Visual FoxPro with SQL Server

Visual FoxPro から SQL サーバーにアクセスするには、リモートビュー、SQL パススルー、ADO、XML を使う方法があります。VFP Ver.8 以降では旧バージョンより、より簡単にリモートデータにアクセスするメカニズムの CursorAdapter がサポートされています。ここでは、汎用的な SQL パススルーを使う方法を解説します。
このドキュメントでは、Microsoft Visual FoxPro 7.0、Windows XP、SQL Server2000 を使うことを前提としています。又、使用するデータベースにはサンプルの pubs を使用します。FoxPro の各バージョンで、言語の拡張が行われています。詳しくはヘルプファイルを参照してください。
他のデータベースであっても、若干の設定の変更で動作を確認することができます。Linux 上の PostgreSql にもアクセス可能です。

注意：このドキュメントは無保証です。

１．ODBC の構成

サーバーに接続する為には、SQL サーバー用の ODBC ドライバーをインストールし、構成する必要があります。SQL サーバーに MS SQL Server を使用する場合は、MS-Office 等のソフトのインストール時に ODBC ドライバーがインストールされているので、新たに ODBC ドライバーをインストールする必要はありません。

ODBC データソースアドミニストレータの「システム DSN」タブをクリックして ODBC を構成してください。

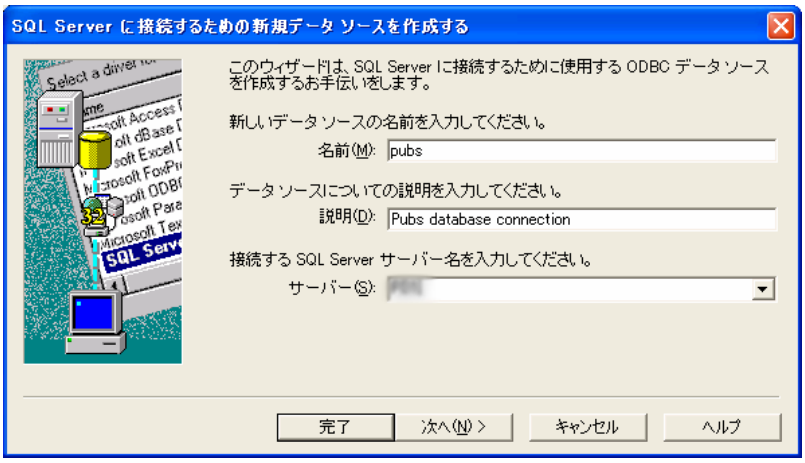


図１．pubs データベースに接続するために、データソースの名前
接続するサーバー名を入力します

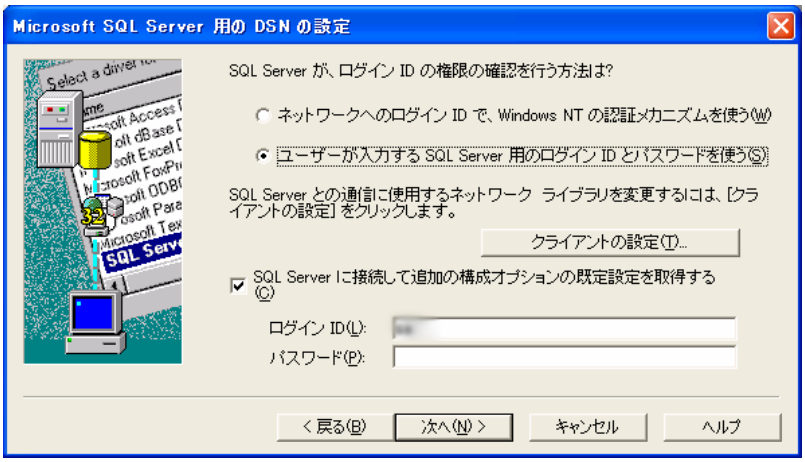


図２．pubs コネクションを SQL Server セキュリティを使うように
構成します。ログイン ＩＤ、パスワードを指定します

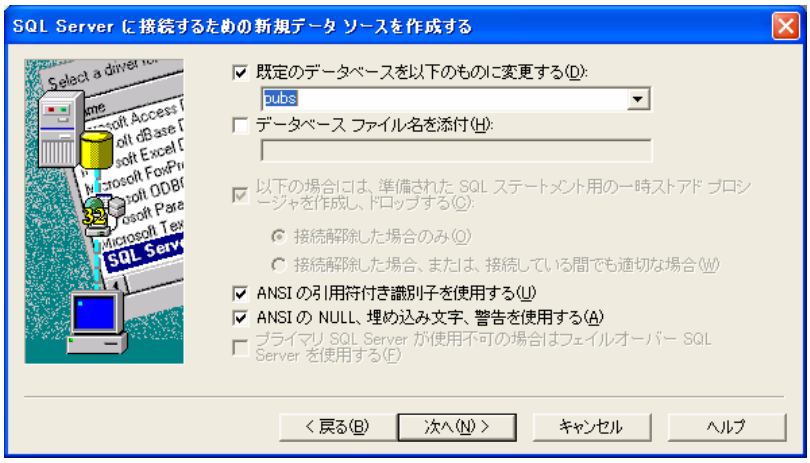


図３．Pubs DSN が　pubs データベースに接続するように構成します

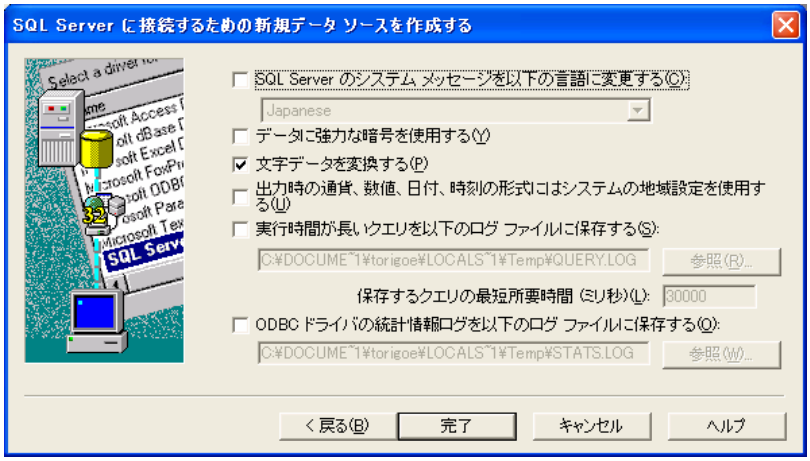


図４．システムメッセージ、文字データ変換の設定をします
（デフォルトのままで構いません）

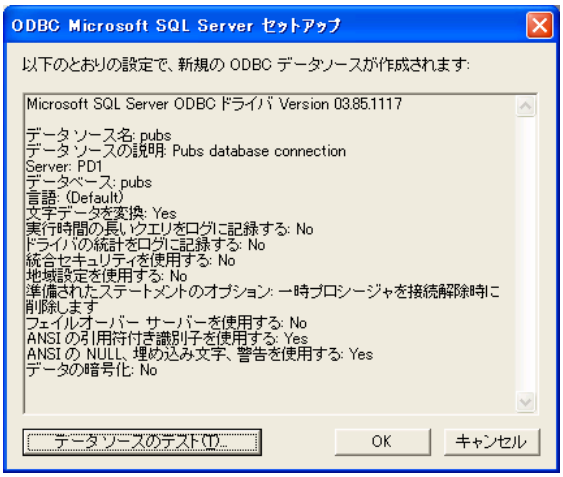


図５．「データソースのテスト」ボタンを
クリックして接続の確認をします

図６．「テストは無事に完了しました」の
表示を確認します

ODBC の定義を使う場合は、各クライアント PC に ODBC の設定を行う必要があります。実際のアプリでは、各クライアント PC での ODBC の設定作業を省略できる SQLSTRINGCONNECT()関数を使い文字列接続を行います。

２．リモートビュー

FoxPro を起動しテスト用に新規のプロジェクトを作成します。今回は、”Testsql”という名前のプロジェクトを作成します。コマンドウィンドウに

CD C:¥testsql

と入力しディレクトリーを作成したフォルダーに移動します。

２．１　データベースの作成

コマンドウィンドウに

CREATE DATABASE Pubs

と入力し、PUBS.DBC を作成します。

次に、プロジェクトマネージャーの「ADD...」ボタンをクリックし、作成した PUBS.DBC をプロジェクトに追加します。

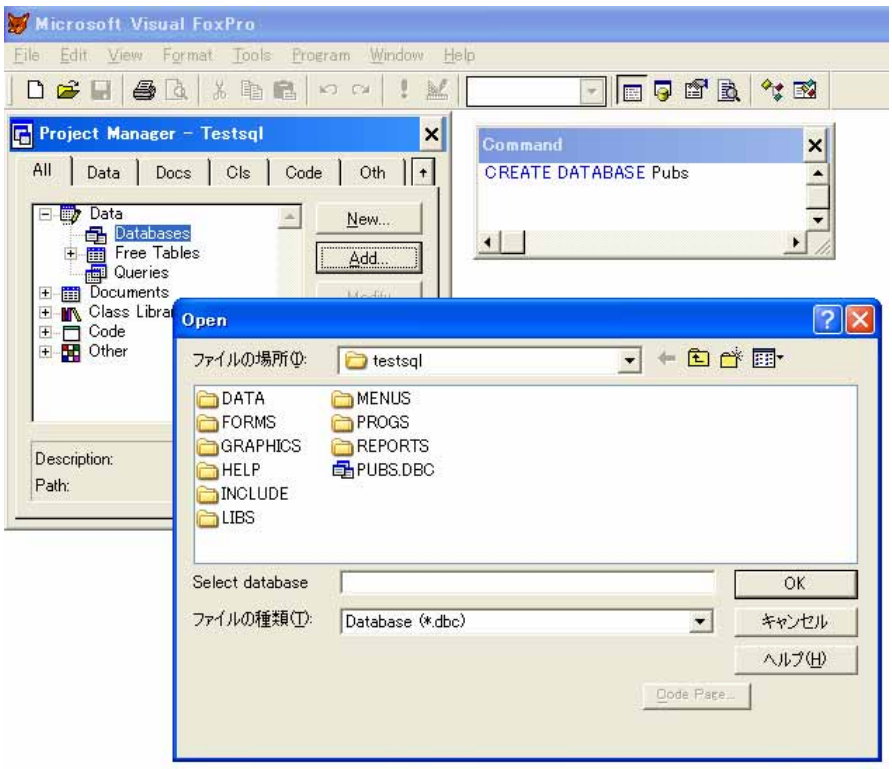


図７．カレントフォルダーに作成された　PUBS.DBC を選択し
プロジェクトに追加する

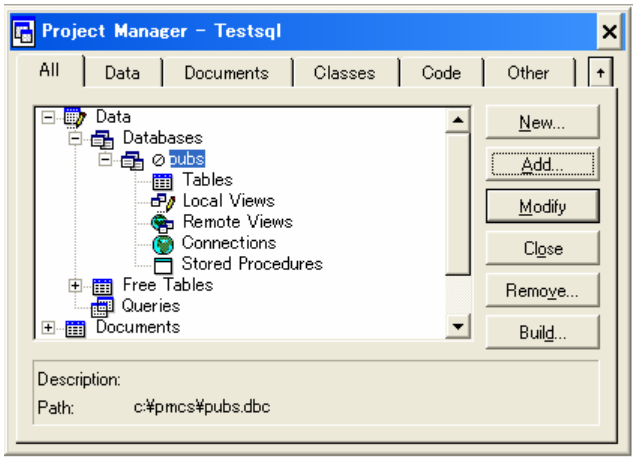


図８．プロジェクトに pubs が追加されます

２．２　リモートビューの作成

pubs データベースにある Titles テーブルを開くビュー Vtitles を作成します。
コマンドウィンドウに以下のコマンドを入力します。

```
OPEN DATABSE Pubs
CREATE SQL VIEW VTitles ;
    REMOTE CONNECTION Pubs ;
    AS SELECT * FROM Titles
```

