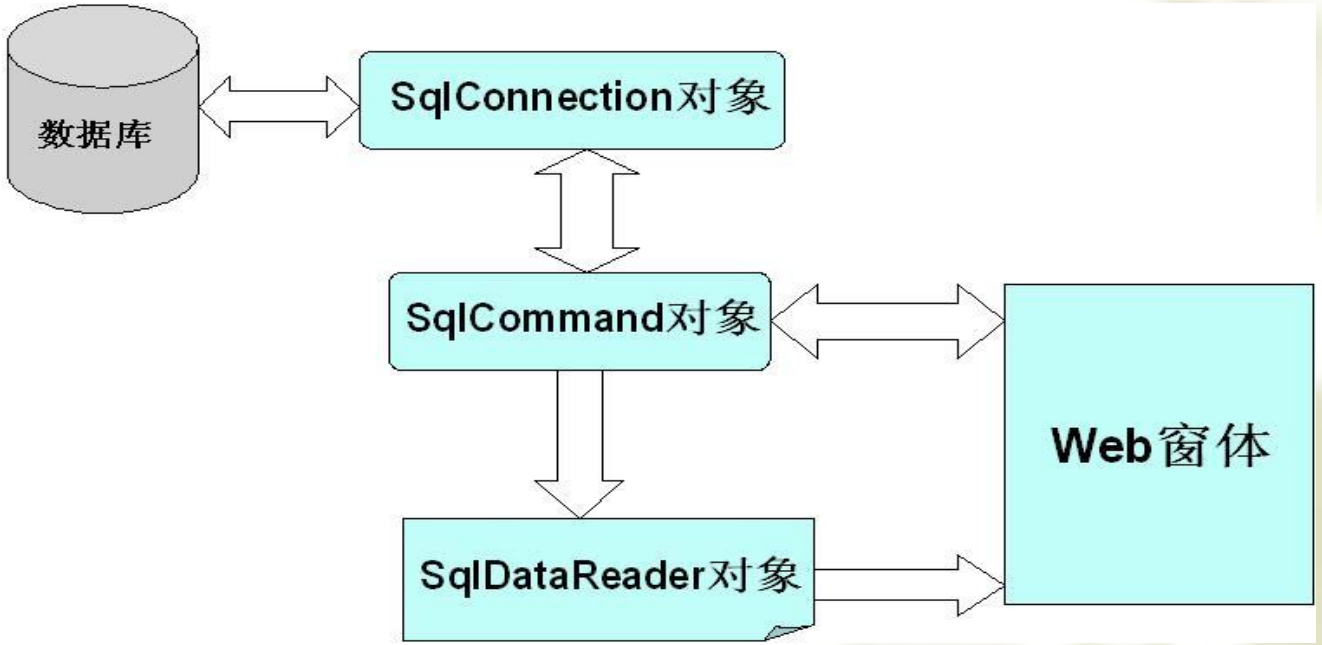
第8章 ADO.NET 数据访问技术2

内容

- ADO.NET基础
- 连接模式数据库访问
- 断开模式数据库访问

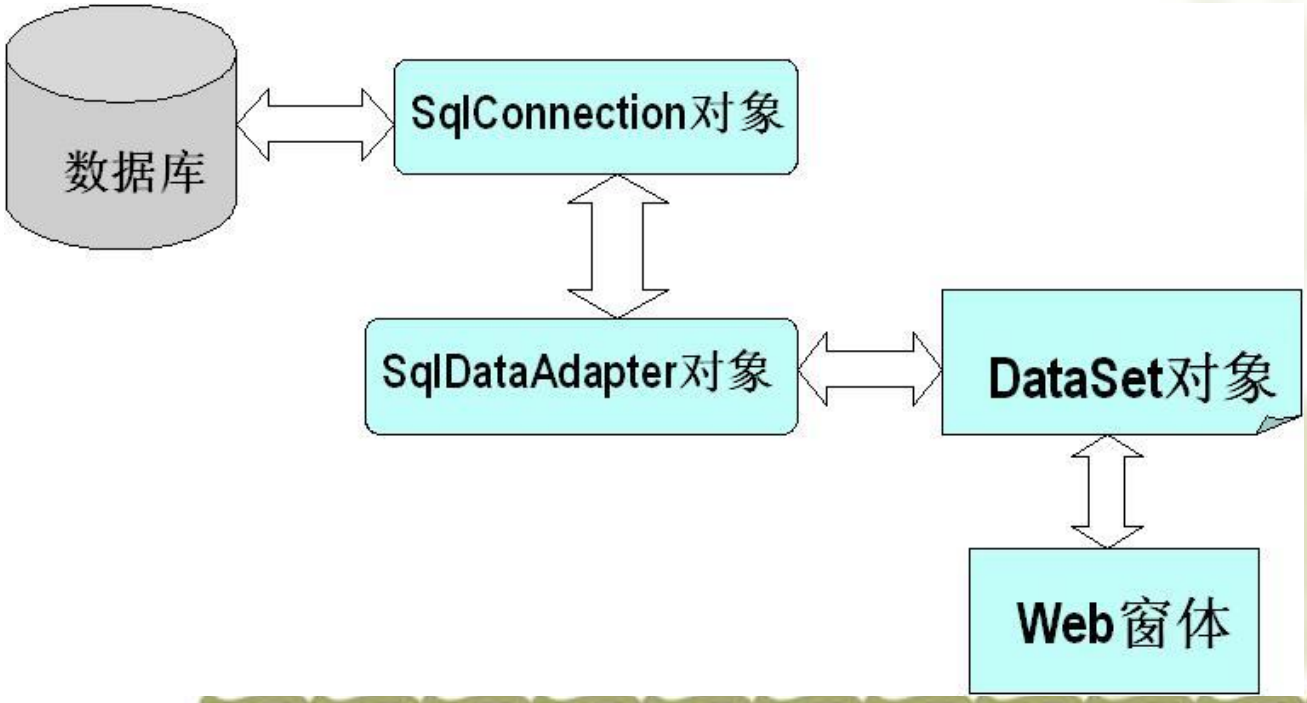
8.2 连接模式数据库访问

- SqlConnection对象：连接数据库
- SqlCommand对象：执行数据库命令



8.3 断开模式数据库访问

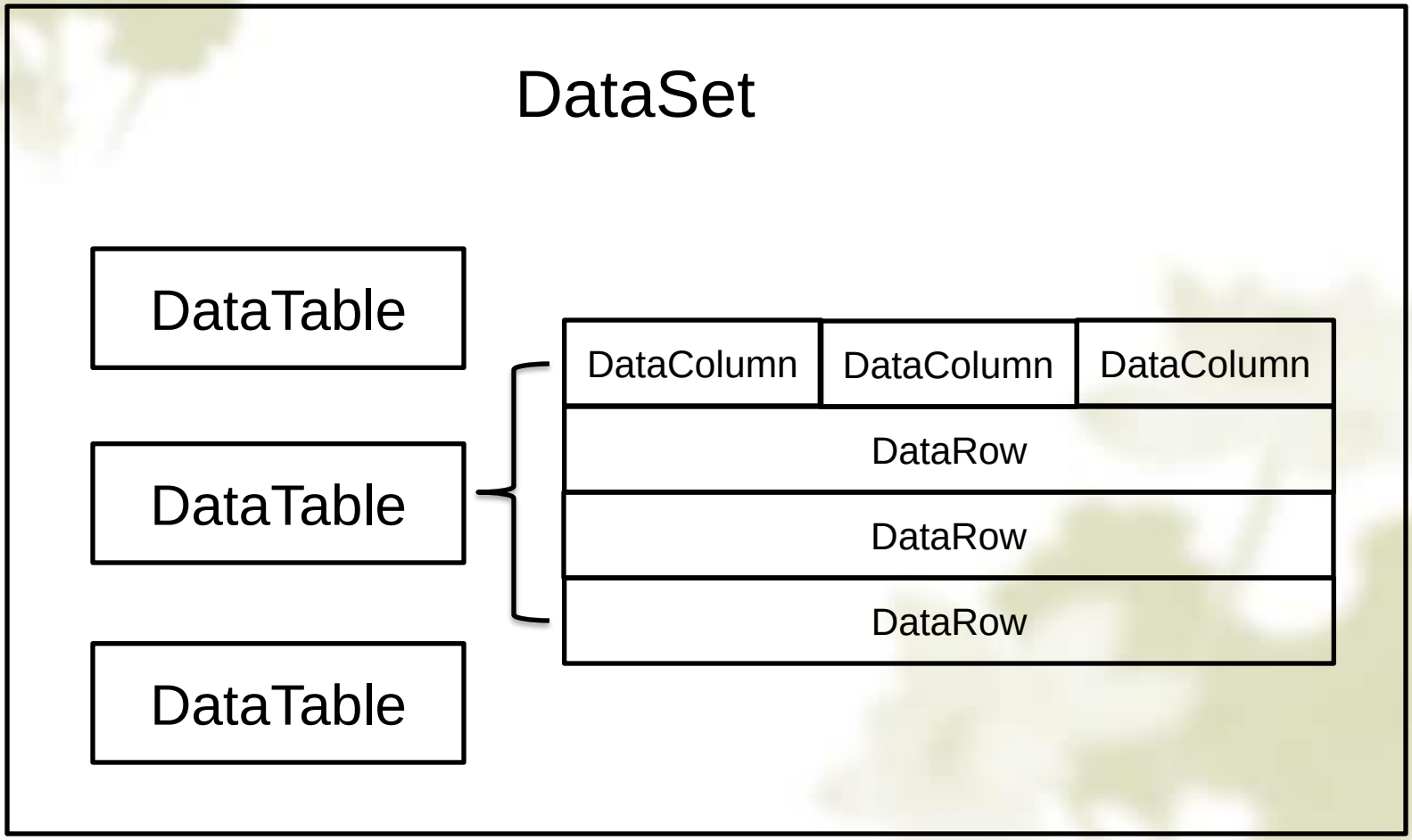
- DataSet数据集
- 使用SqlDataAdapter对象执行数据库命令



8.3.1 DataSet数据集

- 数据集DataSet位于System.Data命名空间下，用于在内存中暂存数据，可以把它看成是内存中的小型数据库。DataSet包含一个或多个数据表（DataTable），表数据可来自数据库、文件或XML数据。
- DataSet一旦读取到数据库中的数据后，就在内存中建立数据库的副本，在此之后的所有操作都是在内存中的DataSet中完成，直到执行更新命令为止。

DataSet数据集结构



1. 数据集（DataSet）对象创建

创建DataSet的语法格式为：

```
DataSet 对象名 = new DataSet();
```

或

```
DataSet 对象名 = new DataSet("数据集名");
```

例如，创建数据集对象dsStu，代码如下：

```
DataSet dsStu = new DataSet();
dsStu.DataSetName = "Student";
```

或

```
