JRA-VAN 競馬ソフト作成体験教室

Lesson-7：JV-Data の内容を読み出す 応用編２（MDB の利用）

当コーナーでは、Microsoft Visual Basic 2019 Professional Edition(以下 VB 2019 と省略)で「JRA-VAN Data Lab.」サービス対応の競馬ソフトを作成していく過程をステップアップ形式で解説していきます。

前回までで、ダウンロードした JV-Data を ADO.NET(※)のデータセットに格納し、その内容を画面に表示するといった仕組みを実装しました。今回は、データセットの内容を MDB に反映する処理、MDB の内容をデータセットに取得する処理を実装し、MDB を利用した情報の表示を行ってみましょう。

※ADO.NET は、.NET Framework クラスライブラリに含まれるデータアクセスを行うためのクラスです。
詳細については、マイクロソフトの MSDN ライブラリなどをご参照ください。

【 今回の目標 】

ADO.NET を用いて、取得した JV-Data を MDB に格納し、また MDB から必要な情報を取り出して画面に表示する。

> 具体的には、データセットに格納した JV-Data の情報を ADO.NET によって MDB に反映します。同じく MDB の内容を取り出しデータセットに格納を行います。
> その上で、JV-Data をダウンロードして内容を表示する場合と、JV-Data をダウンロードせずに MDB から取得して表示する場合等の動作の違いを確認します。

【 やってみよう 】

① Lesson-6 までを実装したフォーム(frmMain)を含むプロジェクトを開きます。
（前回のレッスンの続きから開始するのであれば、この作業は必要ありません。）

② Form1.vb をコードエディタウィンドウで開き、btnGetJVData_Click メソッドに以下のコードを追加します。（灰色部分は前回コーディング済み）

```
Private Sub btnGetJVData_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnGetJVData.Click
    Dim lReturnCode As Long
    ''JV-Dataデータセット
    Dim jvdds As New JVDataDataSet
    ''レース詳細のテーブルアダプタ
    Dim raceTA As New JVDataDataSetTableAdapters.RACETableAdapter
    ''馬毎レース情報のテーブルアダプタ
    Dim umaRaceTA As New JVDataDataSetTableAdapters.UMA_RACETableAdapter
(後省略)
```

Point1

[ソースコード 009-01]

・Point1

データセットとデータベースの橋渡しの役割をするテーブルアダプタを、テーブル毎に定義します。

③ 処理開始時、表示ウィンドウの初期化処理の後に、処理中ボタンを無効にするコードと、MDB を制御するコードを以下のように追加します。(灰色部分は前回コーディング済み)

```
(前省略)

Dim RaceInfo As JV_RA_RACE         ''レース詳細情報構造体
Dim RaceUmaInfo As JV_SE_RACE_UMA  ''馬毎レース情報構造体

'表示ウィンドウの初期化
rtbData.Clear()

'ボタンの無効化
btnGetJVData.Enabled = False

'MDBのデータを削除                                          Point2
If MsgBox("MDBのデータを削除しますか？", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then
    If deleteDBTable(jvdds.RACE.TableName) < 0 Then Exit Try
    If deleteDBTable(jvdds.UMA_RACE.TableName) < 0 Then Exit Try
End If

'MDBからデータを取得                                        Point3
raceTA.Fill(jvdds.RACE)
umaRaceTA.Fill(jvdds.UMA_RACE)

(後省略)
```

[ソースコード 009-02]

・Point2

MDB のデータを削除するため、basADOUtility の deleteDBTable を呼び出します。引数には、String でテーブル名を指定します。

[ワンポイントメモ]

データセットにはテーブル情報やカラム情報が定義されており、それらのプロパティを参照することでテーブル名やカラム名を取得することができます。

・Point3

テーブルアダプタの Fill メソッドにより、MDB の内容をデータセットのデータテーブルに読み込みます。(MDB の対象テーブルの全データが読み込まれます)

④ JV-Link のデータ読み込み処理の実施を分岐するため、以下の位置に If 文を追加します。
（灰色部分は前回コーディング済み）

```
(前省略)

If MsgBox("JV-Linkでデータを取得しますか?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then
    '進捗表示初期設定
    tmrDownload.Enabled = False         ''タイマー停止
    prgDownload.Value = 0               ''JVDataダウンロード進捗
    prgJVRead.Value = 0                 ''JVData読み込み進捗

(後省略)
```

