



ADOとVisual Basicを使用した Adaptive Server Anywhereの データアクセス

はじめに

開発者間では、データ・アクセス目的でActiveX Data Objects (ADO)を使用し、ADO とVisual Basicを使用してOLE DB プロバイダ経由でデータベースのデータにアクセスすることが多くなっています。このホワイトペーパーでは、Adaptive Server Anywhere およびiAnywhere Solutions のOLD DBプロバイダと共にADO の使用に関して役立つ情報を提供するとともに、クライアント・アプリケーションでADO を使用する場合の一般的なサンプル・コードやバグ・フィックスに関する情報、iAnywhere Solutions のOLE DB プロバイダであるASAProv がサポートする機能に関して説明します。

このホワイトペーパーは、次のURLにあるMSDNライブラリと共に使用してください。

<http://msdn.microsoft.com/library/default.asp?url=/library/enus/ado270/htm/mdmscadoprogrammersguide.asp/>

また、ADOやVisual Basicのプログラミングについては説明していないので、ここで取り上げるトピックの詳細については、<http://msdn.microsoft.com>を参照してください。

ここに書かれている内容はAdaptive Server Anywhere 7.0.2 ビルド番号1505 に基づいています(7.0.2 の販売は終了しています)。本書作成中にADO の使用方法改善のための修正がいくつかAdaptive Server Anywhere に対して行われました。従って、問題を回避するために、サポートにご契約いただき、最新のEBF を適用されることをお奨めします。



このホワイトペーパーで使用するサンプル・プロジェクトでは、ストアド・プロシージャを次のように呼び出します。

```
comm.CommandText = "{call foo_in(?,?)}"  
comm.CommandType = adCmdText
```

これは、ASAProv が ADO 標準フォーマットのパラメータやストアド・プロシージャの処理方法に問題があったためです。

これらの問題は最近の EBF で修正されたため、ADO 標準フォーマットを使用してストアド・プロシージャを次のように呼び出すことができます。

```
comm.CommandText = "foo"  
comm.CommandType = adCmdStoredProc
```

Adaptive Server Anywhere バージョン 7.0.2 を使用している場合、この問題はビルド 1559 で修正されます。バージョン 7.0.3 の場合は、ビルド 2096 で修正されます。また、バージョン 8.0 の場合は、ビルド 2196 で修正されます。最新の EBF にアップグレードする方法については、3 ページの「Adaptive Server Anywhere の設定」を参照してください。

このホワイトペーパーは Adaptive Server Anywhere バージョン 7.0.2 に基づいていますが、ここに示すプロジェクトはすべて、バージョン 8 を含む新しいバージョンの Adaptive Server Anywhere でも使用できます。Adaptive Server Anywhere バージョン 8 を使用している場合は、ODBC データ・ソースを作成するときや Adaptive Server Anywhere のバージョンを確認するときに、開始行、ディレクトリ、ファイル名の 7 を 8 に置き換えてください。

Adaptive Server Anywhere バージョン 8 を使用している場合、バージョン 8.0 のカーソル・サポートには、ブックマーク・プロジェクトに影響する動作変更があります。LockType adLockOptimistic、adLockPessimistic、adLockBatchOptimistic のいずれかを使用して静的カーソルを開くと、このエンジンによってカーソル・タイプが動的カーソルに変わります。動的カーソルはブックマークをサポートしないため、ブックマークをサポートする静的カーソルを取得するには、LockType adLockReadOnly を使用して静的カーソルを開く必要があります。ブックマークでの更新も可能にする場合は、キーセット駆動型カーソルも使用できます。

参考のため、このホワイトペーパーにはコード・サンプルを掲載しています。プロジェクトが機能することを確認するために、コード・サンプルをコピー・アンド・ペーストするのではなく、実際のプロジェクトを使用することをお勧めします。プロジェクトとこのホワイトペーパーの PDF は、Sybase の Web サイトから zip ファイルで入手できます。

目次

概要.....	表紙
目次.....	4
ActiveX Data Objects.....	6
入門.....	7
Adaptive Server Anywhere の設定	7
ODBC データ・ソースの作成.....	8
Visual Basic: ADO ライブラリの参照.....	11
プロバイダ.....	12
ADO オブジェクト.....	13
Connection オブジェクト.....	14
Connection オブジェクトの宣言.....	14
Connection Open メソッドの使用.....	14
カーソル・ロケーション	15
接続の切断.....	16
Recordset オブジェクト.....	16
ソース.....	17
カーソル・タイプ.....	17
ロック・タイプ.....	19
Recordset Open メソッドの使用.....	20
レコードセットのデータ更新.....	21
さまざまなコントロールを使用したデータ表示.....	23
DataGrid コントロール.....	24
MSHFlexGrid コントロール.....	27
DataList コントロールと ComboBox コントロール.....	28
ADO Data コントロール.....	29
レコードセットとレコードセットのページのカウンタ.....	34
レコードセット内の移動.....	36
ブックマークの使用.....	36
複数のレコードセットの取り出し.....	39
ストアド・プロシージャ.....	42
レコードセットを使用した切断.....	44
Command オブジェクト	45
Name プロパティの使用.....	47
パラメータ	48
Blob.....	54

Property オブジェクト.....	59
Error オブジェクト.....	60
ASAProv でサポートされているエラー・プロパティ.....	60
On Error 文を使用した ADO エラーのトラップ.....	60
プロバイダ・エラーへの Count プロパティの使用.....	62
Clear メソッドの使用.....	63
ユーザ定義エラーを ADO エラーと共に使用.....	65
付録 A: プロジェクト.....	67
付録 B: Adaptive Server Anywhere への接続.....	68
付録 C: ASAProv プロバイダの使用.....	69
法的注意.....	70

ActiveX Data Objects

ADO は、さまざまなタイプのデータ・ソースのアクセス、編集、更新に使用する API (Application Program Interface) です。ADO が開発される前は、ODBC (Open Database Connectivity) が、リレーショナル・データベースとの間のデータ・アクセスに使用されていました。ODBC は非リレーショナル・データベースやその他のデータ・ソースにアクセスできないという問題を解決するために、Microsoft が ADO を作成しました。アプリケーションで ADO を使用すると、リレーショナル・データベースや非リレーショナル・データベース、テキスト・ファイルや XML ファイル、電子メール、グラフィックなど、どんなデータ・ソースのデータにもアクセスできます。ActiveX Data Objects は、COM (Common Object Model) オブジェクトのセットであり、OLE DB と ODBC を使用してこれらのデータ・ソースにアクセスします。

ADO、OLE DB、ODBC はすべて、Microsoft データ・アクセス・コンポーネント (MDAC) の構成要素です。アプリケーションは、上位のインタフェースである ADO を使用して、下位の API である OLE DB を呼び出し、次に OLE DB がデータベースまたはデータ・ソースを呼び出します。図 1 は、ADO とデータ・ソースとの対話方法を示しています。

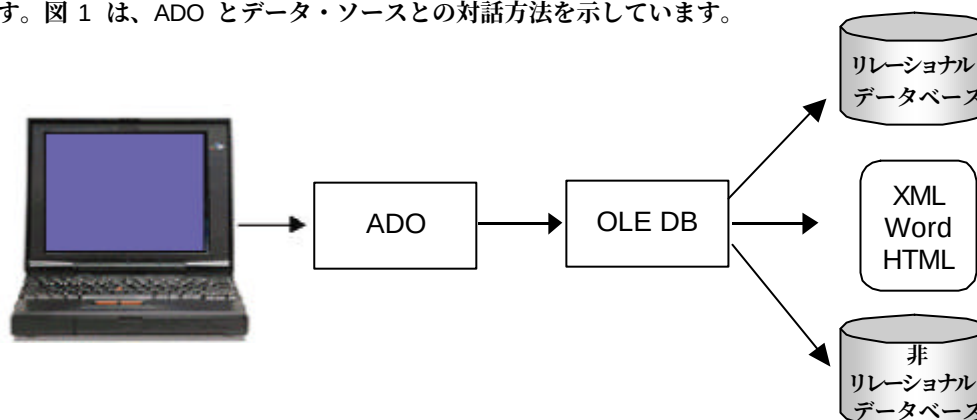


図 1：ADO とデータ・ソースとの対話方法

入門

この項では、ODBC データ・ソースの設定方法、ADO ライブラリの参照方法、Adaptive Server Anywhere の設定方法のほか、プロバイダや ADO オブジェクトについても説明します。

Adaptive Server Anywhere の設定

ASA が、最新のビルドまたは EBF (Emergency Bug Fix) で適切にインストールされていることを確認します。

使用しているバージョンを確認するには、次の手順に従います。

- 1 win32 ディレクトリに移動します。デフォルトでは、このディレクトリは次の場所にあります。
c:\Program Files\Sybase\SQL Anywhere 7\win32
- 2 dboledb7.dll を右クリックして、ポップアップ・メニューから [プロパティ] を選択します。
- 3 図 2 に示すように、[バージョン情報] タブで、[ファイル バージョン] が 7.0.2 1505 (またはこれより新しいファイル・バージョン) であることを確認します。

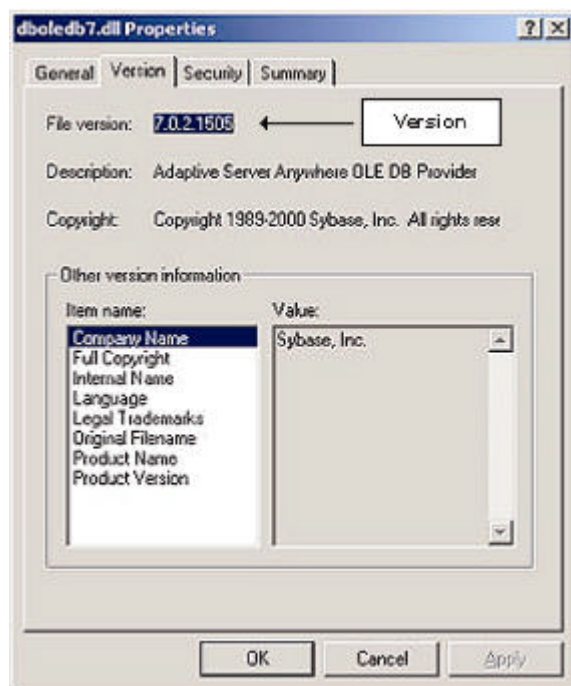


図 2：ファイル・バージョンを示す [バージョン情報] タブ

このホワイトペーパーは、Adaptive Server Anywhere 7.0.2 ビルド 1505 に基づいています。このホワイトペーパーの執筆中に、ADO の使用方法を改良するためのいくつかの修正が Adaptive Server Anywhere に対して行われました。したがって、問題を回避するために、必ず最新の EBF を使用してください。

最新の EBF をダウンロードするには、次の手順に従います。

- 1 Web ブラウザで次の URL を開きます。
www.sybase.com
- 2 左のウィンドウ枠で、[Support] を選択します。
- 3 [Support Services] リストから [EBFs/Updates] を選択します。Sybase Web アカウントをすでにお持ちの場合は、ユーザ名とパスワードを入力します。お持ちでない場合は、[create new account] をクリックします。
- 4 最新の EBF を探してダウンロードします。
特定のバグ・フィックスに関する詳細については、該当する EBF または更新の隣にある [Info] ボタンをクリックします。[Info] リンクは新聞のような形のアイコンで表示されます。[Info] ページ上で OLE DB を検索します。

ODBC データ・ソースの作成

ODBC データ・ソースを作成するには、次の手順に従います。

- 1 [スタート] - [プログラム] - [Sybase SQL Anywhere] - [Adaptive Server Anywhere 7] - [ODBC Administrator] の順に選択します。
- 2 [System DSN] タブで [Add] をクリックします。
- 3 リストから Adaptive Server Anywhere 7.0 を選択して [Finish] をクリックします。
[ODBC Configuration for Adaptive Server Anywhere] ダイアログ・ボックスが開きます。

4 図 3 に示すように、[ODBC] タブで、[Data Source Name] に ASA 7.0 Sample と入力しま

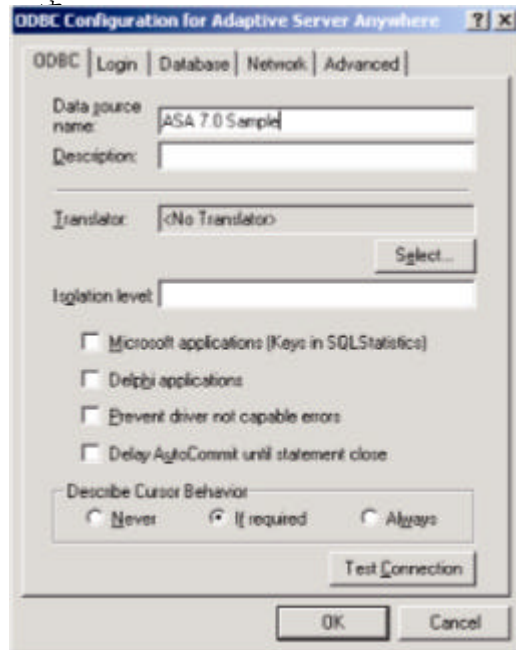


図 3 : [ODBC] タブ

5 図 4 に示すように、[Login] タブで、[User ID] に dba、[Password] に sql と入力します。

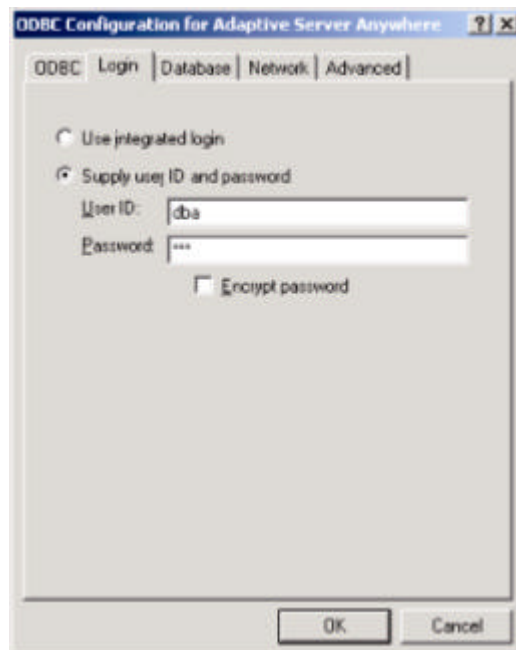


図 4 : [Login] タブ

- 6 [Database] タブで、[Browse] ボタンを使用してデータベース・ファイルを探します。図 5 に示すように、[Start Line] に dbeng7.exe と入力します。-c 12M はオプションです。