DataSet dsStu = new DataSet("Student");
```

DataSet对象的常用属性和方法

属 性 方 法	说　明
DataSetName属性	获取或设置DataSet对象的名称
Tables属性	获取数据集的数据表集合
Clear方法	删除DataSet对象中所有表
Copy方法	复制DataSet的结构和数据，返回与本DataSet对象具有相同结构和数据的DataSet对象

2. 数据表（DataTable）对象的创建

定义数据表DataTable对象的语法格式为：

```
DataTable 对象名 = new DataTable();
```

或

```
DataTable 对象名 = new DataTable("数据表名");
```

例如，创建数据表对象dtStu，代码如下：

```
DataTable dtStuInfo = new DataTable();
dtStuInfo.TableName="StuInfo";
```

或

```
DataTable dtStuInfo = new DataTable("StuInfo");
```

DataTable对象的常用属性和方法

属 性 方 法	说　明
Columns属性	获取数据表的所有字段
DataSet属性	获取DataTable对象所属的DataSet对象
DefaultView属性	获取与数据表相关的DataView对象
PrimaryKey属性	获取或设置数据表的主键
Rows属性	获取数据表的所有行
TableName属性	获取或设置数据表名
Clear()方法	清除表中所有的数据
NewRow()方法	创建一个与当前数据表有相同字段结构的数据行

