الفصل العاشر

فهم كائنات

الوصول إلى البيانات

Access Data Objects

سنتعرف فى هذا الفصل على فكرة تعامل **Access** مع البيانات .

والتعرف على المحرك الحقيقى للتعامل مع البيانـــات **Database Engine** وهو المحرك الذى قامت **Microsoft** ببنائه لاستخدامه فى جميع البرامج التى تتعامل مع قواعد البيانات .

بانتهاء هذا الفصل ستتعرف على :

- العلاقة بين **Access** وقواعد البيانات **Database Engine** .
- تضمين كائنات الوصول الى البيانات .
- استخدام مستعرض الكائنات **Object Browser**.
- استخدام كائنات الوصول الى البيانات .
- انشاء هيكل قواعد البيانات .
- معالجة البيانات باستخدام كائنات الوصول الى البيانات .
- التنقل والبحث عن السجلات.
- انشاء وتنفيذ الاستعلامات .

العلاقة بين Access ومحرك قواعد البيانات Database Engine

أحب قبل أن نبدأ في هذا الفصل أن نوضح حقيقة هامة عن برنامج Access، وهي أن Access نفسه عندما نستخدمه لإنشاء أو تعديل قاعدة بيانات ليس هو الذي يقوم بذلك!! لا تتعجب فالأمر أن Access ما هو إلا برنامج "واجهة" يحجب المحرك الحقيقي لعملية الإنشاء والتعديل وهو مشغل قواعد البيانات Database Engine، استخدمت Access 2000 وكل من Access 2002/2003 المحرك Jet 40 ويعتبر هذا الاصدار هو آخر اصدار من المحرك Jet (كلمة Jet اختصار لعبارة Joint Engine Technology) وعندما قرر مطورو Access 2007 زيادة إمكانيات قواعد البيانات مثل استخدام أنواع جديدة لحقول البيانات وهي حقول Multi valued lookup (MVF) و Attachment ، أنتجوا إصداراً جديداً من Jet سمي Access database engine واحياناً يسمي Access Connectivity Engine وتختصر هكذا ACE.

واستدعي هذا التطوير أن يتم استبدال ملفات قواعد البيانات mdb بملفات جديدة ذات الامتداد accdb لتتمكن Access من التعامل مع الامكانيات الجديدة .

للتعرف علي أنواع الحقول الجديدة MVF و Attachment راجع كتابنا "المرجع الأساسي لمستخدمي Access 2007"

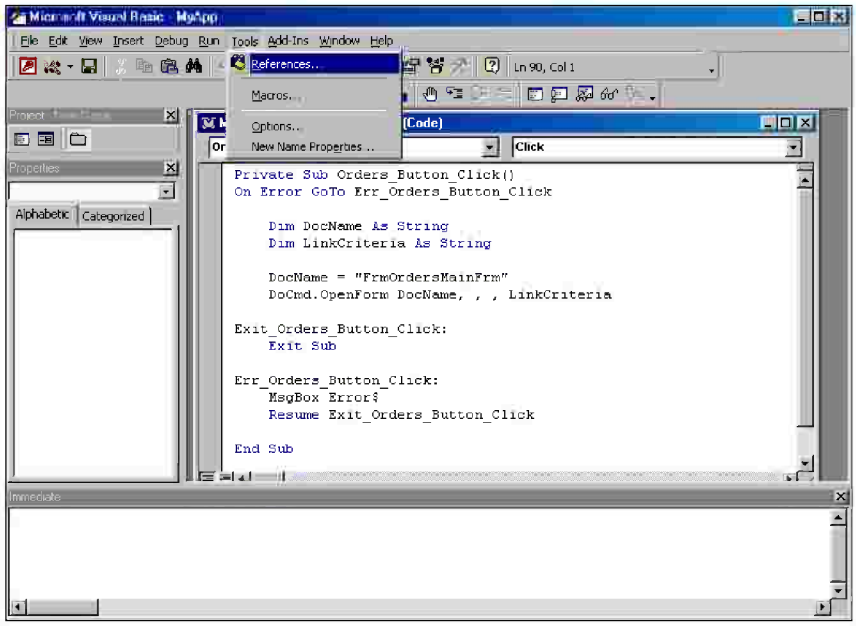
تستطيع Access 2007 فتح وتعديل وحفظ الملفات المنشأة بتنسيق الاصدارات السابقة مثل Access 2000/2002/ 2003.