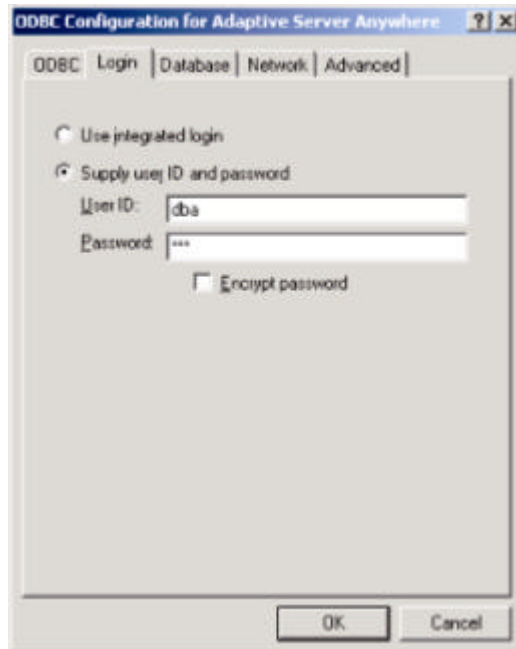


図 5 : [Database] タブ

- 7 [ODBC] タブで [Test Connection] をクリックして接続完了のメッセージを待ちます。
- 8 [OK] をクリックして [ODBC Configuration] ダイアログ・ボックスを閉じます。

このホワイトペーパーに掲載されている例では、使いやすいように、ODBC データ・ソースを使用してデータベースに接続します。ODBC データ・ソースを使用せずに接続を行う場合は、データベースのファイル名と場所を含めてください。このホワイトペーパーのすべてのサンプルで、Adaptive Server Anywhere サンプル・データベースである asademo.db を使用します。ASA 7.0 Sample という ODBC データ・ソースは、サンプル・データベース用に作成されているため、サンプル・コードを使用するときに再作成する必要はありません。

データベース接続の詳細については、Adaptive Server Anywhere のマニュアルの次の項を参照してください。

- Adaptive Server Anywhere バージョン 7 を使用している場合は、『ASA User's Guide』の第 1 部「Starting and Connecting to Your Database」の第 2 章「Connecting to a Database」。
- Adaptive Server Anywhere バージョン 8 を使用している場合は、『ASA Database Administration Guide』の第 2 章「Connecting to a Database」。

Visual Basic : ADO ライブラリの参照

Visual Basic プロジェクトを開始する前に、適切な ADO ライブラリを参照してください。ASA のインストールには、ActiveX Data Objects を使用するために必要な Microsoft Data Access Components (MDAC) 2.5 ライブラリが含まれています。2.6 ライブラリを適切にインストールするためのシステム要件により、また、ASA が自動的に 2.5 ライブラリをインストールすることにより、このホワイトペーパーは参照先の ActiveX Data Objects 2.5 ライブラリに基づいています。Microsoft では、一定の ADO オブジェクトの機能に影響する変更がわずかである ActiveX Data Objects 2.6 ライブラリを参照するように推奨しています。これらの変更は、このホワイトペーパーに掲載されているトピックには影響しません。

ADO ライブラリを参照するには、次の手順に従います。

- 1 [Projects] - [References] を選択します。図 6 に示すように参照メニューが表示されます。必ず、[Microsoft ActiveX Data Objects 2.5 Library] と [Microsoft ActiveX Data Objects Recordset 2.5 Library] を選択してください。

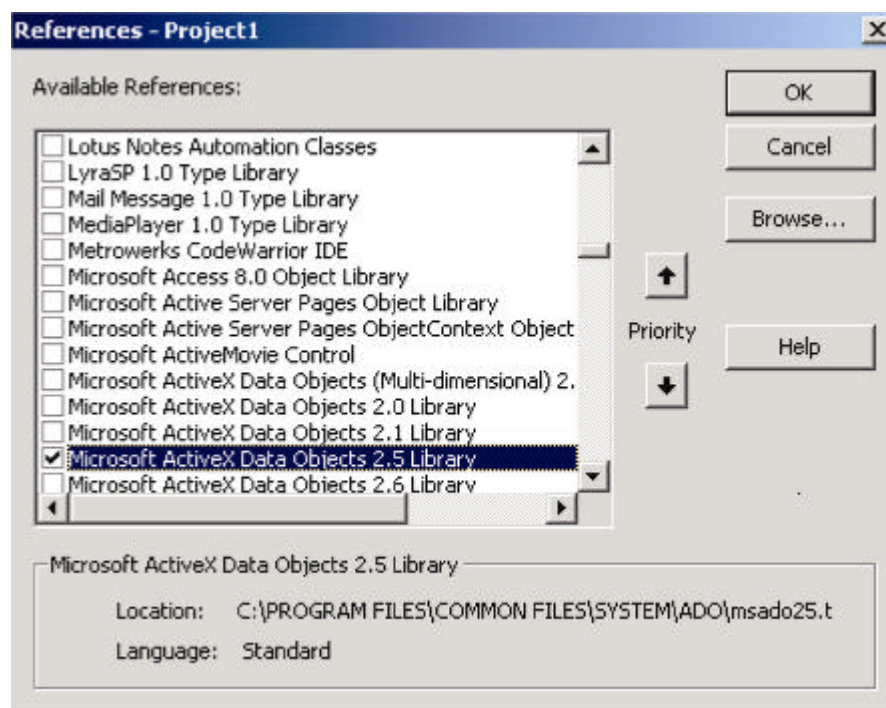


図 6 : ADO 参照ライブラリ

- 2 [OK] をクリックします。

プロバイダ

- プロバイダのプロパティは、データベースへの接続中に設定されます。ASAProv は、Adaptive Server Anywhere の OLE DB プロバイダです。Microsoft の SQL Server 用の OLE DB プロバイダは SQLOLEDB といいます。
- 複数のバージョンの Adaptive Server Anywhere がコンピュータにインストールされている場合、または以前に複数のバージョンがインストールされていた場合は、ランタイム・エラーが発生することがあります。表示されるエラーは、"Provider can not be found. It may not be properly installed" です。この問題を解決するには、**ASAProv.70** の構文を使用してプロバイダを指定し、dboledb7.dll ドライバを呼び出します。バージョン 8 を使用している場合は、**ASAProv.80** の構文を使用して dboledb8.dll ドライバを呼び出します。
- Microsoft から、MSDASQL という OLEDB プロバイダが提供されています。Adaptive Server Anywhere と共に使用すると、MSDASQL は iAnywhere Solutions の ODBC ドライバに対するブリッジの役割を果たします。使用しているアプリケーションで異常な動作が発生した場合は、MSDASQL をテストとして使用し、MSDASQL のプロバイダ動作を比較してください。これを行うには、プロバイダ名を ASAProv から MSDASQL に変更します。図 7 に、ASAProv と MSDASQL をデータ・ソースに対するインタフェースとして使用する方法を示します。

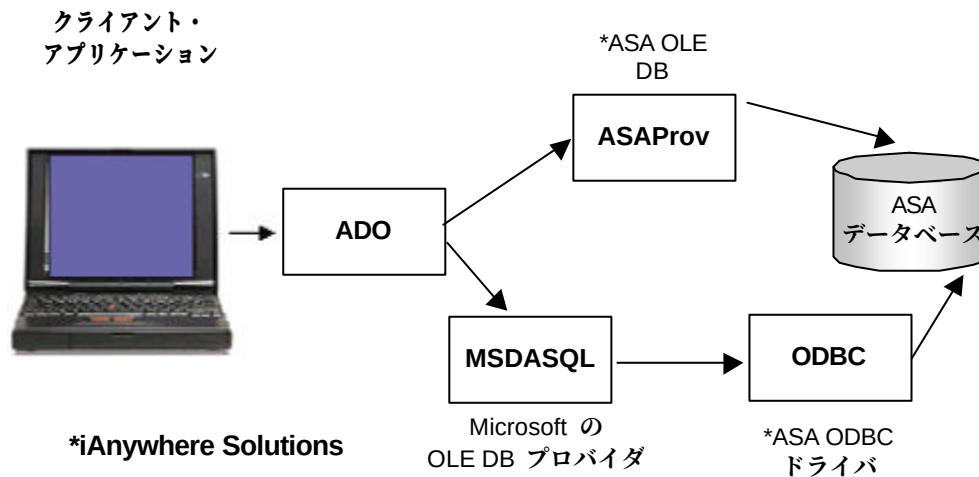


図 7：データ・ソースとインタフェースするための ASAProv と MSDASQL の使用

ADO オブジェクト

MDAC 2.5 ライブラリには 9 個の ADO オブジェクトがあります。これらのオブジェクトの詳細については、MSDN ライブラリを参照してください。Web 上には、ActiveX Data Objects に関するリソースが多数あります。このホワイトペーパーでは、次の頻繁に使用する ADO オブジェクトのみ取り上げます。

- Connection オブジェクト
- Command オブジェクト
- Recordset オブジェクト
- Record オブジェクト
- Stream オブジェクト
- Parameter オブジェクト
- Field オブジェクト
- Property オブジェクト
- Error オブジェクト

これらの ADO オブジェクトのメソッドおよびプロパティについては、<http://www.w3schools.com/ado> を参照してください。

Connection オブジェクト

Connection オブジェクトは、データ・ソースに接続する際に必要です。ただし、レコードセットを暗黙的に開くと、ADO を使用した接続が作成されます。接続を行うとき、特に複数のレコードセットを使用している場合は、Connection オブジェクトを使用することをお勧めします。

Connection オブジェクトの宣言

Connection オブジェクトを宣言するには、次の 2 つの方法があります。

```
1 Dim Connect as new ADODB.Connection
```

```
2 Dim Connect as ADODB.Connection  
   Set Connect = New ADODB.Connection
```

使用しているサブルーチンの外で接続を開いたままにしておきたい場合は、接続をグローバルに宣言してください。Connection オブジェクトをサブルーチン内だけで宣言すると、サブルーチンを終了するとすぐに、メモリを開放するために接続が閉じられます。

Connection Open メソッドの使用

Connection オブジェクトを開くには、さまざまな方法があります。次の例では、Adaptive Server Anywhere 内の ASA 7.0 Sample というサンプル・データベースを使用します。

- データ・ソース名を使用してデータベースに接続する場合は、ODBC ユーザ・データ・ソースがあることを確認します。サービスとして動作するデータ・ソースを使用する場合は、システム・データ・ソースを作成します。
- "Unable to connect to database server: Database server not running" というエラーが表示される場合は、データ・ソースが存在することを、ODBC 管理者に確認してください。
- ユーザ・データ・ソース ASA 7.0 Sample が、サンプル・データベース用に作成済みであるため、新しいデータ・ソースを下記の例のために作成する必要はありません。

```
Private Sub cmdConnect_Click()  
    Dim ConnString As String  
    Dim Connect As ADODB.Connection  
    ConnString = "Provider=ASAProv;Data Source=ASA 7.0 Sample;" & _  
        "User Id=dba;Password=sql;"  
    Set Connect = New ADODB.Connection  
    Connect.CursorLocation = adUseServer  
    Connect.Open ConnString
```

```
End Sub
```

```
Private Sub cmdConnect2_Click()  
    Dim Connect As ADODB.Connection  
    Set Connect = New ADODB.Connection  
    Connect.CursorLocation = adUseServer  
    Connect.Provider = "ASAProv"  
    Connect.ConnectionString = "Data Source=ASA 7.0 Sample"  
    Connect.Open  
End Sub
```

```
Private Sub cmdConnect3_Click()  
    Dim ConnString As String  
    Dim Connect As New ADODB.Connection  
    ConnString = "Provider=ASAProv; Data Source=ASA 7.0 Sample;" & _  
        "User Id=dba;Password=sql;"  
    Connect.CursorLocation = adUseServer  
    Connect.Open ConnString  
End Sub
```

- 接続が成功すると、Adaptive Server Anywhere エンジン・アイコンが  システム・トレイに表示されます。接続が閉じられると、このアイコンは非表示になります。

カーソル・ロケーション

上記のサンプル・コードの中で、Connect.CursorLocation = adUseServer に注目してください。CursorLocation は、データ記憶領域の一時リソースの保存場所を指定します。クライアント側のカーソルは、クライアント・コンピュータ上にメモリを割り付けます。サーバ側のカーソルは、サーバ上に割り付けられているメモリを利用します。現時点では、ASAProv はクライアント側のカーソルを完全にはサポートしていません。パフォーマンスの向上が可能な Adaptive Server Anywhere の優れたクエリ処理機能を利用するために、AdUseServer を使用することをお勧めします。ADO の仕様に基づいたデフォルトのカーソル・ロケーションは AdUseServer です。

接続の切断

```
Connect.close  
Set Connect = nothing
```

- Connect.close で ADO Connection オブジェクトが閉じられます。
- Set Connect = nothing で、データベースが切断され、他の接続がない場合はエンジンが停止します。このオプションの自動停止機能は、適切に切断されていることを確認するのに便利です。

Recordset オブジェクト

Recordset オブジェクトは、ADO で多く使用されるオブジェクトです。データベースからデータを取り出し、XML ファイルを開いて、データをファイルに保存することができます。Recordset オブジェクトは結果セットであり、テーブルのローやカラムを表現します。レコードセットを開くための構文は次のとおりです。

```
RS.Open Source, ActiveConnection, cursortype, locktype, options
```

Recordset オブジェクトを開く前に、Connection オブジェクトを開いておく必要があります。接続が開いていなければ、Recordset Open メソッドによって暗黙的に接続が行われます。ImplicitRS サンプルから次のサンプルが得られます。

```
Option Explicit
```

```
Dim Connect As New ADODB.Connection  
Dim RS As New ADODB.Recordset  
Dim ConnString As String  
Dim SQLString As String
```

```
Private Sub cmdConnectRS_Click()  
    ConnString = "Provider=ASAProv; Data Source=ASA 7.0 Sample;" & _  
        "User Id=dba; Password=sql;"  
    SQLString = "Select * FROM Product "  
    RS.Open SQLString, ConnString, adOpenStatic  
    Set DataGrid1.DataSource = RS  
End Sub
```

```
Private Sub cmdDisconnect_Click()  
    RS.Close  
    Set RS = Nothing  
    Set DataGrid1.DataSource = Nothing  
End Sub
```

Recordset Open メソッドを使用した暗黙的な接続が行われると、切断するときに次の構文が必要になります。

```
RS.close  
Set RS = nothing
```

接続がレコードセットを使用して暗黙的に行われたため、接続を nothing に設定しても、エンジンが停止しません。データベースから適切に切断するには、レコードセットを nothing に設定します。データを DataGrid または Microsoft Hierarchical FlexGrid (MSHFlexGrid) にバインドするときは、データ・ソースも nothing に設定しなければ、データベースから適切に切断できません。次の構文が、これらのデータ・コントロールを使用する際に必要です。

```
Set datagrid1.datasource = nothing
```

