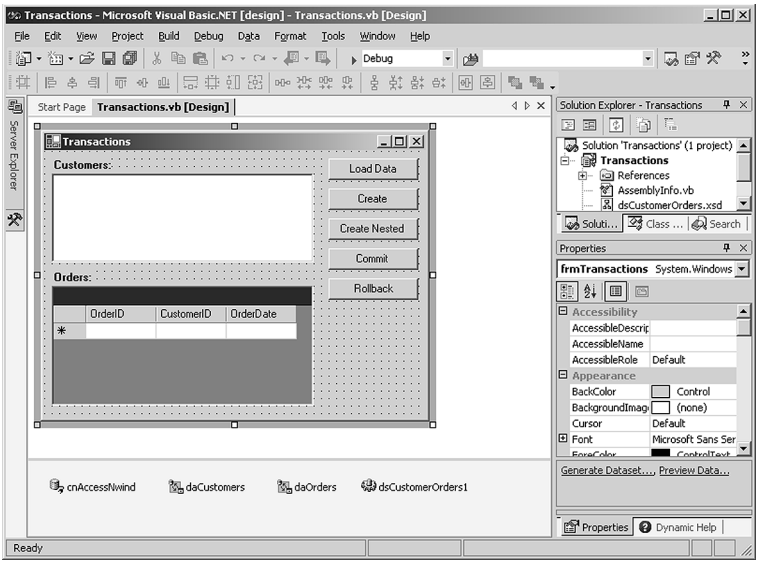
Vrednost	Značenje
Chaos	Izmene na čekanju iz viših rangiranih transakcija koje su u toku ne mogu se prepisati.
ReadCommited	Deljena zaključavanja se čuvaju prilikom čitanja podataka, ali se podaci mogu izmeniti pre kraja transakcije.
ReadUncommited	Ne postavljaju se nikakva deljena zaključavanja niti se zahtevaju bilo kakva ekskluzivna zaključavanja.
RepeatableRead	Ekskluzivna zaključavanja smeštaju se na sve podatke koji se koriste u upitu.
Serializable	Zaključavanje na nivou opsega smešta se na objekat DataSet.
Unspecified	Postojeći nivo izolacije ne može se odrediti.

Tabela 5-3 Nivoi izolovanja

Napravite novu transakciju

Visual Basic .NET

1. Otvorite projekat Transactions ili iz početne stranice Visual Studia ili korišćenjem menija Open.
2. U Solution Exploreru izaberite sa dva uzastopna pritiska na taster miša Transactions.vb da biste otvorili obrazac u dizajneru obrazaca.



3. Dva puta uzastopno pritisnite mišem dugme Create.

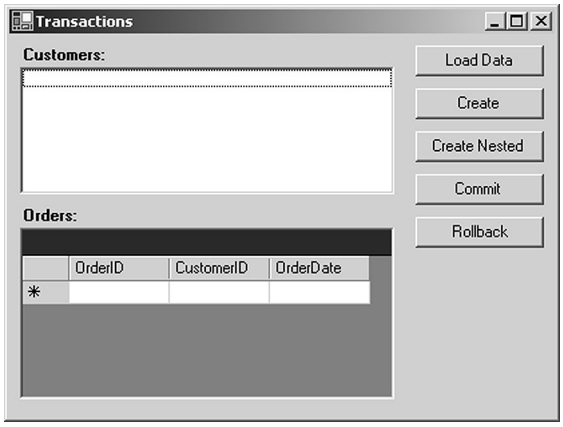
   Visual Studio otvara prozor za uređivanje koda i dodaje rukovaoca događaja Click.

4. Sledeći kod dodajte u proceduru:

```
Dim strMsg As String
Dim trnNew As System.Data.OleDb.OleDbTransaction

Me.cnAccessNwind.Open()
trnNew = Me.cnAccessNwind.BeginTransaction()
strMsg = "Isolation Level:  "
strMsg += trnNew.IsolationLevel.ToString()
MessageBox.Show(strMsg)
Me.cnAccessNwind.Close()
```