عند استخدام هذا الملف في تطبيق Access يصبح لديك مجموعة من الكائنات Objects يمكنك من خلالها الاستفادة من كافة إمكانات هذا المشغل، هذه الكائنات يطلق عليها "كائنات الوصول إلى البيانات" Data Access Objects (DAO). هذه

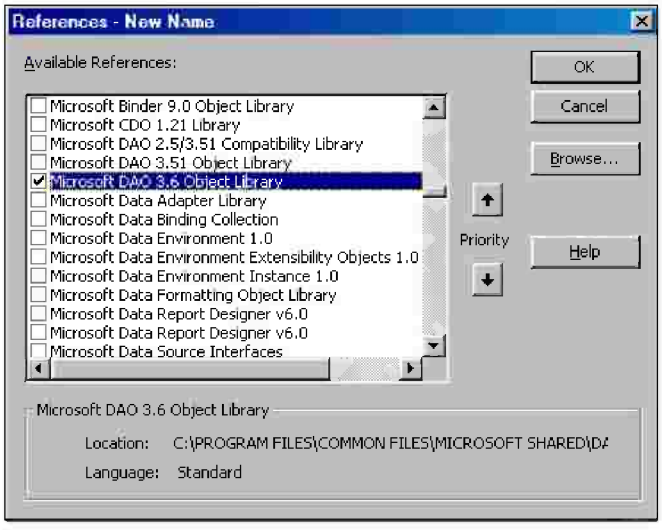
الكائنات مرتبة في هيكل محكم، يمكنك من خلاله برمجة قواعد البيانات "نقصد هنا المشغل طبعا" بطريقة سهلة ومباشرة. ولكن لماذا نفعل ذلك؟ أي لماذا نقوم بكتابة كود بلغة Access VBA لبرمجة كائنات الوصول، وعندنا النماذج، ووحدات الماكرو؟ السبب هو تحقيق أقصى سرعة والاستفادة من جميع إمكانيات هذه الطريقة. وبذلك نتجاوز Access كوسيط للتعامل مع المشغل وهذا يوفر الكثير من الوقت.

تضمين كائنات الوصول الى البيانات

لكي تقوم باستخدام كائنات الوصول إلى البيانات في برامج Access يجب أن تضمها للمراجع التى سيرجع إليها VBA 6.0. هذا الأمر من المفترض أن Access قد قام به تلقائيا عند التنصيب، ولكننا هنا سنفعله لنتأكد ولنعدله إن كان تم تغييره لأي سبب. افتح أي قاعدة بيانات في Access ثم قم بفتح نافذة محرر VBA. اختر أمر Referencies من قائمة Tools كما فى شكل ١-١٠، يتم فتح المربع الحوارى Referencies الموضح في شكل ٢-١٠. قم بالتأكد أن الخيار Microsoft Data Access Object 3.6 Library مختارا (نشطاً). فإن لم يكن مختاراً انقره لاختياره .



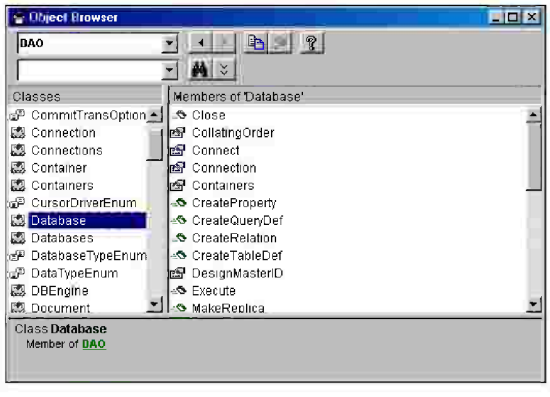
شكل ١-١٠ اختيار قائمة (أدوات والامر مراجع منها) **Tools,References**



شكل ٢-١٠ مربع حوار "مراجع" المستخدم لضم كائنات معينة للمراجع الذي سيرجع إليها **VBA**.