この構文を使用しなければ、データベース・エンジンが停止することではなく、この接続が原因で動作し続けます。

ソース

ここで言うソースとは、SQL 文のことです。Adaptive Server Anywhere は、T-SQL および Watcom-SQL の大容量のサブセットを使用してデータベースをクエリします。

Transact-SQL と Adaptive Server Anywhere の互換性については、Adaptive Server Anywhere のマニュアルの次の項を参照してください。

- Adaptive Server Anywhere バージョン 7 を使用している場合は、『ASA User's Guide』の第 6 部「The Adaptive Server Family」の第 32 章「Transact-SQL compatibility」。
- Adaptive Server Anywhere バージョン 8 を使用している場合は、『ASA SQL User's Guide』の第 12 章「Transact-SQL Compatibility」。

カーソル・タイプ

最も効果的なカーソル・タイプを選択すると、データ・アクセス処理を高速化できます。選択するカーソル・タイプは、結果セットで何をするか、データをどのようにスクロールするかによって異なります。ADO の仕様では、デフォルトのカーソル・タイプは adOpenForwardOnly です。

表 1 と表 2 は、指定するカーソル・タイプによって ASAProv を使用した場合と SQLOLEDB を使用した場合に何がサポートされるかを比較したものです。この 2 つの表は、SQLServer から Adaptive Server Anywhere へ移行する場合に便利です。

表 1：ASAProv の設定

ASAPROV カーソル・タイプ	ブックマーク のサポート	adHoldRecords のサポート	dHoldMove Previous のサポート	adResync のサポート
AdOpenDynamic	×	○	○	○
AdOpenForwardOnly	×	○	×	○
AdOpenKeyset	○	○	○	○
AdOpenStatic	○	○	○	○

表 2：Microsoft SQLOLEDB の設定

SQLOLEDB カーソル・タイプ	ブックマーク のサポート	adHoldRecords のサポート	adHoldMove Previous のサポート	adResync のサポート
AdOpenDynamic	×	×	○	×
AdOpenForwardOnly	×	×	×	×
AdOpenKeyset	○	○	○	○
AdOpenStatic	○	○	○	○

Microsoft のプロバイダは、カーソル・タイプ AdOpenDynamic に関しては adBookmark、adHoldRecords、adResync をサポートしませんが、ASAProv の AdOpenDynamic は、これらのうち adBookmark を除くすべてをサポートします。カーソル・タイプ adOpenForwardOnly に関しては、Microsoft はこれらの機能をまったくサポートしないのに対して、ASAProv はこれらのうちブックマークの使用を除くすべてのメソッドをサポートします。

Supports メソッドを使用すると、RS.Supports(CursorOption) を使用することにより、プロバイダが特定のオプションをサポートするかどうか確認できます。たとえば、カーソル adOpenDynamic を使用するレコードセットがブックマークをサポートするか確認するには、まず CursorTypeEnum を adOpenDynamic に設定してから、レコードセットを開いて Supports メソッドを使用します。

どのサーバ・カーソル・タイプが特定の ADO カーソルに対応するか確認しておく、どの動作が必要か決定するのに便利です。ASAProv は Embedded SQL を実装として使用しませんが、Embedded SQL DECLARE CURSOR 参照で、サーバ・カーソル・タイプが説明されます。表 3 は、ADO カーソル・タイプとサーバ・カーソルの意味の対応表です。

表 3：カーソル・タイプと対応するカーソルの意味

ADO カーソル・タイプ	ASA サーバ・カーソル
AdOpenDynamic	動的スクロール
AdOpenForwardOnly	スクロールなし
AdOpenKeyset	スクロール

AdOpenStatic	反応なし
--------------	------

表 3 に示したサーバ・カーソルの詳細については、Adaptive Server Anywhere のマニュアルの次の項を参照してください。

- Adaptive Server Anywhere バージョン 7 を使用している場合は、『ASA Reference』の第 9 章「SQL Statements」。
- Adaptive Server Anywhere バージョン 8 を使用している場合は、『ASA SQL Reference』の第 4 章「SQL Statements」。

ロック・タイプ

データベースへの変更は、Update メソッドおよび UpdateBatch メソッドを使用して行うことができます。モードはそれぞれ、更新のタイミングと、ローをデータ・ソースに追加する時期を示します。Immediate モードでは、データ・ソースへの変更がローごとに行われます。Batch モードでは、処理が完了したローのコレクションを使用してデータ・ソースが更新されます。ロック・タイプはデータ・ソースのレコードを更新する際に使用するモードを指定します。Batch モードの Update メソッドは変更内容を一時的に保存し、UpdateBatch メソッドは変更内容を永久に保存します。Microsoft では、バッチ更新はキーセットまたは静的カーソルでのみ使用することを推奨しています。ADO API 参照によると、デフォルトのロック・タイプは adLockReadOnly です。ASAProv プロバイダはこの仕様と一致しています。

ADO 仕様によると、指定した更新モードとロック・タイプの関係は表 4 に示すとおりです。

表 4：ADO ロック値

ロック・タイプ	更新モード	値
AdLockBatchOptimistic	Batch モード	4
AdLockOptimistic	Immediate モード	3
AdLockPessimistic	Immediate モード	2
AdLockReadOnly	更新不可	1

この 4 つのロック・タイプは、Adaptive Server Anywhere で、すべてのカーソルで利用できるわけではありません。Adaptive Server Anywhere は、すべてのカーソル・タイプで本当のオプティミスティック・ロックをサポートするわけではありません。キーセット駆動型カーソルではこのロック・タイプをサポートしますが、他のカーソルでは、ロック・タイプが adLockPessimistic に変更されます。AdLockBatchOptimistic はペシミスティック・ロック・タイプを使用しますが、このロック・タイプを使用すると Batch モードに入ることができ、Batch モードで更新バッチが変更内容をサーバに送ります。ロック・タイプを adLockOptimistic または adLockBatchOptimistic に指定すると、登録済みのロック・タイプがペシミスティックに変わります。オプティミスティック・ロックまたはバッチ・オプティミスティック・ロックでは、次の処理を試してください。

```
Debug.Print rs.LockType
```

Adaptive Server Anywhere データベースはバック・エンドでペシミスティック・ロックを使用するため、両方の [Immediate] ウィンドウの値が 2 になります。ロック・タイプの現在の値にかかわらず、ASAProv プロバイダで adLockBackOptimistic を使用するときは、Batch モードになります。

adLockReadOnly では、UpdateBatch を発行したときに変更が可能という問題が確認されました。この動作は、今後の EBF で対応予定です。

Recordset Open メソッドの使用

レコードセットを開くには、次の手順に従います。

```
RS.open Source, ActiveConnection, CursorType, LockType, Options
```

Open メソッドを使用するときは、Quick Info を使用すると、必要なパラメータを確認するのに便利です。Recordset Open メソッドを右クリックして [Quick Info] を選択します。次のコードを使用すると、Connection オブジェクトが開き、次にレコードセットが開いて、サンプル・データベースから product テーブルが格納されます。

```
Option Explicit
Dim Connect As New ADODB.Connection
Dim RS As New ADODB.Recordset
Dim SQLstring As String

Private Sub cmdOpenRS_Click()
    Set Connect = New ADODB.Connection
    Connect.CursorLocation = adUseServer
    Connect.Provider = "ASAProv"
    Connect.ConnectionString = "Data Source=ASA 7.0 Sample"
    Connect.Open

    SQLstring = "Select * from Product"

    Set RS = New ADODB.Recordset

    RS.Open SQLstring, Connect, adOpenStatic, _
        adLockReadOnly, adCmdText

End Sub
```

このコードは RSConnect プロジェクトにあります。

レコードセットのデータ更新

開発者は、レコードセットを使用してデータを更新し、新しいレコードをデータベースに挿入する作業を頻繁に行います。ASAProv プロバイダまたは MSDASQL プロバイダを使用してレコードセットを更新する際には、注意すべきことがあります。適切なロック・タイプを使用してください。また、adLockReadOnly は ADO Recordset のデフォルトであり、レコードの更新に使用するロック・タイプではありません。

これらのクエリは、ANSI 標準に基づいて更新可能とみなされない場合があります。Adaptive Server Anywhere バージョン 7.0 には、ANSI_UPDATE_CONSTRAINTS オプションがあります。このオプションの定義と値を再確認しておくといでしょう。

ANSI_UPDATE_CONSTRAINTS オプションの詳細については、Adaptive Server Anywhere のマニュアルの次の項を参照してください。

- Adaptive Server Anywhere バージョン 7 を使用している場合は、『ASA Reference』の第 5 章「Database Options」。
- Adaptive Server Anywhere バージョン 8 を使用している場合は、『Database Administration Guide』の第 16 章「Database Options」。

このオプションのデフォルト値は、Adaptive Server Anywhere バージョン 7 では CURSORS ですが、この動作は限定的である可能性があります。多くの場合、ORDER BY 句を含むクエリを発行したいと思うでしょう。ANSI 標準によると、これらの句では更新不能なカーソルが作成されます。

レコードを更新し、新しいレコードをデータベースに挿入するため、次の UpdateRS サンプルを検討してください。設定したオプションの実行を削除した場合は、レコードセットの更新は失敗します。

```
Option Explicit
Dim Connect As New ADODB.Connection
Dim RS As New ADODB.Recordset
Dim SQLstring As String

Private Sub cmdOpenRS_Click()
    Dim comm As New ADODB.Command

    Connect.CursorLocation = adUseServer
    Connect.Provider = "ASAProv"
    Connect.ConnectionString = "Data Source=ASA 7.0 Sample"
    Connect.Open

    'Order by statements or queries with criteria can have _
```

```

attributes which make them 'non-updatable' based on the ANSI _
standard.
'The following option can be set to circumvent/override the _
non-updatability.
'For further information on the option below, see SET OPTION _
reference in the help file.

comm.ActiveConnection = Connect
comm.CommandText = _
    "set temporary option ANSI_UPDATE_CONSTRAINTS='OFF'"
comm.Execute

SQLstring = "Select * from Product order by id"

Set RS = New ADODB.Recordset

RS.Open SQLstring, Connect, adOpenStatic, _
    adLockOptimistic, adCmdText

RS.Fields("name") = "Free_Stuff"
RS.Update

End Sub

Private Sub cmdInsert_Click()
Dim comm As New ADODB.Command
Set comm.ActiveConnection = Connect
comm.CommandText = _
    "INSERT INTO department VALUES(606, 'Accounting', 1090)"
comm.CommandType = adCmdText
comm.Execute
MsgBox "Successfully Inserted!"
Set comm = Nothing
Exit Sub
End Sub

```

さまざまなコントロールを使用したデータ表示

Visual Basic には、データベースからデータを表示するときに使用できる一連のデータ・アクセス・コントロール・コンポーネントが含まれています。Visual Basic 6.0 以上のバージョンでは、次のデータ・アクセス・コントロールを使用できます。

ライブラリ名	ライブラリに含まれる コントロール	本書で使用している コントロール名
Microsoft ADO DataControl	ADO Data コントロール	ADO Data
Microsoft DataList Controls	DBList コントロール	DataList
	DBCombo コントロール	DataCombo
Microsoft DataGrid	DataGrid コントロール	DataGrid
Microsoft Hierarchical FlexGrid	MSHFlexGrid コントロール	MSH FlexGrid

これらのコンポーネントを使用するには、まずツールボックスに追加してください。

これらのコンポーネントをツールボックスに追加するには、次の手順に従います。

- 1 図 8 に示すように、ツールボックス・メニューの任意の場所を右クリックして、ポップアップ・メニューから [Components] を選択します。

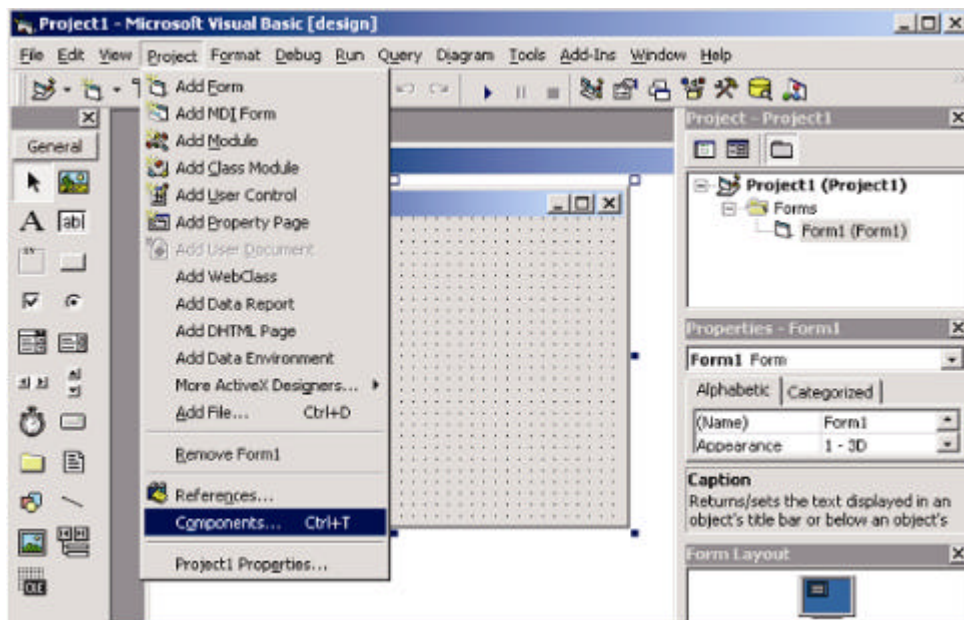


図 8：ADO コンポーネントをツールボックスに追加する

2 図 9 に示すように次のオプションを選択します。

- Microsoft Hierarchical FlexGrid Control 6.0
- Microsoft ADO Data Control 6.0
- Microsoft DataGrid Control 6.0
- Microsoft DataList Controls 6.0