Point4

[ソースコード 009-03]

・Point4

VB2019 のエディタの機能により、インデントが自動的に修正されます。⑤の EndIf 挿入位置にご注意ください。

⑤ JV-Link のデータを MDB へ反映する処理、および④の If 文に対応する EndIf を追加します。（灰色部分は前回コーディング済み）

```
(前省略)

        'タイマー有効時は、無効化する
        If tmrDownload.Enabled = True Then
            tmrDownload.Enabled = False
            prgDownload.Value = prgDownload.Maximum
        End If

        'データセットの情報をDBへ反映
        If MsgBox("取得したデータをMDBへ反映しますか？", MsgBoxStyle.YesNo) =
MsgBoxResult.Yes Then
            'レース詳細を反映
            raceTA.Update(jvdds.RACE)
            '馬毎レース情報を反映
            umaRaceTA.Update(jvdds.UMA_RACE)
        End If

    End If

End If

(後省略)
```

Point5

[ソースコード 009-04]

・Point5

テーブルアダプタの Update メソッドにより、データセットの内容を MDB へ反映します。Update メソッドの引数には反映するデータテーブルを指定します。

⑥ 最後に後処理のコードを追加します。(灰色部分は前回コーディング済み)

```
(前省略)

Finally
    'JVLink終了処理
    lReturnCode = Me.AxJVLink1.JVClose()
    If lReturnCode <> 0 Then
        MsgBox("JVCloseエラー:" & lReturnCode)
    End If

    '後処理
    jvdds.Dispose() 'JV-Dataデータセットのリソース解放
    raceTA.Dispose() 'レース詳細テーブルアダプタのリソース解放
    umaRaceTA.Dispose() '馬毎レース情報テーブルアダプタのリソース開放

    'ボタンの有効化
    btnGetJVData.Enabled = True

End Try

'終了メッセージボックス表示
MsgBox("終了しました。")

(後省略)
```

Point6

[ソースコード 009-05]

・Point6

使用したテーブルアダプタクラスのリソースを解放します。

【 確認しよう 】

それでは、実際に動かしてみましょう。

1．JV-Link で JV-Data を読み込んで表示する。

① メニューから「デバッグ」→「デバッグ開始」を選択し、プログラムを実行します。

② JV-Link を使用するための「利用キー」を設定していない場合は、メインフォーム上のメニューから「設定」→「JV-Link の設定」を選択し、「利用キー」を設定します(詳細は Lesson-1 を参照)。

③ フォームの「データ取得」ボタンをクリックすると、「MDB のデータを削除しますか?」と確認があります。「はい(Y)」をクリックし、MDB のデータを削除します。

④ 次に、「JV-Link でデータを取得しますか?」と確認があります。「はい(Y)」をクリックします。

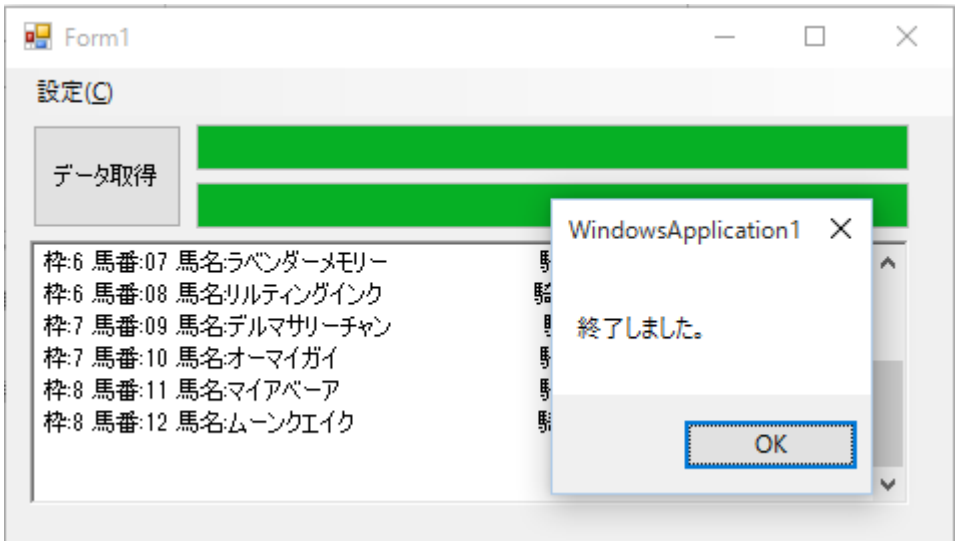
⑤ メッセージボックスがポップアップし、リターンコード、読み込みファイル数、ダウンロードファイル数、最新読み込みファイルのタイムスタンプが表示されます。
これらの数値は、実行するタイミングによって異なります。詳しくは、Lesson-2 をご覧下さい。

