الفصل الرابع عشر

تطوير تطبيقات قواعد بيانات أكثر تعقيداً

CREATING MORE COMPLEX DATABASE APPLICATIONS

أهداف الفصل:

- تطوير تطبيق قواعد بيانات يحتوي على علاقة ترافق "واحد-إلى-واحد".
- تطوير تطبيق قواعد بيانات يحتوي على علاقة ترافق "واحد-إلى-العديد".
- تطبيق العلاقة الشجرية parent-child داخل البيانات المترابطة.
- تطوير صنف المرافقة (Association Class) داخل تطبيق قواعد البيانات.

لقد تعلمنا داخل الفصل السابق كيف نطور الكائنات المستمرة باستخدام عدة أساليب، حيث قدمنا أصناف التعامل مع البيانات وإجراءاتها لكي ننشئ تطبيقات لديها القدرة على تخزين قيم صفات الكائنات واسترجاعها من وإلى الملفات التعاقبية (Sequential Files) باستخدام كل من الصنف StreamReader والصنف Streamwriter، وتعلمنا أسلوب النشر التسلسلي للكائنات وكيف يتم استخدامه لتخزين الكائنات واسترجاعها باستخدام الصنف Stream الذي يدعم كلاً من صيغة التخزين Binary وصيغة التخزين XML، كما تعلمنا كيفية تطوير تطبيقات مع بيانات مخزنة داخل قواعد بيانات علاقية (Relational Databases) باستخدام إجراءات أصناف التعامل مع قواعد البيانات وأوامر لغة الاستعلام الهيكلية SQL (الأمر Select، والأمر Insert، والأمر Update، والأمر Delete) لكي نسترجع سجلات من جدول بيانات علاقي ونضيفها ونعدلها ونحذفها وذلك بواسطة استخدام كل من الخاصية SelectCommand والخاصية InsertCommand والخاصية UpdateCommand والخاصية DeleteCommand المعرفة داخل الصنف OledbDataAdapter.

والآن عزيزي القارئ نمتلك مفاهيم جيدة جداً لكي نطور تطبيقات قواعد بيانات أكثر تعقيداً حيث طورنا سابقاً تطبيقات تتعامل مع جدول واحد فقط، وهذا لا يتفق مع الواقع حيث تحتوي قاعدة بيانات التطبيقات

الحقيقية على العديد من الجداول وأحياناً تصل إلى ألاف الجداول، وفي الحقيقة إن استخدام قواعد البيانات بدلاً من الأساليب الأخرى يقدم لنا ميزة الاستعلام عن معلومات مخزنة داخل أكثر من جدول، فعلى سبيل المثال ربما يكون مفيداً أن نسترجع بيانات جميع العملاء أو نسترجع بيانات عميل بواسطة رقم الهاتف الخاص به، ولكن الأكثر فائدة هو قائمة بيانات العملاء مشتملة على بيانات المراكب التي يمتلكونها.

سوف نتعلم في هذا الفصل كيف نطور تطبيقات قواعد بيانات تحتوي على عدة جداول حيث نطور أولاً علاقة الترافق "واحد-إلى-واحد" بين الصنف Customer والصنف Boat الموضحة في نموذج Class Diagram لنظام شركة برادشو مارينا، ثم نتعلم كيف نطور علاقة الترافق "واحد-إلى-العديد" بين الصنف Dock والصنف Slip، وأخيراً سوف نتعلم كيف نطور التطبيق الأكثر تعقيداً والذي يحتوي على صنف الترافق Lease.

تطوير تطبيق قواعد بيانات يحتوي على علاقة ترافق "واحد-إلى-واحد"
Implementing a One-to-One Association in a Database Application

يوضح الشكل رقم (١٤,١) نموذج Class Diagram لنظام شركة برادشو مارينا الذي راجعناه أثناء الفصل التاسع حيث يوضح أصناف مجال المشكلة المطلوب لشركة برادشو مارينا، وتذكر أن العلاقة بين كل من الصنف Customer والصنف Boat علاقة إلزامية في كلا الاتجاهين حيث يجب أن يملك العميل مركباً واحداً فقط ويجب أن يكون المركب مملوكاً لعميل واحد فقط.

تعلمنا في الفصل التاسع كيف نطور هذه العلاقة بين كائنات هذين الصنفين وكيف ننتقل من كائن إلى الكائن الآخر المرافق له، بينما سنتعلم الآن كيف نطور علاقة الترافق "واحد-إلى-واحد" باستخدام قاعدة البيانات العلاقية لتحقيق استمرارية الكائنات.

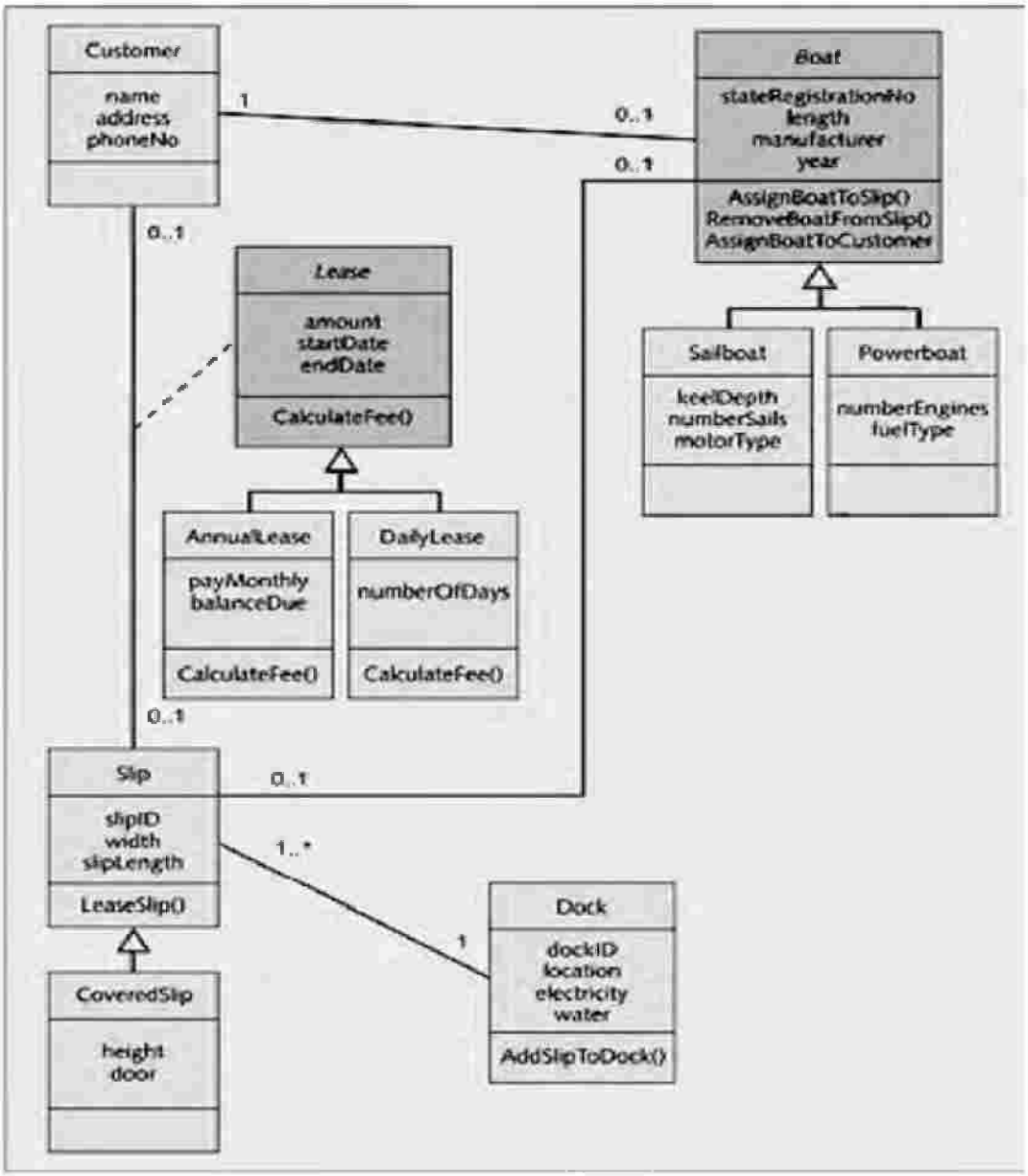
التعرف على قاعدة البيانات CustomerAndBoat
Understanding the CustomerAndBoat Database

ولتطوير تطبيق قواعد بيانات باستخدام لغة فيجوال بيسك .نت، يجب علينا أولاً أن نفهم التركيب الداخلي لجداول قاعدة البيانات التي سيتم استخدامها.

ولفهم التركيب الداخلي لهذه الجداول، اتبع التالي:

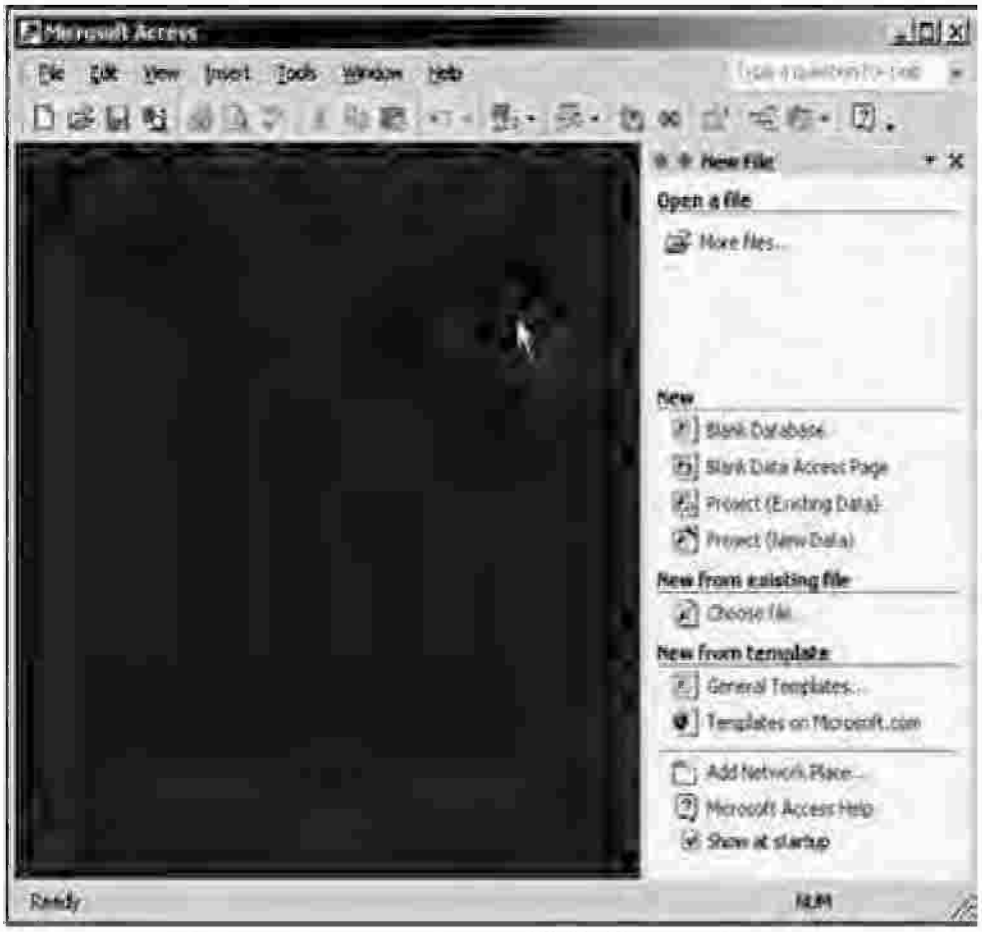
١- أنشئ المسار Chap14\Exercises داخل المجلد الرئيس الخاص بك إذا لم يكن موجوداً بالفعل.

٢- ابحث عن المجلد Ex01 داخل المجلد Chap14\Examples في ملفات الكتاب الخاصة بالطالب ثم قم بنسخه إلى المجلد Chap14\Exercises. يحتوي المجلد Example على المجلد الفرعي bin الذي يحتوي على الملف CustomerAndBoatDatabase.mdb.



الشكل رقم (١٤,١)- نموذج صنف براذهو ماربلا.

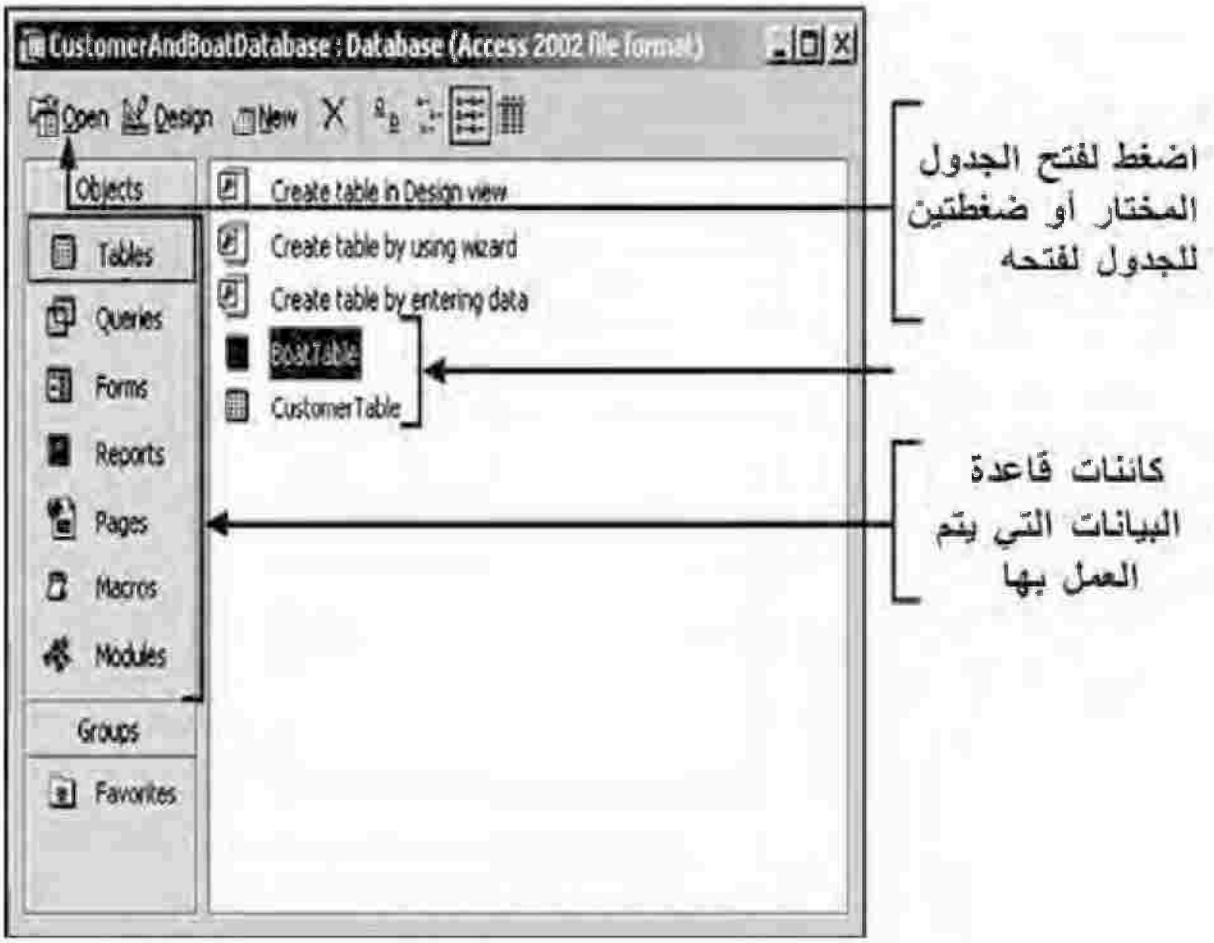
٣- قم بتشغيل برنامج Microsoft Access (اضغط على Start ثم Programs ثم All Programs ثم Microsoft Access، فتتذكر ستظهر نافذة برنامج Microsoft Access كما هو واضح في الشكل رقم ١٤.٢).



الشكل رقم (١٤.٢). فتح قاعدة بيانات في برنامج Microsoft Access 2003.

٤- اضغط على الرابط "More files" الموجود داخل الجزء "Open a file" المتواجد داخل النافذة "New File"، فتتذكر سيظهر الصندوق الحواري "Open File" تجول داخل مجلدات النظام الخاص بك للوصول إلى الملف CustomerAndBoatDatabase.mdb واختياره. ستظهر نافذة قاعدة البيانات التي تسرد جداول قاعدة البيانات (الجدول BoatTable والجدول CustomerTable) كما هو واضح في الشكل رقم (١٤.٣).

٥- اضغط ضغطاً مزدوجاً على الجدول BoatTable لكي ترى محتوياته، ولاحظ أن هذا الجدول يحتوي على خمسة حقول (أو أعمدة) كما هو واضح في الشكل رقم (١٤.٤) (الحقل StateRegistrationNo، والحقل BoatLength، والحقل Manufacturer، والحقل Year، والحقل CustomerPhoneNo).



الشكل رقم (١٤,٣). نافذة قاعدة بيانات العميل والمركب.

BoatTable : Table

StateRegistrationNo	BoatLength	Manufacturer	Year	CustomerPhoneNo
MO12345	26	Ranger	1976	765-4321
MO223344	24	Tracker	1996	467-1234
MO34561	40	Tartan	2001	123-4567
MO457812	19	Ranger	2001	587-4321
MO54321	30	Bayliner	2001	321-4567
MO98765	28	J-Boat	1986	467-1122

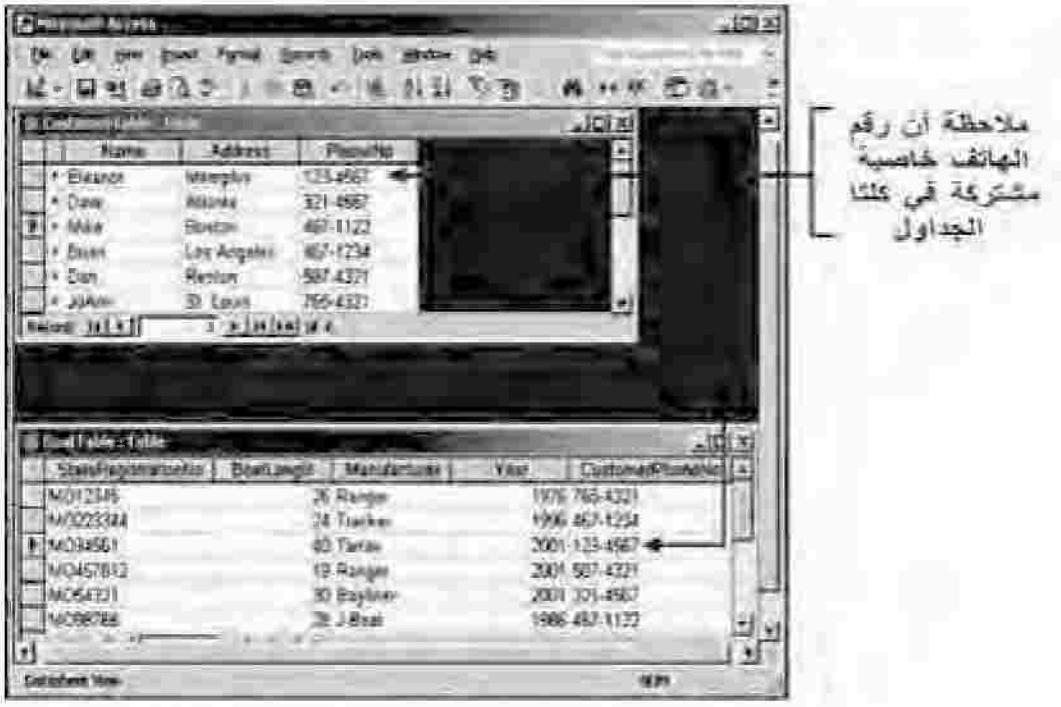
Record: 14 4 3 of 6

الشكل رقم (١٤,٤). محتويات جدول المركب.

٦- أغلق نافذة الجدول BoatTable ثم اضغط ضغطاً مزدوجاً على الجدول CustomerTable لكي ترى محتوياته، ولاحظ أن هذا الجدول يحتوي على ثلاثة حقول كما هو واضح في الشكل رقم (١٤،٥) (الحقل Name، والحقل Address، والحقل PhoneNo).

تذكر أن حقل المفتاح الرئيس (Primary Key) هو الحقل الذي يميز كل سجل عن الآخر داخل جداول قواعد البيانات العلائقية، مع العلم أن المفتاح الرئيس للجدول CustomerTable هو الحقل PhoneNo، وأن المفتاح الرئيس للجدول BoatTable هو الحقل StateRegistrationNo. ويجب أن تلاحظ أيضاً أن كلاً من الحقل PhoneNo في الجدول CustomerTable والحقل CustomerPhoneNo في الجدول BoatTable يحتويان بالضبط على القيم نفسها (انظر الشكل رقم ١٤،٦).

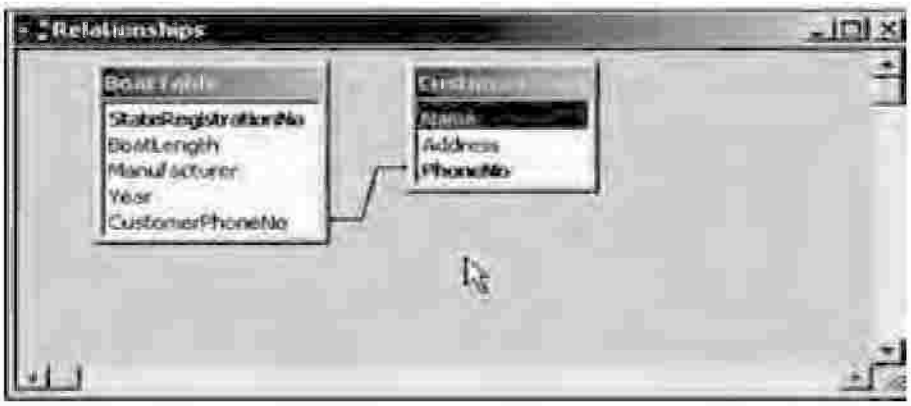
الشكل رقم (١٤،٥). محتويات جدول العميل.