DataTable添加到DateSet中

- 创建好的数据表对象，可以添加到数据集对象中，代码如下：

```
dsStu.Tables.Add(dtStuInfo);
```

3. 数据列(DataColumn)对象的创建

- DataTable对象中包含多个数据列，每列就是一个DataColumn对象。定义DataColumn对象的语法格式为：
  - DataColumn 对象名 = new DataColumn();
- 或
  - DataColumn 对象名 = new DataColumn("字段名");
- 或
  - DataColumn 对象名 = new DataColumn("字段名",数据类型);

DataColumn对象的常用属性和方法

属 性	说 明
AllowDBNull	设置该字段可否为空值。默认为true
Caption	获取或设置字段标题。若为指定字段标题，则字段标题与字段名相同
ColumnName	获取或设置字段名
DataType	获取或设置字段的数据类型
DefaultValue	获取或设置新增数据行时，字段的默认值
ReadOnly	获取或设置新增数据行时，字段的值是否可修改。默认值为false

说明：通过DataColumn对象的DataType属性设置字段数据类型时，不可直接设置数据类型，而要按照以下语法格式：

对象名.DataType = System.Type.GetType("数据类型"));

示例 DataColumn的创建

例如，创建数据列对象**stuNoColumn**，代码如下：

```
DataColumn stuNoColumn= new DataColumn();
stuNoColumn.ColumnName=" StuNo";
stuNoColumn.DataType
   =System.Type.GetType("System.String");
```

或

```
DataColumn stuNoColumn= new
   DataColumn("StuNo",
System.Type.GetType("System.String"));
```

DataColumn添加到DataTable中

- 创建好的数据列对象，可以添加到数据表对象中，代码如下：

```
dtStuInfo.Columns.Add(stuNoColumn);
```

4. 数据行（DataRow）的创建

定义数据行DataRow对象的语法格式为：

DataRow 对象名 = DataTable对象.NewRow();

注意:DataRow对象不能用New来创建，而需要用数据表对象的NewRow方法创建。

例如，为数据表对象dtStu添加一个新的数据行，代码如下：

DataRow dr = dtStuInfo.NewRow();

访问一行中某个单元格内容的方法为：

DataRow对象名["字段名"]或DataRow对象名[序号]

DataRow对象的常用属性和方法

属 性 方 法	说 明
RowState属性	获取数据行的当前状态，属于DataRowState枚举型，分别为：Add、Delete、Detached、Modified、Unchanged
AcceptChanges方法	接受数据行的变动
BeginEdit方法	开始数据行的编辑
CancelEdit方法	取消数据行的编辑
Delete方法	删除数据行
EndEdit方法	结束数据行的编辑

5. 视图对象(DataView)的创建

数据视图DataView是一个对象，它位于数据表上面一层，提供经过筛选和排序后的表视图。通过定制数据视图可以选择只显示表记录的一个子集，同时在一个数据表上可定义多个DataView。

定义DataView对象的语法格式为：

DataView 对象名= new DataView(数据表对象);

例如：

DataView dvStuInfo = new DataView(dtStuInfo);

DataView对象可以通过以下2个属性定制不同的数据视图。

- RowFilter属性：设置选取数据行的筛选表达式。
- Sort属性：设置排序字段和方式。

例如：

```
DataView dvStuInfo = new DataView
(ds.Tables("StuInfo"));
dvStuInfo.Sort = "StuNo desc"; //按StuNo字段降序排，如果要升序，将desc改为asc
dvStuInfo.RowFilter = "Name = '张三'";//筛选出姓名为张三的学生
```

示例 DateSet数据集的创建

【例8-13】在内存中的数据集中，创建一个数据表StuInfo，包含学号（StuNo 字符串型）、姓名（Name 字符串型）和性别（Sex 字符串型）。对于所建立的内存数据表StuInfo，编写程序逐行将数据填入该数据表，最后将数据绑定到页面的GridView控件上。

参见示例\第08章\SqlServerDemo\DataSetDemo.aspx

步骤1 创建数据集及数据表结构

```
DataSet dsStu = new DataSet();

//创建名为StuInfo表对象
DataTable dtStuInfo = new DataTable("StuInfo");
//将表对象添加到数据集对象中
dsStu.Tables.Add(dtStuInfo);

//创建名为StuNo列对象，类型为String
DataColumn columnStuNo = new DataColumn("StuNo", System.Type.GetType("System.String"));
//将创建好的列对象添加到表中
dtStuInfo.Columns.Add(columnStuNo);
DataColumn columnName = new DataColumn("Name", System.Type.GetType("System.String"));
dtStuInfo.Columns.Add(columnName);
DataColumn columnSex = new DataColumn("Sex", System.Type.GetType("System.String"));
dtStuInfo.Columns.Add(columnSex);
```

创建数据集

创建数据表

创建数据列

步骤2 添加表数据

```
//下面的代码为StuInfo表添加一些数据
string[] stuNo = new string[4] { "1001", "1002", "1003", "1004" };
string[] name = new string[4] { "张三", "李四", "王五", "李芳" };
string[] sex = new string[4] { "男", "男", "男", "女" };
for (int i = 0; i < stuNo.Length; i++)
{
    DataRow row = dtStuInfo.NewRow();
    row[0] = stuNo[i];
    row[1] = name[i];
    row[2] = sex[i];
    dtStuInfo.Rows.Add(row);
}
```

创建数据行

数据行添加到表

步骤3 数据集显示

```
//将StuInfo表的数据绑定到GridView1上
GridView1.DataSource = dsStu;//设置数据源
GridView1.DataMember = "StuInfo";//设置显示的表名
GridView1.DataBind();//绑定到GridView控件上
```