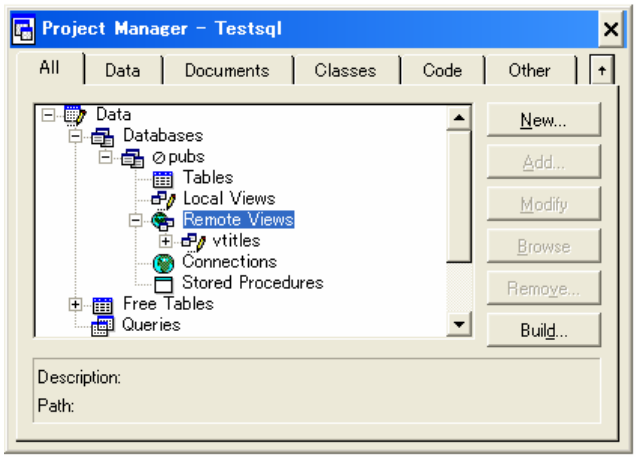


図９．Vtitles ビューが pubs の Remote View に追加されます

では、Vtitles ビューが正常に動作するか確認して見ましょう。
データベースをオープンし、Vtitles を開きます。コマンドウィンドウに以下のコマンドを入力します。

```
OPEN DATABASE Pubs
USE VTitles
BROWSE
```

Title_id	Title	Type	Pub_id	Price
BU1032	The Busy Executive's Database Guide	business	1389	19.9900
BU1111	Cooking with Computers: Surreptitious Balance Sheets	business	1389	11.9500
BU2075	You Can Combat Computer Stress!	business	0736	2.9900
BU7832	Straight Talk About Computers	business	1389	19.9900
MC2222	Silicon Valley Gastronomic Treats	mod_cook	0877	19.9900
MC3021	The Gourmet Microwave	mod_cook	0877	2.9900
MC3026	The Psychology of Computer Cooking	UNDECIDED	0877	
PC1035	But Is It User Friendly?	popular_comp	1389	22.9500
PC8888	Secrets of Silicon Valley	popular_comp	1389	20.0000
PC9999	Net Etiquette	popular_comp	1389	
PS1372	Computer Phobic AND Non-Phobic Individuals: Behavior Variations	psychology	0877	21.5900
PS2091	Is Anger the Enemy?	psychology	0736	10.9500
PS2106	Life Without Fear	psychology	0736	7.0000
PS3333	Prolonged Data Deprivation: Four Case Studies	psychology	0736	19.9900
PS7777	Emotional Security: A New Algorithm	psychology	0736	7.9900
TC3218	Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean	trad_cook	0877	20.9500
TC4203	Fifty Years in Buckingham Palace Kitchens	trad_cook	0877	11.9500
TC7777	Sushi, Anyone?	trad_cook	0877	14.9900

図１０．SQL サーバー上の pubs データベースの Titles テーブルがブラウズされます

Data Session ウィンドウで開いているテーブルやエイリアスを確認できます。Vtitles エイリアスは SQL サーバー pubs データベースの Titles テーブルから取得したデータのカーソルです。

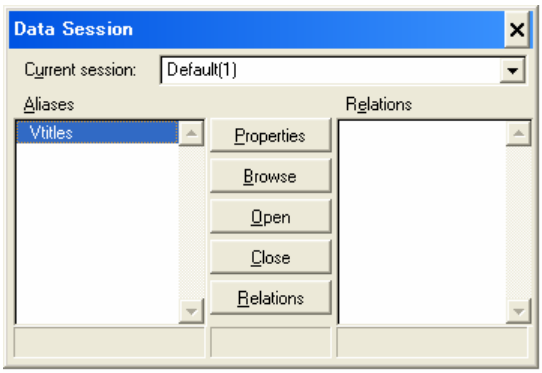


図１１．Data Session ウィンドウ

データのブラウズを確認したらブラウザを閉じ、コマンドウィンドウに

```
USE
```

コマンドを入力して、**Vtitles** ビューをクローズします。

２．３　接続の共有

Authors テーブルを表示するビュー Vauthors を作成します。コマンドウィンドウに以下のコマンドを入力します。

```
CREATE SQL VIEW VAuthors ;
    REMOTE CONNECTION Pubs ;
    AS SELECT * FROM Authors
```

では、ビューをオープンします、コマンドウィンドウに以下のコマンドを入力します。

```
USE VTitles IN 0
USE VAuthors IN 0
```

SQL サーバーとのコネクションハンドルを確認します。コマンドウィンドウに以下のコマンドを入力します。

```
? CURSORGETPROP(“ConnectHandle”, “VTitles”)
? CURSORGETPROP(“ConnectHandle”, “VAuthors”)
```

IDE の中に２つの異なった値が表示されます。”Vtitles”と”Vauthors”にそれぞれ個別の接続が作成されていることを示します。 **USE** コマンドでオープンしたビューを閉じます。
一つの接続には約２４KB のメモリーを消費します、又 接続はコストのかかる処理です。通常はリソースの無駄遣いを避けるために一つの接続を共有するようにします。(接続の共有は、トランザクション処理においても重要な意味を持ちます)
コマンドウィンドウに以下のコマンドを入力します。

```
CREATE CONNECTION PubsConnect DATASOURCE Pubs

CREATE SQL VIEW VTitles ;
    REMOTE CONNECTION PubsConnect SHARE ;
    AS SELECT * FROM Titles

DBSETPROP(“VTitles”, “View”, “SHARECONNECTION”, .T.)


CREATE SQL VIEW VAuthors ;
    REMOTE CONNECTION PubsConnect SHARE ;
    AS SELECT * FROM Authors

DBSETPROP(“VAuthors”, “View”, “SHARECONNECTION”, .T.)
```

CREATE CONNECTION()関数を入力すると、プロジェクトマネージャーの Connections に新たに接続の定義が作成されたことが分ります。

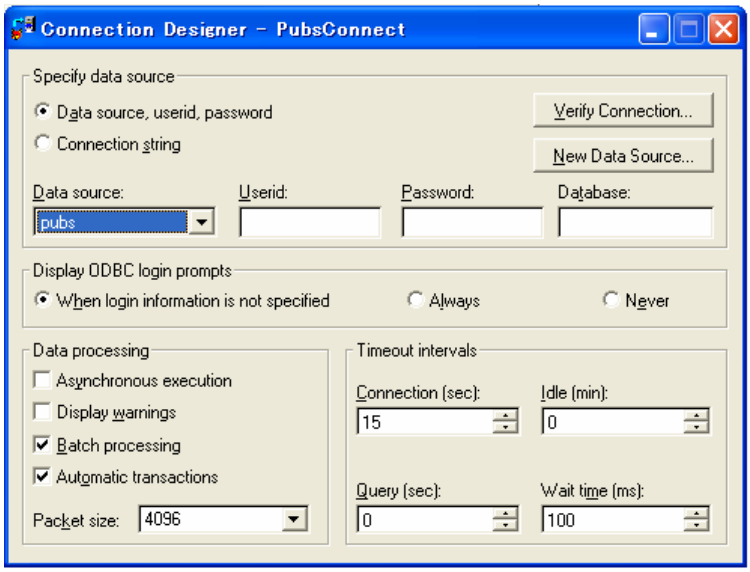


図１２．PubsConnect の定義内容の確認

ここまでの、処理は接続の共有を指定する DBSETPROP()関数を除けば、全てデザイナーを使ってビィジュアルに定義することが出来ます。

２．４　ＧＥＮＤＢＣの利用

DBC は定義の変更等よって、そのサイズが次第に肥大していきます。（追加・削除による未使用領域が増える為）

肥大した DBC を圧縮するには PACK DATABASE コマンドを使用します。又、GENDBC ユーティリティーを使用することもできます。

定義内容の確認と不足の事態でＤＢＣの破壊が起こった場合への対応のためにも、定期的に GENDBC を実行して DBC のジェネレート用のソースを作成するようにしましょう。

コマンドウィンドウに

```
DO HOME() + “tools¥gendbc¥gendbc.prg”
```

と入力し、GENDBC ユーティリティーを実行します。

GENDBC ユーティリティーがジェネレートした DBC のソースを開いて内容を見てみましょう。

この中で、CREATE CONNECTION に定義されている DBSETPROP()関数の

“Asynchronous” ～ “DATABSE”、CREATE SQL VIEW に定義されている”UpdateType” ～ “ShareConnection”は重要な意味を持ちますので、ヘルプファイルで意味を確認してください。

２．５　パラメータクエリーの使用

Vtitles ビューの WHERE 句にパラメータを受け取るように変更します。コマンドウィンドウに以下のコマンドを入力します。

```
CREATE SQL VIEW VTitles ;
```

```
REMOTE CONNECTION PubsConnect SHARE ;
AS SELECT * FROM Titles WHERE pub_id LIKE ?cPub_id

USE Vtitles IN 0
```

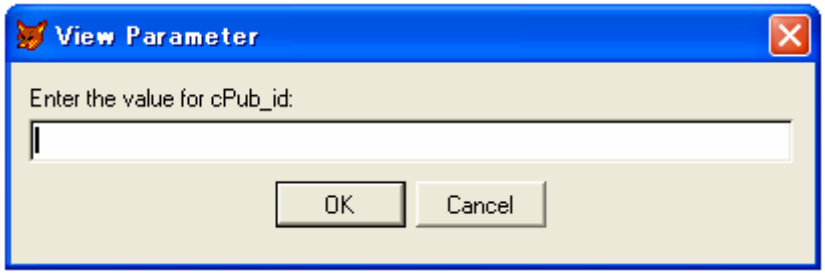


図１３　パラメータ値の入力を促すダイアログ

“0%” と入力します。(pub_id カラムの先頭が “0”のデータが表示されます)
WHERE 句には　WHERE　pub_id　LIKE　?出版社 ID のように書くことができますが、ダイアログは文字化けします。

プログラム上では、USE する前でパラメータの値を用意してビューをオープンします。以下のようになります。

```
cPub_id = “0%”
USE VTitles IN 0
BROWSE
USE
```

２．６　ＶＩＥＷを使ってデータを更新する

SQL サーバーから取得したカーソルは、ローカルでは更新可能ですが更新した内容はＳＱＬサーバには反映されません。

ローカル側での変更を SQL サーバーに反映させるようにするには

- VIEW デザイナーを使用して、ビジュアルに VIEW の定義を変更する
- DBSETPROP()関数を使用して、VIEW の定義を変更する

の、二通りの方法があります。

VIEW デザイナーを使用して Vtitles ビューが、更新結果を SQL サーバーに反映するように変更するには、プロジェクトマネージャーの Vtitles ビューを選択して Modify ボタンをクリックします。

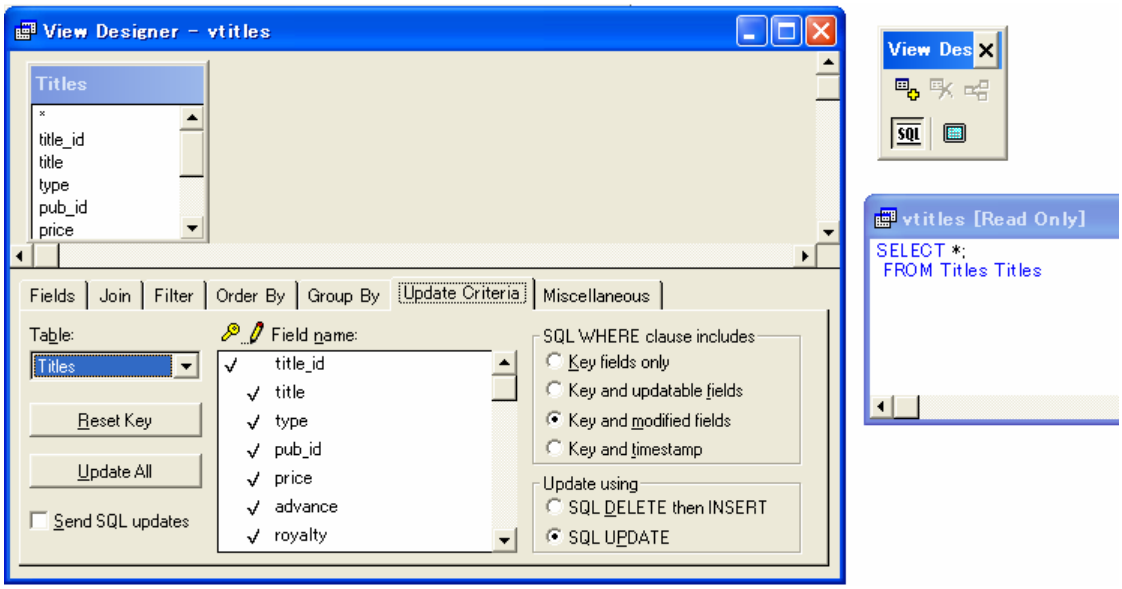


図１４．”Update Criteria”タブをクリックしビューを修正する

ＶＦＰヘルプファイル　”Update Criteria tab, View Designer” を参照

・最低限、一つ以上の一意なキーを指定する

・Update Criteria の “Send SQL Updates” にチェックを付ける

(“Send SQL Updates”をチェックしないと、サーバーのデータを更新しないことに注意)