الشكل رقم (١٤،٦). الحقول العامة لجدول المركب وجداول العميل.

غالباً تتشارك جداول قواعد البيانات العلائقية في حقل ما الذي يعمل على ربط المعلومات في الجدول الأول بمعلومات الجدول الآخر. فعلى سبيل المثال إذا أردنا أن نسترجع بيانات العميل الذي يملك المركب الذي يأخذ الرقم "MO34561" داخل الحقل StateRegistrationNo، فعندئذ يجب استرجاع قيمة رقم هاتف العميل (قيمة الحقل CustomerPhoneNo) الناظرة لسجل هذا المركب (القيمة 123-4567)، ثم البحث داخل المفتاح الرئيس (الحقل PhoneNo) للجدول CustomerTable عن الرقم 123-4567 لاسترجاع كل من اسم العميل وعنوانه. يطلق على الحقل CustomerPhoneNo في الجدول BoatTable اسم الحقل الأجنبي (foreign key) وهو الذي يوجد في جدول ما (الجدول BoatTable) ويوجد في جدول آخر ذات علاقة (الجدول CustomerTable) كمفتاح رئيس، وتستخدم المفاتيح الأجنبية لتطبيق مفهوم الحقول المشتركة التي تسمح لنا بربط معلومات جداول مختلفة، ويمكن رؤية الحقول المشتركة في برنامج Microsoft Access داخل نافذة العلاقات (Relationships window) كما هو واضح في الشكل رقم (١٤,٧)، ولفتح هذه النافذة اضغط على الزر "Relationships" داخل صندوق الأدوات أو اختر القائمة Tools ثم الأمر Relationships.

يشير الخط الذي يصل الحقل CustomerPhoneNo في الجدول BoatTable بالحقل PhoneNo في الجدول CustomerTable بأن هذه الحقول تربط معلومات هذين الجدولين.



الشكل رقم (١٤,٧). نموذج العلاقات لجدول المركب وجدول العميل.

استخدام اللغة SQL لربط جداول قواعد البيانات
Using the SQL to Join Tables in the Database

لقد تعلمنا في الفصل الثالث عشر كيف نستخدم أوامر لغة SQL لاسترجاع معلومات داخل جدول واحد وإضافتها وحذفها وتعديلها. وللتعامل مع بيانات جدولين في الوقت نفسه فإن لغة SQL تستخدم كلاً من المفتاح الأجنبي والمفتاح الرئيس لربط معلومات جدول بمعلومات جدول آخر والذي تطلق عليه ربط الجداول

(Tables Joining)، فعلى سبيل المثال توضح أوامر لغة SQL التالية كيفية البحث وعرض رقم تسجيل المركب StateRegistrationNo واسم شركة تصنيع المركب Manufacturer لجميع المراكب من الجدول BoatTable مع كل من أسماء أصحاب المراكب وعناوينهم وأرقام هواتفهم.

```
SELECT StateRegistrationNo, Manufacturer, Name, Address, PhoneNo
FROM BoatTable, CustomerTable
WHERE CustomerPhoneNo= PhoneNo
```

تذكر أن الجزء الأول Select في الأمر السابق يحدد حقول قاعدة البيانات المراد استرجاعها، ولأن هذه الحقول ستأتي من جدولين مختلفين فسيتم كتابة أسماء هذين الجدولين بعد الكلمة FROM ثم يحدد الجزء WHERE شرط ربط الجدولين وذلك عن طريق مساواة الحقل CustomerPhoneNo من الجدول BoatTable بالحقل PhoneNo من الجدول CustomerTable. وبالمثل يُستخدم الأمر التالي لعرض اسم مالك المركب وعنوانه الذي يأخذ رقم التسجيل MO98765.

```
SELECT StateRegistrationNo, Name, Address
FROM BoatTable, CustomerTable
WHERE CustomerPhoneNo= PhoneNo
AND StateRegistrationNo= 'MO98765'
```

يحتوي الجزء WHERE للأمر السابق على كل من شرط الربط وشرط تساوي القيمة MO98765 للمفتاح الرئيس للجدول BoatTable، وبشكل عام يستخدم الجزء WHERE لكتابة أي عدد من الشروط مع العلم أنك سوف تستخدم العديد من هذا النوع من أوامر لغة SQL عند معالجة بيانات جداول متعددة داخل قاعدة بيانات علاقية.

إنشاء كائن اتصال مشترك لقاعدة البيانات
Establishing a Common Connection to the Database

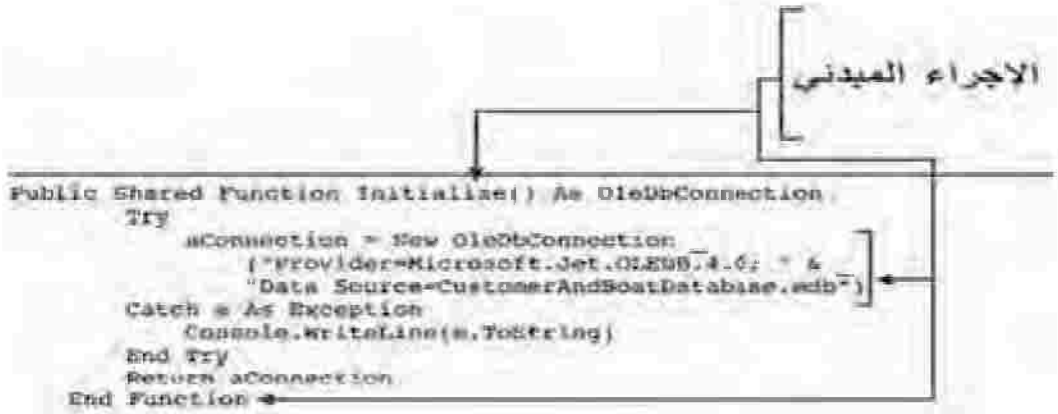
لقد تعلمنا في الفصل الثالث عشر كيف ننشئ أصناف التعامل مع البيانات حيث أنشأنا الصنف CustomerDA لمعالجة بيانات الجدول CustomerTable المتواجد داخل قاعدة البيانات واسترجاعها، وتذكر أن هذا الصنف احتوى على كائن اتصال بقاعدة البيانات CustomerDatabase حيث كان الصنف الوحيد الذي يستخدم هذا الاتصال، ولوجود صنفين يتعاملان مع البيانات في هذا المثال (صنف التعامل مع بيانات المركب والآخر لبيانات العملاء)، فيجب أن يكون هناك اتصال مشترك لكلا الصنفين، ولإنجاز اتصال مشترك بقاعدة البيانات على مستوى جميع أصناف التعامل مع البيانات نحتاج أن نعرف الصنف CustomerAndBoatDatabaseConnection المسؤول عن إدارة الاتصال بقاعدة البيانات، ولكن تذكر أنه يجب أن نشير إلى الفضاء المسمى System.Data.OleDb

قبل تعريف هذا الصنف باستخدام الأمر Imports، ثم تعريف متغير إشارة للصنف OleDbConnection كما هو واضح في الأوامر التالية:

```
Imports System.Data.OleDb
Public Class CustomerAndBoatDatabaseConnect
    Shared aConnection As OleDbConnection
```

ثم يأتي تعريف الإجراء Initialize الذي ينجز الاتصال بقاعدة البيانات (يتشابه هذا الإجراء بالإجراء Initialize المستخدم في الفصل الثالث عشر)، ولأن كائن الاتصال سيكون مشتركاً بين أصناف البرنامج، فيجب تعريفه من النوع Shared (يوضح الشكل رقم ١٤.٨ تعريف الإجراء Initialize). يوجد ملف قاعدة البيانات داخل مجلد التطبيق الحالي (المجلد bin)؛ لذلك لا نحتاج أن نكتب المسار كاملاً ومن ثم قمنا بكتابة اسم الملف أمام العامل Data Source فقط (مع العلم أن جميع ملفات قواعد البيانات الخاصة بهذا الفصل ستكون داخل المجلد bin)، وربما يطلب منك إسناد المسار الطبيعي لقاعدة البيانات كاملاً. إن المسار الطبيعي لقاعدة بيانات هذا المثال هو:

```
"Data Source=C:\Chap14\Exercises\Ex01\Example1\bin\CustomerAndBoatDatabase.mdb"
```



الشكل رقم (١٤.٨). الإجراء الميداني لصنف CustomerAndBoatDatabaseConnect.

إن الإجراء الوحيد الذي نريد أن نضيفه الآن إلى الصنف هو الإجراء Terminate.

```
Public Shared Sub Terminate()
        Try
            aConnection.Close()
        Catch e As Exception
            Console.WriteLine(e.ToString)
        End Try
    End Sub
```

تعديل توصيف الصنف Customer
Modifying the Customer Class

ولإنشاء تطبيق قاعدة بيانات يتعامل مع كل من الجدول CustomerTable والجدول BoatTable نحتاج أن نعدل تعريف الصنف Customer الوارد في الفصل الثالث عشر لكي نعكس علاقة الربط بين الجدولين. نضيف أولاً مؤشراً للصنف Boat كصفة للصنف Customer لكي يتمكن كائن الصنف Customer من التفاعل مع كائن الصنف Boat المرافق له، ثم نسند قيمة ابتدائية "Nothing" لهذا المؤشر داخل إجراء إنشاء الصنف Customer، كما نعرف إجراءي مرور لإسناد قيمة هذه الصفة واسترجاعها، وبالنسبة للاتصال بقاعدة البيانات سوف نستخدم كائن الاتصال المشترك المعرف داخل الصنف CustomerAndBoatDatabaseConnection. يوضح الشكل رقم (١٤,٩) سرداً جزئياً للتعديلات التي تمت على الصنف Customer.

تقديم الصنف BoatDA
Introducing the BoatAD Class

يجب أن يتوفر لدى تطبيق CustomerAndBoatDatabase إمكانية الإضافة والحذف والتعديل والبحث عن سجلات داخل الجدول BoatTable، ومن ثم نحتاج أن نعرف صنف التعامل مع البيانات BoatDA الذي يتشابه مع الصنف CustomerDA؛ ولذلك يبدأ تعريف الصنف BoatDA بالأمر Imports لكل من الفضاء المسمى System.Data.OleDb (لدعم الاتصال بقاعدة البيانات) والفضاء المسمى System.Collections (لدعم التعامل مع ArrayList)، ثم تعريف متغير إشارة للصنف Boat وتعريف المصفوفة Boats من النوع ArrayList (يستخدم بواسطة الإجراء GetAll) ثم تعريف متغيرات محلية لتخزين قيم صفات الصنف Boat ومتغير إشارة للصنف OleDbConnection كما هو واضح في الأوامر التالية:

```
Imports System.Data.OleDb
Imports System.Collections
Public Class BoatDA
    Shared boats As New ArrayList()
    Shared aBoat As Boat

    ' Declare variable for connection to database
    Shared aConnection As OleDbConnection

    ' Declare variables for Boat attribute values
    Shared stateRegistrationNo, manufacturer As String
    Shared length As Single
    Shared year As Integer
    Shared phoneno As String
```

```
' Customer Class with added methods for DA Class
' Chapter 14
Imports System.Data.OleDb
Public Class Customer
    ' Attributes
    Private name As String
    Private address As String
    Private phoneNo As String
    ' Reference variable for Boat instances
    Private theBoat As Boat

    ' Constructor (3 parameters)
    Public Sub New(ByVal aName As String, ByVal anAddress As String, _
        ByVal aPhoneNo As String)

        name = aName
        address = anAddress
        phoneNo = aPhoneNo
        SetBoat(Nothing)
    End Sub

    '-----------Begin Data Access Shared Methods-----
    Public Shared Sub Initialize(ByVal c As OleDbConnection)
        CustomerDA.Initialize(c)
    End Sub
      ...
    Public Shared Sub Terminate()
        CustomerDA.Terminate()
    End Sub
    '------------End Data Access Shared Methods--------

    ...
     ' Get accessor methods
     Public Function GetBoat() As Boat
         Return theBoat
     End Function

     ' Set accessor methods
     Public Sub SetBoat(ByVal aBoat As Boat)
         theBoat = aBoat
     End Sub

     ...

End Class
```

متغير إشارة للصنف Boat

إسناد متغير إشارة للصنف Boat بالقيمة Nothing

استخدام كائن الاتصال بقاعدة البيانات المشترك

الإجراء Get لاسترجاع كائن الصنف Boat

الإجراء Set لإسناد كائن الصنف Boat

الشكل رقم (١٤,٩). الصنف Customer.

الإجراء Initialize والإجراء Terminate للصنف BoatDA

Understanding the Initialise and Terminate Methods of the BoatDA Class

يتعامل كل من الإجراء Initialize والإجراء Terminate للصنف BoatDA مع كائن الاتصال الذي تم إنشاؤه بواسطة الصنف CustomerAndBoatDatabaseConnection حيث يستخدم الإجراء Initialize لفتح الاتصال، بينما يستخدم الإجراء Terminate لغلق الاتصال (انظر الشكل رقم ١٤.١٠).

```
Initialize class--Open connection
  Public Shared Sub Initialize(ByVal c As OleDbConnection)
      Try
          aConnection = c
          If aConnection.State.closed Then
              aConnection.Open()
          End If
      Catch e As Exception
          Console.WriteLine(e.Message.ToString)
      End Try
  End Sub

  ' Terminate class--Close the connection
  Public Shared Sub Terminate()
      Try
          aConnection.Close()
          aConnection = Nothing
      Catch e As Exception
          Console.WriteLine(e.Message.ToString)
      End Try
  End Sub
```

استخدم كائن اتصال قاعدة البيانات المشترك

الشكل رقم (١٤.١٠). الإجراءات المبدئية والمحددة لصنف BoatDA.

الإجراء Find والإجراء GettAll للصنف BoatDA

Understanding the Find and GettAll Methods of the BoatAD Class

يعرف الإجراء Find الأمر Select لاسترجاع سجل معين من الجدول BoatTable حيث يستقبل قيمة المفتاح الأساسي كمعامل للسجل المراد البحث عنه هكذا:

```
' Find method
Public Shared Function Find(ByVal key As String) As Boat
      aBoat = Nothing
      Dim dsBoat As New DataSet()
      ' Declare the SQL statement to find boat
      Dim sqlQuery As String = "Select StateRegistrationNo, BoatLength, " & _
            "Manufacturer, Year From BoatTable " & _
            "WHERE StateRegistrationNO = '" & key & "'"
```

ثم نعرف كائناً من الصنف OleDbDataAdapter لكي ينفذ استعلام الأمر Select واستقبال السجلات العائدة من الاستعلام لملء الجدول BoatTable الذي سيتم تعريفه داخل الكائن dsBoat للصنف Dataset، ولمعرفة هل تم إيجاد السجل المراد أم لا، يجب أن نفحص عدد السجلات العائدة، فإذا كانت أكبر من صفر فعندئذ ننشئ كائناً من الصنف DataRow وإسناده بأول سجل داخل الجدول BoatTable المعرف داخل الكائن dsBoat، ثم إعادته إلى الإجراء المستدعي للإجراء Find، أما إذا كان عدد السجلات يساوي صفراً فعندئذ ننشئ كائناً من صنف الاستثناء الخاص NotFoundExpetion ثم إلقاؤه إلى الإجراء المستدعي للإجراء Find هكذا:

```
' Execute the SQL query
Try ' Declare data adapter and fill the dataset table
   Dim adptBoat As New OleDbDataAdapter(sqlQuery, aConnection)
   adptBoat.Fill(dsBoat, "BoatTable")
   ' Check if dataset table contains any rows
   If dsBoat.Tables("BoatTable").Rows.Count > 0 Then
         ' Declare data row variable set to first row in the table
         Dim dr As DataRow = dsBoat.Tables("BoatTable").Rows(0)
         stateRegistrationNo = dr("StateRegistrationNo")
         length = dr("BoatLength")
         manufacturer = dr("Manufacturer")
         year = dr("Year")
         ' Create the boat instance
         aBoat = New Boat(stateRegistrationNo, length, manufacturer, year)
   Else
         Throw New NotFoundException("Not Found")
   End If
         dsBoat = Nothing
Catch e As OleDb.OleDbException
           Console.WriteLine(e.Message.ToString)
End Try
  Return aBoat
End Function
```

يتشابه عمل الإجراء GettAll مع عمل الإجراء Find، ولكن سيعيد الإجراء GetAll جميع سجلات الجدول BoatTable داخل المصفوفة boats كما هو واضح في الشكل رقم (١٤,١١).

```
' GetAll method
Public Shared Function GetAll() As ArrayList
    Dim daBoat As New DataSet()
    ' Define the SQL query statement for the GetAll method
    Dim sqlQuery As String = "SELECT StateRegistrationNo, BoatLength, " & _
        "Manufacturer, Year FROM BoatTable"

    ' Execute the SQL statement
    Try ' Declare data adapter and fill the dataset table
        Dim adptBoat As New OleDbDataAdapter(sqlQuery, aConnection)
        adptBoat.Fill(daBoat, "BoatTable")
        ' Check dataset table to see if it contains any rows
        If daBoat.Tables("BoatTable").Rows.Count > 0 Then
            Dim dr As DataRow
            ' Clear array list
            boats.Clear()
            ' Loop over each row of dataset table
            ' Extract the boat attributes
            For Each dr In daBoat.Tables("BoatTable").Rows
                stateRegistrationNo = dr("StateRegistrationNo")
                length = dr("BoatLength")
                manufacturer = dr("manufacturer")
                year = dr("Year")
                ' Create boat instance
                Dim aBoat As New Boat(stateRegistrationNo, length, _
                    manufacturer, year)
                boats.Add(aBoat)
            Next
        End If
    Catch e As Exception
        Console.WriteLine(e.Message.ToString)
    End Try
    Return boats
End Function
```

الشكل رقم (١٤,١١)- إجراء GetAll لصنف BoatDA.

الإجراء AddNew للصنف BoatDA

Understanding the AddNew Method of the BoatDA Class

يتشابه عمل الإجراء AddNew للصنف BoatDA مع الإجراء AddNew للصنف Customer المعرف داخل الفصل الثالث عشر حيث يبدأ الإجراء باسترجاع قيم صفات كائن الصنف Boat الذي تم استقباله كمعامل ثم يكمل الأوامر مثل إجراء الفصل الثالث عشر.

```
' AddNew-retrieves the boat attribute values
Public Shared Sub AddNew(ByVal aboat As Boat)
  ' Get boat attribute values
  stateRegistrationNo = aboat.GetStateRegistrationNo
  length = aboat.GetLength
  manufacturer = aboat.GetManufacturer
  year = aboat.GetYear
  phoneno = aboat.GetCustomer.GetPhoneNo
```

افترض معي للحظة أن القيم المستخلصة هي القيم MO33333 و 25 و Tartan و 1999 و 764-7414، إذن نستطيع أن نستخدم أمر Insert لإضافة هذه المعلومات هكذا:

```
INSERT INTO BoatTable
VALUES ('MO33333', 25, 'Tartan', '1999', '764-7414')
```

تذكر أن الأمر INSERT INTO يتطلب كتابة كل من اسم الجدول وأسماء حقول الجدول المراد تخزين بياناتها، ولكن إذا أردنا تخزين بيانات داخل جميع الحقول، فعندئذ يمكن كتابة هذه القيم فقط بترتيب حقول الجدول نفسه داخل أقواس الجزء VALUES، وكما هو واضح من الأوامر السابقة فقد تم كتابة القيم المراد إضافتها للجدول BoatTable داخل أقواس الجزء VALUES حيث فصلنا بين هذه القيم برمز الفاصلة (,)، وكما هو ملاحظ أن كل قيمة نصية محاطة بعلامة تنصيص (')، ومن الملاحظ أيضاً أن هذا المثال يبدو بسيطاً لأننا نضيف قيماً ثابتة، أما عند إضافة قيم متغيرات فعندئذ سيكون الأمر أكثر تعقيداً حيث يجب علينا استخدام معامل اللصق (&) واستخدام كل من علامة التنصيص المفردة وعلامة الفاصلة بحرص لكي ينتج أمر INSERT وكأنه أتى من قيم ثابتة كما هو واضح في الأوامر التالية:

```
' Declare a string SQL statement
Dim sqlInsert As String = "INSERT INTO BoatTable " & _
      "VALUES ('" & stateRegistrationNo & "', " & length & ", " & _
      " '" & manufacturer & " ', " & year & ", '" & phoneno & "')"
Console.WriteLine("The sql Statement is: " & sqlInsert)
```

ربما ينتج أخطاء عند تنفيذ الأوامر السابقة؛ ولذلك أدرجنا الأمر Console.WriteLine لعرض أمر SQL INSERT الناتج قبل تنفيذه لكي نرى هل تم تكوين الأمر كما نريد أم لا، وفي حال الاطمئنان إلى صحة تكوين الأمر SQL INSERT يمكن حذف الأمر Console.WriteLine، كما يمكن استبدال الأمر Console.WriteLine بوضع نقطة توقف (Breakpoint) بعد أمر SQL INSERT مباشرة، ثم نشير إلى المتغير sqlInsert لرؤية محتوياته والتأكد من صحة تكوينها.

