الفصل الرابع عشر
استخدام مجموعات البيانات
DataSets

*بدلاً من تعديل البيانات واسترجاعها من قاعدة البيانات عن طريق الاتصال المستمر، يمكنك أخذ نسخة من جزء معين داخل قاعدة البيانات أو من قاعدة البيانات كلها إن أحببت ثم معالجة هذا الجزء فى نمط الانفصال وتحديث قاعدة البيانات مرةً أخرى. يمكنك أداء كل هذا من خلال كائن مجموعة البيانات **DataSet** وهو أهم سمات **ADO.NET**، وهو موضوع هذا الفصل.*

بانتهاء هذا الفصل ستتعرف على:

- *تركيب واستخدام مجموعة البيانات.*
- *استخدام الكائن **DataAdapter**.*
- *الوصول لقيم الحقول.*
- *تحميل أكثر من جدول.*
- *تخطيط الجداول والأعمدة.*
- *إضافة العلاقات والقيود.*
- *استخدام شبكة البيانات والمرشحات.*
- *التحقق من السجلات المعدَّلة.*
- *تحديث قاعدة البيانات.*
- *استخدام الإجراءات **Transactions**.*
- *تقليل الاتصال بالشبكة.*
- *مجموعات البيانات النوعية.*

تعتبر مجموعة البيانات **DataSet** أداة لتخزين وتحديث البيانات فى نمط عدم الاتصال **Disconnected Mode**، لذا فهى أهم أدوات **ADO.NET** على الإطلاق. وبيانات هذه المجموعة تكون دائماً آمنة وغير قابلة للضياع، كما أنها تحتوى على وظائف مشابهة لتلك الموجودة داخل قواعد البيانات المخزَّنة إلا أنها تقع داخل الذاكرة فقط. وقد تحدثنا فى الفصل السابق عن أهمية الكائن **DataSet** والفرق بينه وبين الكائن **Recordset** الموجود فى الإصدارات السابقة من **ADO**. سنقوم فى هذا الفصل بالتعرف على تركيب الكائن **DataSet** أو مجموعة البيانات وكيفية استخدامها لإجراء العمليات المختلفة على قاعدة البيانات، فكن معنا.

تركيب مجموعة البيانات

يحتوى تصنيف مجموعة البيانات **DataSet Class** على مجموعة من التصنيفات الأخرى التى تمكنك من التعامل مع البيانات الموجودة داخل مجموعة البيانات. ويتكون نموذج مجموعة البيانات أساساً من الجداول **Tables** والأعمدة **Columns** والعلاقات **Relations** والقيود **Constraints** وأخيراً الصفوف **Rows** التى تمثل البيانات الحقيقية، حيث تقوم العناصر الأخرى بتحقيق مخطط العلاقات. ويمكن بيان التركيب الهرمى لمجموعة البيانات كما يلى:

- يحتوى كائن **DataSet** على مجموعة من الجداول المشتقة من التصنيف **DataTable**.
- يحتوى كل تصنيف من التصنيفات **DataTable** على بيانات أحد الجداول الموجودة داخل قاعدة البيانات.
- يحتوى كائن **DataTable** بدوره على مجموعة من الأعمدة المشتقة من التصنيف **DataColumn**. فمثلاً يحتوى الكائن **DataColumn** المرتبط بالعمود **St_ID** داخل الجدول **Students** على بيانات هذا العمود بما فى ذلك اسم العمود ونوع بياناته بالإضافة إلى القيم والصفات والخصائص المرتبطة بهذا العمود (الحقل).

- يتم اشتقاق كائن العلاقات **Relations Object** من التصنيف **DataRelation** ويقوم بتخزين بيانات عن العلاقات بين الأعمدة الموجودة داخل جدول معين **DataTable**.
- تحتوى الصفوف **Rows** الموجودة داخل الجدول على نسخة من البيانات الحقيقة المخزَّنة داخل مجموعة البيانات **DataSet**.
- يتم اشتقاق الصفوف **Rows** من التصنيف **DataRow**. ويمكنك الوصول إلى الصفوف المختلفة الموجودة داخل الجدول من خلال الخاصية **Rows** المصاحبة للتصنيف **DataTable**.

تتشابه عملية الوصول إلى مجموعة البيانات **DataSet** كثيراً مع عملية الوصول إلى قاعدة البيانات نفسها وذلك لأن مجموعة البيانات تحتوى على جميع بيانات الجداول وخصائص الكائنات التى تنتمى لكل جدول. كما تحتوى أيضاً على معلومات عن العلاقات بين الجداول، لذا تعمل مجموعة البيانات فى حالة عدم الاتصال كما لو كانت فى حالة اتصال بالفعل.

استخدام مجموعة البيانات

الخطوة الأولى فى استخدام مجموعة بيانات **ADO.NET** هى تضمين المسمى **System.Data** وتضمين مزود بيانات **OLE DB** (أو **SQL**) كما فى الكود التالى:

```
Imports System.Data
Imports System.Data.OLEDB
```

وبعد ذلك يتم إنشاء مجموعة البيانات عن طريق إنشاء حالة من التصنيف **DataSet** ثم ربط مجموعة البيانات بقاعدة بيانات معينة بإنشاء سلسلة الاتصال المناسبة والتى تقوم بإجراء عملية الاتصال والتأكد من صحتها وذلك من خلال الكائن **OLEDBConnection** (أو **SQLConnection**) الذى يقوم بتأسيس عملية الاتصال. ولا يمكنك استدعاء كائن الاتصال مباشرةً عن طريق مجموعة البيانات وإنما يتم بدلاً من ذلك ربطها من خلال كائنات **DataAdapter** كما سنشرح لاحقاً. لإنشاء

كائن لمجموعة بيانات خالية لاستخدامها فى تخزين بيانات من مجموعة جداول داخل قاعدة البيانات، يمكنك استخدام الصيغة التالية:

```
DataSet ds = New DataSet()
```

استخدام الكائن DataAdapter

بعد الانتهاء من إنشاء كائن مجموعة البيانات، يجب نسخ القيم الموجودة داخل قاعدة البيانات إلى مجموعة البيانات **DataSet** وذلك من خلال الكائن **DataAdapter** الذى يعمل كموصل أو كوبرى بين قاعدة البيانات ومجموعة البيانات الممثلة فى الكائن **DataSet** وهذا يختلف بالطبع عن الكائن **Command** الذى يعمل كما لو كان دالة تقوم بإرجاع مجموعة من السجلات. يحتوى التصنيف **DataAdapter** على الوظائف **()Fill** و**()Update** التى تستخدم فى تبادل البيانات مع قاعدة البيانات المستخدمة، حيث تستخدم الوظيفة **()Fill** لملء مجموعة البيانات بالقيم الموجودة داخل جداول قاعدة البيانات بينما تستخدم الوظيفة **()Update** لتحديث قاعدة البيانات بالبيانات الموجودة بكائن مجموعة البيانات. كما يحتوى الكائن **DataAdapter** على مجموعة خصائص من أهمها **SelectCommand** و**InsertCommand** و**UpdateCommand** و**DeleteCommand** حيث يتم تبادل البيانات باستدعاء أى منها تبعاً لنوع الاستعلام المستخدم.