DBSETPROP()関数を使用してビューを更新可能にするには

```
DBSETPROP(“VTitles.title_id”, “FIELD”, “KEYFIELD”, .T.)
DBSETPROP(“VTitles.title_id”, “FIELD”, “UPDATABLE”, .T.)
DBSETPROP(“VTitles”, “VIEW”, “SENDUPDATES”, .T.)
```

REQUERY()関数でサーバー上の最新の状態を取得することができます。
変更をコミットしないで、REQUERY()関数を実行すると未更新のバッファーがあることを示すエラーが表示されます。

又、View デザイナーの “Update Criteria” タブでは設定できないプロパティがあります、このプロパティの設定には DBSETPROP()関数を使います。

SQL サーバーの Datetime 型を VFP の Date 型に変換する方法。（View でしか出来ない）

```
DBSETPROP(“VTitles.Pubdate”,”FIELD”,”DATATYPE”, “D”)
```

３．ＳＱＬパススルー（ＳＰＴ）

３．１　ＳＱＬサーバーへの接続

FoxPro から SQL サーバーへ接続するには、SQLCONNECT() 又は SQLSTRINGCONNECT()関数を使用して接続を確立します。関数の戻り値には SQL サーバーとの接続のハンドラが設定されます。

```
hConn = SQLCONNECT("pubs", "sa", "")
```

又は、

```
lcDSNLess = "DRIVER=SQL Server;UID=sa;PWD=;Network=DBMSSOCN;
   DATABASE=Pubs;APP=Microsoft Open Database Connectivity;SERVER=PD1"
hConn = SQLSTRINGCONNECT(lcDSNLess)
```

オプション	
DSN	データソースの名前
Driver	利用する ODBC ドライバーの名前
Server	サーバーの名前
UID	ユーザーのログイン ID
PWD	ユーザーが指定したパスワード
Database	データベース名
APP	SQLServer ドライバを呼び出すアプリケーションの名前（省略可能）
WSID	ワークステーションの ID（省略可能）
Trusted Connection	Windows NT Domain 認証を使用するかどうかを指定

表１．接続文字列のオプション（MS-SQL サーバーの場合）

実際のアプリでは、各クライアント PC での ODBC の設定作業を省略するために、SQLSTRINGCONNECT()関数を使い文字列接続を行ないます。

SQL サーバーとの接続を開放するには、SQLDISCONNECT()関数を使います。パラメータには SQLCONNECT() 又は、SQLSTRINGCONNECT()関数で取得した接続ハンドラを指定します。複数の接続を作成している場合には、パラメータに０（ゼロ）を指定すると接続中の全ての接続を開放します。

```
SQLDISCONNECT(hConn)
```

又は、

```
SQLDISCONNECT(0)
```

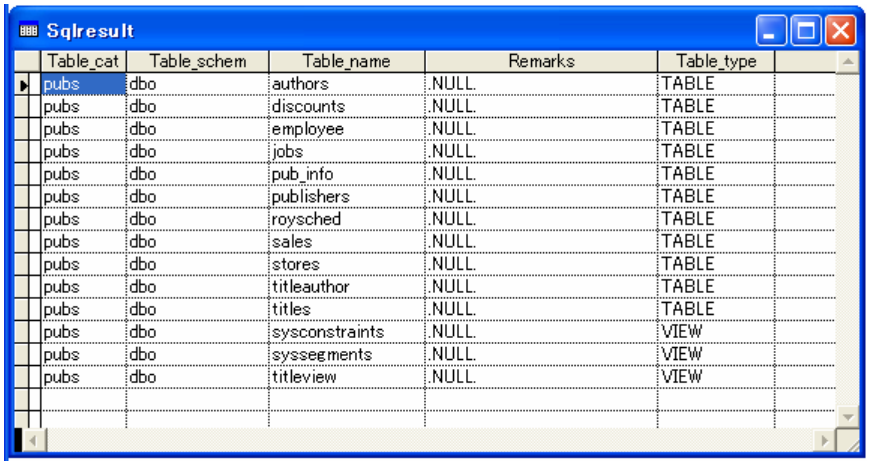
３．２　メタデータアクセス

FoxPro には接続中のデータベースの情報を取得する二つの関数があります。

SQLTABLES()関数はデータベースに存在するテーブル名とビュー名を返します。

再度、pubs データベースに接続し、コマンドウィンドウに以下のコマンドを入力します。これ以降、不要なカーソルは適時 USE コマンドを使用してクローズしてください。

```
SQLTABLES(hConn, "'TABLE", "VIEW")
BROWSE
```



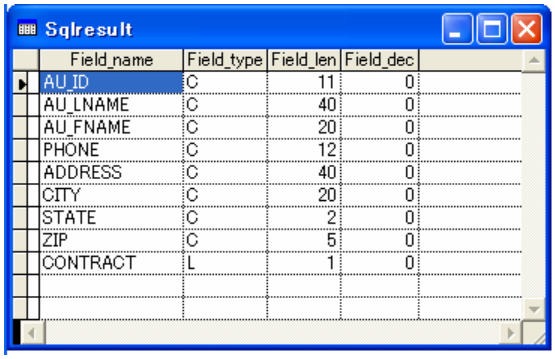
Table_cat	Table_schem	Table_name	Remarks	Table_type
pubs	dbo	authors	.NULL.	TABLE
pubs	dbo	discounts	.NULL.	TABLE
pubs	dbo	employee	.NULL.	TABLE
pubs	dbo	jobs	.NULL.	TABLE
pubs	dbo	pub_info	.NULL.	TABLE
pubs	dbo	publishers	.NULL.	TABLE
pubs	dbo	roysched	.NULL.	TABLE
pubs	dbo	sales	.NULL.	TABLE
pubs	dbo	stores	.NULL.	TABLE
pubs	dbo	titleauthor	.NULL.	TABLE
pubs	dbo	titles	.NULL.	TABLE
pubs	dbo	sysconstraints	.NULL.	VIEW
pubs	dbo	syssegments	.NULL.	VIEW
pubs	dbo	titleview	.NULL.	VIEW

図１５．Pubs データベースに対する SQLTABLES()関数の結果

もう一つの関数は、指定したテーブルのフィールド情報を返す SQLCOLUMS()関数です。この関数は、SQL サーバーのネイティブなフィールド情報を返すように指定するパラメータと、FoxPro のデータに変換される時にどのデータ型に変換されるかを返すパラメータがあります。

コマンドウィンドウに以下のコマンドを入力します。

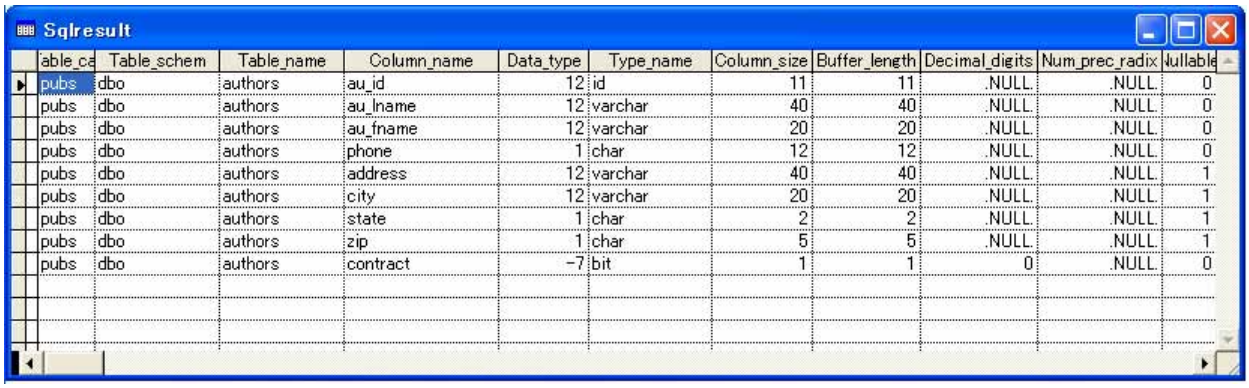
```
SQLCOLUMNS(hConn, "authors", "FOXPRO")
BROWSE
```



Field_name	Field_type	Field_len	Field_dec
AU_ID	C	11	0
AU_LNAME	C	40	0
AU_FNAME	C	20	0
PHONE	C	12	0
ADDRESS	C	40	0
CITY	C	20	0
STATE	C	2	0
ZIP	C	5	0
CONTRACT	L	1	0

図１６．FOXPRO オプションを指定して取得した SQLCOLUMNS()関数の結果

```
SQLCOLUMNS(hConn, “authors”, “NATIVE”)
BROWSE
```



able_ca	Table_schem	Table_name	Column_name	Data_type	Type_name	Column_size	Buffer_length	Decimal_digits	Num_prec_radix	Nullable
pubs	dbo	authors	au_id	12	id	11	11	NULL	NULL	0
pubs	dbo	authors	au_lname	12	varchar	40	40	NULL	NULL	0
pubs	dbo	authors	au_fname	12	varchar	20	20	NULL	NULL	0
pubs	dbo	authors	phone	1	char	12	12	NULL	NULL	0
pubs	dbo	authors	address	12	varchar	40	40	NULL	NULL	1
pubs	dbo	authors	city	12	varchar	20	20	NULL	NULL	1
pubs	dbo	authors	state	1	char	2	2	NULL	NULL	1
pubs	dbo	authors	zip	1	char	5	5	NULL	NULL	1
pubs	dbo	authors	contract	-7	bit	1	1	0	NULL	0

図１７．NATIVE オプションを指定して取得した SQLCOLUMNS()関数の結果

MS-SQL Server **のように** FoxPro **のデータ型の対応が公開されている場合は問題ありませんが、他のデータベースを使用する場合には** SQLCOLUMNS()関数を使ってサーバーのデータ型が FoxPro のどのデータ型に変換されるか調べる必要があります。
又、フィールド名を取得することによって後で解説する CURSORSETPROP()関数を使ってカーソルを更新可能にする UpdateNameList をアプリで作成することも可能になります。

３．３　SQLEXEC()関数で取得するカーソルに名前を付ける

SQL サーバーからデータを取得するには、SQLEXEC()関数を使用します。
Authors テーブルから全データを取得するには、以下のようになります。

```
lnResult = SQLEXEC(hConn, “SELECT * FROM Authors”)
```

データが正常に取得できた場合 SQLEXEC()関数は１以上の値を返します。Data Session ウィンドウを開くと、取得したデータには Sqlresult という名前のエイリアスが付きます。”Sqlresult”はカーソル名を省略した場合に、FoxPro が自動的に付与する名前です。これでは、アプリで使用するときに不便なのでカーソル名を指定してみましょう。コマンドウィンドウに以下を入力します。

```
lnResult = SQLEXEC(hConn, “SELECT * FROM Authors”, “curAuthors”)
```

再度、Data Session ウィンドウでエイリアス名を確認してください。”curAuthors”が作成されています。

SQLEXEC()関数は、複数の SQL 文を発行することもできます。コマンドウィンドウに以下を入力します。

```
lnResult = SQLEXEC(hConn, “SELECT * FROM Authors;
                           SELECT * FROM Titles”, “curAuthors”)
```

この場合、カーソル名は最初の SQL 文の結果には”curAuthors”、二番目の SQL 文の結果には”curAuthors1”というエイリアス名が付き、二番目の SQL 文の結果がカレントになります。戻り値は作成されたカーソルの数、SQL 文のどれかにエラーがあれば負の値が戻り値にセットされ、カーソルは作成されません。