استخدام مستعرض الكائنات Object Browser

نظرا لكون كائنات الوصول الى البيانات DAOs كائنات معقدة وكثيرة الوظائف والخصائص، فمن المفيد أن تتعلم من البداية الاستفادة من إمكانيات مستعرض الكائنات Object Browser والذى شرحناه فى فصلٍ سابق، إلا أننا نريد إلقاء الضوء على كيفية اسخدامه فى الستعراض كائنات DAOs. لكي تفتح هذا المستعرض، يجب أن تكون أولا في طور البرمجة بأن تفتح نافذة محرر VBA. وأثناء وقوفك في نافذة المحرر ، اضغط F2، أو اختر امر Object Browser من قائمة View أو انقر رمز المستعرض في شريط الأدوات. سيظهر لك المستعرض كما في شكل ٣-١٠.



شكل ٣-١٠ نافذة مستعرض الكائنات

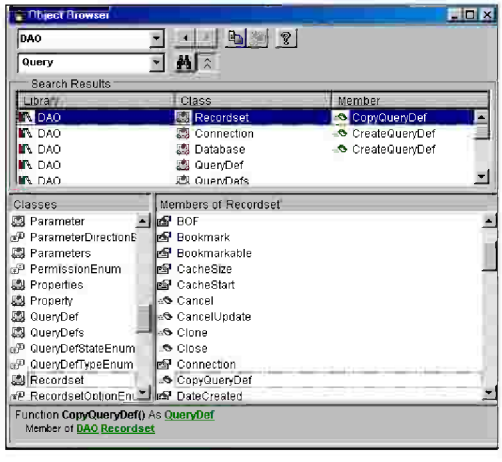
لكي ترى عناصر DAOs ، اختر DAO من المربع المنسدل الأول الموجود يسار النافذة . كما في شكل ٣-١٠. إذا اخترت Database مثلا ، فسترى كافة الوظائف Methods والخصائص Properties الخاصة بهذا الكائن. من خلال هذا المستعرض تتأكد من اسم الوظيفة وكذلك الصيغة التي تستخدم فيها (والتي تعرض أسفل النافذة كما في شكل ٤-١٠) .

يتيح أيضا مستعرض الكائنات خاصيتين هامتين وهما:

- فتح نافذة تعليمات المساعدة الخاصة بوظيفة أو خاصية معينة، وذلك بضغط الزر

المرسوم عليه رمز(?) في أعلى يمين شاشة المستعرض. وبذلك تستطيع أن تقرأ المزيد من المعلومات عن خاصية معينة.

- البحث عن الخصائص أو الوظائف التي يحتوي اسمها كلمات معينة، مثلا لو كنت مهتما بالوظائف التي تتعامل مع كلمة " Query"، أو تبحث عن وظيفة بها هذه الكلمة، اكتب Query في المربع المنسدل الثاني الموجود في أعلى نافذة المستعرض ، ثم اضغط على الزر المرسوم عليه النظارة المكبرة ، سترى نتيجة البحث أسفل المربع، وعند وقوفك على أحد عناصرها سيبدو لك مكانها من خلال المستعرض في مكانها المعتاد من النافذة. كما في شكل ٤-١٠.



شكل ٤-١٠ نافذة مستعرض الكائنات عند استخدامها في البحث.

استخدام كائنات الوصول إلى البيانات

أول الخطوات في استخدام كائنات الوصول هو تحديد قاعدة البيانات التي سنقصدها في أية تعليمات. لو كنا ننشيء وحدة نمطية عامة قابلة للعمل على أكثر من قاعدة بيانات، فلابد أن نحدد بدايةً أننا نعمل على قاعدة البيانات الحالية المفتوحة في Access. وذلك بالعبارة الآتية:

```
Set dbActive = CurrentDB()
```

الدالة CurrentDB() ترجع قاعدة البيانات الحالية، أي المفتوحة داخل Access. وقاعدة البيانات هذه مكافئة للكائن

BEngine.Worksapces(0).Databases(0)

بعد ذلك يمكننا الإشارة إلى أي جدول أو استعلام داخل قاعدة البيانات هذه باستخدام الصور المختصرة:

dbActive.QueryDefs(0).Name, dbActive.TableDefs.Count

أي أن dbActive تنوب عن الاسم الطويل لقاعدة البيانات الحالية.