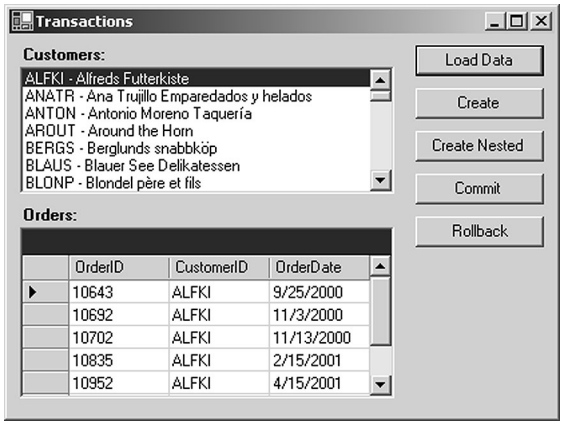
   Ovim kodom se korišćenjem podrazumevanog načina pravi novi objekat Transaction, a zatim njegov nivo izolovanosti prikazuje u okviru za poruku.

5 Pritisnite taster F5 da biste pokrenuli aplikaciju.



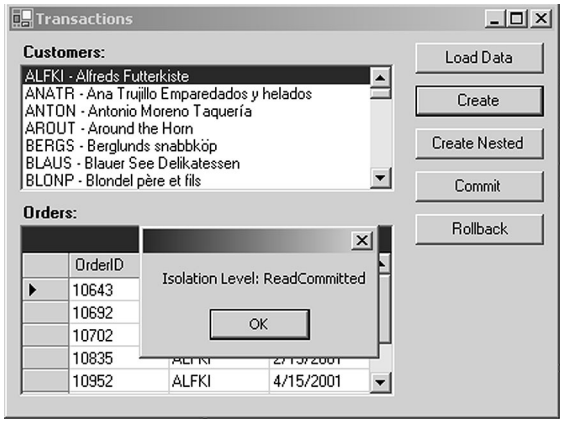
6 Pritisnite mišem dugme Load Data.

Aplikacija popunjava objekat DataSet i prikazuje liste Customers i Orders.



7 Pritisnite mišem dugme Create.

Aplikacija prikazuje nivo izolovanosti transakcije u okviru za poruku.

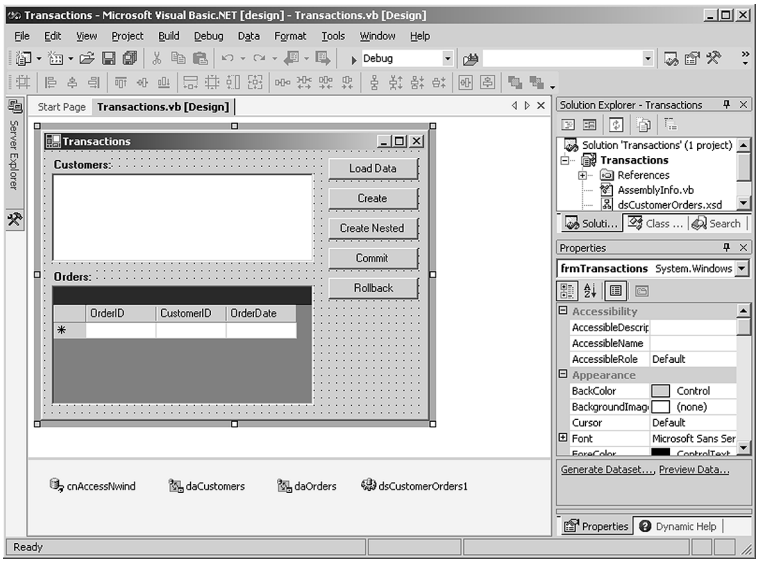


8 Pritisnite mišem dugme OK u okviru za poruku, a zatim zatvorite aplikaciju.

Visual C# .NET

1 Otvorite projekat Transactions ili iz početne stranice Visual Studia ili korišćenjem menija Open.

2 U Solution Exploreru izaberite sa dva uzastopna pritiska na taster miša Transactions.cs da biste otvorili obrazac u dizajneru obrazaca.



3 Dva puta uzastopno pritisnite mišem dugme Create.

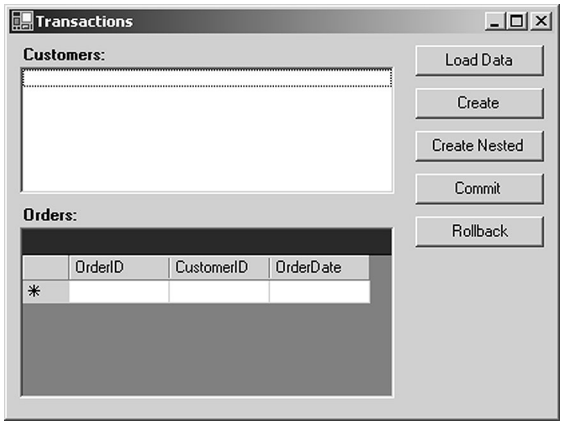
Visual Studio otvara prozor za uređivanje koda i dodaje rukovaoca događaja Click.

