```
/*
-- Distributed Transaction demo

-- na serwerach w AdventureWorks2008
CREATE TABLE [dbo].[Konta](
[id] [int] IDENTITY(1,1) NOT NULL,
[name] [nvarchar](50) NOT NULL,
[value] [money] NOT NULL,
CONSTRAINT [PK_Konta] PRIMARY KEY CLUSTERED
(
[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]


BEGIN TRANSACTION
GO
ALTER TABLE dbo.Konta ADD CONSTRAINT
	CK_Konta_saldo CHECK (value >= 0)
GO
ALTER TABLE dbo.Konta SET (LOCK_ESCALATION = TABLE)
GO
COMMIT


-- Na obu serwerach utworzyć login labuser, password Passw0rd
-- utworzyć użytkowników labuser z loginami labuser
-- przydzielić role datareader, datawriter

-- Na jednym serwerze
INSERT INTO Konta (name, value) VALUES ('John', 1000);
INSERT INTO Konta (name, value) VALUES ('Paul', 5000);

-- Na drugim
INSERT INTO Konta (name, value) VALUES ('Alice', 1500);
INSERT INTO Konta (name, value) VALUES ('Margo', 4500);

GO
*/


// Console Application
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.SqlClient;
```

```
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.Transactions;

namespace BankDistributedTransaction
{
   class Program
   {
      static void Main(string[] args)
      {
         bankTransaction(1000, "Alice", "John");
      }

      public static void bankTransaction(decimal val, string cust1, string cust2)
      {
         int returnValue = 0;

         using (TransactionScope oTran = new TransactionScope())
         {
            using (SqlConnection oConn =
            new SqlConnection(@"Data Source=MSSQLSERVER114;Initial
Catalog=AdventureWorks2008;User Id=labuser;Password=Passw0rd;"))
            {
               try
               {
                  oConn.Open();
                  SqlCommand update = new SqlCommand("update dbo.Konta set value = value +
@value where name = @name", oConn);

                  update.Parameters.Add("@value", System.Data.SqlDbType.Money);
                  update.Parameters.Add("@name", System.Data.SqlDbType.NVarChar);
                  update.Parameters["@value"].Value = val;
                  update.Parameters["@name"].Value = cust1;
                  returnValue = update.ExecuteNonQuery();
                  if (returnValue < 1) throw new Exception();

                  using (SqlConnection remConn =
                     //new SqlConnection("Data Source=MSSQLSERVER114;Initial
Catalog=AdventureWorks2008;Integrated Security=True;"))
                     //Integrated Security=True;  User Id=labuser;Password=Passw0rd;
                     //new SqlConnection("Data Source=MSSQLSERVER80;Initial
Catalog=AdventureWorks2008;User Id=labuser;Password=Passw0rd;"))
                  new SqlConnection(@"Data Source=MSSQLSERVER79;Initial
Catalog=AdventureWorks2008;User Id=labuser;Password=Passw0rd;"))
                  {
                     returnValue = 0;
                     remConn.Open();
```

```
                    SqlCommand updateRemote =
                      // success
                      //new SqlCommand("update dbo.Konta set value = value - 50.00 where name =
'John'", remConn);

                      // bad - AJohn not exist
                      new SqlCommand("update dbo.Konta set value = value - @value where name =
@name", remConn);
                    updateRemote.Parameters.Add("@value", System.Data.SqlDbType.Money);
                    updateRemote.Parameters.Add("@name", System.Data.SqlDbType.NVarChar);
                    updateRemote.Parameters["@value"].Value = val;
                    updateRemote.Parameters["@name"].Value = cust2;
                    returnValue = updateRemote.ExecuteNonQuery();
                    if (returnValue < 1) throw new Exception();
                    oTran.Complete();
                  }
                }
                catch (SqlException exception)
                {
                  Console.WriteLine(exception.Message);

                }
                catch (Exception exception)
                {
                  Console.WriteLine(exception.Message);

                }

              }

            }
            // The returnValue is greater than 0 if the transaction committed.
            if (returnValue > 0)
            {
              Console.WriteLine("transaction complete");

            }
            else
            {
              Console.WriteLine("transaction rollback");
            }
            Console.ReadKey();
          }

        }
}
```