الكائن الذي أعلنا عنه (قاعدة البيانات) يحتوي بداخله ، كما يرينا هيكل DAO في شكل ١٠-١ على مجموعات من (TableDefs وQueryDefs و RecordSet وRelation) بجانب عناصر أخرى. وعلى الرغم من وضوح معنى هذه الكائنات إلا أننا سنشير إليها باختصار كما يلى:

- الكائن TableDef : معناه تعريف جدول ، وهو يقوم بتعريف الجداول المحفوظة ويحتوي داخليا على المعلومات الخاصة بالحقول (مجموعة Fields) والمعلومات الخاصة بالفهارس (Indexes). من أمثلة الخصائص المحددة لكل جدول: اسمه Name وعدد السجلات فيه RecordCount.
- الكائن QueryDef : معناه تعريف الاستعلام ويحتوي على تعريف استعلام محفوظ، كما يحتوى على أسماء الأعمدة (المماثلة للحقول)، وعبارة SQL المعبرة عن الاستعلام، والمعاملات Parameters إن كانت هناك معاملات للاستعلام.
- الكائن RecordSet : وهو عبارة عن صورة من الجداول (المحفوظة في القرص الصلب) موجودة في الذاكرة، وقد يكون كذلك صورة من نتيجة استعلام.
- الكائن Relation : ويحتوي على العلاقات بين الجداول.

إنشاء هيكل قواعد البيانات

لكي لا نطيل في الكلام النظري، دعنا نبدأ هنا بمثال يوضح استخدام DAOs مباشرة، أي ماذا نفعل بهذه الكائنات. المثال التالي يوضح ميزة هامة جدا وهي إمكانية التعامل مع أكثر من قاعدة بيانات في نفس الوقت، رغم أننا لا نؤدي عمليات في المثال التالي، إلا أننا نقوم باستخدام ثلاثة قواعد بيانات، واحدة هي المفتوحة من قِبَل Access عند كتابة الإجراء، الثانية هي قاعدة بيانات جديدة يتم إنشائها و اسمها TestDAO، والثالثة قاعدة بيانات موجودة أساساً ويتم فتحها. كما ترى فإن ذلك يمكنك من التعامل مع البيانات ونقلها بسهولة تامة.

```
Sub DAOExample()

   Dim dbExisting As Database
   Dim dbNew As Database
   Dim dbOther As Database

   Set dbExisting = CurrentDb()
   Set dbNew = Workspaces(0).CreateDatabase("TestDAO.mdb",
dbLangGeneral)
   Set dbOther = Workspaces(0).OpenDatabase("Software.mdb")

   dbExisting.Close
   dbNew.Close
   dbOther.Close

End Sub
```

قم بكتابة هذا الكود في نافذة الوحدة النمطية Module ثم اضغط زر التشغيل، لن يحدث شيء محسوس، ولكن إذا نظرت إلى الفهرس الحالي (الذي تم فتح قاعدة البيانات الحالية من عليه) سترى الملف TestDAO.mdb قد أضيف إليه، بالطبع لا يحتوي هذا الملف على أية محتويات، ولكن المراد هنا أن نعلم الطرق الثلاثة لاستخدام قاعدة بيانات : إنشائها، فتحها، استخدام قاعدة البيانات الحالية. أيضا تلاحظ أننا استخدمنا متغير لكل قاعدة

بيانات من النمط Database، كذلك لابد من إغلاق قواعد البيانات باستخدام الوظيفة Close.

