

The background of the slide is a light blue gradient with abstract, flowing lines. In the top left, there is a faint image of a hand typing on a laptop keyboard. In the bottom right, there is a large, stylized '@' symbol. The text 'ADO.NET 2.0 Features' is centered in the middle of the slide in a bold, red font.

ADO.NET 2.0 Features

Objectives

- On completion of this chapter you will be able to
 - ◆ Use Multiple Active Result Sets
 - ◆ Access data using Asynchronous Processing
 - ◆ Update data in disconnected environment using Batch Update.
 - ◆ Implement Bulk Copy to copy data from different data store into SQL Server.
 - ◆ Implement Transactions

New features in ADO.NET 2.0

- Multiple Active Result Sets (MARS)
 - ♦ Host multiple result sets on the same connection
- Asynchronous operations
 - ♦ Perform data access operations as background tasks
- Multiple Batch Updates
 - ♦ Update multiple dataset rows in one batch
- Bulk Data Copy
 - ♦ Upload large files efficiently
- Transactions
 - ♦ Commit operations if all are successful otherwise rollback.

Multiple Active Result Sets (MARS)

- MARS allows the execution of multiple batches on a single connection.
 - ◆ Allows an application to have more than one `SqlDataReader` open on a connection when each instance of `SqlDataReader` is started from a separate command.
- Add the following to connection string

```
<connectionStrings>  
  <add name="Northwind"  
connectionString="Server=localhost; Database=Northwind;  
User ID=sa; Password=sa; MultipleActiveResultSets=True" />  
</connectionStrings>
```

Multiple Active Result Sets (MARS)

```
SqlDataReader rdrOne = null;
SqlDataReader rdrTwo = null;
SqlCommand cmdOne = new SqlCommand("Select * from
                                   Category;", conn);
SqlCommand cmdTwo = new SqlCommand("Select * from
                                   CDInfo;", conn);

conn.Open();
rdrOne = cmdOne.ExecuteReader();
DataTable dtOne = new DataTable();
dtOne.Load(rdrOne);
GridView1.DataSource = dt;

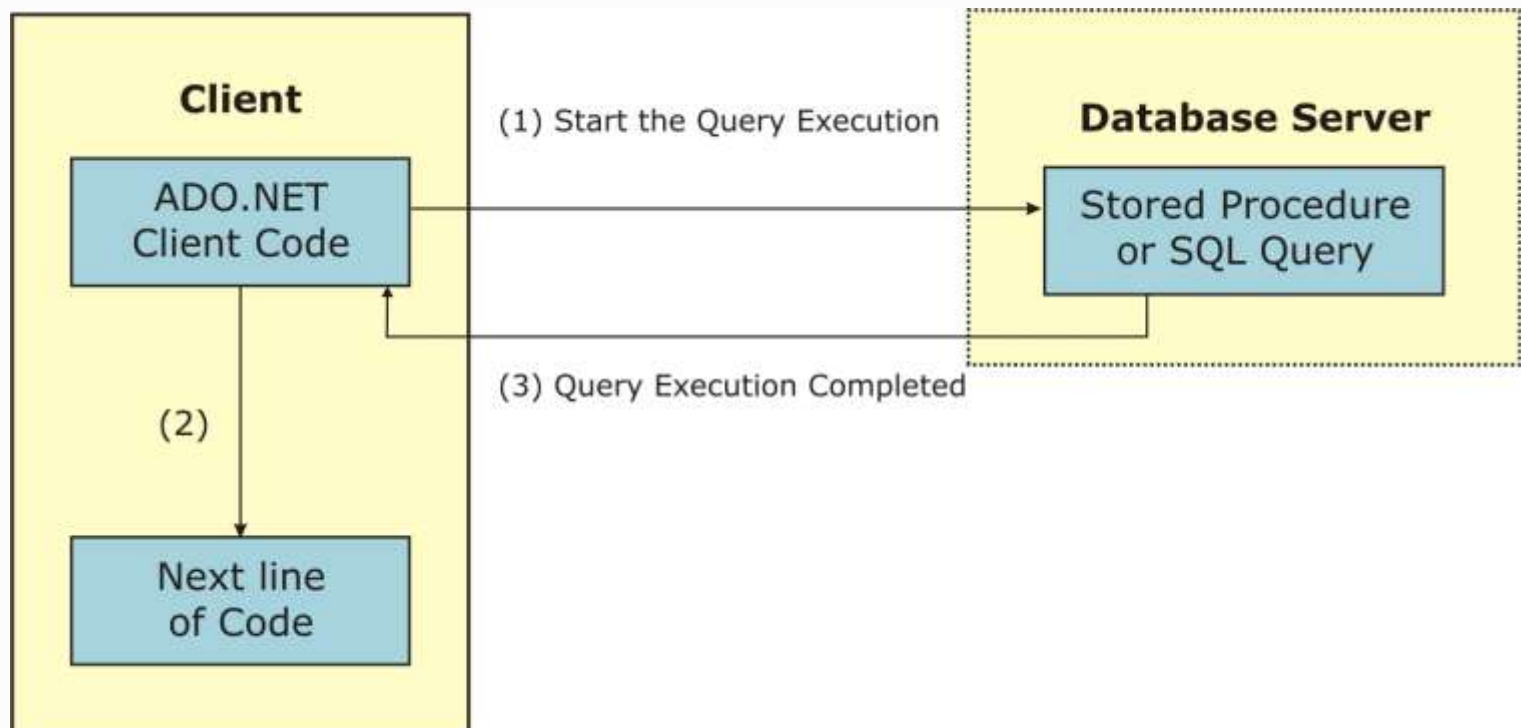
rdrTwo = cmdTwo.ExecuteReader();
DataTable dtTwo = new DataTable();
dtTwo.Load(rdrTwo);
GridView2.DataSource = dtTwo;
conn.Close();
```