4 Sledeći kod dodajte u proceduru:

```
string strMsg;
System.Data.OleDb.OleDbTransaction trnNew;

this.cnAccessNwind.Open();
trnNew = this.cnAccessNwind.BeginTransaction();
strMsg = "Isolation Level: ";
strMsg += trnNew.IsolationLevel.ToString();
MessageBox.Show(strMsg);
this.cnAccessNwind.Close();
```

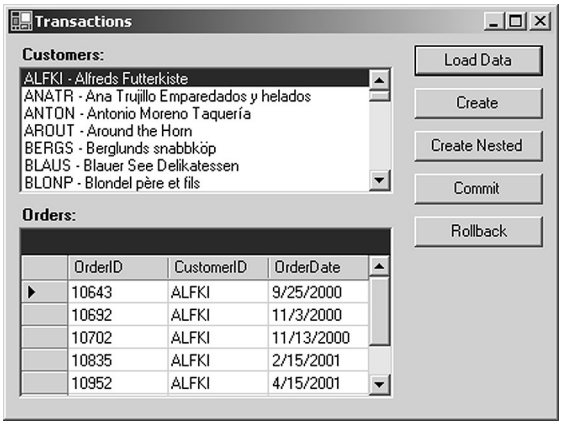
Ovim kodom se korišćenjem podrazumevanog načina pravi novi objekat Transaction, a zatim njegov nivo izolovanosti prikazuje u okviru za poruku.

5 Pritisnite taster F5 da biste pokrenuli aplikaciju.



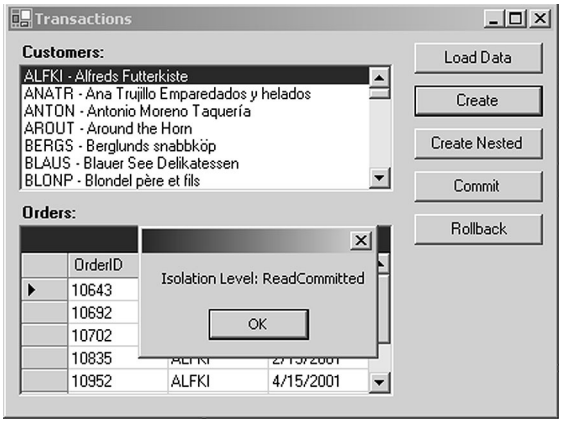
6 Pritisnite mišem dugme Load Data.

Aplikacija popunjava objekat DataSet i prikazuje liste Customers i Orders.



7 Pritisnite mišem dugme Create.

Aplikacija prikazuje nivo izolovanosti transakcije u okviru za poruku.



8 Pritisnite mišem dugme OK u okviru za poruku, a zatim zatvorite aplikaciju.

Pravljenje ugneždenih transakcija

Iako nije moguće da na jednom objektu Connection istovremeno postoje dve transakcije, objekat OleDbTransaction podržava ugneždene transakcije. (One nisu podržane na SQL Serveru.)

ADO

Dok je u ADO-u bilo podržano više transakcija na jednom objektu Connection, u ADO.NET-u to nije slučaj.

Sintaksa za pravljenje ugneždenih transakcija potpuno je ista kao i za pravljenje transakcija prvog nivoa, kao što je prikazano u tabeli 5-4. Razlika je u tome što se ugneždene transakcije prave pozivom metoda *BeginTransaction* na samom objektu Transaction, a ne na objektu Connection.

Sve ugneždene transakcije se moraju potvrditi ili poništiti pre nego što se potvrdi transakcija koja ih sadrži; međutim, ako se roditeljska (ona koja sadrži ostale) transakcija poništi, ugneždene transakcije će se takođe poništiti, čak i u slučaju da su prethodno već bile potvrđene.

Metod	Opis
BeginTransaction()	Počinje transakciju.
BeginTransaction(IsolationLevel)	Počinje transakciju sa specifikovanim nivoom izolovanja IsolationLevel.