⑥ メッセージボックスを「OK」で閉じると、JV-Data のレース詳細、馬毎レース情報

の読み込みとデータセットへの格納が行われます。
JV-Data のダウンロード進捗状況や読み込み進捗状況は上下のプログレスバーで確認することができます。

⑦ JV-Data の読み込み、データセットへの格納が終わると、「取得したデータを MDB へ反映しますか？」と確認があります。「はい(Y)」をクリックし、MDB へデータを反映します。

⑧ MDB へのデータ反映後、データセットのレース詳細情報の「開催年、開催月日、競馬場名、開催回、開催日、レース番号、競走名略称 10 文字」が 1 行表示され、続いて馬毎レース情報の「枠番、馬番、馬名、騎手名」が表示されます。最後に処理の終了を示すメッセージボックスがポップアップします。



2．MDB から JV-Data を読み込んで表示する。

（1－⑧の続きとなります。終了メッセージボックスを閉じてから続きを行ってください）

⑨ フォームの「データ取得」ボタンをクリックします。「MDB のデータを削除しますか？」と確認がありますので、今度は「いいえ(N)」をクリックします。

[ワンポイントメモ]
このタイミングで、MDB のデータが読み込まれます。その為、次の⑩の状態になるまで多少時間がかかります。「はい(Y)」をクリックした場合 MDB のデータが空になるため、データを削除しない場合よりも比較的早く⑩の状態になります。(MDB のデータの状態、マシンスペック等によりそうならない場合もあります。)

⑩ 次に、「JV-Link でデータを取得しますか？」と確認がありますので、「いいえ(N)」をクリックします。

⑪ データセットのレース詳細情報の「開催年、開催月日、競馬場名、開催回、開催日、

レース番号、競走名略称 10 文字」が 1 行表示され、続いて馬毎レース情報の「枠番、馬番、馬名、騎手名」が表示されます。最後に処理の終了を示すメッセージボックスがポップアップします。

[ワンポイントメモ]
MsgBox による分岐を把握し、JV-Data をダウンロードして内容を表示する場合と、JV-Data をダウンロードせずに MDB から取得して表示する場合の動作の違いを確認してみましょう。

◆トピックス：「クエリが複雑すぎます。」エラーへの対処のヒント

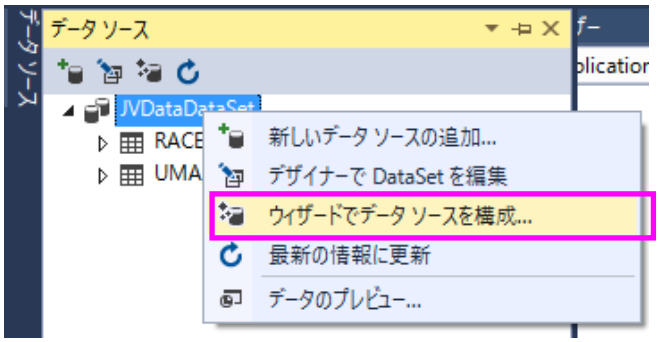
「取得したデータを MDB へ反映しますか？」で「はい(Y)」をクリックすると、「クエリが複雑すぎます。」とメッセージボックスがポップアップされることがあります。（MDB から情報を読み込み、さらに JV-Link でデータを取得した場合）

これは、テーブルアダプタの Update メソッドで使用される UPDATE クエリが複雑であるために発生しています。このクエリは、データソース追加時に自動的に作成されたものです。Update メソッドは必要に応じて INSERT クエリと UPDATE クエリのどちらかが実行されます。今回の Lesson-7 の内容では INSERT クエリは比較的単純であるため、新規追加では発生せずに更新の際にのみ発生します。