Represents
connection
object

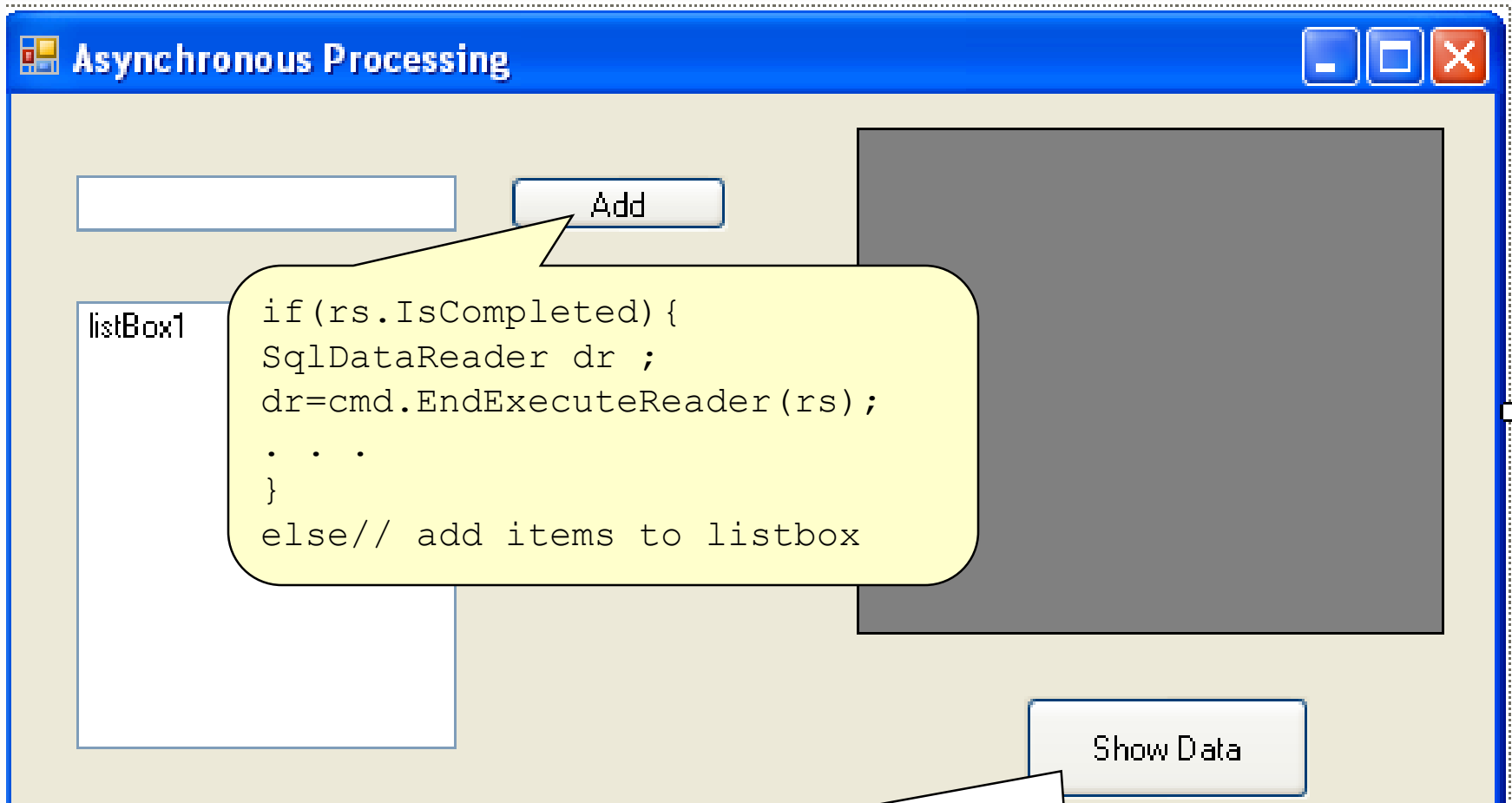
execution of multiple
batches on a single
connection