Tabela 5-4 Metodi *BeginTransaction* objekta Transaction

Napravite ugneždenu transakciju

Visual Basic .NET

1. Iz liste ClassName izaberite btnNested, a zatim iz liste MethodName izaberite Click.

   Visual Studio otvara šablon rukovaoca događaja Click.

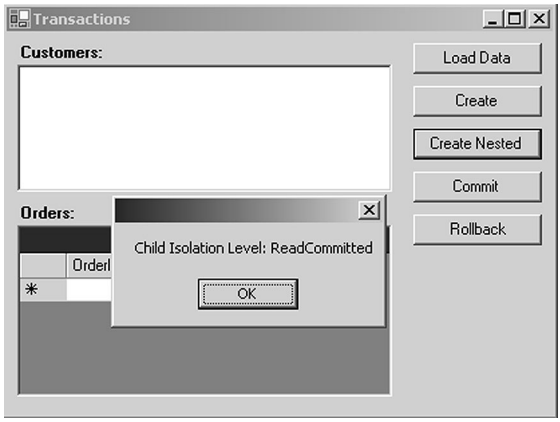
2. U proceduru dodajte sledeći kod:

```
Dim strMsg As String
Dim trnMaster As System.Data.OleDb.OleDbTransaction
Dim trnChild As System.Data.OleDb.OleDbTransaction

Me.cnAccessNwind.Open()

trnMaster = Me.cnAccessNwind.BeginTransaction()
trnChild = trnMaster.Begin()
strMsg = "Child Isolation Level:  "
strMsg += trnChild.IsolationLevel.ToString()
MessageBox.Show(strMsg)

Me.cnAccessNwind.Close()
```

   U kodu se prvo pravi transakcija, trnMaster, na objektu Connection. Zatim se na transakciji trnMaster pravi druga, ugneždena transakcija, trnChild i u okviru za poruku prikazuje njen nivo izolovanosti.

3. Pritisnite taster F5 da biste pokrenuli aplikaciju.
4. Pritisnite mišem dugme Load Data.
5. Pritisnite mišem dugme Nested.

   Aplikacija prikazuje nivo izolovanosti ugneždene transakcije u okviru za poruku.



6 Pritisnite mišem dugme OK u okviru za poruku, a zatim zatvorite aplikaciju.

Visual C# .NET

1 Sledeću proceduru dodajte u kod:

```
private void btnCreate_Click(object sender, System.EventArgs e)
{
  string strMsg;
  System.Data.OleDb.OleDbTransaction trnNew;
  this.cnAccessNwind.Open();
  trnNew = this.cnAccessNwind.BeginTransaction();
  strMsg = "Isolation Level: ";
  strMsg += trnNew.IsolationLevel.ToString();
  MessageBox.Show(strMsg);
  this.cnAccessNwind.Close();
}
```

U kodu se prvo pravi transakcija, trnMaster, na objektu Connection. Zatim se na transakciji trnMaster pravi druga, ugneždena transakcija, trnChild i u okviru za poruku prikazuje njen nivo izolovanosti.

2 Dodajte kod za povezivanje prethodnog rukovaoca događaja na početak procedure frmTransactions():

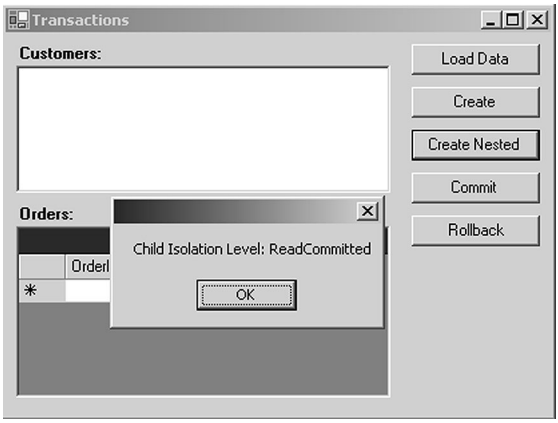
```
this.btnNested.Click += new EventHandler(this.btnNested_Click);
```

3 Pritisnite taster F5 da biste pokrenuli aplikaciju.

4 Pritisnite mišem dugme Load Data.

5 Pritisnite mišem dugme Nested.

Aplikacija prikazuje nivo izolovanosti ugneždene transakcije u okviru za poruku.