ولكن يجب أن نتأكد من عدم وجود بيانات مركب بقيمة المفتاح الأساسي نفسها في قاعدة البيانات قبل تنفيذ أمر SQL INSERT (تذكر أن قاعدة البيانات تمنع تسجيل سجلين بقيمة المفتاح الأساسي نفسها)؛ ولذلك نستدعي الإجراء Find() (كما هو واضح في الأوامر التالية) لنتحقق من عدم وجود سجل للمركب بقيمة المفتاح الأساسي نفسها، فإذا أعاد الإجراء Find استثناءً يفيد عدم وجود هذا المركب فعندئذ يتم تنفيذ الأمر SQL INSERT

لإضافة بيانات المركب الجديد، أما إذا لم يعد الإجراء Find استثناءً فعندئذ تعيد الأوامر التالية استثناءً يفيد وجود المركب (الاستثناء DuplicateException) وعدم إمكانية إضافته.

```
    Dim adptBoat As New OleDbDataAdapter()
    Try ' Use Find method to see if boat exists
        Dim c As Boat = Find(phoneno)
        Throw New DuplicateException(" Boat Exists ")
    Catch e As NotFoundException
        Try ' Execute INSERT SQL statement if not found
            adptBoat.InsertCommand= New OleDbCommand(sqlInsert)
            adptBoat.InsertCommand.Connection = aConnection
            adptBoat.InsertCommand.ExecuteNonQuery()
        Catch sqle As Exception
            Console.WriteLine(sqle.ToString)
        End Try
    End Try
End Sub
```

الإجراء Update والإجراء Delete للصنف BoatDA

Understanding the Update and Delete Methods of the BoatDA Class

يستخلص كل من الإجراء Update والإجراء Delete بيانات كائن الصنف Boat الممرر إليهما ثم إنشاء أمر SQL المناسب. تذكر أن الأمر Update يحدد اسم الجدول المراد تعديله ثم الكلمة SET متبوعاً بأسماء الحقول المراد تعديل قيمها والقيم الجديدة لهذه الحقول، حيث لا يجب علينا سرد جميع حقول الجدول بل ندرج فقط الحقول المراد تعديل قيمها كما هو واضح في المثال التالي:

```
UPDATE BoatTable SET BoatLength = 30, Manufacturer = 'Tartan', Year = 2002
WHERE StateRegistrationNo = 'MO123345';
```

لاحظ أن قيم الحقول الرقمية لا تحاط بعلامة تنصيص فردية وعلى العكس يجب أن تحاط قيم الحقول النصية بعلامة تنصيص، ولاحظ أيضاً أننا نضع بعد كل حقل وقيمته فاصلة (,) للفصل بين الحقول، ولاحظ أيضاً أن الكلمة Year تمثل كلمة محجوزة داخل برنامج MS Access؛ ولذلك يجب أن تحاط بأقواس. يوضح الشكل رقم (١٤,١٢) أوامر الإجراء Update المعرف داخل الصنف BoatDA.

```
' Update method
Public Shared Sub Update(ByVal aBoat As Boat)
        ' Retrieve the Boat attributes
        stateRegistrationNo = aBoat.GetStateRegistrationNo
        length = aBoat.GetLength
        manufacturer = aBoat.GetManufacturer
        year = aBoat.GetYear

        ' Define SQL statement using boat reg key
        Dim sqlUpdate As String = "UPDATE BoatTable " & _
            "SET BoatLength = " & length & ", " & _
                "Manufacturer = '" & manufacturer & "', " & _
                "[Year] = " & year & " " & _
            "WHERE StateRegistrationNo = '" & stateRegistrationNo & "'"
        Dim adptBoat As New OleDbDataAdapter()
        ' See if this boat already exists
        ' NotFoundException is thrown by Find method
        Try
            Dim b As Boat = Boat.Find(stateRegistrationNo)
            ' If found execute the SQL statement
            adptBoat.UpdateCommand = New OleDbCommand(sqlUpdate)
            adptBoat.UpdateCommand.Connection = aConnection
            adptBoat.UpdateCommand.ExecuteNonQuery()
        Catch e As NotFoundException
            Console.WriteLine(e.Message.ToString)
        End Try
End Sub
```

يجب أن تكون الكلمة Year بين الفواصل مربعة

الشكل رقم (١٤،١٢): إجراء التعديل لصنف BoatDA.

ويماثل تذكر أن الأمر DELETE يحدد اسم الجدول المراد حذف سجلات منه وقيمة المفتاح الأساسي للسجل المراد حذفه. توضح الأوامر التالية أمر DELETE الذي يستخدم لحذف سجل من الجدول BoatTable.

```
DELETE FROM BoatTable WHERE StateRegistrationNo = 'MD123345';
```

يوضح الشكل رقم (١٤،١٣) تعريف الإجراء Delete المعرف داخل الصنف BoatDA.

```
' Delete method
Public Shared Sub Delete(ByVal aBoat As Boat)
        ' Retrieve the state registration no (key)
        stateRegistrationNo = aBoat.GetStateRegistrationNo

        ' Create the SQL statement
        Dim sqlDelete As String = "DELETE FROM BoatTable " &
            "WHERE StateRegistrationNo = '" & stateRegistrationNo & "'"
        Dim adptBoat As New OleDbDataAdapter()
        ' See if this boat already exists in the database
        ' NotFoundException is thrown by Find method
        Try
            Dim b As Boat = Boat.Find(stateRegistrationNo)
            ' If found, execute the SQL statement
            adptBoat.UpdateCommand = New OleDbCommand(sqlDelete)
            adptBoat.UpdateCommand.Connection = aConnection
            adptBoat.UpdateCommand.ExecuteNonQuery()
        Catch e As NotFoundException
            Console.WriteLine(e.Message.ToString)
        End Try
End Sub
```

الشكل رقم (١٤,١٣). إجراء الحذف لصنف BoatDA.

يستدعي كل من الإجراء Update والإجراء Delete الإجراء Find لكي يحدد ما إذا كانت قاعدة البيانات تحتوي على بيانات مركب برقم التسجيل المستخدم نفسه داخل أمر SQL أم لا، فإذا كان لا يوجد هناك مركب مسجل برقم التسجيل نفسه فسيتم تنفيذ أمر SQL (أمر Update وأمر Delete)، أما إذا كان المركب موجوداً من قبل فسيتم إلقاء استثناء من النوع NotFoundException.

تعديل الصنف Boat لكي يتناسب مع الصنف BoatDA

Modifying the Boat Class to Work with BoatDA

يمكنك الآن زيادة وظائف صنف مجال المشكلة Boat لكي تستفيد من إمكانيات صنف التعامل مع البيانات BoatDA، حيث نحتاج أن نعرف أربعة إجراءات من النوع Shared لاستدعاء الإجراء Initialize والإجراء Find والإجراء GetAll والإجراء Terminate من الصنف BoatDA، وتعريف ثلاثة إجراءات أخرى لاستدعاء الإجراء AddNew والإجراء Delete والإجراء Update من الصنف BoatDA أيضاً. كما يمكن تحسين الإجراء TellAboutSelf لكي يعيد معلومات المركب بشكل يسهل قراءتها للإجراء المستدعى له. يوضح الشكل رقم (١٤.١٤) أوامر تعريف الصنف Boat.

```
Imports System.Data
Imports System.Data.OleDb
Public Class Boat
    ' Attributes
    Private stateRegistrationNo As String
    Private length As Single
    Private manufacturer As String
    Private year As Integer
    ' Reference variable for Customer instance
    Private theCustomer As Customer

    ' Constructor (four parameters)
    Public Sub New(ByVal aStateRegistrationNo As String, _
        ByVal aLength As Single, _
        ByVal aManufacturer As String, ByVal aYear As Integer)

        StateRegistrationNo = aStateRegistrationNo
        length = aLength
        manufacturer = aManufacturer
        year = aYear
    End Sub

    ' Constructor (four parameters plus customer reference)
    Public Sub New(ByVal aStateRegistrationNo As String, _
     ByVal aLength As Single, _
     ByVal aManufacturer As String, ByVal aYear As Integer, _
     ByVal aCustomer As Customer)
        ' Set values of attributes using accessor methods
        SetStateRegistrationNo(aStateRegistrationNo)
        SetLength(aLength)
        SetManufacturer(aManufacturer)
        SetYear(aYear)
        ' Since boat must have a customer, assign it in constructor
        AssignBoatToCustomer(aCustomer)
    End Sub

    ' Custom method to assign a Boat to a Customer
    Public Sub AssignBoatToCustomer(ByVal aCustomer As Customer)
        SetCustomer(aCustomer)   ' Set customer reference for boat
        aCustomer.SetBoat(Me)    ' Ask the customer to set its boat
    End Sub

    ' Boat DA Shared Methods
    Public Shared Sub Initialize(ByVal c As OleDbConnection)
        BoatDA.Initialize(c)
    End Sub
    Public Shared Function Find(ByVal key As String) As Boat
        BoatDA.Find(key)
    End Function
    Public Shared Function GetAll() As ArrayList
        BoatDA.GetAll()
    End Function
    Public Shared Sub Terminate()
        BoatDA.Terminate()
    End Sub

    ' Boat DA Instance Methods
    Public Sub AddNew()
        BoatDA.AddNew(Me)
    End Sub
    Public Sub Delete()
        BoatDA.Delete(Me)
    End Sub
    Public Sub Update()
        BoatDA.Update(Me)
    End Sub
```

الشكل رقم (١٤,١٤). الصنف Boat.

```
    ' Get Accessor Methods
    Public Function GetStateRegistrationNo() As String
        Return stateRegistrationNo
    End Function
    Public Function GetLength() As Single
        Return length
    End Function
    Public Function GetManufacturer() As String
        Return manufacturer
    End Function
    Public Function GetYear() As Integer
        Return year
    End Function
    Public Function GetCustomer() As Customer
        Return theCustomer
    End Function

    'Set accessor methods
    Public Sub SetStateRegistrationNo(ByVal aStateRegNo As String)
        stateRegistrationNo = aStateRegNo
    End Sub
    Public Sub SetLength(ByVal aLength As Double)
        length = aLength
    End Sub
    Public Sub SetManufacturer(ByVal aManufacturer As String)
        manufacturer = aManufacturer
    End Sub
    Public Sub SetYear(ByVal aYear As Integer)
        year = aYear
    End Sub
    Public Sub SetCustomer(ByVal aCustomer As Customer)
        theCustomer = aCustomer
    End Sub

End Class
```

تابع الشكل رقم (١٤,١٤).

تعديل تعريف الصنف CustomerDA
Modifying the CustomerDA Class

يمكن الآن تعديل تعريف الصنف CustomerDA لكي يدعم ربط معلومات كل من الجدول BoatTable والجدول CustomerTable معاً. يبدأ تعريف الصنف CustomerDA مثل تعريفه السابق، ولكنه الآن يحتوي على متغير إشارة للصنف Boat ومتغيرات أخرى تمثل صفات كائن الصنف Boat كما هو واضح من الأوامر التالية:

```
Imports System.Data.OleDb
Imports System.Collections
Public Class CustomerDA
    Shared customers As New ArrayList()  ' Customer references
    Shared aCustomer As Customer
    Shared aBoat As Boat
    Shared aConnection As OleDbConnection

    ' Declare variables for Customer attribute values
    Shared name, address, phoneno As String

    ' Declare variables for the Boat attribute values
    Shared stateRegistrationNo, manufacturer As String
    Shared length As Single
    Shared year As Integer
```

يتشابه كل من الإجراء Initialize والإجراء Terminate للصنف CustomerDA مع كل من الإجراء Initialize والإجراء Terminate المعرفَيْن داخل الصنف BoatDA ؛ ولذلك لا يتم تكرار تعريفهما هنا.

الإجراء Find والإجراء GetAll للصنف CustomerDA

Understanding the Find and GetAll Methods of the CustomerDA Class

يمكن زيادة إمكانات الإجراء Find للصنف CustomerDA لكي يعيد بيانات كل من العميل والمركب معاً، ولعمل ذلك يجب تغيير الأمر Select حيث نغير الجزء WHERE بإضافة شرط الربط بين الجدولين وإضافة شرط تساوي المفتاح الرئيس برقم العميل المراد البحث عنه، فعندئذ سوف نستخدم البيانات العائدة في إنشاء كائن من الصنف Customer وكائن من الصنف Boat، ثم نستدعي الإجراء AssignBoatToCustomer من كائن الصنف Boat لإقامة علاقة الربط بين الكائنين في كلا الاتجاهين (أي بين العميل والمركب الذي يملكه)، والآن نعيد كائن الصنف Customer الذي يحتوي على مؤشر كائن الصنف Boat إلى البرنامج الداعي للإجراء Find. يوضح الشكل رقم (١٤,١٥) أوامر تعريف الإجراء Find.

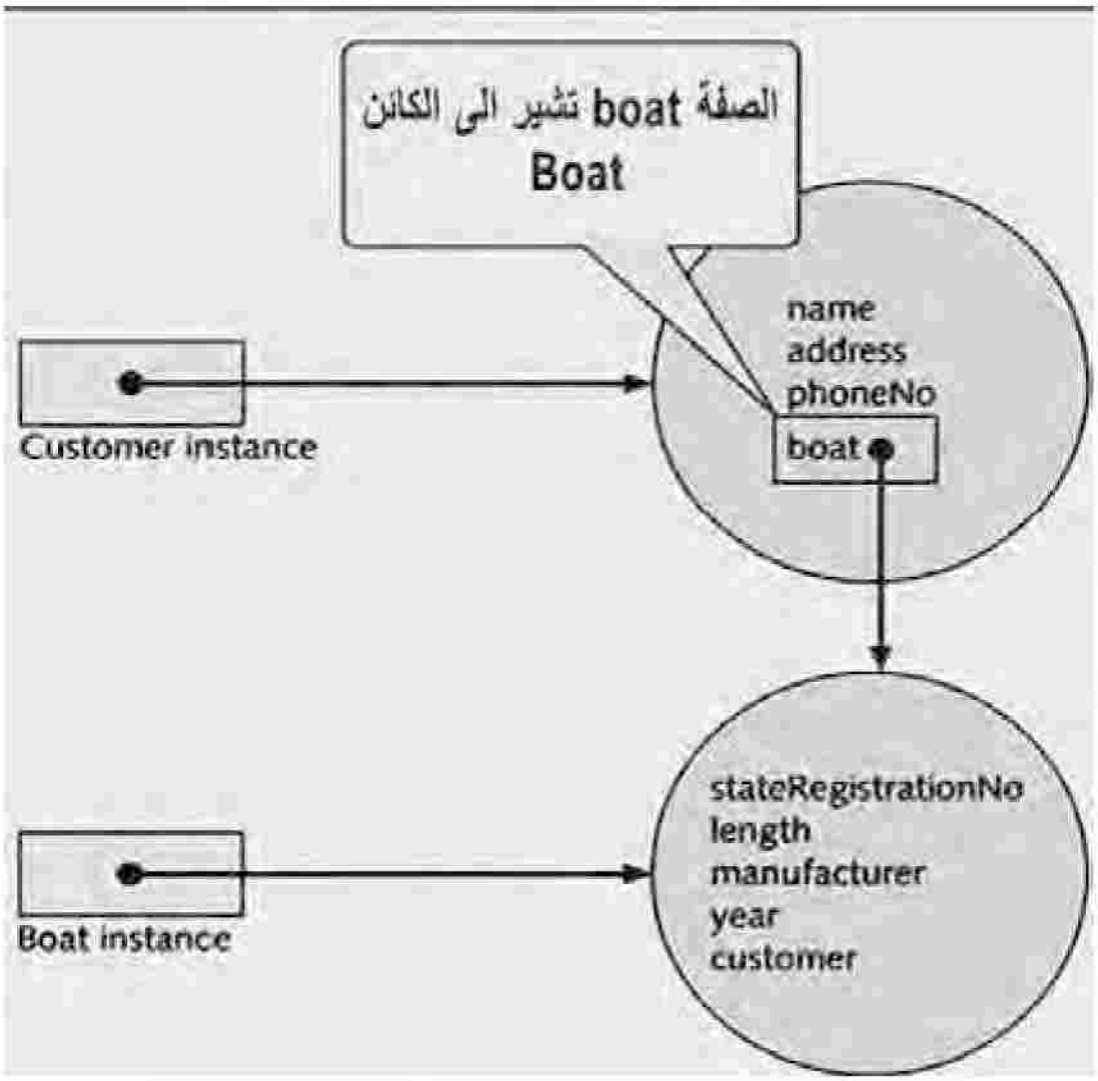
لاحظ أن الإجراء Find مازال يعيد كائناً من الصنف Customer ولكن مع فرق هام وهو أن هذا الكائن يحتوي على مؤشر لكائن من الصنف Boat كما هو واضح في الشكل رقم (١٤,١٦).

وهذا يعني أنه ليس من الضروري تنفيذ أمر Select منفصل للبحث عن بيانات مركب داخل الجدول BoatTable لأن قيم صفات مركب العميل المراد البحث عنه ستكون متاحة لإنشاء كائن من الصنف Boat. وكما ذكرنا سابقاً فإن الإجراء AssignBoatToCustomer المعرف داخل الصنف Boat يستخدم لإقامة علاقة ربط بين كائن الصنف Customer وكائن الصنف Boat في كلا الاتجاهين.

إن عمل الإجراء GetAll يشبه عمل الإجراء Find ولكن يعيد مصفوفة ArrayList تحتوي على كائنات الصنف Customer فقط. لاحظ أن كلا الإجراءين يعيدان معلومات حول العملاء والمراكب التي يملكونها.

```
' Find method--Throws NotFoundException if Not Found
Public Shared Function Find(ByVal key As String) As Customer
      aCustomer = Nothing
      ' Define the SQL statement using the phoneno key
      Dim sqlQuery As String = "SELECT Name, Address, PhoneNo, " & _
            "StateRegistrationNo, BoatLength, Manufacturer, Year " & _
            "FROM CustomerTable, BoatTable " & _
            "WHERE phoneNo = CustomerPhoneNo AND phoneNo = '" & key & "'"
      ' Declare dataset
      Dim dsCustBoat As New DataSet()
      Try      ' Execute query & fill dataset
            Dim adptCustomer As New OleDbDataAdapter(sqlQuery, aConnection)
            adptCustomer.Fill(dsCustBoat, "CustBoatTable")

            ' If record in row, found it; extract data
            If dsCustBoat.Tables("CustBoatTable").Rows.Count > 0 Then
                Dim dr As DataRow
                dr = dsCustBoat.Tables("CustBoatTable").Rows(0)

                name = dr("Name")
                address = dr("address")
                phoneno = dr("phoneno")
                stateRegistrationNo = dr("StateRegistrationNo")
                length = dr("BoatLength")
                manufacturer = dr("Manufacturer")
                year = dr("Year")
                ' Create customer & boat instance; assign to customer
                aCustomer = New Customer(name, address, phoneno)
                aBoat = New Boat(stateRegistrationNo, length, _
                        manufacturer, year)
                aBoat.AssignBoatToCustomer(aCustomer)

            Else     ' Nothing retrieved, throw error
                Throw New NotFoundException("Not Found")
            End If
            dsCustBoat = Nothing
        Catch e As OleDb.OleDbException
            Console.WriteLine(e.ToString)
        End Try
        Return aCustomer
End Function
```

الشكل رقم (١٤,١٥)- إجراء Find للصنف CustomerDA.



الشكل رقم (١٤,١٦). كائن الصنف Customer يحوي على مؤشر لكائن من الصنف Boat.

الإجراء AddNew للصنف CustomerDA

Understanding the AddNew Method of the CustomerDA Class

تذكر أن العلاقة بين المركب والعميل علاقة إلزامية من النوع "واحد–إلى–واحد" وهذا يعني أنه يجب أن يتواجد سجل عميل واحد لكل سجل مركب، ووجود سجل مركب واحد لكل عميل، ومن ثم يجب أن يضمن الإجراء AddNew هذا المعنى للحفاظ على تكامل البيانات (Data Integrity).

بعد إضافة بيانات العميل داخل الجدول CustomerTable، يستدعي الإجراء AddNew المعرف داخل الصنف CustomerDA الإجراء AddNew المعرف داخل الصنف BoatDA لإضافة بيانات المركب المرتبط بهذا العميل، وبذلك نضمن بهذا التعديل تكامل البيانات في قاعدة البيانات (انظر الشكل رقم ١٤.١٧).

```
' AddNew method --Throws DuplicateException if exists
Public Shared Sub AddNew(ByVal aCustomer As Customer)

        ' Get customer information
        name = aCustomer.GetName
        address = aCustomer.GetAddress
        phoneno = aCustomer.GetPhoneNo
        ' Get boat instance via customer
        aBoat = aCustomer.GetBoat

        ' Declare a string to hold SQL statement
        Dim sqlInsert As String = "INSERT INTO customerTable " & _
            "VALUES ('" & name & "', '" & address & "', '" & phoneno & "')"
        Dim adptCustomer As New OleDbDataAdapter()

        Try
            Dim c As Customer = Find(phoneno)
            Throw New DuplicateException(" Customer Exists ")
    Catch e As NotFoundException
        Try
            ' See if boat exists
            Dim b As Boat = Boat.Find(aBoat.GetStateRegistrationNo)
            Throw New DuplicateException("Boat Exists")
        Catch ee As NotFoundException
            Try ' Assign Insert Commands and Execute
                adptCustomer.InsertCommand = New OleDbCommand(sqlInsert)
                adptCustomer.InsertCommand.Connection = aConnection
                adptCustomer.InsertCommand.ExecuteNonQuery()
                ' Add boat
                Try
                    aBoat.AddNew()
                Catch de As DuplicateException
                    Console.WriteLine(de.Message.ToString)
                End Try
            Catch sqle As OleDb.OleDbException
                Console.WriteLine(sqle.ToString)
            End Try
        End Try
    End Try
End Sub
```

الشكل رقم (١٤,١٧). إجراء AddNew الصنف CustomerDA.