لتوضيح ذلك، دعنا نرى الكود التالى:

```
'Object Declaration
1.  Dim con As OleDbConnection
2.  Dim cmd As OleDbCommand
3.  Dim strSQL As String
4.  Dim adStudents As OleDbDataAdapter
5.  Dim ds As DataSet

'Initialize objects
6.  con = New OledbConnection("Provider= Microsoft.Jet.
    OLEDB.4.0; Data Source = d:\Students.mdb")
7.  adStudents  = New OleDbDataAdapter()
```

```
8.    ds = New DataSet()

'Setup Command
9.    strSQL = "SELECT * From Students"
10.   cmd = New OleDbCommand(strSQL,con)
11.   adStudents.SelectCommand = cmd

'Fill DataSet
12.   con.Open()
13.   adStudents.Fill(ds)
14.   con.Close()
```

وعن هذا الكود، نوضح ما يلى:

- يتم فى السطور من ١ إلى ٥ الإعلان عن الكائنات الأساسية وهى كائن الاتصال **OleDbConnection** وكائن الأمر **OleDbCommand** وكائن منظم البيانات **OleDbDataAdapter** وكائن مجموعة البيانات **DataSet** وكذلك سلسلة البيانات **strSQL** التى ستحتوى على عبارة **SQL**.
- يتم فى السطور من ٦ إلى ٨ تعيين القيم الابتدائية للكائنات السابقة وذلك بتعريف كائن الاتصال **con** وكائن منظم البيانات **adStudents** وكائن مجموعة البيانات **ds**.
- يتم فى السطور من ٩ إلى ١١ تعيين عبارة **SQL** المستخدمة ثم تعريف كائن الأمر وذلك بتمرير عبارة **SQL** وكائن الاتصال وأخيراً تخصيص كائن الأمر للخاصية **SelectedCommand** المصاحبة لمنظم البيانات.
- فى السطور من ١٢ إلى ١٤ يتم فتح عملية الاتصال من خلال الوظيفة **()Open** ثم ملء مجموعة البيانات بنتيجة عبارة **SQL** المحددة من قبل وذلك من خلال الوظيفة **()Fill** وأخيراً يتم قطع عملية الاتصال من خلال الوظيفة **()Close**.

الوصول لقيم الحقول

تختلف عملية الوصول للسجلات داخل مجموعة البيانات عن تلك المستخدمة مع قارئ البيانات والتى شرحناها فى الفصل السابق وذلك لعدم وجود مفهوم السجل الحالى أو

وظيفة الحركة المصاحبة. وبدلاً من ذلك يتم استخدام التجمعات فى ترتيب الجداول والصفوف وقيم الحقول. فالسطر التالى على سبيل المثال، يقوم بعرض محتويات الحقل الأول داخل السجل الأول بالجدول الأول:

```
MessageBox.Show(ds.Tables(0).Rows(0).Item(0).ToString)
```

كما يمكنك بهذه الطريقة التنقل داخل المجموعة بسهولة باستخدام المعرفات المسماة والدوارات وعبارة **With** كما فى الكود التالى الذى يقوم بطباعة الاسم الأول من كل صف على نافذة الإخراج.

```
With ds.Tables(0)
    For i = 0 To .Rows.Count-1
        Debug.WriteLine(.Rows(i).Item("St_fname"))
    Next
End With
```

تحميل أكثر من جدول

كما رأيت فإن عملية ملء مجموعة البيانات بجدول واحد أمر غاية فى السهولة. ولكن كى تحصل على الميزات العديدة لمجموعة البيانات، ربما تحتاج إلى تحميل أكثر من جدول إلى نفس المجموعة وإنشاء العلاقات بين هذه الجداول سواءً كانت هذه الجداول من قاعدة بيانات واحدة أو أكثر. ولأداء ذلك، يمكنك استخدام إحدى الطريقتين الآتيتين:

- *الطريقة الأولى:* استدعاء الوظيفة **()Fill** على أكثر من كائن **DataAdapter** باستخدام نفس مجموعة البيانات كمعامل لجميع الكائنات.
- *الطريقة الثانية:* تنفيذ إجراء مخزَّن يحتوى على أكثر من عبارة **SELECT** وفى هذه الحالة يتم استخدام كائن **DataAdaptr** واحد فقط.

وفيما يلى نضرب مثالاً على كلتا الطريقتين.

استخدام أكثر من أمر

يوضح الكود التالى كيفية إضافة السجلات من الجدولين **Students** و**Subjects** داخل نفس مجموعة البيانات:

```
'Object Declaration
```

```
Dim con As OleDbConnection
Dim cmdGetStudents As OleDbCommand
Dim cmdGetSubjects As OleDbCommand
Dim strSQL As String
Dim adStudents As OleDbDataAdapter
Dim adSubjects As OleDbDataAdapter
Dim ds As DataSet

'Initialize objects
con = New OleDbConnection("Provider= Microsoft. _
Jet. OLEDB.4.0;Data Source = d:\Students.mdb")
adStudents = New OleDbDataAdapter()
adSubjects = New OleDbDataAdapter()
ds = New DataSet()

'Setup Command
strSQL = "SELECT * From Students"
cmdGetStudents = New OleDbCommand(strSQL, con)
strSQL = "SELECT * From Subjects"
cmdGetSubjects = New OleDbCommand(strSQL, con)
adStudents.SelectCommand = cmdGetStudents
adStudents.TableMappings.Add("Table", "Students")
adSubjects.SelectCommand = cmdGetSubjects
adSubjects.TableMappings.Add("Table", "Subjects")

'Fill DataSet
con.Open()
adStudents.Fill(ds)
adSubjects.Fill(ds)
con.Close()
```

قمنا فى هذا الكود باستخدام كائن **DataAdapter** مع كل جدول. ولا جديد فى هذا الكود عن الكود السابق غير استخدام مفهوم تخطيط الجداول **Table Mappings** الذى يخبر منظم البيانات بحقول وجداول مجموعة البيانات التى يتم استخدامها عند ملء مجموعة البيانات، أو بمعنى آخر يمكنك استخدام تخطيط الجداول فى تسمية الحقول

والجداول بأسماء غير الموجودة فى قاعدة البيانات نفسها، وسوف نتعرف على هذا المفهوم بشئ من التفصيل بعد قليل.