6 Pritisnite mišem dugme OK u okviru za poruku, a zatim zatvorite aplikaciju.

Korišćenje transakcija

Pošto se transakcije naprave, potrebno je da se obave tri koraka da bi se mogle koristiti. Prvo se transakcije dodeljuju komandama koje će u njima učestvovati, zatim se komande izvršavaju i na kraju se transakcije zatvaraju ili njihovim izvršavanjem ili vraćanjem u pređašnje stanje.

Dodeljivanje transakcija objektu Command

Jednom kada se na konekciji otpočne transakcija, sve komande koje se izvršavaju na toj konekciji moraju da učestvuju u toj transakciji. Nažalost, ovo se ne dešava automatski—potrebno je da svojstvu Transaction objekta Command dodelite referencu na tu transakciju.

Međutim, kada se transakcija jednom potvrdi ili poništi, referenca na transakciju u svim komandama koje učestvuju u transakciji ponovo će se postaviti na vrednost Nothing, tako da nije potrebno da i ovaj korak obavljate ručno.

Potvrđivanje i poništavanje transakcija

Poslednji korak u obradi transakcija je da se izmene napravljene komandama koje učestvuju u transakciji potvrde ili ponište. Ako se transakcija potvrdi, sve izmene će se prihvatiti u izvoru podataka. Ako se poništi, sve izmene će se odbaciti i izvor podataka će se vratiti u stanje u kojem se nalazio pre početka transakcije.

Transakcije se potvrđuju korišćenjem metoda *Commit* objekta Transaction, a poništavaju korišćenjem metoda *Rollback* objekta Transaction. Nijedan od ova dva metoda nema nikakve parametre. Ove akcije se obično nalaze u bloku Try ... Catch.

Potvrdite transakciju

Visual Basic .NET

1. Iz liste ClassName izaberite btnCommit, a zatim iz liste MethodName izaberite Click.

   Visual Studio otvara šablon rukovaoca događajem Click.

2. U proceduru dodajte sledeći kôd:

```
Dim trnNew As System.Data.OleDb.OleDbTransaction

AddRows("AAAA1")

Me.cnAccessNwind.Open()
trnNew = Me.cnAccessNwind.BeginTransaction()
Me.daCustomers.InsertCommand.Transaction = trnNew
Me.daOrders.InsertCommand.Transaction = trnNew
Try
  Me.daCustomers.Update(Me.dsCustomerOrders1.CustomerList)
  Me.daOrders.Update(Me.dsCustomerOrders1.Orders)
  trnNew.Commit()
  MessageBox.Show("Transaction Committed")
Catch err As System.Data.OleDb.OleDbException
  trnNew.Rollback()
  MessageBox.Show(err.Message.ToString())
Finally
  Me.cnAccessNwind.Close()
End Try
```

Procedura AddRows, koji je obezbeđen još u poglavlju 1, dodaje red potrošača u tabelu Customers i porudžbinu za tog potrošača u tabelu Order.

U bloku Try ... Catch najpre se, pod uslovom da uspeju, izvršavaju dve komande Update, a zatim prikazuje poruka koja potvrđuje da se transakcija završila bez grešaka.

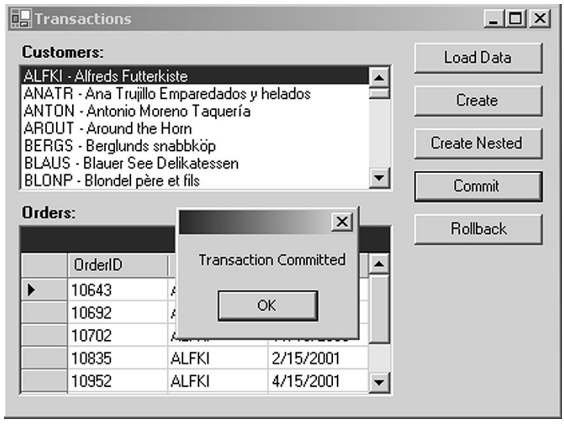
3 Pritisnite taster F5 da biste pokrenuli aplikaciju.