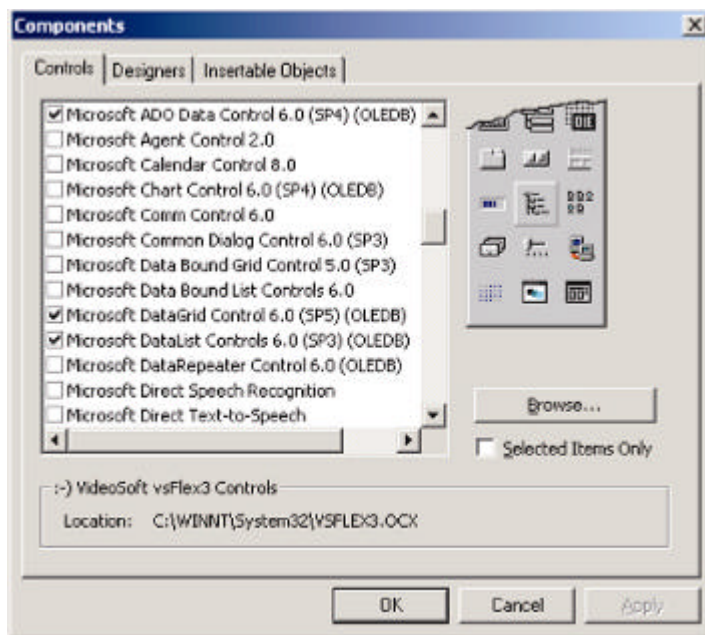


図 9 : [Components] ダイアログ・ボックス

3 [OK] をクリックして [Components] ダイアログ・ボックスを閉じます。

DataGrid コントロール

DataGrid コントロールを使用すると、データはグリッド形式またはテーブル形式で表示されます。データ・コントロールと共に使用して、データを編集することができます。レコードセットを DataGrid コントロールにバインドするには、まずコントロールをツールボックスに追加してから、DataGrid をフォームにドラッグ・アンド・ドロップし、最後に次の構文をコードに追加します。

```
Set DataGrid1.DataSource = RS
```

(下記のコード・サンプルで、この構文の正確な位置を確認してください)。DataGrid では、ブックマークを使用して結果セットを表示するため、ブックマークをサポートするカーソルのみを使用してください。そうしなければ、ローセットにブックマークを付けられないというエラーが生成されます。

DataGrid の使用には、一定の制限があります。DataGrid では小数値を含むデータを安定して表示できません。たとえば、サンプル・データベースの product テーブルで、unit_price は数値 (15,2) として宣言されています。レコードセットでこのカラムを取り出すと、値は adCurrency に変更されます。このタイプのカラムを DataGrid にバインドしたり [Immediate] ウィンドウに表示すると、問題が発生します。このカラムはブランクで表示されることがあるため、このカラムでは計算ができません。次のコードでこの現象を確認できます (同様のコードは DisplayData プロジェクトにもあります。ただし、DisplayData サンプルでは、product テーブルの代わりに employee テーブルを使用しています)。

```
Option Explicit
Dim Connect As New ADODB.Connection

Dim RS As New ADODB.Recordset
Dim SQLstring As String

Private Sub cmdShowData_Click()
    Set Connect = New ADODB.Connection
    Connect.CursorLocation = adUseServer
    Connect.Provider = "ASAProv"
    Connect.ConnectionString = "Data Source=ASA 7.0 Sample"
    Connect.Open

    SQLstring = "Select * from Employee"

    Set RS = New ADODB.Recordset

    RS.Open SQLstring, Connect, adOpenStatic, _
        adLockReadOnly, adCmdText
    Set DataGrid1.DataSource = RS
End Sub

Private Sub cmdDisconnect_Click()
    Connect.Close
    Set Connect = Nothing
End Sub
```

図 10 に示す unit_price カラムに注目しながら、[Show Data] ボタンと [Disconnect] ボタンを順番にクリックします。カラムがブランクになるときとデータが表示されるときがあることを確認してください。また、データベース・エンジンを接続、切断していることも確認してください。データ・ソースから切断すると、エンジン・アイコンはシステム・トレイに表示されません。

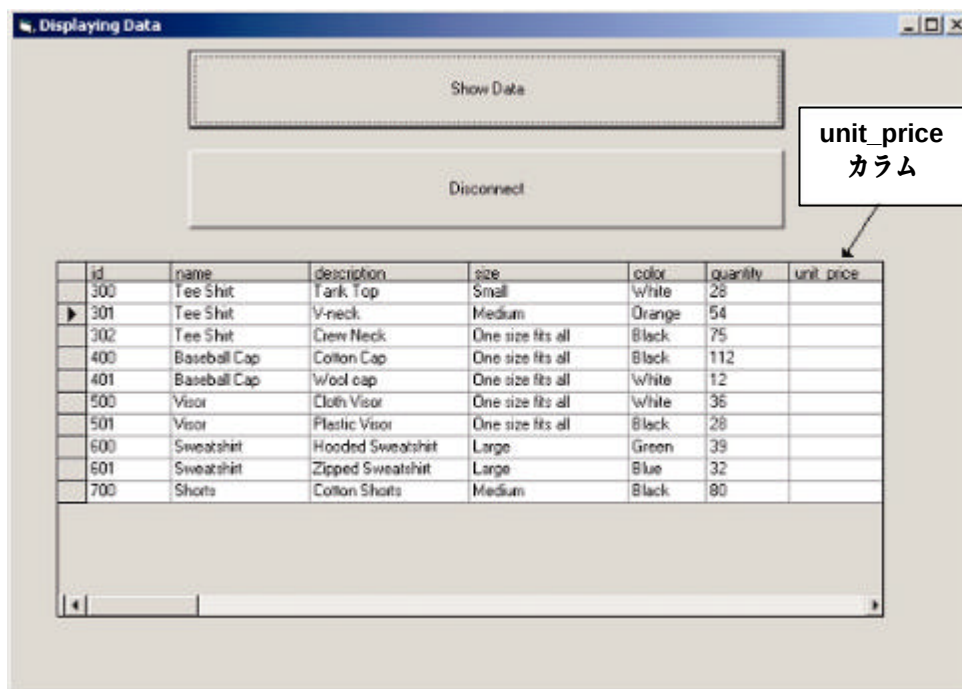


図 10 : [Displaying Data] ウィンドウの unit_price カラムの観察

Adaptive Server Anywhere バージョン 7.0.3 以上を使用している場合は、この問題はビルド 1520 以上で修正されています。最新の EBF へのアップグレード手順については、3 ページの「Adaptive Server Anywhere の設定」を参照してください。

7.0.3 より前のバージョンの Adaptive Server Anywhere を使用している場合は、この問題には次の 3 つの対処方法があります。

- MSDASQL プロバイダを使用する
- DataGrid の代わりに MSHFlexGrid を使用する
- キャスト関数を使用する

キャスト関数は、数値のカラムがブランクになる問題を一時的に解決するデータ型変換関数です。

```
SQLstring = "Select * from Product"
```

を、次のコードに置き換えます。

```
SQLstring = "Select id, name, quantity, size, color," & _  
" quantity, cast(unit_price as char) from Product"
```

MSHFlexGrid コントロール

DataGrid コントロールとは異なり、MSHFlexGrid コントロールではブックマークを使用せずにデータを表示するため、すべてのカーソル・タイプを処理するには、このデータ・コントロールを使用すると便利です。MSHFlexGrid は Visual Basic 6.0 以上で使用できます。MSHFlexGrid は、グリッド内でのデータの編集や更新ができない、読み取り専用のデータ・バインド・コントロールであるため、データを表示するときに使用してください。

MSHFlexGrid を使用しているときは、切断後にデータがグリッド上に残ります。MSHFlexGrid にデータをバインドするときは、DataGrid と同様に、

```
set datagrid1.datasource = RS
```

を、次のコードに置き換えます。

```
set MSHFlexGrid1.datasource = RS
```

MSHFlexGrid をクリアするには、MSHDataGrid.clear を使用します。

DataList コントロールと ComboBox コントロール

図 11 の DataList には、1 度に 1 つのカラムのデータが表示され、ComboBox にはドロップダウン・ボックスにリストが表示されます。

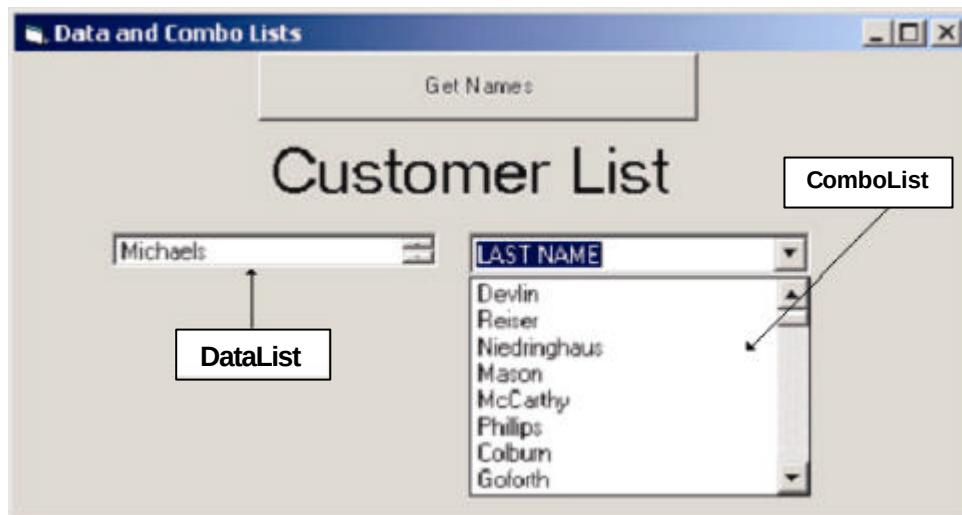


図 11 : DataList と ComboBox の比較

Microsoft DataList Controls 6.0 をツールボックスに追加すると、DataList コントロールと ComboBox コントロールの両方が追加されます。次の例では、DataList と ComboBox の両方を使用します。この 2 つの主な違いは、ComboBox ではドロップダウン・ボックスを使用するのに対して、DataList は結果セットをスクロールする形式である点です。次のプロジェクトは DataList という名前です。

Option Explicit

```
Private Sub cmdFillList_Click()  
    Dim Connect As New ADODB.Connection  
    Dim RS As New ADODB.Recordset  
    Dim SQLstring, ConnString As String  
  
    ConnString = "Provider=ASAProv; Data source=ASA 7.0 Sample;" & _  
        "User Id = dba; Password=sql;"  
    Connect.Open ConnString  
  
    SQLstring = "select lname, fname from customer"  
  
    RS.CursorLocation = adUseServer  
    RS.Open SQLstring, Connect, adOpenStatic
```

```

DataList1.ListField = "fname"
Set DataList1.RowSource = RS
DataCombo1.ListField = "lname"
Set DataCombo1.RowSource = RS

```

End Sub

ADO Data コントロール

ADO Data コントロールは、他のデータ・アクセス・コントロールと組み合わせて使用します。次の例では、DataGrid と ADO Data コントロールを使用して employee テーブルを表示します。

- 1 ADOData コントロールをフォームにドラッグ・アンド・ドロップします。次の変更を行います。

CommandType: adCmdText ([RecordSource] タブ)

ConnectionString: ボックス内をクリックして [...] ボタンをクリック。

- 2 図 12 に示すように、[Use Connection String] フィールドに ODBC データ・ソース名 Adaptive Server Anywhere 7.0 Sample を入力します。

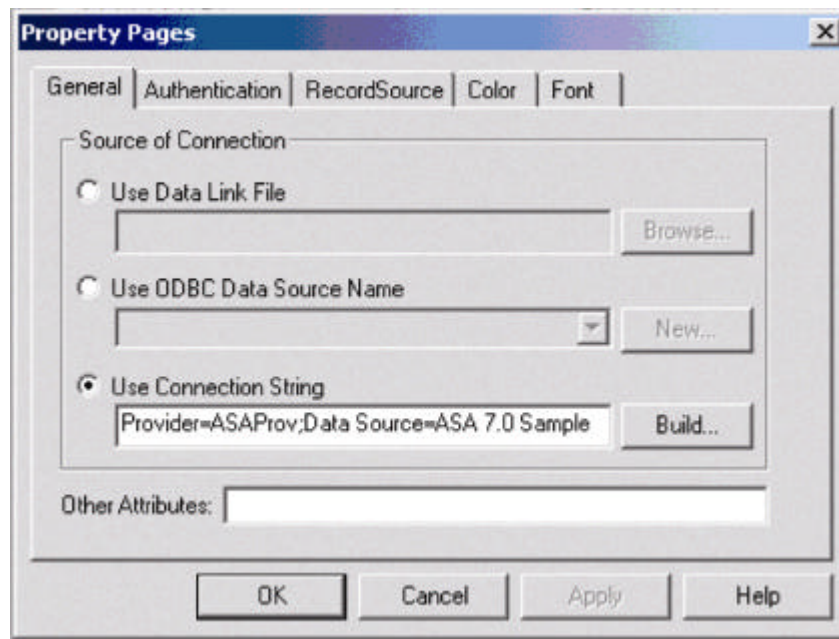


図 12 : ASA 7.0 Sample の ODBC データ・ソース名としての使用

3 [RecordSource] タブを選択します。図 13 のウィンドウが表示されます。

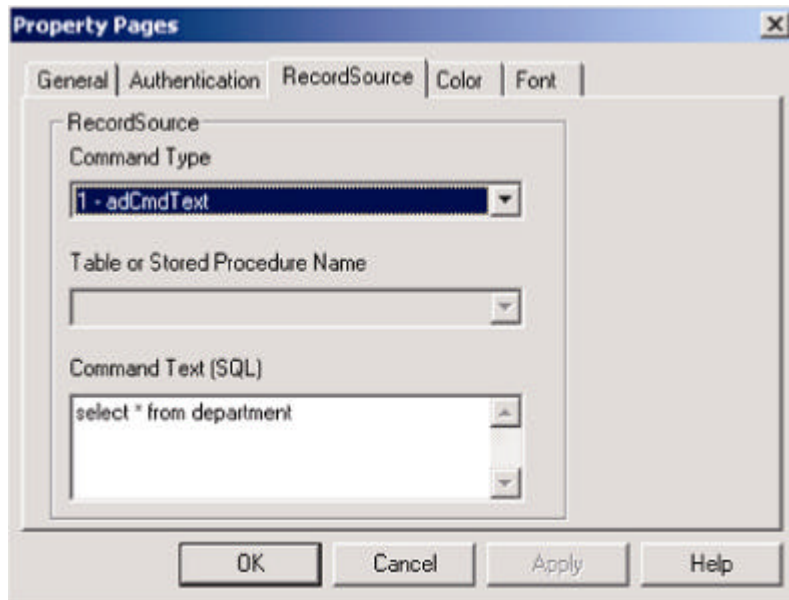


図 13：SQL コマンド・テキストの選択

4 [Command Text] フィールドに次のコマンドを入力します。

```
Select * from department
```

- 5 [Properties] ページで、[CursorLocation] が [AdUseServer] であることと、[CursorType] が [adOpenStatic] または [adOpenKeyset] であることを確認します。これは、この ADO Data は DataGrid またはその他のコントロールとバインドして使用するため、双方向かつブックマークの使用が可能なカーソル・タイプが必要になる可能性が高いからです。[LockType] はニーズに合わせて選択します。図 14 に、ADO Data のデータの値を示します。