دعنا الآن نخطو خطوة أخرى ، وهي أن ننشيء جدول ونحدد حقوله في قاعدة البيانات التي أنشأناها. المثال التالي يوضح ذلك، ننشيء فيه جدول نسميه Customers ثم نضيف إليه الحقول Name و Phone. لو قمت بتشغيل هذا الإجراء بعد كتابته ثم قمت بفتح قاعدة البيانات TestDAO ستجد داخلها هذا الجدول وحقوله.

```
Sub DAOExample2()
   Dim dbNew As Database
   Dim tbNew As TableDef

   Set dbNew = OpenDatabase("TestDAO.mdb")
   Set tbNew = dbNew.CreateTableDef("Customers")
   tbNew.Fields.Append tbNew.CreateField("Name", dbText)
   tbNew.Fields.Append tbNew.CreateField("Phone", dbText)
   dbNew.TableDefs.Append tbNew

   dbNew.Close
End Sub
```

وهذا المثال كما ترى يحتاج إلى وقفة قصيرة لنفهم ماذا يفعل:

- بداية فتحنا قاعدة البيانات كما تعلمنا باستخدام OpenDatabase.
- قمنا بإنشاء "تعريف جدول" باسم Customers باستخدام الدالةCreateTableDef. لاحظ أن تعريف الجدول ليس هو الجدول، و إنما المواصفات التي تحددها لإنشاؤه كالتي تدخلها في نافذة التصميم في Access، أي أننا حتى هذه اللحظة لم ننشيء جدولا في قاعدة البيانات، وإنما قمنا بتعريف الكائن داخل الذاكرة.
- بعد ذلك أضفنا حقول إلى هذا التعريف من خلال استخدام الدالة CreateField لإنشاء "تعريف حقل"، وبدلا من وضعه في كائن، نقوم مباشرة بإضافة هذا التعريف إلى تعريف الجدول TableDef الذي سبق إنشاؤه.

- الخطوة الأخيرة هي الإضافة الفعلية للجدول في قاعدة البيانات باستخدام dbNew.Append ثم إغلاق قاعدة البيانات.

والآن لنفرض أننا نريد أن نضيف حقل آخر إلى هذا الجدول بعد إنشائه. بداية نريد أن نوضح كيفية الإشارة إلى هذا الجدول في قاعدة البيانات. هناك طريقتان للإشارة إلى هذا الجدول:

- الأولى تعتمد على ترتيب هذا الجدول، بما أنه أول الجداول فإن دليله في مجموعة TableDefs هو 0 ، وعليه يمكننا الإشارة إليه بـ TableDefs(0). وهذه الطريقة تصبح غير مناسبة إن كنا نفتح قاعدة بيانات كبيرة بها الكثير من الجداول ولا نعرف ترتيب الجداول، لذا يصبح من المناسب الرجوع إلى الجدول باسمه، كالتالي TableDefs("Customers").
- الطريقة الثانية هي شكل مختصر للطريقة الأولى، وتتمشى مع صيغة SQL وهو أن نكتب اسم الجدول كالتالي TableDefs!Customers أي بوضع ! بين اسم المجموعة واسم الجدول، ولكن لو احتوى اسم الجدول على مسافة أو أحرف خاصة، يجب إحتوائه بين قوسين مربعين كالتالي: TableDefs![Customers].

بعد هذه الملاحظة نعرض فيما يلي الإجراء اللازم لإضافة حقل آخر إلى الجدول:

```
Sub DAOExample3()
   Dim dbNew As Database
   Dim tbNew As TableDef

   Set dbNew = OpenDatabase("TestDAO.mdb")
   Set tbNew = dbNew.TableDefs("Customers")
   tbNew.Fields.Append tbNew.CreateField("IDnum", dbLong)
   dbNew.Close
End Sub
```

يقوم هذا الإجراء بإضافة حقل من النمط Long إلى الجدول الذي سبق إنشاؤه، لاحظ هنا استخدامنا الطريقة الأولى للإشارة إلى الجدول وهي TableDefs("Customers"). من الأمثلة السابقة تستطيع أن تستنتج كيفية إضافة وتعديل وكذلك حذف الجداول

والحقول (الوظيفة Delete في كلٍ منها). ولكنك هنا قد تتساءل، لماذا أقوم بإنشاء وتعديل هيكل قاعدة البيانات باستخدام الكود إذا كانت بيئة Access تتيح ذلك بمنتهى السهولة؟ الإجابة أنه عند إنشاءك لتطبيق متخصص لشركة على سبيل المثال، يكون هذا التطبيق ملبيا لكافة متطلبات الشركة دون الحاجة إلى استخدام Access لأن الموظفين قد لا يكونون على دراية به، فلو كان العميل يحتاج مثلا إلى أن ينشيء جداول فرعية لبعض العملاء (ينشئها عندما يكون التطبيق عنده وليس عند إنشاء التطبيق) فيجب أن يعطيه برنامجك هذه الإمكانية ولهذا نحتاج إلى استخدام DAOs.