4 Pritisnite mišem Load Data.

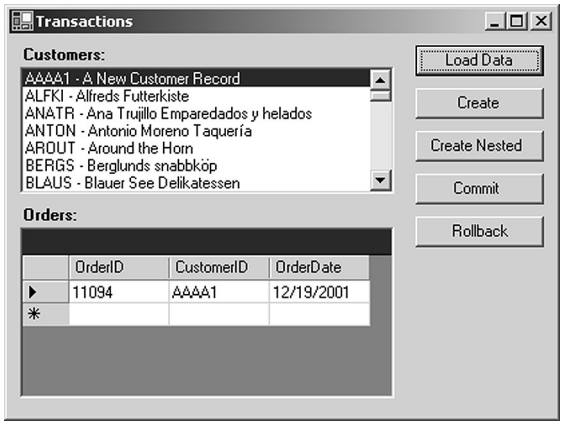
Aplikacija popunjava objekat DataSet i prikazuje liste Customers i Orders.

5 Pritisnite mišem dugme Commit.

Aplikacija prikazuje okvir za poruku koji potvrđuje da su ažuriranja uspešno obavljena.



6 Pritisnite mišem dugme OK iz okvira za poruku, a zatim pritisnite mišem dugme Load Data da biste se uverili da su redovi zaista umetnuti.



7 Zatvorite aplikaciju.

Visual C# .NET

1. Sledeću proceduru dodajte u kod:

```
private void btnCommit_Click(object sender, System.EvantArgs e)
{
  System.Data.OleDb.OleDbTransaction trnNew;

  AddRows("AAAA1");

  this.cnAccessNwind.Open();
  trnNew = this.cnAccessNwind.BeginTransaction();
  this.daCustomers.InsertCommand.Transaction = trnNew;
  this.daOrders.InsertCommand.Transaction = trnNew;
  try
  {
          this.daCustomers.Update(this.dsCustomerOrders1.CustomerList);
          this.daOrders.Update(this.dsCustomerOrders1.Orders);
          trnNew.Commit();
          MessageBox.Show("Transaction Committed");
  }
  catch (System.Data.OleDb.OleDbException err)
  {
          trnNew.Rollback();
          MessageBox.Show(err.Message.ToString());
  }
  finally
  {
          this.cnAccessNwind.Close();
  }
}
```

   Procedura AddRows, koji je obezbeđen još u poglavlju 1, dodaje red potrošača u tabelu Customers i porudžbinu za tog potrošača u tabelu Order.

   U bloku Try ... Catch najpre se, pod uslovom da uspeju, izvršavaju dve komande Update, a zatim prikazuje poruka koja potvrđuje da je transakcija završena bez grešaka.

2. Dodajte kod za povezivanje prethodnog rukovaoca događaja na početak procedure frmTransactions():

```
this.btnCommit.Click += new EventHandler(this.btnCommit_Click);
```

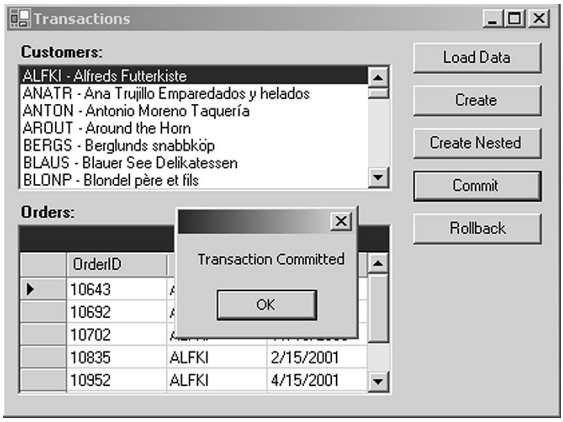
3. Pritisnite taster F5 da biste pokrenuli aplikaciju.

4 Pritisnite mišem Load Data.

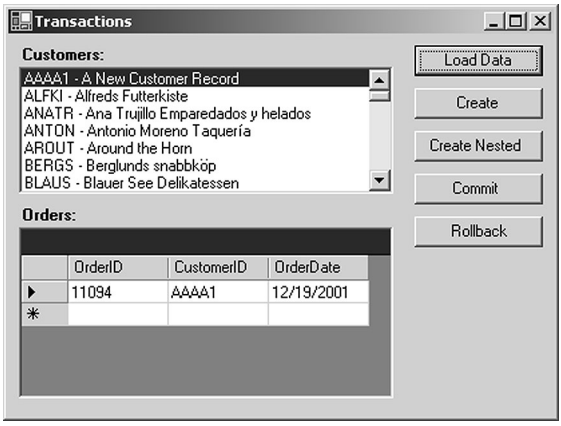
Aplikacija popunjava objekat DataSet i prikazuje liste Customers i Orders.