الإجراء Delete والإجراء Update للصنف CustomerDA

Understanding the Delete and Update Methods of the CustomerDA Class

يجب أيضاً أن يضمن الإجراء Delete تكامل البيانات في قاعدة البيانات بين العميل والمركب وذلك بحذف بيانات المركب المرتبط بعميل ما عند حذف بيانات هذا العميل (انظر الشكل رقم ١٤,١٨).

```
' Delete method--Throws NotFoundException if not found
Public Shared Sub Delete(ByVal aCustomer As Customer)
        phoneno = aCustomer.GetPhoneNo
        Dim sqlDelete As String = "DELETE FROM CustomerTable " & _
            "WHERE PhoneNo = '" & phoneno & "'"
        Dim adptCustomer As New OleDbDataAdapter()
        ' Delete the record
        Try
            Dim c As Customer = Customer.Find(phoneno)
            adptCustomer.DeleteCommand = New OleDbCommand(sqlDelete)
            adptCustomer.DeleteCommand.Connection = aConnection
            adptCustomer.DeleteCommand.ExecuteNonQuery()
            '  Delete Customer's Boat
            Try
                aCustomer.GetBoat.Delete()
            Catch e As NotFoundException
                Console.WriteLine(e.Message.ToString)
            End Try
        Catch e As Exception
            Console.WriteLine(e.ToString)
        End Try
End Sub
```

الشكل رقم (١٤،١٨). إجراء Delete للصنف CustomerDA.

يظل الإجراء Update للصنف CustomerDA دون تغيير (أي مثل الإجراء المعرف داخل الفصل الثالث عشر)، حيث إن عمل هذا الإجراء هو تعديل بيانات العميل فقط داخل الجدول CustomerTable.

اختبار التطبيق CustomerAndBoatDatabase

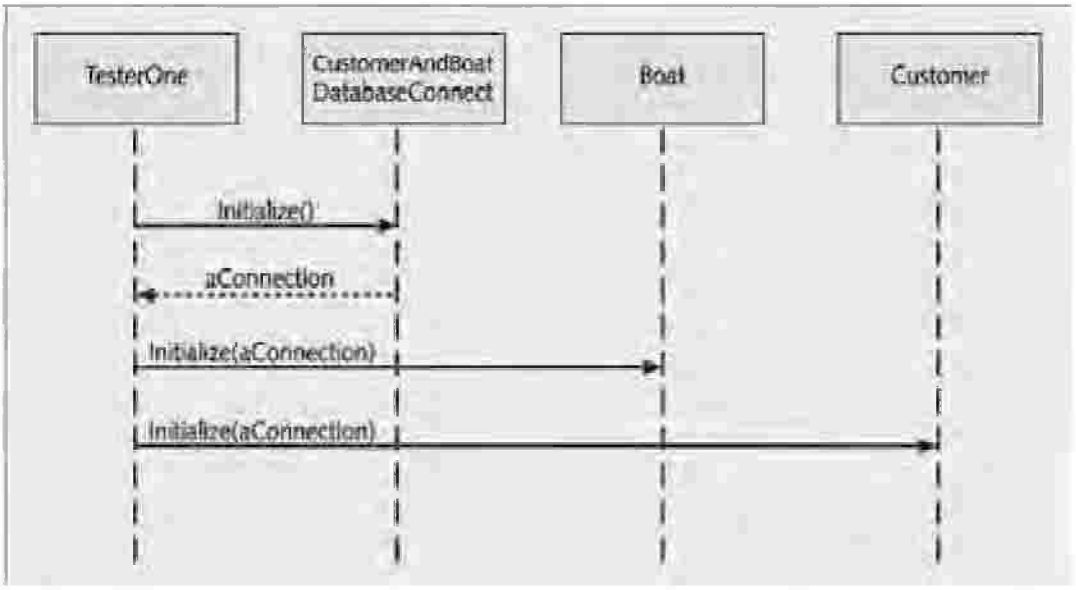
Testing the New CustomerAndBoatDatabase Application

إن الغرض من أوامر برنامج الاختبار هو التحقق من صحة عمل أصناف التطبيق CustomerAndBoatDatabase حيث تعرف أولاً متغيرات الإشارة الضرورية، ثم نستدعي الإجراء Initialize من الصنف CustomerAndBoatDatabase لإقامة الاتصال بقاعدة البيانات، ثم نمرر هذا الاتصال إلى الإجراء Initialize المعرف داخل الصنف Customer والصنف Boat كما هو واضح في الأوامر التالية:

```
Imports System.Data.OleDb
Imports System.Collections
Public Class Form1
        Inherits System.Windows.Forms.Form

Shared aCustomer As Customer
    Shared aBoat As Boat
    Shared customers As ArrayList
```

```
    Shared boats As ArrayList
    Shared o As OleDbConnection = CustomerAndBoatDatabaseConnect.Initialise

Private Sub testDAClasses()
' Initialise the databases
        Boat.Initialise(o)
        Customer.Initialise(o)
```

يوضح الشكل رقم (١٤,١٩) نموذج Sequence Diagram لهذه التفاعلات رسومياً.



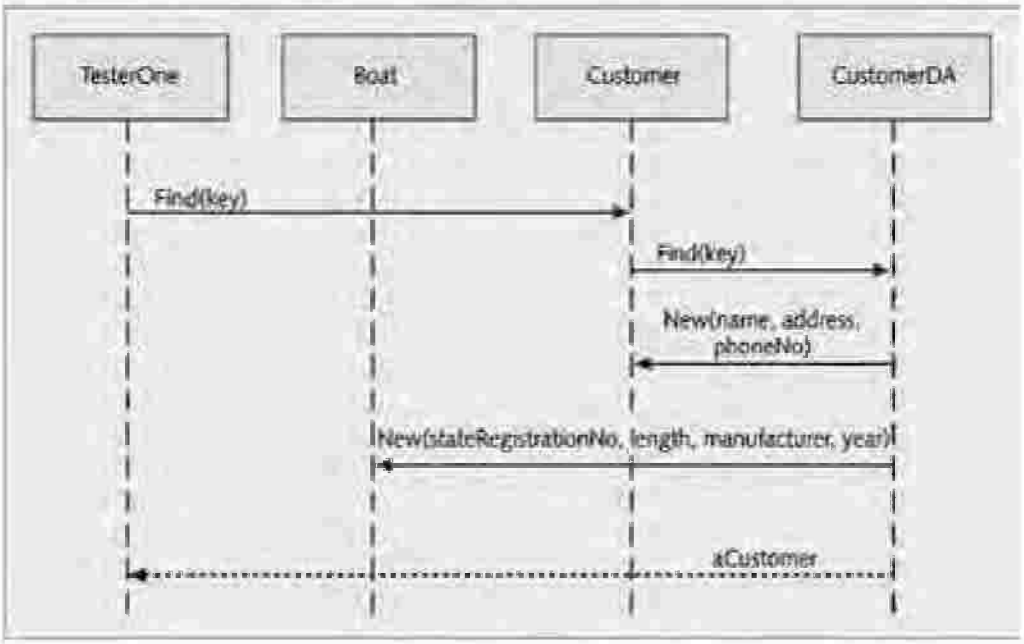
الشكل رقم (١٤,١٩): النموذج المعالي للاتصال بقاعدة البيانات.

لم نحاول استرجاع معلومات العميل الذي يملك رقم الهاتف 123-4567 والمركب الخاص به من قاعدة البيانات، حيث يتم استدعاء الإجراء Find أولاً من الصنف Customer الذي يستدعي الإجراء Find المعرف داخل الصنف CustomerDA، وتذكر أن الإجراء FindDA يستخدم أمر Select الذي يربط الجداول سوياً ويسترجع معلومات حول العميل والمركب الخاص بهما. يستخدم كل من الصنف Customer والصنف Boat هذه المعلومات لإنشاء كائنات من الصنف Customer والصنف Boat ثم يتم استدعاء الإجراء PrintDetails لعرض معلومات العميل والمركب الذي يملكه.

```
Try     ' Get a customer and their boat
        aCustomer = Customer.Find("123-4567")
        Call printDetails()
Catch e As NotFoundException
        Console.WriteLine(e.Message.ToString)
End Try
```

يوضح الشكل رقم (١٤,٢٠) نموذج Sequence Diagram الذي يمثل هذه العملية، وبالعودة إلى تفاصيل الإجراء PrintDetails نجد أن الإجراء TellAboutSelf يتم استدعاؤه من كائن الصنف Customer للحصول على معلومات العميل، ويتم استدعاء الإجراء TellAboutSelf من كائن الصنف Boat المرتبط بهذا العميل للحصول على معلومات المركب الذي يملكه.

```
Private Sub printDetails()
        Try
            Console.WriteLine(vbCrLf)
            Console.WriteLine("Found " + aCustomer.GetName + _
                " and associated boat")
            Console.WriteLine("    " + aCustomer.TellAboutSelf)
            Console.WriteLine("    " + aCustomer.GetBoat.TellAboutSelf)
        Catch e As NotFoundException
            Console.WriteLine("Customer Not Found")
        Catch e As Exception
            Console.WriteLine(e.ToString)
        End Try
End Sub
```



الشكل رقم (١٤,٢٠). النموذج المثالي لإيجاد العميل والمركب الخاص بالعميل.

كما نختبر في هذا البرنامج إمكانية البحث عن عميل غير موجود في قاعدة البيانات وذلك بواسطة الإجراء Find من الصنف Customer وإمرار رقم هاتف غير موجود (000-0000)، وعندئذ لا يعيد تنفيذ أمر Select أي

سجلات ومن ثم يلقي الصنف Customer الاستثناء NotFoundException الذي يتم استقباله بواسطة برنامج الاختبار الذي يظهر رسالة الخطأ "Did not find 000-0000" كما هو واضح في الأوامر التالية:

```
'  Try to get a non-existent customer and their boat
Try
      aCustomer = Customer.Find("000-0000")
      printDetails()
Catch e As NotFoundException
      Console.WriteLine(vbCrLf & "Did not find 000-000")
End Try
```

تم استدعاء الإجراء ()GetAll من الصنف CustomerDA الذي يعيد جميع العملاء الموجودين في قاعدة البيانات والمراكب التي يملكونها (مصفوفة ArrayList تحتوي على كائنات الصنف Customer والمرتبطة بكائنات الصنف Boat) ثم يتجول برنامج الاختبار داخل هذه المصفوفة باستخدام الأمر For Each لطباعة معلومات كل عميل والمركب الذي يملكه.

```
' Get all customers & their boats
customers = Customer.GetAll
Dim cust As Customer
For Each cust In customers
      aCustomer = cust
      printDetails()
Next
```

وكما هو واضح من الأوامر التالية، يتم إنشاء كائن جديد من الصنف Customer (الكائن aCustomer) باستخدام رقم الهاتف 339-4990 وكائن جديد من الصنف Boat (الكائن aBoat)، ثم يتم استدعاء الإجراء AssignBoatToCustomer من الكائن Boat لإقامة علاقة الترابط بين الكائن aCustomer والكائن aBoat، ثم يتم استدعاء الإجراء AddNew من الكائن aCustomer لإضافة بيانات العميل، ولكن تذكر أن الإجراء AddNew المعرف داخل الصنف CustomerDA ينفذ الأمر INSERT لإضافة بيانات العميل داخل الجدول CustomerTable ثم يستدعي الإجراء AddNew المعرف داخل الصنف BoatDA لإضافة بيانات المركب داخل الجدول BoatTable، وعند حدوث تكرار في قيمة المفتاح الأساسي لأي من العميل والمركب عند إضافة بياناتهم سيعيد الإجراء AddNew الاستثناء DuplicateException وتنتهي عملية الإدخال بأكملها.

```
' Add a new customer and their boat
aCustomer = New Customer("Ed", "KC", "339-4990")
aBoat = New Boat("MO112233", 25, "S-2", 1984)
aBoat.AssignBoatToCustomer(aCustomer)
Try
    aCustomer.AddNew()
    Console.WriteLine(vbCrLf & "Ed and his boat added")
Catch e As DuplicateException
    Console.WriteLine(e.Message.ToString)
End Try
```

ولكن نتحقق من نجاح عملية إضافة بيانات العميل الذي يملك الهاتف رقم 339-4990 وبيانات المركب الخاص به إلى قاعدة البيانات، نستدعي الإجراء Find() للبحث عن هذا العميل كما هو واضح من الأوامر التالية:

```
' Try to find new customer and their boat
Try
    aCustomer = Customer.Find("339-4990")
    printDetails()
Catch e As NotFoundException
    Console.WriteLine("Did not Find 339-4990")
End Try
```

ولكي نتحقق من صحة عمل الإجراء Delete، تستدعي الأوامر التالية الإجراء Delete() من الكائن aCustomer لحذف بيانات العميل المضاف حديثاً من قاعدة البيانات الذي بدوره سيستدعي الإجراء Delete لحذف بيانات العميل ثم يستدعي الإجراء Delete من الصنف BoatDA لحذف بيانات المركب الذي يملكه هذا العميل، ومن ثم يتم استدعاء الإجراء Find() للتأكد من نجاح عملية حذف بيانات العميل من قاعدة البيانات.

```
' Try to delete the new customer and their boat
Try
    aCustomer.Delete()
    Console.WriteLine(vbCrLf & "Ed deleted" & vbCrLf)
Catch e As NotFoundException
    Console.WriteLine(e.Message.ToString)
End Try

Try
    aCustomer = Customer.Find("339-4990")
    printDetails()
Catch e As NotFoundException
    Console.WriteLine("Did Not Find 339-4990")
End Try
```

ولاختبار عمل الإجراء Update تستدعي الأوامر التالية الإجراء Find للبحث عن العميل الذي يملك رقم الهاتف 123-4567 والحصول على كائن يحتوي على بيانات هذا العميل (aCustomer)، ثم تستدعي الإجراء SetAddress الذي يغير عنوان هذا العميل في الذاكرة، ثم تستدعي الإجراء Update لتغيير بيانات هذا العميل في قاعدة البيانات، ثم تستدعي الإجراء SetLength لتغيير طول المركب الذي يملكه هذا العميل واستدعاء الإجراء Update من كائن الصنف Boat لتعديل بيانات هذا المركب في قاعدة البيانات. وإذا لم يتم الحصول على أي من العميل أو المركب سيتم إلقاء الاستثناء NotFoundException وإنهاء الإجراء مباشرة، وأخيراً يتأكد برنامج الاختبار من نجاح عملية التعديل بالبحث عن هذا العميل وعرض بياناته.

```
' Change Eleanor's address to Miami and her boat length to 40
Try
    aCustomer = Customer.Find("123-4567")
    printDetails()
    ' Change customer address
    aCustomer.SetAddress("Miami")
    aCustomer.Update()
    ' Change boat length
    aCustomer.GetBoat.SetLength(40)
    aCustomer.GetBoat.Update()
    Console.WriteLine(vbCrLf & "Eleanor updated")
Catch e As NotFoundException
    Console.WriteLine(e.Message.ToString)
End Try

' Get Eleanor and her boat
Try
    aCustomer = Customer.Find("123-4567")
    printDetails()
Catch e As NotFoundException
    Console.WriteLine(e.Message.ToString)
End Try
```

وأخيراً يعرض برنامج الاختبار رسالة تفيد انتهاء تنفيذ أوامره، ثم يستدعي الإجراء Terminate من كل من الصنف Customer والصنف Boat لإغلاق كائن الاتصال بقاعدة البيانات الخاص بهما، ثم يستدعي الإجراء Terminate من الصنف CustomerAndBoatDatabaseConnect، ثم يغلق النموذج ويتوقف عمل البرنامج. يوضح الشكل رقم (١٤,٢١) مخرجات هذا البرنامج.

```
Found Eleanor and associated boat
  Name = Eleanor, Address = Memphis, Phone No = 123-4567
  State Reg No = MO345671, Boat Length = 35, Manufacturer = Tartan, Year = 1998

Did not find 0000000

Found Brian and associated boat
  Name = Brian, Address = Los Angeles, Phone No = 667-1234
  State Reg No = MO223344, Boat Length = 24, Manufacturer = Tracker, Year = 1996

Found Dan and associated boat
  Name = Dan, Address = Boston, Phone No = 587-4321
  State Reg No = MO457812, Boat Length = 19, Manufacturer = Ranger, Year = 2001

Found Eleanor and associated boat
  Name = Eleanor, Address = Memphis, Phone No = 123-4567
  State Reg No = MO345671, Boat Length = 35, Manufacturer = Tartan, Year = 1998

Found Mike and associated boat
  Name = Mike, Address = Boston, Phone No = 667-1122
  State Reg No = MO98765, Boat Length = 28, Manufacturer = JBoat, Year = 1986

Found JoAnn and associated boat
  Name = JoAnn, Address = St. Louis, Phone No = 765-4321
  State Reg No = MO12345, Boat Length = 26, Manufacturer = Ranger, Year = 1976

Found Dave and associated boat
  Name = Dave, Address = Atlanta, Phone No = 221-4567
  State Reg No = MO34321, Boat Length = 30, Manufacturer = Bayliner, Year = 2001

Ed and his boat added

Found Ed and associated boat
  Name = Ed, Address = KC, Phone No = 339-4990
  State Reg No = MO112233, Boat Length = 22, Manufacturer = SZ, Year = 1984
Ed deleted

Did Not Find 339-4990

Found Eleanor and associated boat
  Name = Eleanor, Address = Memphis, Phone No = 123-4567
  State Reg No = MO34561, Boat length = 35, Manufacturer = Tartan, Year = 1998

Eleanor updated

Found Eleanor and associated boat
  Name = Eleanor, Address = Miami, Phone No = 123-4567
  State Reg No = MO34561, Boat Length = 40, Manufacturer = Bertram, Year = 1998
```

الشكل رقم (١٤,٢١). مخرجات برنامج اختبار ليستنك

تطوير علاقة الترابط "واحد-إلى-العديد" داخل تطبيق قاعدة البيانات

Implementing a One-to-Many Association in a Database Application

لقد طورنا في المثال السابق علاقة ترابط من النوع "واحد-إلى-واحد" بين كل من الصنف Customer والصنف Boat أثناء تطوير التطبيق السابق، وبالمثل يمكن تطوير جميع علاقات الترابط التي من النوع نفسه والمتواجدة في برنامج برادشو مارينا باستخدام الأسلوب نفسه (مثل علاقة الترابط بين الصنف Boat والصنف Slip)، وسنتعلم في هذا المثال كيف نطور علاقة الترابط "واحد-إلى-العديد" والتي تربط الصنف Dock (رصيف) والصنف Slip (مرسى)، حيث إن الرصيف يحتوي على العديد من المراسي، والمرسى يوجد داخل رصيف واحد، ومن ثم سيتمكن مستخدم هذا البرنامج من الحصول على تقرير مفصل عن جميع أرصفة الشركة وعن معلومات مراسي كل رصيف.

شرح جداول قاعدة البيانات

Understanding the Tables in DockAndSlipDatabase

قبل البدء في تطوير علاقة الترابط "واحد-إلى-العديد"، يجب أولاً أن نطلع على التركيب الداخلي لجداول قاعدة بيانات التطبيق DockAndSlipDatabase للاطلاع على كل من الجدول DockTable والجدول SlipTable، وللقيام بذلك اتبع التالي:

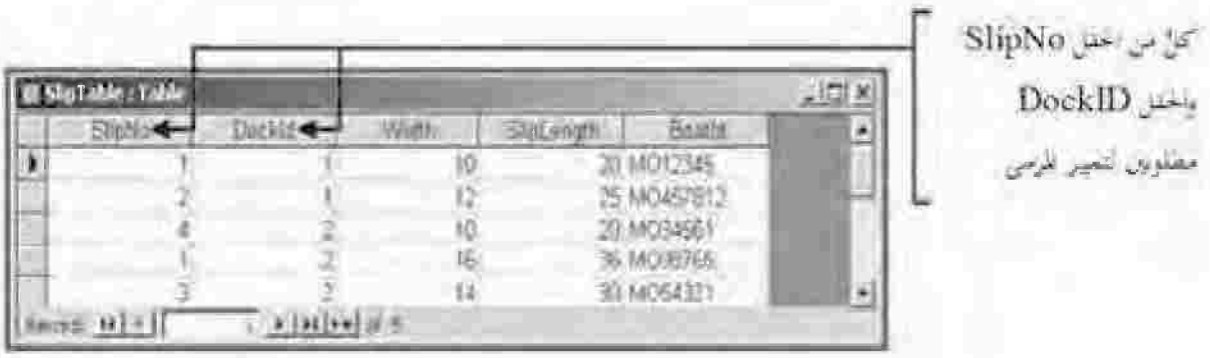
١- ابحث عن المجلد Ex02 داخل المجلد Chap14/Exercises داخل مجلد العمل الخاص بك، وعندئذ ستجد الملف DockAndSlipDatabase.mdb داخل المجلد bin للمجلد Ex02.

٢- اضغط ضغطاً مزدوجاً على ملف قاعدة البيانات لفتح برنامج Microsoft Access، وعندئذ ستجد كلاً من الجدول DockTable والجدول SlipTable.

٣- يحتوي الجدول DockTable على أربعة حقول: الحقل DockId (المفتاح الأساسي للجدول)، والحقل Location، والحقل Electricity، والحقل Water. يأخذ الحقل DockId نوع الرقم الصحيح Integer والذي يمثل رقم الرصيف، ويمثل الحقل Location موقع الرصيف من النوع نص، ويشير كل من الحقل Electricity والحقل Water إلى توفر الكهرباء والمياه في الرصيف وعدم توفرهما، وقد تم تعريف كل من قيم هذين الحقلين في أصناف مجال المشكلة من النوع Boolean (True أو False)، وبما أن برنامج MS Access لا يدعم هذا النوع فإن قاعدة البيانات تستخدم كلاً من القيمة واحد لتشير أن الخدمة متواجدة في الرصيف والقيمة صفر لتشير عدم وجود الخدمة (انظر الشكل رقم ١٤.٢٢).