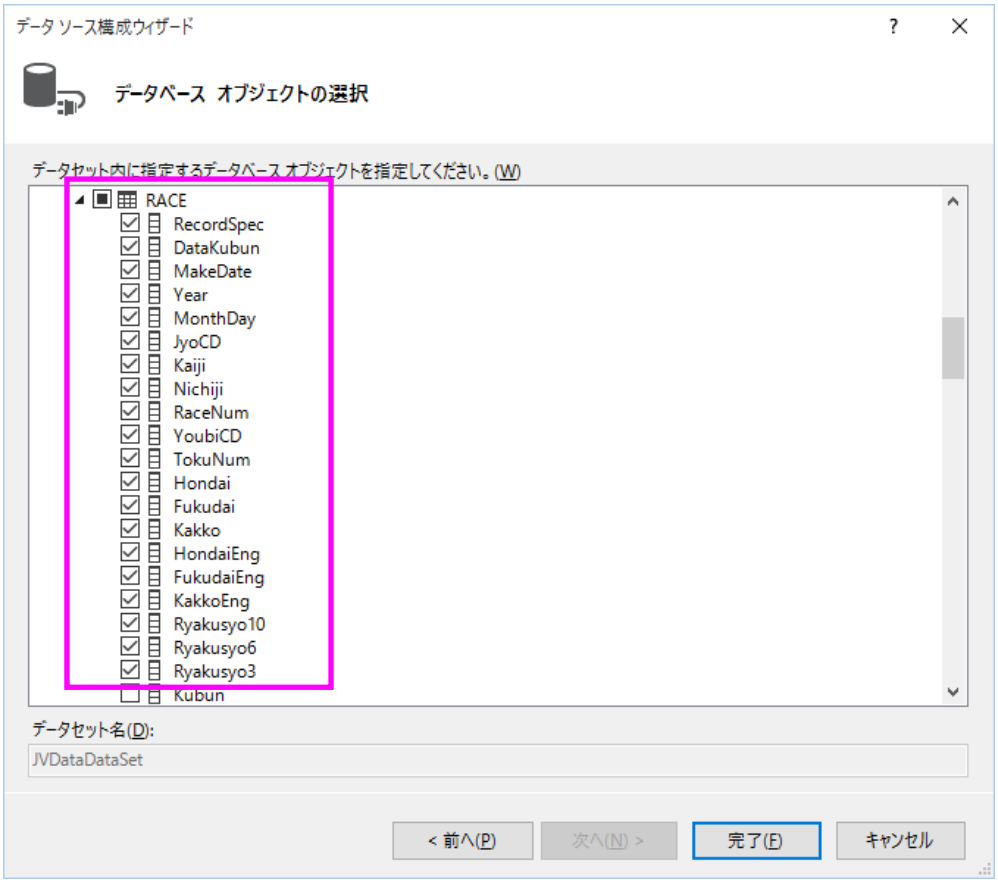
以下で、Lesson-7 におけるこのエラーへの対処例を 2 つ取り上げます。（本 Lesson を超える利用については、影響などを加味しご自身でご判断ください。また、本項は VB2019 の基本的な操作方法、及び SQL の書式について理解されている方向けの解説となります。）