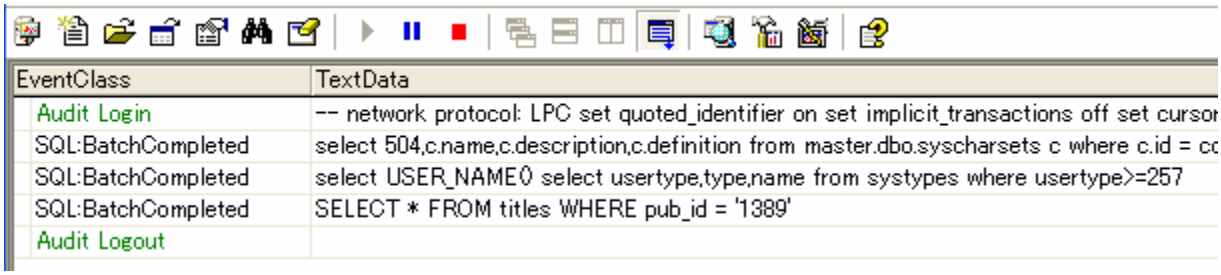
３．４　パラメータクエリーの利用

パラメータクエリーはパラメータに異なる値をセットして繰り返し発行する場合に、パラメータ無しの場合に比べより早く実行できます。

パラメータを使用した SQL 文を発行すると SQL サーバーはどのような動作をするか見てみましょう。MS-SQL サーバーの管理ツールに付属するプロファイラを起動してください。

FoxPro のコマンドウィンドウに以下を入力します。

```
cSql = “SELECT * FROM titles WHERE pub_id = ‘1389’ “
lnResult = SQLEXEC(hConn, cSql, “curTitles”)
```

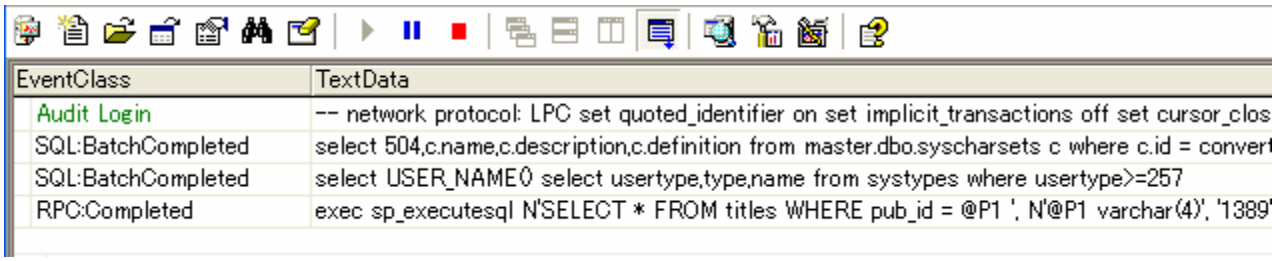


EventClass	TextData
Audit Login	-- network protocol: LPC set quoted_identifier on set implicit_transactions off set cursor
SQL:BatchCompleted	select 504,c.name,c.description,c.definition from master.dbo.syscharsets c where c.id = cc
SQL:BatchCompleted	select USER_NAME() select usertype,type,name from systypes where usertype>=257
SQL:BatchCompleted	SELECT * FROM titles WHERE pub_id = '1389'
Audit Logout	

図１８．パラメータ無しで実行した場合のプロファイラの表示

次に、以下のコマンドを実行して見ましょう。

```
cpub_id = “1389”
cSql = “SELECT * FROM titles WHERE pub_id = ?cpub_id”
lnResult = SQLEXEC(hConn, cSql, “curPubs”)
```



EventClass	TextData
Audit Login	-- network protocol: LPC set quoted_identifier on set implicit_transactions off set cursor_clos
SQL:BatchCompleted	select 504,c.name,c.description,c.definition from master.dbo.syscharsets c where c.id = convert
SQL:BatchCompleted	select USER_NAME() select usertype,type,name from systypes where usertype>=257
RPC:Completed	exec sp_executesql N'SELECT * FROM titles WHERE pub_id = @P1 ', N'@P1 varchar(4)', '1389'

図１９．パラメータ付で実行した場合のプロファイラの表示

同じ結果が返ってきますが、SQL サーバーの動作は違っています。通常のクエリーはサーバーに送られると　構文の解析 -> オプティマイズ -> コンパイル -> 実行で完了します。

しかし、パラメータクエリーは再利用されることを前提にします。１回目の実行の際に、実行プラン作成されサーバー上に保存されます。

２回目以降の実行は、SQL サーバーは実行プランを再使用できるうえ、オーバーヘッドを減らすことができます。パラメータクエリーはコストが掛かる処理なので　再利用されないクエリー をパラメータ化しても意味がありません。

３．５　エラー処理

SPT を実行し関数の戻り値が負の場合は適切なエラー処理が必要になります。SQL サーバーから返されるエラーを取得するには AERROR()関数を使います。

“Authors”を”Author”と変えて、以下のコマンドを実行してみます。

```
lnResult = SQLEXEC(hConn, “SELECT * FROM Author”, “curAuthors”)
```

エラーがあるので、SQLEXEC()関数の戻り値は負になります。エラーを取得します。

```
= AERROR(gaArray)
LIST MEMORY LIKE gaArray
```

```
GAARRAY          Pub        A
  (    1,    1)                N   1526                (      1526.00000000)
  (    1,    2)                C   "Connectivity error: [Microsoft][ODBC SQL Server Driver][SQL Server]オブジェクト名 'AUTHOR
                                 "
  (    1,    3)                C   "[Microsoft][ODBC SQL Server Driver][SQL Server]オブジェクト名 'AUTHOR' は無効です。"
  (    1,    4)                C   "S0002"
  (    1,    5)                N   208                 (       208.00000000)
  (    1,    6)                N   2                   (         2.00000000)
  (    1,    7)                C   .NULL.
```

図２０．AERROR()関数で取得したエラー内容

要素	データ型	内　　容	説　　明
1	Numeric	1526	常に、１５２６
2	Character	エラーメッセージテキスト	FoxPro エラーメッセージ
3	Character	ODBC エラーメッセ維持テキスト	カラム２と同じ。但し、情報は少ない。
4	Character	ODBC SQL	エラーが ODBC に起因するなら、ODBC のエラー番号。 ODBC プログラマーズガイドを参照。
5	Numeric	ODBC データーソースエラー番号	エラーが SQL サーバーに起因するなら、SQL サーバーのエラー番号。 MS-SQL サーバーなら SQL Server Books Online を参照
6	Numeric	ODBC コネクションハンドル	エラーが発生したコネクションハンドルを表す。
7		NULL	常に、NULL

表２．AERROR()関数が取得する ODBC エラーの要素

３．６　カーソルを更新可能にする

デフォルトでは SPT の結果セットはローカルでは更新可能ですが、更新内容はサーバーに反映されません。これは、Non-Updatable View と同じです。更新内容をサーバーに反映するようにはCURSORSETPROP()関数を使います。（DBSETPROP()関数ではない）

```
lnResult = SQLExec(hConn, “SELECT * FROM authors”, “authors”)
CURSORSETPROP(“TABLES”, “dbo.authors”, “authors”)
CURSORSETPROP(“KeyFieldList”, “au_id”, “authors”)
CURSORSETPROP(“UpdatableFieldList”, “au_lname, au_fname, phone, address, city, ;
                    state, zip, contract”, “authors”)
CURSORSETPROP(“UpdateNameList”, “au_id dbo.authors.au_id”, “authors”)
CURSORSETPROP(“UpdateNameList”,”au_lname dbo.authors.au_lname”, “authors”)
CURSORSETPROP(“UpdateNameList”,”au_fname dbo.authors.au_fname”, “authors”)
CURSORSETPROP(“UpdateNameList”,”phone dbo.authors.phone”, “authors”)
CURSORSETPROP(“UpdateNameList”,”address dbo.authors.address”, “authors”)
CURSORSETPROP(“UpdateNameList”,”city dbo.authors.city”, “authors”)
CURSORSETPROP(“UpdateNameList”,”state dbo.authors.state”, “authors”)
CURSORSETPROP(“UpdateNameList”,”zip dbo.authors.zip”, “authors”)
CURSORSETPROP(“UpdateNameList”,”contract dbo.authors.contract”, “authors”)
CURSORSETPROP(“SendUpdates”, .T., “authors”)
```

UpdateNameList プロパティは以下のようにも書けます。

```
CURSORSETPROP(“UpdateNameList”, ;
    “au_id dbo.authors.au_id, ;
    au_lname dbo.authors.au_lname, ;
    au_fname dbo.authors.au_fname, ;
    phone dbo.authors.phone, ;
    address dbo.authors.address, ;
    city dbo.authors.city, ;
    state dbo.authors.state, ;
    zip dbo.authors.zip, ;
    contract dbo.authors.contract”, “authors”)
```

更新するフィールドが少ない場合はコードを書いても大した手間ではありませんが、フィールド数が多い場合には UpdateNameList をアプリで作成する方法を取った方が良いでしょう。UpdateNameList の作成のためのフィールド名の取得に、作成されたカーソルのフィールド名をFIELD()関数を使って取得する方法を思いつくでしょうが、複数のテーブルを JOIN している場合は更新テーブルに含まれるフィールド名であるかどうかを判断できません。これを解決するには、メタデータアクセスで説明した SQLCOLUMNS()関数を使用して更新するテーブルのフィールド名を取得して UpdateNameList をアプリで作成する方法を取った方が良いでしょう。

３．７　同期・非同期

デフォルトでは、Visual FoxPro SQL 関数は、同期・バッチモードとして処理されます。 関数の呼出しが完了するまで、FoxPro はアプリケーションに制御を返しません。Visual FoxPro でインタラクティブな処理をする場合は同期処理を使います。非同期処理は、関数の呼出しの完了を待たずに FoxPro はアプリケーションに制御を返します。例えば、非同期で関数を処理しているとき、アプリケーションは実行しているステートメントの進捗を示すためのプウログレスバーを表示したり、マウス・ポインタを変化させるループ処理を行うことができます。

同期・非同期モードは通常バッチモードと組み合わせて使用します。このモーそのセットにはCURSORSETPROP()関数を使用します。

・同期・バッチモードの場合

```
SQLSETPROP(hConn, “Asynchronous”, .F.)
SQLSETPROP(hConn, “BatchMode”, .T.)
SQLEXEC(hConn, “SELECT * FROM authors ;
                SELECT * FROM titles ;
                SELECT * FROM roysched ;
                SELECT * FROM titleauthor”, “ITEM”)
```

この処理を実行すると、カーソル名(Item, Item1, Item2, Item3)が作成され FoxPro に制御が戻ります。

・同期・非バッチモードの場合

```
SQLSETPROP(hConn, “Asynchronous”, .F.)
SQLSETPROP(hConn, “BatchMode”, .F.)
SQLEXEC(hConn, “SELECT * FROM authors ;
                SELECT * FROM titles ;
                SELECT * FROM roysched ;
                SELECT * FROM titleauthor”, “ITEM”)
```

この処理を実行すると、カーソル名 ”Item ”が一つだけ返されます。返された内容は最初の SELECT 文の “ authors ”テーブルです。次の SELECT 文の結果を得るには、SQLMORERESULTS()関数を使います。

```
SQLMORERESULTS(hConn)
```

このとき、カーソル名を指定すると、既に作成されている同名のカーソルを上書きします。

SQLMORERESULTS()関数の戻り値に注意します。通常は取得したカーソルの数がセット(1)されますが、４回目の SQLMORERESULTS()関数を実行したときの戻り値は、これ以上結果セットは無いことを表す値(2)がセットされます。

・非同期・バッチモード

```
SQLSETPROP(hConn, “Asynchronous”, .T.)
SQLSETPROP(hConn, “BatchMode”, .T.)
SQLEXEC(hConn, “SELECT * FROM authors ;
```

```
                    SELECT * FROM titles ;
                    SELECT * FROM roysched ;
                    SELECT * FROM titleauthor", "ITEM")
```

この処理では、コマンドの実行の完了を待たずに FoxPro に制御が戻るために、カーソルは作成されません。カーソルを得るためには、SQLEXEC()関数を使います。

```
SQLEXEC(hConn)
```

SQLEXEC()関数を実行したときに、用意されたカーソルが作成されます。全てのカーソルを作成していない場合の戻り値はゼロ(0)です。全てのカーソルが作成された場合は １以上の数値（カーソルの数）がセットされます。

・非同期・非バッチモードの場合

```
SQLSETPROP(hConn, "Asynchronous", .T.)
SQLSETPROP(hConn, "BatchMode", .F.)
SQLEXEC(hConn, "SELECT * FROM authors ;
                    SELECT * FROM titles ;
                    SELECT * FROM roysched ;
                    SELECT * FROM titleauthor", "ITEM")
```