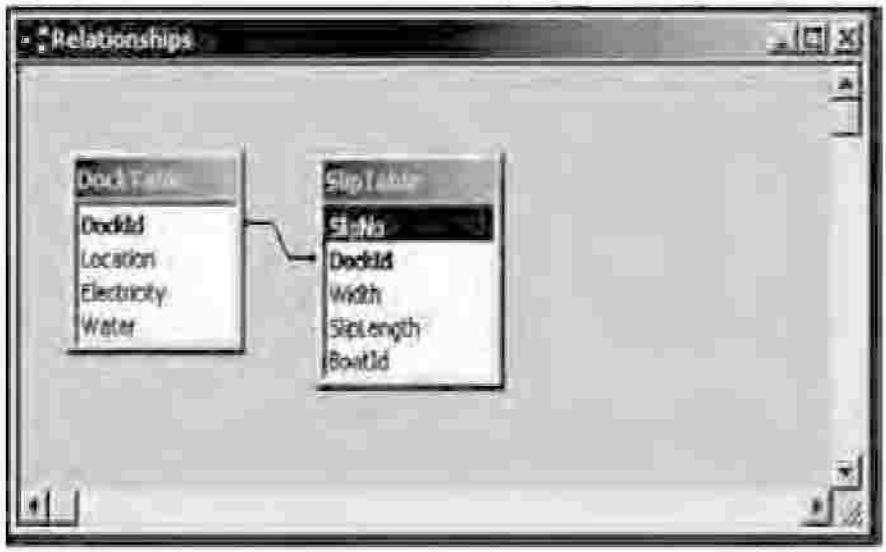
الشكل رقم (١٤,٢٢). محتويات DockTable.

٤- يحتوي الجدول SlipTable على خمسة حقول وهي: الحقل SlipNo، والحقل DockID، والحقل Width، والحقل SlipLength، والحقل BoatID. ولأن مراسي الرصيف الواحد يتم ترقيمها بداية من رقم واحد، فإن المرسى يتم تعريفه بكل من رقم المرسى SlipNo ورقم الرصيف DockID؛ ولذلك فإن كلاً من الحقل SlipNo والحقل DockID يمثلان سوياً المفتاح الأساسي لهذا الجدول (انظر الشكل رقم ١٤,٢٣).



الشكل رقم (١٤,٢٣). محتويات SlipTable.

٥- يمثل كل من الحقل DockID والحقل BoatID حقولاً أجنبية لكل من الجدول DockTable والجدول BoatTable في الجدول SlipTable. ويوضح الشكل رقم (١٤,٢٤) العلاقة بين الجدول DockTable والجدول SlipTable، ولاحظ أن كلاً من الحقل SlipNo والحقل DockID يظهران بخط أسود عريض مما يشير إلى أنهما يمثلان المفتاح الأساسي للجدول SlipTable.



الشكل رقم (١٤,٢٤). نموذج العلاقات للجدول dockstable والجدول slipstable.

إنشاء اتصال مشترك لقاعدة البيانات DockAndSlipDatabase
Establishing a Common Connection to DockAndSlipDatabase

وكما فعلنا في المثال السابق نحتاج أن نعرف صنفاً منفصلاً يستخدم لإنشاء اتصال واحد (كائن من الصنف OleDbConnection) بقاعدة البيانات DockAndSlipDatabase؛ ولذلك فإن الأصناف التي تحتاج إلى اتصال بقاعدة البيانات تشترك في استخدام هذا الاتصال، حيث يتم إنشاء كائن من الصنف OleDbConnection (DockAndSlipDatabaseConnect) كما فعلنا بالضبط في المثال السابق عدا أن اسم مصدر البيانات يجب أن يختلف؛ ولذلك لم نناقش هذا الموضوع مرة ثانية. تتواجد أوامر هذا الصنف داخل المجلد Example2 داخل ملفات الطالب.

تعديل كل من الصنف Dock والصنف Slip
Modifying the Dock and Slip Classes

لقد قدمنا كل من الصنف Dock والصنف Slip في الفصل التاسع، وسوف يتم تعديل كل منهما الآن لتخزين كائنات داخل قاعدة البيانات العلائقية. أولاً سنضطر إلى تغيير اسم الصنف Dock إلى mDock لأن لغة VB .NET تحتوي على صنف بالاسم نفسه مستخدم مع نماذج الويندوز؛ ولذلك لا يمكن وجود صنفين بالاسم نفسه، وعلى هذا سوف نعيد تسمية هذا الصنف إلى mDock لإزالة هذا الخلاف.

نحتاج أيضاً إلى استدعاء الفضاء المسمى System.Data.OleDb داخل الصنف mDock لتوفير إمكانية التعامل مع أصناف قواعد البيانات، كما يجب أن نتذكر أنه قد تم الوصول إلى حقيقة قدرة إضافة بيانات كل من الأرصفة أو

المراسي أو حذفها أو تعديلها أثناء مرحلة التحليل ؛ ولذلك لم نعرف الإجراء Delete داخل الصنف mDock للمثال التالي. وأخيراً نريد أن نضيف الإجراء TellAboutSelf داخل الصنف mDock. يوضح الشكل رقم (١٤.٢٥) تعريف الصنف mDock بعد إزالة إجراءات المرور.

يحتوي الصنف Slip المعرف داخل الفصل التاسع على الأوامر المطلوبة لربط كائن من الصنف Slip إلى الصنف Dock مع العلم أيضاً أن بيانات المراسي نادراً ما يتم إضافتها أو تعديلها أو حتى حذفها من قاعدة البيانات ؛ ولذلك لا يحتوي الصنف Slip على أي من إجراءات الإضافة أو التعديل أو الحذف ، بل أكثر من ذلك فإن بيانات المرسى تكون فقط داخل الإجراء Find أو الإجراء GetAll المعرفَين داخل الصنف DockDA ، ومن ثم لا نحتاج أن نضيف الإجراء Find أو الإجراء GetAll للصنف Slip ، ومن ثم سيبقى الصنف Slip دون تغيير.

```
' Chapter 14 Example2 Dock
' Class mDock
Imports System.Data.OleDb
Imports System.Collections
Public Class mDock
    ' Attributes
    Private dockID As String
    Private location As String
    Private electricity As Boolean
    Private water As Boolean

    ' References to slips associated with dock (1 to many)
    Private slips As ArrayList

    ' Constructor (four parameters)
    Public Sub New(ByVal aDockID as String, ByVal aLocation As String, _
      ByVal anElectricity As Boolean, ByVal aWater As Boolean)
        'invoke setter methods to populate attributes
        SetDockId(aDockID)
        SetLocation(aLocation)
        SetElectricity(anElectricity)
        SetWater(aWater)
        slips = New ArrayList()
    End Sub

    ' Customer method AddSlipToDock
    Public Sub AddSlipToDock(ByVal aSlip As Slip)
        slips.Add(aSlip)
        aSlip.SetDock(Me)
    End Sub
```

الشكل رقم (١٤.٢٥): صنف mDock.

```
' Add the slip reference to the ArrayList instance
Public Sub AddSlip(ByVal aSlip As Slip)
    slips.Add(aSlip)
End Sub

' DA Shared Methods
Public Shared Sub Initialize(ByVal c As OleDbConnection)
    DockDA.Initialize(c)
End Sub
Public Shared Function Find(ByVal key As Integer)
    Return DockDA.Find(key)
End Function
Public Shared Function GetAll() As ArrayList
    Return DockDA.GetAll
End Function
Public Shared Sub Terminate()
    DockDA.Terminate()
End Sub

' Return the ArrayList containing slip references
Public Function GetSlips() As ArrayList
    Return slips
End Function

'TellAboutSelf method
Public Function TellAboutSelf() As String
    Dim dockDetails As String
    dockDetails = "Dock " & dockID _
     & " Location: " & location _
     & ", Has Electricity= " & electricity _
     & ", Has water= " & water
    Return dockDetails
End Function
```

تابع الشكل رقم (١٤,٢٥).

تقديم الصنف DockDA

Introducing the DockDA Class

مما سبق نلاحظ أن الصنف DockDA سيكون مسؤولاً عن استرجاع بيانات Dock والمراسي المرتبطة به (الإجراء Find) أو استرجاع بيانات الأرصفة كلها المخزنة داخل قاعدة البيانات والمراسي المرتبطة بها، ولا يحتوي على إجراءات التعديل والإضافة والحذف. يبدأ تعريف الصنف باستضافة المسمى المطلوب (كما هو واضح في الأوامر التالية)، ثم تعريف متغيري إشارة من الصنف mDock والصنف Slip وتعريف مصفوفة Docks من النوع ArrayList، ثم تعريف متغير إشارة للصنف OleDbConnection متبوعاً بتعريف متغيرات صفات كل من الصنف mDock والصنف Slip.

```
' Chapter 14 -- Example2 DockDA
' Dock DA class
Imports System.Data.OleDb
Imports System.Collections

Public Class DockDA
    Shared aSlip As Slip
```

```
Shared aDock As mDock
Shared docks As New ArrayList()

Shared aConnection As OleDbConnection

' Declare variable for slip attribute values
Shared slipNo, width, dockId As Integer
Shared slipLength As Double

' Declare variables for dock attribute values
Shared id As Integer
Shared location As String
Shared electricity, water As Boolean
```

يتشابه عمل كل من الإجراء Initialize والإجراء Terminate لإجراءات أصناف التعامل مع البيانات السابقة ؛ ولذلك لا يتم إعادتهما هنا، وسوف نجد جميع هذه الأوامر داخل ملفات الطالب.

الإجراء Find للصنف DockDA

Understanding the Find Method of the DockDA Class

يعرف الإجراء Find للصنف DockDA أمر SELECT لاسترجاع بيانات كل من الرصيف mDock المطلوب والمراسي Slips التي توجد بداخله. يحتوي المقطع WHERE الموجود في أمر SELECT على شروط ربط الجدول DockTable بواسطة رقم الرصيف الممرر للإجراء Find، كما يحتوي الأمر SELECT على المقطع ORDER BY المسؤول عن ترتيب السجلات العائدة بواسطة رقم المرسى (الحقل SlipNo) كما هو واضح في الأمر التالي :

```
"SELECT DockTable.DockId, Location, " & _
      "Electricity, Water, SlipNo, Width, SlipLength " & _
      "FROM DockTable, SlipTable " & _
      "WHERE DockTable.DockID = SlipTable.DockID " & _
      "AND DockTable.DockId = " & key & " ORDER BY SlipNo"
```

تحتوي نتيجة الاستعلام على مجموعة من السجلات التي تتكون من حقول الجدول DockTable وحقول الجدول SlipTable مع ملاحظة أن هذه السجلات تناظر سجلات المراسي المتواجدة داخل الرصيف المراد البحث عنه ؛ ولذلك فإن قيم حقول الرصيف المطلوب تتكرر في كل سجل، ويوضح الجدول رقم (١٤.١) نتيجة الاستعلام عن الرصيف رقم ١.

الجدول رقم (١٤,١). مجموعة النتيجة لإجراء Find عندما يكون DockID=1.

DockId	Location	Electricity	Water	SlipNo	Width	SlipLength
1	Main Cove	1	0	1	10	20
1	Main Cove	1	0	2	12	25

لاحظ أن بيانات الرصيف الواحد تتكرر مع بيانات كل مرسى له؛ ولذلك يجب أن نقرأ بيانات الرصيف مرة واحدة فقط لإنشاء كائن من الصنف mDock ثم نقرأ بيانات باقي السجل (بيانات مرسى) لإنشاء كائن من الصنف Slip، ولعمل ذلك يمكن عمل متغير (dockCreated) من النوع Boolean وإسناد القيمة False له قبل أمر التكرار باستخدام أمر If، فإذا وجدنا قيمة تساوي False فسوف نقرأ بيانات الرصيف وننشئ كائناً من الصنف mDock، ومن ثم عند اختباره مرة ثانية سنجد قيمته تساوي True، وعندئذ لم ينفذ محتوى الأمر If. لاحظ أن القيمة False في لغة VB .NET تساوي صفراً وعدا ذلك تساوي True. ولقد أشرنا في هذا المثال إلى False بالقيمة صفر وإلى True بالقيمة واحد كما هو واضح في الأوامر التالية:

```
Dim dsDockSlip As New DataSet()
Try     ' Execute Query, fill dataset
    Dim adptDockSlip As New OleDbDataAdapter(sqlQuery, aConnection)
    adptDockSlip.Fill(dsDockSlip, "DockSlip")
    Dim dr As DataRow
    Dim dockCreated As Boolean = False
    ' Check for rows in data set table
    If dsDockSlip.Tables("DockSlip").Rows.Count > 0 Then
        ' Loop over all the rows in the table
        For Each dr In dsDockSlip.Tables("DockSlip").Rows
            ' Create dock if first time through loop
            If Not dockCreated Then
                dockId = dr("DockId")
                location = dr("Location")
                electricity = dr("Electricity")
                water = dr("Water")

                ' Create a dock instance
                aDock = New mDock(dockId, location, _
                    electricity, water)
                dockCreated = True
            End If
```

لاحظ أن إجراء الإنشاء الصنف Slip يستقبل مؤشر الكائن الصنف mDock وذلك لإقامة علاقة الربط بين كل من المرسى والرصيف في كلا الاتجاهين بواسطة استدعاء الإجراء AddSlipToDock.

```
  slipNo = dr("SlipNo")
  width = dr("width")
  slipLength = dr("slipLength")

  ' Create slip instances--
  ' Note there may be more than one slip per dock
  aSlip = New Slip(slipNo, width, slipLength, aDock)
Next
dsDockSlip = Nothing
```

وعند عدم الحصول على الرصيف المطلوب في قاعدة البيانات فسوف يعيد الإجراء Find الاستثناء NotFoundException ويتم الخروج من الإجراء، أما إذا تم الحصول عليه فسوف يعيد الإجراء Find كائناً من الصنف mDock والمرتبط بمجموعة كائنات من الصنف Slip (المراسي المتواجدة داخل الرصيف) كما هو واضح في الأوامر التالية:

```
        If Not dockCreated Then
            Throw New NotFoundException("Dock Not Found")
        End If
    Catch e As OleDb.OleDbException
    Console.WriteLine(e.Message.ToString)
    End Try
    Return aDock
End Function
```

الإجراء GetAll للصنف DockDA

Understanding the GetAll Method of the DockDA Class

إن عمل الإجراء GetAll يشبه عمل الإجراء Find عدا أنه يعيد معلومات جميع الأرصفة والمراسي المخزنة داخل قاعدة البيانات مرتبة برقم الرصيف (DockID) أولاً ثم رقم المرسى SlipNo كما هو واضح في أمر SELECT التالي:

```
Dim sqlQuery As String = "SELECT DockTable.DockId, Location, " & _
        "Electricity, Water, SlipNo, Width, SlipLength " & _
        "FROM DockTable, SlipTable " & _
        "WHERE slipTable.DockId = DockTable.DockId " & _
        "ORDER BY DockTable.DockId, SlipNo"
```

وعندئذ تشبه البيانات العائدة من تنفيذ أمر SELECT السابق السجلات الظاهرة في الجدول رقم (١٤,٢).

الجدول رقم (١٤,٢). مجموعة النتيجة لإجراء GetAll.

DockId	Location	Electricity	Water	SlipNo	Width	SlipLength
1	Main Cove	1	0	1	10	20
1	Main Cove	1	0	2	12	25
2	Main Marina	0	1	1	16	35
2	Main Marina	0	1	3	14	30
2	Main Marina	0	1	4	10	20

ومن الملاحظ أن بيانات كل رصيف تتكرر مع مستوى جميع سجلات المراسي التي تتبعه، فعلى سبيل المثال إن الجدول رقم (١٤,٢) يوضح أن بيانات الرصيف رقم واحد (DockId=1) تتكرر مرتين لأنه يحتوي على عدد ٢ مرسى، وإن بيانات الرصيف رقم ٢ (DockId=2) تتكرر ثلاث مرات لأنه يحتوي على عدد ٣ مراسي؛ ولذلك يجب على الإجراء GetAll أن يسترجع بيانات كل رصيف مرة واحدة فقط لينشئ كائناً واحداً من الصنف mDock لكل رصيف، ولعمل ذلك يتم استخدام أسلوب منطقي يسمى "Control-Break" الذي يستخدم عندما تتغير قيمة متغير ما والذي يستخدم لتصنيف عناصر مرتبة، وفي هذه الحالة فإن البيانات مرتبة بناء على رقم المرسى SlipNo داخل كل مجموعة تنتمي لرقم رصيف معين DockId؛ ولذلك فإن الحقل DockId يسمى حقل التصنيف (Grouping Field) أو حقل التحكم (Control Field) الذي إذا تغيرت قيمته الحالية تنتقل إلى مجموعة أخرى.

وكما هو واضح من أوامر الشكل رقم (١٤,٢٦) فإن كلاً من المتغير thisDockId والمتغير prevDockId يُستخدمان لتحديد متى يتم تغيير رقم الرصيف (الانتقال إلى مجموعة جديدة)، حيث يسند الإجراء قيمة رقم الرصيف (DockId) أولاً إلى المتغير thisDockId التي تم استرجاعها من أول سجل في نتيجة الاستعلام، ثم يسند هذه القيمة داخل المتغير prevDockId في كل مرة تكرار من التكرار For Next الخارجي، ثم يسترجع بيانات الرصيف لينشئ كائناً من الصنف mDock، ثم يضيف هذا الكائن إلى المصفوفة Docks التي سيتم إعادتها في نهاية تنفيذ الإجراء. ولا ننسى أن أوامر التكرار For Next الداخلي ستنفذ إلى أن يتغير قيمة الحقل DockId (قيمة المتغير thisDockId تصبح أكبر من قيمة المتغير prevDockId) أو إلى أن نصل إلى نهاية سجلات الاستعلام.

وفي كل مرة تكرار من التكرار الداخلي For Next نسترجع بيانات مرسى لننشئ كائناً من الصنف Slip، ولكن تذكر أن الإجراء AddSlipToDock المعرف داخل الصنف Slip يتم استدعاؤه تلقائياً عند إنشاء كائن من الصنف Slip لربطه بكائن الصنف mDock، وكما ذكرنا سابقاً فإننا نقرأ قيمة الحقل DockId في كل مرة ونسندها إلى المتغير thisDockId لمقارنتها بقيمة المتغير prevDockId لمعرفة إذا تغيرت قيمة DockId أم لا، ومن ثم يتم الخروج من التكرار الداخلي إلى التكرار الخارجي ليتم تكرار تنفيذ العملية ثانية من البداية إلى أن نصل إلى نهاية سجلات الاستعلام.

```
Try     ' Execute the SQL statement and fill the dataset
    Dim daDockSlip As New DataSet()
    Dim adptDockSlip As New OleDbDataAdapter(sqlQuery, aConnection)
    adptDockSlip.Fill(daDockSlip, "DockSlip")

    ' Define working variables
    Dim dr, dra As DataRow
    Dim i, j, thisDockId, prevDockId, nbrInDS As Integer
    ' Get the number of rows in the dataset table
    nbrInDS = daDockSlip.Tables("DockSlip").Rows.Count - 1
    ' If rows in the table then process
    If nbrInDS > 0 Then
        thisDockId = daDockSlip.Tables("DockSlip").Rows(0).Item("DockId")
        ' Clear array list
        docks.Clear()
        For i = 0 To nbrInDS
            dr = daDockSlip.Tables("DockSlip").Rows(i)
            ' Save value of control-break field
            prevDockId = thisDockId
            location = dr("location")
            electricity = dr("Electricity")
            water = dr("Water")
            ' Create a dock instance
            aDock = New eDock(thisDockId, location, electricity, water)
            docks.Add(aDock)
            ' Loop over rows until a change in dock id
            For j = i To nbrInDS
                dra = daDockSlip.Tables("DockSlip").Rows(j)
                thisDockId = dra("DockId")
                If thisDockId > prevDockId Then Exit For
                slipNo = dra("slipNo")
                width = dra("width")
                slipLength = dra("SlipLength")
                ' Create slip instances for the dock
                aSlip = New Slip(slipNo, width, _
                    slipLength, aDock)
            Next
            i = j - 1
        Next
        daDockSlip = Nothing
    End If
  Catch e As OleDb.OleDbException
    Console.WriteLine(e.Message.ToString)
  End Try
  Return docks
End Function
```

تكرار خارجي

صفات الصنف Dock تستخلص كل دورة خلال التكرار الخارجي

صفات الصنف Slip تستخلص كل دورة خلال التكرار الداخلي

تكرار داخلي

الشكل رقم (١٤,٢٦). منطق كسر التحكم في إجراء GetAll للصنف DockDA.

والآن يجب أن يكون مفهوم Control-Break واضحاً لنا حيث إن أوامر التكرار الخارجي For Next مسؤولة عن استرجاع بيانات رصيف لإنشاء كائن من الصنف mDock ثم إضافته إلى المصفوفة docks، ثم يأتي دور أوامر التكرار الداخلي For Next المسؤولة عن استرجاع بيانات المراسي التي تنتمي إلى الرصيف الحالي وإنشاء كائنات من الصنف Slip لربطها بالكائن الحالي من الصنف mDock، وفي النهاية يتم إعادة المصفوفة docks التي تحتوي على جميع كائنات الصنف Slip.