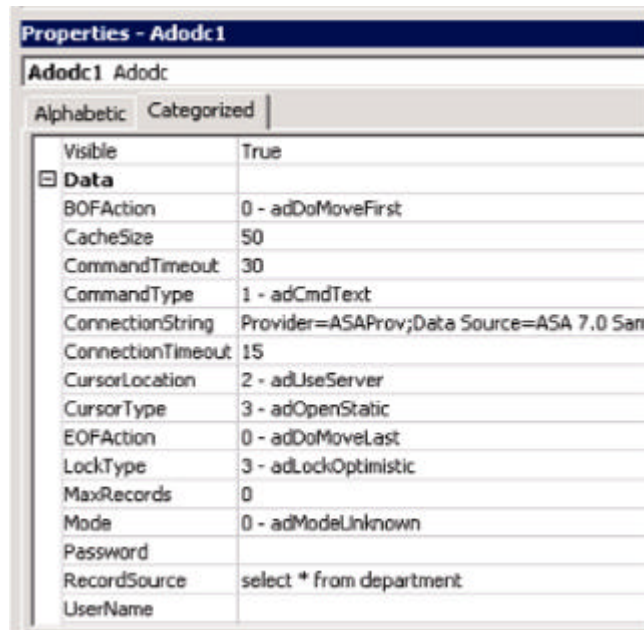


図 14：ADO Data の CursorLocation と CursorType の設定

- 6 DataGrid をフォームにドラッグ・アンド・ドロップします。

- 7 [DataSource] プロパティを [Adodc1] に変更します。[DataSource] プロパティを変更すると、ADO Data コントロールがグリッドにバインドされます。ADO Data コントロールをクリックすると、図 15 のように DataGrid 内を移動できます。

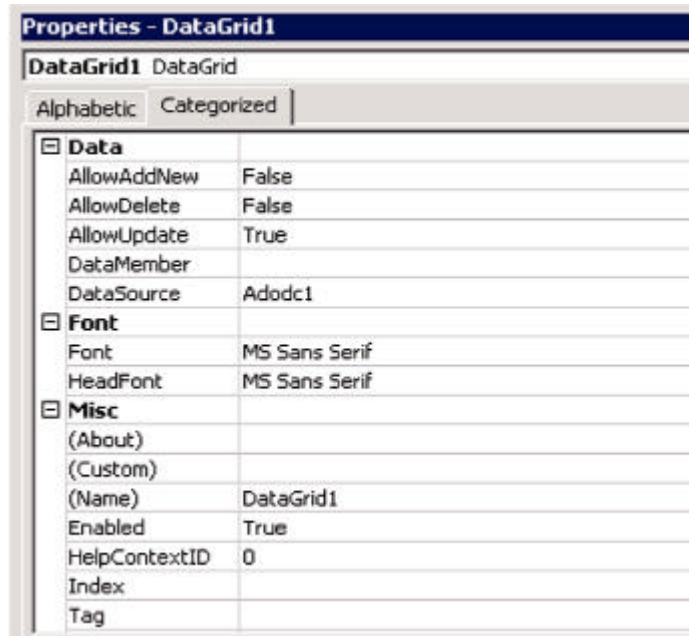


図 15 : DataGrid プロパティの ADO Data コントロールへのバインド

これはプログラム的にも設定できます。次のサンプル・コードは、ASAProv を使用して ADO Data と DataGrid のプロパティをプログラム的に設定する方法を示しています。

```
Private Sub cmdShowData_Click()  
  
    Adodc1.ConnectionString = _  
        "Provider=ASAProv; Data Source=ASA 7.0 Sample"  
    Adodc1.RecordSource = "Select * from employee"  
    Adodc1.CursorLocation = adUseServer  
    Adodc1.CursorType = adOpenStatic  
    Set DataGrid1.DataSource = Adodc1  
  
End Sub  
  
Private Sub cmdDisconnect_Click()  
  
    Connect.Close  
    Set Connect = Nothing  
  
End Sub
```

次のサンプル・コードのように、レコードセットを ADO Data コントロールと共に使用することもできます。このサンプルは、ADO Data の例に含まれています。

```
Option Explicit
Dim Connect As ADODB.Connection

Dim RS As ADODB.Recordset
Dim SQLstring As String

Private Sub cmdShowData_Click()
    Set Connect = New ADODB.Connection
    Connect.CursorLocation = adUseServer
    Connect.Provider = "ASAProv"
    Connect.ConnectionString = "Data Source=ASA 7.0 Sample"
    Connect.Open

    SQLstring = "Select * from employee"

    Set RS = New ADODB.Recordset

    RS.Open SQLstring, Connect, adOpenStatic, _
        adLockReadOnly, adCmdText

    Set Adodc1.Recordset = RS

    Set DataGrid1.DataSource = Adodc1
End Sub

Private Sub cmdDisconnect_Click()

    Connect.Close
    Set Connect = Nothing

End Sub
```

レコードセットとレコードセットのページのカウント

レコードセットの RecordCount プロパティは、Recordset オブジェクト内のレコード数をカウントする際に使用します。RecordCount は、レコードセットを開く際に使用するカーソル・タイプに依存します。ASAProv RS では、RecordSet.RecordCount プロパティを使用すると、カーソル・タイプごとの推定レコード数が得られます。ただし adOpenForwardOnly は例外で、-1 という値になります。Adaptive Server Anywhere の ROW_COUNTS オプションを ON に設定すると、正確なロー数が得られます。ただし、ROW_COUNTS オプションを ON にすると、パフォーマンスが大幅に低下する場合があります。

レコードセットの PageCount プロパティは、レコードセットのページ数をカウントする際に使用します。PageCount の動作は RecordCount と同様です。RecordCount と PageCount の構文は次のとおりです。

RS.RecordCount

RS.PageCount

図 16 に示す AbsolutePosition プロジェクトは、WHILE ループを使用してレコードをカウントし、最大レコード数をレコードセットの RecordCount プロパティに設定します。AbsolutePosition プロジェクトでは、デモンストレーションのために進行状況を示すバーも使用します。このプロジェクトに含まれている [Support] ボタンを使用すると、指定したロック・タイプを用いた場合にブックマークがサポートされるかどうか確認できます。



図 16 : AbsolutePosition

```

Option Explicit
Dim Connect As ADODB.Connection

Private Sub cboSupport_Click()
Call Support
End Sub

Private Sub cmdCount_Click()
Dim RS As ADODB.Recordset
Dim sMessage As String

If Connect.State <> adStateOpen Then
Set Connect = New ADODB.Connection
Connect.Provider = "ASAProv"
Connect.ConnectionString = _
    "uid=dba;pwd=sql;dbf=" & App.Path & "\asademo.db"
Connect.Open
End If

Set RS = New ADODB.Recordset
RS.CursorLocation = adUseServer
RS.Open "DBA.sales_order_items", Connect, adOpenStatic, _
    adLockReadOnly, adCmdTable

ProgressBar1.Min = 0
ProgressBar1.Max = RS.RecordCount

Do While Not RS.EOF
    ProgressBar1.Value = RS.AbsolutePosition
    lblRecordCount.Caption = "Record: " & RS.AbsolutePosition & " _
        of " & RS.RecordCount
    DoEvents
    RS.MoveNext
Loop

End Sub

Private Sub Support()
Dim adoSupport As New ADODB.Recordset
adoSupport.Open "customer", Connect, adOpenForwardOnly, _
    adLockReadOnly, adCmdTable
MsgBox "Bookmark is supported? " & _
    adoSupport.Supports(adBookmark), vbOKOnly + vbInformation, _
    "Properties Support"

```

```

adoSupport.Close
Set adoSupport = Nothing

End Sub

Private Sub cmdSupport_Click()
Call Support
End Sub

Private Sub Form_Load()
Set Connect = New ADODB.Connection
Connect.Provider = "ASAProv.70"
Connect.ConnectionString = "uid=dba;pwd=sql;dbf=" & _
    App.Path & "\asademo.db"
Connect.Open

End Sub

```

レコードセット内の移動

レコードセット内の移動には、Move、MoveFirst、MoveLast、MoveNext、MovePrevious の 5 つのメソッドがあります。これらのメソッドを使用できるかどうかは、使用するカーソルのタイプによって異なります。Move メソッドは、レコード何個分移動するかを指定します。たとえば、RS.move 8 と指定すると、9 番目のレコードに移動します。Move メソッドで負の値を使用すると、後方に移動することもできます。AdOpenForward は前方専用のカーソルであるため、前方移動のみに使用できます。このカーソルに関しては、ASAProv は MoveNext 移動だけをサポートしますが、SQLOLEDB は MoveNext 移動と MoveFirst 移動をサポートします。前方用カーソルと ASAProv で最初に使用する移動が MoveLast だった場合、エラーは表示されず、レコードセット内の最初のレコードが返されます。

ブックマークの使用

ブックマークを使用すると、結果セット内の特定の場所にマークを付け、データをスクロールした後にブックマークを作成した場所に戻ることができます。

このホワイトペーパーが執筆された時点では、ブックマークと DataGrid に関連する問題が存在しました。レコードセットを DataGrid にバインドしてからブックマークを作成すると、誤ったレコードにブックマークが付けられます。ブックマークは、指定したレコードの直後のレコードに付けられます。たとえば、図 17 のように、9 番目のレコードが必要な場合に 10 番目のレコードにブックマークが付けられます。この動作は、ASAProv を使用している場合にのみ発生します。

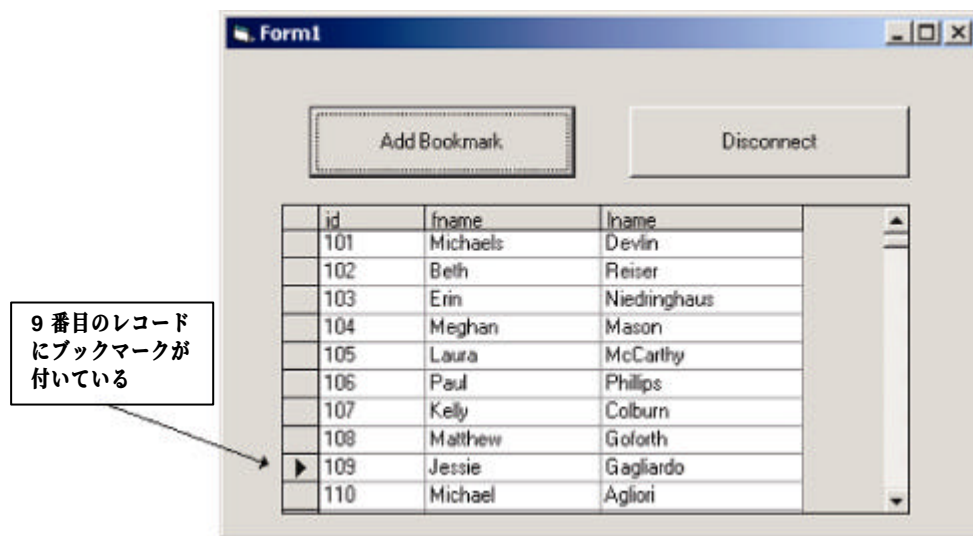


図 17：レコードへのブックマークの付加

この問題には次のような解決法があります。

- ブックマークを作成し、データをスクロールした後にレコードセットを DataGrid にバインドします。下のコードで、データを DataGrid にバインドするのに適切なタイミングを確認してください。
- MSHFlexGrid も使用できますが、このグリッドを使用すると、実際にどのローにブックマークが付いているか確認できません（ブックマークが付いているローを示す矢印が表示されないため）。MSDASQL でもこの問題は修正できますが、RS.MoveLast の行でエラーになります。
- また、MoveFirst を使用して位置を確認してからブックマークを設定するという解決法も考えられます。

このサンプル・コードで使用するもう 1 つのメソッドは、レコードセットの Debug.print メソッドです。テストのために、レコードセット内のデータを [Immediate] ウィンドウに表示できます。これを行うには、次のコードを使用します。

```
Debug.Print RS(0) & vbTab & RS(1) & vbTab & RS(2) & vbTab RS(0)
```

RS(0) は結果セットの第 1 カラム、RS(1) は第 2 カラムを表し、以下同じように続きます。次の例は、AddBookmark プロジェクトにあります。

```
Dim RS As New ADODB.Recordset
Dim Connect As New ADODB.Connection

Private Sub cmdAddBookmark_Click()
```

```

Dim varBookmark As Variant
Dim SQLstring, ConnString As String

SQLstring = "SELECT id, fname, lname FROM customer "
ConnString = "Provider=ASAProv; Data Source=ASA 7.0 Sample;" & _
    "User Id=dba;Password=sql"

Connect.open ConnString
RS.open SQLstring, Connect, adOpenStatic, adLockOptimistic,
adCmdText

'Setting the datagrid here will throw off your bookmark by 1 record
'Set DataGrid1.DataSource = RS

'We recommend here that you call RS.MoveFirst (as long as you
'are not using a ForwardOnly cursor). This ensures that you are
'starting from the first record in the recordset and that your
'bookmark is set to the correct place. The following is the
'alternative method

'Move to the ninth recordset
RS.Move 8
'Create bookmark
varBookmark = RS.Bookmark
'Move to last record and Print
RS.MoveLast
Debug.Print RS(0) & vbTab & RS(1) & vbTab & RS(2) & vbTab
'Move to First record and print
RS.MoveFirst
Debug.Print RS(0) & vbTab & RS(1) & vbTab & RS(2) & vbTab
'Move to bookmarked record
RS.Bookmark = varBookmark
Debug.Print RS(0) & vbTab & RS(1) & vbTab & RS(2) & vbTab

'You want to set the datagrid here
Set DataGrid1.DataSource = RS

End Sub

```

```

Private Sub cmdDisconnect_Click()
    RS.Close
    Set RS = Nothing
    Connect.Close
    Set Connect = Nothing
    Set DataGrid1.Datasource = Nothing
End Sub

```

複数のレコードセットの取り出し

ADO の 1 つのレコードセット・オブジェクト内に含まれている結果セットを複数得ることができます。図 18 に示す [Double Recordset] の例では、2 つの結果セットは、RS レコードセット・オブジェクトに含まれている従業員名と部署名です。このプロジェクトを実行する前に、procedure.sql ファイルを実行してストアド・プロシージャを作成します。次の結果セットに移動するには、次のコマンドを実行します。

```
Set RS = RS.NextRecordset
```