StuNo	Name	Sex
1001	张三	男
1002	李四	男
1003	王五	男
1004	李芳	女

附加功能-利用视图筛选子集

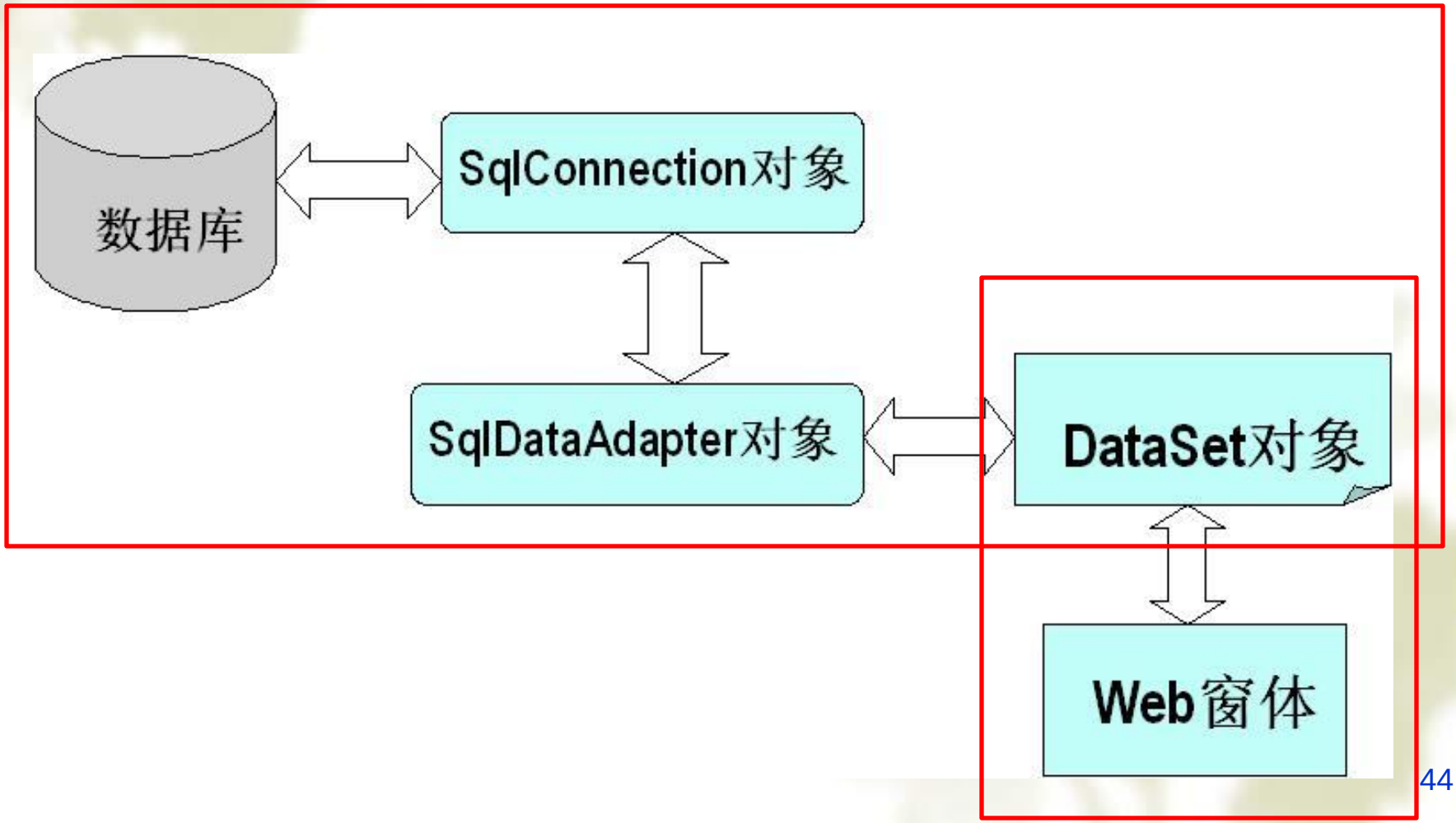
```
protected void Button1_Click(object sender, EventArgs e)
{
    //从Session变量中取出数据集
    DataSet dsStu = (DataSet)Session["dsStu"];
    //获取StuInfo表的默认视图
    DataView dvStuInfo = dsStu.Tables[0
    //定制视图筛选条件
    dvStuInfo.RowFilter = "Name='" + Te
    //将dvStuInfo视图的数据绑定到GridVie
    GridView2.DataSource = dvStuInfo;
    GridView2.DataBind();
}
```