5 Pritisnite mišem dugme Commit.

Aplikacija prikazuje okvir za poruku koji potvrđuje da su ažuriranja uspešno obavljena.



6 Pritisnite mišem dugme OK iz okvira za poruku, a zatim pritisnite mišem dugme Load Data da biste se uverili da su redovi zaista umetnuti.



7 Zatvorite aplikaciju.

Poništite transakciju

Visual Basic .NET

1. Iz liste ClassName izaberite btnRollBack, a zatim iz liste MethodName izaberite Click.

   Visual Studio otvara šablon rokovaoca događaja Click.

2. U proceduru dodajte sledeći kod:

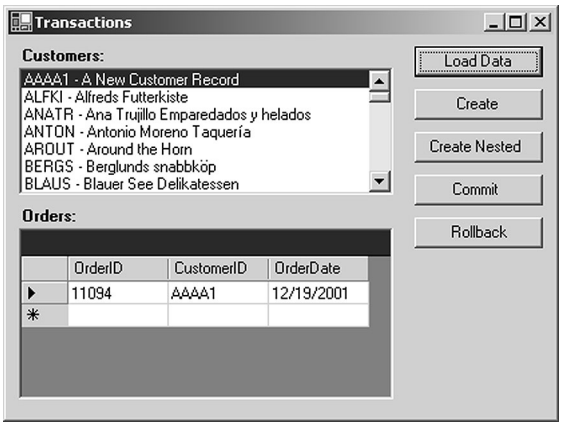
```
Dim trnNew As System.Data.OleDb.OleDbTransaction

AddRows("AAAA2")

Me.cnAccessNwind.Open()
trnNew = Me.cnAccessNwind.BeginTransaction()
Me.daCustomers.InsertCommand.Transaction = trnNew
Me.daOrders.InsertCommand.Transaction = trnNew
Try
  Me.daOrders.Update(Me.dsCustomerOrders1.Orders)
  Me.daCustomers.Update(Me.dsCustomerOrders1.CustomerList)
  trnNew.Commit()
  MessageBox.Show("Transaction Committed")
Catch err As System.Data.OleDb.OleDbException
  trnNew.Rollback()
  MessageBox.Show(err.Message.ToString())
Finally
  Me.cnAccessNwind.Close()
End Try
```

   Ova procedura je gotovo identična sa procedurom Commit iz prethodne vežbe. Međutim, pošto je redosled ažuriranja obrnut tako da se sada porudžbina dodaje pre potrošača, prva naredba Update neće uspeti i okvir za poruku će prikazati grešku.

3. Pritisnite taster F5 da biste pokrenuli aplikaciju.

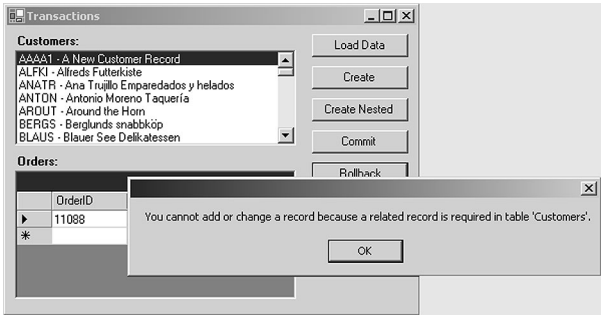
4. Pritisnite mišem Load Data.

   Aplikacija popunjava objekat DataSet i prikazuje liste Customers i Orders.

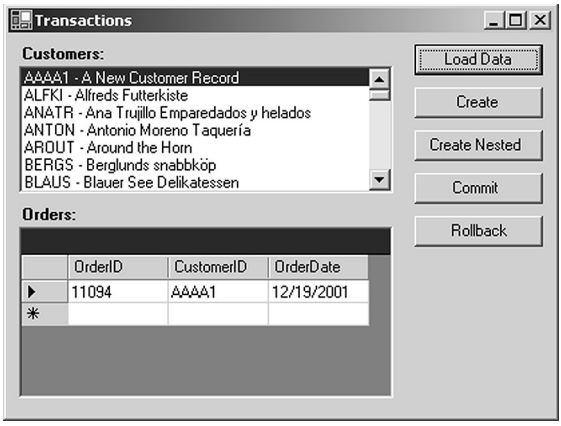


5 Pritisnite mišem dugme RollBack.

Aplikacija prikazuje okvir za poruku koji objašnjava grešku.