この場合も、コマンドの実行の完了を待たずに FoxPro に制御が戻るために、カーソルは作成されません。最初のカーソルを得るためには、SQLEXEC()関数を使います。

```
SQLEXEC(hConn)
```

次の SELECT 文の結果を得るには、SQLMORERESULTS()関数を使います。

```
SQLMORERESULTS(hConn)
```

このとき、カーソル名を指定すると、既に作成されている同名のカーソルを上書きします。SQLMORERESULTS()関数の戻り値に注意します。通常は取得したカーソルの数がセット(1)されますが、これ以上の結果セットは無い場合には ２がセットされます。

３．８　ストアドプロシジャーの呼び出し

FoxPro から SQL サーバー上のストアドプロシジャーを呼び出して見ましょう。コマンドウィンドウに以下を入力して SQL サーバーにストアドプロシジャーを作成します。

```
cProc = “CREATE PROCEDURE p_titles AS ;
                SELECT * FROM titles”
lnResult = SQLEXEC(hConn, cProc)
```

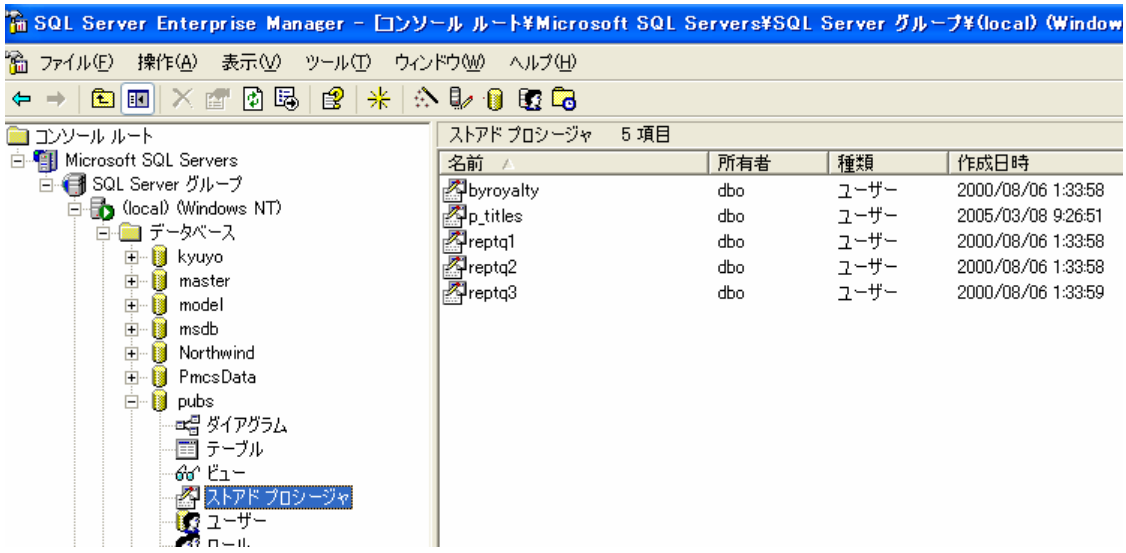
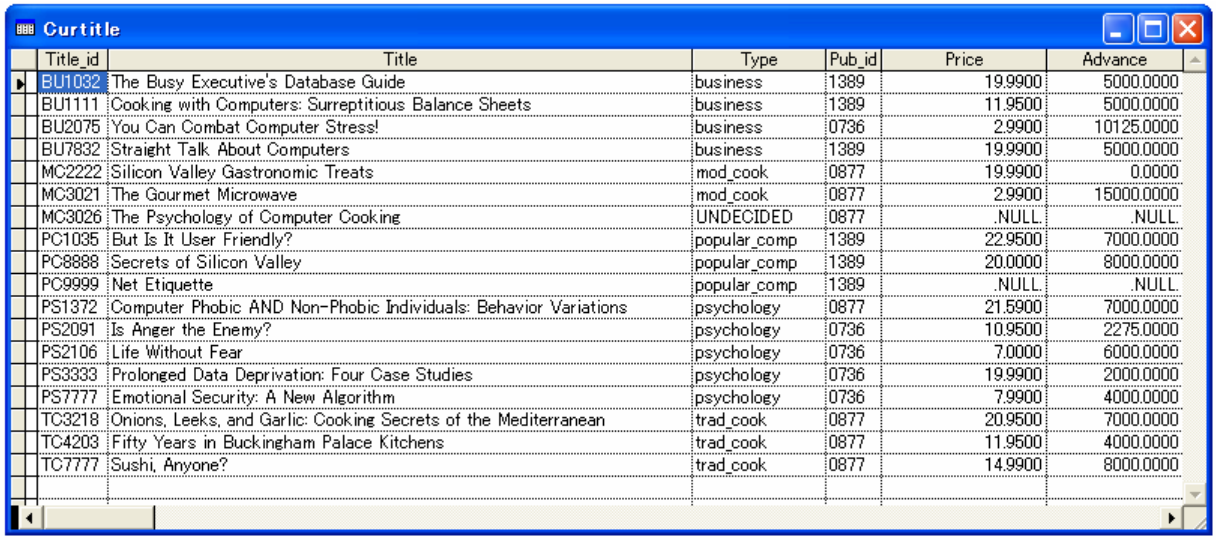


図２１．ストアドプロシジャー　p_titles が作成される

では、FoxPro から SQL サーバー上の p_titles ストアドプロシジャーを呼び出します。コマンドウィンドウに以下を入力します。

```
lnResult = SQLEXEC(hConn, “EXECUTE p_titles”, “curTitle”)
BROWSE
```



Title_id	Title	Type	Pub_id	Price	Advance
BU1032	The Busy Executive's Database Guide	business	1389	19.9900	5000.0000
BU1111	Cooking with Computers: Surreptitious Balance Sheets	business	1389	11.9500	5000.0000
BU2075	You Can Combat Computer Stress!	business	0736	2.9900	10125.0000
BU7832	Straight Talk About Computers	business	1389	19.9900	5000.0000
MC2222	Silicon Valley Gastronomic Treats	mod_cook	0877	19.9900	0.0000
MC3021	The Gourmet Microwave	mod_cook	0877	2.9900	15000.0000
MC3026	The Psychology of Computer Cooking	UNDECIDED	0877	.NULL.	.NULL.
PC1035	But Is It User Friendly?	popular_comp	1389	22.9500	7000.0000
PC8888	Secrets of Silicon Valley	popular_comp	1389	20.0000	8000.0000
PC9999	Net Etiquette	popular_comp	1389	.NULL.	.NULL.
PS1372	Computer Phobic AND Non-Phobic Individuals: Behavior Variations	psychology	0877	21.5900	7000.0000
PS2091	Is Anger the Enemy?	psychology	0736	10.9500	2275.0000
PS2106	Life Without Fear	psychology	0736	7.0000	6000.0000
PS3333	Prolonged Data Deprivation: Four Case Studies	psychology	0736	19.9900	2000.0000
PS7777	Emotional Security: A New Algorithm	psychology	0736	7.9900	4000.0000
TC3218	Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean	trad_cook	0877	20.9500	7000.0000
TC4203	Fifty Years in Buckingham Palace Kitchens	trad_cook	0877	11.9500	4000.0000
TC7777	Sushi, Anyone?	trad_cook	0877	14.9900	8000.0000

図２２．ストアドプロシジャーの実行結果

ストアドプロシジャーが複数の結果を返す場合もあります。先ほど作成した p_titles がて複数の結果を返すように変更します。コマンドウィンドウに以下を入力します。

```
cProc = “ALTER PROCEDURE p_titles AS ;
                SELECT * FROM titles ;
                SELECT * FROM authors”
lnResult = SQLEXEC(hConn, cProc)
lnResult = SQLEXEC(hConn, “EXECUTE p_titles”, “curTitle”)
```

Data Session ウィンドウと Browse コマンドを使用し結果を確認してください。「３．３SQLEXEC() 関数で取得するカーソルに名前を付ける」で説明したようにカーソル名は最初の結果に対して指定した名前が付き。それ以降は １、２、・・・とサフィックスが付きます。又、最後に返された結果が FoxPro ではカレントになります。

３．９　ストアドプロシジャーにパラメータを渡す

ストアドプロシジャーはパラメータを受け取ることも、パラメータに値をセットして FoxPro 側に戻すこともできます。

まずは、ストアドプロシジャーにパラメータを渡して見ましょう。p_titles が WHERE 句のパラメータを受け取るように変更します。

コマンドウィンドウに以下を入力します。

```
cProc = “ALTER PROCEDURE p_titles ;
                @pub_id char(4) AS ;
                SELECT * FROM titles WHERE pub_id = @pub_id”
lnResult = SQLEXEC(hConn, cProc)

lnResult = SQLEXEC(hConn, “EXECUTE p_titles '1389'“, “curTitles”)
```

Curtitles

Title_id	Title	Type	Pub_id	Price	Advance	Royalty	Ytd_sales
BU1032	The Busy Executive's Database Guide	business	1389	19.9900	5000.0000	10	4095
BU1111	Cooking with Computers: Surreptitious Balance Sheets	business	1389	11.9500	5000.0000	10	3876
BU7832	Straight Talk About Computers	business	1389	19.9900	5000.0000	10	4095
PC1035	But Is It User Friendly?	popular_comp	1389	22.9500	7000.0000	16	8780
PC8888	Secrets of Silicon Valley	popular_comp	1389	20.0000	8000.0000	10	4095
PC9999	Net Etiquette	popular_comp	1389	.NULL.	.NULL.	.NULL.	.NULL.

図２３．　Pub_id の 1389 のデータが選択されたストアドプロシジャーの結果

ストアドプロシジャーの Output パラメータを利用して、ストアドプロシジャーが返す値を取得して見ましょう。p_titles プロシジャーが指定された pub_id に該当するレコード件数を Output パラメータで返すように変更します。

```
cProc = “ALTER PROCEDURE p_titles ;
        @pub_id char(4), ;
        @title_count int OUTPUT AS ;
        SELECT @title_count = COUNT(*) ;
            FROM titles ;
            WHERE pub_id = @pub_id”
lnResult = SQLEXEC(hConn, cProc)
nTitle_Count = 0
lnResult = SQLEXEC(hConn, “EXECUTE p_titles '1389', ?@nTitle_Count”)
```

pub_id が ‘ 1389 ’ に該当するレコード件数は、nTitle_Count に設定されます。実行結果を見てみましょう。

```
? nTitle_Count
```

この場合は、カーソルが作成されないことに注意してください。ストアドプロシジャーの Output パラメータを使用する場合は、ストアドプロシジャーの呼び出しの前に受け取る変数を用意します。
又、SQLEXEC()関数では、用意した変数を ?@変数名として指定します。
以降では p_titles プロシジャーは使用しないので、以下のコマンドで SQL サーバー上から削除します。

```
cProc = “DOROP PROCEDURE p_titles”
lnResult = SQLEXEC(hConn, cProc)
```

３．１０　リモートビューをテンプレートとして使う

通常の処理で使用することは無いと思いますが、リモートビューをテンプレートとして使用できます。
以下の例は、リモートビューが使っている接続を SPT で使用する方法です。

```
OPEN DATABASE pubs
USE pubs!vtitles
hConn = CURSORGETPROP(“ConnectHandle”, “Vtitles”)
lnResult = SQLEXEC(hConn, “SELECT * FROM authors”)
BROWSE
```

又、リモートビューに定義してある SQL 文を利用することも可能です。

```
cSql = CURSORGETPROP(“SQL”, “VTitles”)
csql = cSql + “ WHERE pub_id = '1389'“
lnResult = SQLEXEC(hConn, cSql, “curTitles”)
```

CURSORGETPROP()関数で取得できる内容については、FoxPro のヘルプファイルを参照してください。

４．トランザクション

今までの説明は、FoxPro 側は「バッファリング無し」、SQL サーバーは「自動コミット トランザクション」モードで実行していましたので、レコードの更新が直ぐに SQL サーバーに反映されました。
実際のアプリケーションでは、マスターテーブルを更新し明細テーブルを更新し... と、一つの処理の中で複数のテーブルを更新するのが一般的です。一つの処理単位の中でエラーが発生すると、処理中に更新したデータを元に戻す必要があります。これを実現するにはトランザクションを利用します。トランザクションを利用するには同時実効制御（ロック）の知識が必要ですが、それは専門書や使用する SQL サーバーのマニュアルを参照してください。
FoxPro のローカルのデータベースでもトランザクションがサポートされていますが、SQL サーバーのそれとは違います。
FoxPro と SQL サーバーのトランザクション処理の大きな違いは、FoxPro はトランザクションのネストは５レベルまでですが、SQL サーバーにはネストの制限はありません。又、処理が失敗した時にデータを元に戻す ROLLBACK 処理は FoxPro ではそれぞれのトランザクションレベルで発行する必要がありますが、SQL サーバーはネストレベルに関係なく一度の ROLLBACK で全てのデータを元に戻せます。