対処例ａ．使用するカラムの数を減らす

① データソースの「JVDataDataSet」を右クリックし、「ウィザードでデータソースを構成」を選択します。

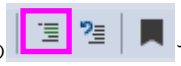


② 「データソース構成ウィザード」で RACE テーブルのチェックを一旦外し、カラムの RecordSpec~Ryakusyo3 までを一つずつチェックします。



チェック後「完了(F)」をクリックしてウィザードを完了します。
JVDataDataSet.xsd に変更が反映されますので、ファイルを保存してください。

③ basADOUtility.vb をコードエディタで開き、SetJVDataRaceDataSet でエラーが発生している行をすべてコメントアウトします。

※対象行を選択後、メニューバーアイコンの  で、まとめてコメントアウトできます。

```
(前省略)

dr(jvdds.RACE.Ryakusyo10Column) = raceDat.RaceInfo.Ryakusyo10      ''競走名略称１０字
dr(jvdds.RACE.Ryakusyo6Column) = raceDat.RaceInfo.Ryakusyo6        ''競走名略称６字
dr(jvdds.RACE.Ryakusyo3Column) = raceDat.RaceInfo.Ryakusyo3        ''競走名略称３字
'dr(jvdds.RACE.KubunColumn) = raceDat.RaceInfo.Kubun               ''競走名区分
'dr(jvdds.RACE.NkaiColumn) = raceDat.RaceInfo.Nkai                 ''重賞回次[第N回]
'dr(jvdds.RACE.GradeCDColumn) = raceDat.GradeCD                    ''グレードコード

 (中略)

'dr(jvdds.RACE.Syukaisu4Column) = raceDat.CornerInfo(3).Syukaisu   ''コーナー通過順4周回数
'dr(jvdds.RACE.Jyuni4Column) = raceDat.CornerInfo(3).Jyuni         ''コーナー通過順4各通過順位
'dr(jvdds.RACE.RecordUpKubunColumn) = raceDat.RecordUpKubun        ''レコード更新区分

'新しい行の場合、jvddsに行を追加
If bnr = True Then
    jvdds.RACE.Rows.Add(dr)
End If

 (後省略)
```

[ソースコード 009-06]

④ 同様に、SetJVDataRaceStructure の以下の不要個所をコメントアウトします。
（こちらはエラーにはなっていません。）

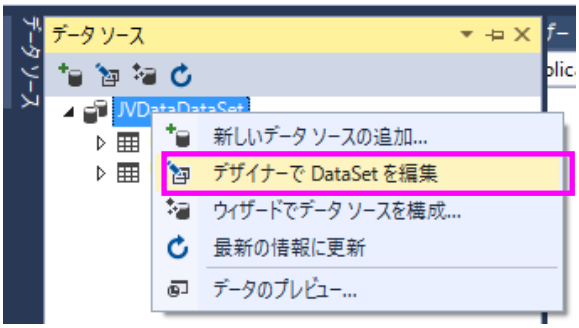
```
(前省略)

    raceDat.RaceInfo.Ryakusyo10 = dr(17)          ''競走名略称１０字
    raceDat.RaceInfo.Ryakusyo6 = dr(18)           ''競走名略称６字
    raceDat.RaceInfo.Ryakusyo3 = dr(19)           ''競走名略称３字
    'raceDat.RaceInfo.Kubun = dr(20)               ''競走名区分
    'raceDat.RaceInfo.Nkai = dr(21)                ''重賞回次[第N回]
    'raceDat.GradeCD = dr(22)                      ''グレードコード

 (中略)

    'raceDat.CornerInfo(3).Syukaisu = dr(107)     ''コーナー通過順4周回数
    'raceDat.CornerInfo(3).Jyuni = dr(108)        ''コーナー通過順4各通過順位
    'raceDat.RecordUpKubun = dr(109)              ''レコード更新区分

End Sub

 (後省略)
```

[ソースコード 009-07]

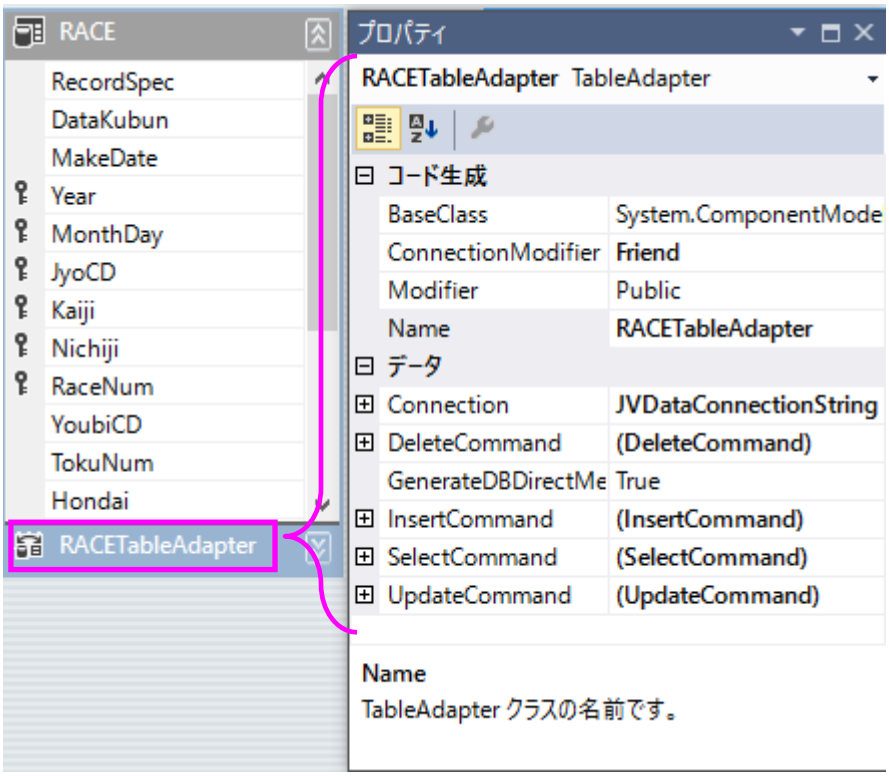
対処例 a は以上です。実際に動かしてエラーが発生しないことを確認してみましょう。