6 Pritisnite mišem dugme OK iz okvira za poruku, a zatim pritisnite mišem dugme Load Data da biste se uverili da su redovi zaista umetnuti.



7 Zatvorite aplikaciju.

Visual C# .NET

1. Sledeću proceduru dodajte u kod:

```
private void btnRollbackžClick(object sender, System.EventArgs e)
{
  System.Data.OleDb.OleDbTransaction trnNew;

  AddRows("AAAA2");

  this.cnAccessNwind.Open();
  trnNew = this.cnAccessNwind.BeginTransaction();
  this.daCustomers.InsertCommand.Transaction = trnNew;
  this.daOrders.InsertCommand.Transaction = trnNew;
  try
  {
          this.daOrders.Update(this.dsCustomerOrders1.Orders);
          this.daCustomers.Update(this.dsCustomerOrders1.CustomerList);
          trnNew.Commit();
          MessageBox.Show("Transaction Committed");
  }
  catch (System.Data.OleDb.OleDbException err)
  {
          trnNew.Rollback();
          MessageBox.Show(err.Message.ToString());
  }
  finally
  {
          this.cnAccessNwind.Close();
  }
}
```

   Ova procedura je gotovo identična sa procedurom Commit iz prethodne vežbe. Međutim, pošto je redosled ažuriranja obrnut tako da se sada porudžbina dodaje pre potrošača, prva naredba Update neće uspeti i okvir za poruku će prikazati grešku.

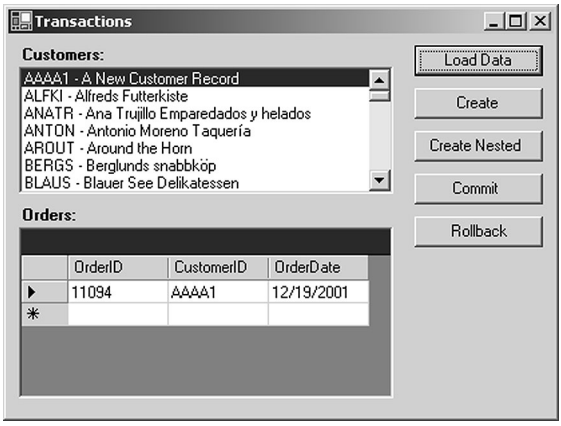
2. Dodajte kod za povezivanje prethodnog rukovaoca događaja na početak procedure frmTransactions():

```
this.btnRollback.Click += new EventHandler(this.btnRollback_Click);
```

3. Pritisnite taster F5 da biste pokrenuli aplikaciju.

4. Pritisnite mišem Load Data.

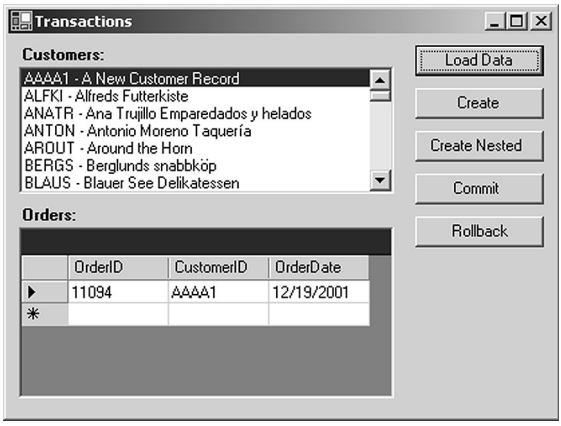
   Aplikacija popunjava objekat DataSet i prikazuje liste Customers i Orders.



5 Pritisnite mišem dugme RollBack.

Aplikacija prikazuje okvir za poruku koji objašnjava grešku.

6 Pritisnite mišem dugme OK iz okvira za poruku, a zatim pritisnite mišem dugme Load Data da biste se uverili da su redovi zaista umetnuti.



7 Zatvorite aplikaciju.

Brzi pregled poglavlja 5

Da biste	Uradite sledeće
Napravili transakciju	Pozovite metod *BeginTransaction* objekta Connection: `myTrans = myConn.BeginTransaction`
Napravili ugneždenu transakciju	Pozovite metod *BeginTransaction* objekta Transaction: `nestedTrans = myTrans.BeginTransaction`
Potvrdili transakciju	Pozovite metod *Commit* objekta Transaction: `myTrans.Commit()`
Poništili transakciju	Pozovite metod *RollBack* objekta Transaction: `myTrans.RollBack()`