４．１　トランザクションを使った更新処理

では、トランザクションを利用して authors テーブルを更新して見ましょう。

```
lnResult = SQLEXEC(hConn, "SELECT * FROM authors", "authors")
lnResult = CURSORSETPROP("Tables", "dbo.authors", "authors")
lnResult = CURSORSETPROP("KeyFieldList", "au_id", "authors")
lnResult = CURSORSETPROP("UpdatableFieldList", "au_id, au_lname, au_fname")
lnResult = CURSORSETPROP("UpdateNameList", "au_id dbo.authors.au_id", "authors")
lnResult  =  CURSORSETPROP("UpdateNameList",  "au_lname  dbo.authors.au_lname",
                                        "authors")
lnResult  =  CURSORSETPROP("UpdateNameList",  "au_fname  dbo.authors.au_fname",
                                        "authors")
lnResult = CURSORSETPROP("SendUpdates", .T., "authors")
```

バッファリングモードを楽観的（オプティミスティック）ロックに設定します。

```
lnResult = CURSORSETPROP(“Buffering”, 5, “curAuthors”)
BROWSE
```

現在、authors テーブルの”au_lname”と”au_fname”が更新可能になっています。
“au_lname”を適当に変更します。SQL サーバーに付属する Enterprise Manage で Authors テーブルを開いて見ます。
SQL サーバー側のデータは変更されていません。これは、FoxPro 側でバッファリングが有効になっているので FoxPro 側のローカルのカーソルが変更されただけです。
ローカル側のカーソルの変更を SQL サーバーに反映させるには、TABLEUPDATE()関数を使います。

```
lnResult = TABLEUPDATE(.T., .T.)
```

TABLEUPDATE()関数を実行すれば直ぐに SQL サーバーに結果が反映されます。では、トランザクションを利用して更新してみます。再度、適当に”au_lname”を変更します。

```
lnResult = SQLEXEC(hConn, “BEGIN TRANSACTION”)
lnResult = TABLEUPDATE(.T., .T.)
```

Eterprise Manager を使用して Authors テーブルを開いて見ます。この状態で、更新は反映されていません。更新を反映するには

```
lnResult = SQLEXEC(hConn, “IF @@TRANCOUNT > 0 COMMIT”)
```

“COMMIT”することで変更が SQL サーバーに反映されます。このように、”BEGIN TRANSACTION”から”COMMIT”までが一つのトランザクションの処理単位になります。もし、処理中にエラーが発生した場合には、”ROLLBACK”を発行します。

```
lnResult = SQLEXEC(hConn, “IF @@TRANCOUNT > 0 ROLLBACK”)
```

では、”ROLLBACK”を発行して更新処理を取り消して見ましょう。再度、適当に”au_lname”を変更します。

```
lnResult = SQLEXEC(hConn, “BEGIN TRANSACTION”)
lnResult = TABLEUPDATE(.T., .T.)
```

次に、更新を取り消し元のデータに戻すために”ROLLBACK”を発行します。

```
lnResult = SQLEXEC(hConn, “IF @@TRANCOUNT > 0 ROLLBACK”)
```

ここで、注意したいのは”ROLLBACK”を発行すると確かに SQL サーバーのデータは更新されませんが、ローカル側のカーソル（ブラウザに表示されているデータ）は更新前の内容に戻りません。ローカル側のカーソルを元に戻すためには、ローカル側のカーソルにもトランザクションを使用する必要があります。

```
* FoxProのトランザクションの開始と
* SQLサーバーのトランザクションの開始
BEGIN TRANSACTION
SQLEXEC(hConn, “BEGIN TRANSACTION” )
* MyCursor１の更新
lisOK = TABLEUPDATE ( .T., .F., “ myCursor1” )
IF lisOK
    * MyCursor２の更新
    lisOK = TABLEUPDATE ( .T., .F., “ myCursor2”)
ENDIF
* SQLサーバーとFoxProのトランザクションを終了する
IF lisOK
    SQLEXEC (hConn, “IF @@TRANCOUNT > 0 COMMIT” )
    END TRANSACTION
```

```
ELSE
    SQLEXEC( hConn, “IF @@TRANCOUNT > 0 ROLLBACK” )
    ROLLBACK
ENDIF
```

TABLEUPDATE()関数には、更新中にエラーが発生した場合にどのような振る舞いをするかを指定するパラメータの組み合わせがあります。又、更新を取り消す為の TABLEREVERT()関数があります。グリッドを使用したデータの更新アプリを作成する場合、CURSORSETPROP(“Buffering”, 5, “カーソル名”)関数は、SET MULTILOCKS ON と組み合わせて使用します。又、グリッド上ではユーザーがどのレコードを変更したかを知る手段がありませんが GETNEXTMODIFIED()関数により、ユーザーが変更したレコードを知ることができます。

これまでの説明は、MS SQL Server の Transact SQL のトランザクション処理に依存しています。SQL Server に依存しないで ODBC 接続ハンドルに対するコミット、ロールバックを行う方法があります。（注：SQL サーバーは方言を話しますから、ＳＱＬ文の修正が必要になる場合があります）

```
lnResult = SQLSETPROP(hConn, 'Transactions', 2)    && トランザクションをマニュアル
                                                      モードにする
lnResult = SQLEXEC(hConn, SQL 文)          &&データの更新、又は追加の SQL コマンドを実行
lnResult = SQLCOMMIT(hConn)                && 変更をコミットする

変更を取り消すには
lnResult = SQLROLLBACK(hConn)              && 変更を取り消す
```

MS SQL Server に対して

SQLSETPROP(hConn, 'Transactions', 2)は、更新・追加、削除の SQLEXEC()関数を実行する直前に　SET IMPLICIT_TRANSACTIONS ON をサーバーに送ります。（接続は暗黙のトランザクションモードに設定されます）

又、SQLCOMMITE()は、IF @@TRANCOUNT > 0 COMMIT TRAN を、SQLROLLBACK()は、IF @@TRANCOUNT > 0 ROLLBACK TRAN をサーバーに対して発行します。

接続するデータベースに合わせたコマンドに変換するのは ODBC ドライバーの役目ですのでデータベースに依存しないアプリ作成は、この方が良いでしょう。

４．２　コネクションのバインドについて

通常、トランザクションは一つの接続にしか使用できません。しかし複数のデータベースに接続して処理する必要がある場合には、複数作成した接続を一つに纏めることによって、トランザクションを複数のデーターベースにまたがって使用することができます。以下に、複数の接続を一つに纏める方法を示します。

```
lcToken = ""
hConn1 = SQLConnect("pubs", "sa", "")
hConn2 = SQLConnect("pubs", "sa", "")
hConn3 = SQLConnect("pubs", "sa", "")
lnResult = SQLExec(hConn1, "EXECUTE sp_getbindtoken ?@lcToken")
lnResult = SQLExec(hConn2, "EXECUTE sp_bindsession ?lcToken")
lnResult = SQLExec(hConn3, "EXECUTE sp_bindsession ?lcToken")

SQLDisconnect(hConn1)
SQLDisconnect(hConn2)
SQLDisconnect(hConn3)
```

４．３　ＳＱＬサーバーアクセスの関数一覧

解説の中では取り上げなかった関数もあります、又、関数のパラメータには取り上げなかったものもあります。FoxPro のヘルプファイルで確認してください。

VIEW	DBSETPROP()	
	DBGETPROP()	
カーソル	CURSORSETPROP()	
	CURSORGETPROP()	
接続・開放	SQLCONNECT()	SQLSTRINGCONNECT()
	SQLDISCONNECT()	
	SQLGETPROP()	SQLSETPROP()
メタデータアクセス	SQLTABLES()	
	SQLCOLUMNS()	
SPT	SQLEXEC()	SQLCOMMIT()
	SQLPREPARE()	SQLROLLBACK()
	SQLMORERESULT()	
	SQLCANCEL()	

５．グリッドを利用したデータ入力の考察

実際のアプリではグリッドを利用したデータ入力を行う場合があります。一例として、購入伝票のデータ入力を考えて見ましょう。伝票は多品一葉で明細行は５行、アプリの処理は、購入データを明細行単位に購入明細テーブルに登録し在庫テーブルを更新するものとします。

オペレータは、伝票№、購入日、各明細行を入力しデータ登録のために登録ボタン（或いはファンクションキー）を押します。登録ボタンを押した後で入力ミスに気づきデータを修正する場合はどのような処理になるでしょうか。オペレータは、再び、伝票№を入力し登録してあるデータを呼び出します。そして入力ミスをしたデータを修正し登録ボタンを押します。

明細行の更新は TABLEUPDATE()関数の使用で問題なくできそうです、在庫テーブルの更新はどうでしょうか？　購入数量の修正、購入品目の修正、それとも両方とも修正？ と、考慮しなければならない事があります。５行なら明細を配列に保存すれば良いと考えがちですが、５行でなく何百、数千の明細を扱う場合はどうでしょう。
オペレータがグリッド上の明細を修正した場合、カレントは修正後の値になります。しかし、在庫データを更新前の値に戻すためには、グリッド上の修正前と更新後の値を知る必要があります。
これを解決するには OLDVAL()関数を使います。
この関数を使う場合は、行又はテーブルバッファリングを有効にして、マルチロックを有効にしておく必要があります。

SET MULTILOCKS ON
カーソルを更新可能にする
CURSORSETPROP(“Buffering”, 5, “カーソル名”)

オペレータがデータを修正し登録ボタンを押したら、アプリ側の処理は以下のようになります。

１．n = 0
２．GOTO TOP コマンドで明細行の先頭にポインターを位置づけます。
３．n = GETNEXTMODIFIED(n) でオペレータが修正した最初の位置を得ます。
４．GOTO ｎ　でポインターをオペレータが修正したレコードに位置づけます。
５．OLDVAL()関数を使用して更新前の品目の値を得ます。
６．在庫テーブルの更新前の品目を読みます
７．在庫数から更新前の値（OLDVAL()）を減算し在庫データを更新します。
８．在庫テーブルの在庫数に更新後の値を加えます。
９．在庫テーブルの更新後の品目を読みます
１０．在庫数に更新後の値を加算し在庫データを更新します。
１１．GETNEXTMODIFIED()関数の戻り値が０（ゼロ）になるまで、３以降を繰り返します。
１２．TABLEUPDATE()コマンドで購入明細テーブルを更新します。

新規レコードの項目を OLDVAL()で取得すると .NULL.になります。
更新途中でエラーになった時の対処のために、トランザクション処理をお忘れなく。

本格的に理解する為に

実際にアプリケーションを作成する場合には、デッドロック等の考慮も必要です。又、FoxPro 側のコマンドや関数だけでなく使用する SQL サーバーについて幅広い知識が必要になります。個別には解説できないので、Microsoft SQL Server 2000 を使用する場合の参考資料を示します。

各 URL のアドレスは 2005 年 3 月 9 日現在のものです。

・Client-Server Applications with Visual FoxPro 6.0 and SQL Server 7.0　Hentzenwerke Publishing

他の SQL サーバーを使う場合でも、必読の書です。

・SQL Server 2000 自習書シリーズ

http://www.microsoft.com/japan/SQL/techinfo/selfstudy/Self_doc.asp

・Microsoft Visual FoxPro ヘルプファイル（各バージョン）

・SQL サーバーに付属のヘルプファイル　Books Online

・大西彰氏のブログ　「楽観的ロックでいいじゃん！」

http://blogs.sqlpassj.org/akiraonishi/articles/5026.aspx

FoxPro のカーソルは他言語のアプリに渡すことが出来ません。複数の言語を組み合わせてｎ階層のアプリをする場合には ADO を使います。マイクロソフトの ”ADO Jumpstart for Microsoft Visual FoxPro Developers ” は、ADO 入門の良い参考資料ですので一読をお勧めします。