اختبار البرنامج DockAndSlipDatabase
Testing the DockAndSlipDatabase Application

يحتوي برنامج اختبار البرنامج DockAndSlipDatabase أولاً على توصيف المتغيرات الضرورية وكائن اتصال بقاعدة البيانات، ثم يحاول البحث عن الرصيف رقم ١، فإن وجده فسوف سيعيد كائناً من الصنف mDock الذي يحتوي على بيانات الرصيف رقم ١ والمرتبط بكائنات الصنف Slip، وعندئذ سيتم استدعاء الإجراء printDetails لعرض بيانات الرصيف رقم ١ وبيانات المراسي الذي يحتويها. وبالمثل يحتوي على أوامر أخرى للبحث عن الرصيف رقم ٢ الذي سيتم عرض كل من معلوماته ومعلومات المراسي الذي يحتويها إذا تم الحصول عليه. وأخيراً يحتوي البرنامج على أوامر استدعاء الإجراء GetAll المسؤول عن إعادة مصفوفة تحتوي على كائنات الأرصفة المخزنة في قاعدة البيانات (لاحظ أن كل كائن في هذه المصفوفة يرتبط بمصفوفة أخرى تحتوي على كائنات الصنف Slip المرتبطة به). يوضح الشكل رقم (١٤,٢٧) أوامر هذا البرنامج، كما يوضح الشكل رقم (١٤,٢٨) مخرجات هذا البرنامج.

تطبيق علاقات الجداول الشجرية
Applying Parent-Child (Hierarchical) Dataset Relationships

عادة ما يستخدم أسلوب Control-Break الذي تم شرحه خلال المثال السابق في تطبيقات معالجة التقارير حيث نجد أن هذا الأسلوب جاهز للاستخدام داخل بعض لغات البرمجة (مثل لغة COBOL) التي لديها القدرة على إنتاج التقارير ومعالجتها، أو يوجد في بعض التطبيقات المنفصلة والمتخصصة في إصدار التقارير ومعالجتها مثل برنامج Crystal Reports، وبشكل عام نحتاج إلى منطق Control-Break عندما تكون البيانات مخزنة داخل ملفات (أو داخل جداول مرتبطة في قاعدة البيانات)، ولكن يكون من الأفضل أن يتم تمثيل هذه البيانات في شكل شجري (Hierarchical أو Parent-Child) داخل التطبيق، فعلى سبيل المثال إن العلاقة بين الرصيف والمرسى علاقة شجرية حيث إن كل رصيف يحتوي على العديد من المراسي وكل مرسى يوجد داخل رصيف واحد، ولكي نصدر تقريراً يحتوي على كل من بيانات الأرصفة والمراسي نحتاج أن نربط الجدول DockTable والجدول SlipTable سوياً داخل جدول واحد من الحقول والسجلات (كما هو واضح في الجدول رقم ١٤,٢)، ولكن تذكر أن معلومات كل رصيف تتكرر مع بيانات المراسي التي تتبعه.

```
Imports System.Data.OleDb
Imports System.Collections

Public Class Form1
    Inherits System.Windows.Forms.Form
    Shared aSlip As Slip
    Shared aDock As mDock
    Shared docks, slips As ArrayList
    Shared c As OleDbConnection = DockAndSlipDataBaseConnect.Initialize

    Private Sub testDAClasses()

        ' Initialize the database
        mDock.Initialize(c)
        Try
            aDock = mDock.Find(1)
            Console.WriteLine(vbCrLf & "Results of Find dock 1:")
            printDetails()
        Catch e As NotFoundException
            Console.WriteLine(e.Message.ToString)
        End Try

        Try
            Console.WriteLine(vbCrLf & "Results of Find dock 2:")
            aDock = mDock.Find(2)
            printDetails()
        Catch e As NotFoundException
            Console.WriteLine(e.Message.ToString)
        End Try

        Console.WriteLine(vbCrLf & "Results of GetAll")
        docks = mDock.GetAll
        Dim dcks As mDock
        For Each dcks In docks
            aDock = dcks
            printDetails()
        Next
        MessageBox.Show("Testing Script Complete, See Results")
        mDock.Terminate()
        Me.Close()
    End Sub
    Private Sub printDetails()
        Console.WriteLine(vbCrLf & aDock.TellAboutSelf)
        slips = aDock.GetSlips
        Dim slps As Slip
        For Each slps In slips
            aSlip = slps
            Console.WriteLine("  " + aSlip.TellAboutSelf)
        Next

    End Sub
    Private Sub btnTester1_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnTester1.Click

        testDAClasses()
    End Sub
```

الشكل رقم (١٤,٢٢). برنامج اختبار الصف.

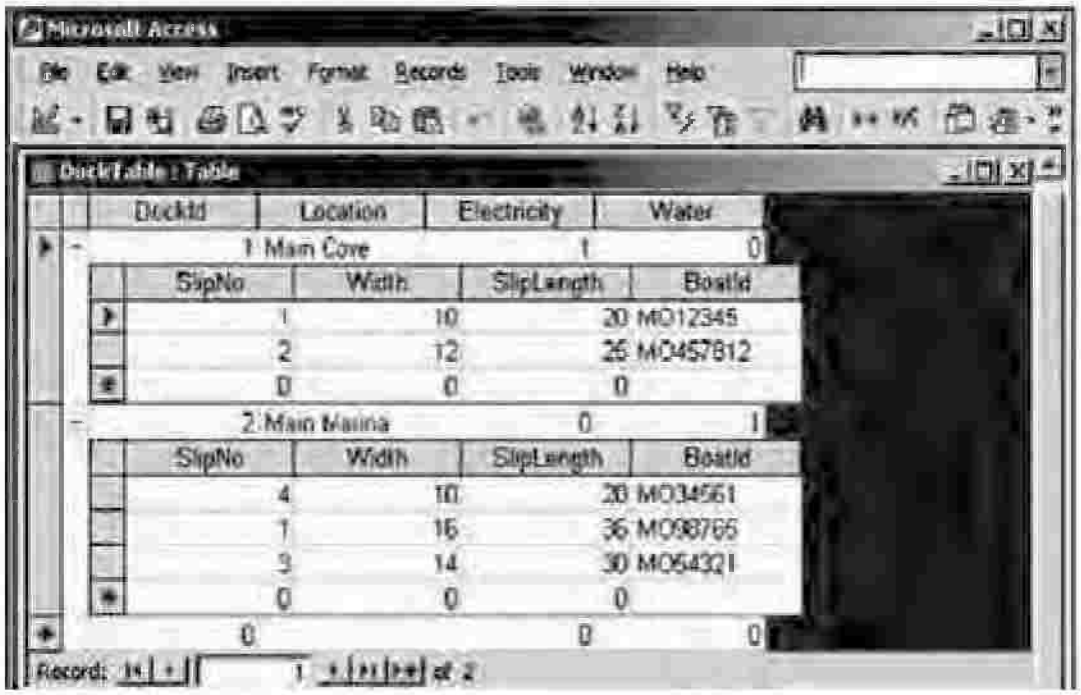
```
Results of Find dock 1:

Dock 1 Location: Main Cove, Has Electricity= True, Has water= False
  Slip 1  Width: 10  Length: 20
  Slip 2  Width: 12  Length: 25

Results of Find dock 2:

Dock 2 Location: Main Marina, Has Electricity= False, Has water= True
  Slip 1  Width: 16  Length: 35
  Slip 3  Width: 14  Length: 30
  Slip 4  Width: 10  Length: 20
Results of GetAll:

Dock 1 Location: Main Cove, Has Electricity= True, Has water= False
  Slip 1  Width: 10  Length: 20
  Slip 2  Width: 12  Length: 25

Dock 2 Location: Main Marina, Has Electricity= False, Has water= True
  Slip 1  Width: 16  Length: 35
  Slip 3  Width: 14  Length: 30
  Slip 4  Width: 10  Length: 20
```

الشكل رقم (١٤,٢٨). مخرجات برنامج اختبار الصنف.

إذا استطعت أن تنشئ علاقة شجرية بين كل من الجدول DockTable والجدول SlipTable، فعندئذ ستصبح أوامر الإجراء GetAll أبسط وأكثر سهولة حيث ستتجول داخل سجلات الجدول DockTable لقراءة بيانات الأرصفة سجلاً تلو الآخر، وعند قراءة بيانات رصيف يتم التجول داخل الجدول SlipTable للحصول على سجلات المراسي المناظرة لهذا الرصيف. يوضح الشكل رقم (١٤,٢٩) العلاقة الشجرية التي تربط كلاً من الجدول DockTable والجدول SlipTable.

يقدم الصنف Dataset إمكانية إنشاء علاقة شجرية بين جدولين، ولتوضيح ذلك يتم إعادة تعريف الإجراء GetAll المعرف سابقاً داخل الصنف DockDA. يبدأ الإجراء GetAll بتعريف المتغير Docks من النوع ArrayList والذي سيحتوي على كائنات الصنف mDock التي سيتم إعادتها في نهاية التنفيذ، ثم تعريف متغيري

إشارة من النوع DataTable (المتغير dtDocks والمتغير dtSlips) حيث يلعب الجدول dtDocks دور الأب ويلعب الجدول dtSlips دور الابن.



الشكل رقم (١٤,٢٩): العلاقة الشجرية الأب–الابن للجدولين DockTable و SlipTable.

ثم يتم تعريف أمري SELECT لاسترجاع بيانات الأرصفة من جدول DockTable واسترجاع بيانات المراسي من جدول SlipTable باستخدام متغيرين من النوع OleDbDataAdapter وملء كائن الصنف Dataset بجدولين نتيجة تنفيذ هذين الأمرين، ثم يتم الإشارة لهذين الجدولين بكل من متغير الإشارة dtDock ومتغير الإشارة dtSlips كما هو واضح من الأوامر التالية:

```
' GetAll method using parent-child approach
Public Shared Function GetAll() As ArrayList
    ' Declare data tables for the dataset
    Dim dtDocks, dtSlips As DataTable
    ' Declare the SQL queries to get each table
    Dim sqlDocks As String = "SELECT DockId, Location, " & _
            "Electricity, Water " & _
        " FROM DockTable ORDER BY DockID"
    Dim sqlSlips As String = "SELECT SlipNo, DockID, " & _
            "Width, SlipLength " & _
        "FROM SlipTable ORDER BY DockId, SlipNo"
    Dim dsDocksSlips As New DataSet("DocksAndSlips")
```

```
Try     ' Create adapaters
    Dim adptDocks As New OleDbDataAdapter(sqlDocks, aConnection)
    Dim adptSlips As New OleDbDataAdapter(sqlSlips, aConnection)
    ' Fill the dataset with two tables, naming them docks and slips
    adptDocks.Fill(dsDocksSlips, "Docks")
    adptSlips.Fill(dsDocksSlips, "Slips")
    ' Set the tables in the datasets to corresponding data tables
    dtDocks = dsDocksSlips.Tables("Docks")
    dtSlips = dsDocksSlips.Tables("Slips")
```

بالإضافة إلى احتواء جداول من النوع DataTable، يسمح صنف Dataset أن ينشئ علاقة ترابط بين جدولين، ولعمل ذلك يتم تعريف كائن من الصنف DataRelation وإسناد اسم له، ثم تخصيص حقل ربط من كل جدول لإنشاء العلاقة الشجرية بين جدول الأب وجدول الابن (الحقل DockId من كل جدول كما هو واضح من الأوامر التالية)، وبعد إنشاء العلاقة يتم إضافتها إلى كائن الصنف Dataset.

```
' Declare a relationship named DocksSlips
' dtDocks is parent table, dtSlips is child table
Dim drDocksSlips As New DataRelation("DocksSlips", _
    dtDocks.Columns("DockId"), _
    dtSlips.Columns("DockId"))

' Add relation to dataset
dsDocksSlips.Relations.Add(drDocksSlips)
```

والآن بعد إضافة العلاقة، يمكنك التجول بين سجلات جدول الأب dtDocks لكي تقرأ سجلاته سجلاً تلو الآخر، ثم تسترجع سجلات المراسي المناظرة لكل سجل وتخزينها داخل المصفوفة drDockslip. لاحظ أن الإجراء GetChildRows هو المسؤول عن استرجاع سجلات الأبناء (المراسي) المناظرة لسجل رصيف ما من جدول الابن، ومن ثم يسهل عليك التجول داخل هذه السجلات، وكما فعلنا سابقاً نكوّن مصفوفة docks من كائنات الصنف mDock المرتبطة بكائنات الصنف Slip، ثم نعيدها كما هو واضح من الأوامر التالية:

```
    ' Declare a datarow for the parent and the child
    ' Declare datarows for all the child rows of the parent
    Dim drDock, drSlip, drDockSlip() As DataRow
    ' Clear array list
    docks.Clear()
    For Each drDock In dtDocks.Rows
        ' Extract the data from the parent table
        id = drDock("DockID")
        location = drDock("Location")
        electricity = drDock("Electricity")
        water = drDock("Water")
```

```
            ' Create a new dock and add to the docks ArrayList
            aDock = New mDock(id, location, electricity, water)
            docks.Add(aDock)
            ' Get all the child rows for this parent
            drDockSlip = drDock.GetChildRows("DocksSlips")
            ' Loop over child rows, extract data and create slip
            For Each drSlip In drDockSlip
                slipNo = drSlip("SlipNo")
                width = drSlip("Width")
                slipLength = drSlip("SlipLength")
                aSlip = New Slip(slipNo, width, slipLength, aDock)
            Next
        Next
        dsDocksSlips = Nothing
    Catch e As OleDb.OleDbException
        Console.WriteLine(e.Message.ToString)
    End Try
    ' Return the docks ArrayList
    Return docks
End Function
```

تطوير الأصناف المرافقة (Association Classes) في تطبيقات قواعد البيانات

Implementing an Association Class in Database Application

لقد تعلمنا في هذا الفصل كيف نطور علاقات ترابط الكائنات "واحد-إلى-واحد" وعلاقة الترابط "واحد-إلى-العديد" في تطبيقات قواعد البيانات، ولتطوير نظام برادشو مارينا نحتاج إلى تطوير صنف المرافقة Lease الذي يرثه الصنف الفرعي AnnualLease والصنف الفرعي DailyLease (لوجود نوعين من العقود)، ولقد قدمنا الصنف Lease في الفصل الثامن ثم استمرت عملية تطوير وظائفه خلال الفصل التاسع دون الأخذ في الاعتبار تخزين العقود المبرمة داخل قاعدة بيانات؛ ولذلك سوف نتعلم خلال بقية هذا الفصل كيف نخزن بيانات صنف المرافقة Lease ونتعامل معها داخل تطبيق قاعدة بيانات.

تقديم جداول قاعدة البيانات CustomerLeaseSlipDatabase

Understanding the Tables in CustomerLeaseSlipDatabase

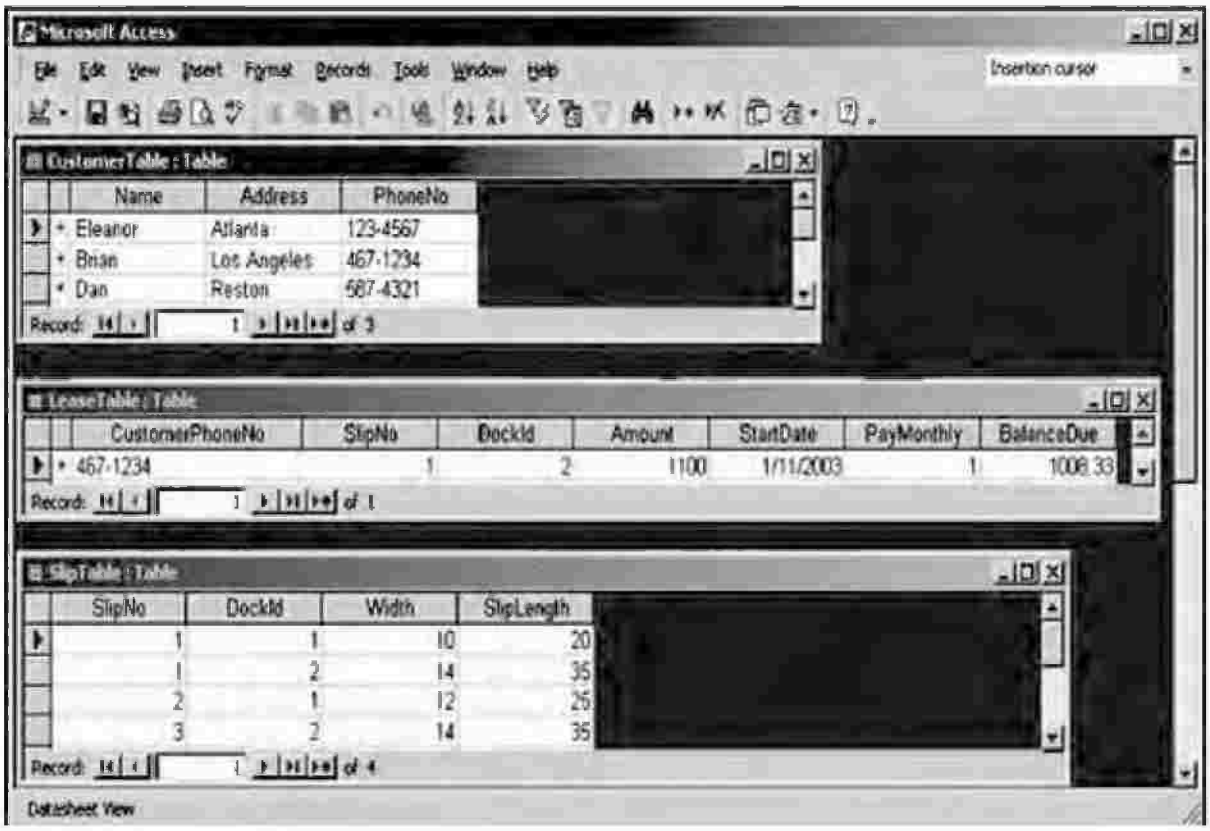
تحتوي قاعدة بيانات هذا التطبيق على ثلاثة جداول وهي: الجدول CustomerTable، والجدول LeaseTable، والجدول SlipTable. وقبل أن نشرع في تطوير هذا البرنامج نحتاج أن نطلع على محتويات هذه الجداول؛ ولذلك اتبع الخطوات التالية:

١- ابحث عن المجلد Ex04 داخل المجلد Chap14\Examples في ملف بيانات الطالب الخاصة بالكتاب ثم قم بنسخه إلى المجلد Chap14\Exercises داخل مجلد العمل الخاص بك.

٢- قم بنسخ الملف CustomerLeaseSlipDatabase.mdb الموجود داخل المجلد Chap14\Exercises\Ex04\Example\ ولاحظ أن قاعدة البيانات تحتوي على ثلاثة جداول (CustomerTable و LeaseTable و SlipTable).

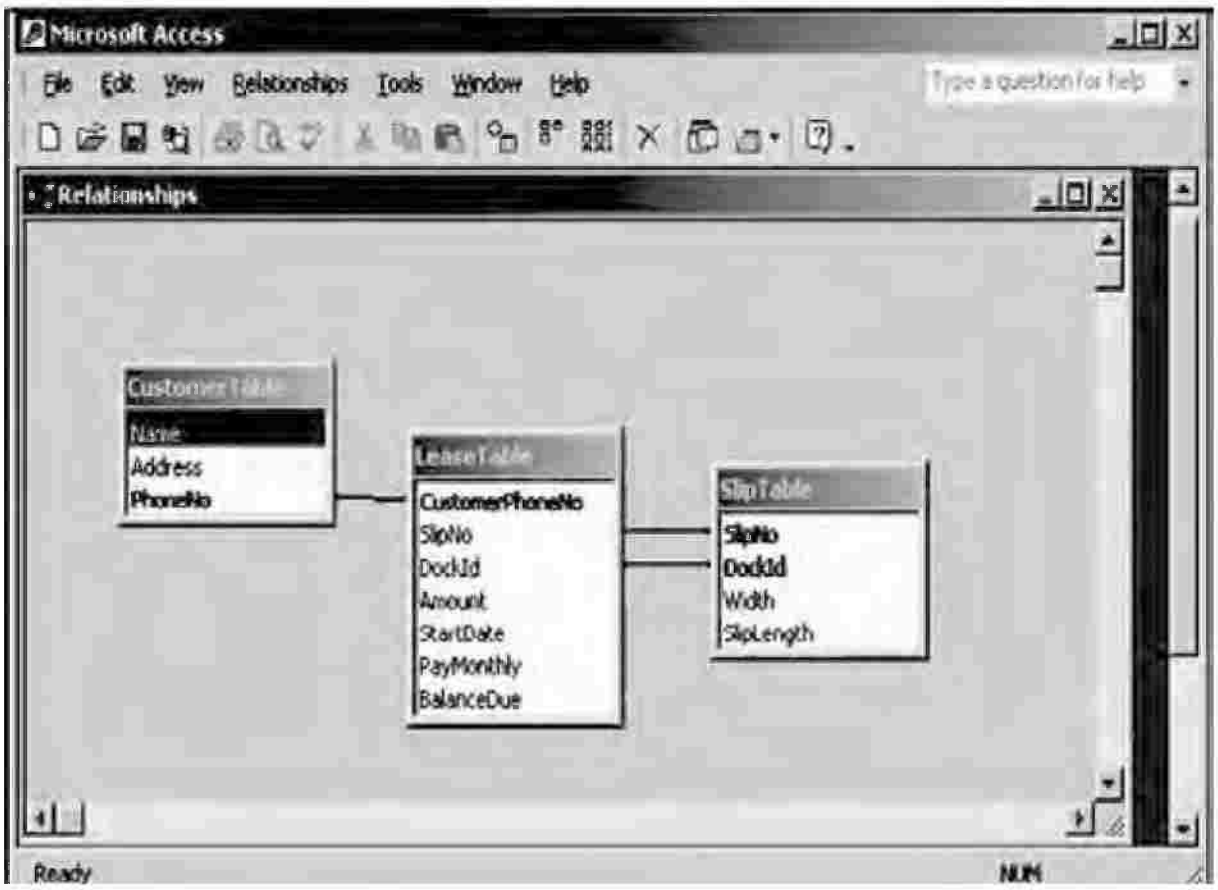
٣- قم بالاطلاع على محتويات هذه الجداول كما هو واضح في الشكل رقم (١٤,٣٠).