استخدام إجراء مخزَّن

يتيح لك استخدام أكثر من منظم بيانات **DataAdapter**؛ كما فى الطريقة الأولى، ملء مجموعة البيانات من قواعد بيانات مختلفة. لكن إذا كانت الجداول داخل نفس قاعدة البيانات، يمكنك استخدام إجراء مخزَّن للحصول على بيانات الجداول فى نفس الوقت. لنقوم أولاً بإنشاء الإجراء المخزَّن كما يلى:

```
CREATE PROCEDURE spGetAllStudentInfo
AS
SELECT * FROM Students
SELECT * FROM Subjects
```

وكما ترى، يحتوى الإجراء المخزَّن على عبارتى **SELECT**، الأولى لاسترجاع جميع بيانات الجدول **Students** والأخرى لاسترجاع بيانات الجدول **Subjects**. والآن نقوم باستخدام هذا الإجراء بتعديل الكود المستخدم فى الطريقة الأولى كما يلى:

```
'Object Declaration
        Dim con As SQLConnection
        Dim cmdGetAll As SQLCommand
        Dim adAllInfo As SQLDataAdapter
        Dim ds As DataSet

'Initialize objects
        con = New SQLConnection("server=localhost;uid =
sa;pwd=;database=Students")
        adAllInfo  = New SQLDataAdapter()
        ds = New DataSet()

'Setup Command
        cmdGetAll = New
SQLCommand("spGetAllStudentInfo",con)
        cmdGetAll.CommandType =
CommandType.StoredProcedure
        adAllInfo.SelectCommand = cmdGetAll
```

```
        'Setup Mappings
        adAllInfo.TableMappings.Add("Table", "Students")
        adAllInfo.TableMappings.Add("Table1", "Subjects")

'Fill DataSet
        con.Open()
        adAllInfo.Fill(ds)
        con.Close()
```

يتم تخصيص الأسماء **Table** و**Table1** لمجموعتى السجلات الناتجة عن الإجراء المخزَّن، لذا تم استخدام التخطيط **Mapping** لتحويلها إلى أسماء معبرة. وعند العمل مع الجداول داخل منظم البيانات (الكائن **DataAdapter**)، تذكر دائماً أن عدد الجداول الموجود بالكائن يعتمد على عدد المجموعات الناتجة وليس عدد الجداول المحدد فى عبارة الاستعلام المستخدمة. فعلى سبيل المثال، تظهر العبارة التالية كجدول واحد فقط بالنسبة للكائن **DataAdapter** على الرغم من احتوائها على جدولين وذلك لأن الناتج عبارة عن مجموعة واحدة فقط:

```
SELECT * FROM Students St, Subjects Sb WHERE St.St_ID =
Sb.St_ID
```

فى حالة رجوع أكثر من حقل بنفس الاسم وفى نفس الجدول (**Table** أو **Table1** وهكذا) إلى الكائن **DataAdapter** (مثل الحقل **St_ID** فى العبارة السابقة)، يقوم الكائن **DataAdapter** بمعاملتها نفس معاملة الجداول (أى تصبح **St_ID** و**St_ID1** وهكذا). والآن لنتعرف على مفهوم تخطيط الجداول والأعمدة بشيءٍ من التفصيل.

تخطيط الجداول والأعمدة

يقصد بتخطيط الجداول والأعمدة (الحقول) تغيير أسماء الجدول أو العمود (الحقل) إلى اسم محبب إلى المستخدم غير الاسم المستخدم داخل الكائن **DataAdapter** والذى يكون **Table** و**Table1** وهكذا، حيث يشير الجدول إلى مجموعة البيانات الناتجة برمتها كما ذكرنا منذ قليل. ويكون ذلك بإنشاء رابط منطقى بين الجدول الموجود بمجموعة البيانات والجدول المماثل الموجود بقاعدة البيانات. وقد رأينا منذ قليل كيفية تخطيط

الجداول، لنرى الآن فى الكود التالى كيفية إنشاء تخطيط الأعمدة الموجودة داخل الجدول **Students** وذلك كما يلى:

```
OLEDBad.TableMappings[0].ColumnMappings.Add("St_ID",
"Student ID")
OLEDBad.TableMappings[0].ColumnMappings.Add("St_fname",
"Student First Name")
OLEDBad.TableMappings[0].ColumnMappings.Add("St_Lname",
"Student Last Name")
```

إذا قمت باستدعاء الوظيفة **()Fill** أو الوظيفة **()Update** (كما سنرى بعد قليل) دون أن تقوم بتحديد الجدول المصدر كما فى العبارة التالية:

```
OLEDBad.Fill(ds)
```

فإن كائن **DataAdapter** يقوم بالبحث عن جدول باسم **Table**. فإذا لم يجد هذا الجدول، يقوم بالبحث داخل مجموعة البيانات عن الجدول البديل الذى قمت بتعيينه أثناء عملية التخطيط.

إضافة العلاقات والقيود

بالإضافة إلى تخزين الجداول، فإن الكائن **DataSet** لديه المقدرة على تخزين العلاقات **Relationships** بين هذه الجداول وكذلك القيود **Constraints** تماماً كما فى قاعدة البيانات العادية. ومجموعة البيانات قابلة للتعديل أثناء الانفصال عن قاعدة البيانات، وبالتالى لا وجود لنظام إدارة قواعد البيانات لتطبيق القيود على بيانات الجداول مثل القيم الفريدة أو المفاتيح الأساسية. مع العلم أنه عند إضافة أكثر من جدول لمجموعة البيانات، لا يتم إضافة العلاقات أو القيود الموجودة على هذه الجداول تلقائياً وإنما يمكنك إضافتها من خلال الكود كما يلى:

```
'Set Up Column Objects For Easy Access
    Dim colIDSt, colIDSb As DataColumn
    colIDSt = ds.Tables("Students").Columns("St_ID")
    colIDSb = ds.Tables("Subjects").Columns("St_ID")

'Create Primary Keys
     Dim arStudKey(1) As DataColumn
```

```
    Dim arSubjKey(2) As DataColumn
    arStudKey(0) = colIDSt
    arSubjKey(0) = colIDSb
    arSubjKey(1) = ds.Tables("Subject").Columns("Sub_ID")
    ds.Tables("Students").PrimaryKey = arStudKey
    ds.Tables("Subject").PrimaryKey = arSubjKey

'Create Foreign Key For Cascading Delete
    Dim fkID As ForeignKeyConstraint
    fkID = New ForeignKeyConstraint("IDForKey", colIDSt,
                                                colIDSb)
    fkID.DeleteRule = Rule.Cascade
    ds.Tables("Subject").Constraints.Add(fkID)
```