対処例ｂ．UPDATE クエリを編集する

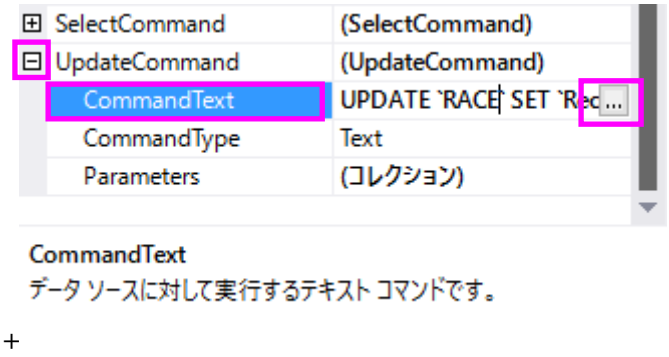
① データソースの「JVDataDataSet」を右クリックし、「デザイナで DataSet を編集」を選択します。



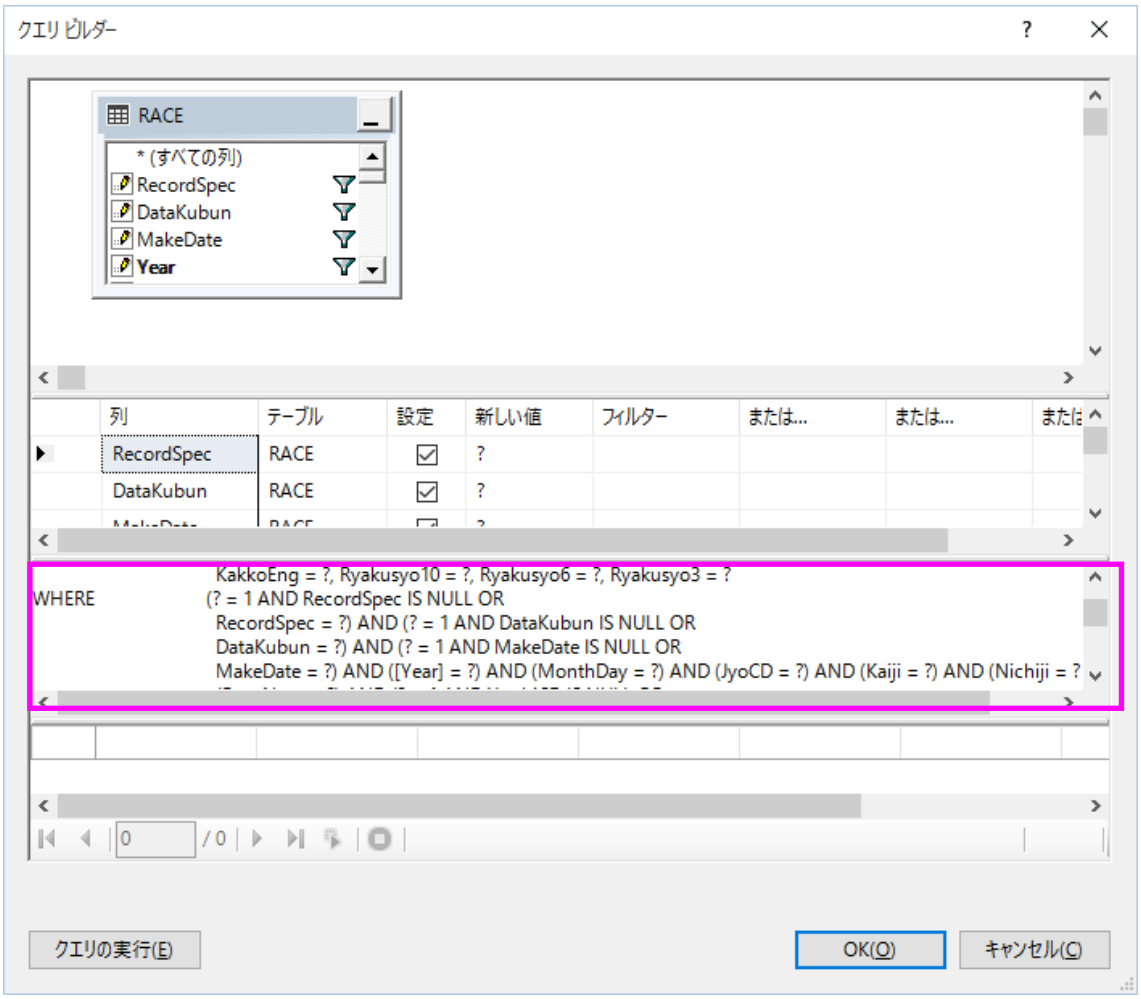
② デザイナの RACE テーブルの「RACETableAdapter」をクリックし、プロパティウィンドウに RACETableAdapter のプロパティを表示します。