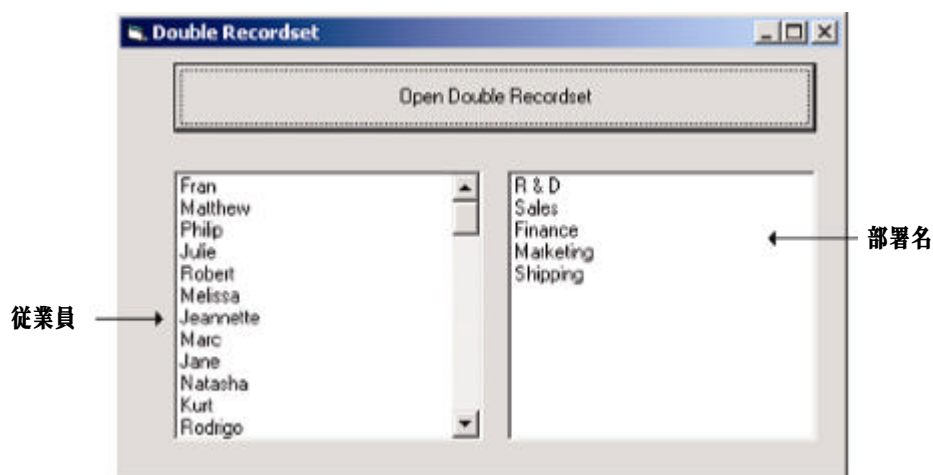


図 18 : Double Recordset

このプロジェクトでは、getemployeesanddept というストアド・プロシージャを作成してください (ストアド・プロシージャのコードはコメントを参照)。現在の EBF および Adaptive Server Anywhere バージョン 8 では、複数結果セットのサポートのデフォルト設定が変更されていることに注意してください。このプロパティの設定手順については、このプロジェクトの末尾にあるコメントを参照してください。

```

Option Explicit
Dim Connect As ADODB.Connection
Dim RS As ADODB.Recordset
Dim objError As ADODB.Error
Dim strErr As String

Private Sub cmdOpenRecordSet_Click()
    On Error GoTo cmdOpenDoubleRecordSet_Err

    Call GetEmployeesAndDept
    Call LoadEmployeeName

    Set RS = RS.NextRecordset

    Call LoadDept
    Call CloseRecordSet
    Exit Sub

cmdOpenDoubleRecordSet_Err:
    Call ErrorHandler

End Sub
Private Sub CloseRecordSet()
    RS.Close
    Set RS = Nothing

End Sub
Private Sub GetEmployeesAndDept()
    On Error GoTo getEmpAndDept_err

    'create procedure getemployeesanddept
    'as
    'select emp_fname from employee
    'select dept_name from department

    Set RS = New ADODB.Recordset
    RS.Open "call GetEmployeesAndDept()", Connect, adOpenStatic,
adLockReadOnly, adCmdText
    Exit Sub

getEmpAndDept_err:
    Call ErrorHandler

```

```

End Sub

Private Sub LoadEmployeeName()
    lstEmployee.Clear
    Do While Not RS.EOF
        lstEmployee.AddItem RS!emp_fname
        RS.MoveNext
    Loop
End Sub

Private Sub LoadDept()
    lstDept.Clear
    Do While Not RS.EOF
        lstDept.AddItem RS!dept_name
        RS.MoveNext
    Loop
End Sub

Private Sub Form_Load()
    On Error GoTo Connection_Err

    Call Conn
    Exit Sub
Connection_Err:
    Call ErrorHandler

End Sub

Private Sub ErrorHandler()
    For Each objError In Connect.Errors
        strErr = strErr & " Description : " & objError.Description &
vbCrLf & vbCrLf & _
            "      SQL CODE : " & objError.NativeError &
vbCrLf & vbCrLf & _
            "      SQL STATE : " & objError.SQLState &
vbCrLf
    Next
    MsgBox strErr, vbCritical + vbOKOnly, "Error Connecting"

End Sub

Private Sub Form_Unload(Cancel As Integer)
    On Error Resume Next

```

```

Connect.Close
Set Connect = Nothing

End Sub

Private Sub Conn()
Set Connect = New ADODB.Connection

Connect.Provider = "ASAProv.70"
Connect.ConnectionString = "Data Source = ASA 7.0 Sample"

Connect.Open
'Recently default for multiple resultset support had been changed.
'Uncomment the following line for ebfs that require this property.
'Connect.Properties("ASA Multiple Results") = True

End Sub

```

ストアド・プロシージャ

ストアド・プロシージャは、クライアント・アプリケーションではなくデータベース内に保存されます。クライアント・アプリケーションは、ストアド・プロシージャを繰り返し呼び出して、特定の関数に自動的にアクセスできます。

Transact-SQL と Adaptive Server Anywhere の互換性については、Adaptive Server Anywhere のマニュアルの次の項を参照してください。

- Adaptive Server Anywhere バージョン 7 を使用している場合は、『ASA User's Guide』の第 6 部「The Adaptive Server Family」の第 32 章「Transact-SQL compatibility」。
- Adaptive Server Anywhere バージョン 8 を使用している場合は、『ASA SQL User's Guide』の第 12 章「Transact-SQL Compatibility」。

Connection プロジェクトは、Recordset Open メソッドでストアド・プロシージャ sp_retrieve_contacts を呼び出します。このストアド・プロシージャは asademo.db データベースにあります。図 19 に、このプロジェクトを実行中の様子を示します。

	id	last name	first name
▶	1	Hildebrand	Jane
	2	Simmon	Larry
	3	Catch	Susan
	4	Lambert	Terry
	5	Sullivan	Dorothy
	6	Paul	Rose
	7	Glassmann	Beth
	8	Powell	Gene
	9	Fish	Jeffrey
	10	Clarke	Molly
	11	Kelley	William

図 19 : Adaptive Server Anywhere への接続

```
Dim Connect As New ADODB.Connection

Private Sub cmdConnect_Click()
    Connect.Provider = "ASAProv.70"
    Connect.ConnectionString = "Data Source = ASA 7.0 Sample"
    Connect.Open
End Sub

Private Sub cmdExecuteProcedure_Click()
    Dim RS As New ADODB.Recordset
    RS.Open "call sp_retrieve_contacts()", Connect,
        adOpenKeyset, adLockReadOnly, adCmdText
    RS.MoveFirst
    Set DataGrid1.DataSource = RS
    DataGrid1.Refresh
End Sub
```

```

Private Sub ErrorHandler()
    Dim strErr As String
    For Each objError In Connect.Errors
        strErr = strErr & " Description : " &
            objError.Description & vbCrLf & vbCrLf & _
            "      SQL CODE : " & objError.NativeError &
            vbCrLf & vbCrLf & _
            "      SQL STATE : " & objError.SQLState & vbCrLf
    Next
    MsgBox strErr, vbCritical + vbOKOnly, "Error Connecting"
End Sub

```

レコードセットを使用した切断

どの切断メカニズムを選択するかは、データベースへの接続方法と、使用しているデータ・コントロールによって異なります。接続オブジェクトを使用して、レコードセット・オブジェクトを使用せずにデータベースに接続している場合の切断は、次のコマンドを実行します。

```

Connect.close
Set Connect = nothing

```

同様に、レコードセットを使用して暗黙的に接続している場合は、次のコマンドを実行します。

```

RS.close
Set RS = nothing

```

暗黙的な接続は、Recordset Open メソッド中に接続文字列をパラメータとして渡すことによって行われます。レコードセットを使用した暗黙的な接続はお勧めできません。DataGrid や MSHFlexGrid などのデータ・コントロールにバインドされているデータでは、データ・ソースを nothing に設定してデータベース・エンジンを停止可能にしてください。これについては次のコードを参照してください。

```

Set MSHFlexGrid1 = nothing
RS.close
Set RS = nothing

```

このコードにより、データベース・エンジンが適切に停止され、SQL アイコンがシステム・トレイから削除されます。次の例は、データベースに接続し、Recordset オブジェクトを開いて、データベースから切断する適切な方法を示しています。

```

Dim Connect As New ADODB.Connection
Dim ConnString As String
Dim RS As New ADODB.Recordset
Dim SQLstring As String

Private Sub cmdConnect_Click()

    ConnString = "Provider=ASAProv; Data Source=ASA 7.0 Sample;" & _
        "User Id=dba; Password=sql;"
    SQLstring = "Select * from Customers"

    'Make the connection
    Connect.Open ConnString

    'Open Recordset Object
    RS.Open SQLstring, Connect, adOpenStatic
    Set DataGrid1.DataSource = RS
End Sub

Private Sub cmdDisconnect_Click()
    Set DataGrid1 = nothing
    RS.close
    Set RS = nothing
    Connect.Close
    Set Connect = nothing
End Sub

```

Command オブジェクト

Command オブジェクトは、データ要求を実行し、データベースからストアド・プロシージャを呼び出すために使用します。Command オブジェクトは、同じ要求を複数回行う場合や、詳細なパラメータを渡す必要がある場合に便利です。

データベースに対して同様のクエリを行う方法には、次のコードも含まれます。

```

Connect.Execute
RS.open SQLstring, ActiveConnection

```

コマンドのタイプを表 5 に示します。最も柔軟性の高いコマンド・タイプは adCmdText です。パフォーマンス向上のために、常にコマンド・タイプを指定してください。コマンド・タイプが不明の場合、正しいタイプが見つかるまでさまざまなタイプが実行されるため、パフォーマンスが低下します。

AdCmdFile および AdCmdTableDirect は、必ず Recordset.open メソッドと共に使用します。

表 5：コマンド・タイプの機能

CommandTypeEnum	機能
AdCmdUnspecified	
AdCmdText	テキストまたはストアド・プロシージャとして評価。
AdCmdTable	SQL クエリによって生成されるテーブル。
AdCmdStoredProc	ストアド・プロシージャとして実行。
AdCmdUnknown	コマンド・タイプが不明のときに使用。まずテキストとして実行し、次にストアド・プロシージャとして、次にベース・テーブルとして実行。
AdCmdFile	常に Recordset.open メソッドと共に使用。
AdCmdTableDirect	常に Recordset.open メソッドと共に使用。

CommandObject の例では、SQL 文は CommandText プロパティ内に格納されます。次にこの SQL 文は RS = Commandobj.Execute という構文を使用して実行され、レコードセットに格納されます。RS = Commandobj.Execute を使用しているときは、カーソル・タイプの変更はできません。カーソル・タイプが変更できない場合、デフォルトのカーソル・タイプは adOpenForwardOnly であり、これは DataGrid と共に使用できるため、問題が発生します。ただしこの問題は、MSHFlexGrid を使用すると解決できます。MSHFlexGrid ではブックマークを使用せずにデータが表示されるためです。この問題は、現在調査中です。次の例は、CommandObject サンプルにあります。

Option Explicit

```
Dim RS As New ADODB.Recordset
Dim ConnString, SQLstring As String
Dim comm As New ADODB.Command
Dim Connect As New ADODB.Connection
```

```
Private Sub cmdCommand_Click()
    'The data request using SQL statements
    comm.CommandText = "SELECT ID, line_id, " & _
        "Prod_id " & _
        "FROM sales_order_items " & _
        "WHERE ID = 2001 "

    comm.CommandType = adCmdText
    ConnString = "Provider=ASAProv;Data Source=ASA 7.0 Sample"
```

```

'Open the Connection
Connect.Open ConnString

'Set Command Object to the active connection
comm.ActiveConnection = Connect

'Execute the command and hold the data from the query in the
recordset
Set RS = comm.Execute

'Display data on the MSHFlexGrid
Set MSHFlexGrid1.DataSource = RS

End Sub

```

Name プロパティの使用

Name プロパティを使用すると、名前を使用してコマンドを識別できます。指定した名前を呼び出すことにより、コマンドを実行できます。次のサンプルは、NameCommand プロジェクトにあります。

```

Dim Connect As New ADODB.Connection
Dim RS As New ADODB.Recordset
Dim comm As New ADODB.Command
Dim ConnString As String

Private Sub cmdGetIDS_Click()
'Connect to data source
ConnString = "Provider=ASAProv.70; Data Source=ASA 7.0
Sample;" & _ "User Id=dba;Password=sql;"
Connect.Open ConnString

comm.CommandText = "SELECT emp_id, emp_lname FROM employee"
comm.CommandType = adCmdText

'Name the CommandText to GetEmployeeIDs
comm.Name = "GetEmployeeIDs"
comm.ActiveConnection = Connect
RS.CursorType = adOpenStatic

'Execute the command
Connect.GetEmployeeIDs RS

```

```

        Set DataGrid1.DataSource = RS

End Sub

Private Sub cmdDisconnect_Click()
    RS.Close
    Connect.Close
    Set Connect = Nothing
    Set comm= Nothing
End Sub

```

パラメータ

パラメータは、ストアド・プロシージャや SQL クエリで使用できます。in、out のパラメータを使用した単純なストアド・プロシージャを下記に示します。このコードは、ASAProv でパラメータを使用する方法を示しています。

下記のコードは、ストアド・プロシージャ内のパラメータを示すものであり、ProcParms サンプルにあります。このプロジェクトは、ストアド・プロシージャとパラメータを使用した 2 つの異なるシナリオを示しています。一方は input パラメータを使用した例で、もう一方は InOut パラメータを使用した例です。このプロジェクトでは、フォーム上の最初のボタンで Parameter 例が呼び出されます (ストアド・プロシージャのコードはコメントを参照)。この動作は、エンジン・コンソールを開くことによって確認できます。2 番目のボタンは、パラメータが変更されて右側のテキスト・ボックスに反映される様子を示します。

* 注意：ストアド・プロシージャの定義はコメントに示してあります。この例では input パラメータを使用します。

```

'General definitions

Dim Connect As New ADODB.Connection
Dim comm As New ADODB.Command
Private Sub Command1_Click()
    Dim inParam As ADODB.Parameter
    Dim inParam2 As ADODB.Parameter

    'create procedure foo_in (in arg1 int, in arg2 char(30))
    'begin
    '    message arg1, arg2;
    'end;

    comm.CommandText = "{call foo_in(?,?)}"
    comm.CommandType = adCmdText

```

```

comm.ActiveConnection = Connect

Set inParam = comm.CreateParameter("inParam", adInteger, _
    adParamInput)
Set inParam2 = comm.CreateParameter("inParam2", adChar, _
    adParamInput, 4)
comm.Parameters.Append inParam
comm.Parameters.Append inParam2
inParam.Value = 4
inParam2.Value = "554"

comm.Execute

End Sub

Private Sub Command2_Click()
Dim inoutParam As ADODB.Parameter
Dim inoutParam2 As ADODB.Parameter

'create procedure foo (inout arg1 int, inout arg2 char(4))
'begin
' message arg1, arg2;
' set arg1=99;
' set arg2=88;
' message arg1, arg2;
' end;

comm.CommandText = "call foo(?,?)"
comm.CommandType = adCmdText
comm.ActiveConnection = Connect

Set inoutParam = comm.CreateParameter("inoutParam", adInteger, _
    adParamInputOutput)
Set inoutParam2 = comm.CreateParameter("inoutParam2", adChar, _
    adParamInputOutput, 4)
comm.Parameters.Append inoutParam
comm.Parameters.Append inoutParam2
inoutParam.Value = 4
inoutParam2.Value = "554"

comm.Execute
Text1.Text = inoutParam.Value
Text2.Text = inoutParam2.Value