StuNo	Name	Sex
1001	张三	男
1002	李四	男
1003	王五	男
1004	李芳	女

张三 Button

StuNo	Name	Sex
1001	张三	男

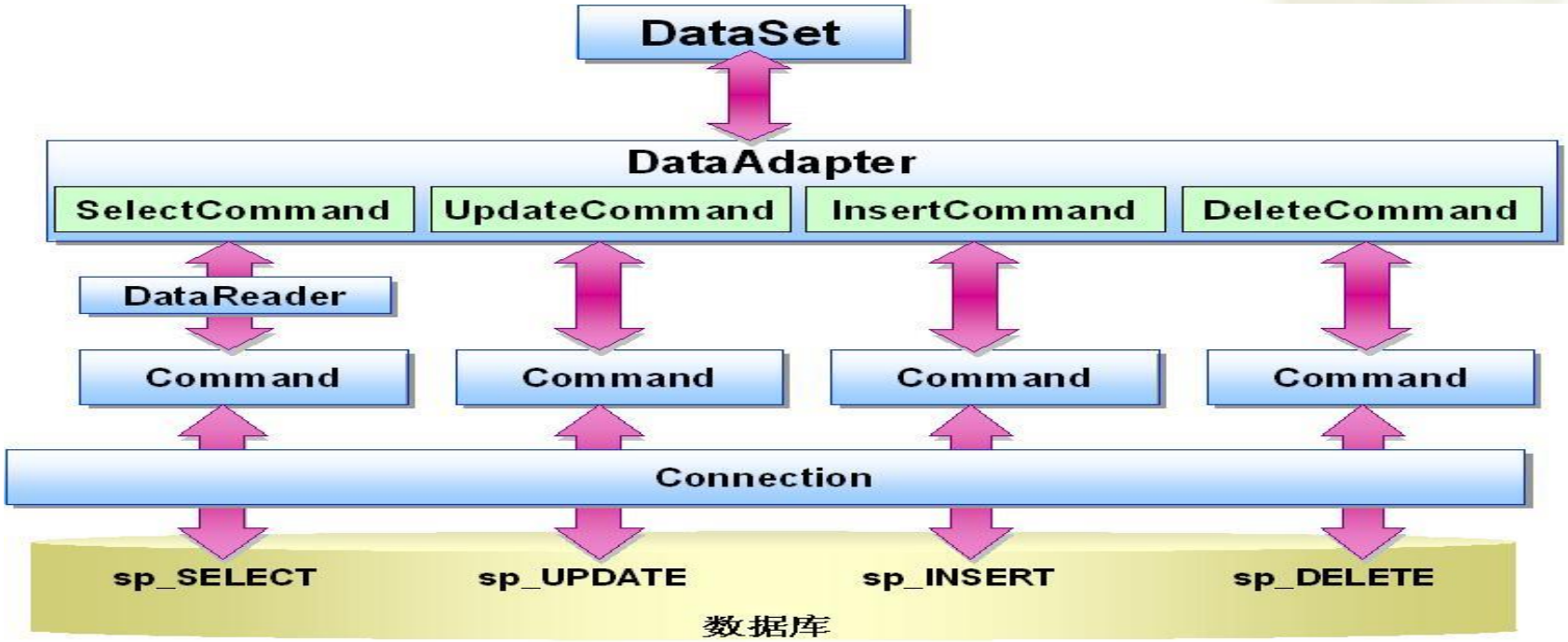
断开模式访问数据库



8.3.2 使用SqlDataAdapter对象执行数据库命令

- DataAdapter是一个特殊的类，其作用是数据源与DataSet对象之间沟通的桥梁。
- DataAdapter提供了双向的数据传输机制，它可以在数据源上执行Select语句，把查询结果集传送到DataSet对象的数据表（DataTable）中，还可以执行Insert、Update和Delete语句，将DataTable对象更改过的数据提取并更新回数据源。

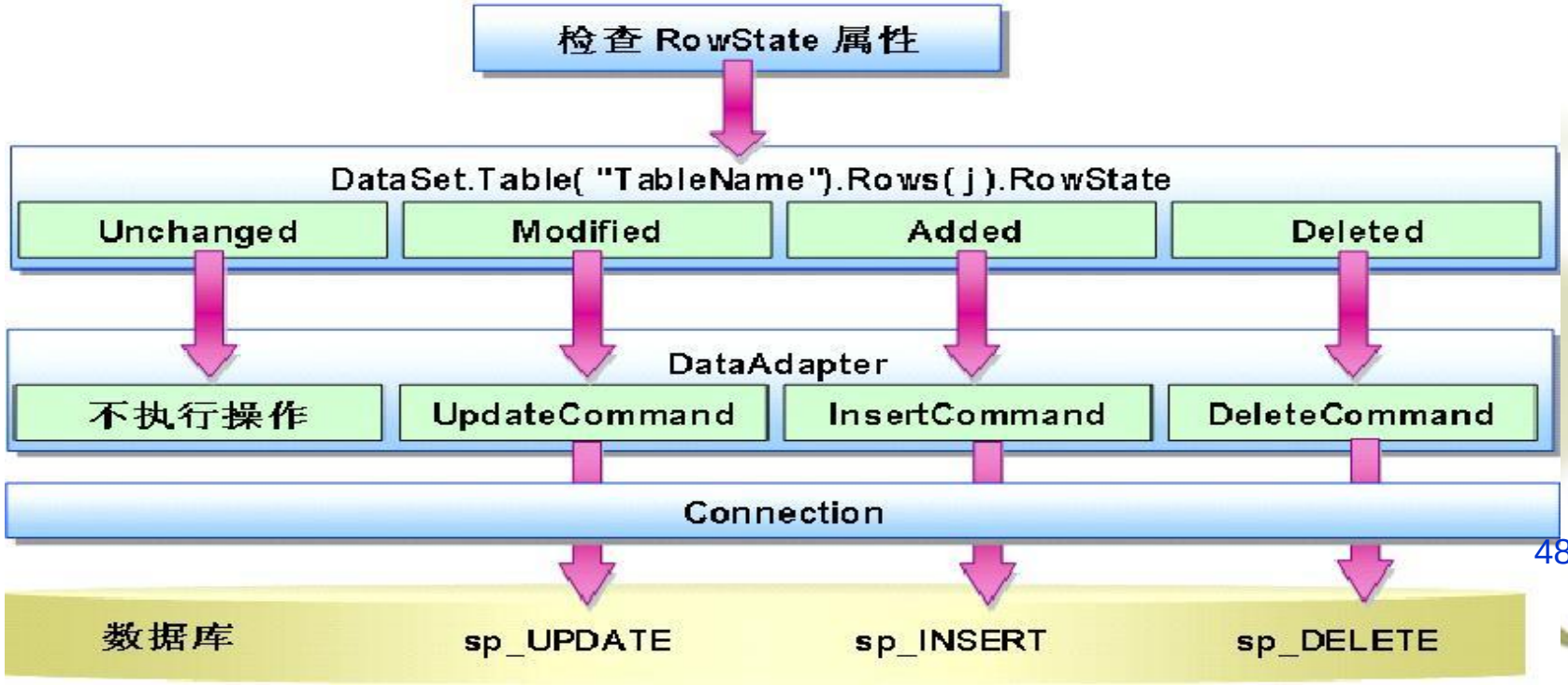
- **DataAdapter对象包含四个常用属性：**
  - **SelectCommad属性：是一个Command对象，用于从数据源中检索数据。**
  - **InsertCommand、UpdateCommand和DeleteCommand属性：也是Command对象，用于按照对DataSet中数据的修改来管理对数据源中数据的更新。**