تعديل هيكل قواعد البيانات يمكن أن يتم، بالإضافة إلى DAO، من خلال SQL، حيث أن شطر هذه اللغة يتعامل مع هيكل قواعد البيانات ويسمى Data Definition Language DDL.

معالجة البيانات باستخدام كائنات الوصول إلى البيانات

تعرفنا في الفقرة السابقة على كيفية إنشاء قواعد البيانات وإضافة الجداول والحقول إليها. نتعرف فيما يلي على كيفية إضافة البيانات نفسها، أي إضافة سجلات إلى الجداول وحذفها، والبحث فيها.

المثال التالي يقوم بإضافة سجل جديد إلى الجدول Customers الذي أنشأناه آنفا، ويقوم بتحديد قيم الحقول فيه:

```
Sub DAOExample4()
   Dim dbNew As Database
   Dim rsDyn As Recordset

   Set dbNew = OpenDatabase("TestDAO.mdb")
   Set rsDyn = dbNew.OpenRecordset("Customers",
    dbOpenDynaset)

   rsDyn.AddNew
   rsDyn!Name = "Ahmed"
   rsDyn!Phone = "2825100"
```

```
    rsDyn!IDNum = 1
    rsDyn.Update

    rsDyn.Close
    dbNew.Close
End Sub
```

لنتأمل هذا الإجراء خطوة بخطوة. موضوع فتح قاعدة البيانات شرحناه سابقا، المهم هنا أن نتذكر ضرورة إغلاق قاعدة البيانات باستخدام **Close** بعد استخدامها.

- في هذا المثال قمنا بفتح الجدول **Customers** للتعامل معه. في المثال السابق **Example2**. عندما أردنا التعديل في هيكل الجدول، استخدمنا الكائن **TableDef** وهو يحمل خصائص الجدول. أما هنا فإننا نريد التعامل مع بيانات الجدول، لذا نستخدم الكائن **RecordSet**، هذا الكائن هو صورة من الجدول (المحفوظ على القرص الصلب) موجودة في الذاكرة. لأننا أردنا هنا إضافة سجل استخدمنا **Recordset** من النمط **Dynaset** وهو نمط يتيح لنا التعديل في البيانات (هناك أيضاً النمط **Table** الموجود بالذاكرة ولكنه محدود الخصائص، ونمط **Snapshot** وهو سريع ولكن غير قابل للتعديل).
- بمجرد الانتهاء من الإضافة قمنا بإغلاق الـ **RecordSet** بالوظيفة **Close**، هذا الأمر هام جدا، لأن تعديل البيانات في **RecordSet** يكون في الذاكرة فقط، ولا ينعكس على البيانات المحفوظة في القرص الصلب حتى يتم استخدام الوظيفة **Update** ثم الإغلاق بالوظيفة **Close**.
- الجزء الهام في الإجراء السابق هو عملية إضافة سجل جديد، هذه العملية تتم على ثلاثة مراحل:

١. استخدام الوظيفة **AddNew**.

٢. تحديد قيم الحقول مباشرة مثل **rsDyn!Phone ="2825100"** لوضع الرقم **2825100** فى حقل **Phone**.

٣. التأكد من انعكاس التعديلات التى قمنا بها على قاعدة البيانات باستخدام الوظيفة Update.

المثال السابق رغم بساطته إلا أنه يوضح كيفية إضافة سجل إلى أي جدول، عبر المرور بالكائن RecordSet كوسيط في الذاكرة يتم نقل الجدول إليه ثم التعديل والكتابة عليه ثم نقله ثانية إلى قاعدة البيانات. إن فهمك لهذه الآلية جيدا يضمن لك مرونة كبيرة في التعامل مع قواعد البيانات من خلال الكود. يمكنك الآن إنشاء نموذجك الخاص ، وأن تضع عليه زر "سجل جديد"، وبدلا من الاعتماد على Access في الإضافة يمكنك أن تجعل هذا الزر يستدعي إجراء شبيه بالإجراء Example3 السابق. وفي معظم الأحوال لا يتم كتابة قيم الحقول مباشرة كما فعلنا في هذا المثال وإنما يتم استخلاصها من مربعات نصوص موجودة على النموذج.