وقد قمنا فى هذا الكود باستخدام الكائن **DataColumn** الذى يمثل حقلاً داخل جدول البيانات. يمكنك تعيين الخاصية **PrimaryKey** لكل جدول داخل مجموعة البيانات **DataSet** بمصفوفة من كائنات **DataColumn** لتعريف المفتاح الأساسى. ويعتبر الحقل **St_ID** هو المفتاح الأساسى داخل الجدول **Students** بينما يمثل الحقل **St_ID** والحقل **Subj_ID** معاً المفتاح الأساسى داخل الجدول **Subjects**. ولأن **St_ID** عبارة عن مفتاح أجنبى أيضاً داخل الجدول **Subjects** وللحفاظ على تكامل قاعدة البيانات، يجب حذف السجل من الجدول **Subject** إذا تم حذف سجل الطالب المصاحب من الجدول **Students**. ويتم ذلك تلقائياً بإنشاء الكائن **ForeignKeyConstraint** وتخصيصه بقاعدة الحذف من خلال الخاصية **DeleteRule**.

استخدام شبكة البيانات

يمكنك استخدام أداة تحكم "شبكة البيانات" **DataGrid** لإظهار محتويات مجموعة بيانات. لتوضيح ذلك، سنقوم بإنشاء تطبيق صغير يحتوى على شبكة البيانات التى تقوم بعرض بيانات الجدول **Students** والجدول **Subjects** الموجودين بقاعدة البيانات **Students.mdb**. تابع معنا الخطوات الآتية:

١. من داخل بيئة تطوير **Visual Studio 2008**، قم بإنشاء تطبيق نوافذى جديد

باسم مناسب وليكن **DataGrid**.

٢. أعد تسمية النموذج **Form1** إلى اسم مناسب وليكن **frmMain**.

٣. قم بإضافة زر أمر إلى النموذج وقم بتغيير اسمه إلى **btnLoad** وعنوانه إلى **Load Data**.

٤. قم بإضافة أداتى شبكة بيانات  إلى النموذج وقم بتغيير اسميهما إلى **grdStudents** و**grdSubjects**.

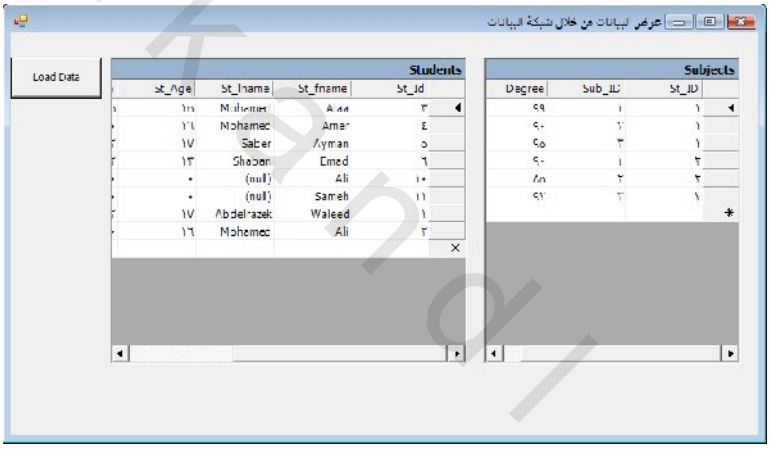
٥. انقر الزر **btnLoad** نقراً مزدوجاً ثم قم بتعديل كود تصنيف النموذج **FrmMain** كما يلى:

```
Dim myds As DataSet
Private Sub btnLoad_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnLoad.Click
    Dim con As OleDbConnection
    Dim cmdGetStud As OleDbCommand
    Dim cmdGetSubj As OleDbCommand
    Dim adStudInfo As OleDbDataAdapter
    Dim adSubjInfo As OleDbDataAdapter

'INITIALIZE OBJECTS
    con = New OleDbConnection("Provider= Microsoft.Jet.
    OLEDB.4.0;  Data Source = d:\Students.mdb")
    adStudInfo = New OleDbDataAdapter()
    adSubjInfo = New OleDbDataAdapter()
    myds = New DataSet()

 'SET UP COMMAND
    cmdGetStud = New OleDbCommand("Select * From
                                          Students", con)
    cmdGetSubj = New OleDbCommand("Select * From
                                          Subjects", con)
   adStudInfo.SelectCommand = cmdGetStud
   adSubjInfo.SelectCommand = cmdGetSubj

 'SET UP MAPPINGS
   adStudInfo.TableMappings.Add("Table", "Students")
```

```
    adSubjInfo.TableMappings.Add("Table", "Subjects")

  'FILL DATASET
     con.Open()
     adStudInfo.Fill(myds)
     adSubjInfo.Fill(myds)
     con.Close()
    'SET UP COLUMN OBJECTS FOR EASY ACCESS
       Dim colIDSt, colIDSb As DataColumn
       colIDSt = myds.Tables("Students").Columns("St_Id")
       colIDSb = myds.Tables("Subjects").Columns("St_Id")

    'CREATE PRIMARY KEYS
       Dim arStudKey(1) As DataColumn
       Dim arSubjKey(2) As DataColumn
       arStudKey(0) = colIDSt
       arSubjKey(0) = colIDSb
       arSubjKey(1) = myds.Tables("Subjects").
                                          Columns("Sub_ID")
       myds.Tables("Students").PrimaryKey = arStudKey
       myds.Tables("Subjects").PrimaryKey = arSubjKey

    'CREATE FOREIGN KEY FOR CASCADING DELETE
       Dim fkSt_ID As ForeignKeyConstraint
       fkSt_ID = New ForeignKeyConstraint("St_IDForKey",
                                          colIDSt, colIDSb)
       fkSt_ID.DeleteRule = Rule.Cascade
       myds.Tables("Subjects").Constraints.Add(fkSt_ID)

    'BIND TABLES TO GRIDS
       grdStudents.DataSource = myds
       grdStudents.DataMember = "Students"
       grdSubjects.DataSource = myds
       grdSubjects.DataMember = "Subjects"
       grdStudents.CaptionText = "Students"
       grdSubjects.CaptionText = "Subjects"

End Sub
```

ويحتوى هذا الكود كما ترى على توليفة من سطور الكود المختلفة التى ذكرناها من قبل، ما عدا السطور الأخيرة التى نقوم فيها بربط شبكة البيانات بالجدول المناسب وذلك بتخصيص مجموعة البيانات ككل إلى الخاصية **DataSource** المصاحبة لشبكة البيانات ثم تحديد الجدول الذى نرغب فى إظهاره من خلال تخصيصه إلى الخاصية **DataMember**، وأخيراً يتم تعيين عنوان لشبكتى البيانات من خلال الخاصية **CaptionText**.