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnfoxgen/html/adojump.asp

SQL Server 2000 120 日間限定評価版 Release A は以下の URL からダウンロードできます。

http://www.microsoft.com/japan/sql/evaluation/trial/2000/

SQL Server 2000 120 日間限定評価版 Release A インストール手順

SQL Server 2000 120 日間限定評価版 Release A をローカルの PC にインストールする手順を示します。

1　上記のサイトから、評価版をダウンロードしてできた、”JPN_SQLEVAL.EXE”をダブルクリックします。

図１．JPN_SQLEVAL.EXE の解凍

2　インストール先のフォルダーに作成された”autorun.exe”をダブルクリックします。

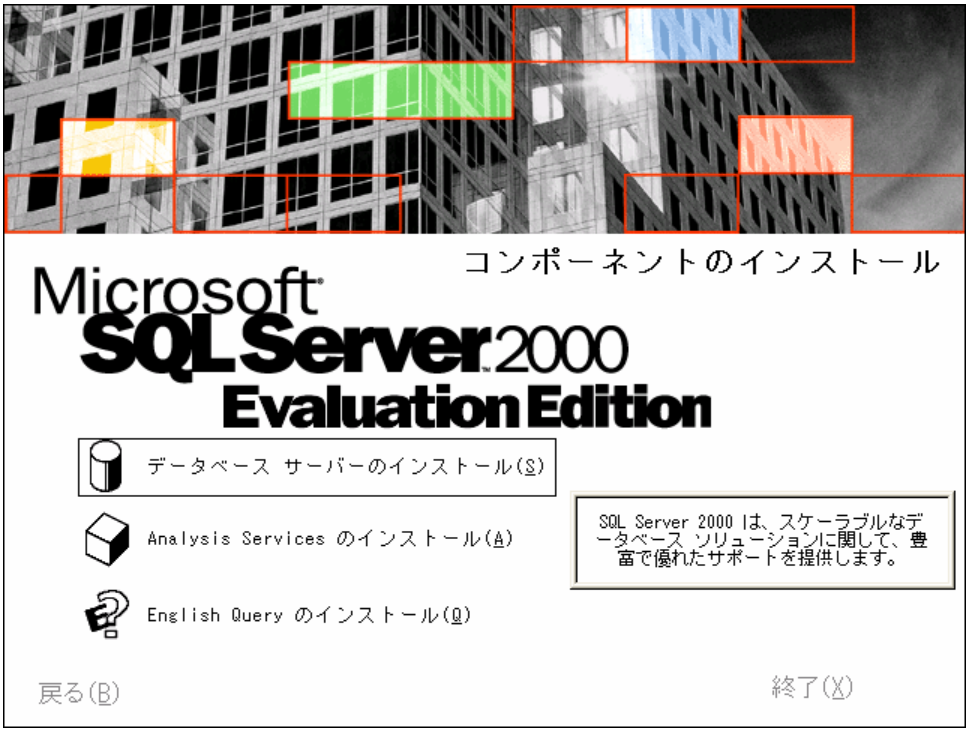


図3　インストールするコンポーネントの選択

「データベース サーバーおインストール(S)」をクリックします。

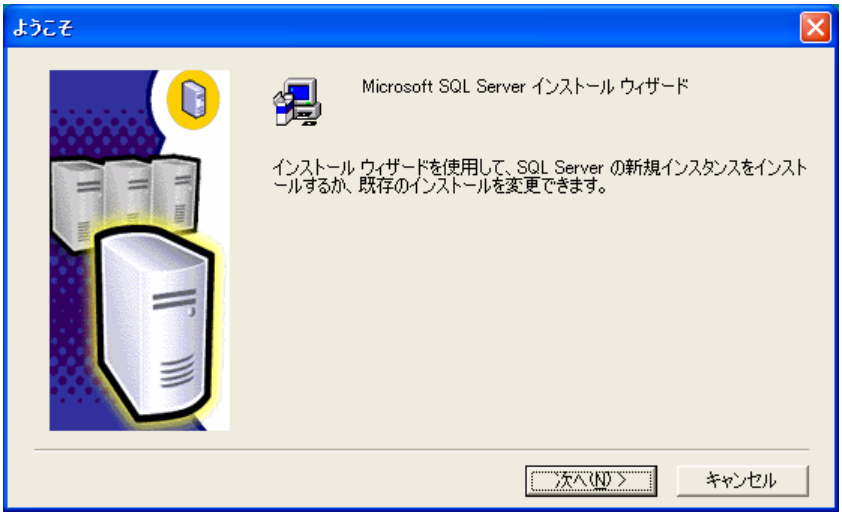


図４　インストールウィザード

「次へ(N)」ボタンをクリックします。

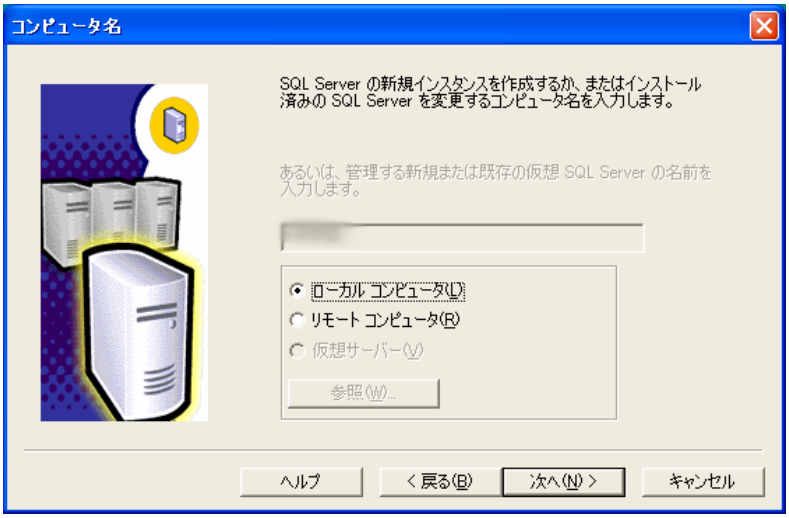


図５．インストール先の指定

「ローカルコンピュータ(L)」を選び「次へ(N)」ボタンをクリックします。

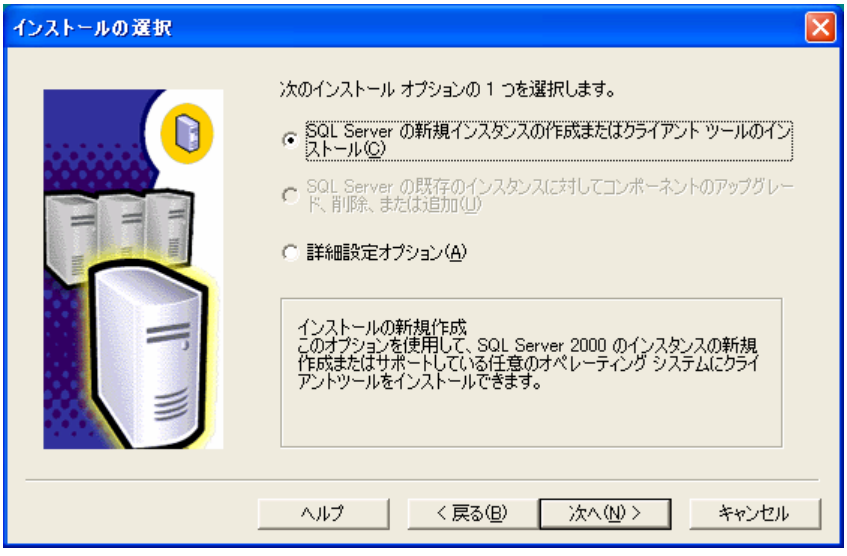


図５．インストールオプションの指定

デフォルトのまま、「次へ(N)」ボタンをクリックします。

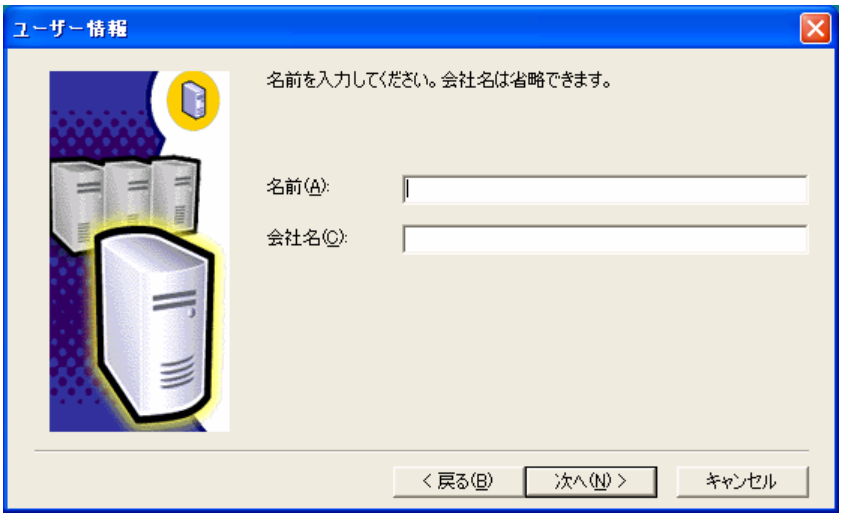


図６．ユーザー情報の入力

デフォルトのまま「次へ(N)」ボタンをクリックします。

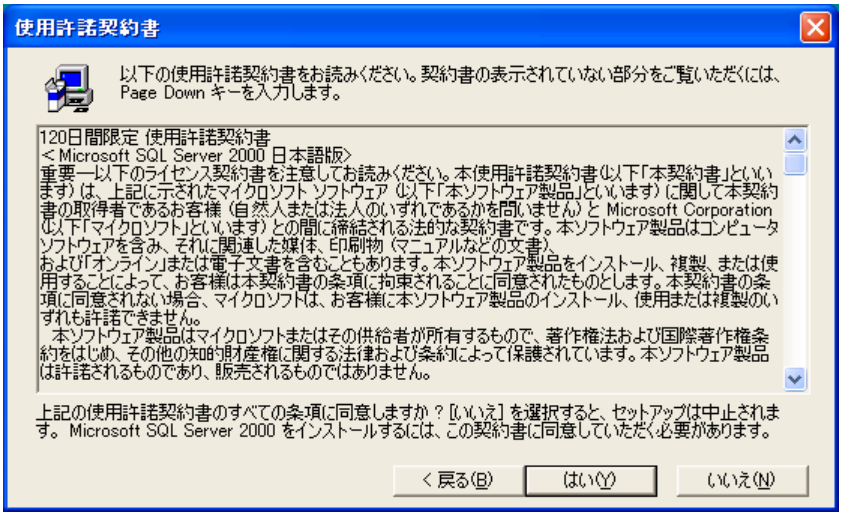


図７．使用許諾契約書

一通り内容を読んで、「次へ(N)」ボタンをクリックします。

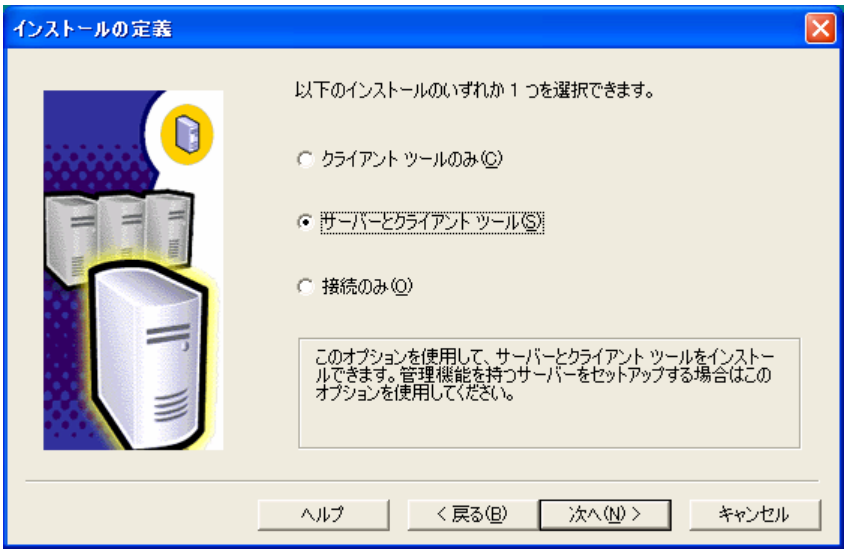


図８．インストールの定義

「サーバーとクライアントツール(S)」を選択して、「次へ(N)」ボタンをクリックします。

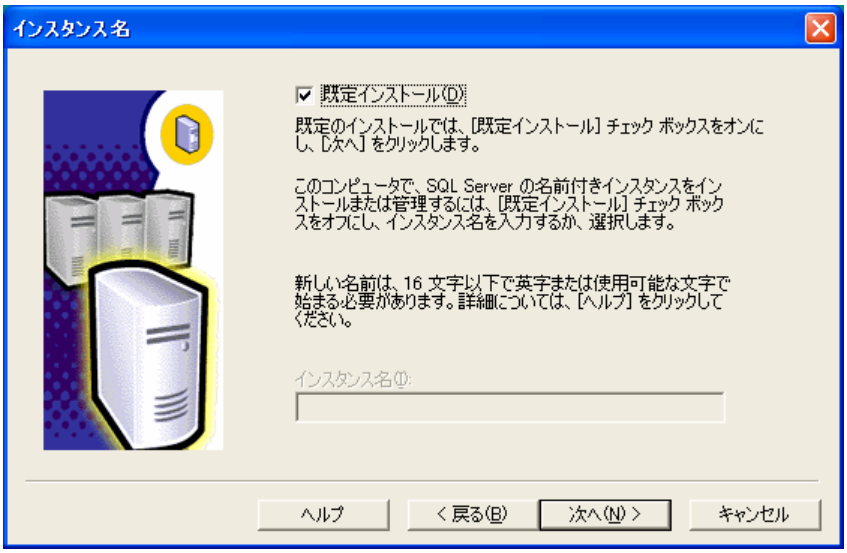


図９．インスタンス名の指定

「既定インストール」（デフォルト）にチェックして、「次へ(N)」ボタンをクリックします。

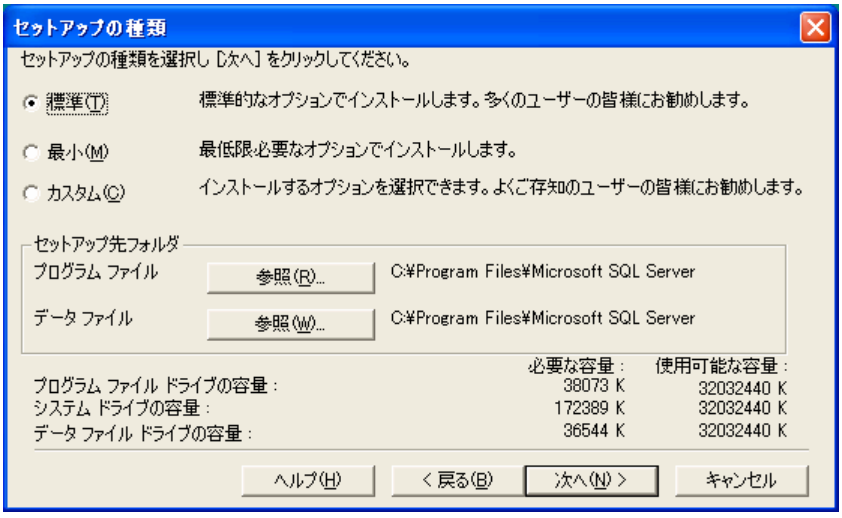


図１０．セットアップの周囲

「標準(T)」（デフォルト）を選んで、「次へ(N)」ボタンをクリックします。

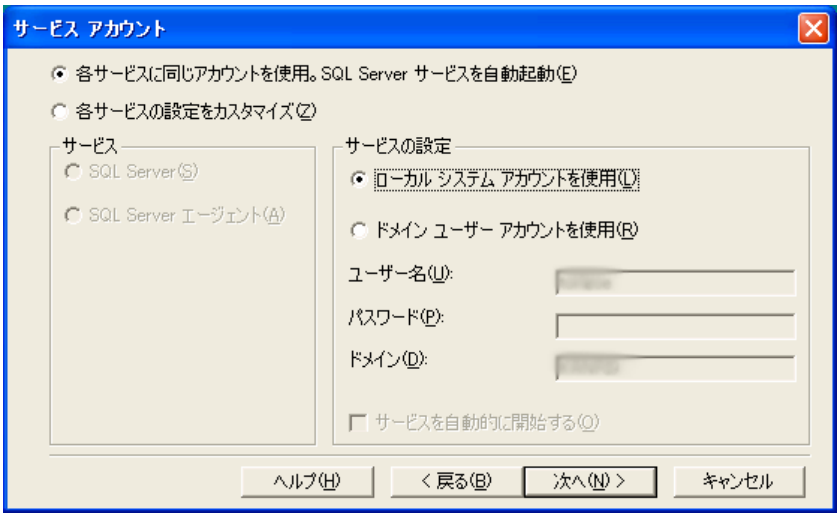


図１１．サービスアカウントの指定

デフォルトは、「ドメインユーザーアカウントを使用(R)」ですが、テスト使用なので「ローカル システム アカウントを使用(L)」を選び、「次へ(N)」ボタンをクリックします。

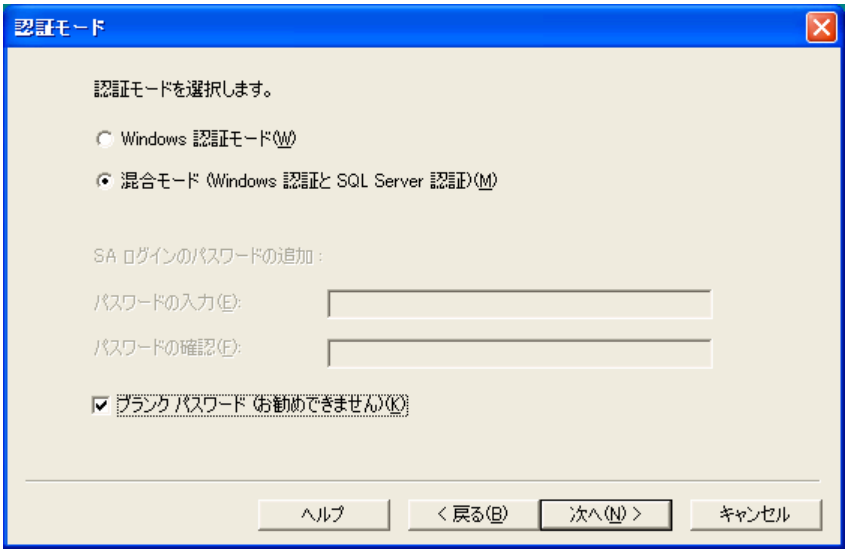


図１２．認証モードの指定

デフォルトは「Windows 認証モード(W)」ですが、テスト使用なので「混合モード(Windows 認証と SQL Server 認証)(M)にし、ブランクパスワード(お勧めできません）(K)にチェックし、「次へ(N)」ボタンをクリックします。実際の運用用のサーバーは、セキュリティ上ブランクパスワードの使用は好ましい設定ではありません。

図１３．ファイルコピーの開始