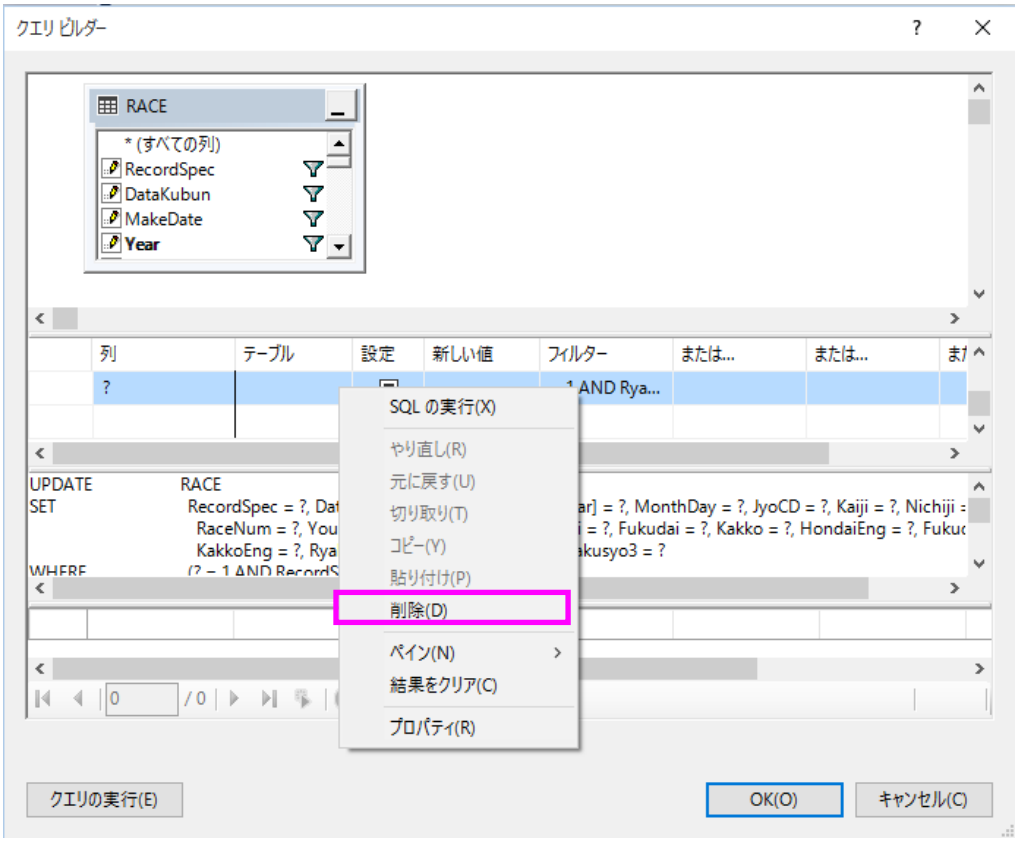
③ プロパティの「UpdateCommand」の左の「＋」をクリックして項目を展開し、CommandText をクリックします。右に「…」ボタンが表示されますので、このボタンをクリックします。