قم الآن بتشغيل التطبيق ثم انقر زر **Load Data**، تلاحظ امتلاء كل شبكة بيانات بالجدول المصاحب (انظر شكل ١٤-١).



شكل ١٤-١عرض مجموعة البيانات من خلال شبكة البيانات

استخدام المرشحات

قمنا فى المثال السابق بتخصيص كائن **DataTable** عبارة عن جدول من جداول مجموعة البيانات للخاصية **DataMember** المصاحبة لشبكة البيانات وبالتالى يتم عرض جميع حقول الجدول مع إعطاء الصلاحيات الكاملة لعملية التعديل والحذف والإضافة. لكن ماذا لو رغبت فى عرض صفوف معينة بدلاً من الجدول بالكامل ؟ أو أردت تقييد المستخدم فى عملية التعديل ؟. يمكنك أداء كل ذلك من خلال المرشحات **Filters**، حيث يعمل

المرشح بنفس طريقة عمل العبارة **WHERE** دخل الاستعلام، فهو يقوم بإرجاع صفوف معينة بناءً على الشرط المحدد. سنقوم فى المثال الذى بين أيدينا بعرض صفوف معينة داخل الجدول **Students** بناءً على الاسم الأول للطالب. لأداء ذلك، تابع معنا الخطوات الآتية:

١. قم بإضافة زر أمر جديد إلى النموذج مع تغيير اسمه إلى **btnFilter** وعنوانه إلى **Filter**.

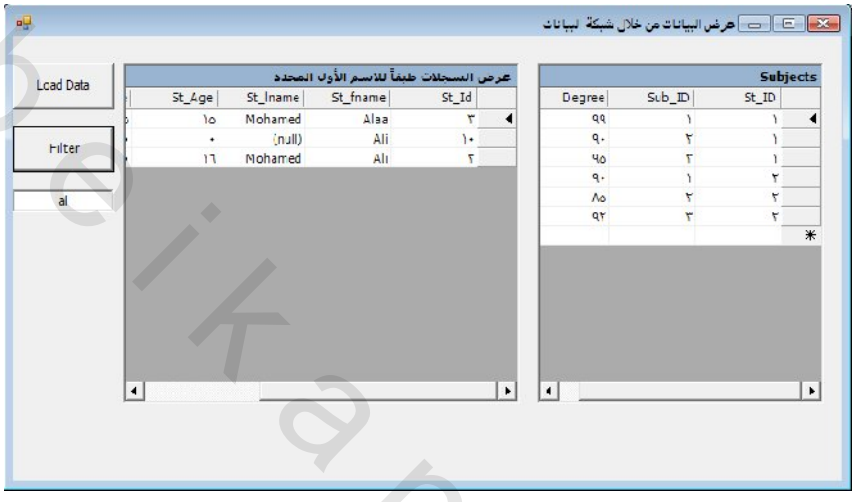
٢. قم بإضافة مربع نص أسفل الزر السابق مع تغيير اسمه إلى **txtFilter** وتأكد من عدم وجود نص بداخله وإلا قم بحذفه.

٣. انقر الزر الجديد نقراً مزدوجاً ثم قم بإدخال الكود التالى إلى إجراء حدث نقر الزر:

```
Private Sub btnFilter_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnFilter.Click

  'CLEAR CURRENT BINDING
     grdStudents.DataMember = ""
     grdStudents.DataSource = Nothing

  'MODIFY AND BIND DEFAULT TABLE VIEW
     With myds.Tables("Students")
        .DefaultView.RowFilter = "St_fname Like '" &
                                    txtFilter.Text & "%'"
        .DefaultView.AllowDelete = False
        .DefaultView.AllowEdit = False
        .DefaultView.AllowNew = False
        grdStudents.DataSource = .DefaultView
     End With
    grdStudents.CaptionText = "عرض السجلات طبقاً للاسم المحدد"
End Sub
```

٤. قم بتشغيل التطبيق ثم انقر زر **Load Data**، تلاحظ عرض الجدولين بجميع بياناتهما.

٥. انقر الحروف الأولى من الاسم الأول للطالب الذى ترغب فى عرضه داخل مربع النص ثم انقر زر **Filter**، تلاحظ تغير شبكة البيانات وعرض السجل المحدد فقط (انظر شكل ١٤-٢).



شكل ١٤-٢ ترشيح البيانات لعرض بيانات معينة

والآن انظر معى سريعاً إلى الكود السابق، تجد أننا قمنا أولاً بحذف الارتباط بين شبكة البيانات ومجموعة البيانات **Myds** وكذلك الجدول **Students** الموجود بهذه المجموعة. بعد ذلك قمنا باستخدام عرض افتراضى **DefaultView** داخل الجدول **Students** مع تعيين الشرط المطلوب أن يبدأ الاسم الأول للطالب بالحروف المحددة داخل مربع النص ثم قمنا بعد ذلك بمنع المستخدم من حذف وتعديل السجلات وإضافة سجلات جديدة وذلك بتخصيص القيمة **False** للخصائص **AllowDelete** و**AllowEdit** و**AllowNew** على الترتيب. وأخيراً يتم تخصيص هذا العرض كمصدر بيانات للشبكة من خلال الخاصية **DataSource** وتغيير عنوان الشبكة من خلال الخاصية **CaptionText**.