٤- تذكر أننا نفترض أن من حق كل عميل أن يتعاقد مرة واحدة؛ ولذلك نستخدم رقم هاتف العميل CustomerPhoneNo كمفتاح أساسي في جدول LeaseTable، كما أن رقم هاتف العميل PhoneNo يستخدم كمفتاح أساسي في الجدول CustomerTable، وبذلك سيتم الربط بين الجدولين باستخدام هذين الحقلين.



الشكل رقم (١٤,٣٠). محتويات CustomerTable و LeaseTable و SlipTable.

٥- كما يحتوي الجدول LeaseTable على كل من الحقل SlipNo (رقم المرسى) والحقل DockId (رقم الرصيف) اللذان يعملان كمفتاح أجنبي ويستخدمان للربط مع جدول SlipTable (انظر الشكل رقم ١٤,٣١).



الشكل رقم (١٤,٣١). نموذج علاقات CustomerTable و LeaseTable و SlipTable.

يحتوي تطبيق DockAndSlipDatabase على أربعة أصناف مجال مشكلة وهي: الصنف Customer، والصنف Slip، والصنف Lease، والصنف AnnualLease، كما يحتوي على ثلاثة أصناف التعامل مع البيانات (الصنف CustomerDA، والصنف SlipDA، والصنف LeaseDA)، هذا بالإضافة إلى كل من صنف الاختبار وصنف الاتصال بقاعدة البيانات، ومن ثم سوف نناقش هذه الأصناف التي لم يتم مناقشتها سابقاً.

إنشاء اتصال بقاعدة البيانات CustomerLeaseSlipDatabase
Establishing a Connection to CustomerLeaseSlipDatabase

كما فعلنا مع الأمثلة السابقة، نحتاج أن ننشئ كائن اتصال بقاعدة البيانات ليكون مشتركاً بين كل من أصناف مجال المشكلة وأصناف التعامل مع البيانات، وتتشابه أوامر صنف الاتصال CustomerLeaseSlipConnection مع بقية أصناف اتصال هذا الفصل عدا بالطبع اسم مصدر البيانات (قاعدة البيانات)، وبذلك لم نناقش هذا الصنف مرة ثانية (يوجد تعريف لهذا الصنف داخل ملفات الطالب الخاص بهذا الكتاب).

تعديل الصنف Customer
Modifying the Customer Class

يجب أن يحتوي تعريف الصنف Customer على متغير إشارة من الصنف Lease لإنشاء علاقة ترابط بين العميل والعقد، ولكن تذكر أنه لا يمكن إنشاء كائن من الصنف Lease لأنه صنف مجرد (Abstract Class) بل يمكن إنشاء كائنات من الأصناف الفرعية التي ترث هذا الصنف (الصنف AnnualLease والصنف DailyLease والتي تمثل أنواع العقود، وبما أننا نفترض أن العقد الذي يمكن أن يعقد في هذا المثال هو العقد السنوي فقط فإن متغير الإشارة سيكون من النوع AnnualLease وإسناد القيمة Nothing عند تنفيذ إجراء إنشاء الصنف Customer، كما يجب تعريف إجراءات مرور (الإجراء SetLease والإجراء GetLease) لإسناد كائن الصنف AnnualLease واسترجاعه. يوضح الشكل رقم (١٤,٣٢) جزءاً من تعريف الصنف Customer، أما تعريف الصنف كاملاً فيوجد داخل المجلد Chap14\Examples\Ex04.

```
Customer Class with added methods for DA Class
' Chapter 14 -- Example4
Imports System.Data.OleDb
Public Class Customer
    ' Attributes
    Private name As String
    Private address As String
    Private phoneNo As String
    ' Reference variable for lease instance
    Private lease As AnnualLease

    ' Constructor (3 parameters)
    Public Sub New(ByVal sName As String, ByVal snAddress As String, _
        ByVal sPhoneNo As String)

        name = sName
        address = snAddress
        phoneNo = sPhoneNo
        SetLease(Nothing)       'Initially set to nothing
    End Sub
    Public Function GetLease() As AnnualLease
        Return lease
    End Function

    Public Sub SetLease(ByVal aLease As AnnualLease)
        lease = aLease
    End Sub
```

الشكل رقم (١٤,٣٢). صنف العميل بصلة الإيجار.

تعديل تعريف كل من الصنف Lease والصنف AnnualLease
Modifying the Lease and AnnualLease Classes

نحتاج أيضاً أن نجري التعديلات نفسها على صنف مجال المشكلة Lease حيث يجب ربط العقد بكل من المرسى محل العقد (Slip) والعميل (Customer)، ولذلك سيتم إضافة متغير إشارة من النوع Slip ومتغير إشارة من

النوع Customer وإسناد قيمة Nothing لكل منهما داخل إجراء إنشاء الصنف Lease، ونحتاج أيضاً إلى إضافة إجراءات مرور لإسناد كائنات من وإلى هذين المتغيرين واسترجاعها. وتذكر أن الجدول LeaseTable يحتوي على بيانات العقود السنوية فقط (كائنات الصنف AnnualLease)؛ ولذلك لم نعرف صنف تعامل مع البيانات لصنف مجال المشكلة Lease، ولكن سنعرف الصنف AnnualLeaseDA لتخزين كائنات الصنف AnnualLease ومعالجتها، ويوضح الشكل رقم (١٤,٣٣) تعريف الصنف Lease.

وبناء على ذلك سنحتاج إلى إضافة إجراءات التعامل مع البيانات داخل الصنف AnnualLease (الإجراء Initialize، والإجراء Terminate، والإجراء Find، والإجراء AddNew) كما هو واضح في الشكل رقم (١٤,٣٤) الذي يوضح أول جزء من تعريف هذا الصنف، وتعريف إجراءات التعامل مع قاعدة البيانات، مع العلم أن تعريف الصنف بشكل كامل يوجد في المجلد Example4 في ملفات الطالب.

تعديل الصنف Slip
Modifying the Slip Class

ويجب أيضاً أن نربط كائنات الصنف Slip بكل من كائن الصنف Customer وكائن الصنف Lease المناظرين، ولعمل ذلك يجب تعريف متغير إشارة من الصنف AnnualLease وإسناد القيمة Nothing له في إجراء إنشاء الصنف Slip (ولاحظ أن الصنف Slip يحتوي على متغير إشارة من النوع Customer) كما هو واضح من الأوامر التالية:

```
' Chapter 14 Example4
' Slip Class
Imports System.Data.OleDb
Imports System.Collections
Public Class Slip
    ' Attributes
    Private slipId As Integer
    Private slipWidth As Integer
    Private slipLength As Integer

    ' Reference variables
    Private DockId As Integer
    Private lease As AnnualLease

    ' Constructor (three parameters plus a Dock reference)
    Public Sub New(ByVal aSlipId As Integer, ByVal aSlipWidth As Integer, _
        ByVal aSlipLength As Integer, ByVal aDockId As Integer)
        'invoke setter methods to populate attributes
        SetSlipId(aSlipId)
        SetSlipWidth(aSlipWidth)
        SetSlipLength(aSlipLength)
        'invoke custom method for Dock association
        setDock(aDockId)
        setLease(Nothing)
    End Sub
```

```
Lease -- abstract association class
' between Slip and Customer

Public MustInherit Class Lease

    ' Attributes
    Private amount As Double
    Private startDate As DateTime
    Private endDate As DateTime

    ' References to slip and customer
    Private theSlip As Slip
    Private theCustomer As Customer

    ' Constructor (1 parameter)
    Public Sub New(ByVal aStartDate As DateTime)
        SetStartDate(aStartDate)
        SetEndDate(Nothing)   ' endDate set by subclass
        SetAmount(0)
        ' No customer or slip yet, set by subclass
        SetCustomer(Nothing)
        SetSlip(Nothing)
    End Sub

    ' TellAboutSelf method
    Public Overridable Function TellAboutSelf() As String
        Return ("Start Date: " & startDate & _
        " End Date: " & endDate & _
        " Lease Amount=" & amount)
    End Function

    ' Custom method CalculateFee based on slip width
    ' Abstract method all subclasses must implement
    Public MustOverride Function CalculateFee(ByVal aWidth As Integer) _
    As Single

    'Get accessor methods
    Public Function GetStartDate() As DateTime
        Return startDate
    End Function
    Public Function GetEndDate() As DateTime
        Return endDate
    End Function
    Public Function GetAmount() As Double
        Return amount
    End Function
    Public Function GetCustomer() As Customer
        Return theCustomer
    End Function
    Public Function GetSlip() As Slip
        Return theSlip
    End Function
```

الشكل رقم (١٤,٣٣). صنف الإيجار لصفات العميل والمرسى.

```
    ' Set accessor methods
    Public Sub SetStartDate(ByVal aStartDate As DateTime)
        startDate = aStartDate
    End Sub
    Public Sub SetEndDate(ByVal anEndDate As DateTime)
        endDate = anEndDate
    End Sub
    Public Sub SetAmount(ByVal anAmount As Double)
        Amount = anAmount
    End Sub
    Public Sub SetCustomer(ByVal aCustomer As Customer)
        theCustomer = aCustomer
    End Sub
    Public Sub SetSlip(ByVal aSlip As Slip)
        theSlip = aSlip
    End Sub

End Class
```

تابع الشكل رقم (١٤,٣٣).

```
' Chapter 14 -- Example4
' AnnualLease -- a subclass of Lease
Imports System.Data.OleDb

Public Class AnnualLease
    Inherits Lease

    ' Attributes in addition to those inherited from Lease
    Private balanceDue As Single
    Private payMonthly As Boolean

    ' Constructor (three values as parameters)
    Public Sub New(ByVal aStartDate As DateTime, _
     ByVal aSlipWidth As Integer, _
     ByVal isPayMonthly As Boolean)
        ' Invoke superclass constructor
        MyBase.New(aStartDate)
        ' Calculate end date
        Dim yearLater As DateTime = aStartDate.AddYears(1)
        ' Invoke superclass method to set end date
        SetEndDate(yearLater)
        ' Invoke superclass SetAmount method after getting
        ' fee amount from CalculateFee method
        SetAmount(CalculateFee(aSlipWidth))
        ' Set payMonthly
        SetPayMonthly(isPayMonthly)
        ' Set balance due if applicable
        If payMonthly Then
            SetBalanceDue(GetAmount() - GetAmount() / 12)
        Else
            SetBalanceDue(0)
        End If
    End Sub

    ' DA shared methods-----------------------------------------
    Public Shared Sub Initialize(ByVal c As OleDbConnection)
        AnnualLeaseDA.Initialize(c)
    End Sub
    Public Shared Function Find(ByVal key As String) As AnnualLease
        Return AnnualLeaseDA.Find(key)
    End Function
    Public Shared Sub Terminate()
        AnnualLeaseDA.Terminate()
    End Sub

    ' DA non-shared method
    Public Sub AddNew()
        AnnualLeaseDA.AddNew(Me)
    End Sub
```

الشكل رقم (١٤,٣٤). صنف AnnualLease يحتوي إجراءات DA.

ويجب إضافة إجراءي المرور Getter و Setter لاسترجاع صفات كائن من الصنف lease وإسنادها.

```
' Get accessor methods
Public Function GetSlipId() As Integer
    Return slipId
End Function
Public Function GetSlipWidth() As Integer
    Return slipWidth
End Function
Public Function GetSlipLength() As Integer
    Return slipLength
End Function
Public Function GetDockID() As Integer
    Return DockId
End Function
Public Function getLease()
    Return lease
End Function

' Set accessor methods
Public Sub SetSlipId(ByVal aSlipId As Integer)
    slipId = aSlipId
End Sub
Public Sub SetSlipWidth(ByVal aSlipWidth As Integer)
    slipWidth = aSlipWidth
End Sub
Public Sub SetSlipLength(ByVal aSlipLength As Integer)
    slipLength = aSlipLength
End Sub
Public Sub SetDock(ByVal aDockId As Integer)
    DockId = aDockId
End Sub
Public Sub SetLease(ByVal aLease As AnnualLease)
    lease = aLease
End Sub
```

ثم يأتي تعريف ثلاثة إجراءات للتعامل مع قواعد البيانات كما هو واضح من الأوامر التالية وهي: (الإجراء Initialize، والإجراء Terminate، والإجراء Find). وتتشابه أوامر هذه الإجراءات مع الإجراءات المماثلة لها في أصناف مجال المشكلة عدا أن الإجراء Find يستقبل معاملين (رقم الصنف ورقم المرسى)، حيث يتم تعريف المرسى بكل من رقم الرصيف ورقم المرسى (المفتاح الأساسي للجدول SlipTable)

```
' DA shared Methods-------------------------------------
Public Shared Sub Initialize(ByVal c As OleDbConnection)
    SlipDA.Initialize(c)
End Sub
Public Shared Function Find(ByVal aSlipNo As Integer, _
```

```
        ByVal aDockId As Integer) As Slip
        Return SlipDA.Find(aSlipNo, aDockId)
End Function
Public Shared Sub Terminate()
    SlipDA.Terminate()
End Sub
```

كما نحتاج إلى تعريف الإجراء الخاص LeaseAnnualSlip لإنشاء العقد السنوي وربطه بكل من المرسى محل العقد والعميل حيث يتطلب هذا الإجراء استقبال ثلاثة معاملات كما هو واضح من الأوامر التالية وهي: متغير إشارة لكائن الصنف Customer (العميل)، ومتغير إشارة لكائن الصنف DateTime (تاريخ بداية العقد)، ومتغير من النوع Boolean للإشارة إلى أسلوب طريقة الدفع (نقداً أو أقساط)، ومن ثم سيتم استخدام كل من تاريخ بداية العقد وأسلوب الدفع مع عرض المرسى لإنشاء كائن من الصنف AnnualLease.

```
' Custom method LeaseAnnualSlip creates AnnualLease instance
Public Function LeaseAnnualSlip(ByVal aCustomer As Customer, _
    ByVal aStartDate As DateTime, _
    ByVal isPaymonthly As Boolean) As AnnualLease

    ' Create AnnualLease instance and assign it to lease attributes
    ' width is an attribute of this slip
    lease = New AnnualLease(aStartDate, slipWidth, isPaymonthly)
```

يستدعي الإجراء LeaseAnnualSlip كلاً من الإجراء SetSlip والإجراء SetCustomer من كائن الصنف Lease لربطه بكل من المرسى والعميل، ثم يستدعي الإجراء SetLease من كائن الصنف Customer لربطه بالعقد، ثم إرجاع كائن الصنف Lease هكذا:

```
    ' Tell lease to set its slip to this slip
    lease.SetSlip(Me)
    ' Tell lease to set its customer
    lease.SetCustomer(aCustomer)
    ' Tell customer to set its lease
    aCustomer.SetLease(lease)
    Return lease
End Function
```

تقديم الصنف SlipDA
Introducing the SlipDA Class

ذكرنا في المثال السابق أنه لا توجد حاجة لتعريف الصنف SlipDA لأن وظيفة البحث ووظيفة استرجاع جميع بيانات الأرصفة والمراسي التابع لها ستكون مهمة الصنف DockDA وذلك بتعريف الإجراء Find والإجراء GetAll داخل الصنف DockDA، ولكن يختلف الحال في هذا المثال حيث نحتاج أن نبحث عن مرسى محدداً لكي يتم تأجيره لعميل ما؛ ولذلك نحتاج إلى تعريف الصنف SlipDA.

يبدأ تعريف الصنف SlipDA بتعريف صفات صنف مجال المشكلة Slip وتعريف متغيرات للاتصال بقاعدة البيانات، ثم تعريف كل من الإجراء Initialize والإجراء Terminate اللذيْن لا يختلفان عن مثيليهما في أصناف التعامل مع البيانات الأخرى كما هو واضح من الأوامر التالية:

```
' Chapter 14 -- Example4
' SlipDA Class
Imports System.Data.OleDb
Public Class SlipDA
    Shared aSlip As Slip

    ' Declare a variables for the database connection
    Shared aConnection As OleDbConnection

    ' Declare variables for Slip attribute values
    Shared slipNo, slipWidth, dockId As Integer
    Shared slipLength As Double

    ' Establish the database connection
    Public Shared Sub Initialize(ByVal c As OleDbConnection)
        Try
            ' Try to open the connection
            aConnection = c
            If Not aConnection.State.Open Then
                aConnection.Open()
            End If
        Catch e As Exception
            Console.WriteLine(e.Message.ToString)
        End Try
    End Sub

    ' Terminate the connect
    Public Shared Sub Terminate()
        ' Define string and assign to file
        Try
            aConnection.Close()
            aConnection = Nothing
        Catch e As Exception
            Console.WriteLine(e.ToString)
        End Try
    End Sub
```

ثم يبقى تعريف الإجراء Find المسؤول عن استقبال كل من رقم الرصيف aDockId ورقم المرسى aSlipNo، ثم استرجاع بيانات المرسى المناظر لهما من الجدول SlipTable التي ستستخدم في إنشاء كائن من الصنف Slip وإعادته إلى الإجراء الذي يستدعي الإجراء Find وهكذا.

```
    ' Find method
Public Shared Function Find(ByVal aSlipNo As Integer, _
    ByVal aDockId As Integer) As Slip

    aSlip = Nothing
    ' Define the SQL statement
    Dim sqlQuery As String = "SELECT SlipNo, DockID," & _
        "Width, SlipLength FROM SlipTable " & _
        "WHERE SlipNo = " & aSlipNo & " AND DockId = " & aDockId & " "

    Dim dockDS As New DataSet()

    Try
        ' Get the Dock
        Dim adptDock As New OleDbDataAdapter(sqlQuery, aConnection)
        adptDock.Fill(dockDS, "Docks")
        Dim dr As DataRow
        ' Check for rows in the data set table
        If dockDS.Tables("Docks").Rows.Count > 0 Then
            dr = dockDS.Tables("Docks").Rows(0)
            slipNo = dr("SlipNo")
            dockId = dr("DockId")
            slipWidth = dr("Width")
            slipLength = dr("SlipLength")
            ' Create a slip instance
            aSlip = New Slip(slipNo, slipWidth, slipLength, dockId)
        Else    ' Dock was not found
            Throw New NotFoundException("Not Found")

        End If
        dockDS = Nothing

    Catch e As OleDb.OleDbException
        Console.WriteLine(e.Message.ToString)

    End Try
    Return aSlip
End Function
```

تقديم الصنف AnnualLeaseDA

Introducing the AnnualLeaseDA Class

وبالطبع نحتاج إلى تعريف الصنف AnnualLeaseDA لتوفير كل من إمكانية البحث عن بيانات عقد سنوي ما من قاعدة البيانات أو إمكانية استرجاعها. يبدأ تعريف هذا الصنف بتوصيف متغيرات صنف مجال المشكلة AnnualLease كالعادة بتعريف كل من الإجراء Initialize والإجراء Terminate هكذا:

```
' Chapter 14 -- Example4
' AnnualLeaseDA class
Imports System.Data.OleDb
```

```
Public Class AnnualLeaseDA
    Shared aCustomer As customer
    Shared aSlip As slip
    Shared aLease As AnnualLease
    Shared payMonthly As Boolean

    Shared name, address, phoneNo As String

    Shared slipNo, slipWidth, dockId As Integer
    Shared slipLength As Double

    Shared amount, balanceDue As Double
    Shared startDate As DateTime

    ' References to customer and slip
    Dim customer As customer
    Dim slip As slip

    ' Declare a variables for the database connection
    Shared aConnection As OleDbConnection

    Public Shared Sub Initialize(ByVal c As OleDbConnection)
        Try
            ' Try to open the connection
            aConnection = c
            If Not aConnection.State.Open Then
                aConnection.Open()
            End If
        Catch e As Exception
            Console.WriteLine(e.Message.ToString)
        End Try
    End Sub
    Public Shared Sub Terminate()
        ' Define string and assign to file
        Try
            aConnection.Close()
            aConnection = Nothing
        Catch e As Exception
            Console.WriteLine(e.ToString)
        End Try
    End Sub
```

الإجراء Find للصنف AnnualLeaseDA

Understanding the Find Method of the AnnualLeaseDA Class

بما أن الإجراء Find المعرف داخل الصنف AnnualLeaseDA سيكون مسؤولاً عن إعادة كائن من الصنف AnnualLease والمرتبط بكل من كائن من الصنف Customer وكائن الصنف Slip، فسيحتاج الإجراء Find أن يستخدم الأمر Select لاسترجاع بيانات من ثلاثة جداول وهي: الجدول SlipTable، والجدول CustomerTable، والجدول LeaseTable. وتذكر أن رقم هاتف العميل يستخدم كمفتاح أساسي في كل من الجدول CustomerTable

والجدول LeaseTable؛ ولذلك يمكن ربط هذين الجدولين باستخدام رقم هاتف العميل، وتذكر أن كلاً من رقم المرسى ورقم الرصيف يستخدمان كمفاتيح أجنبية داخل الجدول LeaseTable لربطه بالجدول SlipTable ومن ثم سيكون أمر Select هكذا:

```
' Find method
Public Shared Function Find(ByVal key As String) As AnnualLease
    ' Retrieve Lease, Customer and Slip Data
    aCustomer = Nothing
    ' Define the SQL query
    Dim sqlQuery As String = "SELECT Name, Address, PhoneNo, " & _
        "LeaseTable.SlipNo, Amount, StartDate, " & _
        "PayMonthly, BalanceDue, " & _
        "SlipTable.DockId, Width, SlipLength " & _
        "FROM CustomerTable, LeaseTable, SlipTable " & _
        "WHERE PhoneNo = '" & key & "' " & _
        "AND CustomerPhoneNo = '" & key & "' " & _
        "AND LeaseTable.SlipNo = SlipTable.SlipNo " & _
        "AND LeaseTable.DockId = SlipTable.DockId"
```