DataAdapter对象的常用方法：

- **Fill方法**：调用**Fill方法会自动执行SelectCommand属性中提供的命令，获取结果集并填充数据集的DataTable对象**。其本质是通过执行SelectCommand对象的Select语句查询数据库，返回DataReader对象，通过DataReader对象隐式地创建DataSet中的表，并填充DataSet中表行的数据。

- **Update方法：调用InsertCommand、UpdateCommand和DeleteCommand属性指定的SQL命令，将DataSet对象更新到相应的数据源。在Update方法中，逐行检查数据表每行的RowState属性值，根据不同的RowState属性，调用不同的Command命令更新数据库。DataAdapter对象更新数据库示例图如图所示。**



创建SqlDataAdapter对象

定义SqlDataAdapter对象的方法有4种：

- SqlDataAdapter对象名= new SqlDataAdapter();
- SqlDataAdapter 对象名 = new SqlDataAdapter(SqlCommand对象);
- SqlDataAdapter 对象名 = new SqlDataAdapter("SQL命令", 连接对象);
- SqlDataAdapter 对象名 = new SqlDataAdapter("SQL命令", 连接字符串);

例如：

- SqlDataAdapter daStu = new SqlDataAdapter("select * from StuInfo", cnn);

SqlDataAdapter对象查询数据库

- 使用SqlDataAdapter查询数据库的步骤为：
  1. 创建数据库连接对象；
  2. 利用数据库连接对象和Select语句创建SqlDataAdapter对象；
  3. 使用SqlDataAdapter对象的Fill方法把Select语句的查询结果放在DataSet对象的一个数据表中或直接放在一个DataTable对象中；
  4. 查询DataTable对象中的数据

【例8-15】演示如何使用SqlDataAdapter对象查询数据库的数据。查询Student数据库中StuInfo表的信息，并在页面上显示。

```
//从web.config中读取连接字符串
string strCnn = ConfigurationManager.ConnectionStrings["StudentCnnString"].ConnectionString
//创建连接对象
using (SqlConnection cnn = new SqlConnection(strCnn))
{
    //创建DataAdapter对象，使用select语句和连接对象初始化
    SqlDataAdapter daStu = new SqlDataAdapter("select * from StuInfo", cnn);
    //创建DataSet对象
    DataSet dsStu = new DataSet();
    try
    {
        //调用Fill方法，填充DataSet的数据表StuInfo
        daStu.Fill(dsStu, "StuInfo");
        //将StuInfo表绑定到GridView控件上显示
        GridView1.DataSource = dsStu.Tables["StuInfo"];
        GridView1.DataBind();
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
}
```

思考：using的用法？

参见示例\第08章\SqlServerDemo\DataAdapter_Select.aspx

SqlDataAdapter对象增/删/改数据库

① 创建数据库连接对象。

② 利用数据库连接对象和Select语句创建SqlDataAdapter对象。

③ 根据操作要求配置SqlDataAdapter对象中不同的Command属性。如增加数据库数据，需要配置InsertCommand属性；修改数据库数据，需要配置UpdateCommand属性；删除数据库数据，需要配置DeleteCommand属性。

④ 使用SqlDataAdapter对象的Fill方法把Select语句的查询结果放在DataSet对象的一个数据表中或直接放在一个DataTable对象中。

⑤ 对DataTable对象中的数据进行增、删、改操作。

⑥ 修改完成后，通过SqlDataAdapter对象的Update方法将DataTable对象中的修改更新到数据库。

【例8-16】演示如何使用DataAdapter对象增加数据库的数据。设计页面，完成对StuInfo表中记录的添加。

【例8-17】演示如何使用SqlDataAdapter对象修改数据库的数据。设计页面，完成对StuInfo表中记录的修改。

【例8-18】演示如何使用SqlDataAdapter对象删除数据库的数据。设计页面，完成对StuInfo表中记录的删除。

参见示例\第08章\SqlServerDemo\DataAdapter_Update.aspx

示例-1 数据查询及显示

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack
    {
        //调用自定义
        gvStuInfo.D
```

使用DataAdapter更新数据:

学　　号:

姓　　名:

性　　别:

出生日期:

专　　业: 计算机应用

添加　修改　删除

StuNo	Name	Sex	Birth	MajorId
1	张三	男	1990/09/20 0:00:00	1
2	李四	男	1990/08/10 0:00:00	1
3	王五	男	1989/03/04 0:00:00	2
4	陈豪	男	1988/02/03 0:00:00	2
5	张庭	女	1991/05/06 0:00:00	3
6	李勇	男	1988/04/06 0:00:00	3
7	王燕	女	1990/05/12 0:00:00	4
8	赵倩	女	1990/05/12 0:00:00	4

```
/// <summary>
/// 使用SqlDataAdapter对
/// </summary>
/// <param name="sql">
/// <returns>以DataTabl
private DataTable FillT
{
    string strCnn = Con
    using (SqlConnectio
    {
        SqlDataAdapter
        DataTable dt =
        da.Fill(dt);
        return dt;
    }
}
```

```
ew控件上
Info ");
g"].ConnectionString;
```

示例-2 新增数据

```
//创建DataAdapter对象，使用select语句和连接对象初始化
SqlDataAdapter daStu = new SqlDataAdapter("select * from StuInfo", cnn);
//建立CommandBuilder对象来自动生成DataAdapter对象的Command对象
//否则就要自己编写InsertCommand、UpdateCommand、DeleteCommand命令
SqlCommandBuilder sbStu = new SqlCommandBuilder(daStu);
//创建DataTable对象
DataTable dtStuInfo = new DataTable();
//使用DataAdapter对象的FillSchema方法可以创建DataTable对象的结构
daStu.FillSchema(dtStuInfo, SchemaType.Mapped);
//上面这段代码也可写成daStu.Fill(dtStuInfo);
//增加新记录
DataRow dr = dtStuInfo.NewRow();
//给记录赋值
dr[0] = txtStuNo.Text.Trim();
dr[1] = txtName.Text.Trim();
dr[2] = txtSex.Text.Trim();
dr[3] = Convert.ToDateTime(txtBirth.Text.Trim());
dr[4] = dpMajor.SelectedValue;
dtStuInfo.Rows.Add(dr);
//提交更新
daStu.Update(dtStuInfo);
lblMsg.Text = "添加成功！";
//重新绑定数据
gvStuInfo.DataSource = FillTable("Select * from ViewStuInfo ");
gvStuInfo.DataBind();
```