```

```
End Sub
```

```
Private Sub Form_Load()
```

```
Connect.Open _
```

```
"Provider=ASAProv; UID=dba; PWD=sql; Data Source=ASA 7.0 Sample"
```

```
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
```

```
Connect.Close
```

```
End Sub
```

次のコードの抜粋は、パラメータを含むストアド・プロシージャを示しています (ストアド・プロシージャの定義がコメントに含まれていることに注意)。この例では input/output パラメータを使用します。

```
Dim inParam As ADODB.Parameter
```

```
Dim inParam2 As ADODB.Parameter
```

```
Dim Connect As New ADODB.Connection
```

```
Connect.Open "Provider=ASAProv; UID=dba; PWD=sql; Data Source=ASA  
7.0 Sample"
```

```
Dim comm As New ADODB.Command
```

```
'create procedure foo (inout arg1 int, inout arg2 char(4))
```

```
'begin
```

```
' message arg1, arg2;
```

```
' set arg1=99;
```

```
' set arg2=88;
```

```
' message arg1, arg2;
```

```
' end;
```

```
Set inParam = comm.CreateParameter("inParam", adInteger,  
adParamInputOutput)
```

```
Set inParam2 = comm.CreateParameter("inParam2", adChar,  
adParamInputOutput, 4)
```

```
comm.Parameters.Append inParam
```

```
comm.Parameters.Append inParam2
```

```
inParam.Value = 4
```

```
inParam2.Value = "554"
```

```
comm.CommandText = "foo(?,?)"
```

```
' or the following works comm.CommandText = "{call foo(?,?)}"
```

```
comm.CommandType = adCmdText
```

```
comm.ActiveConnection = Connect
```

```

comm.Execute
Debug.Print inParam2.Value
Debug.Print inParam.Value
End Sub

```

このトピックの調査中、上記の例でコマンド・タイプを adCmdStoredProc とした場合に問題がありました。adCmdStoredProc コマンド・テキストが繰り返され、パラメータを使用すると SQLOLEDB プロバイダのみが構文を解釈可能というものです。このため、adCmdText コマンド・タイプを、コマンド・オブジェクトやパラメータと共に使用することをお勧めします。この問題は iAnywhere Solutions で確認中です。この問題は、今後ソフトウェアを改訂する際に対処する予定です。

ASAProv は、Command オブジェクトの Prepared プロパティをサポートします。アプリケーション内で 1 つのコマンドを複数回実行する場合は、Prepared プロパティを True に設定するとパフォーマンスが改良されます。実行前に Prepared プロパティを True に設定すると、CommandText がコンパイルされ、メモリに保存されます。最初は、準備が必要なため、実行の速度が低下しますが、コマンドが繰り返し実行されると、パフォーマンスは向上します。次の例では、Command オブジェクトを使用してパラメータが渡されます。Parameter の例では、Command オブジェクトの実行前にカーソル・タイプを設定するときに問題があるため、データを DataGrid に表示することはできません。

```

Set RS.cursorlocation = adUseServer
Set RS.cursorstype = adOpenStatic
Set RS = comm.execute

```

最初は、cursorstype が adOpenStatic に変わりますが、comm を実行すると、レコードセットはデフォルトのカーソル・タイプである adOpenForwardOnly にリセットされます。これは、前のレコードセットに設定されている属性が上書きされるためです。Command オブジェクトは、内部でデフォルトを使用してレコードセットを開きます。

このサンプルを使用した場合、Command オブジェクトで内部のレコードセットが使用されているため、ASAProv はデータベースとの接続を適切に切りません。同様に、ASAProv ではデータが常に表示されるわけではなく、グリッドにデータが表示されないことがあります。パラメータに問題があるため、このホワイトペーパーで使用しているビルドに関しては Microsoft の ODBC ブリッジ MSDASQL をお勧めします。7.0.2 EBF に、パラメータ化されたクエリのパラメータ問題への対処が含まれています。

Parameter プロジェクトに含まれているこのサンプル・コードでは、パラメータとしてアプリケーションに渡す Sales Rep ID が要求されます。

- 1 図 20 に示す [Fill List] ボタンをクリックします。

図 20 : Parameter プロジェクト

- 2 ドロップダウン・ボックスから id を選択して [Enter Parameter] をクリックすると、情報が表示されます。
- 3 ドロップダウン・ボックスから別の id を選択して [Enter Next Parameter] をクリックすると、次のレコードが下側の MSHDataGrid に表示されます。

Option Explicit

```
Dim Connect As New ADODB.Connection
Dim comm As New ADODB.Command
Dim newparameter As Parameter
Dim ConnString As String
Dim RS As New ADODB.Recordset
Dim RS2 As New ADODB.Recordset
Dim SQLstring As String
Dim id As String
Dim idnumber As Integer

Private Sub cmdFillList_Click()
    'ConnString = "Provider=ASAProv.70;Data Source=ASA 7.0 Sample;"
    & _ "User ID=dba;Password=sql"
    ConnString = "Provider=MSDASQL;Data Source=ASA 7.0 Sample;" & _
        "User ID=dba;Password=sql"
```

```

Connect.Open ConnString

SQLstring = "Select id from sales_order"
RS2.Open SQLstring, Connect, adOpenStatic
DataCombo1.ListField = "id"
Set DataCombo1.RowSource = RS2

End Sub

Private Sub DataCombo1_Change()
    id = DataCombo1.Text
    idnumber = CInt(id)
End Sub

Private Sub cmduseParameter_Click()

    'Put a SQL statement as a commandText
    comm.CommandText = "SELECT id, cust_id, " & _
        "order_date, region " & _
        "FROM sales_order " & _
        "WHERE id = ? "

    comm.CommandType = adCmdText

    comm.Prepared = True

    Set newparameter = comm.CreateParameter("id", adInteger, _
        adParamInput, 4, idnumber)
    comm.Parameters.Append newparameter

    comm.ActiveConnection = Connect

    RS.CursorLocation = adUseServer
    RS.CursorType = adOpenStatic
    Set RS = comm.Execute

    Set MSHFlexGrid1.DataSource = RS

```

```

End Sub

Private Sub cmdnextParameter_Click()
    comm("id") = idnumber
    idnumber = CInt(id)
    Set RS = comm.Execute
    Set MSHFlexGrid2.DataSource = RS
End Sub

Private Sub cmdDisconnect_click()
    comm.Parameters.Delete (0)
    Set DataCombo1.DataSource = Nothing
    Set MSHFlexGrid1.DataSource = Nothing
    Set DataCombo1.RowSource = Nothing
    RS.Close
    Set RS = Nothing
    Set comm.ActiveConnection = Nothing
    Set comm = Nothing
    RS2.Close
    Set RS2 = Nothing
    Connect.Close
    Set Connect = Nothing
End Sub

```

Blob

次の Blob サンプルは、blob がフォームにどのように表示されるかを示しています。

- 1 図 21 に示すように、[Fetch Image] の隣のテキストボックスに 1 ～ 3 の数字を入力します。



図 21：Blob のイメージ検索

- 2 [Fetch Image] をクリックします。右側に画像が表示されます。

- 3 [Fetch Image] ボックスに別の数字を入力して [Fetch Direct] をクリックします。[Display Image] ボタンを使用すると、コンピュータ内で jpeg またはビットマップを検索してピクチャ・ボックスにイメージを表示できます。zip ファイルに含まれている readme.txt に、ボタンについてと、このサンプルを Blob ディレクトリで実行する方法についての簡単な説明があります。図 21 は動作中のサンプルを示しています。

```
Dim Connect As ADODB.Connection
Dim RS As New ADODB.Recordset
Dim comm As New ADODB.Command

Private Sub cmdBrowse_Click()
    cmnDlg.Filter = "Bitmap Files (*.bmp)|*.bmp|JPEG File Interchange
Format (*.jpg)|*.jpg"
    cmnDlg.ShowOpen

    txtpath.Text = cmnDlg.FileName
End Sub

Private Sub cmdDisplay_Click()
    On Error Resume Next
    pctImage.Picture = LoadPicture(txtpath.Text)
End Sub

Private Sub cmdFetchDirect_Click()
    If Len(txtPictID.Text) > 0 Then
        RS.Open "select id, logo from logos where id =" &
CInt(txtPictID.Text), Connect, adOpenDynamic, adLockOptimistic

        If RS.BOF And RS.EOF Then
            MsgBox "no record founds"
        Else

            Text1.Text = RS.Fields(0).Value
            Set pctImage.DataSource = RS
            pctImage.DataField = RS.Fields.Item(1).Name

        End If
        RS.Close
        Set RS = Nothing
    End If
End Sub
```

```

Private Sub cmdGetPicture_Click()
    Dim bytChunk() As Byte

    If Len(txtPictID.Text) > 0 Then
        RS.Open "select id, logo from logos where id =" &
        CInt(txtPictID.Text), Connect, adOpenDynamic, adLockOptimistic

        If RS.BOF And RS.EOF Then
            MsgBox "no record founds"

        Else

            Text1.Text = RS.Fields(0).Value
            ReDim bytChunk(Len(RS.Fields(1).Value))

            bytChunk = RS.Fields(1).Value

            Open "C:\temp\image.bmp" For Binary As #1

            Put #1, , bytChunk()
            Close #1
            pctImage.Picture = LoadPicture("C:\temp\image.bmp")

        End If

        RS.Close

    End If

End Sub

'There are two methods to saving an image to the database.
'This method is best suited for the single-user environment.
Private Sub cmdSave_Click()
    On Error GoTo showError

    If Len(txtpath.Text) <> 0 Then

        comm.ActiveConnection = Connect
        comm.CommandText = " insert into logos(logo) SELECT
xp_read_file('" & txtpath.Text & "') "
        comm.Execute
        MsgBox "Successfully Inserted"
    End If

```

```

Exit Sub
showError:
    MsgBox "Failed to insert"

End Sub

Private Sub Command1_Click()

End Sub

'This is the second method for saving an image to the database.
'It is best suited for the multi-user client/server environment.
'(i.e. when the database does not reside on your machine, rather
on the server).
Private Sub cmdSavedDirect_Click()

Dim bytChunk() As Byte

Open txtpath.Text For Binary As #1
ReDim bytChunk(FileLen(txtpath.Text))

Get #1, , bytChunk()

RS.Open "logos", Connect, adOpenKeyset, adLockPessimistic,
adCmdTable

RS.AddNew
RS!logo.AppendChunk bytChunk
RS.Update
RS.Close
Set RS = Nothing

MsgBox "Successfully Inserted!"

Close #1

End Sub

Private Sub Form_Load()
Dim strErr As String
Dim objError As ADODB.Error

```

```

On Error GoTo UnableToStart_Err
Set Connect = New ADODB.Connection
Connect.Provider = "ASAProv.70"
Connect.ConnectionString = "uid=dba;pwd=sql;dbf=" & App.Path &
"\images.db;"

'Connect.Provider = "MSDASQL"
'Connect.ConnectionString = "dsn=vb;uid=dba;pwd=sql"

Connect.Open

If Connect.State <> adStateOpen Then
    MsgBox "Failed to connect"
End If

Exit Sub

UnableToStart_Err:

For Each objError In Connect.Errors
    strErr = strErr & "Description : " & objError.Description &
vbCrLf & vbCrLf & _
        "    SQLCODE : " & objError.NativeError & vbCrLf & vbCrLf & _
        "    SQLSTATE : " & objError.SQLState & vbCrLf
Next

MsgBox strErr, vbCritical + vbOKOnly, "Error Starting the
Database"

End Sub

Private Sub Form_Unload(Cancel As Integer)
On Error Resume Next
    Connect.Close
    Set Connect = Nothing

End Sub

```

Blob の例では、エラー処理をトラブルシューティング目的で使用しています。エラーが存在する場合は、問題をより確実に識別するためにエラー処理を使用することをお勧めします。

Property オブジェクト

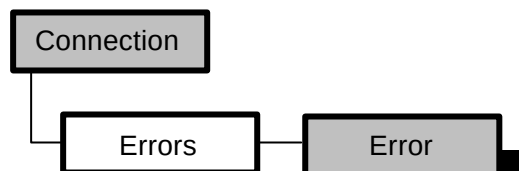
プロパティには、組み込みプロパティと動的プロパティの 2 つのタイプがあります。組み込みプロパティは新しいオブジェクトですぐに使用できますが、動的プロパティは OLE DB プロバイダが定義します。

ASAProv がサポートする動的プロパティには次のものが含まれます。

- Basecatalogname
- Basecolumnname
- Baseschemaname
- Basetablename
- Datetimeprecision
- Isautoincrement
- Iscasesensitive
- Issearchable
- Octetlength

ASAProv はサポートせず、Microsoft がサポートする唯一の動的プロパティは Keycolumn です。

Error オブジェクト



Error オブジェクトはそれぞれ Errors コレクション内に存在します。コードでエラーが発生すると、エラーに固有の Error オブジェクトが Errors コレクションに配置されます。Errors コレクションは、Connection オブジェクトのサブセットです。

ASAProv でサポートされているエラー・プロパティ

- Description 発生したエラーについてユーザに説明するテキスト。
- NativeError 使用しているプロバイダに固有のエラー番号。
- Error Number 発生したエラーの番号を表す長い整数 (ErrorValueEnum カラムに表示)。
- Source エラー発生の原因となったアプリケーションまたはオブジェクトの名前を特定。
- SQLState SQL 文を処理中にプロバイダから返されたエラー・コード (5 文字)。

On Error 文を使用した ADO エラーのトラップ

一般的な ADO エラーは、Connection オブジェクトや Recordset オブジェクトなどのオブジェクトを開き、データの更新や、プロバイダがサポートしていないメソッドまたはプロパティの呼び出しを行おうとしたときに発生します。ADO エラーはランタイム・エラーとしてプログラムに報告されます。エラートラップ・メカニズムである On Error 文を使用すると、エラーをトラップして処理できます。次の例 (OnError) は、ADO エラーのトラップを示しています。

```
' BeginErrorHandling
Private Sub Form_Load()
' Turn on error handling
On Error GoTo FormLoadError

'Open the database and the recordset for processing.
Dim Connect As New ADODB.Connection
Connect.ConnectionString = "Provider=ASAProv.70; Data Source = ASA
7.0 Sample"
Connect.Open
```

```

' The next line of code intentionally causes
' an error by trying to open a connection
' that has already been opened.
Connect.Open

' RS is a Public Recordset because it
' is defined WithEvents
Dim RS As New ADODB.Recordset
Dim SQLstring As String
SQLstring = "Select * FROM Product"
RS.Open SQLstring, Connect, adOpenStatic, adLockOptimistic,
adCmdText

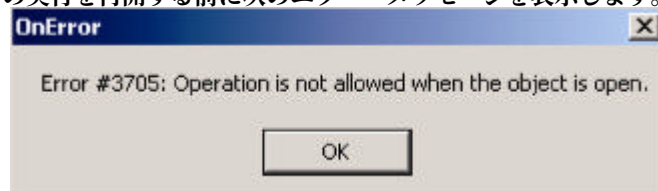
Exit Sub

' Error handler
FormLoadError:
    Dim strErr As String
    Select Case objError
        Case adErrObjectOpen
            strErr = "Error #" & objError.Number & ": " &
objError.Description & vbCrLf
            strErr = strErr & "Error reported by: " & objError.Source & vbCrLf
            MsgBox strErr
            Debug.Print strErr
            Err.Clear
            Resume Next
        ' If some other error occurs that
        ' has nothing to do with ADO, show
        ' the number and description and exit.
        Case Else
            strErr = "Error #" & Err.Number & ": " & Err.Description & vbCrLf
            MsgBox strErr
            Debug.Print strErr
            Unload Me
    End Select
End Sub

' EndErrorHandling