Asynchronous Processing

- Asynchronous Processing
 - ◆ Database operation is performed on a separate thread and currently executing thread can perform some other operation.



Asynchronous Processing



```
IAsyncResult rs;  
con.Open();  
rs=cmd.BeginExecuteReader();
```

Batch Updates

- ♦ The `Update()` method of a `DataAdapter` performed updates to the database one row at a time .
- ♦ The `DataAdapter` exposes an `UpdateBatchSize` property in ADO.NET 2.0.
 - Setting the property to a positive integer value causes updates to the database to be sent as batches of the specified size.

Batch Updates

```
string constr = "Uid=sa;Pwd=sa;Initial  
Catalog=Northwind;Data Source=VEENA";  
  
SqlConnection conn = new SqlConnection(constr);  
SqlCommand cmd = new SqlCommand();  
  
SqlDataAdapter da = new SqlDataAdapter("Select *  
from Customers", conn);  
DataSet ds= new DataSet();  
  
        da.Fill(ds, "Customers");  
  
SqlCommandBuilder cmdBld= new SqlCommandBuilder(da);  
da.UpdateBatchSize = ds.Tables[0].Rows.Count;  
da.Update(ds, "Customers");
```

Bulk Copy

- The `SqlBulkCopy` class
 - ◆ Used to write bulk of data only to SQL Server tables.
 - ◆ The data source not limited to SQL Server as long as the data can be loaded to a `DataTable` instance or read with a `IDataReader` instance.
 - ◆ Can perform
 - A single or multiple bulk copy operations.
 - A bulk copy operation within a transaction.

Bulk Copy Example

```
using System.Data.SqlClient;
. . .
SqlConnection sourceConn = new SqlConnection (srcConstr);
SqlConnection destConn = new SqlConnection (destConstr);
DataSet ds = new DataSet(emptytab);
SqlDataAdapter sda = new SqlDataAdapter("select * from
                                     employee", sourceConn);
Sda.Fill(ds, "emptytab");
SqlBulkCopy mybulkcopy = new SqlBulkCopy(destConn);
mybulkcopy.DestinationTableName = "Emp";

sourceConn.Open();    destConn.Open();
mybulkcopy.WriteToServer(ds.Tables["emptytab"]);
mybulkcopy.Close();
sourceConn.Close();   destConn.Close();
```

Transaction

- A set of operations treated as one logical unit is known as Transaction.
- The `System.Transactions` namespace provides
 - ◆ An explicit programming model based on the `Transaction` class
 - ◆ An implicit programming model using the `TransactionScope` class in which transactions are automatically managed by the infrastructure.

Implicit Transaction

```
using (TransactionScope s = new TransactionScope())
{
    using (SqlConnection con = new SqlConnection("..."))
    {
        con.Open();
        SqlCommand cmd1 = new SqlCommand("insert into ... con);
        cmd1.ExecuteNonQuery();
        using (SqlConnection con2 = new SqlConnection("..."));
        {
            con2.Open();
            SqlCommand cmd2 = new SqlCommand("insert ..., con2);
            cmd2.ExecuteNonQuery();
        }
    }
}
s.Complete();
```

Quick Recap...

- MARS allows the execution of multiple batches on a single connection.
- Asynchronous Processing allows database operation to be performed on a separate thread with user interaction on main thread.
- `DataAdapter` could update multiple rows in one batch in a disconnected scenario.
- `SqlBulkCopy` class performs a single, multiple bulk copy operations, a bulk copy operation within a transaction.
- `System.Transactions` provides a way of implementing a transaction.