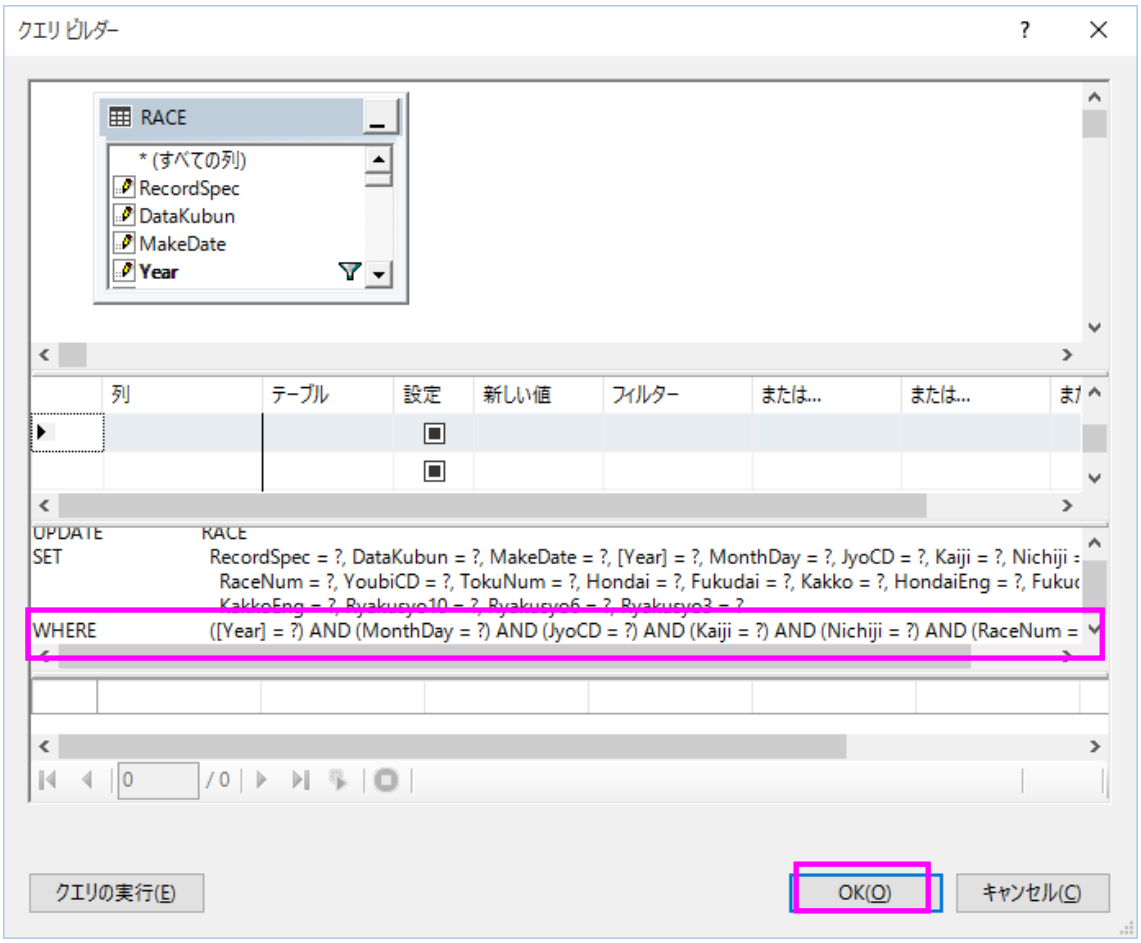
④ RACETableAdapter のクエリビルダーが表示されます。中央下の SQL で Where 句を確認すると、複雑な条件になっていることが分かります。
次に、中央の列のリストをスクロールし、列が「？」になっている行まで移動します。



⑤ 列が「？」の行をすべて選択し（先頭行をクリック後、最終行を Shift を押しながらクリック）、右クリック→「削除(D)」を選択してください。



⑥ 削除完了後、クエリビルダ中央下の SQL で Where 句を確認してください。内容が主キーのみの簡潔なものに変更されています。



「OK(O)」をクリックしてクエリビルダを終了します。
JVDataDataSet.xsd に変更が反映されますので、ファイルを保存してください。

対処例 b は以上です。こちらはソースの修正は不要です。
実際に動かしてエラーが発生しないことを確認してみましょう。