インストールに必要なパラメータの指定が終わったので、「次へ(N)」ボタンをクリックします。この間、インストール進行状況を示すプログレスバーが表示されます。

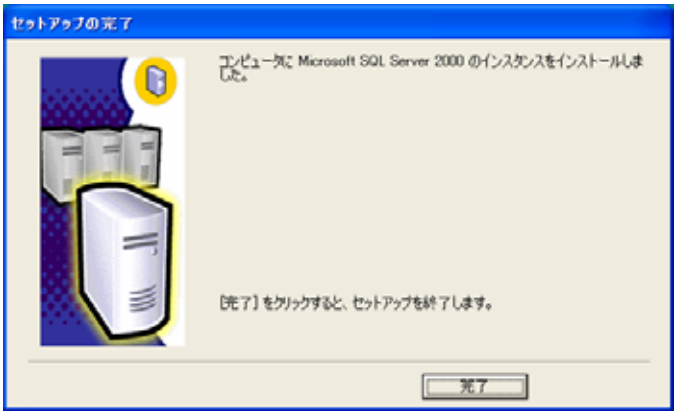


図１４．セットアップの完了

「完了」ボタンをクリックして画面を閉じます。

3　SQL サーバーを起動します。

「スタート」－「全てのプログラム(P)」－「Microsoft SQL Server」から、「サービス マネージャー」を起動します。

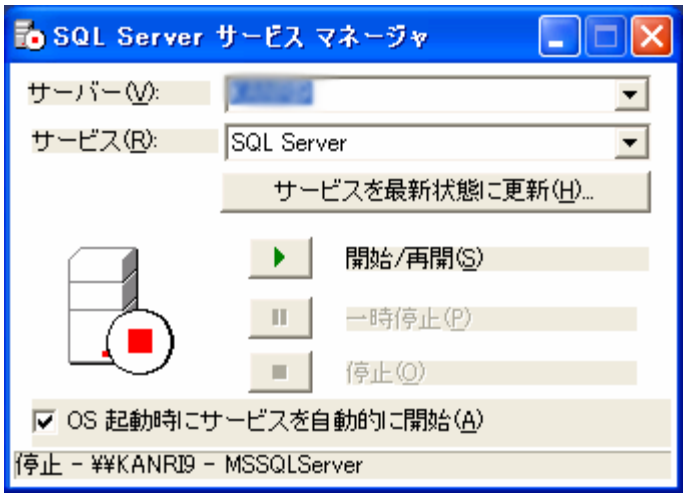


図１５．SQL Server サービス マネージャー

「開始/再開(S)」ボタンをクリックして SQL サーバーを起動します。

4．SQL サーバーの確認

「スタート」－「全てのプログラム(P)」－「Microsoft SQL Server」から、「Enterprise Manager」を起動します。

左ペインのルートを展開していき、以下の画面のように表示されれば SQL サーバーのインストールは成功です。

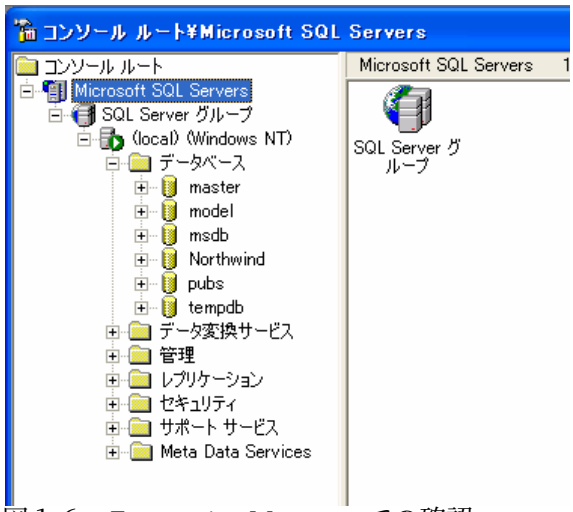


図１６．Enterprise Manager での確認

以上で、SQL Server 2000 120 日間限定評価版 Release A がインストールできたので、FoxPro からの接続を試してください。

FoxPro からデータベースを登録するには

SQL サーバーに MS SQL Server 2000 を使用する場合は、Enterprise Manager を使ってビジュアルにデータベースを作成可能ですが、MSDE をデータベースエンジンとして使用する場合は CUI の ISQL を使ってデータベースを作成する必要があります。ISQL の利用方法に詳しく CUI が苦にならない場合は良いのですが、データーベースアプリ開発用に FoxPro を選択しているわけですから、データベースも FoxPro を使って登録する方法について説明します。

サーバーに登録されるデータベースは master データベースに登録されます。従って、新規データベースを作成するには、先ず master データベースに接続します。

```
hConn = SQLSTRINGCONNECT(“driver={SQL Server};server=(サーバー名);database=Master;
                                    uid=ユーザ ID;pwd=パスワード”)
lnResult = SQLEXEC(hConn, “CREATE DATABASE MYDATABASE”)
SQLDISSCONNECT(0)
```

注：MS SQL Server Books Online の CREATE DATABASE を参照のこと

次に、作成した MyDatabase に接続します。

```
hConn = SQLSTRINGCONNECT(“driver=SQL Server;
                                    server=サーバー名;database=MyDatabase;
                                    uid=ユーザ ID;pwd=パスワード”)
```

この後は、

```
cSql = "CREATE TABLE MyTable(.....)"
lnResult = SQLEXEC(hConn, cSql)
```

のように、CREATE TABLE、CREATE VIEW、CREATE PROCEDURE 等の DML 言語を FoxPro で作成したアプリから SQL Server に発行してやるだけです。