```
//创建DataAdapter对象，使用select语句和连接对象初始化
SqlDataAdapter daStu = new SqlDataAdapter("select * from StuInfo", cnn);
//建立CommandBuilder对象来自动生成DataAdapter对象的Command对象
SqlCommandBuilder sbStu = new SqlCommandBuilder(daStu);
//创建DataTable对象
DataTable dtStuInfo = new DataTable();
//用Fill方法返回的数据，填充DataTable对象
daStu.Fill(dtStuInfo);
//设置dtStuInfo的主键，便用后面调用Find方法查询记录
dtStuInfo.PrimaryKey = new DataColumn[] { dtStuInfo.Columns["StuNo"] };
//根据txtStuNo文本框的输入查询相应的记录，以便修改
DataRow row = dtStuInfo.Rows.Find(txtStuNo.Text.Trim());
//如果存在相应记录，则修改并更新到数据库
if (row != null)
{
    //修改记录值
    row.BeginEdit();
    row[1] = txtName.Text.Trim();
    row[2] = txtSex.Text.Trim();
    row[3] = Convert.ToDateTime(txtBirth.Text.Trim());
    row[4] = dpMajor.SelectedValue;
    row.EndEdit();
    //提交更新
    daStu.Update(dtStuInfo);
    lblMsg.Text = "修改成功！";
    //重新绑定
    gvStuInfo.DataSource = FillTable("Select * from ViewStuInfo ");
    gvStuInfo.DataBind();
}
```

示例-4 删除数据