```

この Form_Load プロシージャは、同じ Connection オブジェクトを 2 回開こうとすることにより、意図的にエラーを生成します。Open メソッドを 2 回目に呼び出すと、エラー・ハンドラがアクティブになります。この場合、エラー・タイプは adErrObjectOpen であるため、エラー・ハンドラはプログラムの実行を再開する前に次のエラー・メッセージを表示します。



Visual Basic Err オブジェクトには、ごく最近のエラーに関する情報のみが含まれます。Connection オブジェクトの ADO Errors コレクションには、ごく最近の ADO 操作で発生したエラーに関して、エラー 1 つにつき 1 つずつ Error オブジェクトが含まれます。複数のエラーを処理するには、Visual Basic Err オブジェクトではなく ADO Errors コレクションを使用してください。このことをより詳細に理解するには、下記のプロバイダ・エラーに関する情報を参照してください。

プロバイダ・エラーへの Count プロパティの使用

Count プロパティが 1 以上の場合は、Error オブジェクトが Errors コレクションに追加されたため、確認が必要です。Errors コレクション内に存在する Error オブジェクトを見つけるには、コレクション内を検索し、Error オブジェクトのプロパティの形式で情報を取り出す必要があります。このエラーは、MsgBox <エラー文字列> を呼び出すことにより、テキスト・ボックスの形式で表示できます。以下は、存在しないテーブルを変更するコードを実行したときに発生するプロバイダ・エラーの ErrorCount の例です。

```
Private Sub Command1_Click()  
  
    Dim Connect As New ADODB.Connection  
    Dim comm As New ADODB.Command  
    Dim objError As ADODB.Error  
    Dim strErr As String  
  
    On Error Resume Next  
    Connect.ConnectionString = "Provider=ASAProv.70;Data Source =  
ASA 7.0 Sample"  
    Connect.Open  
    Set comm.ActiveConnection = Connect  
    comm.CommandText = "update dep set stuff = 'asfadsadfsadfsasd'  
where id =4"  
    comm.Execute  
  
    If Connect.Errors.Count > 0 Then  
        For Each objError In Connect.Errors  
            strErr = strErr & "Error #" & objError.Number & _  
                " " & objError.Description & vbCrLf & _  
                "NativeError: " & objError.NativeError & vbCrLf & _
```

```

        "SQLState: " & objError.SQLState & vbCrLf & _
        "Reported by: " & objError.Source
    Next
    MsgBox strErr
End If
End Sub

```

次のメッセージ・ボックスが表示されます。



Clear メソッドの使用

- Clear Errors コレクションに現存するエラーをすべて削除します。

ADO は、新しいエラーが発生したときに、Errors コレクション内のエラーを自動的にクリアします。このとき、コレクションは新しいエラー・オブジェクトで一杯になります。エラーによっては、コードの実行を止めないものの、エラー・オブジェクト (警告) として Errors コレクションに表示されるものがあります。次に挙げるメソッドは、Errors コレクションで Clear メソッドを呼び出した後に使用してください。

- RecordSet オブジェクトでの Resync メソッド、UpdateBatch メソッド、CancelBatch メソッド
- Connection オブジェクトでの Open メソッド
- RecordSet オブジェクトでの Filter プロパティの設定

下記のコード・サンプルは、上記のサンプルと同様に、Clear メソッドを使用して Errors コレクション内のエラーを削除する方法を示しています。ErrorClear サンプルには次のサンプル・コードが含まれています。まず、上記と同じエラーを含むメッセージ・ボックスが表示されます。[OK] をクリックした後に、コードで Clear メソッドが呼び出され、エラーを含まない新しいメッセージ・ボックスが表示されます。

```

Private Sub cmdErrorClear_Click()

    Dim Connect As New ADODB.Connection
    Dim comm As New ADODB.Command
    Dim objError As ADODB.Error

```

```

Dim strErr As String
Dim strErrNew As String

On Error Resume Next
Connect.ConnectionString = "Provider=ASAProv.70; Data Source =
ASA 7.0 Sample"
Connect.Open
Set comm.ActiveConnection = Connect
comm.CommandText = "update dep set stuff = 'asfdsadfsadfsasd'
where id =4"
comm.Execute

If Connect.Errors.Count > 0 Then
    For Each objError In Connect.Errors
        strErr = strErr & "Error #" & objError.Number & _
            " " & objError.Description & vbCrLf & _
            "NativeError: " & objError.NativeError & vbCrLf & _
            "SQLState: " & objError.SQLState & vbCrLf & _
            "Reported by: " & objError.Source
    Next
    MsgBox strErr
End If

Connect.Errors.Clear

strErrNew = strErrNew & "Error #" & objError.Number & _
    " " & objError.Description & vbCrLf & _
    "NativeError: " & objError.NativeError & vbCrLf & _
    "SQLState: " & objError.SQLState & vbCrLf & _
    "Reported by: " & objError.Source
MsgBox strErrNew

End Sub

```

上記と同じメッセージ・ボックスが表示された後に、このダイアログ・ボックスが表示されます。



これは、Errors コレクションがクリアされたことを示すダイアログ・ボックスです。

ユーザ定義エラーを ADO エラーと共に使用

Adaptive Server Anywhere のユーザ定義エラー機能を利用できます。この機能と ADO Error オブジェクトを使用すると、アプリケーションに固有のエラーを作成できます。次の例 (UserDefinedError プロジェクト) では、この機能を RAISERROR の形式で利用するストアード・プロシージャが作成されます。

たとえば、ADO がエラーをどのように発生させたかを示すエラーを自動的に表示するストアード・プロシージャをサンプル・データベースに追加できます。

- 1 Interactive SQL を開いてサンプル・データベースに接続します。
- 2 次の SQL 文を入力してデータベースにストアード・プロシージャを追加します。

```
CREATE PROCEDURE myerr()  
BEGIN  
    raiserror 27001 'Yikes!!!';  
END
```

- 3 asademo からの接続を切ります。

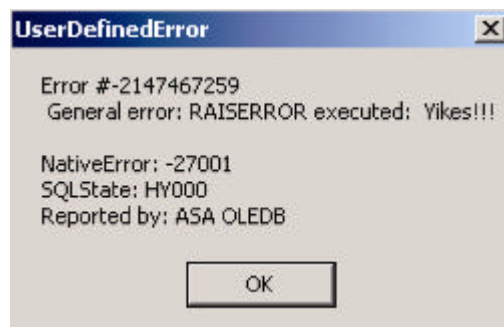
```
Private Sub cmdUserError_Click()  
  
    Dim Connect As New ADODB.Connection  
    Dim RS As New ADODB.Recordset  
    Dim strErr As String  
    Dim SQLstring As String  
  
    On Error Resume Next  
  
    'Connect to the database  
    Connect.Provider = "ASAProv.70"  
    Connect.ConnectionString = "data source = ASA 7.0 Sample"  
    Connect.Open  
  
    'Call the procedure myErr()  
    SQLstring = "call myErr()"  
    RS.Open SQLstring, Connect, adOpenDynamic, adLockReadOnly,  
    adCmdText  
  
    'Begin Error Handling  
    If Connect.Errors.Count > 0 Then  
        For Each objError In Connect.Errors  
            strErr = strErr & "Error #" & objError.Number & vbCrLf & _
```

```

" " & objError.Description & vbCrLf & _
"NativeError: " & objError.NativeError & vbCrLf & _
"SQLState: " & objError.SQLState & vbCrLf & _
"Reported by: " & objError.Source
Next
MsgBox strErr
End If
'End Error Handling
'Reset the recordset back to the start
RS.MoveFirst
End Sub

```

ここで発生させたエラーは、次のように表示されます。



これで、任意のストアド・プロシージャに RAISERROR 引数を含めることができ、ADO でこのストアド・プロシージャを呼び出したときに Error オブジェクトを使用してエラーが表示されるということがわかりました。

付録 A：プロジェクト

次の表に、このホワイトペーパーで使用したすべてのプロジェクトを示します。これらのプロジェクトは、zip ファイルから抽出すると、iAS_ADOProjects ディレクトリに配置されます。

プロジェクト名	ファイル名
RSConnect	RSConnect.vbp
DisplayData	DisplayData.vbp
Implicit RS	ImplicitRS.vbp
DataList	DataList.vbp
CommandObject	CommandObject.vbp
Parameter	Parameter.vbp
ProcParms	ProcParms.vbp
Add Bookmark	AddBookmark.vbp
NameCommand	NameCommand.vbp
AbsolutePosition	AbsolutePosition.vbp
Connection	Connection.vbp
DoubleRecordset	DoubleRecordset.vbp
Blobs	BLOB.vbp
DataControl	DataControl.vbp
UpdateRS	UpdateRS.vbp
ErrorCount	ErrorCount.vbp
On Error	OnError.vbp
ErrorClear	ErrorClear.vbp
UserDefinedError	UserDefinedError.vbp

付録 B：Adaptive Server Anywhere への接続

次の表は、Adaptive Server Anywhere の接続情報をまとめたものです。

Adaptive Server Anywhere のバージョン	このホワイトペーパーは ASA 7.0.2 ビルド 1505 に基づいていますが、それ以降のバージョン (バージョン 8 を含む) でも使用できます。
Adaptive Server Anywhere のデフォルトのユーザ ID とパスワード	デフォルトのユーザ ID : dba デフォルトのパスワード : = sql
接続文字列	一部のプロジェクトにはコード内にコメントが含まれており、そこでプロシージャを作成する必要があります。次の接続文字列を使用すると、ADO コントロールまたはオブジェクトを ASA 7.0 サンプル・データベースに接続できます。 "Provider=ASAProv; UID=dba; PWD=sql; Data Source=ASA 7.0 Sample"

付録 C：ASAProv プロバイダの使用

次の表では、ASAProv プロバイダと共に使用した場合のコントロールやロック・タイプの動作を説明します。

OLEDB プロバイダの名前	ASAProv.70
ADO レコードセットの 優先ロック・タイプ	AdLockReadOnly
DataGrid コントロール	DataGrid コントロールはブックマークを使用して結果セットを表示します。 ブックマークをサポートするカーソルを有効にしてください。
MSHFlexGrid コントロール	このコントロールは Visual Basic 6.0 以上で使用できます。 MSHFlexGrid コントロールは、グリッド内でのデータの更新や編集ができない読み取り専用のデータ・バインド・コントロールであるため、データの表示に使用します。
RecordCount プロパティ	RecordCount プロパティはレコードセットを開くときに使用するカーソル・タイプに依存します。 このプロパティは無効な場合があります。 ASAProv RS では、RecordSet.RecordCount プロパティを使用すると、カーソル・タイプごとの実際のレコード数が得られます。ただし adOpenForwardOnly は例外で、-1 という値になります。
レコードセット内の移動	Move、MoveFirst、MoveLast、MoveNext、MovePrevious の各メソッドを使用できるかどうかは、使用するカーソルのタイプによって異なります。
コマンド・タイプの指定	コマンド・タイプを指定するとパフォーマンスが改良されます。コマンド・タイプが不明の場合、正しいタイプが見つかるまでさまざまなタイプが実行されるため、パフォーマンスが低下します。 最も柔軟性の高いコマンド・タイプは adCmdText です。
Prepared プロパティ	アプリケーション内で 1 つのコマンドを複数回実行する場合は、Prepared プロパティを True に設定するとパフォーマンスが改良されます。

法的注意

Copyright(C) 2003 iAnywhere Solutions, Inc. All rights reserved.

Adaptive Server, Adaptive Server, iAnywhere, iAnywhere Solutions, SQL Anywhere, SQL Anywhere, Sybaseは、米国法人iAnywhere Solutions, Inc.または米国法人Sybase, Inc.とその系列会社の米国または日本における登録商標または商標です。その他の商標はすべて各社に帰属します。

Mobile Linkの技術には、Certicom, Inc.より供給を受けたコンポーネントが含まれています。これらのコンポーネントは特許によって保護されています。

本書に記載された情報、助言、推奨、ソフトウェア、文書、データ、サービス、ロゴ、商標、図版、テキスト、写真、およびその他の資料(これらすべてを"資料"と総称する)は、iAnywhere Solutions, Inc.とその供給元に帰属し、著作権や商標の法律および国際条約によって保護されています。また、これらの資料はいずれも、iAnywhere Solutions, Inc./Sybとその供給元の知的所有権の対象となるものであり、iAnywhere Solutions, Inc./Sybase, Inc.とその供給元がこれらの権利のすべてを保有するものとしします。

資料のいかなる部分も、iAnywhere Solutionsの知的所有権のライセンスを付与したり、既存のライセンス契約に修正を加えることを認めるものではないものとしします。

資料は無保証で提供されるものであり、いかなる保証も行われません。iAnywhere Solutionsは、資料に関するすべての陳述と保証を明示的に拒否します。これには、商業性、特定の目的への整合性、非侵害性の黙示的な保証を無制限に含みます。

iAnywhere Solutionsは、資料自体の、または資料が依拠していると思われる内容、結果、正確性、適時性、完全性に関して、いかなる理由であろうと保証や陳述を行いません。iAnywhere Solutionsは、資料が途切れていないこと、誤りがなく、いかなる欠陥も修正されていることに関して保証や陳述を行いません。ここでは、「iAnywhere Solutions」とは、iAnywhere Solutions, Inc.とその部門、子会社、継承者、および親会社と、その従業員、パートナー、社長、代理人、および代表者と、さらに資料を提供した第三者の情報元や提供者を表します。

＊ 本書は、米国iAnywhere Solutions, Inc.が作成・テストしたものを日本語に翻訳したものです。



アイエニウェア・ソリューションズ株式会社
<http://www.ianywhere.jp>