التحقق من السجلات المعدَّلة

يحتوى الكائن **DataTable**؛ الذى يمثل جدول من الجداول الموجودة بمجموعة البيانات، على مجموعة من الكائنات **DataRow** التى يمثل كل منها سجلاً داخل الجدول. يمكنك

حقيقةً تعديل محتويات مجموعة البيانات سواءً بتغيير قيم كائن DataRow معين أو من خلال أدوات ربط البيانات كأداة شبكة البيانات DataGridView. ومن إحدى السمات الهامة لمجموعة البيانات، القدرة على إخبارك بالصفوف أو السجلات التى تم تعديلها، وهذا الأمر فى غاية الأهمية لأن مجموعة البيانات تكون منفصلة عن مصدر البيانات. ومن هنا يجب أن تعرف هل تم إضافة سجلات جديدة أو تحديث السجلات الموجودة أو حذفها كى تتمكن بعد ذلك من عكس هذه التغييرات على مصدر البيانات الأصلى. فعلى فرض أن أحد جداول قاعدة البيانات يحتوى على ألف سجل، وقام المستخدم بتعديل سجل واحد فقط من هذه السجلات، فمن العبث إرسال السجلات كلها مرةً أخرى إلى قاعدة البيانات، والأولى فى هذه الحالة هو إرسال السجل المعدل فقط. وعامةً يمكنك التعرف على التغييرات التى تمت على مجموعة البيانات من خلال الخصائص والوظائف الموضحة بجدول ١٤-١ التالى.

جدول ١٤-١ الخصائص والوظائف المستخدمة للتحكم فى تغييرات مجموعة البيانات

الخاصية/الوظيفة	الاستخدام
الخاصية RowState	تحدد التغييرات التى تمت على الصف إن وجدت
الوظيفة GetChanges()	تقوم بإرجاع مجموعة بيانات جديدة (كائن DataSet) تحتوى على الصفوف المعدَّلة فقط
الوظيفة AcceptChanges()	تقوم بإرسال جميع التغييرات إلى مجموعة البيانات الحالية وتعديل الخاصية RowState
الوظيفة RejectChanges()	تقوم بإرجاع تغييرات مجموعة البيانات إلى حالتها الأولى أو الحالة التى تم قبلها استدعاء الوظيفة AcceeptChanges

لتوضيح استخدام هذه الوظائف من خلال المثال الذى بين أيدينا، تابع معنا الخطوات الآتية:

١. قم بإضافة ثلاثة أزرار من مربع الأدوات إلى النموذج مع تغيير أسمائها إلى btnAccept وbtnReject وbtnView وعناوينها إلى Accept Changes وReject Changes وView Changes على الترتيب.

٢. قم بإضافة الكود التالى الخاص بإجراءات أحداث نقر الأزرار الثلاثة:

```
Private Sub btnAccept_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnAccept.Click
    myds.AcceptChanges()
    grdStudents.DataSource = myds
    grdSubjects.DataSource = myds
End Sub

Private Sub btnReject_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnReject.Click
    myds.RejectChanges()
    grdStudents.DataSource = myds
    grdSubjects.DataSource = myds
End Sub

Private Sub btnView_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnView.Click
    Dim dsTemp As DataSet
    dsTemp = myds.GetChanges
    dsTemp.Tables("students").DefaultView.RowStateFilter =
                    DataViewRowState.ModifiedCurrent Or
                    DataViewRowState.Added
    dsTemp.Tables("subjects").DefaultView.RowStateFilter =
                    DataViewRowState.ModifiedCurrent Or
                    DataViewRowState.Added
    grdStudents.DataSource = dsTemp
    grdSubjects.DataSource = dsTemp
End Sub
```

٣. قم بتشغيل التطبيق ثم انقر زر Load Data، تظهر بيانات الجدولين داخل شبكتى البيانات.

٤. قم بتعديل الاسم فى أحد الصفوف وقم بحذف صف آخر ثم قم بإضافة صف جديد إلى نهاية الشبكة.

٥. انقر زر **View Changes**، تقوم شبكة البيانات بإظهار الصفوف التى قمت بتعديلها أو إضافتها بينما لا تقوم بإظهار الصفوف التى تم حذفها وإنما يمكنك الوصول إليها من خلال الكود.

٦. انقر زر **Reject Changes**، ترجع محتويات الشبكتين إلى الحالة الأولى قبل إجراء التغييرات.

٧. قم بتكرار الخطوة رقم ٤ ثم انقر زر **Accept Changes** لقبول التعديلات.

٨. انقر زر **View Changes**، تلاحظ عدم ظهور أى بيانات داخل شبكتى البيانات وذلك لأنك قبلت التعديلات فى الخطوة السابقة ولم تقم بأى تعديلات أخرى.

لاحظ أثناء الخطوات السابقة عملية الارتباط بين الجدولين الموجودين بشبكتى البيانات والتى قمنا بتعيينها من قبل من خلال الكود. فإذا قمت بحذف أحد الصفوف الموجودة بالجدول **Students**، يتم حذف الصفوف المصاحبة الموجودة بالجدول **Subjects**. وكذلك عدم القدرة على حذف أحد الصفوف من الجدول **Subjects** دون أن تقوم أولاً بحذف الصف المصاحب بالجدول **Students**. وهذا يرجع إلى مفهوم تكامل البيانات داخل قواعد البيانات.

تحديث قاعدة البيانات

لتحديث قواعد البيانات فى تطبيقات **ADO**، يمكنك تنفيذ عبارات **SQL** البسيطة أو استدعاء الإجراءات المخزَّنة. أما فى **ADO.NET** فهناك صعوبة فى استخدام هذه الطريقة داخل مجموعات البيانات **Datasets** لأنها تحتوى دائماً على بيانات فى حالة عدم الاتصال وبالتالى لا يمكن تنفيذ عبارات التحديث مباشرةً.

يمكنك فى **ADO** استخدام الوظيفة **()UpdateBatch** فى عملية التحديث التى تتم تلقائياً من خلال تعريف الصفوف ومقارنة القيم الموجودة ثم تطبيق التحديثات، كما أن عملية

التحديث تتم من خلال العبارات **Update** و**Insert** و**Delete**.

أما فى **ADO.NET** فيمكنك التحكم فى الوظيفة **()Update** من خلال الخصائص **InsertCommand** و**UpdateCommand** و**SelectCommand** و**DeleteCommand**. حيث تقوم بتخصيص الأوامر المناسبة لهذه الخصائص ثم يتم استدعاء الوظيفة **()Update** التى تختار أى منها تبعاً لنوع عملية التعديل.