حسنا، بعد إضافة السجلات ربما أردت التعديل أو الحذف. يتم التعديل بنفس الكيفية ولكن مع استخدام الوظيفة Edit بدلا من AddNew. أما الحذف فيتم مباشرة باستخدام Delete ولكن قبل استخدام التعديل أو الحذف يجب أن يكون السجل الحالي CurrentRecord هو الذي تريد تعديله أو حذفه. يمكنك القيام بهذا من خلال أوامر التنقل عبر السجلات، أو البحث عن سجل معين وهو موضوع الفقرة التالية.

التنقل والبحث عن السجلات

عملية التنقل عبر السجلات عملية هامة جدا، فهي ضرورية قبل كل عملية تعديل أو حذف للسجلات. هناك أربعة وظائف أساسية في أي RecordSet تستخدم في عملية التحرك هذه وهي :

١. MoveFirst للتحرك إلى أول سجل في ال RecordSet، فمثلا لو أردنا التحرك إلى أول سجل في الجدول Customers بعد فتحه باستخدام الكائن rsDyn فإننا نستخدم rsDyn.MoveFirst.

٢. MoveLast للتحرك إلى آخر سجل في الـ RecordSet.

٣. **MoveNext** للتحرك إلى السجل التالي. نظرا لأن هناك سجل واحد يطلق عليه السجل الحالي **CurrentRecord** هو الذي يقبل عمليات الحذف والتعديل ، فلو كان السجل الحالي هو رقم ١، فإن **MoveNext** تجعل السجل الحالي هو السجل رقم ٢.

٤. **MovePrevious** للتحرك إلى السجل السابق، وهو عكس **MoveNext**.

بالإضافة إلى عملية التحرك المتسلسل هذه أو التحرك إلى أطراف الـ **RecordSet**، هناك وظيفة أخرى وهي **Move** تقوم بالتحرك النسبي، أي بالنسبة إلى السجل الحالي إلى الأمام عدد محدد من السجلات، فمثلا العبارة:

rsDyn.Move 5

تقوم بتحريك المؤشر (مؤشر السجل الحالي) ٥ سجلات (أو صفوف **rows**) إلى الأمام. وهذا أسرع من استخدام **MoveNext** خمس مرات.

عمليات التحرك هي الوظائف التي تنفذها الأزرار التي عليها الأسهم في معظم نوافذ برنامج **Access** مثل الأسهم التي تراها أسفل نافذة أي جدول في حالة عرضه.

الوظائف السابقة مفيدة عند عملية معالجة السجلات في صورة متتالية تبدأ من أحد أطراف الجدول وتمضي سجل بسجل حتى الطرف الآخر، أو تبدأ وتنتهي عند نقط محددة أيا كانت. ولكن في حالة الرغبة في تطبيق التعديل أو الحذف على سجلات معينة بناءً على توافر شروط معينة فيها ،فإننا نستخدم مجموعة أخرى من الوظائف وهي وظائف البحث وتتكون من أربعة وظائف، هي:

١. **FindFirst** : وهي للبحث عن أول سجل من **RecordSet** يحقق شرط معين، وصيغتها كالتالي:

recordset.FindFirst criteria

حيث **criteria** هي الشرط المراد تحققه، وهو عبارة عن سلسلة حرفية **String** تحتوي على عبارة شرطية متوافقة مع **SQL** تحتوي على أسماء الحقول المراد البحث فيها ، والقيم المراد تحققها، وكذلك العلاقة بينهما. مثلا لو أردنا البحث في الـ

RecordSet السابق rsDyn عن أول عميل الذي يبدأ اسمه بـ "Mohamed"، فإننا نستخدم العبارة التالية:

```
rsDyn.MoveFirst
rsDyn.FindFirst "Name > 'Mohamed' "
```

لاحظ هنا أننا استخدمنا الأقواس المزدوجة لاحتواء العبارة ككل، والأقواس المفردة لأي نص جزئي داخلها ، مثل Mohamed كما تنص لغة SQL. أيضا استخدمنا علاقة أكبر من ، التي تعني عدم اشتراط كون الاسم "Mohamed" وفقط، وإنما أي اسم يبدأ به. الأمر الهام أيضا هو استخدام MoveFirst للتأكد من أننا نبحث من بداية ال recordSet .