```
SqlDataAdapter daStu = new SqlDataAdapter("select * from StuInfo", cnn);
//定义DeleteCommand属性，自定义Delete命令，其中@StuNo是参数
daStu.DeleteCommand = new SqlCommand("delete from StuInfo where StuNo=@StuNo", cnn);
//定义@StuNo参数对应于StuInfo表的StuNo列
daStu.DeleteCommand.Parameters.Add("@StuNo", SqlDbType.VarChar, 8, "StuNo");
DataTable dtStuInfo = new DataTable();
//用Fill方法返回的数据，填充DataTable对象
daStu.Fill(dtStuInfo);
//设置dtStuInfo的主键，便用后面调用Find方法查询记录
dtStuInfo.PrimaryKey = new DataColumn[] { dtStuInfo.Columns["StuNo"] };
//根据txtStuNo文本框的输入查询相应的记录，以便修改
DataRow row = dtStuInfo.Rows.Find(txtStuNo.Text.Trim());
// 如果存在相应记录，则删除并更新到数据库
if (row != null)
{
    //删除行记录
    row.Delete();
    daStu.Update(dtStuInfo);
    lblMsg.Text = "删除成功！";
    gvStuInfo.DataSource = FillTable("Select * from ViewStuInfo ");
    gvStuInfo.DataBind();
}
else
{
    lblMsg.Text = "没有该记录！";
}
```

断开模式访问数据库的开发流程

① 创建SqlConnection对象与数据库建立连接；

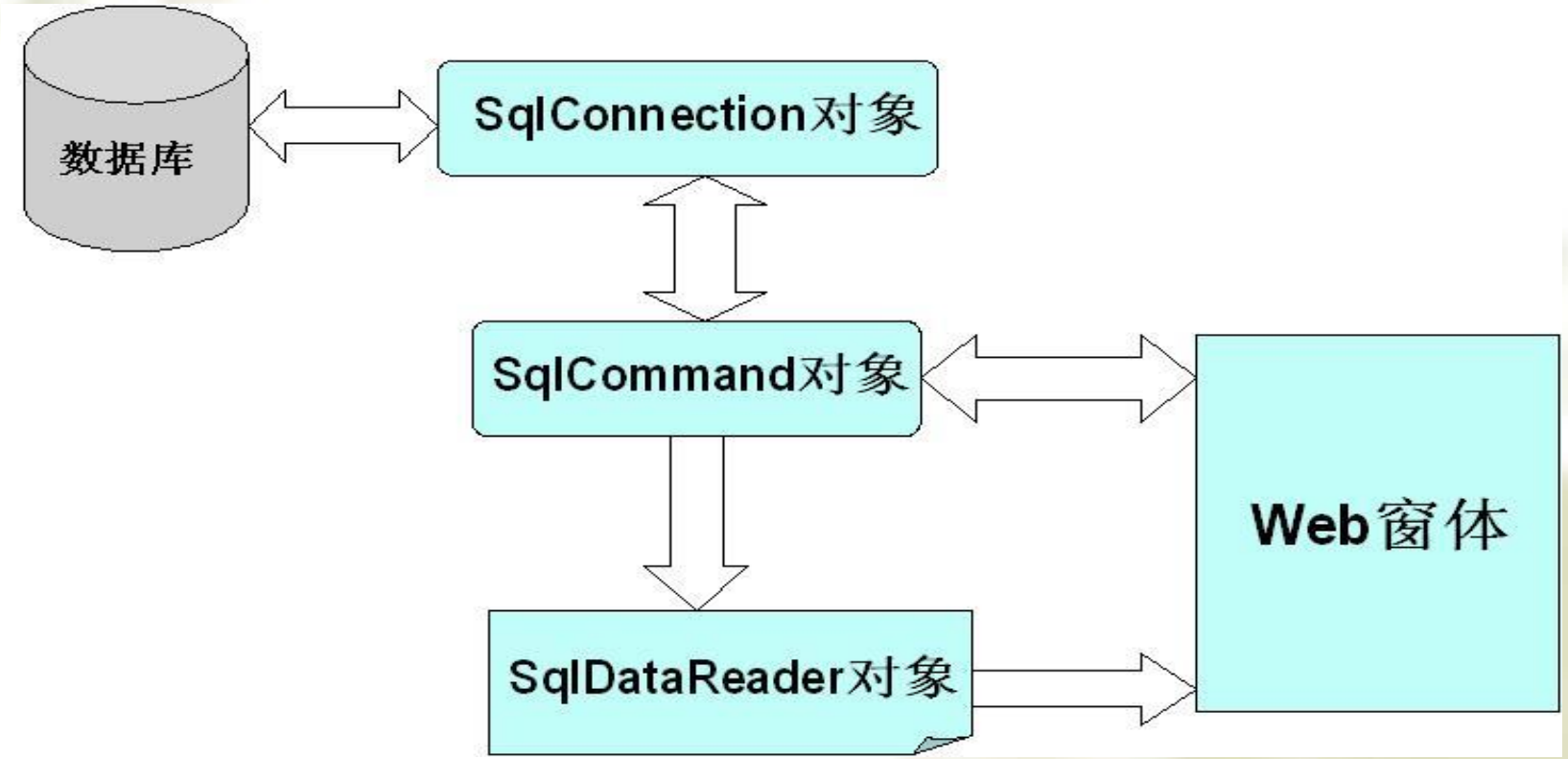
② 创建SqlDataAdapter对象对数据库执行SQL命令或存储过程，包括增、删、改及查询数据库等命令；

③ 如果查询数据库的数据，则使用SqlDataAdapter的Fill方法填充DataSet；如果是对数据库进行增、删、改操作，首先要对DataSet对象进行更新，然后使用SqlDataAdapter的Update方法将DataSet中的修改内容更新到数据库中。使用SqlDataAdapter对数据库的操作过程中，连接的打开和关闭是自动完成的，无需手动编码。

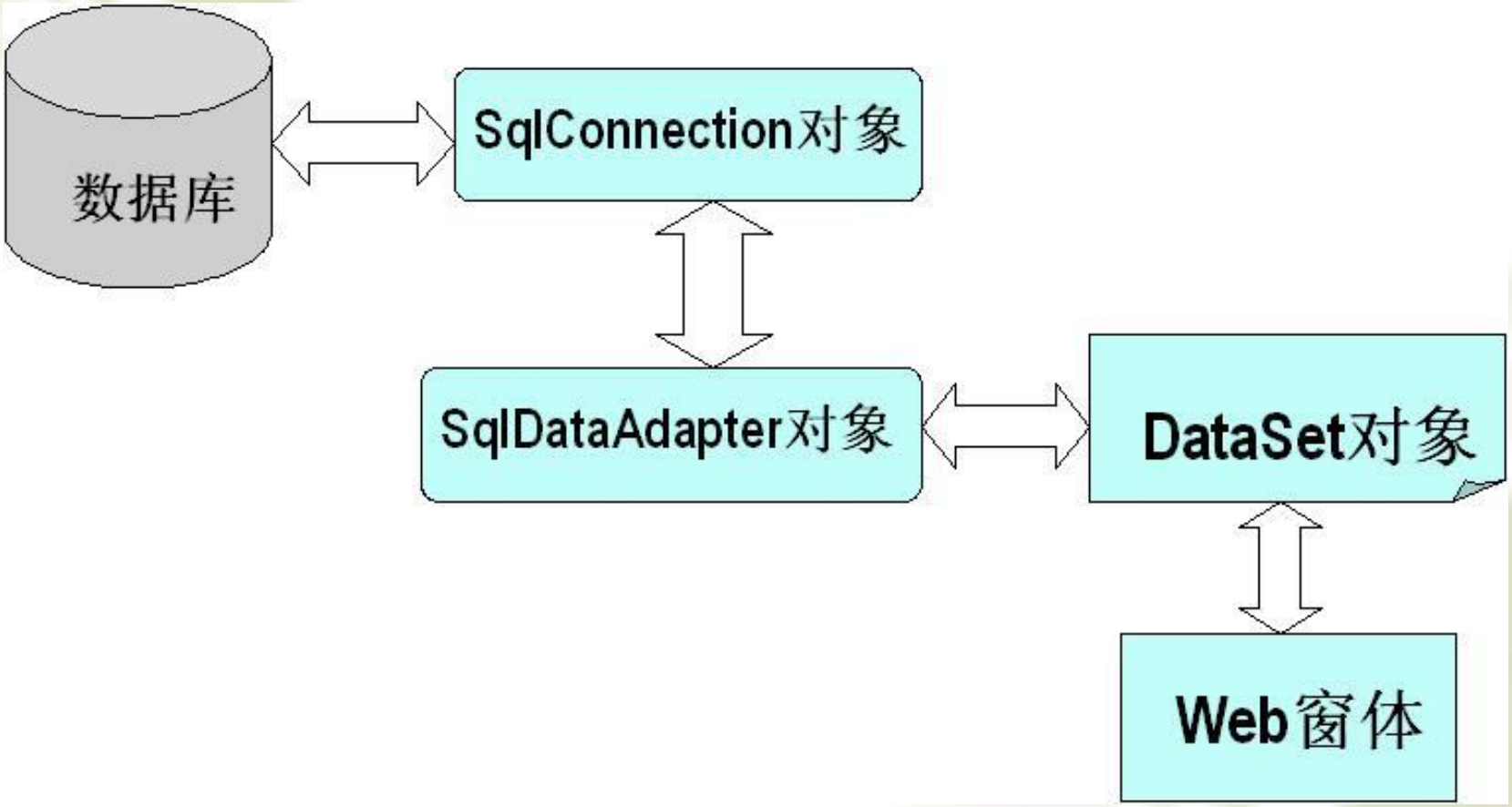
ADO.NET 总结

- 连接模式访问数据库
  - SqlConnection对象
  - SqlCommand对象
  - SqlDataReader对象
- 断开模式访问数据库
  - SqlConnection对象
  - SqlDataAdapter对象
  - DataSet和DataTable对象

连接模式访问数据库


数据库
SqlConnection对象
SqlCommand对象
Web窗体
SqlDataReader对象

断开模式访问数据库



本章结束!