وتحتوى الوظيفة **()Update** على معاملين، الأول عبارة عن مجموعة البيانات التى تحتوى على التغييرات التى أحدثها المستخدم بينما يحتوى المعامل الثانى على اسم الجدول الموجود بمجموعة البيانات والذى يحتوى على هذه التغييرات كما يلى:

```
adStudentInfo.Update(myds, "Students");
```

استخدام الإجراءات Transactions

يمكنك من خلال الإجراءات **Transactions** إرسال مجموعة من التحديثات مرةً واحدة، بحيث إذا حدث خطأ فى أحد هذه التحديثات يتم إلغاء العملية بالكامل. لتوضيح كيفية إنشاء الإجراء **Transaction**، قم بإضافة زر أمر جديد إلى النموذج باسم **btnTransaction** وعنوان **Transaction** ثم قم بإدخال الكود التالى بإجراء حدث نقر الزر الجديد:

```
1.  Private Sub btnTransaction_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs) Handles
    btnTransaction.Click
2.  Dim path As String = "d:\students.mdb"
3.  Dim strcon As String = "Provider=Microsoft.Jet.OLEDB.4.0;
    Data Source=" + path
4.  Dim con As OleDbConnection = New
    OleDbConnection(strcon)
5.  Dim Acmd As OleDbCommand = New
    OleDbCommand("SELECT * FROM Students", con)
6.  Dim Adocmd As OleDbDataAdapter = New
    OleDbDataAdapter("SELECT * FROM Students", con)
7.  Dim ds As DataSet = New DataSet()
8.  Try
```

```
9.          con.Open()
10.         Adocmd.Fill(ds, "Students")
11.         grdStudents.DataSource =
                                ds.Tables("Students").DefaultView
12.          Dim t As OleDbTransaction =
            con.BeginTransaction(IsolationLevel.ReadCommitted)
13.         Acmd.Transaction = t
14.         Acmd.CommandText = "Insert into Students
                           (St_ID,St_fname)" + "VALUES(10, 'Ali')"
15.         Acmd.ExecuteNonQuery()
16.         Acmd.CommandText = "Insert into Students
            (St_ID,St_fname)" + "VALUES(11, 'Sameh')"
17.         Acmd.ExecuteNonQuery()
18.         t.Commit()
19.         MessageBox.Show("Both the records were added.")
20.         Adocmd.Fill(ds, "Students")
21.         grdStudents.DataSource = Nothing
22.         grdStudents.DataSource =
23.                               ds.Tables("Students").DefaultView

24.     Catch ept As Exception
25.         MessageBox.Show(ept.ToString())
26.         MessageBox.Show("Neither records were written and
27.                                the transaction was rolledback.")

28.     Finally
29.          If (con.State = ConnectionState.Open) Then
30.              con.Close()
31.          End If
32.     End Try
33.     End Sub
```

وعن هذا الكود، نوضح ما يلى:

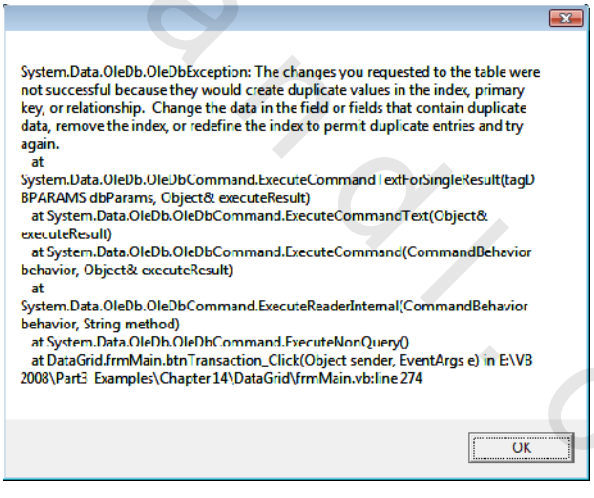
١. فى السطر رقم ١٢ يتم بدء الإجراء من خلال الوظيفة BeginTransaction().

٢. فى السطور من ١٣ إلى ١٧ يتم تعيين محتويات الإجراء وهى إضافة سجلين إلى الجدول Students.

٣. فى السطر رقم ١٨ يتم إرسال الإجراء من خلال الوظيفة **Commit** وفى السطور التالية يتم إظهار رسالة بذلك مع إعادة ملء شبكة البيانات لإظهار عملية التحديث على الشاشة.

قم الآن بتشغيل التطبيق ثم انقر زر **Transaction**، تظهر رسالة الاستثناء التى تخبرك باستحالة إضافة هذه السجلات لأنها تتعارض مع المفتاح الأساسى للجدول **Students** (انظر شكل ١٤-٣) وهذا يرجع بالطبع إلى العبارة **Try ... Catch ... Finally** المستخدمة فى معالجة الاستثناءات.

سنتعرف على معالجة الاستثناءات من خلال العبارة **Try ... Catch ... Finally** فى أحد الفصول المتقدمة من الباب الأخير لهذا الكتاب.



شكل ١٤-٣ رسالة الاستثناء الناتجة عن محاولة إضافة قيمة مكررة للمفتاح الأساسى

تقليل الاتصال بالشبكة

قمنا فى جميع الأمثلة السابقة بملء كائن مجموعة البيانات **DataSet** من المصدر الممثل فى قاعدة البيانات. وهذه الطريقة متبعة فى معظم التطبيقات وخاصةً حينما يرغب المستخدم فى تعديل البيانات الموجودة بقاعدة البيانات. ولكن إذا اقتصرت المهمة على إضافة بعض

الصفوف (السجلات) الجديدة إلى قاعدة البيانات، فبدلاً من إجراء الاتصال مع قاعدة البيانات لأخذ نسخة من البيانات إلى كائن مجموعة البيانات ثم إضافة الصفوف الجديدة وأخيراً الاتصال مرةً أخرى بقاعدة البيانات لإجراء عملية التحديث، يمكنك اختصار ذلك كله لتقليل زحمة الاتصال بالشبكة التى تحتوى على قاعدة البيانات، بأن تقوم بإنشاء مجموعة البيانات ومحتوياتها فى تطبيقك ثم إرسالها إلى قاعدة البيانات، وبذلك يتم الاتصال بقاعدة البيانات مرةً واحدةً فقط. لأداء ذلك، قم بإنشاء الجدول من خلال الكائن DataTable ثم قم بإضافته إلى مجموعة البيانات كما فى الكود التالى:

```
Dim TempTable As DataTable
TempTable = New DataTable()
TempTable.Columns.Add(New DataColumn("Sub_ID",
                                     GetType(Integer)))
TempTable.Columns.Add(New DataColumn("Sub_name",
                                     GetType(String)))
Dim TempDs As DataSet
TempDs = New DataSet()
TempDs.Tables.Add(TempTable)
```