٢. الوظيفة الثانية هي FindNext ، وتستخدم بعد إيجاد أول سجل لإكمال البحث عن السجل التالي.

٣. FindPrevious و هي عكس FindNext حيث تبحث عن السجل السابق الذي يحقق الشرط.

٤. FindLast يبحث من أخر ال RecordSet عن أول سجل يحقق الشرط. وجميع الوظائف Find لها نفس الصيغة التي وضحناها للوظيفة FindFirst.

المثال التالي يستخدم الوظيفة FindFirst للبحث عن سجل، ويعقبها استخدام الوظيفة Edit للتعديل فيه.

```
Sub DAOExample5()
   Dim dbNew As Database
   Dim rsDyn As Recordset

   Set dbNew = OpenDatabase("TestDAO.mdb")
   Set rsDyn = dbNew.OpenRecordset("Customers",
dbOpenDynaset)

   rsDyn.FindFirst " Phone='2825100' "
   rsDyn.Edit
   rsDyn!Phone = "2525100"
   rsDyn.Update
```

```
  rsDyn.Close
  dbNew.Close
End Sub
```

إنشاء وتنفيذ الاستعلامات

يحتوي الكائن Database على وظيفة تتيح تنفيذ استعلامات إجرائية action query مباشرة وهي Execute، حيث تقبل عبارة SQL كسلسلة حرفية string، كالتالي:

```
Database.Execute " Update Customers Set IDNum=IDNum+100"
```

كما ترى يقوم هذا الاستعلام بتعديل الحقل IDNum بإضافة 100 إلى هذا الحقل في كل السجلات من الجدول Customers.

بالطبع هذا الاستعلام لا يتم حفظه في قاعدة البيانات، ولا يوجد إلا في البرنامج.فإذا أردت حفظ استعلام بصورة دائمة في قاعدة البيانات ، قم بتعديل هيكل قاعدة البيانات، وهذا يستلزم استعمال الكائن QueryDef مثلما استعملنا الكائن TableDef من قبل. المثال التالي يقوم بإنشاء الاستعلام السابق كاستعلام دائم و يحفظه في قاعدة البيانات:

```
Sub DAOExample6()
  Dim dbNew As Database
  Dim qdNew As QueryDef

  Set dbNew = OpenDatabase("TestDAO.mdb")
  Set qdNew = dbNew.CreateQueryDef("IncNum", "Update
    Customers Set IDNum=IDNum+100")

  dbNew.Close
End Sub
```

المثال السابق لا يقوم بتنفيذ الاستعلام وإنما يقوم فقط بإنشاؤه. فإذا أردت تنفيذ هذا الاستعلام، قم باستخدام الوظيفة Execute الخاصة بالكائن queryDef كما في المثال التالي:

```
Sub DAOExample7()
  Dim dbNew As Database
  Dim qdNew As QueryDef

  Set dbNew = OpenDatabase("TestDAO.mdb")
```

```
    Set qdNew = dbNew.QueryDefs("IncNum")
    qdNew.Execute

    dbNew.Close
End Sub
```

في نهاية هذا الفصل نقول أننا لم نقصد لأن يكون هذا الفصل مرجعا كـــاملاً لاســـتخراج لـــ DAOs فهي مجموعة كبيرة من الكائنات (١٧ نوع من الكائنات) التي تحتاج كتـــاب كامل لشرحها ولكن قصدنا أن نضع قدمك على طريق استخدامها حيث لا تحتاج بعـــد ذلك إلا استخدام ملفات المساعدة و مستعرض الكائنات، بعد فهمك لأســـلوب عملـــها والفكرة التي قامت عليها.