يتم تخزين البيانات التي تم استرجاعها من الجداول داخل كائن الصنف Dataset والتي يتم بعد ذلك استخدامها لتكوين كائنات من كل من الصنف Customer والصنف AnnualLease والصنف Slip، وبما أن لغة VB .NET تستخدم القيمة صفر للتعبير عن القيمة False وتستخدم أي قيمة أخرى للتعبير عن القيمة True فلا نحتاج إلى تحويل القيمة العائدة من حقل قاعدة البيانات PayMonthly هكذا:

```
Dim ALds As New DataSet()
Dim dr As DataRow
Try
    Dim adptAL As New OleDbDataAdapter(sqlQuery, aConnection)
    adptAL.Fill(ALds, "ALtable")
    If ALds.Tables("altable").Rows.Count > 0 Then
        dr = ALds.Tables("altable").Rows(0)
        ' Get the values from the datarow
        ' Customer info
        name = dr("name")
        address = dr("address")
        phoneNo = dr("phoneno")
        ' Slip info
        slipNo = dr("slipNo")
        amount = dr("amount")
        startDate = dr("startdate")
        payMonthly = dr("paymonthly")
        balanceDue = dr("balanceDue")
        ' Dock info
        dockId = dr("dockId")
        slipWidth = dr("Width")
        slipLength = dr("slipLength")
```

وبعد إنشاء كائنات الصنف AnnualLease والصنف Customer والصنف Slip، يستدعي كائن الصنف AnnualLease كلاً من الإجراء SetCustomer والإجراء SetSlip لربطهما بكل من كائن الصنف Customer وكائن الصنف Slip، ثم يتم استدعاء الإجراء SetLease من كائن الصنف Slip لربطه بكائن الصنف AnnualLease (العقد)، وأخيراً يتم استدعاء الإجراء SetLease من كائن الصنف Customer لربطه بكائن الصنف AnnualLease كما هو واضح في الأوامر التالية. وإذا افترضنا عدم حدوث أخطاء عند تنفيذ هذه الأوامر فسيعيد الإجراء Find كائناً من الصنف AnnualLease مرتبطاً بكائن من الصنف Customer وكائن من الصنف Slip.

```
            ' Create Customer, Lease, & Slip instances
            aCustomer = New Customer(name, address, phoneNo)
            aLease = New AnnualLease(startDate, slipWidth, payMonthly)
            aSlip = New Slip(slipNo, slipWidth, slipLength, dockId)

            aLease.SetCustomer(aCustomer)    ' Link Lease to Customer
            aSlip.SetLease(aLease)           ' Link Slip to Lease
            aCustomer.SetLease(aLease)       ' Link Customer to Lease
            aLease.SetSlip(aSlip)            ' Link Lease to Slip

        Else    ' Nothing was retrieved
            Throw New NotFoundException("Not Found")
            ALds = Nothing
            adptAL = Nothing
        End If
    Catch e As OleDb.OleDbException
        Console.WriteLine(e.Message.ToString)
    End Try
    Return aLease
End Function
```

الإجراء AddNew للصنف AnnualLeaseDA

Understanding the AddNew Method of the AnnualLeaseDA Class

يستخدم الإجراء AddNew لإضافة بيانات عقد سنوي في قاعدة البيانات؛ ولذلك يستقبل هذا الإجراء كائناً من الصنف AnnualLease، ثم يسترجع قيمة العقد والمبلغ المتبقي وتاريخ بداية العقد ونوع الدفع (PayMonthly) التي يتم تحويلها إلى صفر أو واحد لتخزينها في قاعدة البيانات وذلك باستخدام إجراءات المرور لهذا الكائن، ثم يسترجع كلاً من كائن الصنف Customer وكائن الصنف Slip المرتبطيْن بهذا الكائن كما هو واضح من الأوامر التالية:

```
Public Shared Sub AddNew(ByVal aLease As AnnualLease)
        ' Declare integer variable for payMonthly
        Dim intPayMonthly As Integer

        ' Retrieve the lease attributes
```

```
        amount = aLease.GetAmount
        balanceDue = aLease.GetBalanceDue
        payMonthly = aLease.GetPayMonthly
        If payMonthly = True Then
            intPayMonthly = 1
        Else
            intPayMonthly = 0
        End If
        startDate = aLease.GetStartDate

        aSlip = aLease.GetSlip
        aCustomer = aLease.GetCustomer
```

ثم يتم استرجاع رقم هاتف العميل من كائن الصنف Customer وكلاً من رقم المرسى ورقم الرصيف من كائن الصنف Slip هكذا:

```
' Retrieve the customer's phoneno
phoneNo = aCustomer.GetPhoneNo
' Retrieve the slipno and dockid from the slip
slipNo = aSlip.GetSlipId
dockId = aSlip.GetDockID
```

وقبل أن يتم إضافة سجل العقد السنوي إلى قاعدة البيانات بواسطة تنفيذ الأمر INSERT يجب أولاً أن نتأكد من عدم وجود عقد آخر بالمفتاح الأساسي نفسه؛ ولذلك سيتم استدعاء الإجراء Find من الصنف AnnualLease ممرراً له رقم هاتف العميل، فإذا تم الحصول عليه فسيتم إعادة استثناء من النوع DuplicateException، وإذا لم يتم الحصول عليه فسيتم إضافته إلى قاعدة البيانات هكذا:

```
        ' Create the SQL statement
        Dim sqlInsert As String = "INSERT INTO LeaseTable " & _
            "VALUES ('" & phoneNo & "', " & slipNo & ", " & _
            "" & dockId & ", " & amount & ", '" & startDate & "', " & _
            "" & intPayMonthly & ", " & balanceDue & " )"

        Dim adptAL As New OleDbDataAdapter()
        ' Determine if this lease already exists in the database
        Try
            Dim a As AnnualLease = AnnualLease.Find(phoneNo)
            Throw New DuplicateException("Lease Exists")
        Catch e As NotFoundException
            Try ' Assign Insert Commands and Execute
                adptAL.InsertCommand = New OleDbCommand(sqlInsert)
                adptAL.InsertCommand.Connection = aConnection
                adptAL.InsertCommand.ExecuteNonQuery()
            Catch ee As OleDb.OleDbException
                Console.WriteLine(ee.Message.ToString)
            End Try
        End Try
    End Sub
End Class
```

اختبار تطبيق قاعدة البيانات CustomerLeaseSlip

Testing the CustomerLeaseSlip Database Application

تبدأ أوامر برنامج الاختبار بتنفيذ أمر Imports لاستدعاء أصناف التعامل مع المجموعات وقاعدة البيانات، ثم تعريف متغيرات إشارة لكل من الصنف Customer والصنف AnnualLease والصنف Slip، وأخيراً إنشاء كائن اتصال بقاعدة البيانات بواسطة استدعاء الإجراء Initialize لكل من الصنف Customer والصنف AnnualLease والصنف Slip هكذا:

```
' Chapter 14 -- Example4
' Testing script
Imports System.Data.OleDb
Imports System.Collections

Public Class Form1
    Inherits System.Windows.Forms.Form

    Shared c As OleDbConnection = CustomerLeaseSlipConnect.initialize
    ' Declare instances of AnnualLease, Customer, and Slip
    Shared aLease As AnnualLease = Nothing
    Shared aCustomer As Customer = Nothing
    Shared aSlip As Slip = Nothing

    Private Sub testDAClasses()
        ' Initialize the database connections

        Customer.Initialize(c)
        AnnualLease.Initialize(c)
        Slip.Initialize(c)
```

ولإنشاء عقد سنوي جديد للعميل الذي يملك رقم الهاتف "123-4567" لإيجار المرسى رقم ١ داخل الرصيف رقم ١ وإضافته إلى قاعدة البيانات، يستدعي برنامج الاختبار الإجراء Find من الصنف Customer ممرراً رقم الهاتف "123-4567" لاسترجاع كائن من الصنف Customer، وبالمثل يستدعي الإجراء Find من الصنف Slip ممرراً له من المرسى رقم ١ والرصيف رقم ١ لاستقبال كائن الصنف Slip، ثم يتم استدعاء الإجراء LeaseAnnualSlip من كائن الصنف Slip لإنشاء العقد (كائن من الصنف AnnualLease) هكذا:

```
Try     'Get a Customer and Slip
    aCustomer = Customer.Find("123-4567")
    Console.WriteLine(vbCrLf & "Customer Information:" & _
        vbCrLf & aCustomer.TellAboutSelf)
    aSlip = Slip.Find(1, 1)
    Console.WriteLine(vbCrLf & "Slip Information:" & _
        vbCrLf & aSlip.TellAboutSelf)
```

```
    ' Lease slip to customer
    Dim aStartDate As DateTime = #8/26/2003#
    Dim payMonthly As Boolean = False

    aLease = aSlip.LeaseAnnualSlip(aCustomer, aStartDate, payMonthly)

Catch e As NotFoundException
    Console.WriteLine(e.Message.ToString)
End Try
```

ثم يتم استدعاء الإجراء AddNew من كائن الصنف AnnualLease لإضافة العقد الجديد إلى قاعدة البيانات، فإذا كان العميل الحالي متعاقد بالفعل على مرسى آخر، فعندئذ سنستقبل استثناء من النوع DuplicatExption هكذا:

```
Try     ' Add a new lease to the database
    aLease.addNew()
    Console.WriteLine(vbCrLf & "New lease record added")
Catch e As DuplicateException
    Console.WriteLine(vbCrLf & "Lease Already Exists")
End Try
```

ثم يحاول البرنامج استرجاع بيانات هذا العقد ثانياً من قاعدة البيانات؛ ولذلك سيتم استدعاء الإجراء Find من الصنف AnnualLease ممرراً له رقم هاتف العميل، وعندئذ سيجد الإجراء Find عقداً سنوياً لهذا الرقم ويعيد كائناً من الصنف AnnualLease المرتبط بكائن من الصنف Customer وكائن من الصنف Slip؛ ولذلك سيتم استدعاء الإجراء TellAboutSelf من كلا الكائنين لعرض معلوماتهما، وأخيراً يستدعي الاختبار أوامر غلق قاعدة البيانات هكذا:

```
    Try
        aLease = AnnualLease.Find("123-4567")
        aLease.GetCustomer()
        aLease.GetSlip()
        Console.WriteLine(vbCrLf & "Lease Information: " & _
            aLease.TellAboutSelf & vbCrLf & _
            aCustomer.TellAboutSelf & vbCrLf & aSlip.TellAboutSelf)
    Catch ee As NotFoundException
        Console.WriteLine(ee.Message.ToString)
        Customer.Terminate()
        Slip.Terminate()
        AnnualLease.Terminate()
        CustomerLeaseSlipConnect.Initialize()

    End Try

    MessageBox.Show("TestScript Completed--Review Output")
```

```
    Customer.Terminate()
    AnnualLease.Terminate()
    Slip.Terminate()

    Me.Close()
End Sub

Private Sub btnTester1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles _ btnTester1.Click

    testDAClasses()
End Sub
```

يعرض الشكل رقم (١٤,٣٥) مخرجات برنامج الاختبار.

```
Customer Information:
Name = Eleanor, Address = Atlanta, Phone No = 123-4567

Slip Information:
Slip 1  Width: 10  Length: 20

New lease record added

Lease Information: Start Date: 8/26/2003  End Date: 8/26/2004 Lease Amount=800,
   BalanceDue=0  PayMonthly: False
Name = Eleanor, Address = Atlanta, Phone No = 123-4567
Slip 1  Width: 10  Length: 20
```

الشكل رقم (١٤,٣٥). مخرجات برنامج الاختبار.

ملخص الفصل

Chapter Summary

- تحتوي قاعدة بيانات التطبيقات الحقيقية على العديد من الجداول حيث توزع البيانات على أكثر من جدول، فعلى سبيل المثال يمكن ربط بيانات العملاء ببيانات المراكب التي يمتلكونها بواسطة رقم هاتف العميل.
- يمثل المفتاح الرئيس (Primary Key) الحقل الذي يميز كل سجل عن الآخر داخل جدول قواعد البيانات العلاقية.

- يمثل الحقل الأجنبي (Foreign Key) الحقل الذي يوجد في جدول ما ويوجد في جدول آخر ذات علاقة كمفتاح رئيس، وتستخدم المفاتيح الأجنبية لتطبيق مفهوم الحقول المشتركة التي تسمح لنا بربط معلومات الجداول المختلفة.
- يتكون المفتاح المتصل (Concatenated Key)، سواء كان مفتاحاً أساسياً أم أجنبياً، من عدة حقول داخل جدول قاعدة البيانات.
- عندما يتم التعامل مع أكثر من جدول يجب الحفاظ على التكامل بين البيانات، فعلى سبيل المثال عندما تقوم بإضافة (حذف) عميل يجب إضافة المركب الذي يملكه (أو حذفه).
- عندما يتم التعامل مع عدة جداول يقوم الإجراء GetAll (أو الإجراء Find) بإعادة مجموعة من الكائنات التي يمكن من خلالها القيام باسترجاع بيانات الكائنات المرتبطة بها.
- لاسترجاع بيانات جدولين في نفس الوقت، نستخدم كلاً من المفتاح الأجنبي والمفتاح الرئيس لربط معلومات الجدول الأول بمعلومات الجدول الثاني والذي نطلق عليه ربط الجداول (Tables Joining) وذلك بواسطة أمر WHERE (أحد أوامر لغة SQL).
- يحتوي أمر Select على المقطع ORDER BY المسؤول عن ترتيب السجلات العائدة بواسطة أحد الحقول.
- عندما نتعامل مع عدة جداول علاقية مرتبطة ببعض بعضاً، يجب أن يتم تعديل إجراءات أصناف التعامل مع البيانات، وكذلك يجب إجراء تعديلات بسيطة على إجراءات أصناف مجال المشكلة. فعلى سبيل المثال يجب تعديل كل من إجراء الحذف والتعديل داخل أصناف التعامل مع البيانات للحفاظ على تكامل البيانات.
- يستخدم أسلوب "Control-Break" عندما تتغير قيمة متغير ما والذي يستخدم لتصنيف عناصر مرتبة، حيث يجب أن تكون البيانات مرتبة داخل كل مجموعة تنتمي لحقل ما والذي يسمى حقل التصنيف (Grouping Field) أو حقل التحكم (Control Field) الذي إذا تغيرت قيمته الحالية يتم الانتقال إلى مجموعة أخرى.
- تقدم لغة VB .NET إمكانية عرض البيانات بشكل شجري وذلك لوجود علاقة الربط parent-child، ويمكنك عمل ذلك من خلال إنشاء كائن من الصنف DataRelation الذي سيحتوي على الجدول Parent والجدول Child وحقول الربط بينهما.

المصطلحات الأساسية

Key Terms

الحقل الأجنبي (Foreign Key)	أمر Select
المفتاح الرئيس (Primary Key)	المقطع ORDER BY
ربط الجداول (Tables Joining)	أسلوب "Control-Break"
علاقة الربط Join	الجدول Parent
المفتاح المتصل (Concatenated Key)	الجدول Child

أسئلة المراجعة

Review Questions

١- باعتبار العلاقة بين الأصناف Person (شخص) و Pet (حيوان أليف) افترض أن كل حيوان أليف له بالضبط مالك واحد، ويمكن للشخص الواحد أن يمتلك العديد من الحيوانات الأليفة أو لا يمتلك شيئاً. اشرح كيف يمكنك تطبيق هذه العلاقات في الفيجوال بيسك .نت.

٢- افترض أن المعلومات المتبعة في تطبيق Person و Pet تحتوي على رقم تعريف الحيوان الأليف، واسم الحيوان الأليف، ونوع الحيوان الأليف (كلب، قطة، إلخ)، ورقم الضمان الاجتماعي للشخص، واسم الشخص، وعنوان الشخص، ورقم هاتف الشخص. كيف تنشئ الجداول في قاعدة البيانات العلاقية في هذا التطبيق؟ وما هي الأعمدة التي يجب إضافتها في جدول Pet؟ وما هي الأعمدة التي يجب إضافتها في جدول Person؟

٣- حدد الصف (أو الصفوف) في كل جدول الذي يخدم كمفتاح رئيس في التطبيق Person و Pet.

٤- حدد العمود (أو الأعمدة) في كل جدول الذي يخدم كمفتاح أجنبي في التطبيق Person و Pet.

٥- أي من إجراءات الوصول للبيانات تقترح انضمامها في الصنف Person؟ وأي منها تود انضمامها في الصنف Pet؟ ولماذا؟

٦- افترض أن قائمة جميع الملاك وحيواناتهم الأليفة مطلوبة. اذكر جملة SQL التي تستخلص هذه المعلومات من قاعدة البيانات.

٧- افترض أن الشخص الذي له رقم ضمان اجتماعي 111-11-1111 سوف يحصل على قطه جديدة تسمى Molly، ورقم تعريفها هو C123. اذكر جملة SQL لإضافة سجل جديد إلى جدول Pet. وافترض أن معلومات هذا المالك موجودة في جدول Person.

٨- اشرح الخطوات التي يجب اتخاذها لحماية تكامل قاعدة البيانات إضافة حيوان أليف Pet إلى قاعدة البيانات.

٩- اشرح الخطوات التي يجب اتخاذها لحماية تكامل قاعدة البيانات عند حذف مالك موجود من قاعدة البيانات.

١٠- هل تستطيع التفكير بالوقت متى يكون من الضروري إضافة سجل من جدول Pet أو حذفه وعدم حذف سجل من جدول Person أو إضافته؟ اشرح كيف تصف هذه الحالة.

أسئلة المناقشة
Discussion Questions

١- بالعودة للفصل التاسع، فإن العلاقة "واحد-إلى-واحد" تطبق في الاتجاهين. حدد كيفية تركيب الجداول داخل التطبيق BoatAndSlip الذي يوضح العلاقة بين المركب والمرسى (يشغل المركب مرسى واحداً، وينشغل المرسى بمركب واحد).

٢- تذكر أن علاقة الربط بين الصنف Customer والصنف Boat في البرنامج CustomerAndBoat تتطلب إضافة صفة تشير لكل واحد منهما في الصنف الآخر، وقد قمنا بإضافة الإجراء AssignBoatToCustomer داخل الصنف Boat لإنشاء هذه العلاقة في كلا الاتجاهين (ربط عميل بمركب وربط المركب بالعميل). لماذا لم نقم بتعريف هذا الإجراء داخل الصنف Customer؟ وما هي الفائدة من تعريف هذا الإجراء داخل كلا الصنفين؟

٣- لقد قمنا بتعريف صفة من الصنف Boat داخل الصنف Customer والعكس كذلك داخل البرنامج CustomerAndBoat لإنشاء علاقة ربط بين كل من العميل والمركب الذي يملكه، ولقد عرفنا الإجراء Find داخل الصنف CustomerDA للبحث عن العميل والمركب الذي يملكه. لماذا لم نقم بتعريف هذا الإجراء داخل الصنف BoatDA بدلاً من الصنف CustomerDA لإيجاد المركب الذي ملكه العميل؟ وما هي مميزات تعريف هذا الإجراء وعيوبه في أي من الصنفين؟

٤- تنفذ العلاقات داخل قواعد البيانات العلاقية باستخدام المفاتيح الأجنبية، بينما تنفذ داخل تطبيقات الكائنات الموجهة بواسطة متغيرات الإشارة. ناقش الفرق بين كلا الأسلوبين.

مشاريع الفصل
Projects

١- قم بفتح قاعدة البيانات DockAndSlipDatabase، ثم أضف عدة سجلات (بيانات مراسي) إلى الجدول SlipTable، ولكن اترك الحقل BoatID فارغاً لتشير أن هذه المراسي غير مشغولة بمراكب. قم بتعديل برنامج الاختبار والصنف Dock والصنف DockDA في التمرين لكي تسترجع معلومات حول المراسي غير المشغولة بمراكب (موقع المرسى، هل يوجد مياه أو كهرباء بالرصيف الذي يتبعه المرسى أم لا).

٢- قم بتعديل داخل البرنامج السابق الذي يؤدي تغيير سلوك الإجراء Find المعرف داخل الصنف Dock حيث لا تقوم بإرجاع بيانات حول الرصيف والمراسي فقط، بل حول المراكب التي تشغل هذه المراسي والعملاء الذين يملكون هؤلاء المراكب. ولكي تفعل ذلك، يجب عليك أن تضيف كلاً من الجدول BoatTable والجدول CustomerTable إلى قاعدة البيانات، قم بتعديل برنامج الاختبار لكي تستدعي الإجراء Find من Dock1 التي تقوم بإرجاع بيانات المراسي والمراكب والعملاء المناظرين، ثم من Dock2 وهكذا. لا نحتاج استدعاء الإجراء GetAll من برنامج الاختبار.

٣- لقد قمت بتطوير برنامج للمشروع رقم ٣ داخل الفصل التاسع لكي نتابع تسجيل خدمات المراكب. تذكر أنه ربما يُقدم للمركب خدمة أو أكثر ولكن كل خدمة تُقدم لمركب واحد. تتكون صفات الصنف BoatServiceRecord (خدمة المركب) من رقم الفاتورة، وتاريخ الخدمة، ونوع الخدمة، والمبلغ الكلي للخدمة. ويحتوي البرنامج على هذا الصنف بالإضافة إلى الصنف Boat والصنف Customer وكذلك إمكانية ربط العميل بالمركب الذي يملكه، وأخيراً الإجراء RecordBoatService المعرف داخل الصنف Boat لإنشاء خدمة جديدة (إنشاء كائن من الصنف BoatServiceRecord). قم بتحويل هذا البرنامج إلى برنامج قاعدة بيانات علاقية، ثم قم بكتابة برنامج اختبار لإضافة عدد من الخدمات داخل الجدول ServiceTable، وتأكد أن كل مركب مخصص له أكثر من خدمة. ويجب أن يسترجع برنامج الاختيار سجلات من جدول المراكب للمراكب التي لها خدمات وتقرير شامل عن كل مركب (يشمل المالك والخدمات).