وقد قمنا بإنشاء جدول واحد يحتوى على عمودين (حقلين) فقط، لذا ظهر الكود بسيطاً للغاية. لكن إذا أردت إنشاء أكثر من جدول يحتوى كل منها على العديد من الأعمدة وبينها علاقات معقدة، فإن كتابة الكود فى هذه الحالة سيكون شاق للغاية. فإذا كان لديك شبكة اتصال سريعة، وأردت الاتصال بقاعدة البيانات لسببٍ آخر، يمكنك استرجاع مخطط الجداول من خلال الكائن DataAdapter وباستخدام الوظيفة FillSchema كما فى الكود التالى:

```
adAllInfo.FillSchema(myds, SchemaType.Mapped)
```

يتسبب هذا الكود فى ملء كائن مجموعة البيانات بمخطط جميع الجداول الموجودة بقاعدة البيانات بما فى ذلك العلاقات والقيود الموجود بهذه الجداول، إلا أنها لا تقوم بإرجاع أية بيانات موجودة بقاعدة البيانات. ويمكنك فى هذه الحالة إضافة الصفوف من خلال التطبيق كما فى الكود التالى الذى يقوم بإضافة صف جديد للجدول Students:

```
Dim rowTemp As DataRow
Dim tblStudents As DataTable
tblStudents = myds.Tables("Students")

'CREATE THE NEW ROW
rowTemp = tblStudents.NewRow
rowTemp.Item("St_Id") = "20"
rowTemp.Item("St_fname") = "Sara"
rowTemp.Item("St_lname") = "Elsayed"

'ADD THE ROW
tblStudents.Rows.Add(rowTemp)
```

وقد استخدمنا الوظيفة **()NewRow** لنقل المخطط المناسب إلى الكائن **rowTemp**.

من الجدير بالذكر هنا أنك تستطيع إنشاء جدول جديد من خلال الكائن **DataTable** بمعزلٍ عن كائن مجموعة البيانات **DataSet** كما يمكنك استخدامه مع كلٍ من الوظيفة **()Fill** والوظيفة **()FillSchema** المصاحبتين للكائن **DataAdapter** كما فى الكود التالى:

```
tblTemp = New DataTable()
adStudents.Fill(tblTemp)
grdStudents.DataSource = tblTemp
```

فإذا كانت متطلبات بياناتك بسيطة، يمكنك فى هذه الحالة استخدام الكائن **DataTable** بدلاً من الكائن **DataSet** الأكثر تعقيداً بالطبع.

مجموعات البيانات النوعية

تتمثل أحد السمات الهامة داخل نطاق **.NET** فى إمكانية تمثيل كائنات مجموعات البيانات داخل برامجك فى صورة وحدات بيانات نوعية أو **Typed Entities**. لتوضيح ذلك، انظر معى إلى سطرى الكود التاليين:

```
FirstName = dsResults.Tables(0).Row(5).Item("St_fname")
FirstName = dsResultsTyped.Students(5).St_fname
```

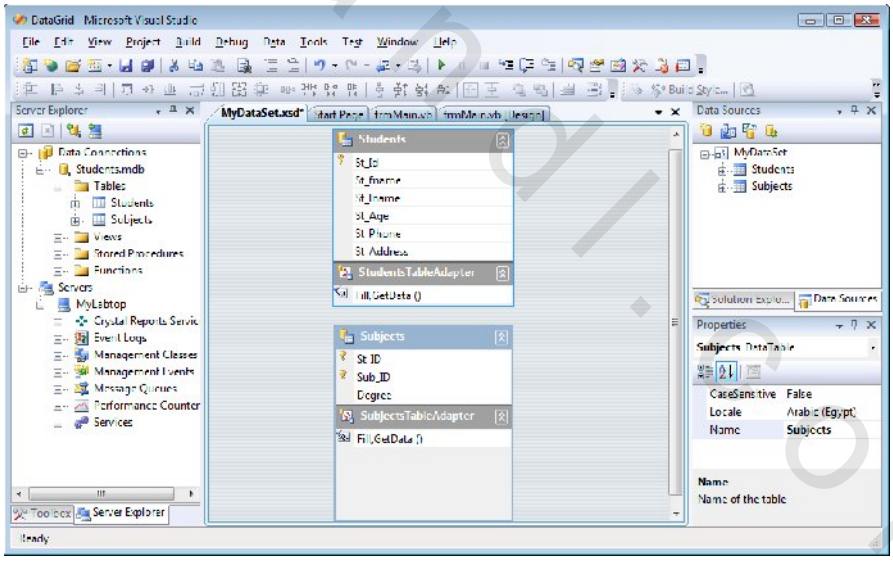
قمنا فى السطر الأول بالوصول إلى حقل الاسم الأول داخل الصف السادس فى جدول

Students بالطريقة التقليدية. أما فى السطر الثانى، فقد قمنا بالوصول إلى الحقل مباشرةً وهو ما يسمى الوصول المنوع Typed Access. وكى تتمكن من استخدام هذه الصيغة، يجب أن تقوم بإنشاء مجموعة البيانات أثناء التصميم. لأداء ذلك، تابع معنا الخطوات الآتية:

١. من نافذة مستكشف الحل، انقر اسم المشروع بزر الفأرة الأيمن ثم اختر Add و New Item من القوائم الناتجة، يظهر المربع الحوارى Add New Item.

٢. اختر DataSet من المربع Templates ثم قم بتعيين اسم مناسب لمجموعة البيانات وليكن MyDataSet ثم انقر زر Add، يتم إضافة ملف جديد باسم MyDataSet.xsd عبارة عن مجموعة بيانات جديدة إلى تطبيقك.

٣. من نافذة مستكشف الخادم Server Explorer قم بسحب الجدولين إلى نافذة الملف الجديد (انظر شكل ١٤-٤).



شكل ١٤-٤ إنشاء مجموعة البيانات النوعية

وبمجرد الانتهاء من تصميم مجموعة البيانات، يمكنك إنشاء حالة جديدة من هذا الكائن داخل التطبيق تماماً كما تفعل مع جميع الكائنات الأخرى هكذا:

```
Dim dsInfo As New MyDataset()
```

ولأن التصنيف **MyDataSet** مشتق أساساً من التصنيف **DataSet**، يمكنك ملء مجموعة البيانات **dsInfo** من قاعدة البيانات باستخدام نفس الوظائف التى ذكرناها من قبل.

إذا أردت تغيير اسم مجموعة البيانات النوعية الجديدة، قم بتعيين الاسم الجديد من خلال الخاصية **DataSetName** بمربع الخصائص المصاحب للكائن بنافذة مصمم **XML** المستخدمة فى تصميم هذا النوع من مجموعات البيانات.